A Framework for Design Assurance in Developing Embedded Systems

by

Kim Randal Fowler

B.S., University of Missouri – Rolla (now Missouri Institute of Science and Technology), 1978
M.S., Johns Hopkins University, 1982

AN ABSTRACT OF A DISSERTATION

submitted in partial fulfillment of the requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Electrical and Computer Engineering
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2018

Abstract

Embedded systems control nearly every device we encounter. Examples abound: appliances, scientific instruments, building environmental controls, avionics, communications, smart phones, and transportation subsystems.

These embedded systems can fail in various ways: performance, safety, and meeting market needs. Design errors often cause failures in performance or safety. Market failures, particularly delayed schedule release or running over budget, arise from poor processes. Rigorous methods can significantly reduce the probability of failure.

Industry has produced and widely published “best practices” that promote rigorous design and development of embedded systems. Unfortunately, 20 to 35% of development teams do not use them, which leads to operational failures or missed schedules and budgets.

This dissertation increases the potential for success in designing and developing embedded systems through the following:

1. It identifies, through literature review, the reasons and factors that cause teams to avoid best practices, which in turn contribute to development failures.
2. It provides a framework, as a psychologically unbiased mediator, to help teams institute best practices. The framework is both straightforward to implement and use and simple to learn.
3. It examines the feasibility of both crowdsourcing and the Delphi method to aid, through anonymous comments on proposed projects, unbiased mediation and estimation within the framework. In two separate case studies, both approaches resulted in underestimation of both required time and required effort. The wide variance in the surveys’ results from crowdsourcing indicated that approach to not be particularly useful. On the other hand, convergence of estimates and forecasts in both projects resulted when employing the Delphi method. Both approaches required six or more weeks to obtain final results.
4. It develops a recommendation model, as a plug-in module to the framework, for the build-versus-buy decision in design of subsystems. It takes a description of a project, compares designing a custom unit with integrating a commercial unit into the final product, and generates a recommendation for the build-versus-buy decision.

A study of 18 separate case studies examines the sensitivity of 14 parameters in making the build-versus-buy decision when developing embedded systems. Findings are as follows: team expertise and available resources are most important; partitioning tasks and reducing interdependence are next in importance; the quality and support of commercial units are less important; and finally, premiums and product lifecycles have the least effect on the cost of development.

A recommendation model incorporates the results of the sensitivity study and successfully runs on 16 separate case studies. It shows the feasibility and features of a tool that can recommend a build-or-buy decision.

5. It develops a first-order estimation model as another plug-in module to the framework. It aids in planning the development of embedded systems. It takes a description of a project and estimates required time, required effort, and challenges associated with the project. It is simple to implement and easy to use; it can be a spreadsheet, a Matlab model or a webpage; each provides an output like the model for the build-versus-buy decision.

A Framework for Design Assurance in Developing Embedded Systems

by

Kim Randal Fowler

B.S., University of Missouri – Rolla (now Missouri Institute of Science and Technology), 1978
M.S., Johns Hopkins University, 1982

A DISSERTATION

submitted in partial fulfillment of the requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Electrical and Computer Engineering
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2018

Approved by:

Co-Major Professor
Dr. Stephen A. Dyer

Approved by:

Co-Major Professor
Dr. Steven Warren

Copyright

Abstract

Embedded systems control nearly every device we encounter. Examples abound: appliances, scientific instruments, building environmental controls, avionics, communications, smart phones, and transportation subsystems.

These embedded systems can fail in various ways: performance, safety, and meeting market needs. Design errors often cause failures in performance or safety. Market failures, particularly delayed schedule release or running over budget, arise from poor processes. Rigorous methods can significantly reduce the probability of failure.

Industry has produced and widely published "best practices" that promote rigorous design and development of embedded systems. Unfortunately, 20 to 35% of development teams do not use them, which leads to operational failures or missed schedules and budgets.

This dissertation increases the potential for success in designing and developing embedded systems through the following:

1. It identifies, through literature review, the reasons and factors that cause teams to avoid best practices, which in turn contribute to development failures.
2. It provides a framework, as a psychologically unbiased mediator, to help teams institute best practices. The framework is both straightforward to implement and use and simple to learn.
3. It examines the feasibility of both crowdsourcing and the Delphi method to aid, through anonymous comments on proposed projects, unbiased mediation and estimation within the framework. In two separate case studies, both approaches resulted in underestimation of both required time and required effort. The wide variance in the surveys' results from crowdsourcing indicated that approach to not be particularly useful. On the other hand, convergence of estimates and forecasts in both projects resulted when employing the Delphi method. Both approaches required six or more weeks to obtain final results.
4. It develops a recommendation model, as a plug-in module to the framework, for the build-versus-buy decision in design of subsystems. It takes a description of a project, compares designing a custom unit with integrating a commercial unit into the final product, and generates a recommendation for the build-versus-buy decision.

A study of 18 separate case studies examines the sensitivity of 14 parameters in making the build-versus-buy decision when developing embedded systems. Findings are as follows: team expertise and available resources are most important; partitioning tasks and reducing interdependence are next in importance; the quality and support of commercial units are less important; and finally, premiums and product lifecycles have the least effect on the cost of development.

A recommendation model incorporates the results of the sensitivity study and successfully runs on 16 separate case studies. It shows the feasibility and features of a tool that can recommend a build-or-buy decision.

5. It develops a first-order estimation model as another plug-in module to the framework. It aids in planning the development of embedded systems. It takes a description of a project and estimates required time, required effort, and challenges associated with the project. It is simple to implement and easy to use; it can be a spreadsheet, a Matlab model or a webpage; each provides an output like the model for the build-versus-buy decision.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I thank my major professor, Dr. Stephen A. Dyer, for his invitation to graduate school and his support during my studies and research. I am grateful that you were willing to take me on and help me achieve a life-long dream.

I thank the remainder of my Supervisory Committee for considering my research and work: Dr. Sanjoy Das, Dr. John Hatcliff, and Dr. Steven Warren. I thank Dr. Byron Jones for being the outside chair for my oral defense.

Steve Lammlein, my friend for 50 years, provided invaluable advice concerning surveys and motivation to participate. Thank you, my friend.

I could not have run the Delphi study without the five willing participants as subject matter experts: Geoff Patch, Dwight Clayton, Max Cortner, Dr. Mike Gard, and Tim Cooper. Thank you. I also appreciate and thank the anonymous survey takers for the crowdsourcing study.

I am grateful for Dr. Sanjoy Das showing up at just the right moment in June 2018 to show me and help me understand parameter sensitivity in determining feasibility.

I thank the professors who taught courses that I thoroughly enjoyed and from whom I learned a great deal: Dr. Gary Brase, Dr. Sanjoy Das, Dr. David Steward, Dr. John Hatcliff, and Dr. Balasubramaniam Natarajan. Finally, I thank the Graduate School at Kansas State University for making my graduate experience a good and memorable one.

Dedication

This dissertation is dedicated to the memory of my wife, Oonagh E. G. Fowler (b. Jan. 20, 1959, d. Jan. 18, 2016). She encouraged me to return to graduate school for my doctoral degree and was part of that fateful telephone call on October 31, 2011, when Steve Dyer invited me to Kansas State for graduate school. She made the long moves, first from Great Britain to marry me in 1989, and then the 1200 miles from Baltimore, Maryland, to Kansas in August 2012. She supported me and my studies up until her passing.

I love you and miss you, Oonagh.

Chapter 1 - Overview, Motivation and Contributions

Overview

This dissertation and its research focus on methods for good design and development of embedded systems. The focus is on the process of disciplined design and development, not on specific techniques or procedures, such as a design tool, requirement, or component.

An embedded system depends on computing components as critical elements in its function. Ganssle and Barr define an embedded system as "*a combination of computer hardware and software, and perhaps additional mechanical or other parts, designed to perform a dedicated function."* [1] Embedded systems are ubiquitous; they control most devices and equipment that we encounter. Examples include the small microcontrollers in appliances, such as coffee makers, engine-control modules in automobiles, medical equipment, and avionics in aircraft.

Motivation

Embedded systems can fail and induce financial loss, physical damage, injury, or death. Outside of failures from usual fatigue and wear, many failures could be avoided by following known, good practices during the design and development of embedded products. Furthermore, many products miss delivery deadlines because design and development teams do not account for contingencies that are quite typical in most products.

A colloquial saying, "There is not enough time to do it right, but always time to do it over," indicates that estimating product delivery and building quality into a product are linked. The motivation for this work is to help development teams establish and meet reasonable schedules and budgets while designing quality products that avoid most failures.

Outline of the Dissertation

The approach in this work herein is to identify the problems, review the literature, and propose a simple and usable framework for addressing the problem of design assurance in developing embedded systems. A framework, in the context of this dissertation, is an approach and structure to provide baseline estimates of time, effort, cost, and potential challenges.

The work in this dissertation divides into six chapters in addition to this introductory chapter and includes appendices with specific details that support various aspects of the work. Chapter 2 begins by identifying the problems to be addressed, and then it provides a literature review to support the need for, and characteristics of, a framework. Chapter 3 introduces the proposed framework and outlines its operation and utility; the framework presents a flexible format to persons using it for baseline estimations; as a structure, the framework accepts plug-in modules, each of which performs specific functions. Chapters 4 through 6 elaborate on the three plug-in modules established as part of this research effort. Chapter 4 presents the first plug-in module, which is an estimator for effort, time, cost, and potential challenges. Chapter 5 presents the second plug-in module, which is a recommender for the build-versus-buy decision. Chapter 6 presents the third plug-in module, which is a human-centered advisor. Chapter 7 presents the conclusions from this work and outlines potential future work.

The framework will help developers of embedded systems meet the three major constraints in most projects: features, schedule and budget. The framework is a structured approached to estimating time, effort and potential challenges. It uses plug-in modules to perform specific operations during the planning phase, followed by a checklist for users to consult during development of the embedded system. Figure 1.1 outlines the proposed framework and its plug-in modules, which suggest effort, schedules, budgets, and potential challenges associated with the development of embedded systems.

Contributions of this Work

This work is novel because the proposed framework generates baseline constraints for the time, effort and cost to develop an embedded system with less information than needed by other methods. It can also supply rank-ordered lists of potential challenges and a checklist of necessary activities before beginning the project. It is more robust in the face of missing information than other techniques or methods, which would not function if some information is missing.

The information generally expected in this framework is of three basic types:

1. Parameters of time, full-time-equivalents (FTEs) for available staffing, and the type of industry;
2. A sparse description of the potential project (the description is expected to be less than 300 words); and

3. Min-max ranges of simple parameters (lines of code [LOC], number of circuit connections, number of mechanical connections, and number of enclosures).

With any mixture of these three types of information, the framework can generate a baseline. If only the sparse description is given, for example, then the human-advisor module can still generate an estimate of time and effort along with a potential list of technical challenges. If only the parameters in lines 1 and 3 above are given, then the estimator will give a boundary constraint that has a measure of confidence.

Other methods give means, confidence intervals, and trend lines, but they also need more parameters. They also need defined measures of complexity. For example, COCOMO, the estimator for software effort, requires 15 separate parameters, including complexity, that users of COCOMO must supply. These methods do not function if some of the required values are not supplied.

The framework described herein has at least three plug-in modules that can work separately or together to generate output. Two of these plug-in modules, the estimator and the recommender, have a sanity checker that uses industry and published values for ensuring the boundary constraints. In addition, the human-advisor module can be used to provide a sanity check for the values of the other two plug-in modules.

Finally, the proposed framework can incorporate additional plug-in modules, which can supply more-accurate estimates if more information is supplied. Some commercial software packages might fit this application.

One note about this proposed framework is this: the numerical values representing the design/development times that accompany the number of circuit connections, the number of mechanical connections, and the number of enclosures are basic placeholders at this writing. These parameters need time-motion studies and research to update their values with confidence intervals.

The purpose of this work is to improve success in designing and developing embedded systems. The principal contributions of this work are five-fold:

1. Identification of the reasons and factors that cause teams to avoid good practices.
2. Development of a framework, which is a structured approach, to help teams institute good practices.

3. Development of an outline plan and checklist to undergird a design contract between the development team and management.
4. A study of the feasibility of plug-in modules to the framework that includes the following:
   - determining parameters influential to the build-versus-buy decision when designing subsystems, and
   - comparing crowdsourcing and the Delphi method.
5. Implementation and test of plug-in modules to the framework that are first-order estimators to recommend baseline values and which include the following:
   - a module that estimates time, effort, and potential challenges for the project independently from either the crowdsource or Delphi methods,
   - a module that recommends a build-versus-buy decision for subsystems, and
   - a module that prepares a Delphi-method survey, if deemed useful by the user of the framework, to estimate time, effort, and potential challenges for the project.

The framework and its plug-in modules have characteristics of good processes: they are easy to use, they encourage (nudge) good practices, they do not require additional learning by users, and they avoid biased thinking that skews decisions. Wherever possible, the proposed framework incorporates industry-sourced data.

Focus

The work of this dissertation focuses on the process for designing and developing embedded systems, favoring small, clean-sheet developments for embedded systems. It is restricted to several important issues necessarily addressed to demonstrate feasibility and utility. Ignored are specifics of requirements, performance, coding techniques, designs, manufacturing, tooling, and safety in design.

It must be emphasized that whereas the principles, guidelines and metrics espoused in this dissertation suffice for embedded systems, they may not suffice for larger, more complex systems, although they form a necessary subset of good processes for larger systems. The work herein can act as a basis for the development of similar tools for other application areas, such as large systems, systems of systems, distributed and decentralized teams, information technology

(IT), marketing, business, and accounting. The development of each such tool for large systems will itself be a significant research effort.

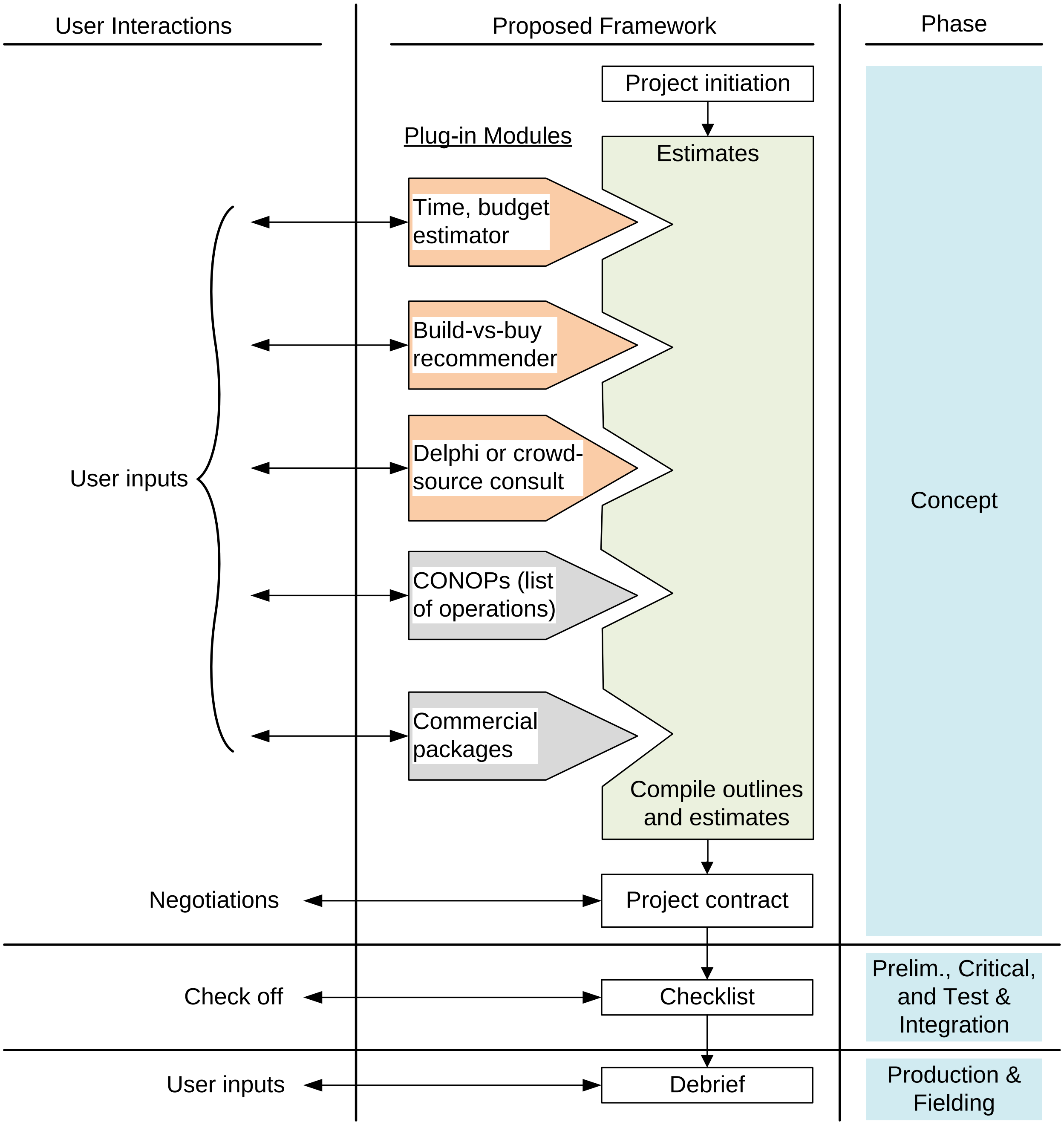


Figure 1.1 Outline of the proposed framework with plug-in modules to suggest desired features, schedules and budgets of embedded systems.

Chapter 2 - Problem Statement and Literature Review

Introduction and Definitions

Engineering projects fail. Often, they fail for preventable reasons. A quick scan of the literature readily finds many failures. A few examples follow.

These failures motivate this work to propose a better way to develop embedded systems. First, I will show that significant numbers of development teams do not follow known, good practices and processes. Furthermore, not following known, good practices guarantees latent errors and eventual failure. Next, I will show that most development teams overrun either schedule or budget or both. Then, I will argue that not following good practices guarantees latent errors, which cost time and money, thus overrunning schedule and budget.

Literature review will point to the need to use of known, good practices. It will consider the reasons why developers do not always use known, good practices. Finally, the literature review will identify the availability of tools and techniques that might help persuade or "nudge" the use of known, good practices.

The process of designing and developing is fundamentally a human problem. The focus of this work is on the process of designing and developing embedded systems. The focus does not include a specific technique, method, requirement, or design.

Ganssle and Barr define an embedded system as "*a combination of computer hardware and software, and perhaps additional mechanical or other parts, designed to perform a dedicated function."* [1] In other words, an embedded system depends on a computing component, or components, as a critical element, or elements, in its function.

Recalls

Federal recalls required of manufacturers is a rich source of material. ". . . recalled 226 [electric vehicles] to reprogram the logic in the motor electronic control unit (ECU), which can mistakenly detect a failure of an electrical current sensor at speeds above 50 mph. It can cause the sudden loss of power and unexpected deceleration [2]." ". . . recalling certain ALL-IN-ONE Power Converter . . . The affected power converters, used in recreational vehicles to convert A/C power, have internal components that may short circuit. . . A short circuit may lead to a vehicle fire [3]."

Recalls of automobiles, dating back to the 1970s, may be found at the website for the National Highway Traffic Safety Administration. Beyond automobiles, recalls of appliances may be found at the Consumer Safety Product Commission [4]. The sources of the failures that lead to recalls often are incorrect designs, selecting bad components, or faulty manufacturing. Regardless, human error is at the root of all these failures.

Recent Failures

Within the past two years, Galaxy Note 7 smartphones had catastrophic failures with batteries. “Samsung, the South Korean conglomerate, blamed battery manufacturing problems and design flaws for the embarrassing and costly failure of its Galaxy Note 7 smartphone . . . Samsung had pushed itself to the limit, company officials said. It rushed the Note 7 to the market before Apple rolled out its iPhone 7. The accelerated production was also driven by fear; Huawei, Xiaomi and other Chinese cellphone makers were fast catching up. By packing the Note 7 with new features, like waterproofing and iris-scanning for added security, Samsung also wanted to prove that it was more than a fast follower [5].”

In a similar timeframe, a Tesla Model S sedan had a fatal collision during autonomous driving. “It is the first known fatal accident involving a vehicle being driven by itself by means of sophisticated computer software, sensors, cameras and radar. . . Neither autopilot nor the driver noticed the white side of the tractor-trailer against a brightly lit sky, so the brake was not applied [6].” Clearly, this accident resulted from an oversight in design, development, or training of the Tesla system.

Historical Failures

While all the details of recent failures are not yet available, two case studies may be instructive and illustrative for some of the causes of failures. One case study is for the Ariane 5 launch vehicle. The other case study is for the Therac 25, a medical radiological device. As in the recalls mentioned previously, human error during design and development is at the center of these failures.

Case Study 1: Ariane 5

During the 1970s, 1980s, and 1990s, the French developed a family of launch vehicles for satellites. One generation of boosters, called the Ariane 4, successfully orbited many satellites. The follow-on booster was the new, larger Ariane 5 [7][8].

The first launch of an Ariane 5, on 4 June 1996, self-destructed within 40 seconds and destroyed four satellites. A shutdown of the control guidance allowed aerodynamic loads to separate the booster stages, which initiated the self-destruct sequence. Redundant computers in its inertial reference system (IRS) comprised the control guidance. Both computers in the IRS were identical, ran identical software, and would shut down upon numerical overflow [8].

A numerical overflow in an unnecessary software routine had shutdown both computers in the IRS. The shutdown occurred because three variables were left unprotected from calculation overflow in the floating point-to-integer conversions [8].

The IRS had been developed for the Ariane 4; it was reused in the Ariane 5 because of its successful operation with the Ariane 4 booster. The IRS was developed for the Ariane 4 and fit the smaller booster's flight dynamics and performance. The developers assumed they had a large enough operational margin to avoid overflow. Unfortunately, the Ariane 5 had larger than expected drift in the horizontal velocity, which used up the operational margin and caused a numerical overflow. Since both computers ran the same software, when one experienced the numerical overflow and shutdown, the other – in lockstep – followed suit [8].

Simulation, ground tests, and extensive technical reviews did not detect the potential failure in the Ariane 5. No one performed any overall system tests to test all systems within the booster at the same time. A report from the investigation stated, "Limitations of software were not fully analyzed . . . [8]"

Case Study 2: Therac 25

The Therac-25 was a medical linear accelerator developed during the 1970s and early 1980s to treat tumors with high-energy beams. Between 1985 and 1987, Therac-25 accelerators accidentally and massively overdosed six patients with radiation and caused both serious injuries and death [9]. In each of case, the Therac-25 delivered up to 250 Gy of radiation yet registered "no dose given." (NOTE: 5 Gy of whole-body radiation can kill.)

Systems analysis showed that dynamic issues in the custom operating system and real-time executive with a preemptive scheduler led to the catastrophic overdoses. While each task ran well – and without failure – in isolation, asynchronous operation and communicating in the multitasking environment set up conditions for system failure. First, a race condition caused by an operator's quick editing at the keyboard passed incorrect data to various tasks. Then a cryptic error message appeared with no follow-up explanation in the user manual, so operators did not know what remedial action to take. The Therac-25 depended upon software safety mechanisms; it had no mechanical interlocks. For instance, it had a 1-byte status register that rolled over every 256 passes; a zero indicated that it was okay to proceed when that was not necessarily the case. Finally, the system did not have an independent physical limit to prevent overdoses of radiation [9].

Follow-up investigations revealed that a single developer wrote the code without any checks, audits, or inspections. Very little of the software was documented. The development for the Therac-25 had no general plan and, consequently, there were no analyses for failure modes or hazards. There was no encompassing system test that exercised the entire system for dynamic interactions [9].

Commentary on Failure

Failure is a part of engineering and the human experience. The preceding examples give a brief taste of what can go wrong. Failure raises some questions:

HOW do engineering projects, specifically embedded systems, fail?

WHAT factors contribute to the failures?

WHO is involved in the failures?

WHEN and WHERE do the failures occur?

WHY do these factors and people contribute to failure?

Again, the broad focus of this dissertation to expose and avoid problems in designing and developing embedded systems. Many studies over the past 40 years point to major categories of failure and error and to the factors that lead to failures and error. The major categories are:

- Missed schedules [10], [11]
- Overrun budgets
- Incorrect or missed requirements [12]

- Unexpected interactions, problems, and failures [13].

Factors

Factors that lead to these failures and errors include the following:

- Not spending enough time designing [14]
- Not developing appropriate requirements [12], [14]
- Lack of understanding of the environment. "It's the Environment – To deal with a system as if it were a bundle of unrelated individual systems is, on the one hand, the method that saves the most cognitive energy. On the other hand, it is the method that guarantees neglect of side effects and repercussions and therefore guarantees failure. If we have no idea how the variables in a system influence one another, we cannot take these influences into account." [13]
- Too many features (problem with requirements and priorities) [12]
- Not using well-understood processes in software development [15], [16]
    - version control [12]
    - test-driven development
    - defect tracking
    - adequate review or test
    - coding standards
    - static analysis
- Cutting development short and rushing to market (one example is the Galaxy Note 7) [5]
- Mandating deadlines, budgets, and features (overconstraining the problem). Jack Ganssle bluntly puts schedule slips to, "We lie!" [11]
- Lack of coordination, communication, and collaboration [17]

One insight on meeting schedule comes from Quinnell's 2015 blog report on an industry-wide survey, ". . . only 38% of teams finished on or ahead of schedule. And the schedule shortfall has been growing over the years, even though more than half of teams now start their project with an off-the-shelf development board, 36% purchase rather than developing their own hardware, 71% re-use some of their hardware designs, and 86% re-use some part of their previous code." [10] This comment indicates that no one popular technique will magically cause development teams to meet schedule. This same survey indicates that most teams spend about 55% of their time in design (developing specifications, conceptual design, detailed design), while

45% of their time was in other activities (simulation, test and debug, prototyping, production handoff, documentation, administration) [10].

A Personal Example

Here is a personal example that illustrates several of the factors listed above. In the summer of 2010, I interviewed for a position as engineering manager for a startup company in the medical device industry. The company wanted to rush their diagnostic device to market by shortening the estimated approval time by the Food and Drug Administration (FDA) to four months or less; the standard range of time to get device approval through the FDA is somewhere between two and four years (and sometimes longer). They had a military general write a letter to the FDA to request expediting the process; while the device was primarily aimed at a consumer market, it could have a military application, albeit a much smaller market. The CEO asked me for my thoughts after my interview; for a small part of my response, I wrote, “You did say something about two (2) months versus six (6) months - I did not know if this meant to expedite FDA approval or to the next step with the FDA? My feeling is that 6 months is optimistic - but then again, I have never had military generals writing letters on my behalf to speed something up.” My experience was that FDA approval just could not be rushed. This company did not have that experience and was working various angles to reduce the time of approval – they did not understand the regulatory environment and were aggressively mandating schedules accordingly.

The company did not offer the position to me. After the rejection, I wrote this to a group of friends, “. . . [the company] told me that they didn't think that I was enthusiastic enough for the job and that they were going after other candidates. (I was asking questions about their FDA submission because I have some serious misgivings about their aggressive schedule as even being possible.)” Four and a half years after my interview, the company received the first FDA approval on its device [18].

Causes

As to primary causes of problems in development, industry insider Jack Ganssle states four problems: [14]

- Lack of complete requirements.
- Jumping into building too early; teams do not spend enough time to design the product.

- Lack of focus on quality. (Teams need to manage quality to achieve appropriate schedule and features.)
- Focus on fixing bugs, which does not achieve quality.

Several influential books on failure and the causes of failure have published within the past three decades. These books provide a more general perspective as context for why failures occur, and they prescribe ways that we might prevent failure. Dietrich Dorner's book cites the 1986 nuclear accident at Chernobyl as a case study among several; he claims that people have problems with complexity, dynamics of systems, invisible operations, ignorance, mistaken hypotheses, and time sequences of operations [13]. James Chiles' book cites many case studies, including accidents and disasters with deep sea platforms and airliners, the accident in 1979 at Three Mile Island, the 1986 Challenger accident, the Mark 14 torpedo failures of the early 1940s (almost no testing was done), the original incorrectly ground mirror on the Hubble Space Telescope, the 1963 Thresher submarine disaster, and the 1970 Apollo 13 accident. Chiles attributes these accidents and disasters to the same causes that Dorner does, plus Chiles adds that failures can cascade from small to massive proportions; and he contributes additional factors that lead to failure, including management pressures, operational and maintenance deviations, and human limitations such as tunnel vision, cognitive lock, and hypervigilance [19]. Henry Petroski's book agrees in large part with Dorner and Chiles. On p. 205, Petroski lists some causes from Thomas McKaig's 1962 book on building failures based on human foibles: ignorance in multiple forms during development and operation, economy (being cheap) in development and maintenance, carelessness, lack of coordination and communication, and being unprepared for unusual occurrences (e.g., fire, storms, earthquake) [17], [20].

Errors, faults, and failures ultimately come down to human weaknesses, flaws, and oversights. The Systems Engineering Book of Knowledge (SEBoK), ver. 1.8, states, "Studies in human reliability have shown that people trained to perform a specific operation make around 1─3 errors per hour in best case scenarios." [21] The goal of this dissertation is to develop a framework that avoids potential problems that can lead to errors, faults, and failures.

Preventing Failure or Reducing Its Effect

One can do several things to reduce failure. These recommended actions deal with the human condition and relationships. None of them requires technical tools, but they may benefit from the use of some tools, such as the proposed framework in this dissertation. These activities will not guarantee success, but they will help reduce the frequency and impact of failure:

- Establish a culture of integrity.
- Build trust.
- Continually remain open and communicate appropriately.
- Encourage experimentation, such as building prototypes.

What We Can Learn from the Ariane 5

Scaling up a design is a difficult task. Reusing a successful component in a different design does not guarantee success. This is where an understanding of systems' behaviors and component interactions becomes important. Just because a component works correctly in one environment with its attendant set of interactions does not mean that you can infer that it will operate correctly in another environment. You must use complete peer reviews and full systems tests, with functional production modules, to understand all the system dynamics and performance.

What We Can Learn from the Therac-25

Only good design produces a safe system; testing for faults does not guarantee safety. Leveson and Turner wrote, "Focusing on particular software bugs is not the way to make a safe system. Virtually all complex software can be made to behave in an unexpected fashion under certain conditions . . . The particular coding error is not as important as the general unsafe design of the software overall." [9]

The unforeseen consequences of complex interactions often cause the problems. Even fully tested modules that exhibit no erroneous behavior themselves can produce wildly variant results when interacting with other modules and systems.

Again, this is a case where a systems' view of component interactions, system behavior, and system synthesis is important. Understanding component interactions and nonobvious uses by operators is important in developing systems. This means that safety-critical systems need a general plan for system development that includes the full gamut of analyses, system design, design reviews, code inspections, tests, and system tests. Furthermore, independent systems should monitor and limit critical operations; an example would be hardware safeguards for software operations.

The Cost of Fixing Design Faults and Errors

Definitive studies of the costs for fixing errors, faults, and flaws in systems are difficult to find, but many authors assume that significant costs arise when fixing errors, faults, and flaws. The SWEBoK states on p. 182 (10-9), “In cases where system failure may have extremely severe consequences, overall dependability (hardware, software, and human or operational) is the main quality requirement over and above basic functionality. This is the case for the following reasons: system failures affect a large number of people; users often reject systems that are unreliable, unsafe, or insecure; ***system failure costs may be enormous*** [emphasis is mine]; and undependable systems may cause information loss.” [22] On p. 176 (10-3), SWEBoK is a little bit more specific on costs, “There are four cost of quality categories: prevention, appraisal, internal failure, and external failure. . . ***Costs of internal failures*** are those that are incurred to fix defects found during appraisal activities and discovered prior to delivery of the software product to the customer. ***External failure costs*** [again, emphases are mine] include activities to respond to software problems discovered after delivery to the customer.” [22]

Testing and review will never find all defects, errors, faults, or flaws, consequently there will always be costs with fixing problems. The SEBoK states the following characteristics of software that prevents exhaustive detection of all problems, “In all but the simplest cases, exhaustive testing of software is impossible because of the large number of decision paths through most programs. Also, the combined values of the input variables selected from a wide combinatorial range may reveal defects that other combinations of the variables would not detect. Software test cases and test scenarios are chosen in an attempt to gain confidence that the testing samples are representative of the ways the software will be used in practice. Structured reviews of software are an effective mechanism for finding defects, but the significant effort required limits exhaustive reviewing. . . Other points include:

1. All software testing approaches and techniques are heuristic. Hence, there is no universal "best" approach, practice, or technique for testing, since these must be selected based on the software context.
2. Exhaustive testing is not possible.
3. Errors in software tend to cluster within the software structures; therefore, any one specific approach or a random approach to testing is not advised.

4. Pesticide paradox exists. As a result, running the same test over and over on the same software-system provides no new information.
5. Testing can reveal the presence of defects but cannot guarantee that there will be no errors, except under the specific conditions of a given test.
6. Testing, including verification and validation (V&V), must be performed early and continually throughout the lifecycle (end to end).
7. Even after extensive testing and V&V, errors are likely to remain after long term use of the software." [21]

A simple, Monte Carlo simulation in Appendix B shows that even highly rigorous development will still leave hidden errors. It also shows that less effort to prevent errors will guarantee that more latent errors will remain hidden in the design.

Cost Escalation for Fixing Errors

The cost of fixing errors is much less when done in an early, rather than in a later, stage. INCOSE published a study on the cost to fix requirement errors at various stages of development. "If the cost of fixing a requirements error discovered during the requirements phase is defined to be 1 unit, the cost to fix that error if found during the design phase increases to 3 - 8 units; at the manufacturing/build phase, the cost to fix the error is 7 - 16 units; at the integration and test phase, the cost to fix the error becomes 21 - 78 units; and at the operations phase, the cost to fix the requirements error ranged from 29 units to more than 1500 units." [23]

Figure 2.1 graphs the potential range of cost escalation. Figure 2.2 graphs the lower bound of cost escalation for fixing errors.

Connect the cost to fix errors with the message of the previous section, which indicated that skipping known, good practices will guarantee latent design errors. Consequently, less rigor and not following known, good practices will also increase the cost dramatically.

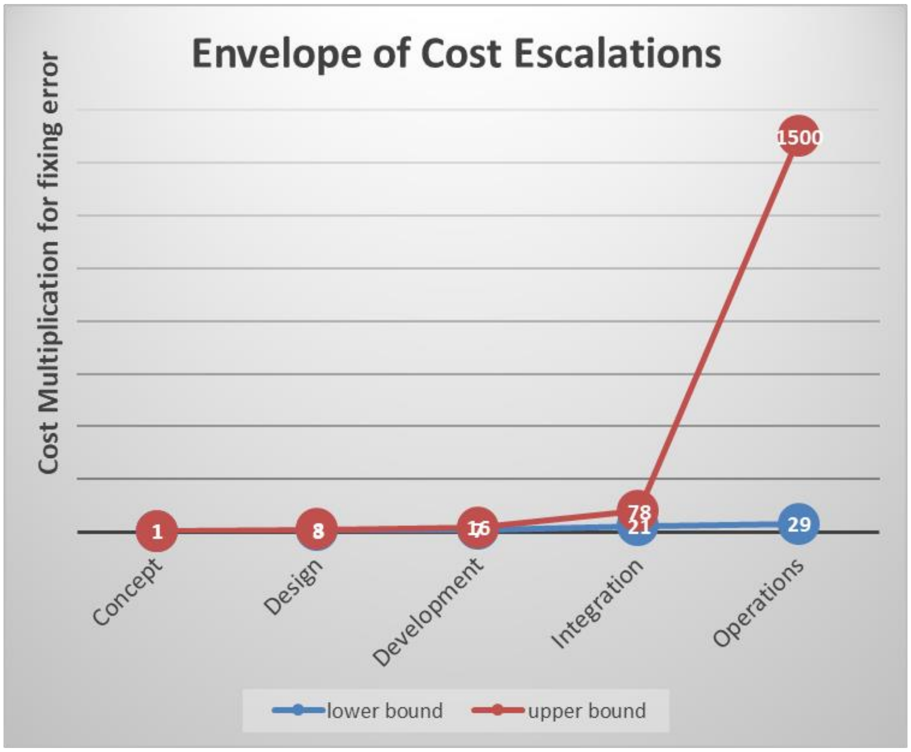


Figure 2.1 Range of cost escalation from INCOSE report for finding and fixing errors in requirements.

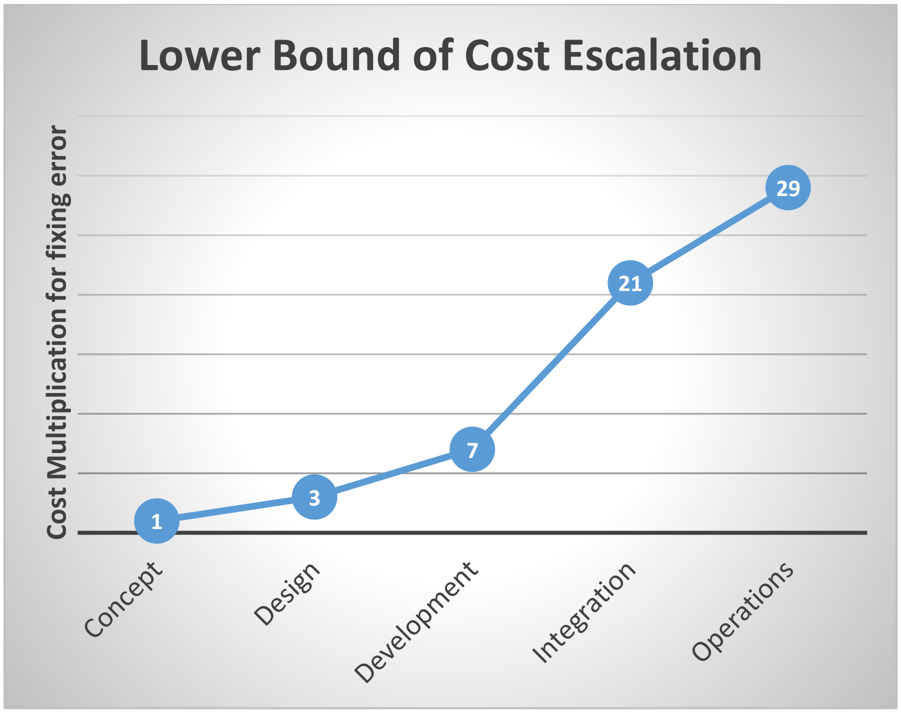


Figure 2.2 Lower bound of the cost escalation from INCOSE report for finding and fixing errors in requirements.

Known Good Practices

Barr Group's 2017 survey stated, "There is a large opportunity to easily improve the safety of embedded systems by more broadly using well-known software development best practices." They listed and then surveyed embedded developers for these known, good practices [16]:

- Coding standards
- Code reviews
- Static analysis
- Defect tracking
- Test-driven development
- Version control
- Industry and government standards
- Testing
    - System-level
    - Unit
    - Regression
    - Hardware-in-loop

A consistent and rationale application of these known, good practices are at the core of design assurance.

Why Do People Not Follow Known, Good Practices?

There are several lines of thought and philosophy in explaining why people make poor decisions and how we can reduce these problems. These tend to be found in psychology and behavioral economics but apply to design and development in engineering, as well.

Some heuristics and biases are reviewed in the following sections, explaining many dysfunctions of design teams and management. Other explorations include crowd intelligence, which is often as good or better than a best single estimate; swarming intelligence; and the Delphi method. Yet another approach, which can provide a single, optimized solution is game theory, such as the stable-marriage problem. Finally, the literature review covers various aspects of contract theory and behavioral economics that may institute rigor and discipline in system development.

Heuristics and Biases

Kahneman’s book on heuristics and biases helps explain the root of problems in reasoning and decision-making. The premise is that identifying the problems is part of the solution to avoiding them [24]. Lists of cognitive biases seem to have exploded to a ridiculous number, which may dilute individual impact and usefulness [25], [26]. Here are six prominent heuristics and biases that affect the estimation of effort, schedule, and the potential challenges that might be faced in embedded system development:

- The Law of Small Numbers [24]
- Anchoring [24]
- Confusing possibility with probability [24]
- Prospect Theory [27]
- Planning Fallacy [24]
- Sunk-Cost Fallacy [24]

The Law of Small Numbers

Early in the study of biases and heuristics, Kahneman and Tversky developed the tongue-in-cheek bias called the Law of Small Numbers. They succinctly summed up the problem in the abstract to their 1971 paper, “People have erroneous intuitions about the laws of chance. In particular, they regard a sample randomly drawn from a population as highly representative, that is, similar to the population in all essential characteristics [28].” I see this belief occur frequently in industry when a manager or team lead says, “We’ve done this before and it has always worked.” What is not said is that they did it a few times before, not many hundreds or thousands of times. The sample size is simply too small to reliably indicate what the population of possible outcomes could be.

Closely allied to the Law of Small Numbers is the Bias of Confidence over Doubt. Kahneman writes, “The strong bias toward believing that small samples closely resemble the population from which they are drawn is also part of a larger story; we are prone to exaggerate the consistency and coherence of what we see [24].” Tversky went on to study the hot hand streak in professional basketball only to conclude that, “The hot hand is entirely in the eye of the beholders, who are consistently too quick to perceive order and causality in randomness. The hot hand is a massive and widespread cognitive illusion [24], [29].” Development teams can fall into this trap, too, believing that they have a hot-hand-streak of success.

Anchoring

Kahneman, Tverksy, and Ariely provide clear examples of how low- or no-value information could influence people’s judgments [24], [30]. It is called anchoring and it provides a reference point from which people can move to arrive at a poor decision. Anchoring is bias in our thinking that has two demonstrable effects: one is called anchor-and-adjust while the other is called priming. Anchor-and-adjust works by, “. . . start[ing] from an anchoring number [or offer or situation], access whether it is too high or too low, and gradually adjust your estimate by mentally ‘moving’ from the anchor. The adjustment typically ends prematurely, because people stop when they are no longer certain that they should move farther [24].” Priming works differently than anchor-and-adjust, priming is a suggestion that, “. . . selectively evokes compatible evidence . . . and the selective activation of compatible thoughts produces a family of systematic errors that make us gullible and prone to believe too strongly whatever we believe [24], [31].”

Anchoring affects experts and novices alike. In a famous study of seasoned real estate agents versus business school students, Northcraft and Neale demonstrated that anchoring skewed real estate pricing almost equally between the two groups. The only real difference was that the students admitted that they were influenced by the anchor, whereas the real estate professionals denied the effect [24], [32].

Anchoring has direct implications for negotiations. The first offer or proposition can be a strong anchor. Galinsky and Mussweiler have proposed ways to minimize the anchoring effect; they suggest that negotiators focus their attention on arguments against the anchor [24], [33].

Another example that Kahneman raises has to do with plans and forecasts. He says that plans are best-case scenarios and suggests using Galinksy and Mussweiler’s proposal to avoid anchoring on plans when you forecast actual outcomes by thinking about how the plan could be wrong [24], [33].

Confusing Possibility or Plausibility with Probability

Often when returning home from a business trip I would telephone my wife and let her know that the plane had landed. Invariably, her response would be, “Good, you are home safe.” My immediate (and possibly ill-advised) reply would be, “No! I am now embarking on the most dangerous part of my journey.” The reason was that I lived in the Baltimore-Washington area and had a 22-mile automobile drive to reach my home. For my wife, the possibility of a plane crash with fatalities substituted for the probability of an actual airline crash. The probability of injury or death was much higher in that the homeward bound car drive, but not in her mind. Unfortunately, most of us can and do substitute possibility or plausibility for probability at times; we can do it in engineering, too. Kahneman writes, “The uncritical substitution of plausibility for probability has pernicious effects on judgments when scenarios are used as tools of forecasting [24].” Later in his book he talks about how people make judgments in the face of unlikely events:

- “People overestimate the probabilities of unlikely events.
- People overweight unlikely events in their decisions [24].”

I began my career, 35 years ago, by designing a system for the U.S. Navy to detect arcing faults in power switch cabinets and then to extinguish them. The one requirement, which was impossible for us to achieve, was to detect and to extinguish arcing faults without false alarm. This was a case where the possibility of an arcing fault causing very serious damage to a vessel was so magnified that the Navy required the false alarm probability to be 0. We designed, built, and fielded the system anyway without calculating a final probability of false alarm. (We used two different types of sensors to detect different physical aspects of an arc and required the system to receive timed activation signals from both types of sensors before concluding an arcing fault was indeed present. If the two types of signals appropriately coincided, then the system would trip an upstream circuit breaker to cut power and extinguish the arc. We did not know of any other situations, outside of an arcing fault, that would simultaneously activate both types of sensors [34].) The design has performed extraordinarily well in decades of operation on many navy ships and has had no reported false alarms in 20,000 ship-years of operations but we still cannot guarantee the absence of false alarms. This example demonstrates how a potentially plausible event with a very, very small probability event resulted in extremely risk-averse requirements; plausibility trumped probability in setting the requirements.

Prospect Theory

Prospect Theory may be the most obvious contribution that Kahneman and Tversky made to our understanding of human bias in decision making. Kahneman and Tversky observed that people have contrasting attitudes towards risk with favorable and unfavorable prospects [27]. The theory requires a reference point and couches changes from the reference point in terms of gains and losses. Most, if not all, people like gains more than losses. Kahneman writes, “We concluded from many such observations that ‘losses loom larger than gains’ and that people are *loss averse* [24], [35].” Figure 2.3 illustrates the perceived loss versus the perceived gain for the same increment in risk.

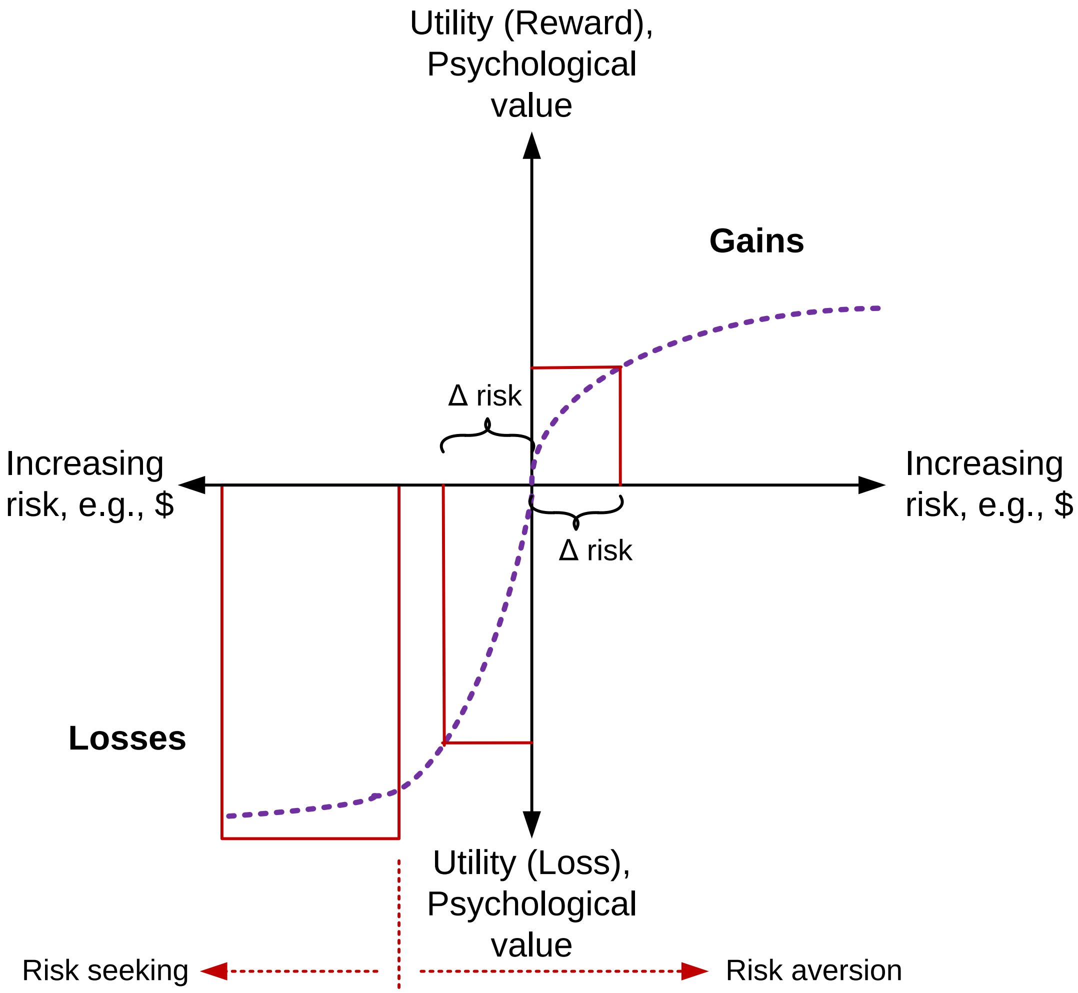


Figure 2.3 Prospect Theory indicates that the perceived loss will be greater than the perceived gain for the same increment in risk [24], [27]**. In the risk-seeking regime, an increment towards greater risk generates a much smaller increment in psychological loss. In risk aversion, an increment towards greater risk generates a much larger increment of psychological loss or much smaller increment of reward.**

One problem with this behavior is that people will throw good money after bad. They will not walk away from a bad investment if there is some glimmer of possibly making the money back. Kahneman says that this is not a contradiction to risk aversion; the bending of the value curve in the Losses quadrant is diminishing sensitivity and causes risk seeking! Consequently, Prospect Theory predicts these two claims:

- “In mixed gambles, where both a gain and a loss are possible, loss aversion causes extremely risk-averse choices.
- In bad choices, where a sure loss is compared to a larger loss that is merely probable, diminishing sensitivity causes risk seeking [24], [27].”

The second claim has application to engineering. While many people generally know that “throwing” more resources and people at a problem that is late will only delay the problem even more, too many times engineering management still does it [36], [37]. Prospect Theory would predict this risk-seeking behavior in the face of loss type and leads to the Sunk-Cost Fallacy.

Tversky, Kahneman, and Prelec have refined the theory to provide decision weightings that people tend to use [38], [39]. Tverksy and Kahneman said, “. . . for both positive and negative prospects, people overweight low probabilities and underweight moderate and high probabilities [38].” Figure 2.4 illustrates this behavior.

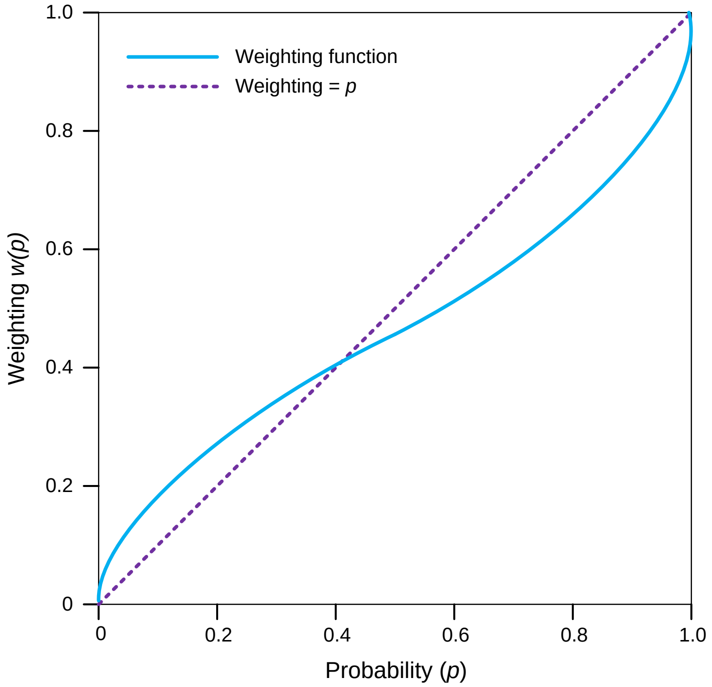


Figure 2.4 Example of a weighting function that people use to predict probabilities. It over-estimates low probabilities and under-estimates moderate and high probabilities.

Two Fallacies

Two fallacies arise from these biases and heuristics that affect engineering teams: the Planning Fallacy and the Sunk-Cost Fallacy. The Planning Fallacy describes plans and forecasts with these two characteristics, they:

- "are unrealistically close to best-case scenarios
- could be improved by consulting the statistics of similar cases [24]."

The Sunk-Cost Fallacy is, "The decision to invest additional resources in a losing account, when better investments are available . . . [24]" Engineering teams often commit both fallacies when they estimate shorter schedules and lower budgets than prudent and then when overruns inevitably occur, push more people and resources into the projects than prudent.

Stress in Decision-Making

Ariely claims that, ". . . every one of us . . . underpredicts the effects that passion have on our behavior. . . predictions are wrong by a large margin [30]." His experiments have shown that in a state of arousal people make different decisions than in a calm state. A bit later, he writes, ". . . study suggested that our inability to understand ourselves in a different emotional state does not seem to improve with experience. . . [30]" Ariely discusses aroused state versus a normally calm demeanor. It seems that we do not learn to improve our decision-making even with experience. I posit that nearly the same can occur to a degree in stress versus calm. Under stress, we make decisions that we would not make in a calm state [40].

Ariely also describes failures in procrastination and control. When we stress, we tend to lose control and if we have a problem with procrastination, then we procrastinate under stress. To avoid these problems, Ariely says that we need to avoid situations that cause that stress. He prescribes concrete precommitment, a contract to set deadlines and boundaries, to avoid situations where we get into stress [30], [41].

Market Norms Versus Social Norms

Ariely lays out the premise that we live and work in two different modes or norms. The first is the market norm, the second is the social norm. Clearly, when we work, we are in the market norm; we supply labor in return for payment. He writes, ". . . social norms also apply to the workplace. In general, people work for a paycheck, but there are other, intangible benefits we get from our jobs. These are also very real and very important, yet much less understood [30]." As a team strives to develop a new product or service, it needs to understand that various motivations are at work to push us towards peak performance and efficiency. Understanding how social norms play into the workplace will help these teams to perform better.

Wisdom of Crowds, and Crowdsourcing

Characteristics

In 2004, Surowiecki wrote a book, *The Wisdom of Crowds*, that describes situations where large groups of disparate people can collectively make very good decisions. He cited many different cases, such as the 1986 *Challenger* accident; the search for the submarine, *Scorpion*, following its 1968 disappearance; stock market actions and picks; and even the estimation of weights of bulls at county fairs. While this technique is not universally applicable, the accumulated average opinions/votes/picks of many people can make good decisions. Surowiecki writes, "With most things, the average is mediocrity. With decision making, it's often excellence. You could say it's as if we've been programmed to be collectively smart [42]." I'll start with Surowiecki's term "wisdom of crowds" but over the past decade the name of the activity has morphed to crowdsourcing, which I will use later in this section.

For decision-making to function well, the crowd must have four characteristics:

1. "diversity of opinion (each person should have some private information, even if it's just an eccentric interpretation of the known facts),
2. independence (people's opinions are not determined by the opinions of those around them),
3. decentralization (people are able to specialize and draw on local knowledge), and
4. aggregation (some mechanism exists for turning private judgments into a collective decision) [42]."

Surowiecki writes that diversity and independence are necessary because decisions are the product of disagreement and contest, not from consensus and compromise. Independence has two characteristics; one, it prevents mistakes from being correlated, and two, new information tends to arise from individuals; both give a more robust perspective of the problem. Decentralization aids specialization and independence, which harbors new or local knowledge that is not easily passed along but is still important, even valuable. Finally, decentralization only works if all the information within the population can be aggregated to a cumulative answer or decision.

Cognition, Coordination, and Cooperation

Crowd intelligence works in specific situations; Surowiecki listed these situations as problems with cognition, coordination, and cooperation. Cognition problems deal with understanding a problem and predicting an answer. Coordination problems are those that have individual components or individuals navigating environments with not only their own motivations and self-interests in mind but predictions of actions by other components or individuals; swarms are an expression of this type of problem; they are bottom-up not top-down (mandated) solutions. Cooperation is like coordination but requires the members of the group to "... adopt a broader definition of self-interest than the myopic one [of] maximizing profits in the short term demand. . . they [must] trust those around them [42]. . ."

A downside for groups that either have few members or are homogeneous in expertise is that consensus in decision-making is problematic. Surowiecki wrote on p. 36, "Homogeneous groups become cohesive more easily than diverse groups, and as they become more cohesive they also become more dependent on the group, more insulated from outside opinion, and therefore more convinced that the group's judgment on important issues must be right. [42]" Edwards outlines this thought with others in his 2009 article; he argues, as does Surowiecki, for diversity in groups and the consideration of contrary viewpoints [43]. Surowiecki expands on this thought on p. 184, "One of the real dangers that small groups face is emphasizing consensus over dissent . . . The confrontation with a dissenting view, logically enough, forces the majority to interrogate its own positions more seriously. . . Without the devil's advocate, though, it's likely that the group's meetings actually made its judgment about the possible problem worse. [42]" The point is that diversity of expertise and skills and the ability to express and accommodate them is important to avoid consensus decision-making that performs poorly.

Crowdsourcing

Over the past decade, crowdsourcing has become the preferred term of art to describe the actions of crowds. Some recent papers are referenced [42], [44]–[55]. In 2010, researchers at the University of Washington and Carnegie Mellon University demonstrated that human players of a game, called FoldIt, could configure a complex protein and could perform much better and faster than a computer program. They wrote, ". . . top Foldit players excel at solving challenging structure refinement problems in which substantial backbone rearrangements are necessary to achieve burial of hydrophobic residues. Players working collaboratively develop a rich assortment of new strategies and algorithms; unlike computational approaches, they explore not only conformational space but also the space of possible search strategies [56]." They wanted to know if, ". . . more complex scientific problems can be similarly solved with human-directed computing." One thing that becomes apparent is that part of the solution is a well-posed problem and method. The FoldIt game was sophisticated but well-designed so that even people without any biochemistry background could play it.

Taxonomy

More recently Hosseini et al., at Bournemouth University, UK, have defined crowdsourcing in this way, "Crowdsourcing is a form of innovation seeking, problem solving and knowledge acquisition paradigm which is based on the power of voluntary participation of a usually large, diverse number of people for performing tasks within loose contractual settings through an open call [57]." They analyzed 113 research papers to find key elements to apply crowdsourcing. They came up with five problem types where crowdsourcing applies well: opinion collection tasks, basic tasks that use common sense and everyday knowledge, tasks that require expertise, competitive tasks, and collaborative fundraising [58]. I will only use the classification of tasks that require expertise in this dissertation. The crowd features for solving tasks that require expertise competence and motivation; the studies indicate that people involved in these crowds want feedback and social incentives, they want to see the results of their contributions; finally, the platforms for crowdsourcing must provide enrollment and authentication, skill declaration, task broadcast, feedback, quality thresholds, management of platform misuse, ease of use, and attraction [57].

Crowdsourcing in Software Engineering

Within software engineering, two recent articles in IEEE Software describes some of the motivations of crowds and applications of crowdsourcing. The first article, by LaToza and Hoek in 2016, identified motivations of contributors to crowdsourcing in software development as, ". . . gain experience with new technologies, bolster their reputation, and contribute to a good cause [59]." The article also listed eight dimensions for crowdsourcing [59]:

1. crowd size – numbers to address problems effectively
2. task length – amount of time to complete a task
3. expertise demands – level of domain knowledge required
4. locus of control – ownership of the creation of tasks
5. incentives – factors that motivate contributors
6. task interdependence – how tasks build on one another
7. task context – system information required by contributor
8. replication – number of times a task must be completed in a redundant fashion.

Speed of completion depends on short tasks that are largely self-contained with minimal coordination. One of the surprising results is that contributors with diverse backgrounds and experiences can sometimes lead to unanticipated, alternative solutions.

The second article, by Stol, LaToza, and Bird in 2017, discussed a taxonomy for crowdsourcing tasks and reasons for using crowdsourcing in software development. Their effort was an attempt that might be more general than Hosseini, et. al. (2015). Stol, LaToza, and Bird maintain that there are four primary crowdsourcing tasks: rating, processing, creation, and problem solving [60]. They also claim that there are four reasons for using crowdsourcing in software development: as an alternative to outsourcing, reduce the time to market, generate a range of solutions, and employ specific experts to find the best solution for a problem.

While there are benefits to crowdsourcing in software development, there are problems, too. Hasteer et al. indicated that they found problems in three areas [61]. They found that cost could escalate due to a much longer time developing specifications and answering queries from the crowd. They found problems with the schedule, too; contributors did not always adhere to deadlines. They found that quality assurance was very difficult, particularly in removing bugs.

The Delphi Method

"The Delphi method is an iterative process to collect and distill the anonymous judgments of experts using a series of data collection and analysis techniques interspersed with feedback. The Delphi method is well suited as a research instrument when there is incomplete knowledge about a problem or phenomenon; . . . The Delphi method works especially well when the goal is to improve our understanding of problems, opportunities, solutions, or to develop forecasts. [62]" The method and its variants have gone by at least three different names: Delphi, modified Delphi, and wideband Delphi [49], [63]–[66]. The common kernel of operation is a small group of anonymous experts share opinions over a few iterations with the goal to converge on a consensus opinion. I call it the Delphi method in this dissertation.

The difference between surveys and the Delphi method is that a small group of experts consider the questions of a survey rather than a statistically large group and they do so through multiple iterations and feedback their opinions for due consideration during the next iteration. Lilja, Laakso, and Palomäki claim that the Delphi method is, ". . . useful when the phenomenon under study is complex or when the topic is somehow delicate – difficult to define, awkward to talk about, politically delicate, etc – or the number of members in the focus group is relatively small. [67]"

"The original Delphi method was developed by Norman Dalkey [68] of the RAND Corporation in the 1950's for a U.S. sponsored military project. . . Rowe and Wright (1999) [69] characterize the classical Delphi method by four key features:

1. Anonymity of Delphi participants: allows the participants to freely express their opinions without undue social pressures to conform from others in the group. Decisions are evaluated on their merit, rather than who has proposed the idea.
2. Iteration: allows the participants to refine their views in light of the progress of the group's work from round to round.
3. Controlled feedback: informs the participants of the other participant's perspectives, and provides the opportunity for Delphi participants to clarify or change their views.
4. Statistical aggregation of group response: allows for a quantitative analysis and interpretation of data. [62]"

General Format of Operations

I use a variant of the Delphi method from what is described by Skulmoski, Hartman, and Krahn [62]. The Delphi method is very similar to a survey except that you run it for multiple iterations with the same group of subject matter experts (SMEs). For each iteration, a moderator distributes the survey, collects the comments and data, distributes the results, and initiates the next iteration. The first round can be the longest and most involved because the you need to set up the area of research and the questions to answer, and then select the SMEs. The goal is for the group to move to a consensus in opinion and in data results. Should consensus not converge completely, then a limit on the iterations can stop the operation. Figure 2.5 outlines my version of the Delphi method; it still serves as a reasonable description of the general method.

The Delphi method has been used extensively in education, economics, and management research, but much less so in engineering. A partial reason for engineering's reticence to use the Delphi method is that it tends to address qualitative issues and data more easily. Quantitative results need a well posed problem.

Selecting Experts

Selecting the appropriate subject matter experts (SMEs) is an important concern for the success of the Delphi method. Skulmoski, Hartman, and Krahn provide four primary qualifications for SMEs in the Delphi method:

1. "knowledge and experience with the issues under investigation,
2. capacity and willingness to participate,
3. sufficient time to participate in the Delphi,
4. effective communication skills [62]."

For most engineering estimates and forecast, a SME will need to be an experienced engineer in the area of interest (embedded systems for this dissertation); this almost always means someone with a breadth of experience in several different industries and a long tenure doing engineering and management. Lilja, Laakso, and Palomäki developed a similar definition and process for selecting a panel of SMEs, ". . . an expert panel should consist of 3-9 members as a minimum, even numbers should be allowed, experts should have sufficient knowledge of the discipline gathered via education and / or experience, and their expertise should be recognized by colleagues or some third party with the capability of evaluating expertise in this field. [67]"

The number of SMEs needed varies widely. The minimum seems to be three or four SMEs. The maximum is in the hundreds; some studies have upwards of 200. Skulmoski, Hartman, and Krahn referenced 30 studies, with some that used three SMEs and one that used 171 SMEs [62]. Once the number of SMEs becomes large – maybe 400 or more – then the study becomes swarm intelligence at work.

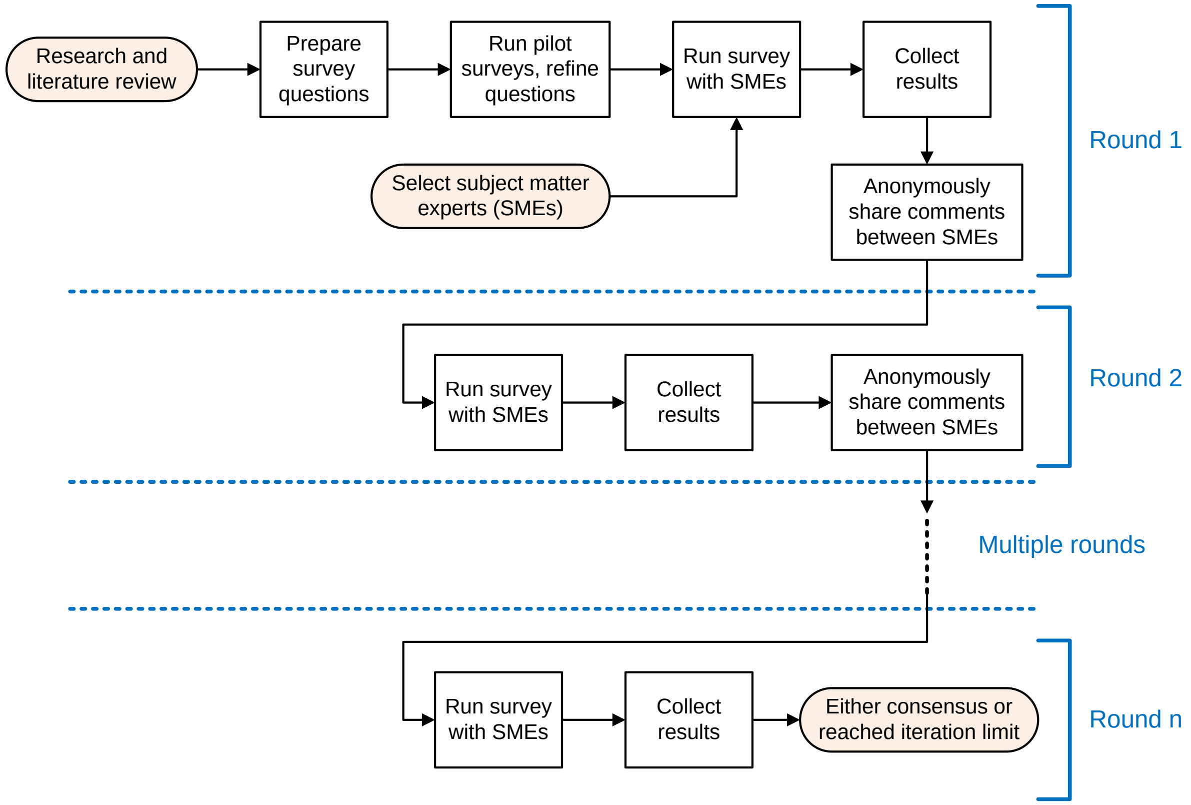

Research and literature review
Prepare survey questions
Run pilot surveys, refine questions
Run survey with SMEs
Collect results
Select subject matter experts (SMEs)
Anonymously share comments between SMEs
Round 1
Run survey with SMEs
Collect results
Anonymously share comments between SMEs
Round 2
Multiple rounds
Run survey with SMEs
Collect results
Either consensus or reached iteration limit
Round n

Figure 2.5 Outline of the Delphi method.

Arenas for Applying the Delphi Method

Lilja, Laakso, and Palomäki begin the conclusion of their paper by stating, "Although mainly used in futurology and social, health, and medical disciplines, the Delphi method can be applied to certain types of research in technical sciences, software engineering and related disciplines as well. [67]"

For this dissertation and its research, I focus on engineering applications. Skulimowski discusses several application areas for the Delphi method; it can, ". . . identify elements (subsystems, variables, structural coefficients) of socio-econometric and techno-econometric models or [can] define characteristics of future decision problems included in anticipatory models. . . [it can also] create rankings of technologies, technological trajectories and technology development projects. Delphi exercises can help us to discover, confirm or exclude causal relationships between events, trends, and variables. [70]" Skulimowski's recent work in 2017 is for an expert Delphi survey in a decision support service that is cloud-based [70], which strongly supports the framework that I propose in this dissertation.

In 2016, Lan, Chen, Chiu, and Lai used the Delphi method to uncover factors that identify critical components and maintenance strategies for air-compressor turbine used in aircraft environmental systems [71]. Their work used the Delphi method to identify factors, not predict the actual maintenance strategies.

In 2017, Qi et al., used the Delphi method to identify risk factors and their weights in billing for low-voltage services [72]. In 2015, Lu and Fan used the Delphi method to identify factors for a risk assessment system for studying security in a port facility [73]. In 2011, Wang and Yuan used the Delphi method with other methods to study the economics of Ocean Energy Development in China [74]. In 2008, Zhang, Zhou, and Wu used the Delphi method to identify factors for identifying security factors for an embedded system [75]. These four works used the Delphi method in conjunction with other methods to perform the final analysis.

In 2011, Wu, Huang, and Hung the Delphi method to collect expert opinions and rank them for designing a curriculum for fuel cell technologies [66]. Their work used the Delphi method in conjunction with a Grey Relational Analysis to perform the final analysis.

Some researchers used the Delphi method to compare to a specific technique that they used for specific applications; their specific techniques can give more reliable and significant results than the Delphi method [76], [77]. This makes sense – custom techniques for specific applications often can give better results than more general methods, such as the Delphi method.

Advantages and Disadvantages of the Delphi Method

The advantages of the Delphi method over surveys can be numerous and significant. It is easy to use and can provide a reasonable analysis. It has wide application, requires many fewer people than a survey. Here are some enumerations of the advantageous of the Delphi method:

- Okoli and Pawlowski state, “The Delphi group size does not depend on statistical power, but rather on group dynamics for arriving at consensus among experts. . . Studies have consistently shown that for questions requiring expert judgment, the average of individual responses is inferior to the averages produced by group decision processes; research has explicitly shown that the Delphi method bears this out. . . the Delphi method can employ further construct validation by asking experts to validate the researcher’s interpretation and categorization of the variables. The fact that Delphi is not anonymous (to the researcher) permits this validation step, unlike many surveys. . . Respondents are always anonymous to each other, but never anonymous to the researcher. This gives the researchers more opportunity to follow up for clarifications and further qualitative data. . . Non-response is typically very low in Delphi surveys, since most researchers have personally obtained assurances of participation. . . attrition [also] tends to be low in Delphi studies, and the researchers usually can easily ascertain the cause by talking with the dropouts. . . Delphi studies inherently provide richer data [than surveys], because of their multiple iterations and their response revision due to feedback. Moreover, Delphi participants tend to be open to follow-up interviews. [78]"

- Lilja, Laakso, and Palomäki state that the anonymity of the Delphi method guarantees more objective results. "Anonymity supports independence by avoiding the limits and problems of expression and listening to one another, which are always present in face-to-face expert groups. The official position or unofficial status of a panelist does not affect the opinion or its formation and expression. Also, a member of the panel can be free from the fear of losing face, even if she/he gives an answer or comment that others might find to be wrong or inaccurate. A panelist also does not need to be wary of attitudes, which her/his employer might find inappropriate to be stated in public. In interest or value conflicts, issues do not become personalized in the same way as in face-to-face communication. Avoiding face-to-face communication between the members of the group also avoids impacts of mimicking and other forms of inarticulate communication that occurs as a problem in other kind of focus group methods. Furthermore, anonymity provides safety for panelists in cases where the panelists or their employers are competitors, especially if some or all of the panelists come from the business world. [67]"

The Delphi method may have a few, potential disadvantages. It requires a deeper commitment of time and effort by the SMEs than a survey. Until recently, the Delphi method provided primarily qualitative insights, so detailed, quantitative forecasts or estimations were not always available. Here are some concerns:

- Lilja, Laakso, and Palomäki indicate three areas of potential problems, ". . . there has been a lot of discussion, occasionally even strong disagreement, on the scientific reliability of the results assessed by Delphi. The critics argue that the number of respondents in an average Delphi research study is too small to guarantee the reliability of the work. The second argument presented by critics is that the method by which the respondents are selected for the

Delphi panel is not objective or based on probability, and therefore the answers cannot be considered reliable in the scientific meaning. The fact that results obtained from different panels may differ from each other has also been seen as a sign of the unreliability of the Delphi method . . . [67]”

- Specific techniques for very specific applications can sometimes perform more reliably and with higher significance than the Delphi method [76], [77].

Game Theory

Traditional versus Behavioral Game Theory

Game theory has formal rigor but simplifying assumptions that constrain formulations to apply to specific situations. Traditional game theory uses rational, or idealized, human behavior, which does not always give results that line up with empirical results. Wikipedia describes traditional game theory as using “. . . theoretical models to determine the most beneficial choice of all players in a game. Game theory uses rational choice theory along with assumptions of players' common knowledge in order to predict utility-maximizing decisions. It also allows for players to predict their opponents' strategies. Traditional game theory is a primarily normative theory as it seeks to pinpoint the decision rational players should choose, but does not attempt to explain why that decision was made. Rationality is a primary assumption of game theory, so there are not explanations for different forms of rational decisions or irrational decisions. [79]” Behavioral game theory combines game theory with experimental economics and psychology and allows deviations from strict (and simplifying) rationality by including framing, fairness, altruism, and a lack of independence. The papers cited by Wikipedia distinguish between traditional and behavioral game theory, “Traditional game theory focuses on mathematical equilibriums, utility maximizing, and rational choice; in contrast, behavioral game theory focuses on choices made by participants in studies and is game theory applied to experiments. Choices studied in behavioral game theory are not always rational and do not always represent the utility maximizing choice. [79]”

Recent Developments – Stackelberg Game

More recently behavioral game theory has been moving into engineering thought. Colleagues at Kansas State University have used Prospect Theory to weight decisions within a

Stackelberg game to model and analyze interactions between active consumers and aggregators in variable electricity pricing; they are studying the impact of irrationality on payoffs for both consumers and aggregators and the resulting variations in voltage on a power bus [80]. This is just one example how behavioral game theory is permeating engineering; it holds promise for much research and elucidation of how people interact with technology in the future.

Another thesis out of Iowa State University uses a sequential game in negotiations for defense acquisitions [81]. It focuses on bargaining between the government as player 1 and a defense contractor as player 2. While bargaining usually applies to sharing a single resource, this thesis uses different value propositions for the two players; the government, player 1, focuses on operational success; the defense contractor, player 2, focuses on expected profit. As a Stackelberg game, the first proposer has the advantage in the final payoff; typically, this would be the government or player 1.

For this dissertation, a Stackelberg game may fit some negotiations between management and engineering teams in determining the mix of preferred schedule, budget, features, and quality. This is a form of cooperative decision-making, as is the Gale-Shapley algorithm that follows.

Recent Developments – Matching Game with Gale-Shapley Algorithm

Considering negotiations and contracts between management and engineering teams, the stable-marriage problem that uses the Gale-Shapley algorithm of matching may be a better choice than the Stackelberg Game [82]. The Gale-Shapley algorithm has been used to match doctors to residency programs, match recipients to kidney donors, and match roommates [83]. In this dissertation, matching could apply to negotiations between management and engineering teams in developing a product. Management has a set of preferred solutions for schedule and budget while engineering teams have a set of preferred solutions for features and quality. Gale and Shapley showed that the final set of matches between these preferences will be stable and unchanging [82]. They also showed that the first mover (e.g., in the stable marriage game, men proposing marriage, women then accept or reject the proposals) has the dominant strategy; this dominant strategy favors management in setting a course of action, assuming management makes the first negotiation proposal, that fits the final set of options containing schedule, budget, features, and quality. There are potential ways to manipulate the Gale-Shapley algorithm, but they are beyond the scope of this dissertation [84].

Decision Theory

Background to Decision Theory

Decisions are often made with incomplete data and incomplete understanding. Waiting to gather all the necessary data before making decisions may not be feasible or even possible to complete projects within set or reasonable timeframes.

Decision theory under uncertainty has been an active area of research for 60 years. Decision theory has a well-developed framework for rational choices and known probabilities [85]–[87]. The tools of decision theory are useful if they are used but, as already mentioned, people have biases and heuristics that circumvent the effectiveness of these tools in many, if not most, situations.

Systems engineering also uses decision theory for specific applications. The textbook edited by Parnell, Driscoll, and Henderson provides a number of basic operations used with systems engineering [88]. In chapter 3, Driscoll and Kucik outline the typical life cycle model for most systems; on p. 69 of reference [88] they divide the model into seven basic phases:

1. Establish the need – identify stakeholders and preliminary requirements
2. Develop the system concept
3. Design and develop the system
4. Produce the system
5. Deploy the system
6. Operate the system
7. Retire the system.

In the same chapter, Driscoll and Kucik go on to address risk management by outlining six common issues (p. 79 of reference [88]):

1. What can go wrong?
2. What is the likelihood?
3. What are the consequences?
4. What are the options?
5. What are the tradeoffs?

6. What are the impacts of current decisions on future options?

Chapter 4 (pp. 95 – 131 of reference [88]), by West, Kobza, and Goerger, discusses how models are used in decision making. All decisions are based on some level of modeling and analysis; the question becomes, "Are the models sufficient to make a good decision?" Chapter 5 (pp. 137 – 178 of reference [88]), by Pohl and Nachtmann, covers life cycle costing, which has different potential components including:

- Expert judgment
- Production estimates
- Activity-based costing
- Learning curves
- Breakeven analysis
- Uncertainty and risk analysis
- Replacement analysis.

Most of these are empirical relationships and difficult to quantify with precision or accuracy.

Negotiations in Decision Making

Clearly, design and development in other than the simplest situations has multiple agents. These agents must negotiate the form of the system and the ultimate set of solutions to achieve that system. On p. 265, Middelkoop et al. write, "Different domain experts and domain specific tools are needed to bring specialized knowledge and analysis to bear on different aspects of the design problem. . . Since design decisions made by the different individuals are not independent, the team must coordinate and integrate their decision in order to reach a final design agreeable to all participants. . . Even when all the participants can be brought together in one place, reaching the best compromise to a constrained multicriteria decision problem can be challenging. [89]"

Multicriteria and multi-objective methods may not resolve to a single, integrated decision that is optimized. These methods must deal with many components for the realization of a product: product description, design, manufacturing, testing, and life-cycle support. Those methods that do attempt resolution include preference aggregation, compromise solution, bargaining, fair division, and heuristics (such as diffusion, annealing, and genetic algorithms). Middelkoop et al. review and reference these methods [89]. I will return to bargaining and negotiations shortly in outlining contract theory. At the end of their chapter on p. 278

Middelkoop et al. write, "... design of a society of agents that is ideally suited to solving a complex design problem remains an open problem. [89]"

Contract Theory, Negotiations, and Bargaining

Background to Contract Theory

In October 2016, the Nobel Prize in Economics (its formal title is the, "Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel 2016") was awarded to Oliver Hart and Bengt Holmström for their contributions to contract theory. The Royal Swedish Academy of Sciences produced an excellent report [90] that summarizes Hart and Holmström's sets of works. The report begins, "An eternal obstacle to human cooperation is that people have different interests. In modern societies, conflicts of interests are often mitigated – if not completely resolved – by contractual arrangements. Well-designed contracts provide incentives for the contracting parties to exploit the prospective gains from cooperation. [90]" Li and Kolotilin clarify that contract theory is more than a strict study of legally binding contracts, "Contract theory . . . studies the design of formal and informal agreements that motivate people with conflicting interests to take mutually beneficial actions. Contract theory guides us in structuring arrangements between employers and employees, shareholders and chief executives, and companies and their suppliers. [91]" This dissertation focuses on the agreements between employers and employees in developing embedded systems.

The report of the Royal Swedish Academy of Sciences goes on to set up a typical contracting problem, "A *principal* engages an *agent* to take certain actions on the principal's behalf. However, the principal cannot directly observe the agent's actions, which creates a problem of *moral hazard*: the agent may take actions that increase his own payoff but reduce the overall surplus of the relationship. . . To alleviate this moral-hazard problem, the principal may offer a compensation package which ties the manager's income to some (observable and verifiable) performance measure. We refer to this as *paying for performance*. . . But any performance measure is likely to be imprecise and noisy, so in the end the optimal compensation schedule must trade off incentive-provision against risk-sharing. [90]" Hart and Holmström's work then identified a number of issues in tradeoffs between incentives and risks: informativeness principle, free-loading, multitasking, career concerns, incomplete contracting, and allocation of decision rights. [90], [91] The informativeness principle encapsulates concerns and issues when parties to a contract cannot observe every action by their counterparts. In particular, when a principal pays compensation to an agent, that compensation should depend on variables, called signals, that provide clear and unambiguous information about the agent's actions [90]. An example of this would be for a client to pay a consultant to develop an embedded subsystem based on the performance and reliability of the final subsystem; the signals would be specified performance parameters and reliability estimates. Here lies the rub – actual reliability can be difficult to confirm until long after the subsystem has been delivered, installed, fielded, and made operational. There could be unexpected uses of the subsystem that shorten its operational life; or there could be undiscovered flaws in the design; or there could be environmental extremes encountered only after the subsystem has departed on a mission, for example, a scientific instrument on a spacecraft. Generating signals for these conditions may be extraordinarily difficult to conceive or measure.

Another problem within the informativeness principle is developing adequate signals to assess individual contributions within a development team. A free-loading problem is often difficult to see when individual efforts combine in a collective output. The previous example of developing an embedded subsystem is often performed by a team of engineers who develop circuitry and software modules within the subsystem. The ultimate functionality and robustness of the subsystem may not become apparent for years after delivery. A free-loading member of the team might not complete a fully functional or robust module but still be paid equally to other team members. Holmström proposes a "budget breaker" as a neutral third party to control incentives and improve satisfactory results for all parties [90], [91]. I propose an expanded use of a neutral third party, an unbiased mediator, to help set stronger agreements (contracts) within and between development teams.

Holmström also studied multitasking and career concerns as moral hazards within the informativeness principle [90], [91]. Multitasking means that an agent performs several tasks within a contract; the problem is that some tasks may be more observable than others or one task masks the signals from other tasks, but all could be important. Multitasking requires very clear signals to implement appropriate incentives for the agents. On the other hand, career concerns may drive agents at different intensities during different phases of their careers. Career concerns can be a motivating factor for some agents but less so for others. Motivation and incentives are tightly coupled along some dimensions; these are an active field of research for industries that develop embedded systems. Developing appropriate signals for motives and their related incentives is a sub-goal of this dissertation research.

Hart studied the incomplete contract and decision or control rights. Li and Kolotilin write, ". . . it is impossible to write a contract that anticipates every potentially relevant future contingency. Consequently, the allocation of control rights becomes a powerful tool for creating incentives. . . When unforeseen contingencies arise, the parties have to bargain over what to do. . . asset owners have stronger bargaining power . . . [91]" For the purpose of this dissertation, development teams and management bring various resources and capabilities that could be called assets. The unbiased mediator will provide insight into the industry and the subsystem's environments and regulations, which will help anticipate some, though not all, future contingencies; regardless, an unbiased mediator should reduce the number of unforeseen contingencies. The unbiased mediator will also provide rationale for its suggestions to the development team and management, which will help even the playing field during negotiations on the development contracts between developers and management.

Motivation

The study of motivation goes back many years to the 1950s. Maslow published his book, "Motivation and Personality," in 1954 and Herzberg wrote, "The Motivation to Work," in 1959 [92], [93]. The technical literature is sparse in uniformly addressing the motivations of development teams for embedded systems. Incentives are tied to motivation; understanding the motivation of a contractual partner or agent quickly points to appropriate incentives. Here are few different aspects of motivations that have been published:

- In 1964, Garth Mangum wrote about wage incentives with the focus on repetitive tasks, including piecework assembly. While his focus was on wage incentives for factory workers, he mentioned leisure time as non-wage incentive [94]. Some non-wage incentives can be very important to professional staff.
- In 2000, George Farris published findings on rewards versus retention of technical staff; he found, ". . . that one-time recognition awards are not effective in preventing turnover; rather, small, non-cash awards and good old fashioned permanent salary increases and promotion

were most effective in reducing the reported likelihood of turnover. One-time awards are no substitute for permanent rewards when it comes to retention [95].”

- In 2009, Soares, Júnior, and Meira discussed incentive systems in software organizations. They find that incentive systems need appropriate measurement systems (this recalls the signals in contract theory). Wrong measurements (signals) lead to the opposite of incentives [96].
- Fear can be a de-motivating factor. In 2008, Casey and Richardson wrote about three case studies concerning geographically separated teams attempting to work together [97]. The work built on two other case studies that they covered in 2005 and found very similar results. Fear was the factor and distance exacerbated it. They wrote in conclusion, “Our research has highlighted fear and its impact on motivation, trust, teamness, communication and knowledge transfer as having a direct bearing on the success of implementing this approach.[97]” This results were upheld by Steinberga and Smite in 2011 [98] and Junior et al. in 2012 [99].
- In 2013, França, de Araújo, and da Silva studied motivation in software engineers. They found, “The results point to task variety and technical challenges as the main drivers of motivation, and inequity and high workload (caused by poor estimations in the software process) as the main obstacles to motivation in the organization. [100]” This was a qualitative study in one firm; 14 engineers and managers were interviewed and then five were selected for a diary study over 17 business days. While not statistically significant and not over many different firms, the results point to similar motivating and de-motivating factors found by other researchers.
- Between 2007 and 2009, Hall, Beecham, Baddoo, Sharp, and Robinson at three different universities in the United Kingdom found a number of factors, most were non-financial, that motivate software developers [101]–[104]. These factors conceivably are very similar to the motivating factors for other types of engineers, such as electrical, computer, mechanical, industrial, and manufacturing.
- Recently, in 2016, Holle, Elsesser, Schuhmacher, and Lindemann proposed a methodology to detect motivating factors for organizations to work together. This might be different than assessing individual motivation, but the work may have some interesting parallels [105].

Out of these studies, the works by Beecham et al. and Kenneth Browne appear to be the most directly applicable to this dissertation research and proposal. Each of these sets of studies were well conceived and executed. They lay a reasonable and testable basis for studying motivation within development teams.

For example, Beecham et al. wrote, “The most frequently cited motivators in the literature are, ‘the need to identify with the task’ such as having clear goals, a personal interest, understanding the purpose of a task, how it fits in with the whole, having job satisfaction; and working on an identifiable piece of quality work. Having a clear career path and a variety of tasks is also found motivating in several papers. The literature suggests it is important to involve the engineer in decision making, and to participate and work with others, which appears to go against characteristics of independence and introversion which are cited in many papers. [101]” Beecham et al. went on to say, “What might motivate someone in the early stages of their career may end up de-motivating them in the latter stages of their career. For example, the newly recruited Software Engineer could be highly motivated by job security and close supervision, whereas these same factors, especially close supervision, could turn out to be de- motivating to a seasoned Software Engineer. An experienced Software Engineer is more likely to be motivated by challenges, opportunities for recognition and autonomy. [101]”

Kenneth Browne, in his 2013 dissertation, found the following. “Engineers were found to have a significant difference in the motivational factor of extrinsic/instrumental motivation (t = -2.33, df = 176.1, p < .05). The effect size (d) was .31. There were no significant differences found for sources of motivation factors relative to gender, tenure or age. . . Motivator factors including achievement, recognition, responsibility, how interesting the work is, and growth and advancement were found, in this study, to not differ significantly between engineers and their peer knowledge workers. . . This research suggests that engineers place significantly less value on extrinsic/instrumental motivation factors of motivation, and that they will demonstrate more persistence and intensity in their work than their peer knowledge workers as the likelihood of tangible rewards diminish. Engineers will be less responsive to motivators such as performance-based rewards, bonuses linked to goal attainment, or group goal attainment payments than their peer knowledge workers. [106]”

Product Development and Project Management

The popular literature is rife with articles and blogs about motivation, references [107]–[111] are but a very few examples of such pieces. Not to take away from their veracity but many are anecdotal and insufficient to provide a solid basis for motivating and managing technical staff. Project management, on the other hand, has a huge base in the peer-reviewed literature, particularly for the technical aspects of project management. The “soft skill” of understanding engineers and developers, however, seems to be less well represented in the technical, peer-reviewed literature for the management and development of embedded systems. Managing motivation is important, as it will feed into the contracts within and between development teams and management. Here are a few examples:

- In 1962, Hahn wrote about the relationships between managers and researchers; these seem to extrapolate to engineers and developers, too. Hahn wrote, “The primary element in this team is the common goals of the manager and researchers – the solution to a mutually recognized problem. This is strengthened by the larger and more enduring set of common objectives of the enterprise of which all parties are members. Another important element is the notion of peers. The manager and the researcher must each recognize the expertise and unique contributions the other can make. [112]” The concept of each other’s value to the team is foundational to the proposition within this dissertation. Hahn also recognized some characteristics of a properly functioning team, “1) involvement, 2) continuous learning, and 3) professionalism. . . The weaknesses of the team are two: communication and measurement. [112]” The fundamental proposition within this dissertation implicitly recognizes these characteristics.
- During the mid-1980s, General Electric began instituting a quality assurance organization within research and development. The primary finding was, “. . . the majority of the problems involve getting management commitment. Upper management must dedicate themselves to quality and to following the methods and standards that result. This commitment must occur in the form of ***visual*** evidence, as well as verbal evangelizing. Without persistent dedication, quality assurance (the organization and the concept) will not be successful. In addition, "ownership" is a critical issue. People must be included in the creation of processes they will be expected to use. [113]” Ownership and commitment at all levels are necessary for the assurance of quality.

- In 1995, Brown and Eisenhardt performed an extensive review of literature and concluded that research into new product development primarily followed one of three paths of investigation: rational plan, communications, and disciplined problem solving. Brown and Eisenhardt synthesized these three paths into one proposed model. These issues still concern industry today. [114]
- In 2001, Zhang and Doll espoused clear vision and teamwork to achieve success in new product development. ". . . [to achieve] success in [new product development], the project team should have a clear vision of product lines and platforms for specific markets. . . [a multi-functional team should develop] project priorities consistent with product strategy and resource availability based on information of customer, technology and competition. Furthermore, product definition should be explicitly developed and documented, and project targets (time, cost, quality) and relative priorities should be clear. . . to reduce chaos and the unpredictable, some knowledge bases and team vision must be emphasized. . . [then] product development process needs to be disciplined with important teamwork, focus on some specific quantitative measurements, and commit to the goal. Finally, market activities should seek predictability and order with strong financial orientation, and commit to established values and businesses. [115]" Vision and clear goals are keystones to good project management; they support the motivations of the team and team members. The work by Beecham et al. and Kenneth Browne both identify vision and goals as important motivating factors for engineers and developers.
- A variety of papers identify aspects of good program management. Schwarz and Pothier noted that communication, relationships, and flexibility were important [116]. Closely related to program management is systems engineering; in small companies and teams, one person may perform both program management and systems engineering; Moore indicates that the systems engineer should have multiple capabilities: understand technology, facilitate analysis, test, and debug, be a skillful diplomat, negotiator, and unpopular steward, and be a multi-disciplinary designer [117]. Peters reviewed the differences in motivating factors for managers and software engineers and then outlined a plan to increase both intellectual and professional investment in the project by the software engineers [118]. For installing new train control for the New York Transit, Alexander and Mortlock found that contractors, suppliers, and the client all needed to act together, provide mutual support, and maintain

open communications [119]; Huatao and Jixin found some of the same things, particularly good interactions and mutual support, within research and development teams [120]. Jetter and Sperry found that different levels of innovation and uncertainty require different approaches to development strategy, a single-approach to development strategy is inefficient at best [121]. Hutchinson, Rouncefield, and Whittle found that understanding and buy-in are important when implementing model-driven engineering (MDE); autocratic, top-down management caused problems with adoption of MDE [122]. For multi-team collaboration, Pang et al. found, ". . . that by working closely together, with a common goal, both government and industry can benefit greatly from the hosted payload model. [123]" Finally, Chiang and Wu model and point out that there are times to stick to a strict schedule and contract while there are other times when flexibility under contractual contingencies is better [124].

Cultural and Societal Differences

Over the past 60 years, sociologists and psychologists have studied cultural differences. Hofstede performed a ground-breaking study on cultural differences for IBM in the 1960s. He divided attitudes into four dimensions; many people have followed his lead over the ensuing 40 years to examine and expand those dimensions [125]–[132]. In 2001 and again in 2011, Hofstede wrote review papers on cultural differences and expanded the dimensions to six, they are [133]:

1. *Power Distance*, related to the different solutions to the basic problem of human inequality;
2. *Uncertainty Avoidance*, related to the level of stress in a society in the face of an unknown future;
3. *Individualism* versus *Collectivism*, related to the integration of individuals into primary groups;
4. *Masculinity* versus *Femininity*, related to the division of emotional roles between women and men;
5. *Long Term* versus *Short Term Orientation*, related to the choice of focus for people's efforts: the future or the present and past.
6. *Indulgence* versus *Restraint*, related to the gratification versus control of basic human desires related to enjoying life.

The first three, power distance, uncertainty avoidance, and individualism versus collectivism, will have the greatest impact on negotiations and contracts within business.

Negotiations and Bargaining within Contract Theory

Harvard Business Press, in its Pocket Mentor series, publishes a book on negotiations [134]. The book describes a nine-step process for negotiations and focuses on negotiations between different firms, consequently, it assumes that parties to the negotiations have only partial knowledge of each other; the degree of ignorance may be more extensive and general an assumption than may exist within a team or company. Along the way, though, it gives some tactics that are both diplomatic and useful: set a positive tone, explain your interest and understand theirs, actively listen; these are useful techniques both between external firms and within a team or company.

For this dissertation, only some techniques of negotiation are important. A report by the Harvard Law School says, "Negotiation often involves some degree of conflict, . . . [135]" Even within teams this is true and usually unavoidable to a degree (Tuckman's work, begun in 1965 and summarized in his 1977 paper, indicates conflict is normal) [136]. The techniques include rapport, face-to face meeting, trust, and open communications.

". . . rapport is a powerful force that can promote mutually beneficial agreements. . . Rapport can be thought of as a state of positive mutual attention marked by harmony and affinity [135]." Good rapport does the following, ". . . take turns speaking and show signs of understanding, such as nods. . . mimicry – of posture, facial expressions, tone of voice, and mannerisms – which often occurs without conscious awareness. [135]"

". . . there's no substitute for a face-to face meeting. . . [with its] greeting rituals of face-to-face interaction, such as small talk and personal disclosure . . . [but] rapport, such as eye contact and mimicry, are impossible when bargaining remotely. . . When negotiators have had no prior relationship or contact, communication technologies can perpetuate unfamiliarity and distrust. . . The impersonal nature of e-mail, in particular, makes it difficult to establish feelings of trust and interpersonal connection, which can lead to misunderstandings and even impasse. [135]"

Even small talk before negotiations are important to rapport. The reports says, "Experimental research confirms that small talk sets the stage for an atmosphere of positivity, trust, and openness that ultimately creates value. Even when talks must occur remotely, negotiators can build substantial rapport through prenegotiation chats. [135]" The report went to say, "Nurturing mutual self-disclosure can reap benefits when unexpected opportunities arise. [135]" Finally, the report says, "Shrewd negotiators are proactive rapport builders who go the extra mile for face-to-face discussions, make time for small talk, and reveal their personalities and interests. [135]"

Project Contract and Team Contract

In the early-1990s, research in program management began to espouse contracts between team members and management. In 1996, Goldense wrote the following about team contracts. "Team Contract metrics . . . are the central focus of discussions between executive management and autonomous empowered teams at the point that development funding is approved for projects. These metrics result from the team's own estimates of the resource and time requirements necessary to complete the project. These are the measures that the team is willing to live by during the development process, and be measured by upon conclusion of the project. . . The operating assumption . . . is that the team has accurately estimated the project and that management should believe the estimates. . . the team is an equal participant with management. If the team is willing to commit to their estimates and live by them, than [sic] management can only undermine the team's buy-in by mandating different figures than the ones that the team spent many hours developing together. There must be a 'trust' between management and the team regarding development estimates and time frames in accelerated environments. The team, in conjunction with management . . . commit to each other's specific goals. These goals typically include overall time-to-market, product cost, and market size estimates. [137]" A webpage prepared by Kristi Tyran outlines team contracts concisely. [138]

Since the Goldense lecture in 1996, the concept of team contracts has not gained a foothold in most technical businesses. Personal communications with industry leaders indicate that team contracts are not used in most situations:

- Jack Ganssle wrote, "I never see team contracts in industry. Perhaps partly as a result, I get a never-ending stream of emails from people working in broken, dysfunctional teams. [139]"
- Michael Barr indicates that he has not heard of anyone using team contracts. [140]

- Bob Rassa wrote, “I don't think companies use such anymore - at least not in the defense [industry]. [141]”
- Michael Gard wrote, "The working arrangements in most places I’ve worked have been semi-formal at best – the sort of thing that can appear on performance appraisals. Working engineers hate the formal details for their lack of flexibility, and Management tends to avoid it because a history of ‘contract’ changes makes it evident the exercise wasn’t terribly well planned (by Management) in the first place. I’ve typically worked on projects with multi-year development cycles, so personnel changes, personnel reassignments (or resignations), and changes in response to competing priorities usually compromised most formal contractual arrangements. [142]"

One discipline that picked up and implemented design/development by contract to some degree has been software engineering. Barry Boehm and Bertrand Meyer have championed commitment and rigorous processes for developing software since the 1980s [143], [144]. (Their ideas were preceded in some measure by Floyd in 1967 and Hoare in 1969 [145], [146].) Meyer has been a proponent of the design-by-contract method for software, which has a use case format with preconditions, actions, assumptions, guarantees, and postconditions to achieve a complete contract that can be rigorously checked and verified [144]. Recently, design by contract has moved beyond just software and made inroads to electronic subsystems; in 2014, Nuzzo et al. applied design by contract to the design of aircraft power systems [147].

Benveniste et al. have extended the design by contract to include hardware and systems; their work aimed to manage software, platform, and integration complexities, supply chain, requirements, and risk [148], [149]. More recently in 2015, Benveniste et al. have prepared two reports on system design by contract where they develop and then apply a formal algebra of contracts [150], [151]. They address abstraction and refinement, which relate to the vertical flow of design between different layers of abstraction, and composition and decomposition, which relate to design processes at the same layer. They provide guidelines for operators, such as refinement, conjunction, and composition, and for the assumptions and guarantees that alter the definition of the operators. They consider the different viewpoints of different teams that contribute different subsystems. Examples of viewpoints, suggested by Benveniste et al., include the behavioral viewpoint for different functions, the timing viewpoint for different activities, and the safety viewpoint that considers fault propagation, effect, and handling [150].

The formal approach currently advocated in the design by contract is laudable; the theoretical foundation needs to be established. The concern is that there is no readily useful and easy to operate tool that helps industry personnel implement contracts for design.

Small Group Dynamics

The literature review to this point focuses on specifics of individuals and groups for developing embedded systems. Most development involves one or more small teams; hence group dynamics are very important to successful operation.

Much has been written about group dynamics. Sutcliffe has applied group dynamics specifically to requirements engineering [152]–[154]. He summarized most issues in his 2011 paper by dividing issues into three sets: 1) tasks, agents, and roles, 2) knowledge, skills, and abilities, and 3) values, beliefs, and attitudes [152]. He provided the following checklist that contains most of the necessary topics collaboration and coordination:

- "Achieving goals: . . . Shared awareness of collective social goals and progress towards their achievement are established needs . . .
- Generating and sharing knowledge: this social goal implies technical system requirements for visualising and presenting information, as well as for selective access controls to organise sharing according to different roles. These requirements are important components in ecommunities.
- Maintaining integrity: this goal reinforces the need for shared awareness . . . [make] group members' contributions visible to deter social loafing, and rewarding contributions with public displays . . .
- Promoting agreement: tools for negotiation support, summarising of different arguments, and voting functions are indicated. Design rationale representations are one effective means of supporting negotiation.
- Motivating members: shared awareness tools and making contributions visible help to motivate group members. . . promote the sense of belonging to groups and social identity. [152]"

Estimation

Studies in estimating the effort to develop software and systems began in the 1970s. Barry Boehm published the COCOMO model for estimating the effort to develop software in

1981 [155]. Furthermore, [156] states, "The model parameters [for COCOMO] are derived from fitting a regression formula using data from historical projects (61 projects for COCOMO 81 and 163 projects for COCOMO II)." Work in developing estimation models and methods has expanded beyond software alone and progressed steadily into systems engineering over the past 15 years [21], [155], [157]–[162]. Valerdi (2005), Ilseng (2006), and Holmes (2012) all outlined the same basic estimation methods [163]–[165]:

- Top-down
- Bottom-up and activity based
- Estimation by analogy, past history, heuristic and rule of thumb, case studies
- Expert judgment, Delphi methods, guesstimates
- Design to Cost
- Parametric cost estimation methods

Valerdi, Ilseng, and Holmes describe the advantages and disadvantages of each set of methods. Ideally, the bottom-up method would be the most accurate but for two very important caveats: 1) we don't necessarily foresee all the activities that will go into design and development of a system, and 2) things change during development and the bottom-up estimate does not predict it unless it is regularly reviewed, revised, and updated. Consequently, using a mixture of the methods helps bound the estimation, which does not guarantee a good estimate but might provide a more accurate picture than a single method.

If crowdsourcing proves feasible, it might fit with "guesstimates" or stand on its own as a seventh method. Crowdsourcing might be considered similar in some aspects to the Delphi method [63], [64], only it is less formal and potentially faster than the Delphi methods; crowdsourcing does not use feedback or iteration. Crowdsourcing versus the Delphi method is an active area of research; at least one study in software engineering found that crowdsourcing was more accurate than the Delphi method in predicting the distribution pattern of defects in software [49].

Swarm intelligence might be considered a combination of the Delphi method and crowdsourcing but with many more people who may not be experts. Swarm intelligence and the Delphi method both require feedback and collaboration, unlike crowdsourcing. In 2016, Rosenberg and colleagues compared swarm intelligence to human experts and to crowdsourcing in predicting the scores of college football games [46], [50], [55]. Comparing swarm intelligence to experts, Rosenberg writes, ". . . by forming a real-time swarm intelligence, the group of random sports fans boosted their collective performance and out-performed experts. [55]" In comparing swarm intelligence to crowdsourcing, Rosenberg and colleagues write, ". . . swarming, with closed-loop feedback, is potentially a more effective method for tapping the insights of groups than traditional polling. [50]"

Several concerns arise with these comparisons to swarm intelligence. First, Rosenberg's studies were for single-point outcomes, the scores of football games, and not the complex sets of features and requirements that embedded systems might have; granted, football is a complex game with many factors that influence the final score; maybe the success of an embedded system can be boiled down to a single score but that is not as obvious to do as arriving at the final score of a football game. Second, swarm intelligence was not tested and compared against the Delphi method arrangement; this would be an interesting experiment to conduct – Zhai et al. touched on this type of engineering in their recent papers on expert-citizen engineering [166], [167].

Requirements

Requirements provide both the basis for specifying design and development and then the metrics for confirming that the development is correct. Requirements are at the pivot point between design execution and development review and revision; consequently, requirements are an essential part of design assurance.

The Systems Engineering Book of Knowledge, p. 33, gives a very brief historical perspective on systems engineering and requirements; codification of requirements as a necessary component began in the Industrial Revolution and accelerated after World War II [21]. Laplante does an outstanding job introducing the art and science of requirements in his book, "Requirements Engineering for Software and Systems, Second Edition." [168] A very small sampling of papers reveals that requirements and their development are very active areas of research [12], [169]–[172].

Design Assurance

Closed Loop and Iterative

Good design is an iterative and closed-loop activity. While putting together the terms, "closed-loop" and "iterative," may seem redundant and is in many situations, they are not if feedback occurs only once and stops. Good design runs multiple iterations through the closed loop.

Figure 2.6 illustrates a general closed-loop form that can describe activities at all levels of detail – from a basic component to subsystems to system integration; loops can be nested and performed in parallel or sequentially. Closing the loop requires module reviews, peer reviews, design reviews, tools (requirements-test management, test-driven development, configuration management), and testing. All these activities must be planned. They also tend to center around the requirements. Larman and Basili published an interesting review of the history of the iterative development of systems, particularly software systems, that succeed well versus projects that did not use iterative development and failed [173].

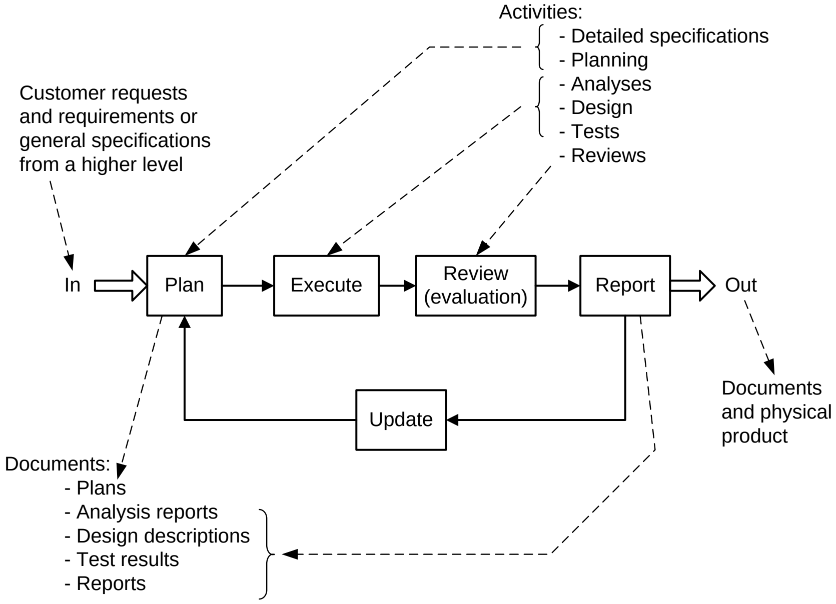


Figure 2.6 General format of closed-loop activities that can occur at all levels of detail. PERRU for Plan, Execute, Review, Report, and Update. (© 2008-2014 by Kim Fowler. Used with permission. All rights reserved.)

Design assurance requires timely, thoughtful, and thorough implementation so that activities (e.g., schematic capture, coding, prototyping components and subsystems, testing, production, demonstration), installation, service, and documents are prepared and useful for future explanation, utility, modifications, and upgrades. Larman and Basili indicated that rigid application of specific practices, specifically the waterfall model of development, did not assure success [173]. A flexible and thoughtful process that allows adjustments and revisions over multiple, short durations has a higher probability of success, an example is Shlaer and Mellor's

1997 position paper, “Recursive Design of an Application-Independent Architecture” [174]. But Shlaer and Mellor did conclude with two important problems that are still unsolved:

1. “. . . we can expand the characterization step. Although a great deal of information is collected during system characterization, we are by no means confident we are collecting the right information. In particular, experience indicates that a better quantitative understanding of the control threads is significant in making numerous architectural decisions. [174]” Research into quantitative system characterization is ongoing, but even now, boundaries on characterization are not well understood.
2. “. . . we can build a collection of sample architectures. We believe that such a collection would provide an expanded repertoire of designs for consideration by system architects. Each architecture could be more closely linked to the characterization work. . . [174]“ A collection of sample architectures might be useful, but it begs the question how is one architecture selected as applicable or modified to be applicable? Shlaer and Mellor coupled the architecture selection with characterization.

In 1995, Bowen and Hinckey defined a formal method as, “. . . a mathematically based technique for describing a system. Using formal methods, people can systematically specify, develop, and verify a system. [175]” Formal methods, codesign, standards, and industry-standard processes all contribute to design assurance. These have been emphasized since the 1960s as important for assuring design; a very small sampling is included in the references as an indication of the extent and breadth of research these issues have occupied engineering research in recent years [21], [22], [117], [175]–[182]. While these methods, standards, and processes have been shown to be good practices for assuring design quality, the surveys of the Barr Group in recent years show that a sizable fraction of companies do not implement basic good practices in designing and developing embedded systems [15], [16].

Checklists

Atul Gawande’s delightful book, “A Checklist Manifesto,” discusses four different fields of application for checklists: medicine, civil engineering construction, commercial aviation, and financial investing [183]. For this research and this dissertation, I propose a format for checklists that combines the planning for civil engineering’s "skyscraper construction" and the financial market investment checklist as longer term, more prescriptive frameworks and away from the surgery checklist or pilot checklist that tend to be a format that is shorter term, more reactive.

Gawande’s book also touches on degree of difficulty and how well a checklist might cover different types of problems. The book provides compelling evidence that checklists can significantly reduce mistakes and omissions. Most importantly, the book gives three significant results from properly implementing checklists are [183]:

1. Encourage teamwork
2. Encourage discipline
3. Improve outcomes with no increase in skill.

Checklists should have a basic set of good characteristics [183], [184]:

- Efficient
- Concise, reminders of only the most critical and important steps
- Easy to use
- Practical
- Make priorities clearer
- Prompt people to function better as a team.

On the other hand, checklists need to avoid being vague, imprecise, and impractical. Wood, in his book “Building Maintenance,” wrote, “A checklist has much to commend it. It is normally clearly set out, based on previous experience, and easy and quick to complete. It is important, however, to keep in mind the limitations of checklists – their strengths can also be weaknesses; they can be rather reductionist. Checklists can be either too short or too detailed. Short checklists have advantages in terms of speed and simplicity of application, but they may miss some important aspect for the particular situation. A long list may be thought to be comprehensive but it may still be missing that particular and unexpected something. There can be great value in anticipating this possibility by adding a heading or two labelled ‘other’ to prompt the surveyor to think and look more widely. [185]”

The Evaluation Center at Western Michigan University has developed a number of checklists primarily for evaluating educational situations [186]–[188]. Wingate writes, “Michael Scriven has identified and analyzed several types of checklists: laundry lists, sequential checklists, iterative checklists, diagnostic checklists, and criteria of merit checklists. But regardless of focus, purpose, or complexity, all checklists share in common a basic mnemonic function: they remind (or inform) us what we are supposed to do, check, look for. . . [186],[187]" Wingate goes on to discuss, ". . . 3 overlapping categories: (1) evaluation planning and management checklists; (2) criteria of merit checklists; and where these categories overlap, (3) metaevaluation checklists—checklists that can serve both as evaluation guides as well as criteria of merit checklists for evaluating evaluations. [187]"

More detailed checklists, such as the civil engineering plans in Gawande's book, might be considered plans [183]. For larger projects, such as civil engineering or spacecraft, checklists and lists of best practices may be found in the literature. An example of best practices for spacecraft may be found in reference [189].

There are caveats to the slavish application of checklists. Two reference sources originate from the arena of clinical health services but easily extend to the development of mission-critical embedded systems. Henriksen and Brady write, "Checklist development, use and acceptance come with challenges. Development requires a team of individuals or a consensus body that is adept in best practice guidelines and the underlying evidence base, in the realities of clinical work, in measurement and in human factors design principles, and has the perseverance to engage in successive pilot-testing trials and improvements. If put together too rapidly, checklists can be excessively lengthy, ambiguous, devoid of clinical reality and insensitive to the needs of front-line users. Even when well developed and accepted by end-users, there is potential for cognitive drift that repetition, by itself, seems to induce. . . investigators who have successfully implemented checklists are quick to tell us that it is not all about the checklist. A prevailing patient safety culture, teamwork, leadership commitment, well-conceived measurement, and attention to implementation, workflow and organisational change issues all need to be carefully aligned before checklists can be properly tested. [190]" Raman, Leveson, et al. write, "Time-outs and checklists can prevent some types of adverse events, but they need to be carefully designed. Additional interventions aimed at improving safety controls in the system design are needed to augment the use of checklists. Customization of checklists for specialized surgical procedures may reduce adverse events. [191]"

Collaboration Tools

Regan and Richardson published a checklist for implementing field tests in transportation systems; it provided two layers of procedures, activities and tasks, and critical considerations that contained both rationale and “dos” and “don’ts” [192]. Regan and Richardson’s checklist could be viewed as migrating towards a collaborative tool for a technical, mission-critical field.

Nan et al. published a dynamic clinical checklist that also borders on a collaborative tool, they proposed, “. . . a meta-model for dynamic checklist, which reuses existing infrastructure from outside the clinical domain. By analyzing a use case derived from clinical practice, we summarize four checklist modeling requirements, which are the clinical workflow, the clinical rule, the layout and the interoperability between the workflow and the rule. [193]”

Over the past 60 years, many people have advocated various aspects of systems engineering and system development; by way of example, Hopkins, in 1961, and Belev, in 1989, both laid groundwork for systems engineering and future collaborative tools [194], [195]. Both used an iterative checklist approach, after Scriven, to provide guidelines during development [186]. Hopkins published a diagram of the various stages through development that portrays the multiplicity and complexity of feedback paths during development. Hopkins also wrote, “Although it is imperative that research, development, design, and test of techniques, parts, processes, and/or components be pursued before requirements are fully defined, the tendency is to proceed too rapidly with physical design and not to go into nearly enough detail on requirements. Likewise, although the danger is always present of carrying a preliminary step too far when it would be better to proceed into design, use cut-and-try methods, and then re-analyze the requirements before specifications are finalized, there is much more of a tendency in the workaday world of development contracts to consider too few pertinent factors in each early step than too many. . . the system engineer must know or gain a working understanding of the pertinent characteristics of all those mechanisms, techniques, interrelationships, environments, etc., that may be involved in the structure, processes, flows, or behavior of the system. Since this understanding may involve many fields of knowledge, use of the interdisciplinary team approach in system development is natural and essential. [194]” Clearly some people have understood the need for balance, feedback, and iteration in developing system over the past 60 years.

Collaboration tools are ubiquitous on the worldwide web, even in the narrow space of embedded systems engineering; one website alone lists 61 different tools for collecting and maintaining requirements [196]. While many tools exist, few are quickly accepted. Wierba, Finholt, and Steves wrote, “. . . collaborative tools must be clearly superior to existing practices to merit the effort of deployment, adoption, and subsequent use, since the burden of learning and mastering a new tool in a corporate environment may not outweigh the perceived benefits. [197]" Integrated collaboration tools, such as the Mentor Expedition Multi-Board System Design package, offer a comprehensive environment that speeds the development of an embedded system [198]; the downsides are cost and learning curve – commitment to the tool for multiple projects should quickly offset these concerns, but that is just the sort of hurdle that many engineering teams don't make. The challenge seems to be taking the long-term view and accepting the short-term expense to incorporate effective tools.

How Do You Motivate Use of Good Practices?

The clear need is for a method or tool that encourages a more rigorous approach to designing and developing products. Following the groundwork laid by Kahneman and Tversky, Atul Gawande, and Richard Thaler, any usable method must have these characteristics:

- Accounts for biases in human thinking [24], [27], [35], [38],
- Is easy to use and does not require learning new techniques [183],
- Is fast and cheap, and
- Provides a "nudge" or "prompted choice" [199]. Thaler defines a nudge as, "some small feature in the environment that attracts our attention and influences behavior." [199]

A new method or tool must have these characteristics to motivate use and to make it nearly irresistible to the engineering and business community. For the method to succeed, motivation and utility require a human-centered solution. A closed-form engineering solution is not enough to provide the necessary motivation and utility.

Part of the research in this work is to find or develop a "nudge" that makes good practices something that development teams want to do. The "nudge" must account for the four bulleted characteristics above and be attractive to the development team. It must provide value-added utility to each project.

Chapter 3 - Proposed Framework for Design Assurance

Purpose

With the characteristics of a successful method having been enumerated in the previous chapter, I can now approach a solution to the problem at hand. The solution should aid unbiased thinking by participants, be easy to use, not require learning of new techniques, be inexpensive, and provide a "nudge." A good solution will encourage users to follow known, good practices and simultaneously encourage the preparation of better estimates of schedule and budget.

The Framework

The proposed framework is one of estimation and checklist activities to aid upfront planning. It provides baseline minimum values for time, effort and cost for which a project cannot improve. The framework addresses process and procedures, without being specific to design features or requirements. Figure 3.1 expands the detail of Figure 1.1 and outlines the proposed framework with plug-in modules.

The framework provides a structured approach to addressing planning, contracting, and executing clean-sheet developments of embedded systems. It allows for the insertion of plug-in modules, which in turn can address specific concerns within project development and perform specific functions, such as estimating time or recommending a specific decision.

Instituting process is a very large problem. This dissertation focuses on developing, testing, and confirming operations of three modules:

- an estimator of time, effort and potential challenges within a project,
- a recommender for the build-versus-buy decision of subsystems, and
- human-driven advice.

Chapters 4, 5, and 6 detail these three proposed modules and their potential efficacy.

I did not consider the implementation of a project contract within the scope of this work. It can be a work unto itself, as noted by the Nobel prize in economics in 2016 [90].

I consider the checklist to be a future project, as well. Appendix C outlines some of the features of a potential checklist.

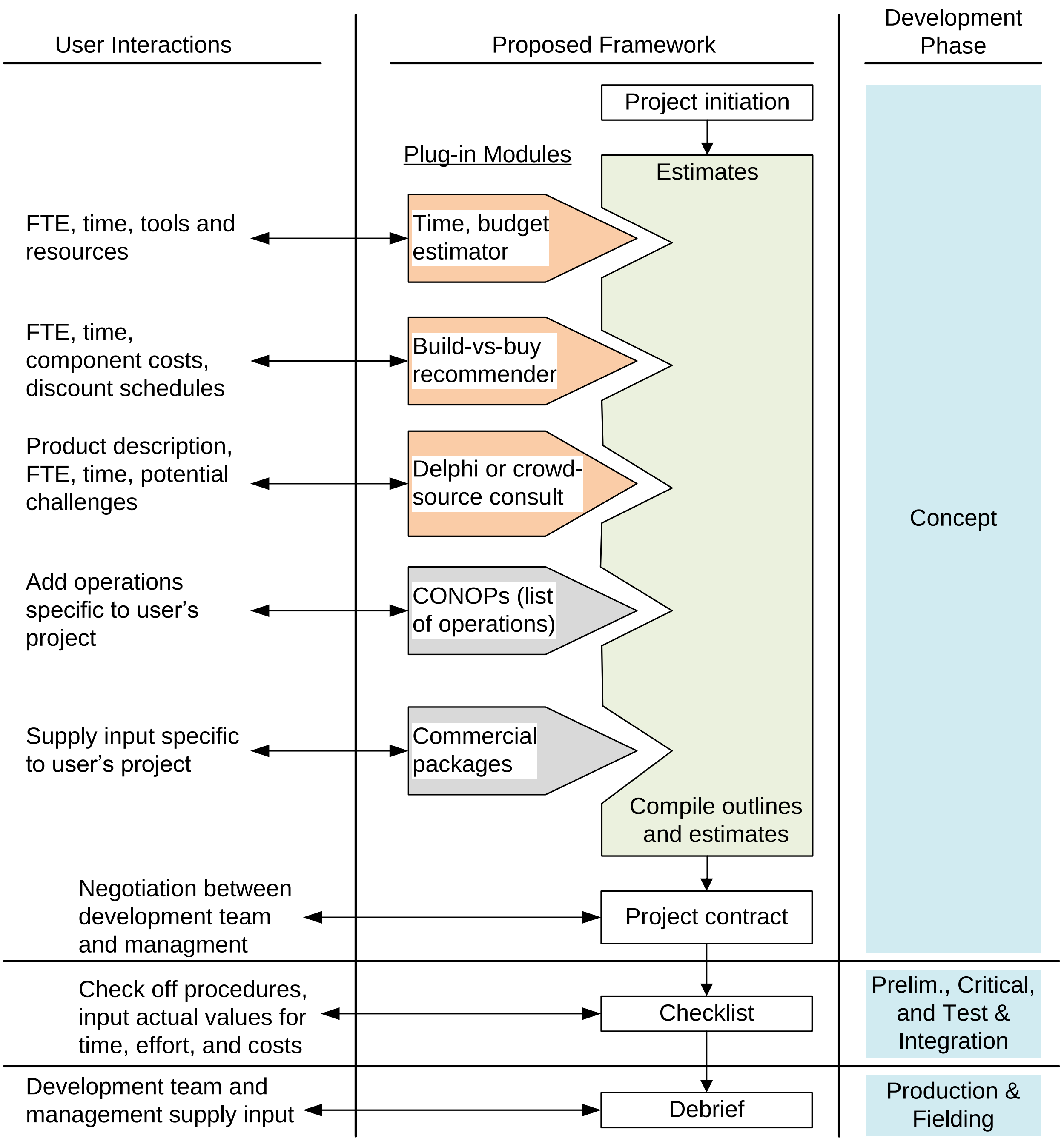

User Interactions
Proposed Framework
Development Phase
Project initiation
Plug-in Modules
Estimates
FTE, time, tools and resources
Time, budget estimator
FTE, time, component costs, discount schedules
Build-vs-buy recommender
Product description, FTE, time, potential challenges
Delphi or crowd-source consult
Add operations specific to user's project
CONOPs (list of operations)
Supply input specific to user's project
Commercial packages
Compile outlines and estimates
Concept
Negotiation between development team and managment
Project contract
Check off procedures, input actual values for time, effort, and costs
Checklist
Prelim., Critical, and Test & Integration
Development team and management supply input
Debrief
Production & Fielding

Figure 3.1 Outline of the proposed framework with plug-in modules suggests features, schedules and budgets to project teams during the development of embedded systems.

Operation

Users of the Framework

The framework could be used by anyone under the assumption that most people using the framework will already have technical expertise to develop some aspect of embedded systems.

What the Framework Provides

The framework provides a baseline set of estimates for time, effort and potential challenges for developing a new embedded system. The set of estimates are minimal values or constraints to implement all the known, good practices to develop embedded systems. Also, one plug-in module can help a user quickly analyze the build-versus-buy decision for a specific subsystem.

In the future, the framework might accept plug-ins that provide various sets of information, such as reference designs drawn from concepts-of-operation, templates of contracts, and links to commercial packages.

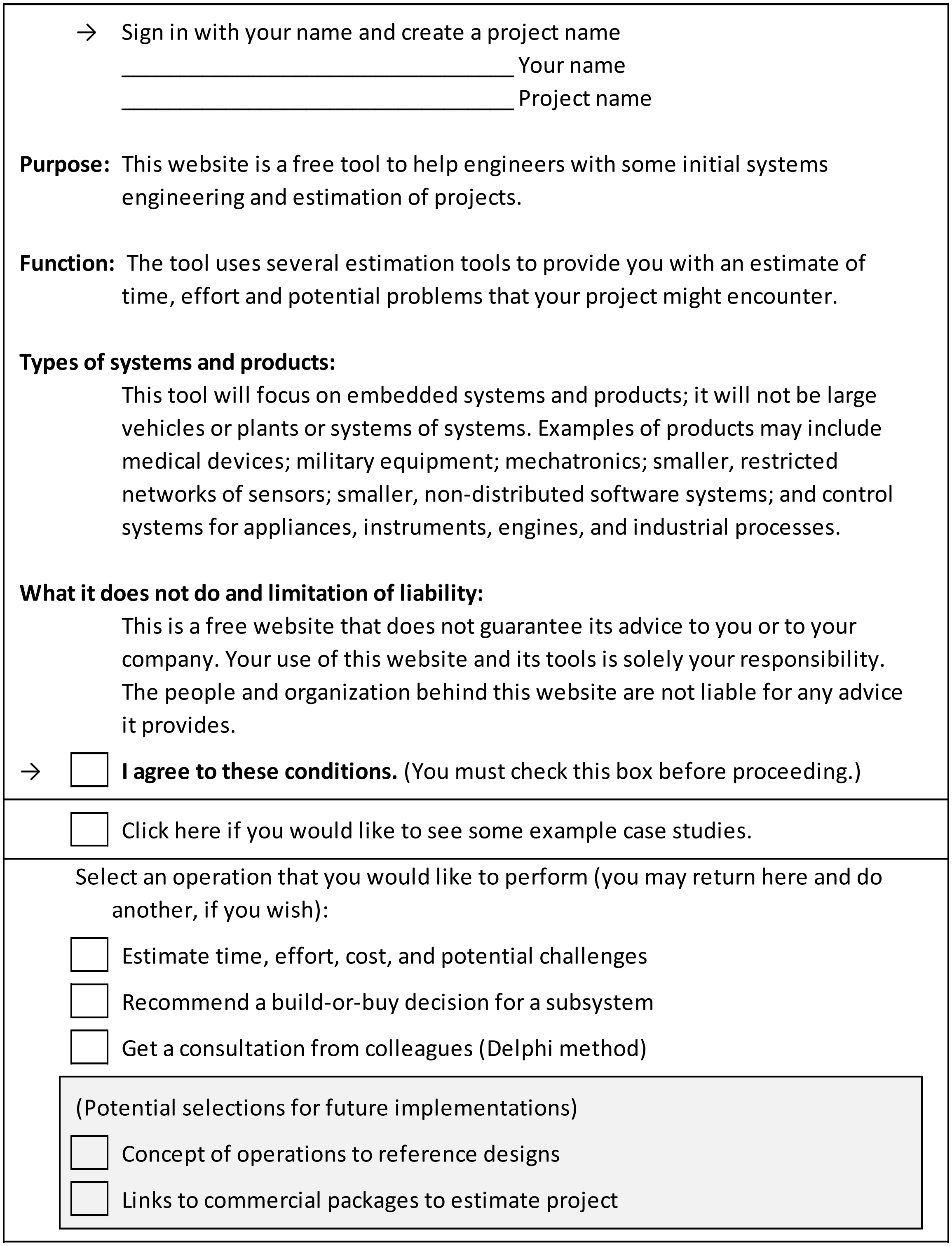
Implementation of the Framework

The framework can be implemented several different ways. It could be a collection of spreadsheets that a user downloads from the internet. It could be a software package delivered online (or on CD, but this defeats the purpose of fast delivery). It could be a website that has software-screen buttons and simple fill-in boxes.

Figure 3.2 illustrates a potential implementation of the opening webpage on a website. NOTE: This example page requires the user to agree to the limitation of liability before moving forward with the rest of the framework.

After agreeing to the liability limitation, the user may view a selection of case studies for time and effort from industry sources [200]. Table 3.1 illustrates one case study for a clothes-washing machine. Other case studies may be found in Appendix D.

The user may then choose to select one of three procedures. Each is operated by a plug-in module, which should operate seamlessly within the framework, as each is independent of the others. Each plug-in module is explained in detail in one of the next three chapters; potential webpage operations for each plug-in module will also be explained in those chapters. Additional plug-in modules may be added in the future (see the gray box in Figure 3.2).



→ Sign in with your name and create a project name

_________________________________ Your name

_________________________________ Project name

Purpose: This website is a free tool to help engineers with some initial systems engineering and estimation of projects.

Function: The tool uses several estimation tools to provide you with an estimate of time, effort and potential problems that your project might encounter.

Types of systems and products:

This tool will focus on embedded systems and products; it will not be large vehicles or plants or systems of systems. Examples of products may include medical devices; military equipment; mechatronics; smaller, restricted networks of sensors; smaller, non-distributed software systems; and control systems for appliances, instruments, engines, and industrial processes.

What it does not do and limitation of liability:

This is a free website that does not guarantee its advice to you or to your company. Your use of this website and its tools is solely your responsibility. The people and organization behind this website are not liable for any advice it provides.

→ □ **I agree to these conditions.** (You must check this box before proceeding.)

□ Click here if you would like to see some example case studies.

Select an operation that you would like to perform (you may return here and do another, if you wish):

□ Estimate time, effort, cost, and potential challenges

□ Recommend a build-or-buy decision for a subsystem

□ Get a consultation from colleagues (Delphi method)

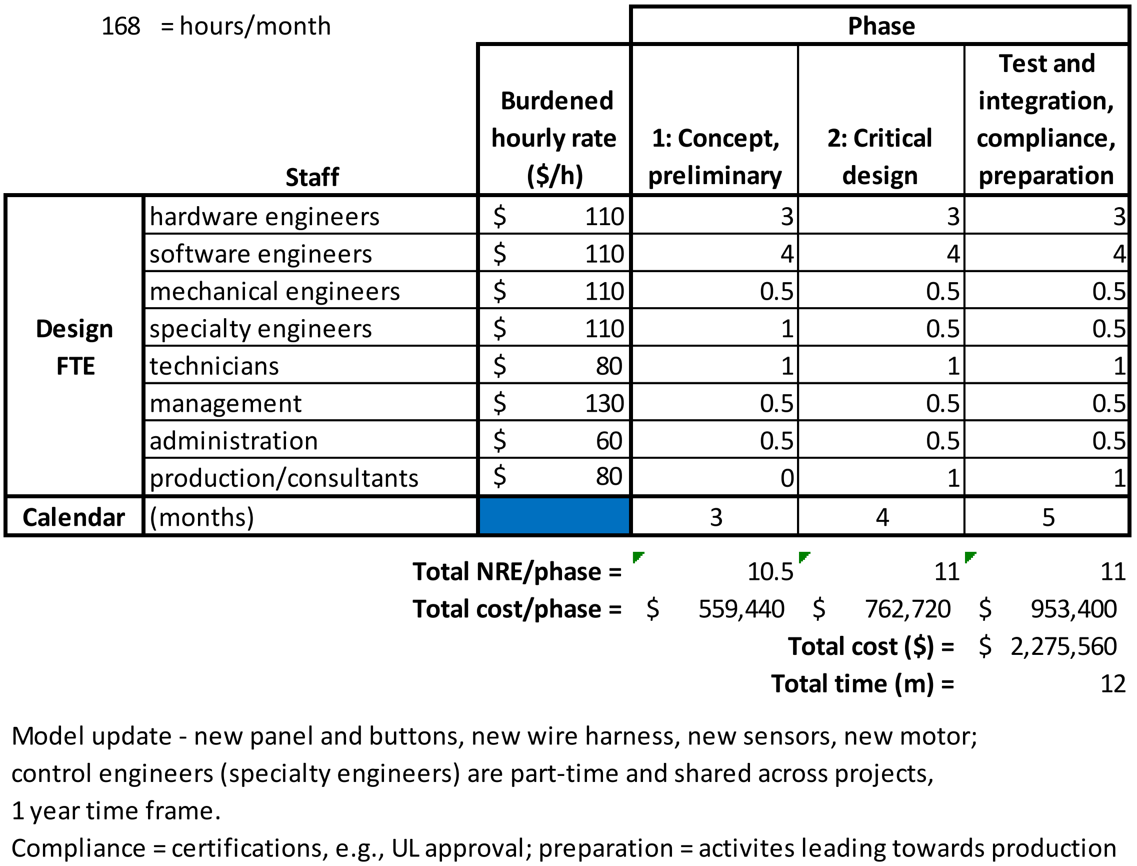
(Potential selections for future implementations)

□ Concept of operations to reference designs

□ Links to commercial packages to estimate project

Figure 3.2 Potential opening page or pages to the framework. The limitation of liability must be agreed to by the user before moving forward with the selections.

Table 3.1 An Example Case Study for Developing an Embedded System for a Clothes-Washing Machine.



168 = hours/month

			Phase		
	Staff	Burdened hourly rate ($/h)	1: Concept, preliminary	2: Critical design	Test and integration, compliance, preparation
Design FTE	hardware engineers	$ 110	3	3	3
	software engineers	$ 110	4	4	4
	mechanical engineers	$ 110	0.5	0.5	0.5
	specialty engineers	$ 110	1	0.5	0.5
	technicians	$ 80	1	1	1
	management	$ 130	0.5	0.5	0.5
	administration	$ 60	0.5	0.5	0.5
	production/consultants	$ 80	0	1	1
Calendar	(months)	🟦	3	4	5

Total NRE/phase =	10.5	11	11
Total cost/phase =	$ 559,440	$ 762,720	$ 953,400
		Total cost ($) =	$ 2,275,560
		Total time (m) =	12

Model update - new panel and buttons, new wire harness, new sensors, new motor;
control engineers (specialty engineers) are part-time and shared across projects,
1 year time frame.
Compliance = certifications, e.g., UL approval; preparation = activites leading towards production

The framework can take the output results from the plug-ins and generate a checklist. This can be a separate plug-in module. Appendix D has eight more case studies in the same format.

Once the checklist is generated (see Appendix C), it can be the basis for negotiating a project contract between management and the development team. The contract negotiation follows the next paragraph.

The team can then execute the procedures in the checklist as it completes them and sign off each line item procedure, which would normally happen after a review agrees that the line item has indeed been accomplished.

Contract between Teams and Management

I propose that development teams and company management follow contractual agreements for technical performance based on these checklists and plans suggested in the previous section. The contracts can attach the checklists and plans as part of the agreements. Contracts can be templates held in the tool’s database and called up after the checklists and plans are selected; contract templates are beyond the scope of this work.

Contracts should define success, schedule, budget, and features of the product. They should also specify development processes to be followed and the expected outcomes. Additionally, contracts should be part of the gateway design reviews, such as the conceptual design review, preliminary design review, critical design review, and production-handoff review. During a gateway design review, the team should examine the progress on the contract for how closely the team and management are adhering to the original intent. Should issues arise, such as with the market or client intent or technology or difficulties, the team can agree to revise the contract to suit the needs (i.e., the contract should be flexible [142]).

Need for Thorough, Unbiased Mediation

Dictionary.com defines bias as, “a particular tendency, trend, inclination, feeling, or opinion . . .” This means that “unbiased” is the ability to avoid bias, or the tendency, feeling, or opinion to believe something. As used herein, “unbiased” is a psychological term, not a statistical term; the word means “without preconceived notions or beliefs.”

Complete checklists, plans, and contracts require an unbiased mediator (one without preconceived notions or beliefs) to handle the contract negotiations between management and teams that represent design, development, production, marketing, sales, and support. A mediator is important to bring structure, even-handedness, and exposure to issues not considered previously by the various teams. On p. 216 of his book [30] Ariely states, “If we acknowledge that we are trapped within our perspective, which partially blinds us to the truth, we may be able to accept the idea that conflicts generally require a neutral third party – who has not been tainted with our expectations – to set down rules and regulations. Of course, accepting the word of a third party is not easy and not always possible; but when it is possible, it can yield substantial benefits. And for that reason alone, we must continue to try.”

In addition to being unbiased, a good mediator must be thorough. Being unbiased will reduce ignoring some issues and placing too much emphasis on some issues at the expenses of others that are equally important, whereas thoroughness will help remove or reduce the random and unobservable actions by exposing issues.

To be thorough and unbiased, a mediator should uncover issues that occur within the product lifecycle, which includes design, development, production, marketing, sales, and support. Template checklists can be a basis for thoroughness. Regular and independent updates to the checklists can maintain their utility in uncovering issues and still reduce bias. Furthermore, to be thorough and unbiased, a mediator should cover at least five different arenas during the product lifecycle: technical, business, personal, societal, and cultural.

Mediation

Mediation levels the playing field by exposing issues. A mediator could be either a human or an expert system that handles the checklist/plan development and the contract negotiations. As well as being unbiased and thorough, a mediator must be knowledgeable and consistent. By revealing issues and exposing all parties to these issues, a mediator can suggest more equitable contracts that have greater assurance of success. The more a mediator knows and reveals to all parties, the more likely negotiations will be equitable for all parties and result in higher assurance of success for the project or product. This reduces the random and unobservable actions that contract theory attempts to address.

Finally, a mediator also needs to include any personal and cultural issues in negotiations. Specifically, a mediator should connect personal motivations and cultural deference into the rewards that accompany any definition of success.

Estimating during Mediation

As part of the mediation to reveal issues, a mediator could provide estimations of specific quantities, such as amount of effort, schedule of calendar time, budget, and potential obstacles. As mentioned in the literature review, Valerdi (2005), Ilseng (2006), and Holmes (2012) all outlined the same six estimation methods [163]–[165]. A mediator could use all six; however, focus here will be only on the bottom-up/activity-based method and on the expert judgment/guesstimates.

A potential facet of this mediation would be to call on the technical community to vote on approaches and to estimate quantities in an independent fashion. The Delphi method and crowdsourcing both could potentially fill this role; a technical community could be the diverse and independent voice to provide confirmation of approach and quantities. These two methods follow closely “Reference Class Forecasting” from Kahneman p. 251 [24]:

- Identify a reference class
- Obtain statistics on this reference class
- Generate a baseline prediction
- Use specific case information to adjust from this baseline.

Both methods, crowdsourcing and the Delphi method, can be unbiased and potentially thorough (under the assumption that each problem is posed carefully). A good mediator can consistently pose problems and format solutions that are both thorough and unbiased.

One concern is the motivation for participants to contribute to crowdsourcing or the Delphi method consultation. Anyone wanting these services would need to provide some sort of incentive for participants in either. This could be an entire field of study itself; examples of motivational incentives could include the following: monetary, such as consulting fees, or gift cards for randomly selected participants (which were used in the crowdsourcing survey and the Delphi method in this dissertation), or recognition of effort, or receiving a custom report of the final results.

Archiving and Debriefing during Mediation

A mediator should record and archive all inputs and provide an auditable trace or timeline of the activities. A mediator should also provide a basis for debriefing after the product or project has been developed and delivered. Debriefing provides feedback and the opportunity to learn from the completed project. The results from debriefing can then be incorporated in updating plans and checklists.

Validity

The development and use of experimental results in the proposed framework are subject to reviews of validity. According to [201], “The results are said to have adequate validity if they are valid for the population to which we would like to generalize.” There are four general types of validity: conclusion, internal, construct, and external. Each type of validity has a specific threat. Figure 3.3, at the end of this chapter, shows the generalized flow of data through the system.

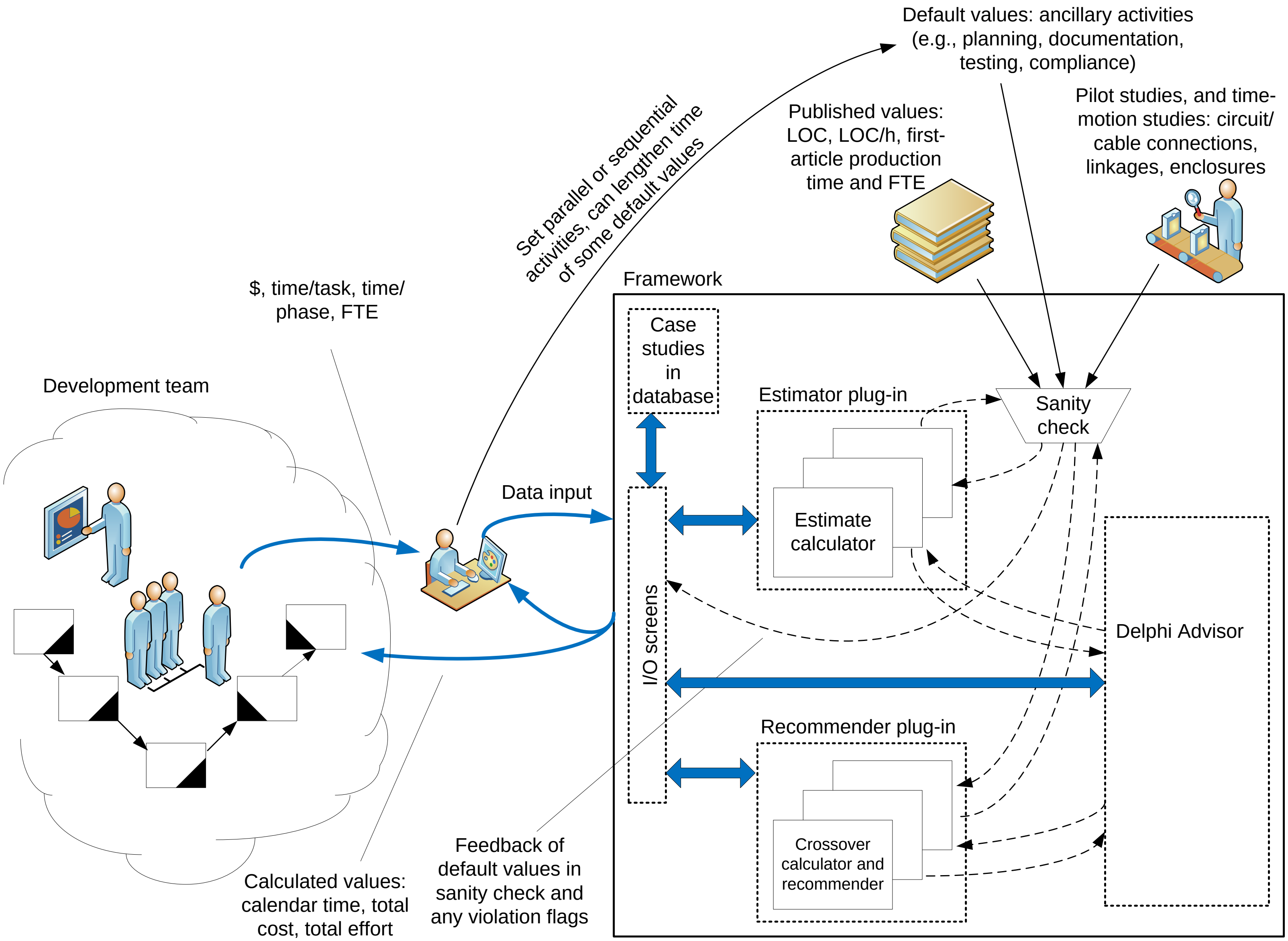

Default values: ancillary activities (e.g., planning, documentation, testing, compliance)
Published values: LOC, LOC/h, first-article production time and FTE
Pilot studies, and time-motion studies: circuit/ cable connections, linkages, enclosures
Set parallel or sequential activities, can lengthen time of some default values
$, time/task, time/ phase, FTE
Framework
Development team
Case studies in database
Estimator plug-in
Sanity check
Data input
Estimate calculator
I/O screens
Delphi Advisor
Recommender plug-in
Crossover calculator and recommender
Feedback of default values in sanity check and any violation flags
Calculated values: calendar time, total cost, total effort

Figure 3.3 Data flow from and to users through the framework and three plug-ins described in this work.

Each plug-in module to the framework is affected to some degree by each of these threats. The chapter for each plug-in module describes how these threats to validity affect that specific module.

Threats to Conclusion Validity

These threats affect the ability to draw a correct conclusion between the independent variable (or treatment) and the dependent variable (or outcome). These threats have to do with the appropriate selection of experiment parameters, implementation, and sample sizes [201]. Examples include low statistical power of the test, violated assumptions, reliability of measures such as poor wording, reliability of implementation, and disturbance of the results caused by random elements outside the experiment.

One example apropos to this work is that of using lines of code (LOC) rather than function points to reduce the amount of human judgment required of the user of the framework. Another example fitting this work is that of using the number of wired connections rather than circuit board complexity to estimate time in design; counting wired connections is simpler and requires much less human judgment required of the user of the framework. A third example is to highlight and stress very clearly that the estimated results are the minimum values for completely successful projects with no requirements changes along the way; the results are *<u>not expected values</u>*, but *<u>boundary constraints no project can transgress</u>*.

Threats to Internal Validity

These threats can skew or alter the perspective on the actual relationship between the independent variable (or treatment) and the dependent variable (or outcome) by indicating a casual relation different than actual [201]. Examples include bad instrumentation (e.g., poorly worded data-collection forms), statistical regression toward the mean, selection of volunteers who do not represent the population, and maturation of subjects with progress of the experiment.

For users of this framework, two examples of threats to internal validity exist. One threat is that of poorly worded instructions in the framework. The other threat is maturation of users while using the framework multiple times for different projects; they may learn to “game” the tools to give results that look good to them but do not bear a semblance to reality.

Threats to Construct Validity

These threats can skew or alter the perspective on the actual relationship between the experiment setting and the actual setting under study [201]. There can be design threats such as unclear theory, which does not translate well into measures, or mono-method bias in which a single type of measure does not have cross-checks to prevent inappropriate interpretations, or confounding constructs, such as users with different experience levels, or interaction of various treatments.

For users of this framework, two examples of threats to construct validity exist. One threat is that of relying on a mono-method, specifically, a single type of form to estimate effort, time and cost. A cross-check to the mono-method of estimation is to have a separate form with different parameters to provide a "sanity check" of the first estimator. The other threat is that of users with various experience levels; again, the separate, "sanity-check" form will nudge users towards a more likely result, in spite of different experience levels.

Threats to External Validity

These threats can alter the generalization of the experimental results to industrial practice [201]. Examples include that of the wrong people participating in the experiment, or conducting experiments on "toy" problems, or interactions between history and treatment of the experiment.

For users of this framework, two examples of threats to external validity exist. One threat is that of the wrong people using the framework and supplying inappropriate or incorrect data. The other threat concerns basing framework operations on bad or "toy" case studies.

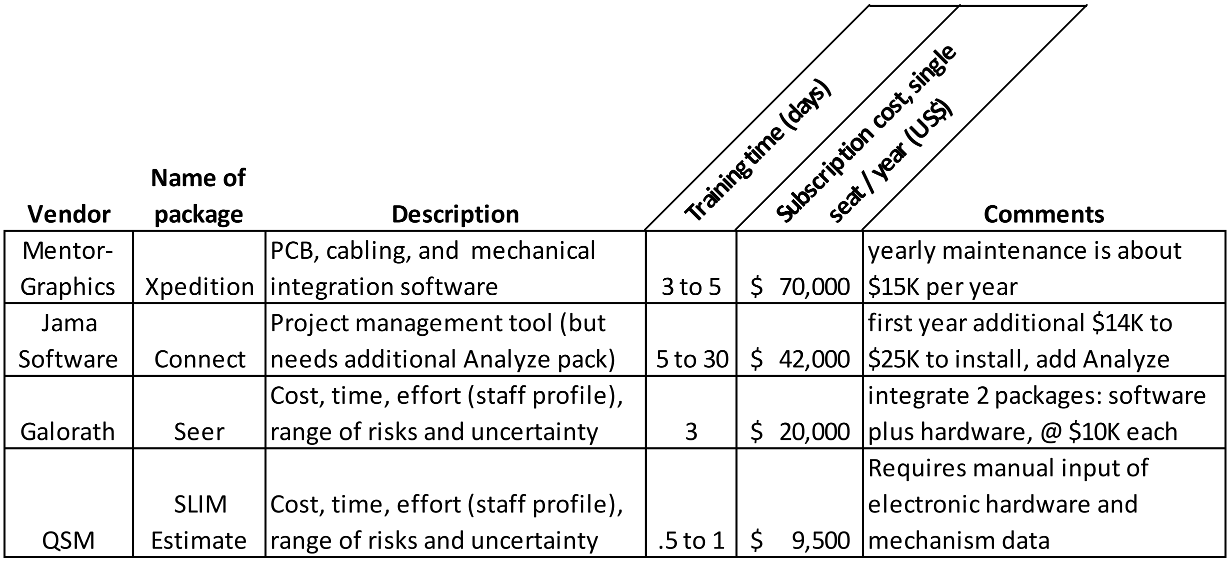
Comparison with Available Packages

This framework sets itself apart from all other commercial or published estimation packages by providing a "sanity check" comparison for every estimation that the user attempts. The "sanity check" provides baseline values for time, effort and cost when developing embedded systems from a clean sheet. These baseline values are boundary constraints for projects; they are minimum values of time, effort and cost that can ever be expected for the described projects. No project will achieve lower values than those offered by the framework. These are boundary values, not expected estimates of values with ranges of uncertainty. They establish that no project will better their values; all projects will exceed these values. In other words, the "sanity check” in this framework is the baseline constraint for a project development. The “sanity check” causes the framework to flag an estimation violation when a user underestimates a project.

This framework is not reinvention of what others have done and it is not a small version of a commercial package. An extensive literature review uncovered no available tools or packages that do what this framework does with its “sanity check” nor tools that are as inexpensive or easy to use as this framework. Table 3.2 lists three commercial packages that can estimate time, effort and costs with ranges of risks and uncertainties.

The distinctives of this framework is that that it has parallel paths to “sanity check” estimations, it costs little or nothing to use, and it does not require learning any new packages or techniques. Users merely click buttons or fill in boxes with numbers or short descriptions in text boxes; the operations are similar to those in a webpage on a website or a simple spreadsheet. Using the tool requires no learning curve. Moreover, it gives immediate answers. In the future, it will even accept other packages, such as those in Table 3.2, as plug-in modules. Because the framework only gives boundary values to estimates, it does not give expected results or ranges of uncertainties.

Table 3.2 Several Commercial Packages that Provide Estimates of Time, Effort, Costs, and Ranges of Risks and Uncertainties, but Do Not Flag Violations of Baseline Constraints.



Vendor	Name of package	Description	Training time (days)	Subscription cost, single seat / year (US$)	Comments
Mentor-Graphics	Xpedition	PCB, cabling, and mechanical integration software	3 to 5	$ 70,000	yearly maintenance is about $15K per year
Jama Software	Connect	Project management tool (but needs additional Analyze pack)	5 to 30	$ 42,000	first year additional $14K to $25K to install, add Analyze
Galorath	Seer	Cost, time, effort (staff profile), range of risks and uncertainty	3	$ 20,000	integrate 2 packages: software plus hardware, @ $10K each
QSM	SLIM Estimate	Cost, time, effort (staff profile), range of risks and uncertainty	.5 to 1	$ 9,500	Requires manual input of electronic hardware and mechanism data

Commercial packages, however, do give mean estimates of time, effort and cost with ranges of risk and uncertainty. The downsides are that the packages are expensive, especially for small companies, they require significant investment in training time and installation, and they require histories of previous projects for places where their databases do not information, such as electronics or mechanics. These packages require a user to supply data from previous projects to generate a history of interactions between software, electronics and mechanics. My plug-ins do not require users to input history of previous projects; my plug-ins have default values built in to "sanity check" user estimates.

Novelty

Chapter 1 gives the justification for novelty. One note about this proposed framework: the numerical values representing the design/development times that accompany the number of circuit connections, the number of mechanical connections, and the number of enclosures is a basic placeholder at this writing. These parameters need time-motion studies and research to update their values with confidence intervals.

Commercial tools, published methods and this framework all address the *Practice Standard for Project Estimating* that aligns with *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)* [202], [203]. A distinctive of this framework is that it gives immediate feedback for violations of unrealistically low estimates for time, effort and cost. The commercial tools and published methods provide mean estimated values and risk uncertainties that derive from accumulated data in databases, which is valuable, but they do not flag a hard constraint.

The proposed framework is a set of guidelines. It is as easy to use as a checklist – it encourages teamwork, discipline, and does not require new skills, which is not necessarily true for many engineering tools. In addition, it can be iterative and potentially non-sequential to fit individual applications. A mediator tool can be a webpage or a computer dashboard; it can even be paper-based if a company wants it secure from potential web hacking.

The proposed framework could be developed to be psychologically unbiased with pre-prepared checklists. A professional organization, such as a technical society, could potentially host a web tool that provides the unbiased consultation. One caveat is that psychological bias could creep into processes as checklists are modified within and by the design/development team (e.g., important activities could be eliminated that are necessary to design assurance or extraneous activities, such as adding unnecessary features, could move the design and development in the wrong direction – these would be threats to internal validity and construct validity).

Future Work

To access the construct validity, the framework needs pilot studies to test its claim of speed and no requirement for learning. Pilot studies will also help access external validity for providing generalized baseline studies to users.

Chapter 4 - Plug-in Estimator for Effort, Time and Challenges

Purpose

The first step in most projects is to estimate baseline values for budget and time to design, development, review, test, and integration before full-scale production. (Once these are set, then a team moves on to complete the requirements.) Models and software packages exist to aid estimation. Furthermore, management can rely on experience to build heuristics and to guide estimation. Unfortunately, while help is available, development teams and management still fail to meet deadlines, budgets, and specifications for new products. Quinnell's surveys show that more than 60% of all embedded projects miss their deadlines [10]; since time is money, most projects run over budget, as well. Quinnell's surveys also show that many projects reuse portions of previous designs; specifically, "36% [of projects] purchase rather than developing their own hardware, 71% re-use some of their hardware designs, and 86% re-use some part of their previous code" [10]. Even with most projects attempting to leverage previous work, they still run over schedule and budget. Thus, no matter whether a "clean-sheet" design or a reused and modified previous design is undertaken, most teams do not meet their estimates.

There are four basic aspects of systems development that pose major challenges to developing estimates of system development. The first challenge is the properties of systems that directly affect development; these include scale (number of functions provided), complexity and managing interactions between subsystems and between the system and its environment, dependability (including reliability, availability, robustness, and maintainability), and mission-assurance. The second challenge is the capability of the development team; do they have the skills and experience necessary for the project? The third challenge is the availability of resources to the team: tools, facilities, and materials. The forth challenge is understanding the stakeholders and their contributions to the design and requirements within the project. Commercial design tools address these challenges by drawing on databases of completed projects, which account for these challenges in some measure; more projects in the database should provide a higher degree of confidence in these tools, assuming there is sufficient diversity in the different types of projects in the database. Several open-source tools, such as COCOMO and COSYSMO, are also available and are indeed useful [155], [204], [205]. Several downsides to these tools, whether commercial or open-source, are that they require significant time to learn, cost large amounts of money to license, and some detailed assessments of the project, such as complexity, team capabilities, and technology maturity.

The purpose of this plug-in module in the proposed framework is to provide users of the framework with a baseline set of boundary constraints on time, effort, cost and challenges; these would be “best-case” constraints, meaning that most teams could not improve on those values. A second purpose is to make a tool readily available at little or no cost and one that does not require a lot time to learn. A third purpose is to minimize the subjective estimation of complexity, team capabilities, and technology maturity.

The users will approach the module with some concept of design already in mind, which is true for using any other tool, as well. The module will ask users to supply the minimum-maximum ranges of a few parameters, such as lines of code and number of connections. It will also ask users to review and confirm default values already built into the module. From these user inputs and the default values already in the module, it will calculate and present a baseline set of boundary constraints on time, effort, cost and challenges. The module has several different levels of detail, each cross-checked by a “sanity-checker,” and users may revisit any level at any time to revise the baseline values for their planning purposes.

Operation of Estimator

Users of the Framework

The framework could be used by anyone under the assumption that most people using the framework will already have technical expertise to develop some aspect of embedded systems. The most likely users will be project leads or system engineers.

What the Framework Provides

The framework provides baseline constraints for time, effort and potential challenges for developing a new embedded system. This set of constraints are minimal values to implement all the known, good practices to develop embedded systems. Planning to use less time or effort will mean skipping some known good practices, which both the literature review and the simulation in Appendix B have shown will guarantee latent errors or faults in an embedded system.

Implementation of the Estimator

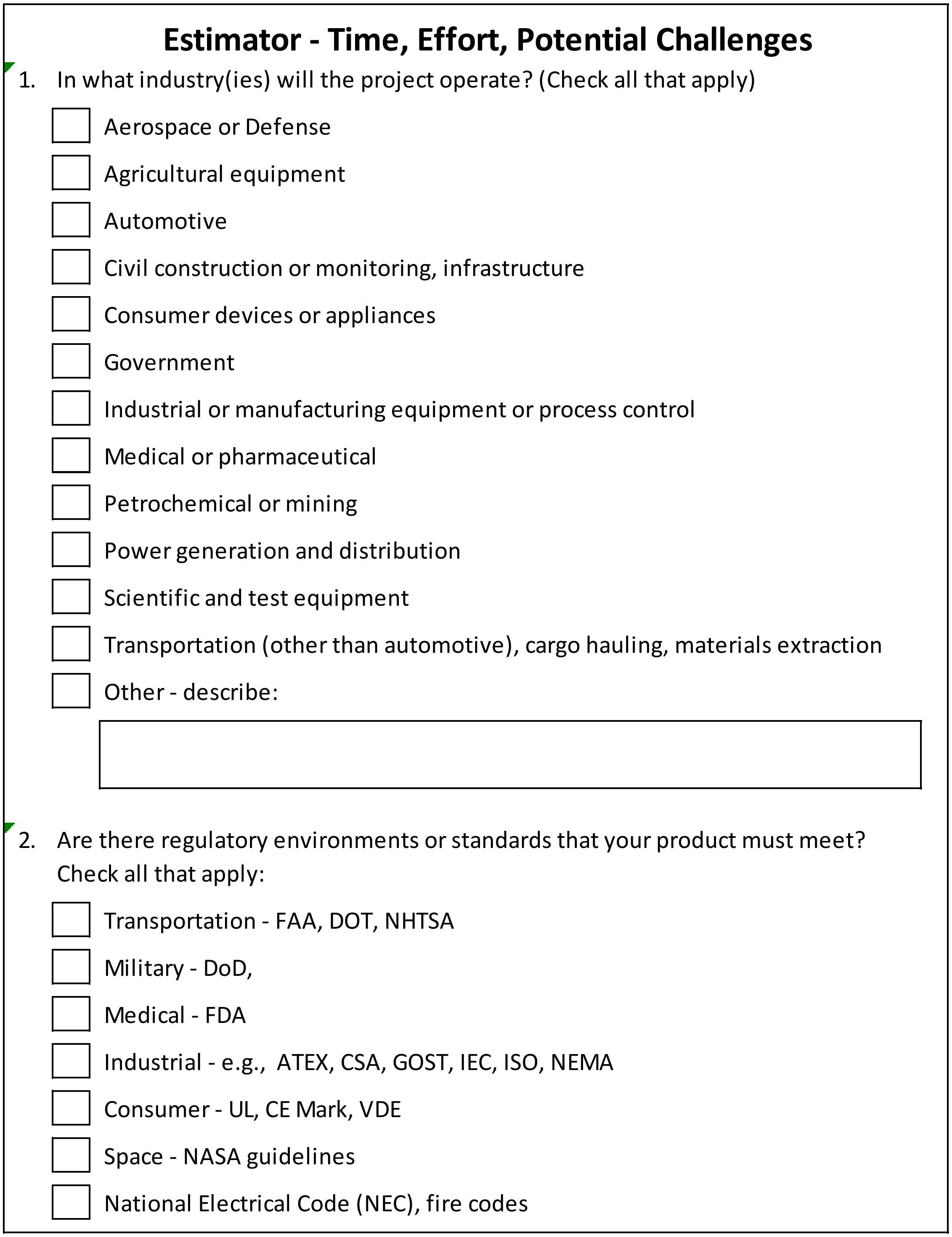
The opening webpages on a website for the estimator plug-in ask for some basic industry information, such as standards, and the expected environment. This information will establish which template to select for the estimator: commercial, industrial, military, aerospace, medical, or extreme environments.

It requires the user to fill-in the sanity check first; other options are either not shown or "grayed out" to prevent their selection. After filling in the sanity check, the user may select one of several options: Level 1, Level 2, or Level 3. The levels correspond to amount of detail that user may supply. The higher the level number the more likely the estimator will "nudge" the user towards realistic numbers in their inputs to the estimator. The sanity check will then compare their final values with its baseline constraints; its purpose is to help users reduce their psychological bias to the planning fallacy.

Implementation of the Estimator's Opening Pages

Figure 4.1 through Figure 4.4 illustrate a potential implementation of the opening webpages on a website for the estimator plug-in. It asks for some basic industry information, such as standards, and the expected environment; these values will help the estimator to choose a template, stored in the tool's database, for the sanity checker to fix boundary constraints.

Note: these templates are not within the scope of this dissertation, but they are reserved for future research. Appendix C provides some suggested plans and checklists; future research can modify them to provide default values to the sanity checker.



Estimator - Time, Effort, Potential Challenges

1. In what industry(ies) will the project operate? (Check all that apply)

□ Aerospace or Defense

□ Agricultural equipment

□ Automotive

□ Civil construction or monitoring, infrastructure

□ Consumer devices or appliances

□ Government

□ Industrial or manufacturing equipment or process control

□ Medical or pharmaceutical

□ Petrochemical or mining

□ Power generation and distribution

□ Scientific and test equipment

□ Transportation (other than automotive), cargo hauling, materials extraction

□ Other - describe:

2. Are there regulatory environments or standards that your product must meet? Check all that apply:

□ Transportation - FAA, DOT, NHTSA

□ Military - DoD,

□ Medical - FDA

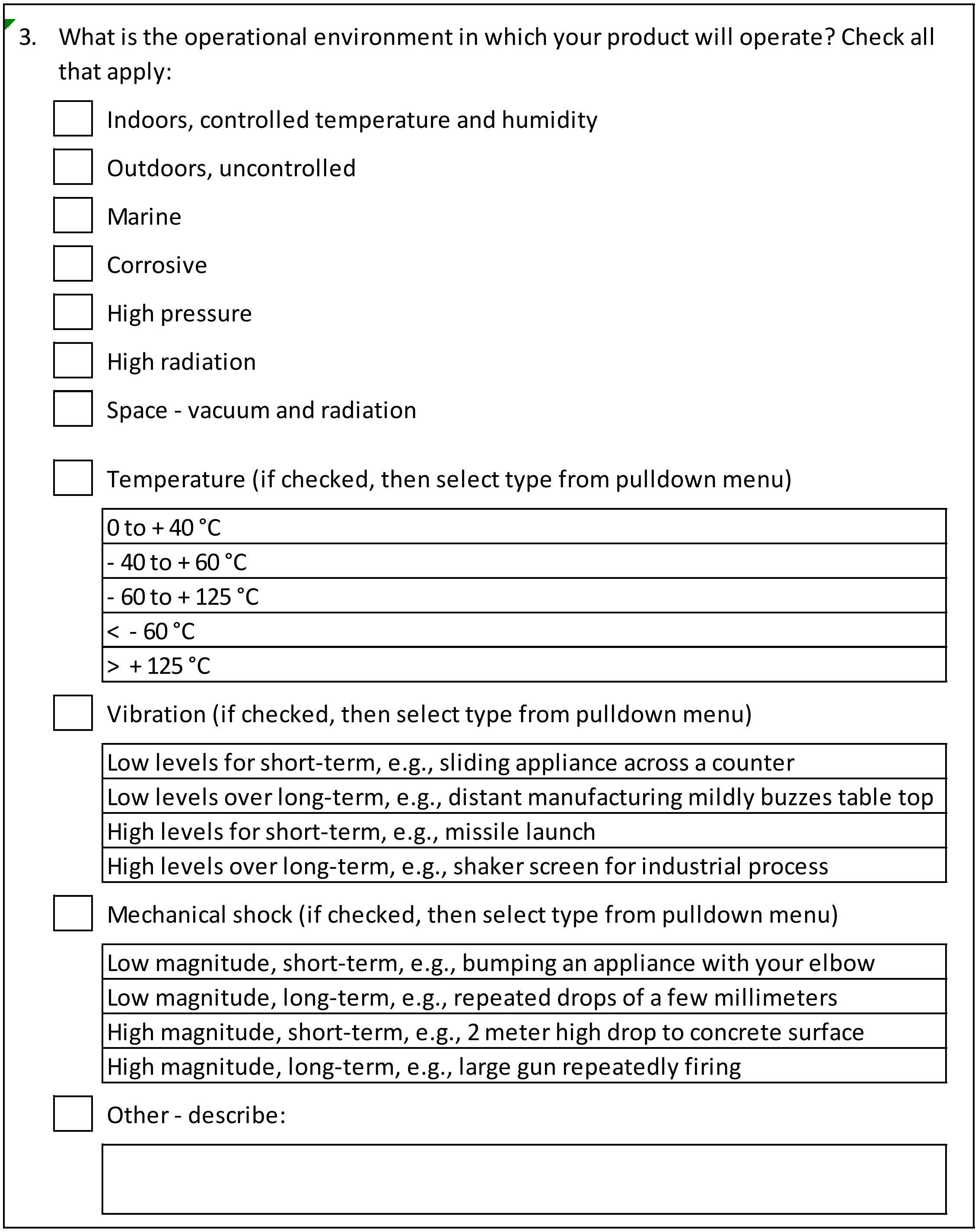
□ Industrial - e.g., ATEX, CSA, GOST, IEC, ISO, NEMA

□ Consumer - UL, CE Mark, VDE

□ Space - NASA guidelines

□ National Electrical Code (NEC), fire codes

Figure 4.1 Potential opening page to the estimator plug-in. It will help the program choose the appropriate template to use in establishing the baseline constraints.



3. What is the operational environment in which your product will operate? Check all that apply:

□ Indoors, controlled temperature and humidity

□ Outdoors, uncontrolled

□ Marine

□ Corrosive

□ High pressure

□ High radiation

□ Space - vacuum and radiation

□ Temperature (if checked, then select type from pulldown menu)

0 to + 40 °C
- 40 to + 60 °C
- 60 to + 125 °C
< - 60 °C
> + 125 °C

□ Vibration (if checked, then select type from pulldown menu)

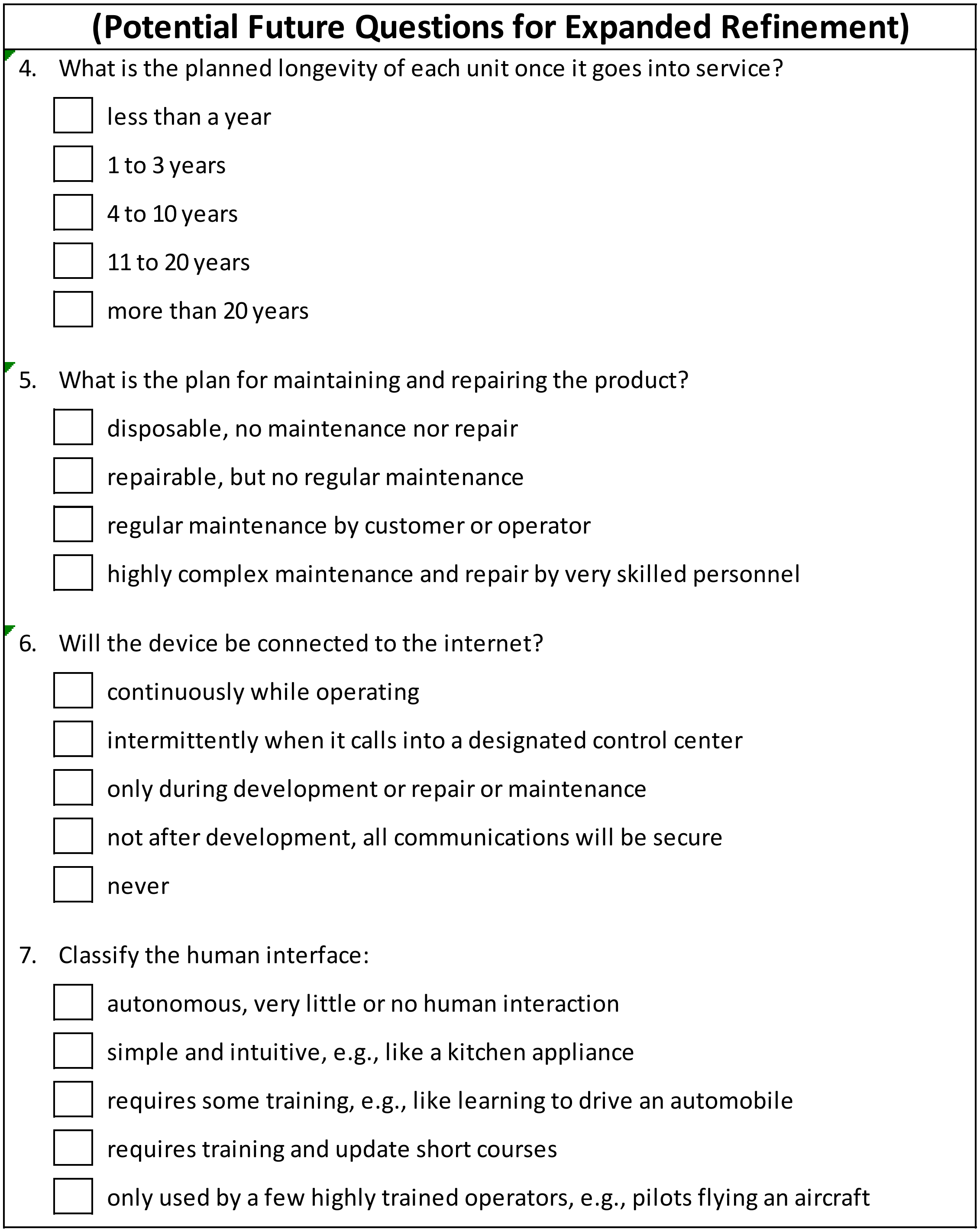
Low levels for short-term, e.g., sliding appliance across a counter
Low levels over long-term, e.g., distant manufacturing mildly buzzes table top
High levels for short-term, e.g., missile launch
High levels over long-term, e.g., shaker screen for industrial process

□ Mechanical shock (if checked, then select type from pulldown menu)

Low magnitude, short-term, e.g., bumping an appliance with your elbow
Low magnitude, long-term, e.g., repeated drops of a few millimeters
High magnitude, short-term, e.g., 2 meter high drop to concrete surface
High magnitude, long-term, e.g., large gun repeatedly firing

□ Other - describe:

Figure 4.2 Potential second page to the estimator plug-in to help establish which templates to use.



(Potential Future Questions for Expanded Refinement)

4. What is the planned longevity of each unit once it goes into service?

   □ less than a year

   □ 1 to 3 years

   □ 4 to 10 years

   □ 11 to 20 years

   □ more than 20 years

5. What is the plan for maintaining and repairing the product?

   □ disposable, no maintenance nor repair

   □ repairable, but no regular maintenance

   □ regular maintenance by customer or operator

   □ highly complex maintenance and repair by very skilled personnel

6. Will the device be connected to the internet?

   □ continuously while operating

   □ intermittently when it calls into a designated control center

   □ only during development or repair or maintenance

   □ not after development, all communications will be secure

   □ never

7. Classify the human interface:

   □ autonomous, very little or no human interaction

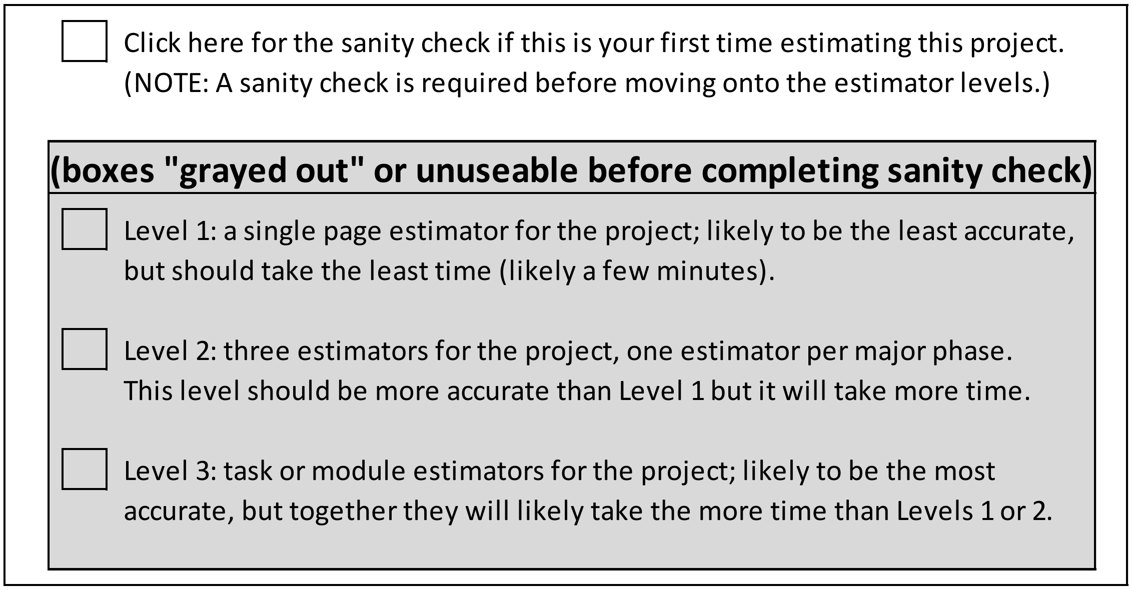
   □ simple and intuitive, e.g., like a kitchen appliance

   □ requires some training, e.g., like learning to drive an automobile

   □ requires training and update short courses

   □ only used by a few highly trained operators, e.g., pilots flying an aircraft

Figure 4.3 Potential, additional information asked by the estimator plug-in to refine the chosen template. This page could be implemented in future versions of the estimator.



□ Click here for the sanity check if this is your first time estimating this project. (NOTE: A sanity check is required before moving onto the estimator levels.)

(boxes "grayed out" or unuseable before completing sanity check)

□ Level 1: a single page estimator for the project; likely to be the least accurate, but should take the least time (likely a few minutes).

□ Level 2: three estimators for the project, one estimator per major phase. This level should be more accurate than Level 1 but it will take more time.

□ Level 3: task or module estimators for the project; likely to be the most accurate, but together they will likely take the more time than Levels 1 or 2.

Figure 4.4 Potential final opening page of the estimator. It requires the user to set up the sanity check before filling in values of time and effort.

Most users should have a good concept of what projects they are planning to develop. Consequently, the first two or three pages of button clicks should take less than 10 seconds apiece to click; the first two pages in Figure 4.1 and Figure 4.2 should take a minute or less to operate. The third page in Figure 4.3 might add another 40 seconds to a minute of time. Summed up, operation of these opening pages should take less than two minutes. A future study, suggested in Appendix E, that times the actual operations would be immediate research opportunity, along with timing the other pages in this chapter. Regardless, these opening operations should be easy and fast to most users.

Setting up the sanity check should be easy and fast, as well (see Figure 4.4). The description of the sanity check follows next.

Implementation of the Sanity Check

The purpose of the sanity check is to help users reduce their psychological bias to the planning fallacy. It provides a “best case” boundary that the particular project cannot better; the “best case” means that the project cannot have a shorter time or less effort or less cost than what the sanity check provides.

The sanity check uses industry-sourced values for the best values of effort and time, which also translates to lowest cost for the features in the project. Where industry-sourced values are not available yet, it provides a rationale for the values it incorporates. Then the sanity check uses readily available numbers that the user supplies; these numbers and values should be more objective and less prone to underestimation. Examples of more objective numbers are the minimum and maximum numbers of circuit boards expected or the minimum and maximum numbers for expected lines of code.

The sanity check should account for project development that is a combination of purchased or reused modules and clean-sheet design. Figure 4.5 illustrates the flow of information to prepare the sanity check. Figure 4.6 and Figure 4.7 show a potential implementation of the webpages for modules or subsystems that are purchased or reused from previous designs and then integrated into the embedded system.

The term FTE, in Figure 4.7, means full-time equivalent, which is the number of people needed to work a task at full-time effort. The FTEs may be fractional, as people divide their time between different tasks. The chart allows the same person or people to be shared between different tasks (many development teams have people perform multiple tasks across disciplines). One person, for instance, might order and then integrate the electronic modules and cables and harnesses into the embedded system.

Another concern that might arise in using the chart in Figure 4.7 is that of modifying an existing design. It raises the concerns of how much regression testing should be performed to verify that the modifications are correct and effective. Consequently, if some design modification of an existing design or product is needed, then the delivery or integration times should reflect the extra time required. The caution, which might be considered a good practice, is that if redesign requires changing more than a small part of the original design, then the user probably should perform a clean-sheet design on the specific subsystem [206]. Future research is needed to define what a “small change to the original design” really means.

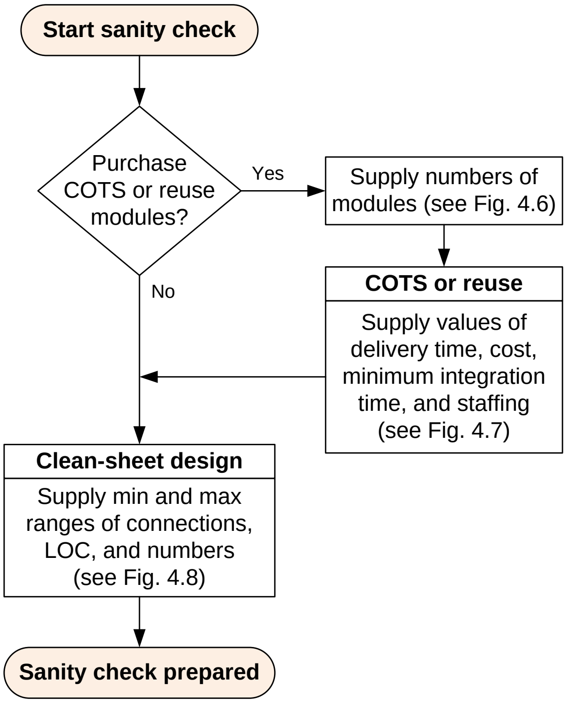
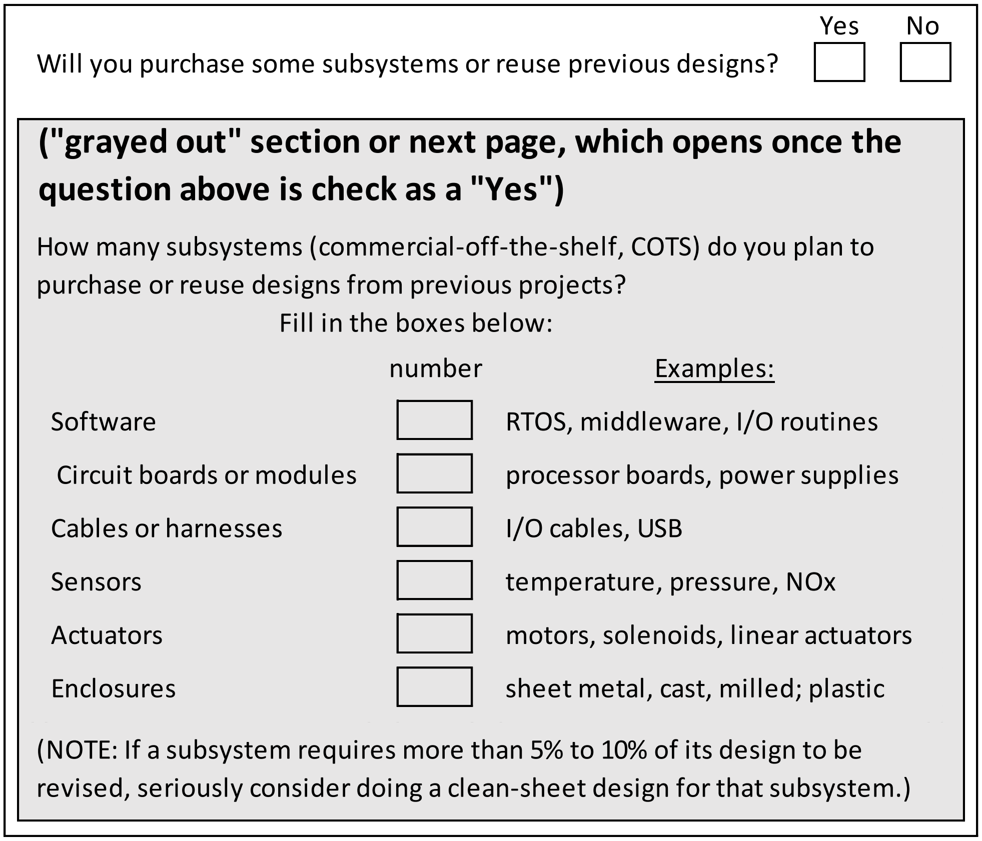

Start sanity check
Purchase COTS or reuse modules?
Yes
Supply numbers of modules (see Fig. 4.6)
COTS or reuse
Supply values of delivery time, cost, minimum integration time, and staffing (see Fig. 4.7)
No
Clean-sheet design
Supply min and max ranges of connections, LOC, and numbers (see Fig. 4.8)
Sanity check prepared

Figure 4.5 Flowchart for preparing the sanity check before running any estimators.



Will you purchase some subsystems or reuse previous designs? Yes □ No □

("grayed out" section or next page, which opens once the question above is check as a "Yes")

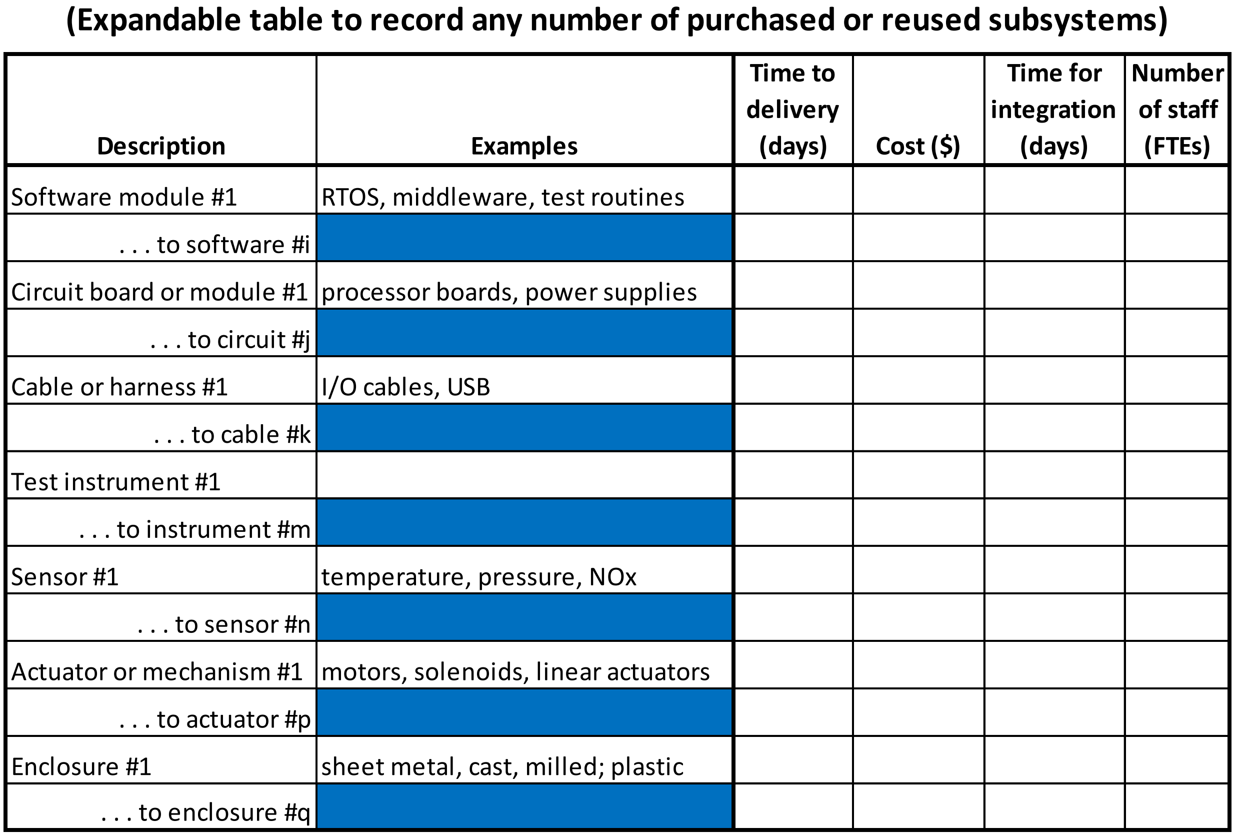
How many subsystems (commercial-off-the-shelf, COTS) do you plan to purchase or reuse designs from previous projects?

Fill in the boxes below:

	number	Examples:
Software	□	RTOS, middleware, I/O routines
Circuit boards or modules	□	processor boards, power supplies
Cables or harnesses	□	I/O cables, USB
Sensors	□	temperature, pressure, NOx
Actuators	□	motors, solenoids, linear actuators
Enclosures	□	sheet metal, cast, milled; plastic

(NOTE: If a subsystem requires more than 5% to 10% of its design to be revised, seriously consider doing a clean-sheet design for that subsystem.)

Figure 4.6 Potential opening page for the sanity check before running any estimators.



(Expandable table to record any number of purchased or reused subsystems)

Description	Examples	Time to delivery (days)	Cost ($)	Time for integration (days)	Number of staff (FTEs)
Software module #1	RTOS, middleware, test routines				
. . . to software #i					
Circuit board or module #1	processor boards, power supplies				
. . . to circuit #j					
Cable or harness #1	I/O cables, USB				
. . . to cable #k					
Test instrument #1					
. . . to instrument #m					
Sensor #1	temperature, pressure, NOx				
. . . to sensor #n					
Actuator or mechanism #1	motors, solenoids, linear actuators				
. . . to actuator #p					
Enclosure #1	sheet metal, cast, milled; plastic				
. . . to enclosure #q					

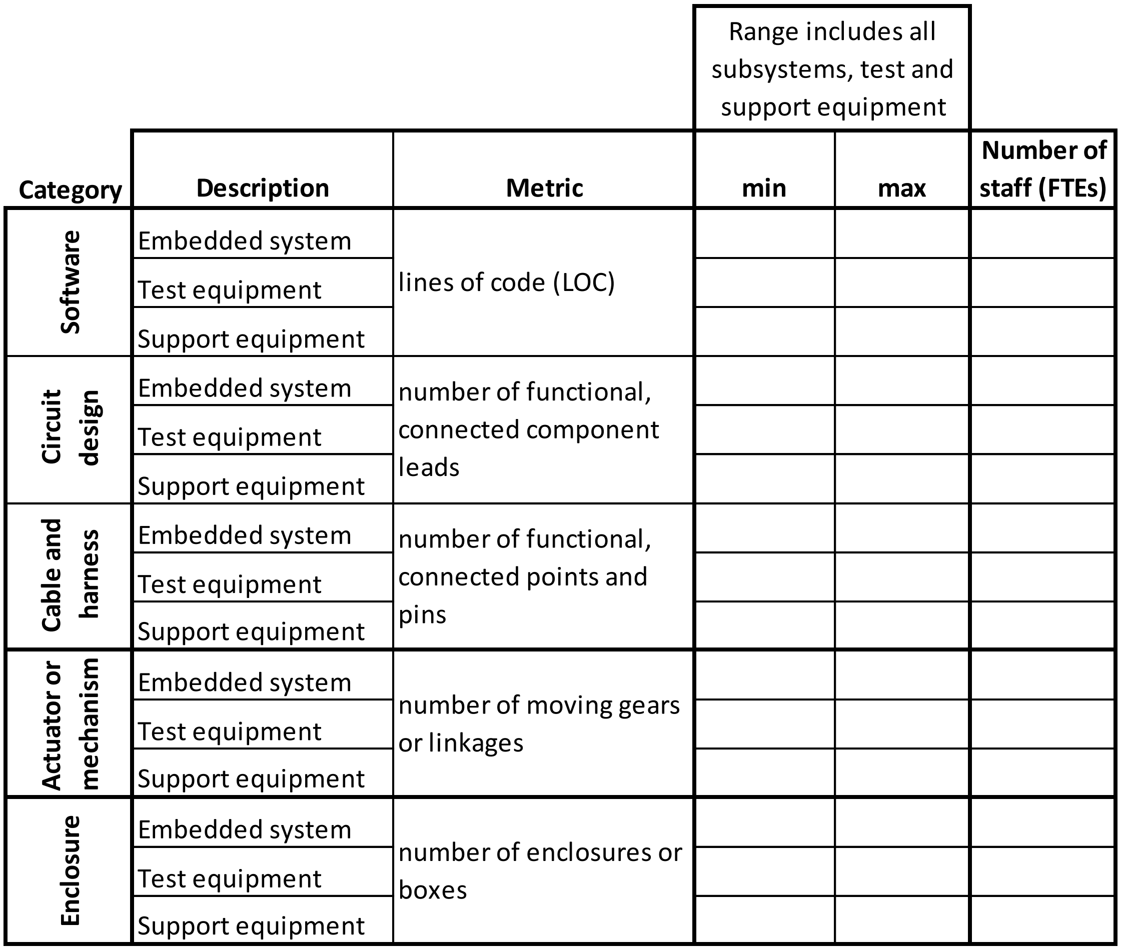
Figure 4.7 Potential page for the sanity check to collect information on purchased or reused modules. It expands to accommodate any number of purchased or reused subsystems. FTE = full-time equivalent, which is the number of people needed to work a task at full-time effort.

Figure 4.8 shows a potential implementation of the webpage for the clean-sheet portion of the embedded system. This chart requires the user's best estimate of the minimum and maximum values that the user would expect for this project and the number of FTEs of people involved in each line item. Appendix E contains suggested studies to provide default values.

Figure 4.9 shows a potential implementation of the default-settings webpage that follows the clean-sheet portion of the embedded system. This chart contains default settings of processes in most embedded project developments; these are activities not necessarily covered by lines of code (LOC) or the number of signal/power connections in circuits. These are minimum values found in most projects for clean-sheet designs. The user sets the check in the column marked "Perform in parallel" if most of the line items are performed by separate people within a category. If not and most line items with a category are performed sequentially by the same person, the user sets the check in the column marked "Performed sequentially."

The last set of default values are for producing prototypes and first-article systems. These prototypes and systems are used for integration tests, environmental tests, electromagnetic compatibility tests, and field tests. Figure 4.10 shows a potential implementation of the webpage that contains default settings of processes to produce prototypes and first-article systems in most embedded project.

Users may change these default values to custom fit specific projects that may reuse significant portions of previous projects. In general, users should not need to adjust these values unless they believe that the defaults values are not long enough or do not include enough effort. Once more research refines these default values, the implementation can be easily updated by the hosting organization.

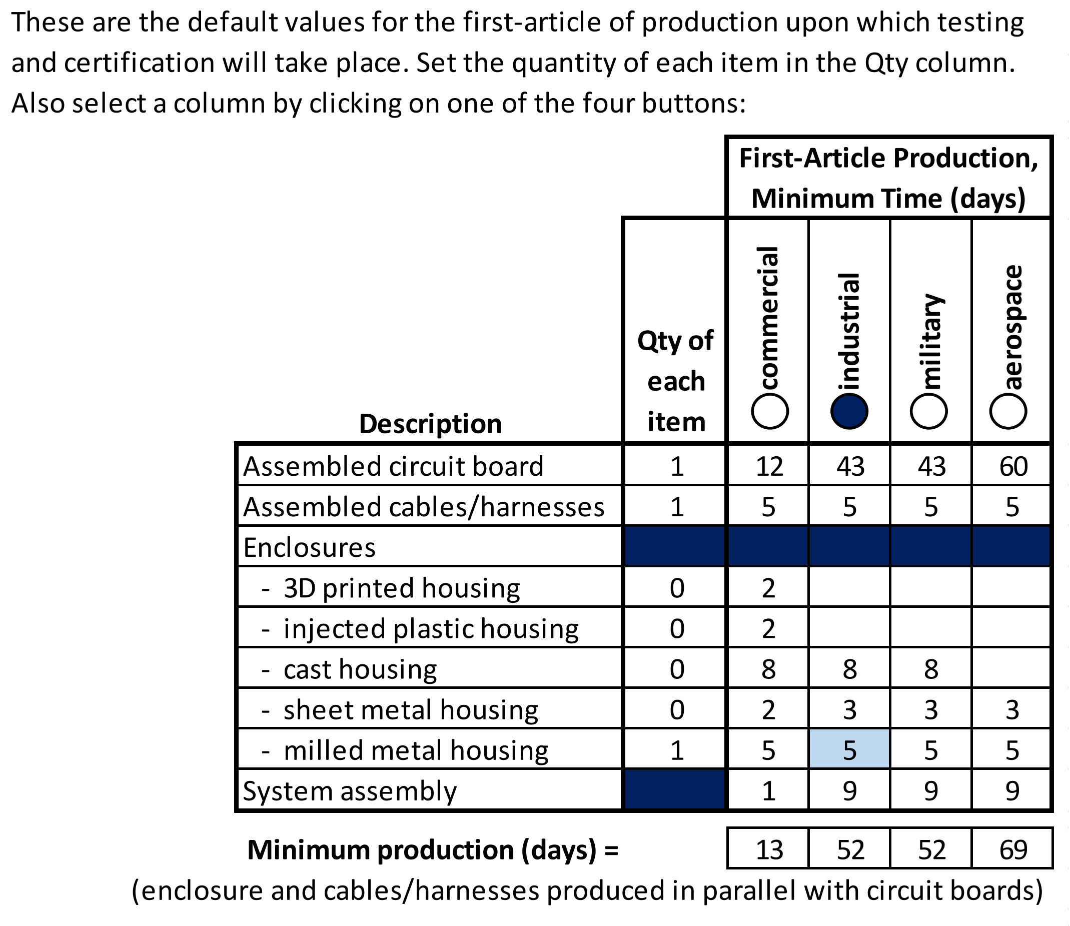


			Range includes all subsystems, test and support equipment		
Category	**Description**	**Metric**	**min**	**max**	**Number of staff (FTEs)**
Software	Embedded system	lines of code (LOC)			
	Test equipment				
	Support equipment				
Circuit design	Embedded system	number of functional, connected component leads			
	Test equipment				
	Support equipment				
Cable and harness	Embedded system	number of functional, connected points and pins			
	Test equipment				
	Support equipment				
Actuator or mechanism	Embedded system	number of moving gears or linkages			
	Test equipment				
	Support equipment				
Enclosure	Embedded system	number of enclosures or boxes			
	Test equipment				
	Support equipment				

Figure 4.8 Potential page for the sanity check to collect information on the clean-sheet design portions of the embedded system. Each category includes all new or modified components to the embedded system, its test equipment and its support equipment. FTE = full-time equivalent, which is the number of people needed to work a task at full-time effort.

			Time		Staffing (FTEs)		Assumed operation	
Category	Description	Units of time	Default min	User update	Default min	User update	Performed in parallel	Performed sequentially
Analysis	Business case	days	5		2		✓	
	Concept of operations		5		1			
	Requirements		5		1			
	Performance		5		1			
	Function		5		1			
	Risk management		5		1			
	Standards		5		1			
	Feasibility, reliability		5		1			
Documentation	Project plan	days	5		2		✓	
	Concept of operations		5		1			
	Architecture and requirements		5		1			
	Data, software design description		5		1			
	Electronic design description		5		1			
	Mechanical design description		5		1			
	Interface control documents		5		1			
	User interface		5		1			
	User Manual		5		1			
Test and integration	Unit tests	days	5		1			✓
	Integration		5		1			
	Field tests on prototypes		5		1			
	Commissioning		5		1			
Compliance	Environmental tests	months	0.5		1			✓
	Electromagnetic compatibility		0.5		1			
	Safety		0.25		1			
	Certification : UL, CE		1		2			
	Approval: FDA, FAA		24		2			
Administration	Meetings: status, working groups, company policies	hours/week	2		3			✓
	Communications: email, phone calls, memos		2		3			
	Contingencies		10		1			

Figure 4.9 Potential page for the sanity check to set default values on the clean-sheet design portions of the embedded system. Colored boxes are highly project dependent. The default minimums are place-holder values until further research, as proposed in Appendix E, has surveyed industry participants for more realistic values.



These are the default values for the first-article of production upon which testing and certification will take place. Set the quantity of each item in the Qty column. Also select a column by clicking on one of the four buttons:

Description	Qty of each item	First-Article Production, Minimum Time (days)			
		○ commercial	● industrial	○ military	○ aerospace
Assembled circuit board	1	12	43	43	60
Assembled cables/harnesses	1	5	5	5	5
Enclosures					
- 3D printed housing	0	2			
- injected plastic housing	0	2			
- cast housing	0	8	8	8	
- sheet metal housing	0	2	3	3	3
- milled metal housing	1	5	5	5	5
System assembly		1	9	9	9
Minimum production (days) =		13	52	52	69

(enclosure and cables/harnesses produced in parallel with circuit boards)

Figure 4.10 Default values for first-article production on the clean-sheet design of embedded systems. This example selected the industrial column with milled metal housings.

The default values in Figure 4.10 are based on published sources and industry values [207]–[209]. These values can be easily updated by the organization hosting the framework's website. Note that preparing for full-scale production means that enclosures must be mass produced, as do the circuit boards and cables/harnesses. Molds for injection molding of plastic enclosures can take 8 to 10 weeks to prepare [208]; that effort is not included here because it is only one of several different technologies that might be chosen for production.

Sanity Check Calculations and Calculation Rationale

The sanity check takes user-supplied values and divides them by internal values of the metric per unit time, which are from industry-cited sources or research values or minimum possible operations. The first published source of default values is lines of code (LOC) produced per hour; these LOC values include all the time to design, code, review, test, and integrate. The rate for lines of code per hour (LOC/h) is a function of module size and tasking during development; Table 4.1 shows average and maximum rates of code production [210], [211]. Table 4.2 provides examples of expected time and effort for various size embedded programs with module tasks of various sizes; it uses Table 4.1 to develop its estimates.

Table 4.1 Industry Values for Production Rates to Develop Software Code.

Module size (KLOC)	Average LOC/h	Maximum LOC/h
1	2.6	5
10	1.3	1.97
20	0.78	1.28
30	0.68	1.08
40	0.61	0.95
50	0.57	0.85
60	0.53	0.78
70	0.5	0.72
1,000	0.3	0.32

KLOC = 1000 LOC
LOC = lines of code

Table 4.2 Time and Effort Estimates for Different Sizes of Programs and Modules.

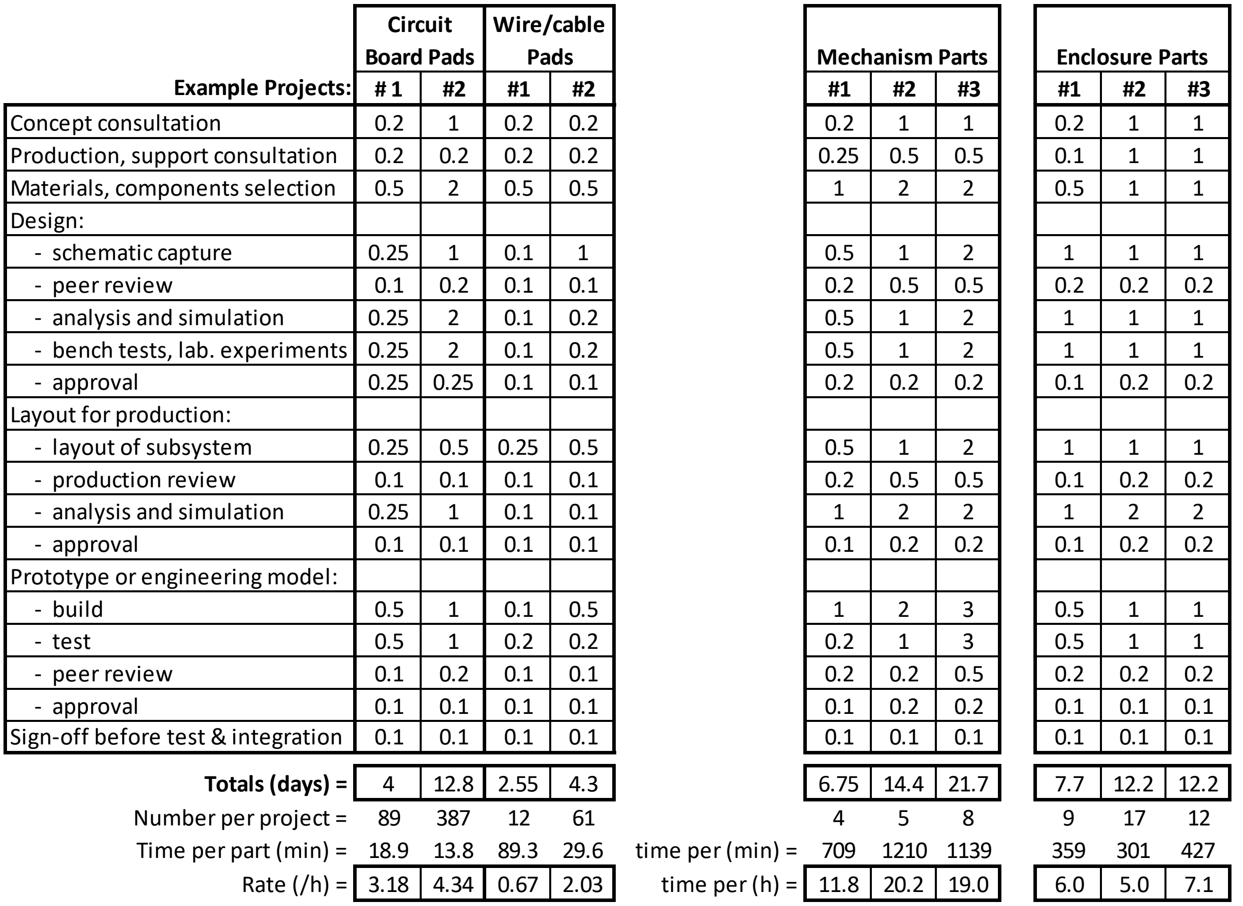
Total program (KLOC)	Module size (KLOC)	Ave. coding time (h)	Design time (h)	Total time (h)	Total (person-years)
10	2	4223	862	5085	2.52
10	5	5792	1182	6975	3.46
100	2	42230	16900	59120	29.3
100	5	62400	25000	87400	43.3
100	10	76900	30800	10800	53.4
100	20	128000	51300	179000	89
1000	2	422000	186000	608000	302
1000	5	624000	275000	899000	446
1000	10	769000	338000	1108000	549
1000	20	1280000	564000	1850000	916

KLOC = 1000 LOC
LOC = lines of code

The value that seems most likely to provide a default for embedded software is about 2.0 lines of code (LOC) per person [210], [211]. This means that about 14 LOC per day for an eight-hour day with one hour of interruptions, communications, meetings, phone calls, and a myriad of disruptive details. A code rate of 2 LOC/h gives a rate of 294 LOC per month of completed code that has been reviewed, tested, and integrated into the embedded system.

Designing circuits needs a simple and reasonable metric, too. While most published efforts use the complexity of each circuit board and number of components to estimate time and effort, this work takes a different approach. It uses a simple metric, which is like the software LOC/h, it only counts the number of connected, component leads on circuit boards plus the number of pins or sockets or wires connected within all the cables and wire harnesses. It uses a simple, "bottom-up" analysis to generate a "best case" shortest time. Table 4.3 currently shows "place-holder" values for the maximum rate of circuit development. A conductor represents either a circuit board trace or a wire in a cable or wiring harness. Further research into time-motion studies can refine this metric (see Appendix E); any line in Table 4.3 can be changed easily by the hosting organization.

Table 4.3 Rationale for "Best Case" Times for Design and Development of Circuits, Cables, Wire Harnesses, Mechanisms, and Enclosures for Embedded Systems to be refined by research described in Appendix E.



	Circuit Board Pads		Wire/cable Pads			Mechanism Parts				Enclosure Parts		
Example Projects:	**# 1**	**#2**	**#1**	**#2**		**#1**	**#2**	**#3**		**#1**	**#2**	**#3**
Concept consultation	0.2	1	0.2	0.2		0.2	1	1		0.2	1	1
Production, support consultation	0.2	0.2	0.2	0.2		0.25	0.5	0.5		0.1	1	1
Materials, components selection	0.5	2	0.5	0.5		1	2	2		0.5	1	1
Design:												
- schematic capture	0.25	1	0.1	1		0.5	1	2		1	1	1
- peer review	0.1	0.2	0.1	0.1		0.2	0.5	0.5		0.2	0.2	0.2
- analysis and simulation	0.25	2	0.1	0.2		0.5	1	2		1	1	1
- bench tests, lab. experiments	0.25	2	0.1	0.2		0.5	1	2		1	1	1
- approval	0.25	0.25	0.1	0.1		0.2	0.2	0.2		0.1	0.2	0.2
Layout for production:												
- layout of subsystem	0.25	0.5	0.25	0.5		0.5	1	2		1	1	1
- production review	0.1	0.1	0.1	0.1		0.2	0.5	0.5		0.1	0.2	0.2
- analysis and simulation	0.25	1	0.1	0.1		1	2	2		1	2	2
- approval	0.1	0.1	0.1	0.1		0.1	0.2	0.2		0.1	0.2	0.2
Prototype or engineering model:												
- build	0.5	1	0.1	0.5		1	2	3		0.5	1	1
- test	0.5	1	0.2	0.2		0.2	1	3		0.5	1	1
- peer review	0.1	0.2	0.1	0.1		0.2	0.2	0.5		0.2	0.2	0.2
- approval	0.1	0.1	0.1	0.1		0.1	0.2	0.2		0.1	0.1	0.1
Sign-off before test & integration	0.1	0.1	0.1	0.1		0.1	0.1	0.1		0.1	0.1	0.1
Totals (days) =	4	12.8	2.55	4.3		6.75	14.4	21.7		7.7	12.2	12.2
Number per project =	89	387	12	61		4	5	8		9	17	12
Time per part (min) =	18.9	13.8	89.3	29.6	time per (min) =	709	1210	1139		359	301	427
Rate (/h) =	3.18	4.34	0.67	2.03	time per (h) =	11.8	20.2	19.0		6.0	5.0	7.1

Designing mechanisms also uses a simple metric, much like designing a circuit. It only counts the number of components, such as gears, bearings, linkages, and belts. It uses a simple, “bottom-up” analysis to generate a “best case” shortest time. Table 4.3 shows the maximum rate of mechanism development; this equates to two or three mechanical components designed, developed and delivered per week for seven productive hours per business day. Further research into time-motion studies can refine this metric (see Appendix E); any line in Table 4.3 can be changed easily by the hosting organization.

Designing enclosures also uses a simple metric, much like designing a circuit or a mechanism. It uses a simple, “bottom-up” analysis to generate a “best case” shortest time. Table 4.3 shows the maximum rate of enclosure development; this equates to less than one enclosure designed, developed and delivered per week for seven productive hours per business day. Further research into time-motion studies can refine this metric (see Appendix E); any line in Table 4.3 can be changed easily by the hosting organization.

The module software within the sanity check compiles the minimum time, minimum effort, and equipment costs, which will then be compared later as boundary constraints against user-estimated values for the project. The most common calculation is for effort, which is calendar time per line item multiplied by FTE. Here are the calculations and their rationale:

1. Calculations for purchased or reused subsystems (Figure 4.7).
    - From the input to the chart, take the maximum delivery time as the “best” and shortest time for delivering all the purchased or reused subsystems.
        - Rationale: assume that all these activities happen in parallel; in reality they do not: one person may be responsible for several subsystems and handle their procurement sequentially, which might be longer than the single, maximum delivery.
    - Sum up all the line item subsystem costs to total the final, delivered costs for purchased or reused subsystems.
    - For each line item, multiply the integration time by the staffing FTE and then divide by 21 days to generate effort in units of person-months.
    - Sum up all the line item results for effort to set the final, “best” effort for receiving and integrating the purchased or reused subsystems.

2. Determine the minimum and maximum default values for design and development of all parts of the embedded system; for lines of code, the calculated number is the final, delivered code; for the physical components, the circuit boards, cables, mechanisms, and enclosures, the final values are only reviewed and finalized designs and does not include the first-article test prototype (Figure 4.8).
    - All these calculations assume seven (7) productive hours of design and development, which assumes an hour of lost to meetings and communications during an eight (8) hour day.
    - Assume that each of these activities occur in parallel, which gives the minimum totals. Many projects share people between disciplines due to limitations on numbers of staff available; sharing people usually only lengthens the final totals for time and effort.
    - To generate the minimum and maximum default values for the time to complete all code, divide the lines of code (LOC) by 14 per day and then divide by 21 days per month to calculate the "best" or shortest time to an operating system in units of months.
        - Rationale: using the published industry values of two (2) LOC per hour or 14 LOC per day.
        - For both minimum and maximum calculated values of time in the previous bullet, multiply the time by the staffing FTE to generate effort in units of person-months.
    - To generate the minimum and maximum default values for the time to complete all lead connections on components and circuit boards, divide the connected, functional leads by some value between 500 and 729 per month to calculate the "best" or shortest time to an operating system in units of months.
        - Rationale: using the assumed best value for completed, reviewed, tested, and implemented leads as 3 to 4 per hour or less than 35 per day or 729 per month.
        - For both minimum and maximum calculated values of time in the previous bullet, multiply the time by the staffing FTE to generate effort in units of person-months.

- To generate the minimum and maximum default values for the time to complete all lead connections on cables and wire harnesses, divide the number of connected, functional leads by some value between 113 and 336 per month to calculate the "best" or shortest time to an operating system in units of months.
    - Rationale: using the assumed best value for completed, reviewed, tested, and implemented leads as less than 1 to about 2 per hour or 16 per day or less than 336 per month.
    - For both minimum and maximum calculated values of time in the previous bullet, multiply the time by the staffing FTE to generate effort in units of person-months.
- To generate the minimum and maximum default values for the time to complete all mechanical connections on mechanisms, divide the number of mechanical connections by some value between 8 and 14 per month to calculate the "best" or shortest time to an operating system in units of months.
    - Rationale: using the assumed best value for completed, reviewed, tested, and implemented mechanical connections as some value between 0.4 and 0.7 per day or less than 15 per month.
    - For both minimum and maximum calculated values of time in the previous bullet, multiply the time by the staffing FTE to generate effort in units of person-months.
- To generate the minimum and maximum default values for the time to complete all enclosures, divide the number of enclosure pieces and penetrations by some value between 23 and 34 per month to calculate the "best" or shortest time to an operating system in units of months.
    - Rationale: using the assumed best value for completed, reviewed, tested, and implemented enclosures as some value between 1.6 and 1.13 per day or less than 34 per month.
    - For both minimum and maximum calculated values of time in the previous bullet, multiply the time by the staffing FTE to generate effort in units of person-months.

- Finally, sum the minimum and maximum values of time and staffing FTE to develop the range of boundary constraints for the clean-sheet portion of the project design.

3. Calculations for ancillary activities (Figure 4.9).
   - For each line item, calculate the effort in person-months by multiplying time by staffing FTE and then dividing the result by 21 days per month.
   - For parallel efforts, use the maximum time as the time for that category.
     - Rationale: this assumes that these tasks are independent and performed by different people, which is not also the case in project development, but it does give the "best" or shortest time for the lower boundary constraint.
   - For sequential efforts, sum the time to determine the total time for that category.
   - For the administration category, just determine the effort in person-months.
4. Calculations for first-article production (Figure 4.10).
   - The software looks at what the user clicked in Figure 4.10 and selects those values.
   - The user may force the use of 3D printing or injected plastic house in the selected column if this is acceptable for prototype purposes.
   - The calendar time is collected from the bottom row and the selected column.
   - Calculate the effort in person-months in two parts:
     - First, multiply the quantity in each row by the time in the selected column.
     - Then divide the calculated time, in the previous line, by 21 days per month.
     - Finally, sum all the to obtain the total effort.
     - Rationale: assume one FTE per item per row; a separate person will complete each copy of the line item if there are multiple copies to be completed in parallel.
5. Final calculations:
   - Sum up all the total times from the previous four sets of calculations (representing Figure 4.7 through Figure 4.10). This is the "best" case or shortest time for developing an embedded system and submitting it to production.
   - Sum up all the total efforts from the previous four sets of calculations (representing Figure 4.7 through Figure 4.10). This is the "best" case or lowest effort for developing an embedded system and submitting it to production.

- One check on time, to cover attempt to uncover hidden time (through parallel tasks but by the same person, for which this analysis does not account), is to divide the total effort (person-months) by the total available staff (assuming they are available the entire project) to get another value of time. Use the longer of the two calculated times, this one or the one above.

Sanity Check Operational Speed

The sanity check should only need to be done once for a project. Like the opening pages, it should be reasonably fast to operate, but it is more involved than the opening pages of the Estimator plug-in module.

Most users should have a good concept of what projects they are planning to develop. Consequently, assuming that a user has numbers in hand, then operating the plug-in module should be fairly fast. Assuming it takes 10 seconds to look up each number and fill in the appropriate box with the number then:

- For Figure 4.7 each line should only take 40 seconds or less. Completing the chart depends on the number of lines, each of which represents a purchased or reused subsystem. For example, an embedded system that has four purchased or reused subsystem should take less than three minutes.
- For Figure 4.8 each line should only take 30 seconds or less. Completing the chart depends on the number of lines, each of which represents a purchased or reused subsystem. If no test or support equipment needs development, then completing the chart should take less than two and a half minutes. If test and support equipment need development, then completing the chart should take less than seven and a half minutes.
- For Figure 4.9 each line should only take 20 seconds or less. Completing the chart depends on the number of lines, each of which represents a purchased or reused subsystem. If no changes are made to the default values, then completing the chart should take only the time to scan through it. If each line needs updating, then completing the chart should take less than 10 minutes.
- For Figure 4.10 each line should only take 10 seconds or less. Completing the chart depends on the number of lines, each of which represents a purchased or reused

subsystem. If no changes are made to the default values, then completing the chart should take only the time to set the quantities, select the column, and select the enclosure type, which totals to less than a minute and a half. If every default time value in the selected column needs updating, then completing the chart should take less than two minutes.

Totaling the operation times gives a minimum worst case of less than 14 minutes and a maximum worst case of less than 19.5 minutes in Figure 4.7 through Figure 4.10. Each of these total times would be extended, in the worst case, by an extra 40 seconds for each purchased or reused subsystem. (Appendix E suggests further studies to establish more precise values and numbers.)

Implementation of the Various Levels of Estimation

The purpose of this section of the estimator is allows users to input their best guesses for time and effort. It has three levels of input as seen in Figure 4.11. Level 1 is the simplest and easiest to complete; it probably will be the least accurate. Level 2 has three pages of inputs, each one for a specific phase of development: Phase 1 – Preliminary, Phase 2 – Critical Design, and Phase 3 – Test, Integration, Compliance, and Preparation Phase (sometimes called the Production Handoff or Manufacturing Transfer Phase). Level 3 has as many input pages as the project has subsystems multiplied by three phases.

Each level can use a simple format, which does not have much detail, as shown in Figure 4.12. Or the input could have greater detail, as found in Figure 4.13. The two different amounts of input detail and the three levels should help "nudge" the user towards thinking more critically about realistic values for time and effort.

After a user completes one of the levels, the estimator calculates the time, effort and cost. Then the sanity check compares the "best case" boundary against the calculated values for time and effort. If the estimator outputs are less than the sanity check values, it will flag a violation of the constraints to the user. The sanity check will help users reduce their psychological bias to the planning fallacy and to act according to the biases of Prospect Theory by underestimating the probability of delay and budget overruns.

Implementation of Level 1 Estimation

The purpose of the Level 1 estimator is to acclimatize a user to the tool; it shortens the time between input guesses of time and effort to output estimates for time and cost by using a format like one of the two in Figure 4.12. The sanity check will flag a violation if either the total project time or total project effort is less than the its constraint boundary. The estimator will still produce time and effort totals even if the user cannot provide average loaded salaries. It is a simple spreadsheet, which anyone could implement; Quinnell's survey results indicate that they do not [10]. Therefore, to ease the upfront burden of preparing a detailed spreadsheet, this proposed module and framework provide it for immediate use.

Most users should have a good concept of what projects they are planning to develop. Consequently, assuming that a user has numbers in hand, then operating the plug-in module should be fairly fast. Assuming it takes 10 seconds to look up each number and fill in the appropriate box with the number then filling in 20 boxes (five rows by four columns) should take less than three and a half minutes. If it is 25 boxes (five rows by five columns), then the time should be four minutes or less.

Once users have run the Level 1 estimator, they may wish to develop a more accurate estimate. They can use the very detailed chart in Figure 4.13, but it will take more time; most projects might not fill in many columns, but if they did they would fill in 27 rows and 19 columns, which is 513 boxes or about an hour and 25 minutes. To shorten time, users may wish to move to a Level 2 estimation.

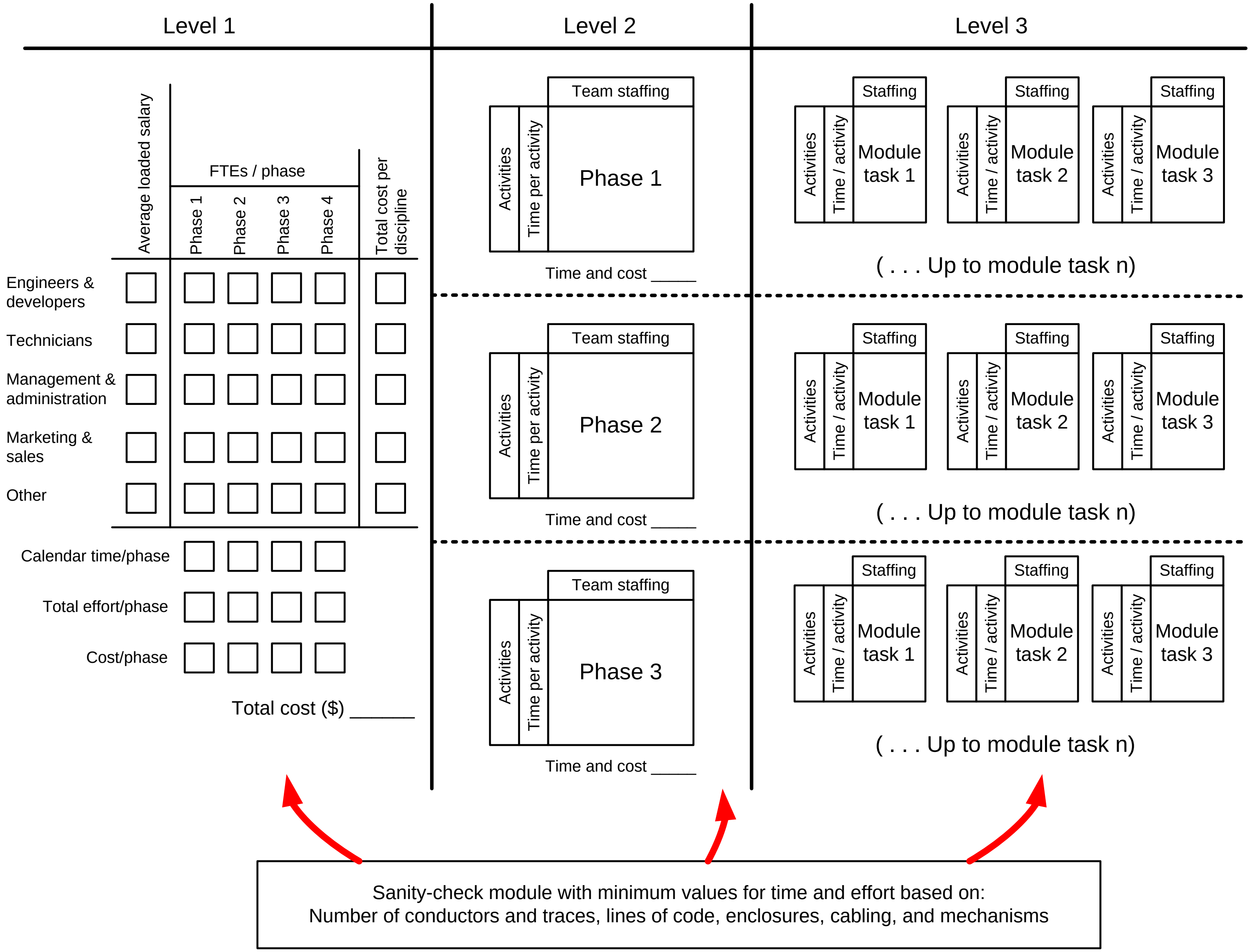
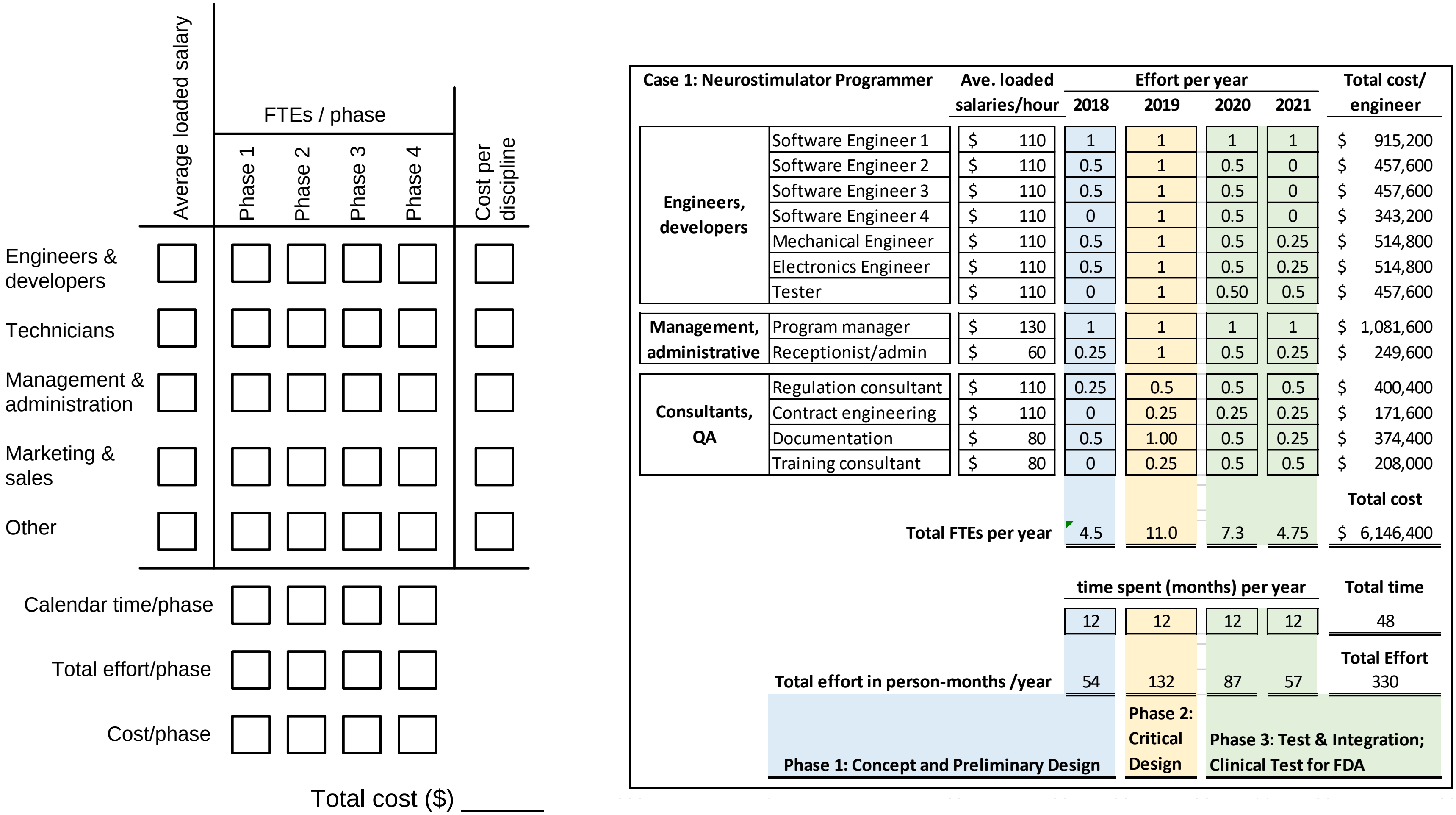


Figure 4.11 Outline of three levels of detail for estimating time, effort and cost for the development of an embedded system.

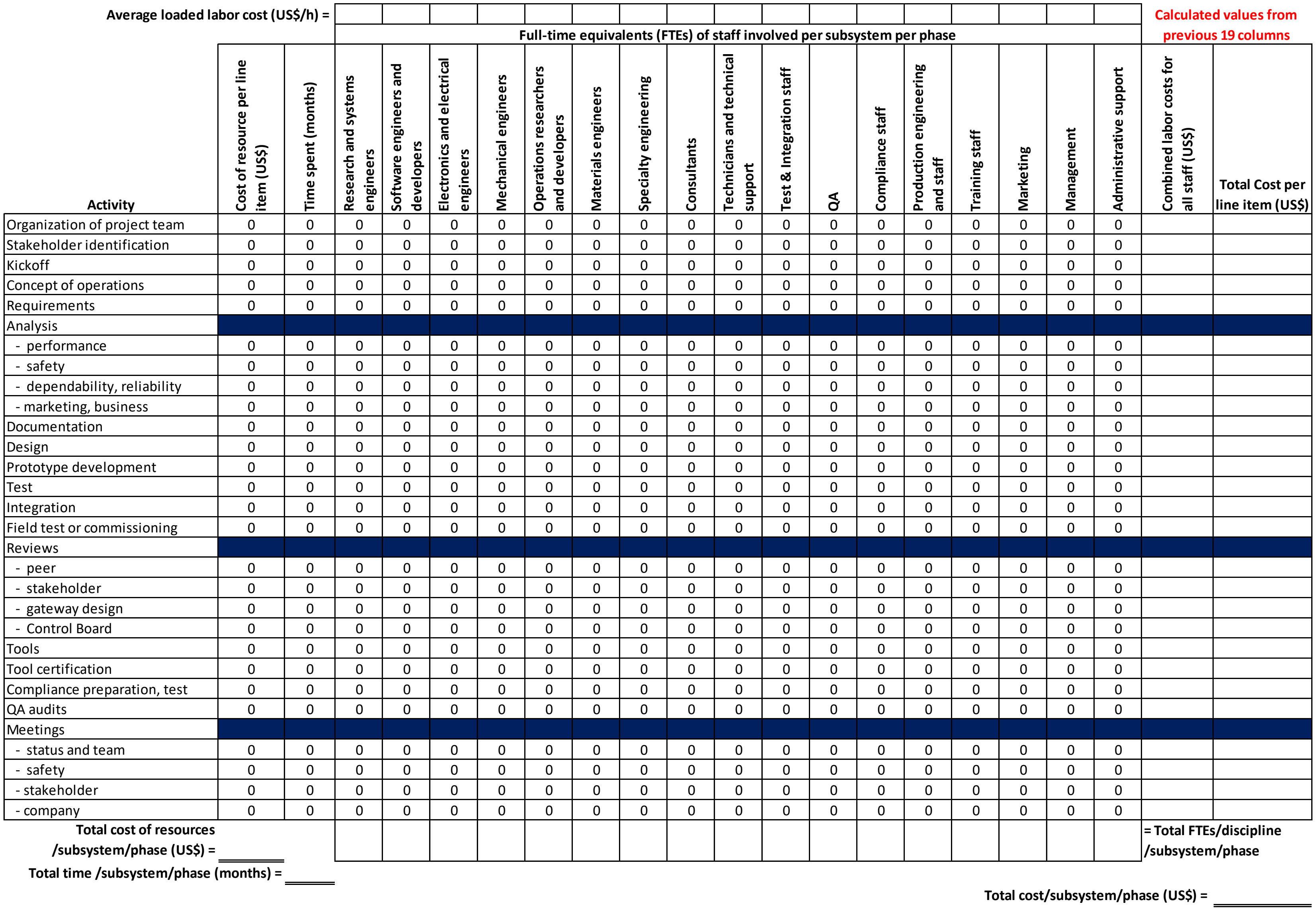


	Average loaded salary	FTEs / phase Phase 1	Phase 2	Phase 3	Phase 4	Cost per discipline
Engineers & developers	□	□	□	□	□	□
Technicians	□	□	□	□	□	□
Management & administration	□	□	□	□	□	□
Marketing & sales	□	□	□	□	□	□
Other	□	□	□	□	□	□
Calendar time/phase		□	□	□	□	
Total effort/phase		□	□	□	□	
Cost/phase		□	□	□	□	

Total cost ($) ______

Case 1: Neurostimulator Programmer		Ave. loaded salaries/hour	Effort per year 2018	2019	2020	2021	Total cost/ engineer
Engineers, developers	Software Engineer 1	$ 110	1	1	1	1	$ 915,200
	Software Engineer 2	$ 110	0.5	1	0.5	0	$ 457,600
	Software Engineer 3	$ 110	0.5	1	0.5	0	$ 457,600
	Software Engineer 4	$ 110	0	1	0.5	0	$ 343,200
	Mechanical Engineer	$ 110	0.5	1	0.5	0.25	$ 514,800
	Electronics Engineer	$ 110	0.5	1	0.5	0.25	$ 514,800
	Tester	$ 110	0	1	0.50	0.5	$ 457,600
Management, administrative	Program manager	$ 130	1	1	1	1	$ 1,081,600
	Receptionist/admin	$ 60	0.25	1	0.5	0.25	$ 249,600
Consultants, QA	Regulation consultant	$ 110	0.25	0.5	0.5	0.5	$ 400,400
	Contract engineering	$ 110	0	0.25	0.25	0.25	$ 171,600
	Documentation	$ 80	0.5	1.00	0.5	0.25	$ 374,400
	Training consultant	$ 80	0	0.25	0.5	0.5	$ 208,000
							Total cost
		Total FTEs per year	4.5	11.0	7.3	4.75	$ 6,146,400
			time spent (months) per year				Total time
			12	12	12	12	48
							Total Effort
		Total effort in person-months /year	54	132	87	57	330
	Phase 1: Concept and Preliminary Design			Phase 2: Critical Design	Phase 3: Test & Integration; Clinical Test for FDA		

Figure 4.12 Two example formats for estimating time, effort and cost for the development of an embedded system. The left-hand format has four phases of development, while the right-hand format combines Phase 3 and 4 into one Phase 3. Either format works fine within the proposed framework.



Average loaded labor cost (US$/h) =																				Calculated values from previous 19 columns	
			Full-time equivalents (FTEs) of staff involved per subsystem per phase																		
Activity	Cost of resource per line item (US$)	Time spent (months)	Research and systems engineers	Software engineers and developers	Electronics and electrical engineers	Mechanical engineers	Operations researchers and developers	Materials engineers	Specialty engineering	Consultants	Technicians and technical support	Test & Integration staff	QA	Compliance staff	Production engineering and staff	Training staff	Marketing	Management	Administrative support	Combined labor costs for all staff (US$)	Total Cost per line item (US$)
Organization of project team	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Stakeholder identification	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Kickoff	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Concept of operations	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Requirements	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Analysis																					
- performance	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
- safety	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
- dependability, reliability	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
- marketing, business	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Documentation	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Design	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Prototype development	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Test	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Integration	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Field test or commissioning	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Reviews																					
- peer	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
- stakeholder	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
- gateway design	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
- Control Board	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Tools	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Tool certification	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Compliance preparation, test	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
QA audits	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Meetings																					
- status and team	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
- safety	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
- stakeholder	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
- company	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Total cost of resources /subsystem/phase (US$) =	________																			= Total FTEs/discipline /subsystem/phase	
Total time /subsystem/phase (months) =		______																			
Total cost/subsystem/phase (US$) =																					____________

Figure 4.13 Example of more-detailed tabular inputs to the estimator and the calculations. Default values are 0; users need only to fill boxes in columns and rows appropriate to their projects.

Implementation of Level 2 Estimation

The purpose of the Level 2 estimator is to “nudge” users to provide better guesses of time and effort by giving more detailed guesses for each phase of development. They would use a format like one of the two in Figure 4.12. The difference from Level 1 estimation is that they will have three charts to complete, one for each phase of development (or four charts if users have four phases of development). The plug-in module will add together the total time from each phase and, separately, the total effort from each phase. The sanity check will flag a violation if either the total project time or total project effort is less than the its constraint boundary. The estimator will still produce time and effort totals even if the user cannot provide average loaded salaries.

If users did use the Level 1 estimator, then they will have begun to learn the tool and probably should take less time to fill in each chart in the Level 2 estimator. Assuming learning, then filling in the three charts, each with 15 boxes (five rows by three columns, users should not have to repeat the salary column) at 10 seconds per box should take less than seven and a half minutes. If it is 20 boxes per chart (five rows by four columns) at 10 seconds per box, then the time should be less than 10 minutes.

Once users have run the Level 2 estimator, they may wish to develop a more accurate estimate. They can use the very detailed chart in Figure 4.13, but it will take more time; most projects might not fill in many columns, but if they do, they require over four hours. To shorten time, users may instead wish to move to a Level 3 estimation.

Implementation of Level 3 Estimation

The purpose of the Level 3 estimator is to “nudge” users even farther than Level 2 and to provide better guesses of time and effort by giving more detailed guesses for each subsystem in the embedded system. They would use a format like one of the two in Figure 4.12. The difference from Level 2 estimation is that they will have to complete a chart for every subsystem during each phase of development (or four charts if users have four phases of development). The plug-in module will add together the total time from each subsystem per phase and, separately, the total effort from each subsystem per phase; then it will sum up the time totals from each phase and separately the effort totals from each phase. The sanity check will flag a violation if either the total project time or total project effort is less than the its constraint boundary. The estimator will still produce time and effort totals even if the user cannot provide average loaded salaries.

If users did use the Level 1 and 2 estimators, then they will have begun to learn the tool and probably should take less time to fill in each chart in the Level 3 estimator. Assuming learning, then filling in the three charts, each with 15 boxes (five rows by three columns, users should not have to repeat the salary column) at 10 seconds per box should take less than seven and a half minutes for each subsystem. If it is 20 boxes per chart (five rows by four columns) at 10 seconds per box, then the time should be less than 10 minutes for each subsystem.

Validity

This plug-in module for estimating time and effort is affected to some degree by threats to validity. This particular module probably suffers the most from threats to its validity, more so than the other two plug-in modules.

Threats to Conclusion Validity

These threats affect the ability to draw a correct conclusion between the independent variable (or treatment) and the dependent variable (or outcome). These threats have to do with the appropriate selection of experiment parameters and implementation [201]. Figure 4.14 illustrates threats to conclusion validity. The threats to conclusion validity, in this estimator, are:

- Inputs of time and effort by users; the threat is that these require human judgment to make guesses.
- Inputs from inexperienced users.
- Inputs of project parameters do not accurately represent the subsystem
    - Lines of code (LOC)
    - Numbers of connects for component leads and cable conductors
    - Numbers of linked mechanical parts
    - Numbers of enclosures
- Setting of default parameters for the sanity checker
    - LOC/h
    - Rate of design and development for circuit connections of components and cables
    - Rate of design and development for linked mechanical parts
    - Rate of design and development for enclosures

- Production rates for first-article systems

Here are remedies and measures that reduce the threats to conclusion validity:

1. To counter the human judgment in making guesses for time and effort, the sanity checker provides boundary constraints to flag if the guesses are too low. If a violation occurs, users may go back and revise their inputs upwards.
2. To counter the human judgment in preparing the sanity checker, the estimator uses easily counted numbers, instead of more involved estimates of complexity and function:
   - It uses lines of code (LOC), which can be estimated as min-max ranges; function points, while potentially more accurate, tend to be more difficult to estimate [201]. Two downsides to using LOC metrics: function points can be more accurate, and LOC does not accurately account for automatically generated code from model-based design. This is a weakness and warrants more study.
   - It uses numbers of circuit connections to the components and cables, which can be estimated as min-max ranges, rather than developing measures of complexity of the circuit boards and cables.
   - It uses numbers of linked mechanical parts, which can be estimated as min-max ranges, rather than developing measures of complexity of the mechanisms.
   - It uses numbers of enclosures, which can be estimated, rather than developing measures of complexity of the enclosures.
3. To counter the human judgment in setting the default parameters in the sanity checker, the hosting organization may use published values and time-motion studies and assume the “best-case” values. This will give the shortest time and least effort in the sanity checker, which determines the optimistic, best boundary constraints. Further research, as suggested in Appendix E, will help reduce the impact of human judgment in setting the default parameters.
4. The sanity checker also collects a range of values in Figure 4.8. It then totals the minimum values to give the min-min boundary. It also totals the maximum values to give the max-min boundary.
5. The estimator will highlight and stress very clearly to users that the estimated results are the minimum values for completely successful projects with no requirements changes

along the way; the results are *not expected values*, but *boundary constraints no project can transgress*.

6. Finally, the third plug-in module, which orchestrates a review by the Delphi method in Chapter 6, has as its sole purpose to get independent judgment of the estimates.

Threats to Internal Validity

These threats can skew or alter the perspective on the actual relationship between the independent variable (or treatment) and the dependent variable (or outcome) by indicating a casual relation different than actual [201]. For users of this estimator, two threats to internal validity exist. One threat is that of poorly worded instructions in the estimator. The other threat is maturation of users while using the framework multiple times for different projects; they may learn to "game" the tools to give results that look good to them but do not bear a semblance to reality. Figure 4.15 illustrates threats to internal validity.

The first threat of poorly worded instructions may be countered in two ways. The first way is to have the hosting organization set up a thorough procedure for editing and publishing the estimator. The second way is to run pilot studies and revise the estimator's wording when problems are found.

The second threat of maturation of users who then "game" the tools, the sanity check is a partial answer and will reduce the problem of underestimating to a degree. This is a difficult, if not impossible, problem to solve and afflicts all estimating tools.

Threats to Construct Validity

These threats can skew or alter the perspective on the actual relationship between the experiment setting and the actual setting under study [201]. For users of this estimator, two threats to construct validity exist. Figure 4.16 illustrates threats to construct validity.

One threat is relying on a mono-method, specifically, a single type of form to estimate time, effort and cost. To counter the mono-method of estimation, a separate form with different parameters to provides a "sanity check" of the first estimator. A second counter to the mono-method is the third plug-in module, which orchestrates a review by the Delphi method in Chapter 6; the review has as its sole purpose to get independent judgment of the estimates, which directly counters the problems of the mono-method.

The other threat is that of users with various experience levels. The separate, “sanity-check” form will “nudge” users towards a more likely result, in spite of different experience levels. Providing the three levels of estimation will also “nudge” users to better guesses. Providing a debrief at the end of the project will also “nudge” users to better, initial guesses.

Threats to External Validity

These threats can alter the generalization of the experimental results to industrial practice [201]. For users of this estimator, two threats to external validity exist. Figure 4.17 illustrates threats to external validity.

One threat is that the wrong people use the estimator and supply inappropriate or incorrect data. The check for this is when the data from the estimator is presented at the company for project contract negotiations, knowledgeable colleagues should spot the incorrect data. Otherwise, there is no security feature, or lock-out, to allow only specific users of the tool.

The other threat concerns basing estimator operations on bad or “toy” case studies. This is particularly a concern for the default parameters. Ongoing research into time-motion studies for the default values and support by the hosting organization will refine the operation of the estimator and reduce this threat.

The third plug-in module, which orchestrates a review by the Delphi method in Chapter 6; the review has as its sole purpose to get independent judgment of the estimator. Being independent and exercised by experts, the Delphi method will help reduce the impact of threats to external validity.

Justification and Novelty

The novelty of this estimator is the same as that in Chapter 1. One, current shortcoming of the proposed framework and of this module is that it does not have a complete set of empirically derived default values. [156] states, "The model parameters [for COCOMO] are derived from fitting a regression formula using data from historical projects (61 projects for COCOMO 81 and 163 projects for COCOMO II).” The module’s cost estimations have no empirical studies evaluating the effectiveness of this module. Future research and pilot studies will further justify or refute its operation utility (Appendix E).

Future Work

To access the construct validity, the estimator needs pilot studies to test its claim of speed and no learning curve. Pilot studies of time-motion values in design and develop will also help establish the default values of the sanity checker. Appendix E suggests outlines for further research and pilot studies.

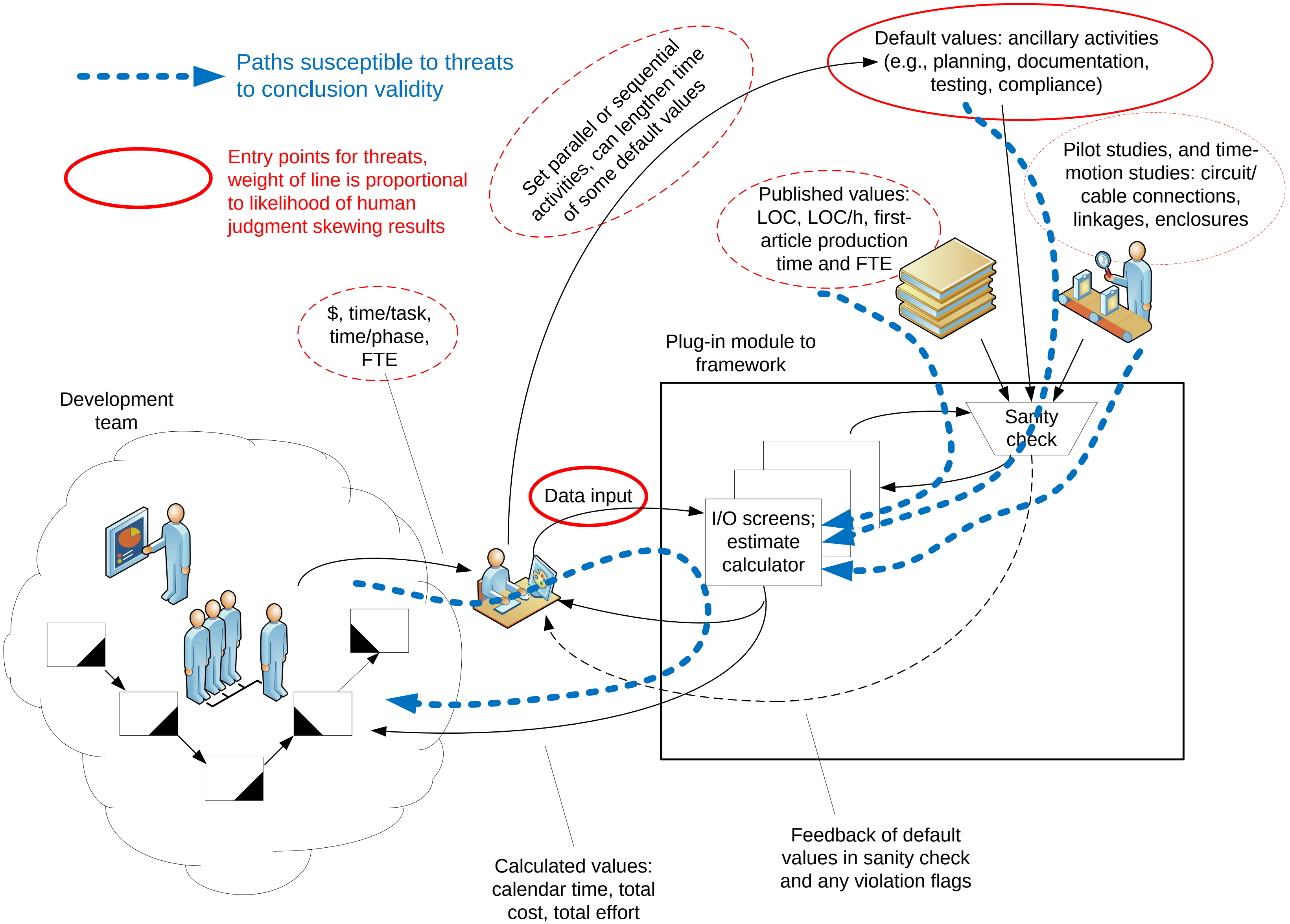

Paths susceptible to threats to conclusion validity
Entry points for threats, weight of line is proportional to likelihood of human judgment skewing results
Set parallel or sequential activities, can lengthen time of some default values
Default values: ancillary activities (e.g., planning, documentation, testing, compliance)
Pilot studies, and time-motion studies: circuit/ cable connections, linkages, enclosures
Published values: LOC, LOC/h, first-article production time and FTE
$, time/task, time/phase, FTE
Plug-in module to framework
Development team
Sanity check
Data input
I/O screens; estimate calculator
Calculated values: calendar time, total cost, total effort
Feedback of default values in sanity check and any violation flags

Figure 4.14 Threats to Conclusion Validity

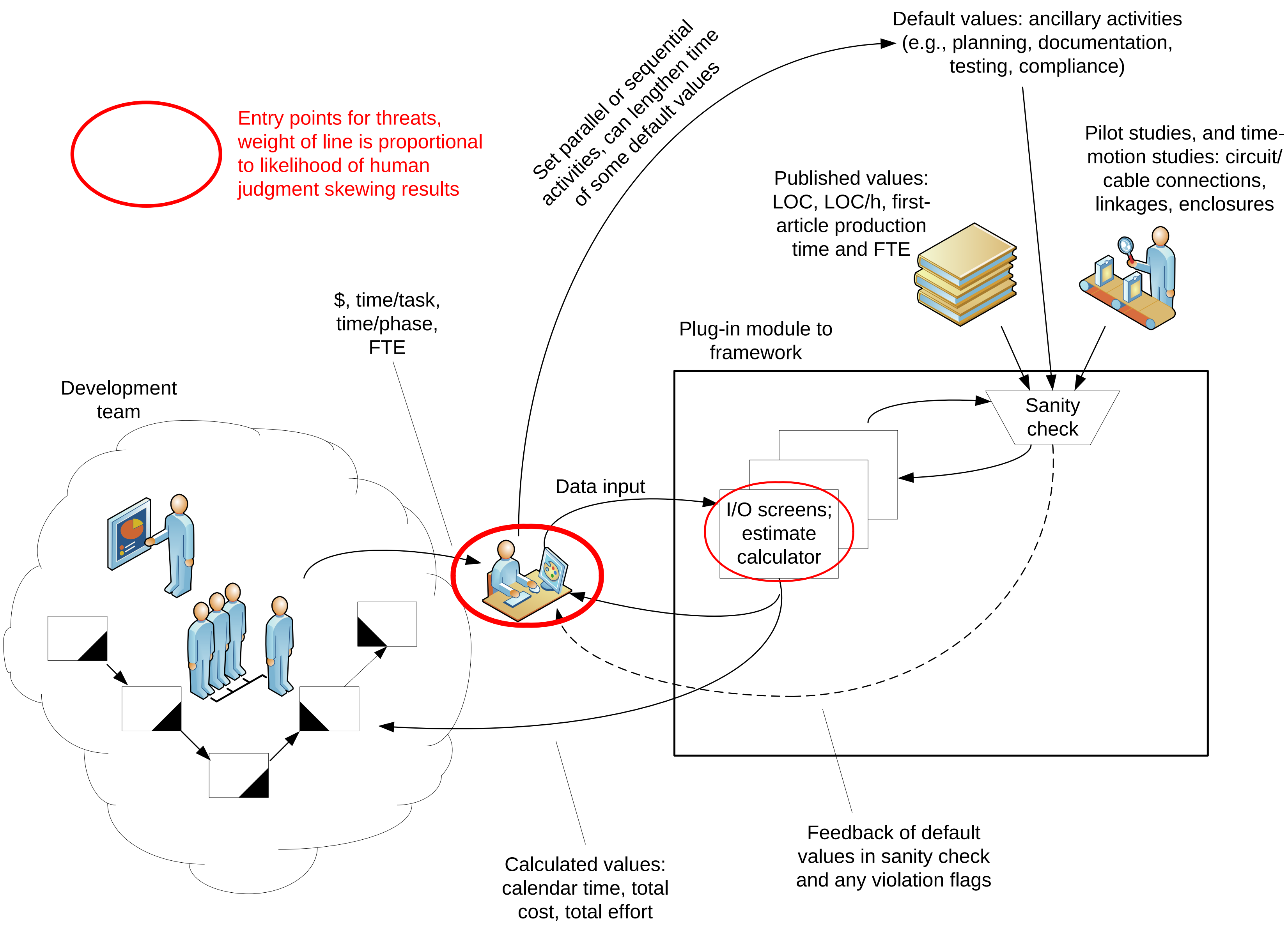

Entry points for threats, weight of line is proportional to likelihood of human judgment skewing results
Set parallel or sequential activities, can lengthen time of some default values
Default values: ancillary activities (e.g., planning, documentation, testing, compliance)
Pilot studies, and time-motion studies: circuit/ cable connections, linkages, enclosures
Published values: LOC, LOC/h, first-article production time and FTE
$, time/task, time/phase, FTE
Plug-in module to framework
Development team
Sanity check
Data input
I/O screens; estimate calculator
Calculated values: calendar time, total cost, total effort
Feedback of default values in sanity check and any violation flags

Figure 4.15 Threats to Internal Validity

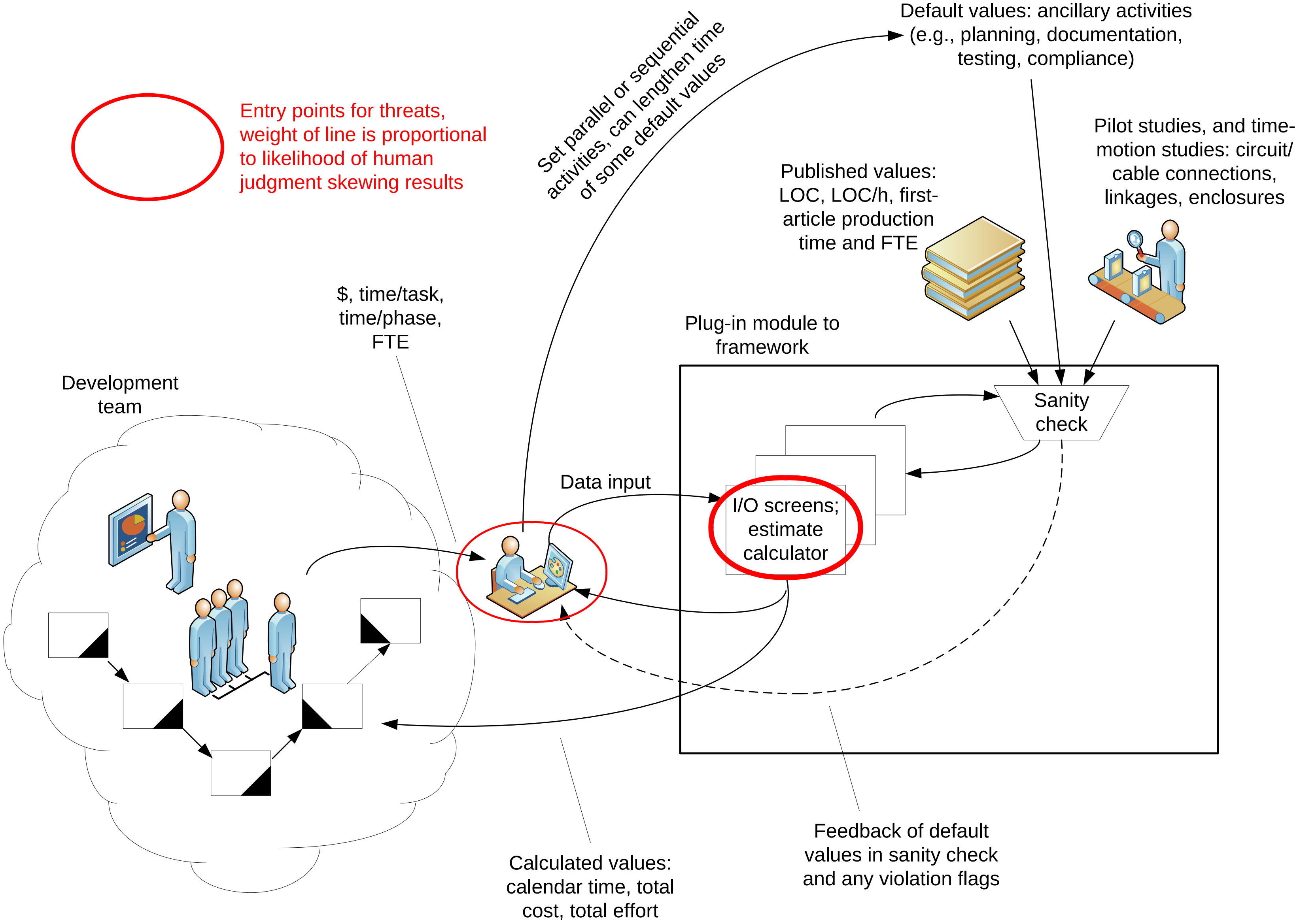

Entry points for threats, weight of line is proportional to likelihood of human judgment skewing results
Set parallel or sequential activities, can lengthen time of some default values
Default values: ancillary activities (e.g., planning, documentation, testing, compliance)
Pilot studies, and time-motion studies: circuit/ cable connections, linkages, enclosures
Published values: LOC, LOC/h, first-article production time and FTE
$, time/task, time/phase, FTE
Plug-in module to framework
Development team
Sanity check
Data input
I/O screens; estimate calculator
Calculated values: calendar time, total cost, total effort
Feedback of default values in sanity check and any violation flags

Figure 4.16 Threats to Construct Validity

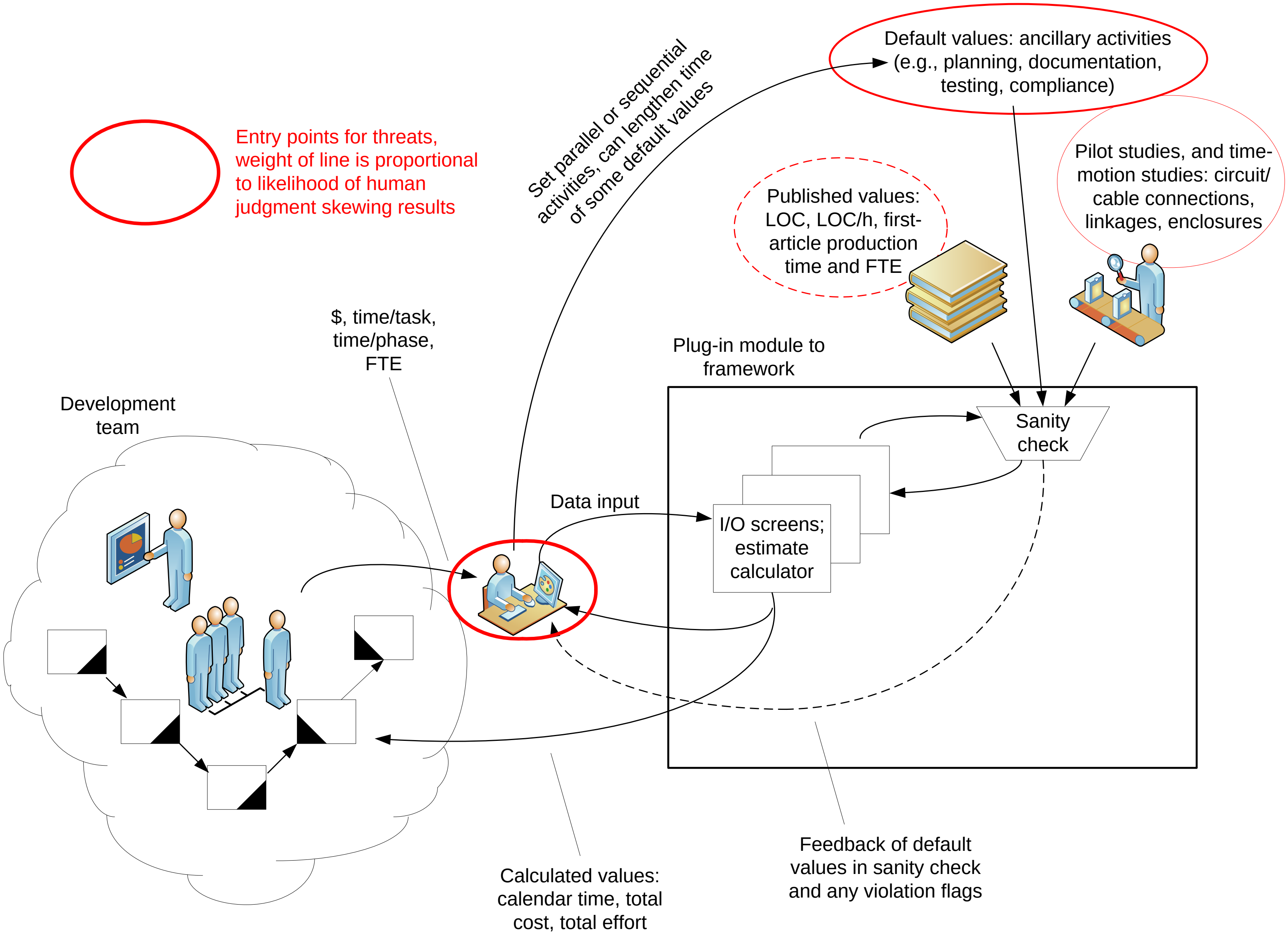

Entry points for threats, weight of line is proportional to likelihood of human judgment skewing results
Set parallel or sequential activities, can lengthen time of some default values
Default values: ancillary activities (e.g., planning, documentation, testing, compliance)
Published values: LOC, LOC/h, first-article production time and FTE
Pilot studies, and time-motion studies: circuit/ cable connections, linkages, enclosures
$, time/task, time/phase, FTE
Plug-in module to framework
Development team
Sanity check
Data input
I/O screens; estimate calculator
Calculated values: calendar time, total cost, total effort
Feedback of default values in sanity check and any violation flags

Figure 4.17 Threats to External Validity

Chapter 5 - Plug-in Recommender, the Build-versus-Buy Decision

Overview and Motivation

Designers face the quandary of cost for subsystems when either buying or building them. The development costs must be amortized over the production life of the product that incorporates these subsystems. Sometimes, buying a subsystem can be far less expensive than building a custom design. Many times, it can also save a lot of time. At other times, however, the purchased subsystem is more expensive because it offers too many features or because it requires modification to meet very precise specifications or because it does not fit well in the final system and engineers must design around its limitations.

The literature is sparse and anecdotal for the build-versus-buy decision in developing embedded systems [214]–[226]. Project management articles and business literature, on the other hand, extensively cover engineering and manufacturing. They describe basic issues in making the build-versus-buy decision; these issues include: cost (materials, labor, manufacturing overhead), direct control of work, control of intellectual property (IP), staff availability, core competency with available resources, learning new skills, and license fees [225], [226]. These basic concepts are necessary but do not encompass all the concerns confronting the developer of embedded systems.

The motivation for this work is the need for a recommender tool for the build-versus-buy decision. The work in this chapter defines parameters for the build-versus-buy decision, studies their impacts on the decision, and proposes a model to recommend decisions. The study examines parameter sensitivity within a breakeven analysis between building a custom subsystem and buying a subsystem. The recommender model uses the results of the study to help recommend a build-versus-buy decision.

The study and recommender use supplied values of cost, time and effort during the design and development of a subsystem to be integrated into an embedded system. Both the study and the recommender incorporate manufacturing costs of the subsystem; neither include the cost of manufacturing the entire system, marketing and sales, distribution, repair, maintenance, or disposal because these costs are assumed equal between a custom subsystem and a purchased subsystem after integrated in the final embedded system. The recommender can stand alone, or it can be a plug-in module in the framework to aid the development of embedded systems.

The recommender model is based on 18 case studies; the data in each case study derive from industry sources. Details of those industry sources are in Appendix F. The chapter wraps up with suggestions for improvements to the method.

Definitions and Concepts

Here are some definitions and concepts needed in this chapter:

- NRE – non-recurring engineering costs that accumulate over the design and development phases.
- COGS – cost of goods sold during manufacturing and production.
- COTS – commercial off-the-shelf; refers to components and subsystems that are readily available for purchase.
- ROTS – rugged off-the-shelf; like COTS but for extreme environments.
- FTE – full-time equivalent, which is the number of people needed to work a task at full-time effort.

Lifecycle refers to the entire duration for which the product exists, from concept to disposal. The phrase, *up-front lifetime buy*, refers to stocking up components or subsystems to inventory over the lifetime of the product. It often is used when vendors announce that they will make components or subsystems obsolete; sometimes it is called an *obsolete-part buy*. *Premium* is the interest on a loan to buy and inventory COTS subsystems over the lifetime of the final product; this is one cost of product longevity.

Cost Calculations

Cost affects nearly all projects. One must weigh the cost of buying system components or subsystems versus building them; many factors affect your decision. Equation 1 is the cost per-unit for custom design and building a subsystem. Equation 2 is the cost per-unit for buying a COTS subsystem.

$$Custom\ unit\ cost = [Comp(N) \cdot discount] + COGS + \\ [NRE + tooling + support + cost\ of\ longevity]/N \quad (1)$$

$$COTS\ unit\ cost = [COTS(N) \cdot discount \cdot premium] + COGS + [NRE + support + cost\ of\ quality\ loss]/N \quad (2)$$

Where:

N = quantity of product manufactured or integrated

Comp(N) = cost of components as a function of *N*

discount = percent reduction on the single-unit price for quantity buys

COGS = in these equations it includes everything, e.g., labor, materials, energy, but not components, which are separate

premium = inventory premium, the factor (or total loan interest) that increases costs to keep COTS product on shelves longer

NRE = function of specifications, team resources, staff, vendor technical support, time:

- specifications = function of scope, complexity, maintenance, client support and expectation, market
- team resources = function of assets, tools, facilities, prototyping components
- staff = function of expertise and availability
- time = time to design, develop, and test the subsystem or subsystem before production

tooling = function of assets, resources, and time where time is to prepare for manufacturing and production

support = function of product complexity, maintenance, operational complexity, projected time span of lifecycle

- Support for development team
- Support for final customer

cost of product longevity = function of market, component obsolescence, inventory, inventory premium, projected time span of lifecycle

cost of quality loss = function of support and documentation, which can force extra effort to be expended by the staff to make up the deficiencies.

Parameters Considered

Many parameters drive the build-versus-buy decision. Figure 5.1 illustrates a model of the decision process. The study varies 18 parameters that are of primary concern for simulating the build-versus-buy decision:

1. Cost
2. Quantity
3. Discount schedule of COTS subsystem
4. Discount schedule of components for custom design
5. Specifications for the subsystem
6. Quality of COTS vendor’s consulting advice and documentation
7. Vendor technical support to development team
8. Vendor support to end customer/client
9. Team resources
10. Team expertise
11. Product longevity in market
12. COTS availability in market
13. Tooling
14. Time to market and lost opportunity costs

Two sigmoid logistics functions

15. Midpoint for complications incorporating COTS
16. Steepness for complications incorporating COTS
17. Midpoint for complications during custom design
18. Steepness for complications during custom design

Model for Studying Parameter Sensitivities

The model is a combination of deterministic calculations and Monte Carlo simulation inside the “black box” in Figure 5.1. It computes the crossover point between the number of units for custom design and for COTS. It also calculates the estimated cost of the units at crossover, the NRE, and time to complete the design development. It incorporates manufacturing costs to calculate the COGS, which is a part of the costs.

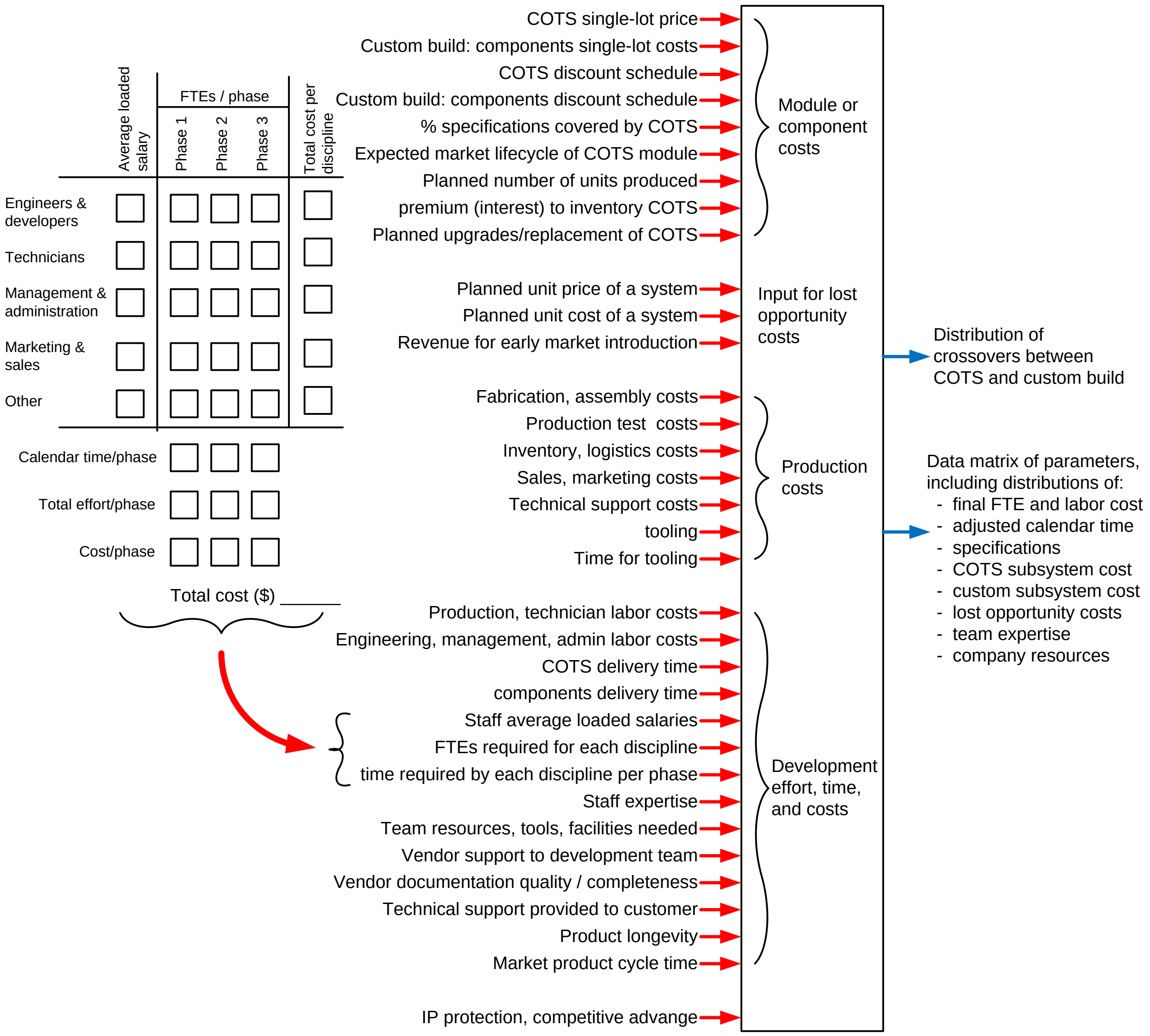

Average loaded salary
FTEs / phase
Phase 1
Phase 2
Phase 3
Total cost per discipline
Engineers & developers
Technicians
Management & administration
Marketing & sales
Other
Calendar time/phase
Total effort/phase
Cost/phase
Total cost ($) ______
COTS single-lot price
Custom build: components single-lot costs
COTS discount schedule
Custom build: components discount schedule
% specifications covered by COTS
Expected market lifecycle of COTS module
Planned number of units produced
premium (interest) to inventory COTS
Planned upgrades/replacement of COTS
Module or component costs
Planned unit price of a system
Planned unit cost of a system
Revenue for early market introduction
Input for lost opportunity costs
Fabrication, assembly costs
Production test costs
Inventory, logistics costs
Sales, marketing costs
Technical support costs
tooling
Time for tooling
Production costs
Production, technician labor costs
Engineering, management, admin labor costs
COTS delivery time
components delivery time
Staff average loaded salaries
FTEs required for each discipline
time required by each discipline per phase
Staff expertise
Team resources, tools, facilities needed
Vendor support to development team
Vendor documentation quality / completeness
Technical support provided to customer
Product longevity
Market product cycle time
Development effort, time, and costs
IP protection, competitive advange
Distribution of crossovers between COTS and custom build
Data matrix of parameters, including distributions of:
- final FTE and labor cost
- adjusted calendar time
- specifications
- COTS subsystem cost
- custom subsystem cost
- lost opportunity costs
- team expertise
- company resources

Figure 5.1 Outline of the simulation model that uses Eqns. (1) and (2) to generate useful outputs in support of the build-versus-buy decision; it is used to test parameter sensitivity.

The model accepts the user input for effort, salaries, and calendar time for designing a custom unit or integrating a COTS unit. Figure 5.1 illustrates the input on the left side of the diagram. It derives from tables like those found in Figure 4.12.

Most inputs to the model are fixed, but 14 parameters were randomly varied and applied to the model during each pass through the model. The 14 parameters are listed above as bullets 5 through 18.

To study parameter sensitivities, an iterative simulation generates the random values for the 14 parameters and then calls the model to complete one set of calculations that generate a single value of the number of units at crossover during each iteration. The variances of the parameters are compared to the collected output values and to the other parameters to determine sensitivity

Deterministic Calculations

The model first takes all inputs at face value and calculates the costs: NRE, COGS, time, and lost-opportunity cost. Then it uses the discount schedules for both the components in the custom design and the COTS units and calculates the crossover points in terms of numbers of units and cost. Figure 5.2 illustrates three crossover points from one of the case studies. The jogs in the curves are points where discount values change.

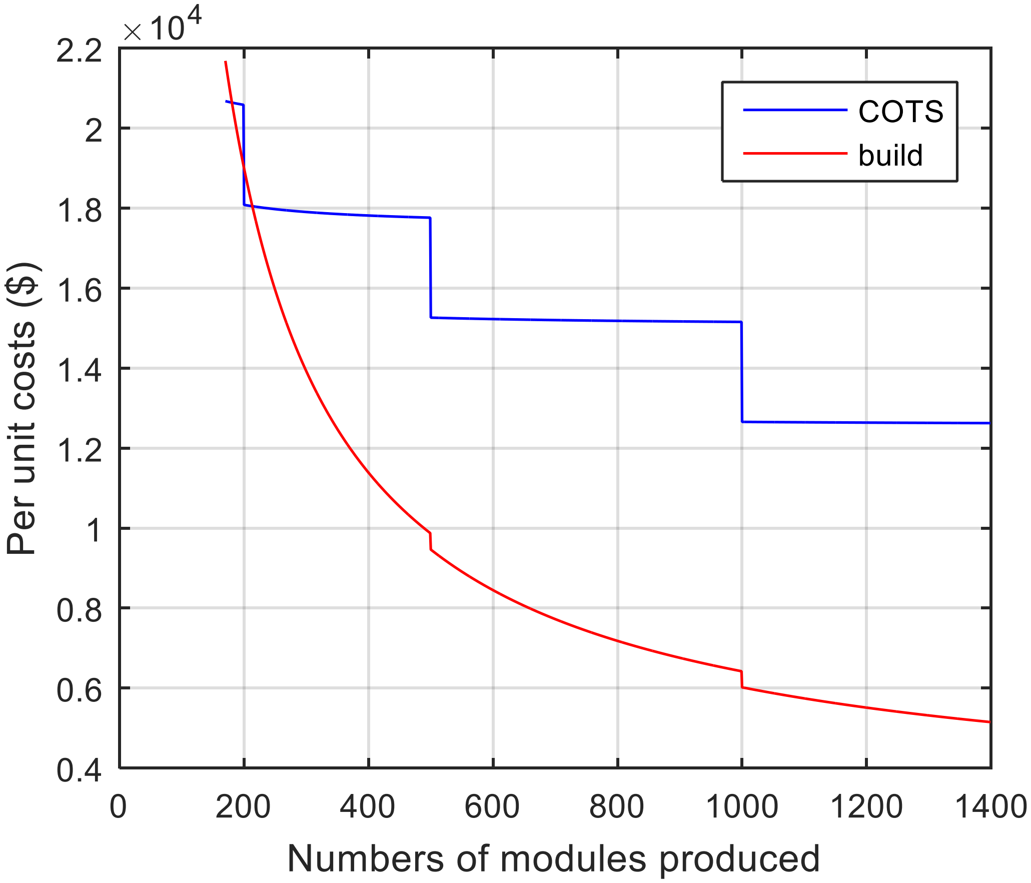


Figure 5.2 Example of the crossover between custom-built units (red line) and COTS subsystems (blue line); the steps in the curves are where discounts change value.

Random Variables – Rayleigh Distribution

I applied an offset Rayleigh distribution (Figure 5.3) to estimates for time spent during design and development of both custom-design and COTS insertion. I chose the Rayleigh distribution because it has a sharp beginning point, below which there are no values; the high-side of the distribution can stretch to multiples beyond unity, which adjusts values of time to simulate schedule overruns. I offset the distribution by 0.8 to allow minor adjustment for over-estimation of development time in some simulation runs, which seldom occurs. Based on the findings of [10], the model assumes that most users will give optimistic values for calendar time.

The models assume that most users will know how many people can be committed to a project, full or part-time, and accepted their inputs of FTEs without modification. They also assume that most users will give optimistic values for calendar time. The models use a random variable, based on the Rayleigh distribution, to adjust the calendar time; the scale is 1.0 and the offset is 0.8. This means that the variable adjusts most estimates to longer time, but a few cases will show a shorter time than supplied to allow for overestimation. A Rayleigh distribution approximates the findings of [10], which are shown in Figure 5.4.

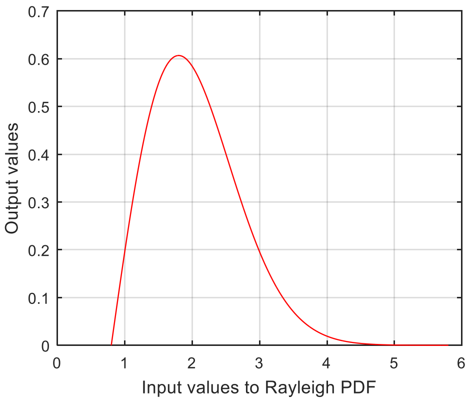


Figure 5.3 Offset Rayleigh distribution to adjust estimates of time in the simulations for crossover and costs.

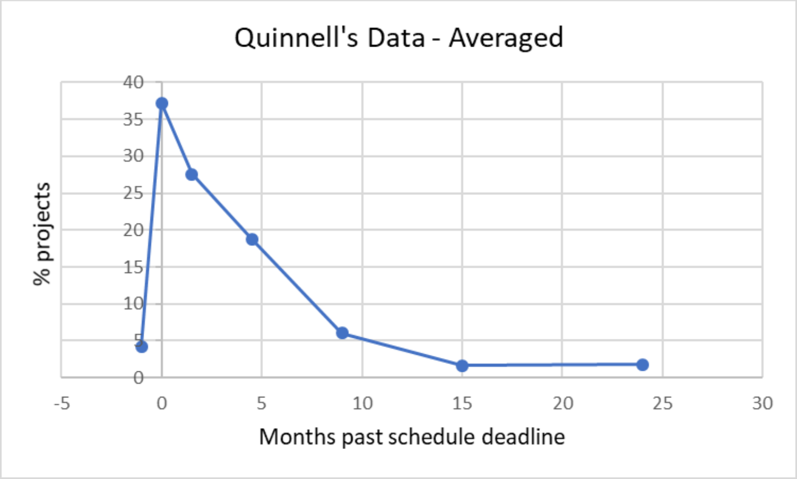


Figure 5.4 Quinnell’s survey data in [10].

Random Variables – Sigmoid Logistics Functions

I assume that lowered amounts of expertise or available resources could interact in such a way to extend the effort by following a sigmoid logistics function rather than a linear function. I used the sigmoid logistic function (Figure 5.5) because it can approximate the compounding of difficulties when various parameters are less than adequate. Here are two examples of interactions that compound difficulties:

- Suppose a COTS subsystem does not have enough I/O pins, the development will then need to design and build an I/O expansion circuit board to accommodate all the necessary input and output signals. The specifications of the COTS subsystem were not sufficient for the desired product, so extra time and effort must be expended to meet requirements.
- Suppose a development team does not have all the necessary expertise to implement a design, using either custom or COTS, then extra time and effort will be expended to learn and cover the shortfall. A small shortfall might put the extra development time at the first knee (nearer to 0%), which only requires some education by the team to solve the problem. A large shortfall in expertise may put the extra development time up the curve closer to 100%, which means that a lot of education and consulting may be needed to finish the design.

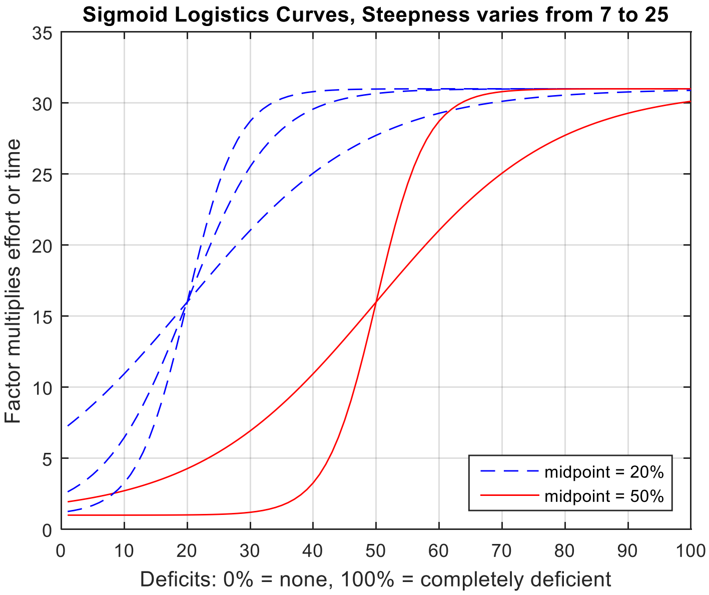


Figure 5.5 Forms of the sigmoid logistics function to adjust variables for deficiencies in expertise, available tools, coverage of specifications, quality of vendor support and documentation that force more time and effort expended by the team, to receive support from the vendor. The scaling factor, which varies from 1 to 31, is an example.

Figure 5.5 illustrates the range of parameters for the sigmoid logistics function. I use two separate sigmoid logistics functions: one for custom design, the other for integrating a COTS unit. I do not know which sigmoid curve to specify, so the models drive the midpoint and steepness values with random variables from flat distributions; the midpoint varied between 0.2 and 0.5 and the steepness varies between 7 and 25.

For custom design, the variables that supplied values to the sigmoid function were expertise and required resources. I assume a factor of three on the sigmoid curve, which could multiply the estimated time by up to three times. While projects can exceed the total effort (FTE multiplied by calendar time) by more than three times, this model assumes that most projects would suffer less than three times overrun in schedule.

For COTS integration, the primary variables that supplied values to the sigmoid function were coverage of specifications, the quality of vendor support and documentation, and the time needed by team to receive support from the vendor. The model assumes a scaling on the sigmoid curve of the input custom effort divided by the COTS effort, which would stretch the estimated time by 30 times or more in some cases (Figure 5.5). Because of the "sunk-cost" fallacy, many projects will exceed the total effort (FTE and time) computed by a scaling factor that compares custom to COTS, but the models capture the overrun in schedule for many projects by rationalizing not to use COTS if its integration exceeded the total effort for custom design.

Intellectual Property or Competitive Advantage

I did not include intellectual property (IP) protection in the model. IP protection is very difficult to model and can be an overarching decision that ignores all other considerations. Sometimes, despite the clear cost advantages for integrating COTS subsystems, a company will still design and build custom subsystems arguing that the competitive advantage of IP protection justifies the decision.

Investigating Parameter Sensitivities

In collaboration with Dr. Sanjoy Das, we studied parameter sensitivities to find the contribution of each parameter to the build-versus-buy decision [227], [228]. We used the eFAST algorithm to determine sensitivities [229], [230]; we ran the eFAST algorithm 50 times for each case study; the algorithm iterated 10,000 times per run. Iterating 10,000 times per run, with 50 runs is similar to that has been done in the literature [231]–[239].

The eFAST algorithm produces two types of sensitivities: a set of ratios of individual parameter variance to the output variance and a set of ratios of all other parameter variances to the output variance, excluding the parameter of interest, and then each ratio subtracted from one. To avoid samples from clumping from two-dimensional random variables, we used importance sampling on the two variables derived from the Rayleigh distribution [240].

We studied 14 parameters, listed as bullets 5 through 18, in the list above. The eFAST algorithm drove each of the parameters with a random variable that had a flat distribution.

Case Studies Used in Modeling the Build-Versus-Buy Decision

I used 18 different case studies of embedded subsystems to simulate the build-versus-buy decision. These studies varied among one other: three studies were software modules, one study was an electronics module, 12 studies were circuit boards with board support software, and two studies were complete subsystems. Table 5.1 illustrates the standard set of parameters for each study; for both custom design and COTS, it includes design FTE, design time, COGS, market lifecycle, inventory premium (total interest over lifecycle), and license and support fees. I chose the initial estimates for FTE and calendar time from personal experience, from the literature review of Chapter 2 and from the case studies in Appendices D and F; Monte Carlo simulations then modified these values.

Case Study 1 – RTOS for a Consumer Appliance

Consumer appliances usually contain small microcontrollers with simple software, but may have several different functions besides direct control, such as clock, timer, and internet connection. A real-time operating system (RTOS) can relieve the burden of scheduling tasks. The choices for development are a freeware RTOS, a commercial RTOS, and a custom design. I compared a commercial RTOS (free but a yearly support license of US$2,000) with a custom design. Eqns. (1) and (2) calculated no crossover point; consequently, I did not run a Monte Carlo simulation.

Case Study 2 – RTOS for a Laboratory Instrument

Laboratory instruments are similar to consumer appliances but may fill mission-critical applications. Like Case Study 1, an RTOS aids the scheduling of tasks. I compared a commercial RTOS (license fee of US$20 cost per manufactured unit plus yearly support of US$2000) with a custom design.

Case Study 3 – RTOS for a Medical Device

Medical devices are similar to laboratory instruments but are safety-critical systems that require approval from the Food and Drug Administration (FDA) in the U.S. An approved RTOS aids the scheduling of tasks. I compared a commercial RTOS (license fee of US$500 cost per manufactured unit plus yearly support of US$10,000) with a custom design.

Case Study 4 – Point-of-Load Converter

Point-of-load (POL) converters are purely electronic modules that transform higher distribution voltages to low, circuit-board voltages. By way of example, Ref. [220] used a simple heuristic for the decision: if the quantity to be produced was greater than 1,000,000 and the required output power was less than 20 W, then staff designed a custom converter; otherwise they bought a COTS module.

I compared a commercial POL module (US$10.99 single-lot quantity, a discount schedule that went from 0% to 20% over 100,000 units, and a premium or total interest of 3.5) with a custom design (US$4.95 in components and a discount schedule that went from 0% to 60% over 100,000 units).

Case Study 5 – ROTS Medical Device Touchscreen

I chose the neurological stimulator programmer from Chapter 6 as the base medical device that would need a touchscreen. I compared a rugged off-the-shelf (RTOS) computer (US$2530 cost per unit and a discount schedule that went from 0% to 20% over 20,000 units) with a custom design (US$230 in components and a discount schedule that went from 0% to 55% over 20,000 units).

Case Study 6 – iPad Medical Device Touchscreen

This is the same as Case Study 5 except I compared an iPad tablet (US$430 cost per unit, a discount schedule that went from 0% to 20% over 20,000 units, and total interest of two times cost to stock inventory) with a custom design (US$230 in components and a discount schedule that went from 0% to 55% over 20,000 units).

Case Study 7 – AFD with a PIC Microcontroller

I chose the arc-fault detector (AFD) from Chapter 6 as the base design, which would need a control board. I compared a PIC microcontroller development board (US$67 cost per unit, a discount schedule that went from 0% to 30% over 20,000 units, and total interest of 3.5 times cost to stock inventory) with a custom design (US$41 in components and a discount schedule that went from 0% to 30% over 20,000 units).

Case Study 8 – AFD with an ARM Microcontroller

This is the same as Case Study 7 except I compared an ARM microcontroller development board (US$175 cost per unit, a discount schedule that went from 0% to 30% over 20,000 units, and total interest of 3.5 times the cost to stock inventory) with a custom design (US$41 in components and a discount schedule that went from 0% to 30% over 20,000 units).

Case Study 9 – AFD Atom Processor, 40% Variation

This is the same as Case Studies 7 and 8 except I compared a ruggedized ePCI unit with an Atom Processor (US$1700 cost per unit, a discount schedule that went from 0% to 30% over 20,000 units, and no premium or interest to stock inventory) with a custom design (US$400 in components, a discount schedule that went from 0% to 30% over 20,000 units, and no inventory premium). The random variables vary over a range of 40% for expertise, available resources, COTS coverage of specifications, quality of vendor support, and the time needed by the team to receive support from the vendor.

Case Study 10 – AFD Atom Processor, 20–40% Variation

This is the same as Case Study 9 except while the random variables vary 40% for expertise and available resources; everything else varies only 20%.

Case Study 11 – AFD Atom Processor, 10–40% Variation

This is the same as Case Study 9 except while the random variables vary 40% for expertise and available resources; everything else varies only 10%.

Case Study 12 – AFD Atom Processor, 10% Variation

This is the same as Case Study 9 except all the random variables vary only 10%.

Case Study 13 – Grain-Bin Controller

Grain-bin controllers monitor sensors for temperature and humidity within grain storage bins, or silos. The architecture is the same as Case Study 8, where I compared an ARM microcontroller development board with a custom design. The two differences from Case Study 8 are that the market lifecycle is 30 years and the premium, or total interest, for stocking inventory is six times the cost of the development board.

Case Study 14 – Network Controller for VOC Sensors

Refineries use sensors to monitor for leakage of volatile organic compounds (VOCs). The architecture can be similar to Case Studies 8 and 13, where I compared a COTS unit with a custom design. The two differences from those case studies are that the COTS market lifecycle is 5 to 15 years and the premium, or total interest, for stocking inventory is three times the cost of COTS.

Case Study 15 – Rugged Processor Board

Many control applications, such as industrial control, ships, and military vehicles, require a mechanically rugged processor that is a circuit board rather than a ruggedized ePCI module found in Studies 9 through 12. I compared a generic, single-processor COTS board (US$6,000 cost per board, a discount schedule that went from 0% to 50% over 1,000 units, no premium or interest to stock inventory, and licensing and support totaling US25,000 per year) with custom design (US$900 in components and a discount schedule that went from 0% to 30% over 1,000 units).

Case Study 16 – Rugged Multi-Processor Board

This case study is very similar to Case Study 15 except multiple processors are required on a single circuit board. I compared a generic COTS board (US$25,000 cost per board, a discount schedule that went from 0% to 50% over 1,000 units, no premium or interest to stock inventory, and licensing and support totaling US$25,000 per year) with custom design (US$4,000 in components and a discount schedule that went from 0% to 30% over 1,000 units).

Case Study 17 – Spacecraft Data System, 40% Variation

This case study a data-acquisition system that has five cameras, seven cables, and a processing unit. I compared a generic COTS system (US$150,000 cost per system, a discount schedule that went from 0% to 40% over 1,000 units, no premium or interest to stock inventory, and no licensing or support) with custom design (US$43,900 in components and a discount schedule that went from 0% to 30% over 1,000 units). Like Case Study 9, the random variables vary 40%, from 60% to 100%, for expertise, available resources, COTS coverage of specifications, and quality of vendor support, and from 0% to 40% for the time needed by team to receive support from the vendor.

Case Study 18 – Spacecraft Data System, 10% Variation

This is the same as Case Study 17 except all the random variables vary only 10%.

Results for Studying Parameter Sensitivities

The Monte Carlo simulations produced distributions of crossover points. Figure 5.6 provides one example of a distribution of crossover points. Some crossovers were at 0, meaning that the custom design is always cheaper than purchasing and integrating COTS in those simulations. Some crossovers were greater than 3000 units.

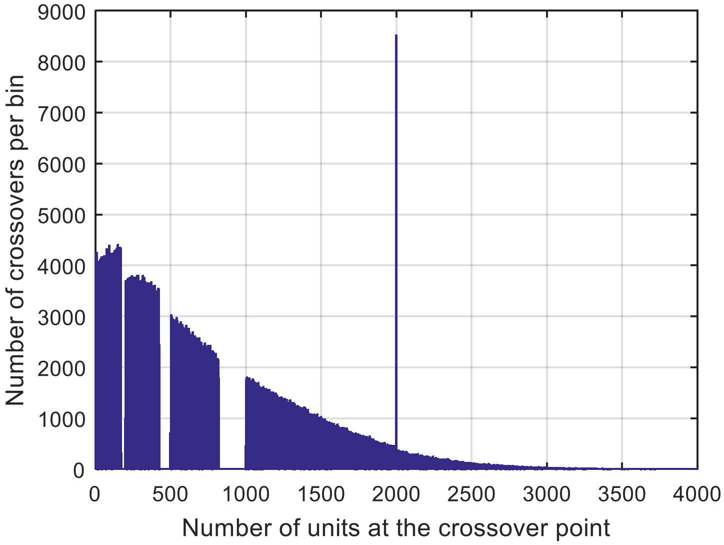


Figure 5.6 An example of a distribution of potential crossover points (or intersections between custom design and COTS, each similar to the ones in Figure 5.2) collected over many runs from the Monte Carlo simulation.

Discount transition points at 200, 500, 1000, and 2000 units cause the gaps and one spike in Figure 5.6. Note that Figure 5.2 has three crossover points, one at 185 units, another at 200 units, and a third at 213 units; multiple crossovers are caused by discount transitions in the curves. As discounts decrease in transition size, then the gaps decrease in width. The model chooses the last crossover point to include in the distribution, which can be a legitimate business decision; alternatively, the model could choose the first crossover and eliminate the gaps in the distribution, either way depends on the user. Either view of the distribution can be appropriate.

While the wide range of possible crossovers is interesting, Figure 5.6 does not reveal how the parameters interact to spread out the distribution. The crossovers can be plotted versus clusters of parameters, which provides some more insight into the parameter interactions. Figure 5.7 is an example of how the crossover distribution spreads and changes shape with increasingly worse values for COTS parameters (specifications, quality of product and service, and time required for support), from dark blue at the front to yellow at the rear. Figure 5.8 is an example of how the crossover distribution spreads and changes shape with increasingly worse values for expertise and available resources, from blue at the front to yellow at the rear.

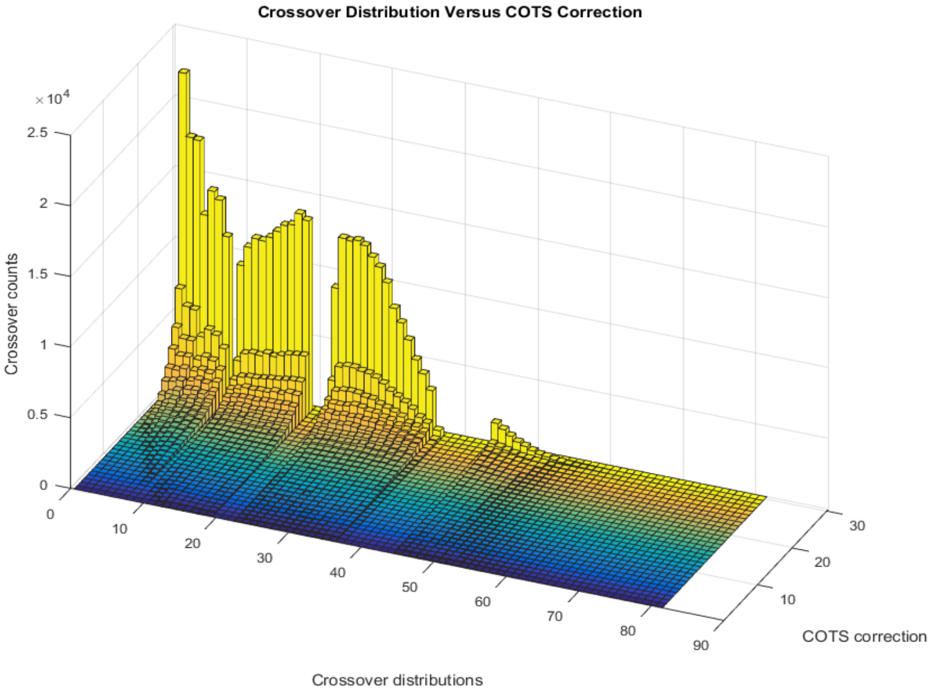


Figure 5.7 A crossover distribution as a cluster of COTS parameters degrade in capability from complete (front, dark blue) to inadequate (rear, yellow).

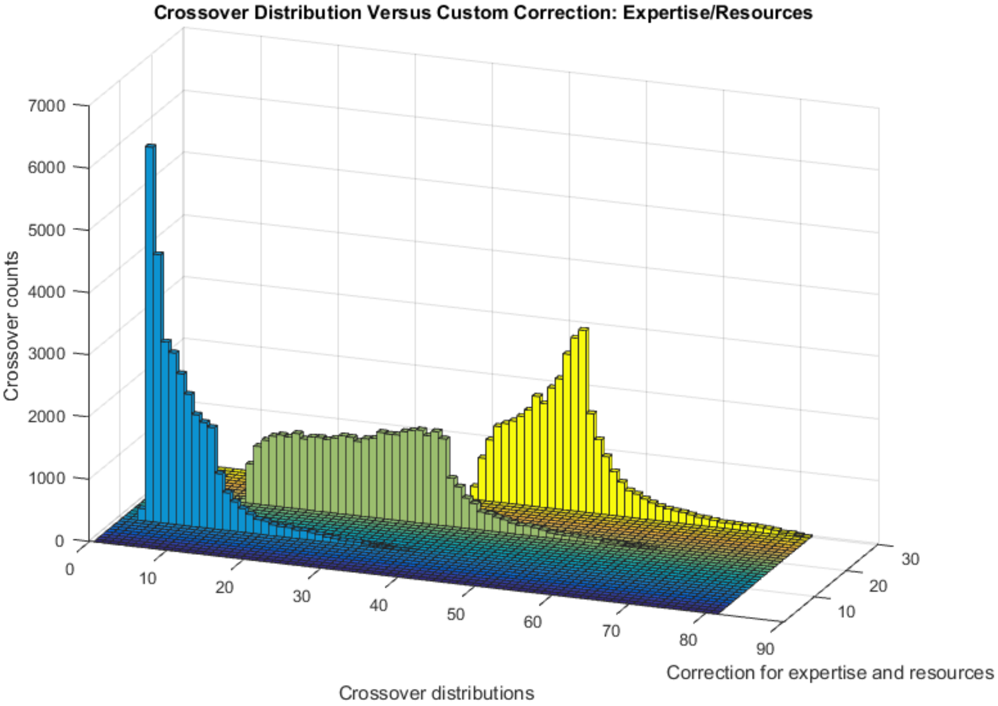


Figure 5.8 A crossover distribution as parameters representing expertise and available resources degrade from complete (front, blue) to inadequate (rear, yellow).

While the graphics of Figure 5.7 and Figure 5.8 give a qualitative, high-level view of what happens to the crossover distribution, they do not give a clear, quantitative answer. The eFAST algorithm provides the necessary quantitative answer.

Table 5.1 contains results from the model, the deterministic calculations, the Monte Carlo simulations, and the study of parametric sensitivities. The first set of boxes, under the Case Studies label, gives the results of the deterministic calculations and the primary output, the number of units at crossover when the custom design becomes cheaper to produce than integrating COTS. The next two sets of boxes, under the Custom Design and COTS labels, give the primary parameters for each case study used in all the calculations and simulations.

The final set of boxes, under the Parameter Total Sensitivity label, gives the output of the eFAST algorithm for parameter sensitivities in each case study. In nearly all case studies, the standard deviations of the parameters were only 1.5% to 6% of the mean of each parameter for total sensitivity. In notable instances, the standard deviation could be as much as 16% of the mean, but these were for situations where parameters contributed very little to the output of the model.

The label called “range of random variables” indicates the range of variation for six random variables:

- 40%: variables X1 to X6 had a range of 40%.
- 20 – 40%: variables X1 to X4 varied only 20%, while variables X5 and X6 varied 40%.
- 10 – 40%: variables X1 to X4 varied only 10%, while variables X5 and X6 varied 40%.
- 10%: variables X1 to X6 had a range of 10%.

Table 5.1 Summary of Study Results for the Parameter Sensitivities in the Build-versus-Buy Decision.

Case Studies

	RTOS				Touchscreen		Arc Fault Detector								Processor			
	appl.	lab. Instr.	med. dev.	POL	ROTS	iPad	PIC	ARM	Atom				Grain bin	VOC	single	multi	Space DAS	
Parameters	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
Crossover (units)	none	28576	4288	1342	1157	4476	13453	3808	311	311	311	311	2046	119	1308	213	135	135
Crossover (unit cost, US$)	none	$ 32	$ 651	$ 271	$7,264	$2,132	$ 224	$ 575	$6,232	$6,232	$6,232	$6,232	$1,054	$42,833	$3,130	$18,058	$162K	$162K
Custom design																		
Single-lot cost (US$)	0	0	0	$ 4.65	$ 230	$ 230	$ 41	$ 41	$ 400	$ 400	$ 400	$ 400	$ 41	$ 705	$ 900	$ 4,000	$43,900	$43,900
NRE cost (US$)	$423K	$617K	$2.64M	$250K	$7.97M	$7.97M	$1.82M	$1.82M	$1.82M	$1.82M	$1.82M	$1.82M	$1.95M	$5.02M	$3.04M	$3.04M	$10.9M	$10.9M
Development time (months)	15	20	48	12	48	48	21	21	21	21	21	21	21	36	30	30	60	60
COTS																		
Single-lot cost (US$)	0	$ 20	$ 500	$10.99	$2,530	$ 430	$ 67	$ 175	$1,700	$1,700	$1,700	$1,700	$ 175	$ 9,375	$6,000	$25,000	$150K	$150K
NRE cost (US$)	$32.5K	$45.9K	$494K	$32.8K	$5.63M	$5.63M	$262K	$262K	$262K	$262K	$262K	$262K	$399K	$2.07M	$107K	$107K	$1.78M	$1.78M
Development time (months)	5	8	15	5	42	42	14	14	14	14	14	14	16	24	6	6	18	18
License (US$/y)	$2K	$2K	$10K	0	0	0	0	0	0	0	0	0	0	0	$25K	$25K	0	0
Support (US$/y)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$25K	$25K	0	0
Premium/total interest	1	1	1	3.5	1	2	3.5	3.5	3.5	3.5	3.5	3.5	6	3	1	1	1	1
COTS lifecycle (y)	5 to 10	5 to 10	5 to 10	2 to 4	5 to 10	0.4 to 1	2 to 4	2 to 4	2 to 4	2 to 4	2 to 4	2 to 4	2 to 4	5 to 15	2 to 4	2 to 4	5 to 10	5 to 10
System lifecycle (y)	5 to 10	5 to 10	5 to 10	10 to 20	5 to 10	5 to 10	10 to 20	10 to 20	10 to 20	10 to 20	10 to 20	10 to 20	10 to 30	10 to 30	4 to 8	4 to 8	5 to 10	5 to 10
Total Parameter Sensitivity (each column normalized to sum to 1)																		
Range of random variables:	**40%**	**40%**	**40%**	**40%**	**40%**	**40%**	**40%**	**40%**	**40%**	**20-40%**	**10-40%**	**10%**	**40%**	**40%**	**40%**	**40%**	**40%**	**10%**
X1: COTS specifications	none	0.021	0.020	0.049	0.044	0.065	0.019	0.019	0.019	0.019	0.005	0.054	0.069	0.028	0.019	0.019	0.020	0.054
X2: COTS quality	none	0.020	0.019	0.048	0.043	0.064	0.018	0.019	0.019	0.019	0.005	0.054	0.069	0.027	0.019	0.019	0.020	0.054
X3: Support to team, NRE	none	0.020	0.020	0.048	0.041	0.064	0.018	0.019	0.019	0.019	0.005	0.054	0.069	0.028	0.019	0.019	0.020	0.054
X4: Support customer, NRE	none	0.020	0.020	0.049	0.042	0.064	0.018	0.019	0.018	0.019	0.005	0.054	0.069	0.027	0.019	0.019	0.020	0.054
X5: Expertise of team	none	0.305	0.305	0.201	0.238	0.123	0.310	0.307	0.308	0.291	0.328	0.066	0.082	0.155	0.311	0.306	0.304	0.064
X6: Available resources	none	0.308	0.306	0.201	0.238	0.123	0.310	0.307	0.309	0.291	0.329	0.066	0.083	0.159	0.310	0.306	0.304	0.065
X7: Product longevity	none	0.002	0.002	0.026	0.016	0.049	0.002	0.002	0.002	0.001	0.002	0.006	0.068	0.170	0.002	0.002	0.002	0.006
X8: COTS lifecycle	none	0.002	0.002	0.026	0.016	0.049	0.002	0.002	0.002	0.001	0.002	0.006	0.068	0.137	0.002	0.002	0.002	0.006
X9: Vendor support cost	none	0.002	0.002	0.025	0.016	0.049	0.002	0.002	0.002	0.001	0.002	0.006	0.068	0.023	0.002	0.002	0.002	0.006
X10: Final customer support	none	0.002	0.002	0.026	0.016	0.048	0.002	0.002	0.002	0.001	0.002	0.007	0.068	0.023	0.002	0.002	0.002	0.006
X11: COTS sigmoid steepness	none	0.008	0.007	0.037	0.023	0.056	0.007	0.007	0.007	0.008	0.004	0.042	0.068	0.025	0.007	0.007	0.007	0.042
X12: COTS sigmoid midpoint	none	0.020	0.019	0.049	0.043	0.064	0.019	0.019	0.018	0.072	0.026	0.350	0.069	0.031	0.019	0.019	0.021	0.357
X13: Design sigmoid steepness	none	0.032	0.032	0.045	0.045	0.070	0.031	0.034	0.033	0.029	0.032	0.057	0.070	0.039	0.030	0.034	0.034	0.055
X14: Design sigmoid midpoint	none	0.241	0.245	0.171	0.180	0.112	0.244	0.245	0.245	0.227	0.255	0.178	0.080	0.129	0.242	0.245	0.243	0.175

Orange indicates largest values within a column 🟧 Yellow indicates second largest values within a column 🟨 Blue indicates largest values in the chart 🟦

Gray indicates parameters with least impact ⬜ Green column has the least parameter variation 🟩

Dominant Parameters – Expertise and Resources

The dominant parameters in these simulations were team expertise and available resources. Table 5.1 highlights rows for X5 and X6 with an orange background to show the dominance over other parameters.

The definition of expertise is the knowledge and capability to understand the problem and generate a solution. A team with all the necessary expertise would rate at 100%. A team deficient in either knowledge or capability derated from 100%; we limited the lowest values to 60%. A deficient team will need time and appropriate instruction to acquire the necessary knowledge or to develop the capability, which increases the time and effort in the project according to the sigmoid logistics function.

The definition of available resources is the tools the team should have to complete the work efficiently. Examples of tools include board-support software; diagnostic equipment, such as logic analyzers and oscilloscopes; computer-aided design software, such as circuit board layout; software checking tools, such as compilers, and tracking and versioning software.

Secondary Parameters

The secondary parameters in these simulations were the sigmoid logistics parameters of midpoint and steepness for design. Table 5.1 highlights row X14 and parts of row X13 with a yellow background to show the dominance over other parameters. The midpoint in X14 is the most prominent. Both rows, X13 and X14, are tied to expertise and available resources, as described in the section above.

Least Impactful Parameters

The parameters with the least effect on these simulations were rows X7 through X10, which involved lifecycles, licensing, and support. Table 5.1 highlights rows X7 through X10 with a gray background to show their minimal effect on the model output.

Most Variable Parameter – Case Studies 12 and 18

The parameter with the greatest variation in Table 5.1 is the midpoint for the sigmoid logistics function that modified the parameters in rows X1 through X4: sufficient coverage of specifications by the COTS subsystem, quality of COTS support and COTS documentation, and COTS vendor's support of the development team. Table 5.1 highlights the boxes in row X12 with a blue background to show their dominance in the entire chart. These two case studies had the minimum parameter variations, 0 to 10%, for the Monte Carlo simulations. Clearly, when expertise and resources approach 100% (varying here from 90% to 100%), other parameters, such as COTS specifications, quality, and support, become as important.

Case Study 13 Has Flattest Distribution Among Studies

Case Study 13, the sensor controller for grain bins, has the flattest distribution of parameter sensitivities among the 18 case studies. Table 5.1 green-highlights the column. This case study has the largest ratio for system lifecycle to COTS lifecycle. Its premium is 6, which is the largest and corresponds to a total of interest of nearly 8% per annum (30 y – 4 y = 26 y of inventory stock assumed paid by loan at 8% annual interest).

Case Study 14 Dominated by X5 Through X8

Case Study 14, the network controller for VOC sensors, is dominated by the parameters for expertise, available resources, and lifecycles. This case study has a large ratio for system lifecycle to COTS lifecycle. Its premium is 3, which is high and corresponds to a total of interest of nearly 8% per annum (30 y – 15 y = 15 y of inventory stock assumed paid by loan at 8% annual interest).

Case Studies 7 Through 11, Variations in X1 Through X4

Case Study 11 experienced a four-fold decrease in parameter total sensitivities for a two-fold decrease in percent variation from Case Study 10. Case Study 10 had a two-fold decrease in parameter total sensitivities for a two-fold decrease in percent variation from Case Studies 7, 8, and 9.

Case Studies 5 and 6, Changes in Sensitivities

The parameter sensitivities for X1 through X4 and X13 in Case Study 6 were between 50% and 200% greater than in Case Study 5. At the same time X5, X6, and X14 decrease by 50% to 100% from Case Study 5 to Case Study 6.

Both case studies had identical NRE costs and allotted calendar time; the difference was in the lifecycles and premiums. The Case Study 6 had a premium of 2, which corresponds to a total of interest of nearly 8% per annum (10 y – 1 y = 9 y of inventory stock assumed paid by loan at 8% annual interest). Case Study 5 had a premium of unity in the simulations.

Discussion of the Parameter Sensitivities

The primary finding in this study is that the expertise of the development team and the resources available to the team are the most important factors in developing the embedded systems studied in this paper. A secondary finding, which is closely tied to the first, is that deficits in expertise or resources will lead to significantly greater cost and expended time in overcoming the deficits. Premiums, or interest on inventory, and product lifecycles play less important, tertiary roles towards increasing the development costs and schedules.

Clearly, providing educational opportunities to sharpen technical skills will improve expertise. These opportunities might include short courses, tutorials, webinars, textbooks, tradeshow attendance, and technical-conference attendance. Consider that a course or conference registration and a week's salary for attending is a very small price to pay when compared to schedule and budget overruns that extend both by 50%, 100%, or more within a single project!

Similarly, for tools and resources, a small investment can save significant time and money in a project. A diagnostic instrument that costs US$5,000 could reduce to seconds a troubleshooting effort that might take two weeks or more. If two engineers are involved in the effort, that is a month of salary lost and much more than the cost of the US$5,000 instrument.

The secondary finding in this study is that the simulation is sensitive to the adjustments made by the sigmoid logistics function. The function models the interactions of tasks; the more complicated and interdependent the tasks are, the steeper the curve and the farther its midpoint moves to the left. Partitioning tasks carefully and decoupling them will reduce the sigmoid steepness and move the midpoint towards the right; this means that small deficiencies will have a lesser impact on the project's schedule and budget.

Few COTS subsystems are straight drop-ins to the developing system, even though advertised as plug-and-play. There is always time and effort expended to integrate the module or subsystem into the final system. I provided reasonable estimates for effort and time to generate integration costs above the simple addition of purchase price and COGS handling.

Assuming specifications are met, COTS quality and support play only a tertiary role towards increasing development costs, while premiums and product lifecycles play very minor roles. Appendix G contains additional data and results, including details of the simulation inputs.

Model for Recommending Whether to Build or to Buy

Cost Crossover Model

The model, in Figure 5.9, is a set of calculations with some extreme values derived from the Monte Carlo simulation inside the “black box” in Figure 5.1. It computes the crossover point between the number of units for custom design and for COTS. It also calculates the estimated cost of the units at crossover, the NRE, and time to complete the design development. It incorporates manufacturing costs to calculate the COGS, which is a part of the costs.

The model accepts the user input for effort, salaries, and calendar time for designing a custom unit or integrating a COTS unit. Figure 5.1 illustrates the input on the left side of the diagram. It derives from tables like those found in the input tables of Appendix F.

The model provides expected ranges of crossover for different ranges of parameters. It does not provide single-value estimates of the crossover numbers or costs to the user because that would require determining specific values for either very subjective or very difficult evaluations of parameters such as expertise, available resources, and the quality of COTS support and documentation. Instead, the model will provide statements that take an example form of “half of all expert teams with the necessary resources available to them will experience total effort below the median value of [X], while half of all teams with significantly less than the necessary expertise and resources will experience total effort above the median value of [Y].”

Other Recommender Models

Cost crossover is used in many business situations, but cost crossover is not always the primary concern. Decisions could be based on IP protection or because no COTS are available with the necessary specifications or because the company wants to learn new technology building a custom design. There are many other management objectives that would force the decision away from considering COTS; these objectives and considerations are beyond the scope of this work.

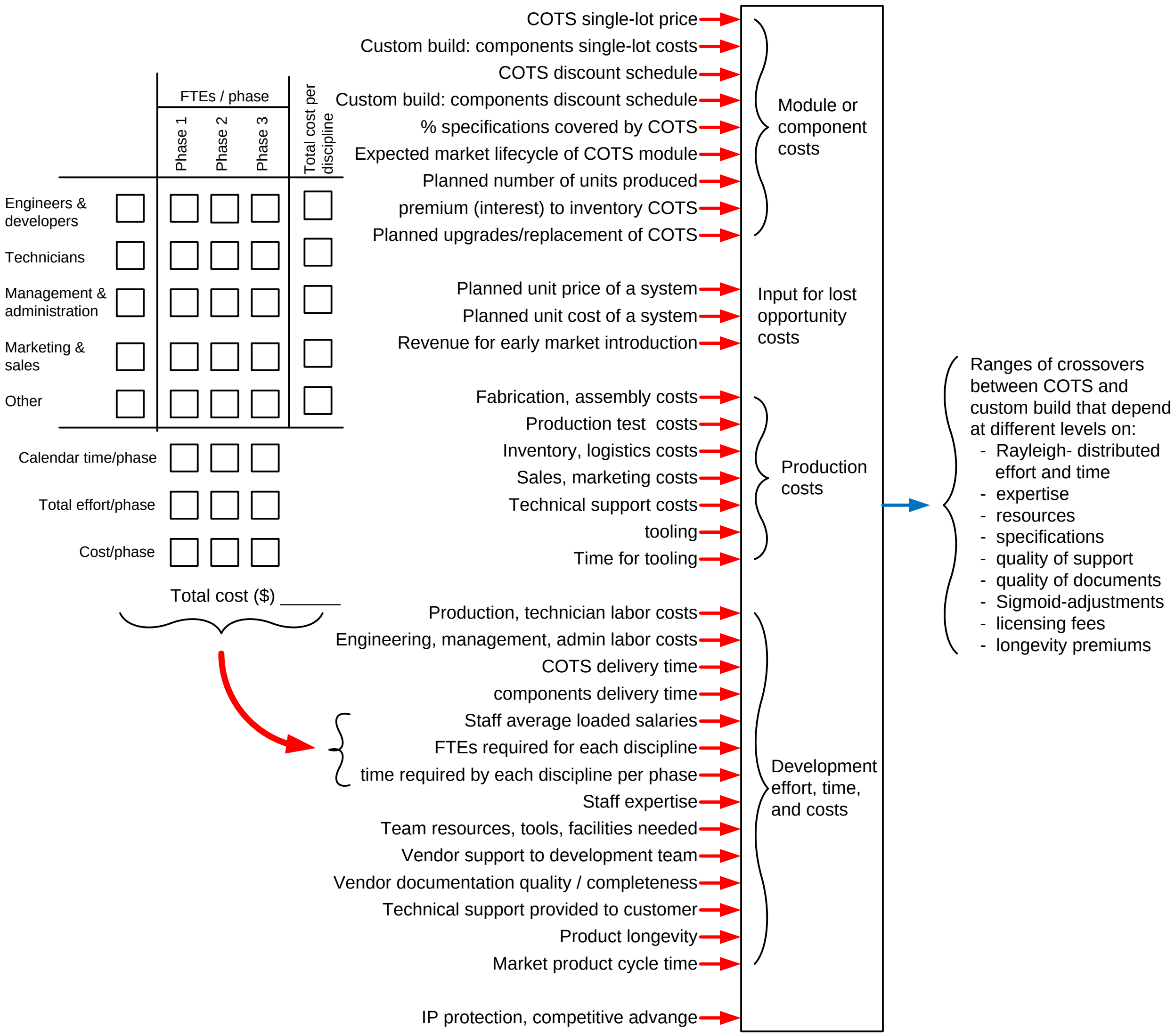

FTEs / phase
Phase 1
Phase 2
Phase 3
Total cost per discipline
Engineers & developers
Technicians
Management & administration
Marketing & sales
Other
Calendar time/phase
Total effort/phase
Cost/phase
Total cost ($) ______
COTS single-lot price
Custom build: components single-lot costs
COTS discount schedule
Custom build: components discount schedule
% specifications covered by COTS
Expected market lifecycle of COTS module
Planned number of units produced
premium (interest) to inventory COTS
Planned upgrades/replacement of COTS
Module or component costs
Planned unit price of a system
Planned unit cost of a system
Revenue for early market introduction
Input for lost opportunity costs
Fabrication, assembly costs
Production test costs
Inventory, logistics costs
Sales, marketing costs
Technical support costs
tooling
Time for tooling
Production costs
Production, technician labor costs
Engineering, management, admin labor costs
COTS delivery time
components delivery time
Staff average loaded salaries
FTEs required for each discipline
time required by each discipline per phase
Staff expertise
Team resources, tools, facilities needed
Vendor support to development team
Vendor documentation quality / completeness
Technical support provided to customer
Product longevity
Market product cycle time
Development effort, time, and costs
IP protection, competitive advange
Ranges of crossovers between COTS and custom build that depend at different levels on:
- Rayleigh- distributed effort and time
- expertise
- resources
- specifications
- quality of support
- quality of documents
- Sigmoid-adjustments
- licensing fees
- longevity premiums

Figure 5.9 Outline of the model that recommends a decision regarding whether to build or buy. It is very similar to Figure 5.1.

Primary Parameters

Figure 5.9 illustrates many parameters that drive the build-versus-buy decision. I focus on 18 parameters that I consider of primary concern for modeling the build-versus-buy decision; these 18 parameters are listed above.

Model Calculations

The model first takes all inputs at face value and calculates the costs: NRE, COGS, time, and lost-opportunity cost. Then it takes the discount schedules for both the components in the custom design and the COTS units and uses Eqns. (1) and (2) to calculate the crossover points in terms of numbers of units and cost. Figure 5.2 illustrates three crossover points from one of the case studies. The jogs in the curves are points where discount values change.

Rayleigh Distribution – Ranges of Effort

The model assumes that most users will know how many people can be committed to a project, full or part-time, and accepts their inputs of FTEs without modification. Based on the findings of Ref. [10], the model assumes that most users will give optimistic values for calendar time. The model fits three different Rayleigh distributions to the survey values [10] to adjust the calendar time; the scale parameter may be 0.15, 0.5, or 1.0 while the offset is either 0.9 or 0.8 for the distributions.

Figure 5.10 illustrates three offset Rayleigh distributions, with scale parameters of 0.15, 0.5, or 1.0. This means that the model adjusts most estimates to longer ranges of time, but a small percentage of cases will fall in a shorter range of time than supplied, which allows for overestimation. Table 5.2 provides the values used in the model for ranges generated by the Rayleigh distributions.

Table 5.2 Values of the Rayleigh Distribution Used in the Model to Recommend a Build-Or-Buy Decision.

Scale, σ	Median	Offset	Median + offset	% of CDF < 1
0.15	0.177	0.9	1.077	20%
0.5	0.589	0.8	1.389	7.7%
1.0	1.177	0.8	1.977	2.0%

CDF = cumulative distribution function

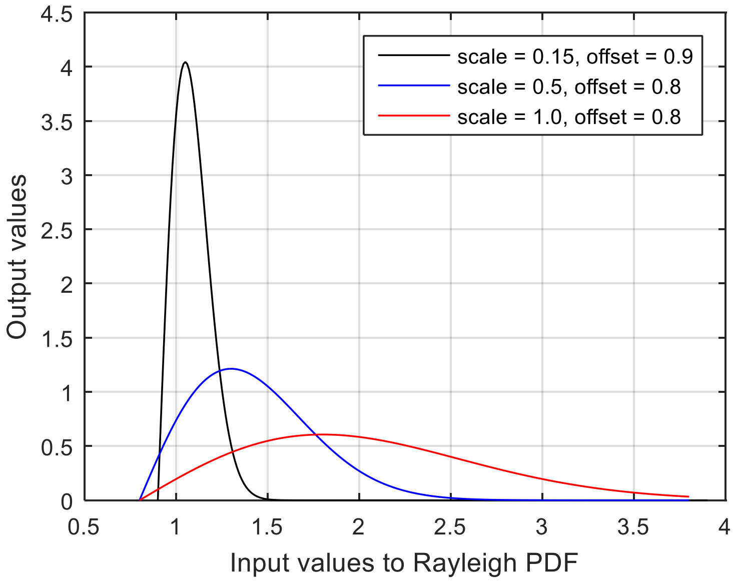


Figure 5.10 Rayleigh distributions used in model.

Sigmoid Logistics Functions

The model assumes that lesser expertise or available resources would interact in such a way so as to extend the effort through a sigmoid logistics function rather than a linear function. It uses the same curves in Figure 5.5, which illustrates the range of parameters for the sigmoid logistics function. The model uses two separate sigmoid logistics functions: one for design, the other for integrating a COTS unit. The model does not specify a single sigmoid curve; instead, it gives ranges for the values of midpoint and steepness; midpoint values range between 0.2 and 0.5 and steepness values range between 7 and 25.

The first set of sigmoid curves adjust the project time for the design-and-development effort according to expertise and available resources. By way of example, an expert team with no deficiencies and all available resources will have a factor multiplier of 1.0. A team with no expertise and missing many resources will have a factor multiplier of 3.0. While projects can exceed the total effort (FTE multiplied by calendar time) by more than three times, I assumed most projects would suffer less than three times overrun in schedule.

The second set of sigmoid curves adjust the project time for COTS integration. The primary variables that supply values to the sigmoid function are coverage of specifications, quality of vendor support and documentation, and time needed by the team to receive support from the vendor. I assumed a factor multiplier on the sigmoid curve of the custom effort divided by the COTS effort, which could stretch the estimated time; by way of example, Figure 5.5 shows a factor of 31 times.

If projects are partitioned very carefully into manageable tasks that are decoupled, then the sigmoid midpoint moves to the right towards 0.5, and the steepness is a minimum of 7.0. As projects are mishandled and highly coupled with ill-defined tasks, then the sigmoid midpoint moves to the left towards 0.2, and the steepness approaches a maximum of 25.

Intellectual Property or Competitive Advantage

The model does not include intellectual property (IP) protection. IP protection is very difficult to model and often an overarching decision that ignores all other considerations. Sometimes, despite the clear cost advantages for integrating COTS subsystems, a company will still design and build custom subsystems, arguing that the competitive advantage of IP protection justifies the decision.

Learning New Technology or Market

The model does not include the situation where management might decide to do a custom design because it wants to move into an entirely new field or to develop the expertise of the staff. I considered this beyond the scope of this work.

Maintaining Facilities and Staff

The model does not include the situation where management might decide to do a custom design because it wants to keep facilities and staff fully employed. I considered this beyond the scope of this work.

Outline of Model Operation

The model goes through a series of qualification steps to indicate potential outcomes in the build-versus-buy decision. Figure 5.11 is the flowchart of the steps within the model. Each stage has a circled number on the right side of the diagram, which is a reference to the stage explained in the text below.

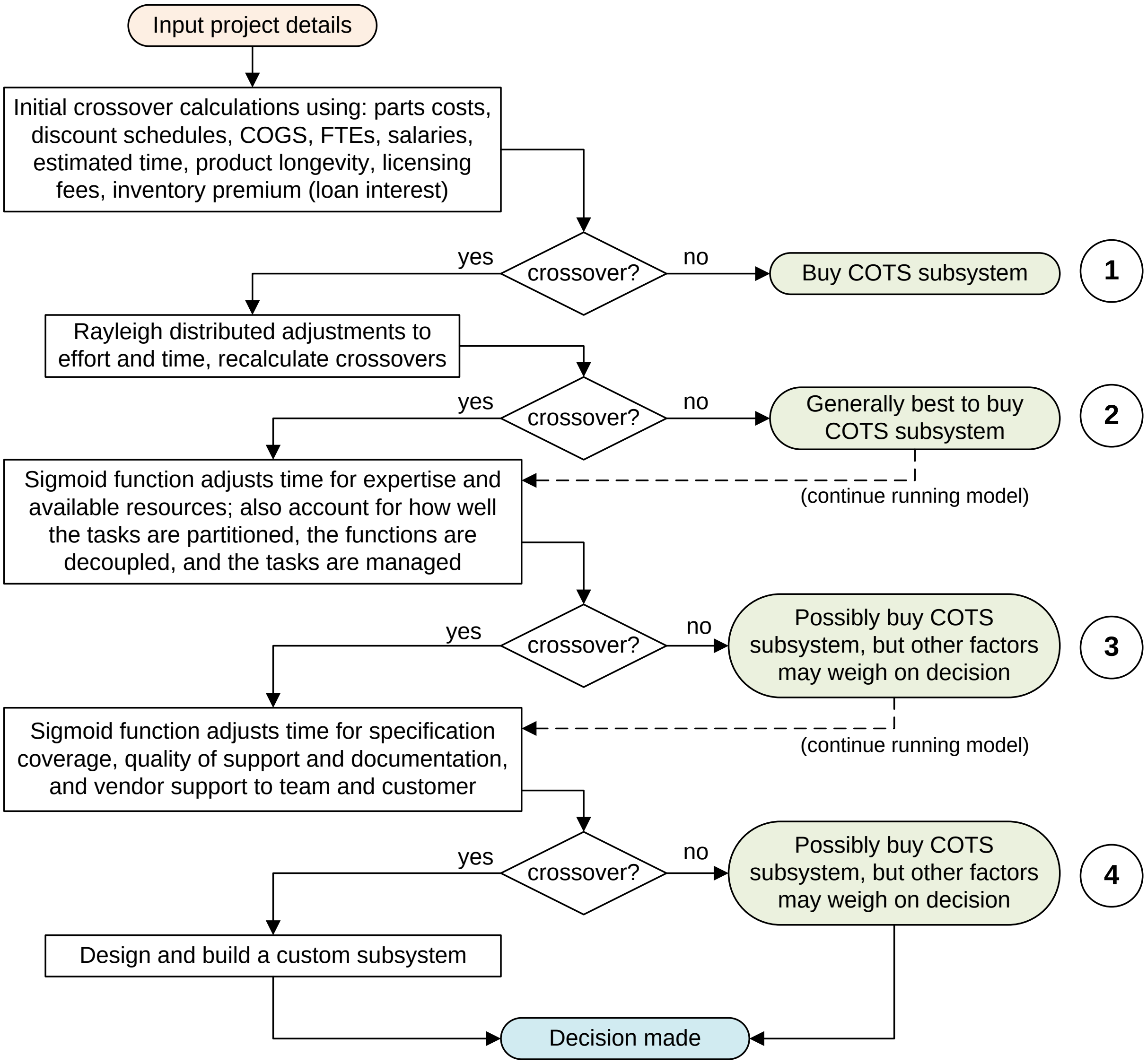

Input project details
Initial crossover calculations using: parts costs, discount schedules, COGS, FTEs, salaries, estimated time, product longevity, licensing fees, inventory premium (loan interest)
crossover?
yes
no
Buy COTS subsystem
1
Rayleigh distributed adjustments to effort and time, recalculate crossovers
crossover?
yes
no
Generally best to buy COTS subsystem
2
Sigmoid function adjusts time for expertise and available resources; also account for how well the tasks are partitioned, the functions are decoupled, and the tasks are managed
(continue running model)
crossover?
yes
no
Possibly buy COTS subsystem, but other factors may weigh on decision
3
Sigmoid function adjusts time for specification coverage, quality of support and documentation, and vendor support to team and customer
(continue running model)
crossover?
yes
no
Possibly buy COTS subsystem, but other factors may weigh on decision
4
Design and build a custom subsystem
Decision made

Figure 5.11 Outline of the model, its operational steps and qualifications, when recommending a build-or-buy decision.

A user first fills in the initial parameters that describe a project, Appendix F has the details of the inputs. Then the initial calculations determine where Eqn. (1) and Eqn. (2) cross, the lost-opportunity cost (which is projected revenue minus projected cost), and the premium for stocking COTS inventory until the end of the lifecycle. The calculations also produce the first cut at the total effort and cost for custom design and separately for integrating COTS. Each of the following stages refines the calculations to estimate the project's cost and schedule and the needs of the development team.

The least-dominant parameters that usually have minimal effect on the outcome of the project are included in these initial calculations because they do not vary much once the specific values are known. These parameters, including licensing fees, support fees, and inventory premiums, adjust the project costs only for the COTS integration. If the product lifetime is 15 years greater than the lifecycle of COTS subsystems, then the premium for inventory or the interest on the loan for a lifetime buy may overwhelm any advantages to incorporating COTS subsystems. If the licensing or support fees are exorbitant, then they may also have the same effect as a very high premium. I assumed an interest of 8% per annum for business loans.

Stage 1:

First, determine if a crossover exists within the full production life of the product. If no crossover exists, meaning that COTS is always cheaper than custom design, then buy and integrate a COTS subsystem. Several other criteria also apply at this stage; buy and integrate COTS if:

- the custom development costs 10 times or more than integrating COTS [241], or
- the opportunity costs are judged too large.

This is the only stage where the decision is nearly certain for most projects if there is no crossover. The one caveat is that lost-opportunity costs can be very subjective when using projections of future revenues.

Stage 2:

Next, use the Rayleigh distribution in Table 5.2 and adjust the total efforts for either building a custom design or integrating a COTS subsystem. Separately multiply each estimate of time by the result of the median plus the offset. The model uses the results from [10] as representative of industry to set percentages.

Development teams with excellent track records for successfully meeting deadlines use the value of 108% to adjust their original estimate of time [10]. Teams that are very successful and have not missed deadlines can use the resulting value as the upper limit on their estimation of time.

Development teams with good track records for meeting deadlines should use the value of 140% to adjust their original estimate of time [10]. Teams that have not missed many deadlines can use the resulting value as the upper limit on their estimation of time.

Teams new to the market or technology or with a much less successful record at meeting deadlines should use 200% to adjust their original estimate of time [10]. They should consider the resulting value as the lower limit on their estimation of time and double or triple their estimated time for development.

If the adjusted estimate of time pushes the crossover point beyond the total, estimated production quantity, then the team would be advised to buy and integrate a COTS subsystem, rather than develop a custom design.

Stage 3:

This stage is the most complex and deals with the parameters that have the greatest effect on the outcome of the project (as seen in the section on parameter sensitivities above). It adjusts the total effort for expertise, available resources, partitioning of tasks, decoupling of functions, and managing tasks efficiently. It uses the sigmoid logistics function to adjust the total effort. For custom design, the factor multiplier of the sigmoid logistics function ranges from one to three; for integrating COTS, the factor multiplier of the sigmoid logistics function ranges from one to one plus the initial ratio of custom development effort to COTS effort. The midpoint ranges from 0.2 to 0.5, while the steepness ranges from 7 to 25.

The model produces four extreme values based on sigmoid curves in Figure 5.5:

- Development teams with excellent track records for successfully meeting deadlines and producing good quality products, will be close to 0 in the sigmoid function of Figure 5.5. Their multiplier is 1.0.
- New teams or teams with recognized deficiencies in expertise or resources will move to the right in Figure 5.5; the value of 40% is used on the abscissa.
- The midpoint and steepness of the sigmoid logistics function will depend on how well the project is managed:
  - A project that has been partitioned, decoupled, and managed well will have a sigmoid function with a midpoint close to 0.5 and a steepness close to 7.
  - A chaotic project will have a sigmoid function with a midpoint close to 0.2 and a steepness close to 25.

Stage 4:

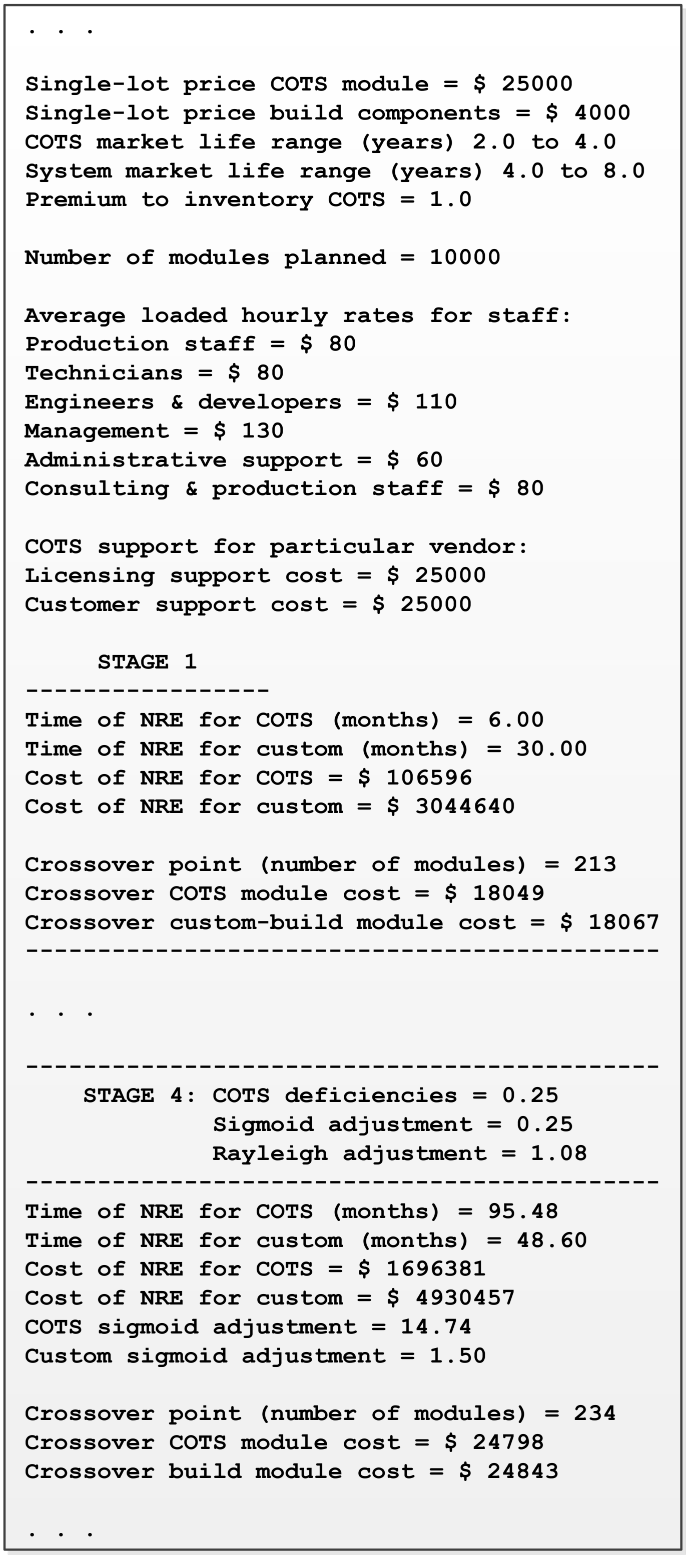
This last stage is also complex but deals with parameters that have lesser effect on the outcome of the project (as seen in the section on parameter sensitivities above). It only adjusts the total effort for the COTS integration and does so based on how well the COTS subsystem meets specifications, the quality of vendor support and documentation, and the extra effort to receive the vendor support. To a lesser degree, it also depends on carefully partitioning tasks and decoupling functions, and efficiently managing tasks. It uses the sigmoid logistics function to adjust the total effort. For integrating COTS, the multiplying factor of the sigmoid logistics function ranges from one to one plus the initial ratio of custom development effort to COTS effort. The midpoint ranges from 0.2 to 0.5, while the steepness ranges from 7 to 25.

The model produces four extreme values based on Figure 5.5:

- High-quality COTS products that meet specifications with good support that does not sap the effort the development team will be close to 0 in the sigmoid function of Figure 5.5. Their multiplier is 1.0.
- Poor-quality COTS products that do not sufficiently cover specifications or have poor support that consumes the effort the development team will move to the right in Figure 5.5; the value of 40% is used on the abscissa.
- The midpoint and steepness of the sigmoid logistics function will depend on the qualities of the COTS subsystem:
    - High-quality COTS products that meet specifications with good support that does not sap the effort the development team will have a sigmoid function with a midpoint close to 0.5 and steepness close to 7.
    - Poor-quality COTS products that do not sufficiently cover specifications or have poor support that consumes the effort the development team will have a sigmoid function with a midpoint close to 0.2 and steepness closer to 25.

Results

I ran this model on the same 18 case studies listed above (and in Table 5.1) to confirm its effectiveness. I combined several case studies because they had different ranges of variation in parameters, which were appropriate for the previous sensitivity analysis but not used in this model. Consequently, I ran the model on 16 case studies. Figure 5.12 illustrates an example of some of the output from the model. Appendix H contains the data generated in the 16 case studies.



```
. . .

Single-lot price COTS module = $ 25000
Single-lot price build components = $ 4000
COTS market life range (years) 2.0 to 4.0
System market life range (years) 4.0 to 8.0
Premium to inventory COTS = 1.0

Number of modules planned = 10000

Average loaded hourly rates for staff:
Production staff = $ 80
Technicians = $ 80
Engineers & developers = $ 110
Management = $ 130
Administrative support = $ 60
Consulting & production staff = $ 80

COTS support for particular vendor:
Licensing support cost = $ 25000
Customer support cost = $ 25000

     STAGE 1
-----------------
Time of NRE for COTS (months) = 6.00
Time of NRE for custom (months) = 30.00
Cost of NRE for COTS = $ 106596
Cost of NRE for custom = $ 3044640

Crossover point (number of modules) = 213
Crossover COTS module cost = $ 18049
Crossover custom-build module cost = $ 18067
--------------------------------------------

. . .

--------------------------------------------
    STAGE 4: COTS deficiencies = 0.25
             Sigmoid adjustment = 0.25
             Rayleigh adjustment = 1.08
--------------------------------------------
Time of NRE for COTS (months) = 95.48
Time of NRE for custom (months) = 48.60
Cost of NRE for COTS = $ 1696381
Cost of NRE for custom = $ 4930457
COTS sigmoid adjustment = 14.74
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 234
Crossover COTS module cost = $ 24798
Crossover build module cost = $ 24843

. . .
```

Figure 5.12 Example of some output from the model that recommends whether to build or buy a subsystem. Each stage gives values for module cost, calendar time, and cost of NRE effort.

Table 5.3 is a snapshot view of some estimates for crossover, calendar time, and total effort in each case study; Appendix H contains the data. Table 5.3 contains the results for a single projected quantity for the planned production for each case study; it does not contain the unit costs at crossover, the NRE costs, or the NRE calendar time for various conditions in Stages 2 through 4.

The yellowish boxes in Table 5.3 indicate no recommendation; other unknown factors need consideration. In Stages 2 and 3, yellow is used when the mean crossovers either straddle the one-half point of production quantity or when the crossovers exceed the one-half point but are also less than full quantity. A red box indicates a “buy” recommendation because the crossover value is greater than the planned production quantity. A green box indicates a “build” recommendation because the crossover value is less than the planned production quantity.

Table 5.3 Recommendations from the Model for a Build-versus-Buy Decision in each of 16 Case Studies.

		Case Studies														
	Parameters	**1**	**2**	**3a**	**3b**	**4**	**5**	**6**	**7**	**8**	**9 to 12**	**13**	**14**	**15**	**16**	**17 to 18**
Custom design inputs	Single-lot cost (US$)	0	0	0	0	$ 4.65	$ 230	$ 230	$ 41	$ 41	$ 400	$ 41	$ 705	$ 900	$ 4,000	$ 43,900
	NRE cost (US$)	$423K	$617K	$2.64M	$2.64M	$250K	$7.97M	$7.97M	$1.82M	$1.82M	$1.82M	$1.95M	$5.02M	$3.04M	$3.04M	$10.9M
	Development time (months)	15	20	48	48	12	48	48	21	21	21	21	36	30	30	60
COTS inputs	Single-lot cost (US$)	0	$ 20	$ 200	$ 500	$ 10.99	$2,530	$ 430	$ 67	$ 175	$ 1,700	$ 175	$ 9,375	$6,000	$25,000	$150K
	NRE cost (US$)	$32.5K	$45.9K	$494K	$494K	$32.8K	$5.63M	$5.63M	$262K	$262K	$262K	$399K	$2.07M	$107K	$107K	$1.78M
	Development time (months)	5	8	15	15	5	42	42	14	14	14	16	24	6	6	18
	License (US$/y)	$2K	$2K	$10K	$10K	0	0	0	0	0	0	0	0	$25K	$25K	0
	Support (US$/y)	0	0	0	0	0	0	0	0	0	0	0	0	$25K	$25K	0
	Premium/total interest	1	1	1	1	3.5	1	2	3.5	3.5	3.5	6	3	1	1	1
	COTS lifecycle (y)	5 to 10	5 to 10	5 to 10	5 to 10	2 to 4	5 to 10	0.4 to 1	2 to 4	2 to 4	2 to 4	2 to 4	5 to 15	2 to 4	2 to 4	5 to 10
	System lifecycle (y)	5 to 10	5 to 10	5 to 10	5 to 10	10 to 20	5 to 10	5 to 10	10 to 20	10 to 20	10 to 20	10 to 30	10 to 30	4 to 8	4 to 8	5 to 10
	Stage 1: initial calculation															
	Crossover (units)	none	28576	10720	4288	178725	1157	4476	13453	3808	311	2046	119	1308	213	135
	Planned production qty. (units)		20000	10000	10000	4000000	10000	10000	20000	20000	20000	40000	5000	10000	10000	500
	Crossover unit cost (US$)	none	$ 32	$ 282	$ 651	$ 58.54	$7,264	$2,132	$ 224	$ 575	$ 6,232	$1,054	$42,833	$3,130	$18,058	$162K
Team capability, multipler	Recommendation	buy	buy	buy												
	Stage 2: Rayleigh median adjustment to NRE cost (US$) for potential schedule over-run															
outstanding, 108%	Crossover (units)				4631	193023	1250	4834	14528	4112	336	2208	129	1412	230	146
	Recommendation															
good record, 140%	Crossover (units)				6003	250215	1621	6300	20705	5313	436	2853	167	1831	298	213
	Recommendation								buy							
new or deficits, 200%	Crossover (units)				8576	357451	2347	9000	29567	7587	661	4063	238	2514	503	305
	Recommendation								buy							
deficit limit, 300%	Crossover (units)				12864	536176	3521	13963	44336	11911	1054	6069	358	3771	754	503
	Recommendation				buy			buy	buy							buy
	Stage 3: Two sigmoid adjustments to NRE cost (US$) for expertise, resources, partitioning, decoupling, and sizing															
	Recommendation does not change from Stage 2 for teams with full expertise and resources for projects that are carefully partitioned, decoupled, and sized.															
outstanding, 108%	Crossover (units), 25% sigmoid				6737	268915	4652	18445	22510	5776	503	3284	250	1817	296	237
	Crossover (units), 75% sigmoid				8947	362611	5489	21807	30607	7854	684	4356	314	2427	411	317
	Recommendation					build		buy	buy	build	build	build	build	build	build	build
good record, 140%	Crossover (units), 25% sigmoid				8733	348593	6195	24610	29171	7485	652	4248	324	2265	384	307
	Crossover (units), 75% sigmoid				11598	470052	7116	28269	39667	10657	886	5629	407	3146	629	411
	Recommendation				buy	build		buy	buy		build	build	build	build	build	
new or deficits, 200%	Crossover (units), 25% sigmoid				12476	497990	8850	35158	41661	11193	931	6046	463	3235	647	438
	Crossover (units), 75% sigmoid				16568	671503	10438	40385	56216	15221	1347	8028	616	4495	1096	644
	Recommendation				buy	build	buy	buy	buy		build	build	build	build	build	buy
deficit limit, 300%	Crossover (units), 25% sigmoid				18715	746986	13631	52737	61994	16785	1485	9054	735	4852	1183	722
	Crossover (units), 75% sigmoid				24853	1007255	15657	60577	84310	24683	2076	12549	924	6491	1644	1024
	Recommendation				buy	build	buy	buy	buy	buy	build	build	build	build	build	buy
	Stage 4: 25% adjustment to COTS NRE cost (US$) for specifications coverage, quality, and support															
	Recommendation does not change from Stage 3 for COTS with full coverage of specifications and good quality of support and documentation.															
outstanding, 108%	Crossover (units), 25% sigmoid				5025	204730	2844	11279	15767	4463	365	2461	173	1440	234	158
	Crossover (units), 75% sigmoid				7235	298427	3536	14021	25438	6527	568	3533	237	2064	350	258
	Recommendation					build	build	buy	buy	build	build	build	build	build	build	build
good record, 140%	Crossover (units), 25% sigmoid				6515	265390	3687	14621	22471	5766	502	3181	224	1866	304	231
	Crossover (units), 75% sigmoid				9379	386849	4584	18175	32967	8459	737	4571	307	2676	535	335
	Recommendation					build	build	buy	buy	build	build	build	build	build	build	build
new or deficits, 200%	Crossover (units), 25% sigmoid				9307	379129	5411	21499	32090	8234	717	4532	320	2563	512	330
	Crossover (units), 75% sigmoid				13399	552642	6727	26725	47084	12650	1119	6506	439	3822	764	525
	Recommendation				buy	build		buy	buy		build	build	build	build	build	buy
deficit limit, 300%	Crossover (units), 25% sigmoid				13961	568694	8117	32248	48121	12928	1144	6772	509	3843	768	544
	Crossover (units), 75% sigmoid				20099	828964	10361	40088	70065	20513	1679	10167	698	5520	1398	788
	Recommendation				buy	build	buy	buy	buy	buy	build	build	build	build	build	buy

🟨 = no recommendation, continue with model

buy = buy recommendation

build = build recommendation

🟨 = no recommendation, find other criteria for the final decision

Discussion

The model's recommendations will change as a function of the planned production quantity. Recommendations generally move towards "build" when the production quantity increases, and they will generally move towards "buy" when quantity decreases. The situation that contradicts this trend is when the COTS subsystems are deficient. Table 5.3 shows that Stage 4 has lower values of crossover than Stage 3 because the deficits in the COTS subsystems take more time and effort to integrate, which balances deficiencies in Stage 3. What the table does not show in Stage 4 is the greater costs and time required over Stage 3.

This model allows users to self-evaluate their own expertise and resources. It gives them a range of potential outcomes from which they may consider and pick a recommendation that will help them make the build-versus-buy decision. There are three reasons for the model not giving a more refined estimate:

- unknown constraints within the user's project,
- defining subjective parameters as precise values, and
- liability.

I will address the last two points.

First, converting subjective evaluations of expertise and available resources into numbers poses huge problems for a computer program. People are biased (i.e., not completely honest) when they consider their own capabilities. Having a computer program or website ask questions obliquely to infer expertise and available resources is a challenging research project, which is beyond the scope of this work. Otherwise, Chapter 6 suggests crowdsourcing and the Delphi method as more objective measures for assessing expertise and available resources. The results from Chapter 6 demonstrate that the difficulties with crowdsourcing and the Delphi method are motivating people outside the project company to participate and when they do participate, they do not always do so in a timely manner.

Second, there is some level of liability in suggesting specific estimates to users. A website that provides a specific recommendation with an estimate of cost, time, and problems could be subject to litigation if it is wrong (or even if it is right, but the user does not want to admit it). Suggesting estimate boundaries reduces the liability because the user still must make the final decision.

The model does not explicitly include reuse of software or hardware, though one could implicitly replace the COTS subsystem with the reused module. The issue of reuse is a separate research topic; there is industry experience that suggests reuse must be carefully handled to avoid making changes that cascade into a complete redesign of the module [206], [242].

Validity

This plug-in module for recommending a build-versus-buy decision is affected to some degree by threats to validity. This module probably suffers less than the estimator plug-in module described in Chapter 4.

Threats to Conclusion Validity

These threats affect the ability to draw a correct conclusion between the independent variable (or treatment) and the dependent variable (or outcome). These threats have to do with the appropriate selection of experiment parameters and implementation [201]. For users of this recommender, the threats to conclusion validity are the inputs of time and effort by users; these require human judgment to make guesses.

Here are remedies and measures that can reduce the threats to conclusion validity:

1. To counter the human judgment in making guesses for time and effort, this plug-in module could implement a sanity checker that provides boundary constraints to flag if the guesses are too low. If a violation occurs, users may go back and revise their inputs upwards.
2. NOTE: a sanity checker is not currently in the present recommender model, it can be added following the same structure of the sanity check used in the estimator plug-in from Chapter 4. To counter the human judgment in preparing the sanity checker, the recommender uses easily counted numbers, instead of more involved estimates of complexity and function:
    - It uses lines of code (LOC), which are easily counted, rather than function points, which tend to be more difficult to estimate [201].
    - It uses numbers of circuit connections to the components and cables, which are easily counted, rather than developing measures of complexity of the circuit boards and cables.

    - It uses numbers of linked mechanical parts, which are easily counted, rather than developing measures of complexity of the mechanisms.
    - It uses numbers of enclosures, which are easily counted, rather than developing measures of complexity of the enclosures.
3. The third plug-in module, which orchestrates a review by the Delphi method in Chapter 6, has as its sole purpose to get independent judgment of the recommender's output.

Threats to Internal Validity

These threats can skew or alter the perspective on the actual relationship between the independent variable (or treatment) and the dependent variable (or outcome) by indicating a casual relation different than actual [201]. For users of this recommender, two threats to internal validity exist. One threat is that of poorly worded instructions in the recommender. The other threat is maturation of users while using the framework multiple times for different projects; they may learn to "game" the tools to give results that look good to them but do not bear a semblance to reality.

The first threat of poorly worded instructions may be countered in two ways. The first way is to have the hosting organization set up a thorough procedure for editing and publishing the recommender. The second way is to run pilot studies and revise the recommender's wording when problems are found.

The second threat of maturation of users who then "game" the tools, the sanity check is a partial answer and will reduce the problem of underestimating to a degree. This is a difficult, if not impossible, problem to solve and afflicts all estimating tools.

Threats to Construct Validity

These threats can skew or alter the perspective on the actual relationship between the experiment setting and the actual setting under study [201]. For users of this recommender, two threats to construct validity exist.

One threat is relying on a mono-method, specifically, a single type of form to calculate time, effort, cost, and number of units at crossover. To counter the mono-method, a separate form with different parameters to provides a "sanity check" of the first recommender. A second counter to the mono-method is the third plug-in module, which orchestrates a review by the

Delphi method in Chapter 6; the review has as its sole purpose to get independent judgment of the recommender, which directly counters the problems of the mono-method.

The other threat is that of users with various experience levels. The separate, "sanity-check" form will "nudge" users towards a more likely result, in spite of different experience levels. Providing a debrief at the end of the project will also "nudge" users to better, initial guesses.

Threats to External Validity

These threats can alter the generalization of the experimental results to industrial practice [201]. For users of this recommender, two threats to external validity exist.

One threat is that the wrong people use the recommender and supply inappropriate or incorrect data. The check for this is when the data from the recommender is presented at the company for project contract negotiations, knowledgeable colleagues should spot the incorrect data. Otherwise, there is no security feature, or lock-out, to allow only specific users of the tool.

The other threat concerns basing recommender operations on bad or "toy" case studies. This is particularly a concern for the default parameters. Ongoing research into time-motion studies for the default values and support by the hosting organization will refine the operation of the recommender and reduce this threat.

The third plug-in module, which orchestrates a review by the Delphi method in Chapter 6; the review has as its sole purpose to get independent judgment of the recommender. Being independent and exercised by experts, the Delphi method will help reduce the impact of threats to external validity.

Justification and Novelty

The novelty of this build-versus-buy recommender is the same as that in Chapter 1. Future research and pilot studies will further justify or refute its operation utility.

Future Work

To access the construct validity, the recommender needs pilot studies to test its claim of speed and no learning curve. Pilot studies of time-motion values in design and develop will also help establish the default values of the sanity checker. Appendix E suggests outlines for pilot studies. Research in game theory may also provide better and more detailed recommendations.

Chapter 6 - Plug-in Module for Human-Centered Advisor

Overview and Motivation

Arguably the most difficult hurdle to overcome in estimating and advising the development plans is removing the underestimation that results from the related, psychological biases of the Planning Fallacy and Prospect Theory. Asking questions in several different fashions, formats and directions and then parsing out the truth in the responses is a very challenging problem in software and beyond the scope of this work. Instead, an independent method that is free of psychological bias is needed.

One avenue of investigation is to turn to independent, outside sources of wisdom and insight for estimating and advising the development of embedded systems. There are two prominent types of those independent, outside sources: crowdsourcing surveys, and the Delphi Method. Crowdsourcing surveys provide a large base of potentially interesting opinions that might converge to a mean response. The Delphi Method provides a more expert view that has the potential for some deep insights.

Either method could provide an independent verification of the results in the previously described plug-in modules for estimation and recommendation in Chapters 4 and 5. Independent verification of the results by either method could be a very effective antidote for threats to construct validity and external validity.

Outline of Studies

The studies in this chapter are for feasibility of both crowdsourcing and the Delphi method in estimating effort for developing embedded systems. The goal of this research is twofold: 1) determine whether unbiased (in the psychological sense, not statistical) mediation and estimates are possible or not, and, 2) determine whether good estimates may derive from minimal, sparse information supplied by the development team or not.

First, I chose two embedded systems, a neurostimulator programmer and an Arc-Fault Detector of which I have personal knowledge, as ground-truth for comparing both methods for potential feasibility in estimating. I establish the effort for each product based on actual experience, which is described in Appendix I. Second, I developed a survey instrument that could be used with both crowdsourcing and the Delphi method. Third, I found channels to distribute the survey and developed a team to participate in the Delphi study. Finally, I ran both studies concurrently.

The basic outline of each study followed these set of activities:

1. Provide instructions and define terms and expectations.
2. Describe the desired function of the embedded system in the case study in less than 300 words.
3. Supply the survey (entire survey is in Appendix J) and ask for estimates of effort (in FTE), calendar time and potential challenges.
4. Collect the results and compare the results to the actual efforts in Appendices I and K.

The goal to use less than 300 words was to maintain a sparse description of each project. The hope is that a sparse description will provide enough information to generate a reasonable set of estimates without revealing company identity or secrets. Both methods ran with anonymous participants to determine their best estimates for the total effort in each of these products.

If the results of either or both methods are statistically close to the actual results, then feasibility is indicated and would warrant further study. Finally, if there are statistically significant numbers of participants in different experience categories in the crowdsourcing survey, then I might be able to compare differences in levels of experience with the accuracy of their estimates.

Case Study 1: Neurostimulator Programmer

Participants in both the survey and the Delphi method received the same explanation:

"Chronic pain in the lower back and legs may be treated by electrical stimulation of the spinal cord, which blocks pain signals traveling up the spinal cord to the brain. Medical personnel program the stimulation through a transmitter that encodes parameters and then couples them through radio frequency to the implant. Adjusting the stimulator for each patient, however, is extremely tedious because billions of possible combinations exist for the parameters' values.

This project is a programmer used primarily by patients in physicians' offices. One of the staff tapes the programmer's antenna over the site of the implant, sets up a session on the programmer, and then hands the programmer to the patient. Patients follow simple instructions *and answer questions posed on the screen; they draw outlines on body outlines where they feel either pain or stimulation effects. The system has a tablet with a mass of less than 0.5 kg, consumes less than 5 W power, and is similar in size to an iPad. It needs to survive for five years (ten years would be preferable) in daily use, five days a week.*

The system is a new, clean-sheet design. The tablet must be either a custom design or available for more than five years from a vendor. The development team has never designed this type of system before. The system is a Class II, 510K device that requires premarket approval or PMA. In "FDA-speak" this means that it must be shown safe and effective through clinical trials before the FDA grants approval for distribution."

The original research took 13 years and culminated in a commercial development that took another six years [243]–[245]. I was a co-founder of a business that undertook the commercial development of the programmer. The business development took 57 months (1998 to 2003), with 12 people expending 439 person-months; this was a new company and all the people were hired for the project; none had previous experience in the design of commercial, medical devices. Should that project be performed again with an experienced team, the time should be shorter, and the effort should be lower as shown in Table 6.1; the learning curve numbers are only for comparison purposes, projects and their development teams can have different values for learning.

Table 6.1 Outline of the Effort and Time Required in Redesigning the Neurostimulator Programmer in Case Study 1.

				Learning Curve Improvement		
Project	**Activity**	**Previous**	**Units**	**10%**	**20%**	**30%**
Neurostimulator Programmer	effort	439	person-months	395	351	307
	time	57	months	51	46	40

To account for advances in technology and gains in experience, three different levels of potential improvement attributed to learning are shown. These are improvements over the actual values in the previous design and development. NOTE: these are for comparison only, projects and their development teams can have different levels of improvement.

Case Study 2: Arc-Fault Detector

Participants in both the survey and the Delphi method received the same explanation:

"Arcing faults are high-impedance short circuits in AC power switching systems. They conduct sufficient current to sustain an arc but remain below the trip threshold of circuit

breakers. Arcs generate white-hot heat that melts and consumes the metal in switchgear in one or two seconds.

The Arc-Fault Detector (AFD) monitors and extinguishes arcing faults within the power switchboards on ships. It uses two different types of sensors to detect arc faults: a photodetector for the light of an arc and a fast-acting pressure sensor to detect the arc's shock wave; once an arc is detected the computer control unit signals the upstream breaker to open and clear the arcing fault. The system must detect arcs but have a miniscule probability of false alarms. Each AFD can monitor up to 100 photosensors and 40 pressure sensors per control unit. Each computer can monitor up to 20 switchboard cabinets.

The AFD system has been through two generations of design updates already. This third-generation AFD will be a new, clean-sheet design. It must be smaller and cheaper. It will run off ship power. The system must meet naval or maritime standards for operation on ships."

The original research and development took 12 years. I was a young engineer during the original development, which took 144 months (1979 to 1991), with many people expending a total effort of 982 person-months [246]. A second phase of development to replace the older technology took place from 1995 to 2002; the redesign time consumed 84 months, with an effort of 447 person-months. Should that project be performed again with an experienced team, the time will be shorter, and the effort will be lower as shown in Table 6.2; the learning curve numbers are only for comparison purposes, projects and their development teams can have different values for learning.

Table 6.2 Outline of Effort and Time Required in Redesigning the Arc-Fault Detector in Case Study 2.

				Learning Curve Improvement		
Project	**Activity**	**Previous**	**Units**	**10%**	**20%**	**30%**
Neurostimulator Programmer	effort	447	person-months	402	358	313
	time	84	months	76	67	59

Improvements are for comparison, teams can have different levels of improvement.

Feasibility Analyses

The definition of potential feasibility for either method was that the mean estimates for both time and effort would be within specified ranges. Another important aspect for potential feasibility for either method would be that the estimates for the standard deviation for each line item is less than 0.5 and that the total standard deviation from total of the ratings is less than 20% of the mean of the 22 ratings.

Feasibility Analysis – Case Study 1: Neurostimulator Programmer

The expected ranges for estimating both time and effort for the Neurostimulator Programmer depends on the learning curve, which is unknown. The FDA approval, in the best case, is usually not less than two years; this means that the time range for development cannot go much below four years.

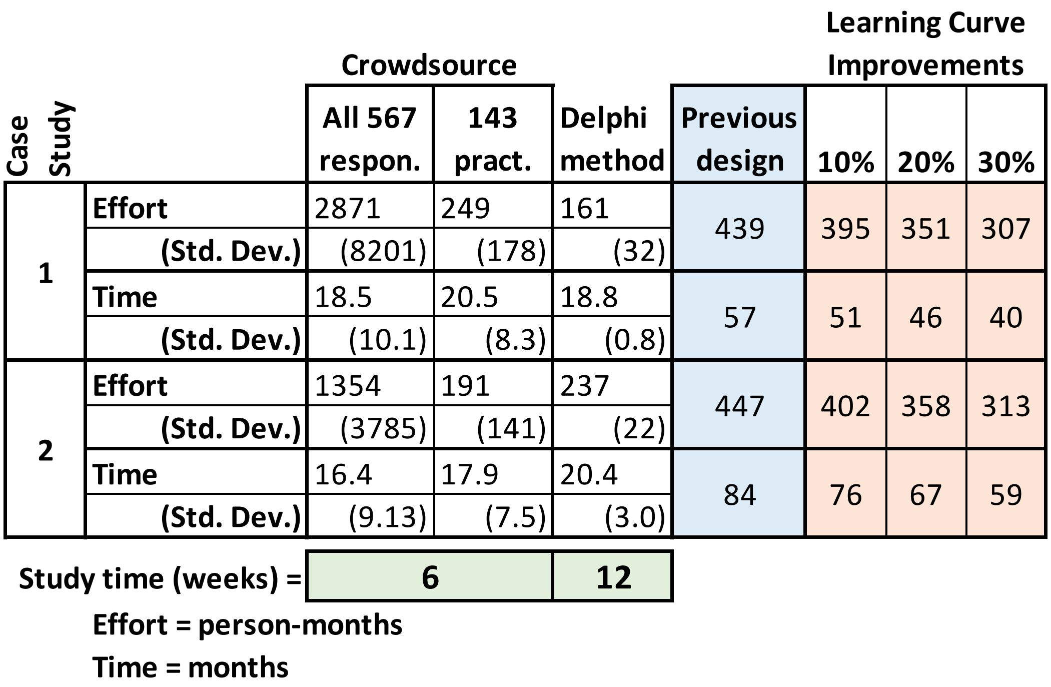
shows a range of improvement from learning. Feasibility of the analyses would be indicated if the results fall within the range between the previous design and the 30% improvement from learning. 0contains the history and calculations for these values.

Feasibility Analysis – Case Study 2: Arc-Fault Detector

The expected ranges for estimating both time and effort for the Arc-Fault Detector depends on the learning curve, which is unknown. I would expect some benefit from learning from the previous generations. This will be offset by the inelastic time for sea trials and evaluation of the new system; evaluation time should be comparable to the original system and should not go much below four years. Table 6.3 shows a range of improvement from learning. Feasibility of the analyses would be indicated if the results fall within the range between the previous design and the 30% improvement from learning. Appendix K contains the history and calculations for these values.

Table 6.3 summarizes the results of both the crowdsource survey and the Delphi method; as mentioned previously, the learning curve numbers are only for comparison purposes, projects and their development teams can have different values for learning. The mean of the general population grossly overestimates effort but the means of both sets of experienced practitioners in the embedded field underestimated effort. Everyone underestimated the time required; both results support the biases of the Planning Fallacy and of Prospect Theory.

Table 6.3 Summary of Results from the Crowdsource Survey and from the Delphi Method.



Case Study		Crowdsource				Learning Curve Improvements		
		All 567 respon.	143 pract.	Delphi method	Previous design	10%	20%	30%
1	Effort	2871	249	161	439	395	351	307
	(Std. Dev.)	(8201)	(178)	(32)				
	Time	18.5	20.5	18.8	57	51	46	40
	(Std. Dev.)	(10.1)	(8.3)	(0.8)				
2	Effort	1354	191	237	447	402	358	313
	(Std. Dev.)	(3785)	(141)	(22)				
	Time	16.4	17.9	20.4	84	76	67	59
	(Std. Dev.)	(9.13)	(7.5)	(3.0)				
	Study time (weeks) =	6		12				

Effort = person-months

Time = months

Crowdsourcing Surveys

The crowdsourcing method was simply a survey sent to both technical and nontechnical participants. The ultimate purpose for using a crowdsourcing survey is to provide a psychologically unbiased but accurate estimate of time, effort and potential challenges in developing an embedded system. For this research, I selected two projects in very different engineering fields, one in medicine and the other in monitoring power systems, to consider feasibility of crowdsourcing estimates of effort. 0provides the entire survey.

I prepared the survey with the Qualtrics survey tool through a license with Kansas State University. Participants were instructed on several points:

- "the survey considers the estimated time and effort expended during design and development. It DOES NOT consider the production, distribution, or support for a project."
- "ASSUME that your estimates target competent teams who will likely complete the work at least 9 times out of every 10 projects."

- “Some projects have five phases or more, additional phases, which may include: separate Concept and Preliminary Phases, a Production Phase, and an Operational Phase. There can be some overlap between adjacent phases.”
- “Description of each project is intentionally sparse; we are studying the feasibility of estimation with very abbreviated descriptions.”
- “FTE – full-time equivalent, which is the equivalent number of persons devoted full-time to a project. Here are some examples:”

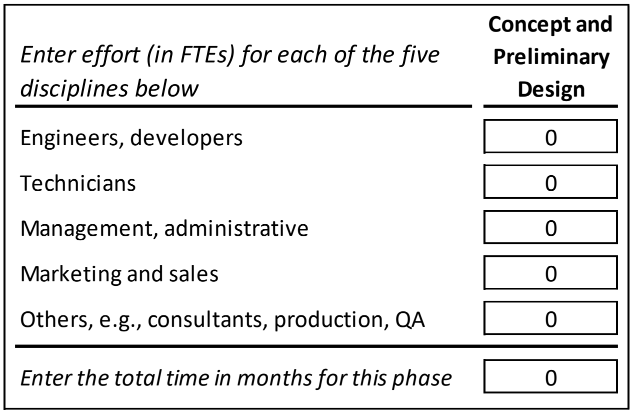
I assured participants that the survey was voluntary and that they could stop at any time. I also clearly stated that all responses were anonymous and would be confidential.

In both case studies, participants filled in numerical values that represented either FTEs (full-time equivalents) for staffing or number of months for time (Figure 6.1). Participants entered their estimates of potential difficulties or challenges by pressing radio buttons, one for each row, that represent the level of challenge (Figure 6.2 has a portion of the 22 rows in the survey that use a Likert rating criterion):

- 1 = **Low.** Effort, cost, and technology are understood and controlled; e.g., system upgrade to a servomechanism that does not require software changes and only minor differences in the motor and drive.
- 3 = **Medium.** Effort, cost, and technology are understood; e.g., clean-sheet design of a servomechanism that uses available components.
- 5 = **High.** Effort, cost, and technology are new to the team; e.g., clean-sheet design of a spacecraft instrument with new software and autonomy.

Distribution for the survey went through different channels. Amazon gift cards for 10 randomly selected participants provided motivation to complete the survey. I distributed the survey through four different venues:

- EmbeddedRelated newsletter on February 1, 2018
- Barr Group newsletter, “Firmware Update,” and on Twitter on February 8, 2018
- Jack Ganssle’s newsletter, “The Embedded Muse,” Issue No. 344, February 19, 2018
- Emailed directly to colleagues in industry through Linked-In and asked them to forward the survey.



Enter effort (in FTEs) for each of the five disciplines below	**Concept and Preliminary Design**
Engineers, developers	0
Technicians	0
Management, administrative	0
Marketing and sales	0
Others, e.g., consultants, production, QA	0
Enter the total time in months for this phase	0

Figure 6.1 General form of effort and time entry; this format repeats for the Critical Design Phase and for the Test/Integration Phase.

The Delphi Method Study

The Delphi study used a virtually identical survey. Both sets of participants received the same explanations for the two case studies. The ultimate purpose for using the Delphi method study is to provide an unbiased but accurate estimate of effort and potential challenges in developing an embedded system. For this research, I selected the same two projects as the crowdsourcing survey. 0provides the entire survey used repeatedly in the Delphi method study.

Here is how the study runs:

- Prepare the questions
- Invite and select 5 to 7 participants with expertise in embedded systems. Experts are qualified by at least 20 years of relevant experience and being a senior or chief engineer or project/program manager in embedded systems. Also, a mix of software and electronic hardware background will be important.
- Provide the following instructions:
    1. Give the background for the study
    2. Keep participants anonymous to avoid deference and bias from knowing background and accomplishments of the participants.
    3. Participants fill in answers to the same crowdsourcing survey questions.
    4. Collect the survey results, questions, comments, and rationale
    5. Summarize and post the survey results with the questions, comments, and rationale that have been made anonymous.

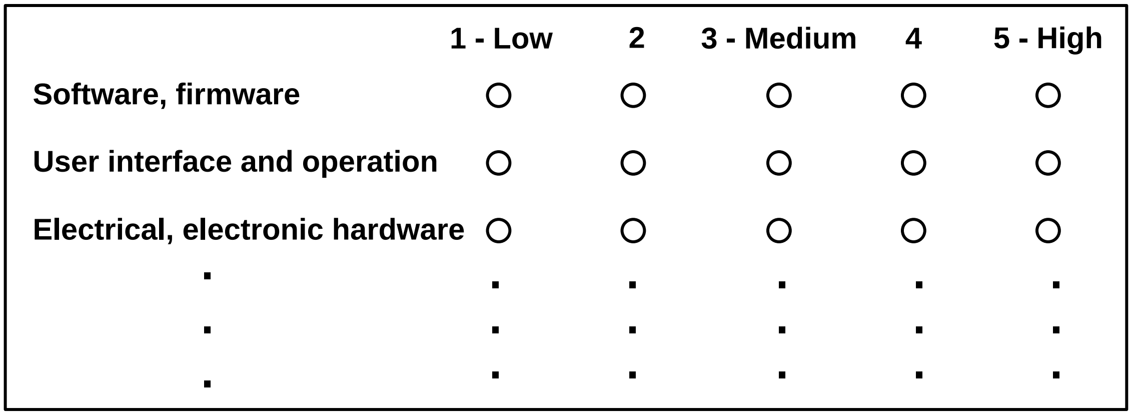
6. Participants then fill in the same crowdsourcing survey questions but with updated answers.
7. For each iteration, repeat steps 3 through 6 until either the standard deviation is low or we complete five iterations.

- On each iteration give each participant the anonymous answers and the ensuing anonymous discussion.
- Stopping criteria is either standard deviations of less than ± 0.5 on the challenges facing teams or completion of iteration 5.

I invited eight industry experts to participate in the Delphi study of both case studies, of which five responded and participated. The identities of the five participants were kept anonymous to each other; only I, as the moderator, knew who they were. The biographies found in Appendix K were randomized and do not correspond to the participant numbers in other figures.

I distributed to each participant the same information, in the same order and format as was found in the survey. Participants could comment on each case study at any length and send the remarks to the moderator; after I reviewed them and made any edits to ensure anonymity, I broadcast them to the group.

For each round, I distributed the surveys in a Word file with embedded Excel spreadsheets. The participants completed the files and returned them to the moderator for compilation.



	1 - Low	2	3 - Medium	4	5 - High
Software, firmware	○	○	○	○	○
User interface and operation	○	○	○	○	○
Electrical, electronic hardware	○	○	○	○	○
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.

Figure 6.2 Format of radio buttons to select challenges foreseen in the project. Likert scale: 1 = low effort, 3 = medium effort, 5 = high effort.

Analysis

I set several criteria for the lengths of the studies to generate data that might be statistically significant. I planned for 800 or more completed surveys or close after six weeks. The Delphi study had several different criteria that could be met to end the study: either converge to a consensus represented by 0.50 standard deviation in the all the answers to the forecasted problems or a cumulative sum less than 7.0 for the standard deviations or have a standard deviation of less than 5.0 on the estimates of time or effort, or if no convergence, then complete five rounds.

For the survey results, I calculated the mean and standard deviation for the summed estimates of effort and time from all the respondents and then repeated the same calculations only for people experienced in developing embedded systems.

For each case study within the Delphi method, I calculated the mean and standard deviation for the summed estimates of effort and time in each round of deliberation from the five participants. I then plotted the estimates for effort and time from each participant over the rounds of the study; these plots graphically indicated the trends towards convergence.

Results and Discussion

Crowdsourcing Survey Results

I collected data results from the survey from February 1 through March 18, 2018. I received 567 useable responses to the survey over six weeks in February and March 2018. I received responses from around the world:

- Africa = 11
- Asia = 81
- Australia, New Zealand = 12
- Central, South America = 26
- Europe = 150
- North America (USA and Canada) = 271

These responses total to 551, not 567, because the survey allowed people to skip questions. 459 respondents indicated they participated in the management, design, development, integration, or manufacture of embedded systems, such as electronic devices, appliances, devices, or equipment; of these 368 were directly involved in design and development; the next-largest group comprised 48 academics. Respondents indicated that they were spread across the spectrum of industries from aerospace to agriculture to process industries to medical devices to transportation. 145 respondents had more than 20 years of experience, and 111 had 10 to 20 years of experience. Only 62 worked in an environment that used design contracts, while another 68 worked with informal agreements; otherwise, most did not use project contracts. Table 6.3 summarizes the results.

The survey results from all the respondents were as follows:

- Case Study 1, Neurostimulator Programmer:
  - Effort: mean = 2871 (std. dev. = 8201) person-months
  - Time: mean = 18.5 (std. dev. = 10.1) months
- Case Study 2, Arc-Fault Detector:
  - Effort: mean = 1354 (std. dev. = 3783) person-months
  - Time: mean = 16.4 (std. dev. = 9.13) months

After reviewing the huge differences in responses, I then selected only people directly involved in developing embedded systems. From this group, I eliminated erroneous responses that were a factor of three greater than those in Appendix I (any FTE > 30 for a single category in a phase – we had several in the thousands) and any calendar time set at 0. The results for the remaining 143 practicing respondents were as follows:

- Case Study 1, Neurostimulator Programmer:
  - Effort: mean = 249 (std. dev. = 178) person-months
  - Time: mean = 20.5 (std. dev. = 8.3) months
- Case Study 2, Arc-Fault Detector:
  - Effort: mean = 191 (std. dev. = 141) person-months
  - Time: mean = 17.6 (std. dev. = 7.5) months

When compared to the original projects (Appendix I), the survey results from practitioners were low. They were 25% low for effort and 57% low for time in Case Study 1. They were 45% low for effort and 71% low for time in Case Study 2.

The ratings for 21 different, potential problems or concerns or challenges within project development were not definitive. Figure 6.7 and Figure 6.8 show the ratings of respondents directly involved in embedded systems; even these are not conclusive. Verification, validation, compliance (#8) was one area that most respondents seemed to think would represent the greatest challenge; it was followed closely by software (#1), user interface (#2), electronics (#3), cost of development (#13), time to market (#14), and changing requirements (#15).

The survey results forecast the challenges to be high, at a mean difference of 0.52 above the original project with a standard deviation of 1.02, for Case Study 1. The results were high, at a mean difference of 1.24 above the original project with a standard deviation of 0.95, for Case Study 2.

The Delphi Method Results

The five participants in the Delphi study are listed in Appendix K. The study ran for 12 weeks from mid-February to the end of April 2018. Table 6.3 summarizes the results. The participants' answers converged for both case studies within four rounds of the study. Figure 6.3 through Figure 6.6 show the convergence.

- Case Study 1, Neurostimulator Programmer:
  - Effort: mean = 161 (std. dev. = 32) person-months
  - Time: mean = 18.8 (std. dev. = 0.8) months
- Case Study 2, Arc-Fault Detector:
  - Effort: mean = 237 (std. dev. = 82.1) person-months
  - Time: mean = 20.4 (std. dev. = 3.0) months

When compared to the original projects (Appendix I), the survey results from practitioners were low. They were 51% low for effort and 61% low for time in Case Study 1. They were 32% low for effort and 66% low for time in Case Study 2.

The ratings for 21 different, potential problems or concerns or challenges within project development converged, also. Figure 6.7 shows the ratings from the five participants in Case Study 1. Verification, validation, compliance (#8) was one area that all participants thought would represent the greatest challenge; it was followed closely by software (#1) and time to market (#14) in Case Study 1. Figure 6.8 shows the initial and final ratings for both case studies. Case Study 2 had different primary concerns: Extreme environments (#11) was most important, followed by verification, validation, compliance (#8) and electronics (#3).

The results from the Delphi study forecast the challenges to be low, at a mean difference of 0.46 below the original project with a standard deviation of 1.02, for Case Study 1. The results were high, at a mean difference of 0.51 above the original project with a standard deviation of 0.97, for Case Study 2.

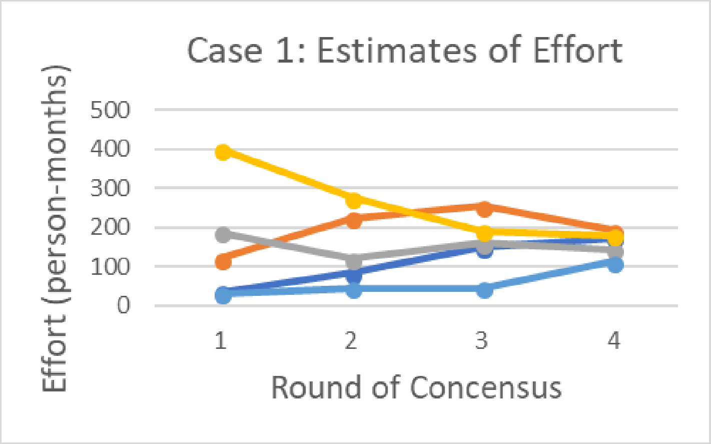


Figure 6.3 Individual trends for estimating the amount of effort required to implement Case Study 1 in the Delphi Study.

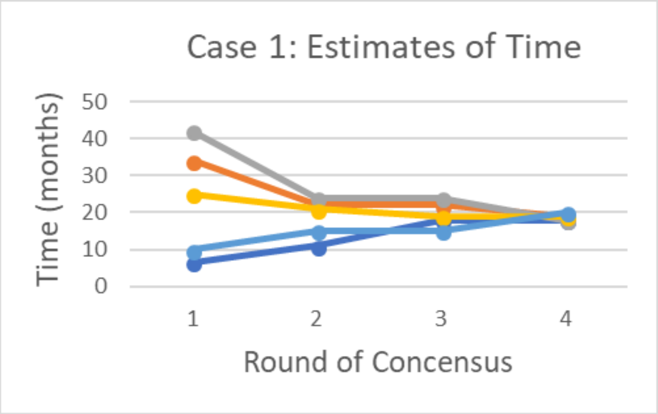


Figure 6.4 Individual trends for estimating the amount of time required to implement Case Study 1 in the Delphi Study.

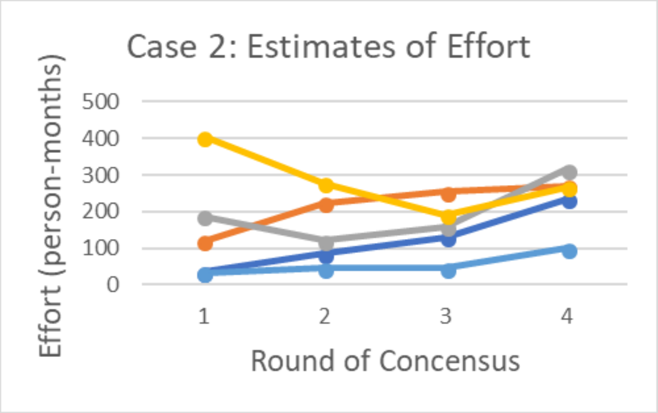


Figure 6.5 Individual trends for estimating the amount of effort required to implement Case Study 2 in the Delphi Study.

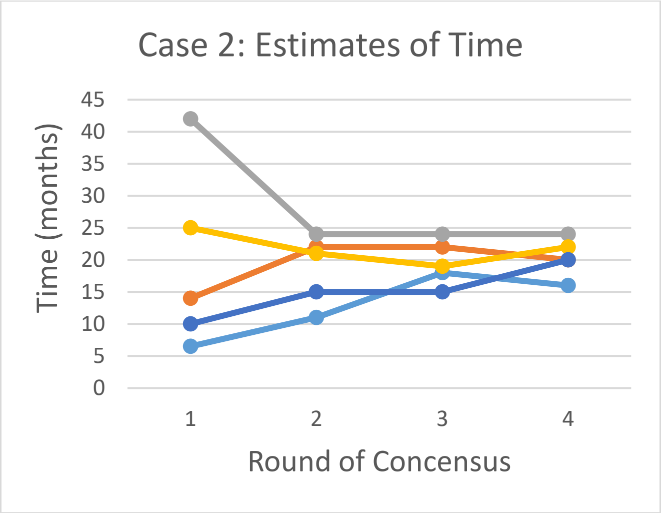


Figure 6.6 Individual trends for estimating the amount of time required to implement Case Study 2 in the Delphi Study.

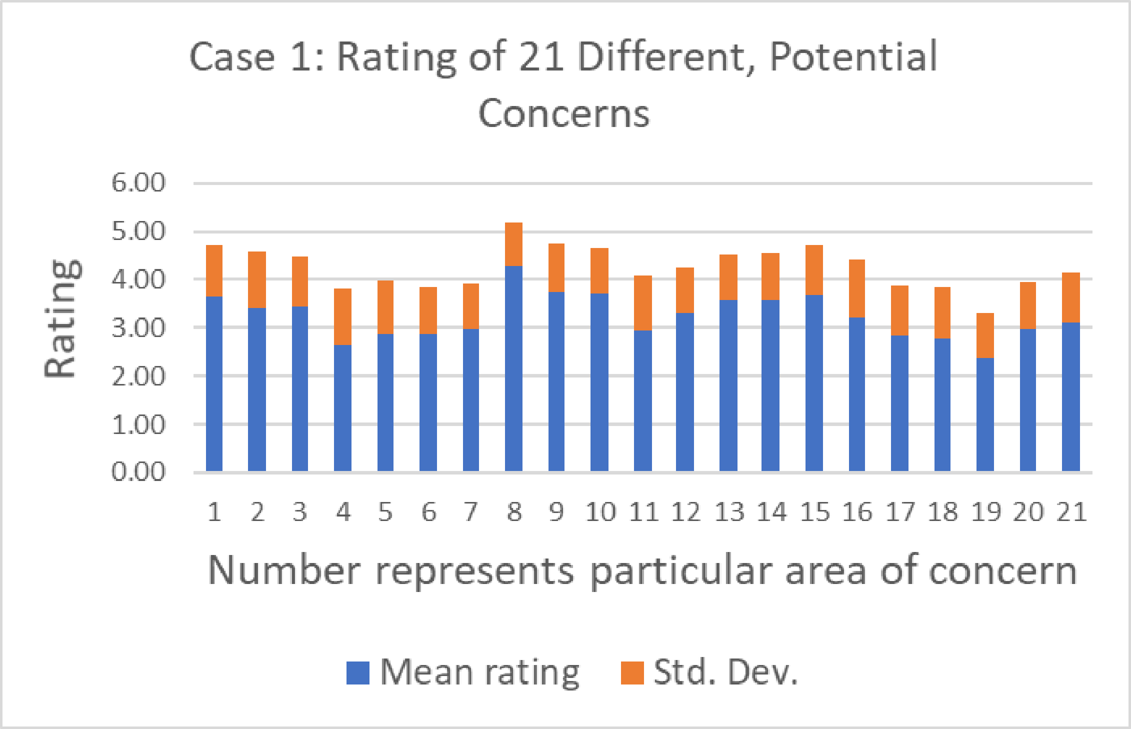


Figure 6.7 Practitioners' ratings for potential concerns to project development for Case Study 1 in the Delphi study.

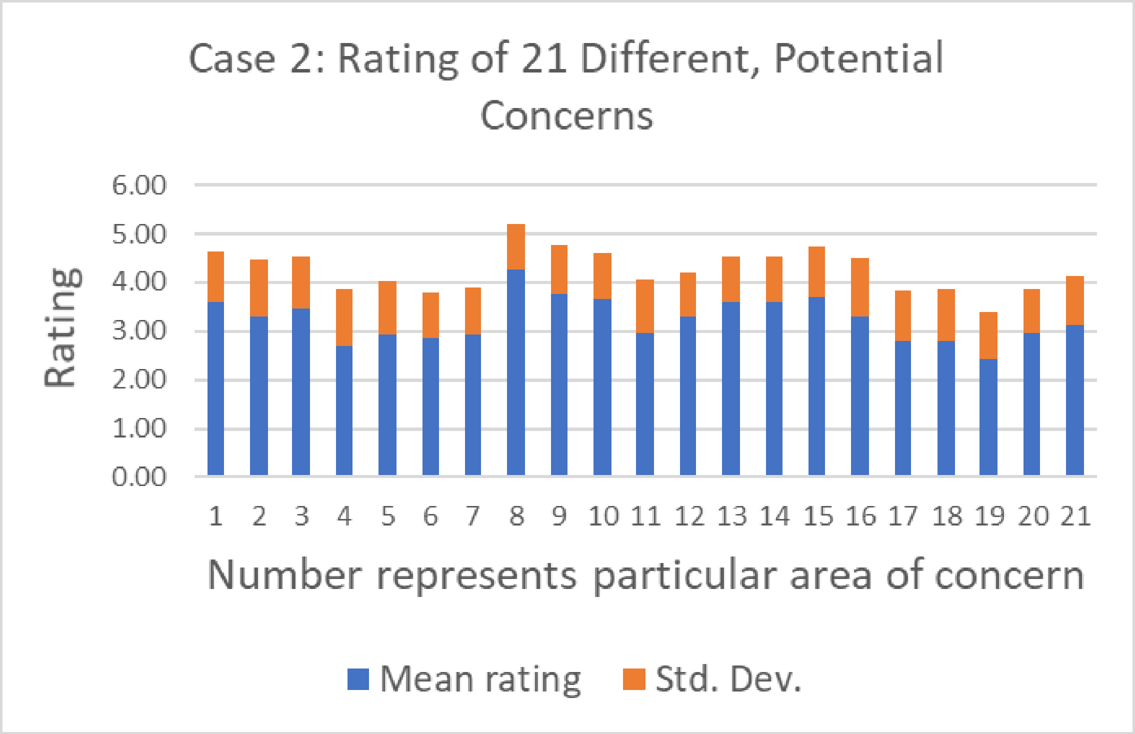


Figure 6.8 Practitioners' ratings for potential concerns to project development for Case Study 2 in the Delphi study.

Discussion

Both studies took a long time to complete: the survey ran for six weeks; the Delphi study ran for 12 weeks. Delayed results in estimation at the beginning of projects is usually unacceptable.

Many of us, including experts, estimate low for effort and time as predicted by both the Planning Fallacy and Prospect Theory. These results are borne out in so many projects that overrun schedule and budget. These results do argue for an automated method for estimating effort and time, such as the parametric models in COCOMO and COSYSMO [155], [158], [162], [163], [204], [205].

Underestimating in Case Study 1, the medical project, may have resulted from some combination of the following: 1) only two of the five experts had experience in the medical industry, and 2) the participants may not have realized that FDA approval was part of the test-and-integration phase, which should add about two years to the time, plus additional staff effort to monitor the clinical testing. Nearly the same reason may be given for underestimating in Case Study 2; participants may not have realized that technical evaluation onboard a ship was part of the test and integration phase, which could add several years to the time and effort.

This study showed that the Delphi method was better at estimating and forecasting than a crowdsourced survey for these two case studies. The study shows feasibility of the Delphi method, which parallels the literature in Chapter 2. A potential follow-on Delphi study would be to repeat both case studies with experts familiar with each industry ─ medical and military, maritime ─ and note whether the results are closer to reality.

The study does not, however, demonstrate conclusively that either crowdsourcing or the Delphi method is ultimately better suited for estimating and forecasting in technical projects. Swarm intelligence, which combines crowdsourcing with the Delphi method and allows revisions of views over several iterations, might be the most-effective method [46], [50]; one problem, though, is that of motivating a large group of people to remain involved throughout the activity. The successful Foldit project made a game of the problem of protein folding and configuration [247]; this sort of game then becomes one of developing a well-posed problem.

Conclusion

Both methods generate results slowly. The crowdsourcing survey is not accurate enough to be generally useful. If time allows, the Delphi method might give a reasonable estimate of time and effort, but participants should be experts in the field; follow-on study would be needed to test this possibility.

The brief description that leads to reasonable estimates seems feasible, which is important since Delphi participants do not desire to spend much time in deliberating projects outside their company. Brevity would be necessary for a study of swarm intelligence, with the added problem of motivating participants to stay through multiple iterations over several weeks.

Validity

This human-centered advisor can be affected to a small degree by threats to validity. This module suffers fewer threats to its validity than the estimator plug-in module described in Chapter 4.

Threats to Conclusion Validity

These threats affect the ability to draw a correct conclusion between the independent variable (or treatment) and the dependent variable (or outcome). These threats have to do with the appropriate selection of experiment parameters and implementation [201]. For users of this human-centered advisor, the threats to conclusion validity are the inputs of time and effort by users; these require human judgment to make guesses.

The primary remedy that can reduce the threats to conclusion validity is that this human-centered advisor provides an independent assessment of the estimates derived from values supplied by users. It still has the concern that the anonymous participants must use their own human judgment in making guesses for time and effort; the hope is that they are impartial to the project and might be more experienced in that type of embedded system.

Threats to Internal Validity

These threats can skew or alter the perspective on the actual relationship between the independent variable (or treatment) and the dependent variable (or outcome) by indicating a casual relation different than actual [201]. For users of this human-centered advisor, two threats to internal validity exist. One threat is that of poorly worded instructions to the human-centered advisor. The other threat is that of a poorly worded description of the potential project. Hopefully, the expert participants will accommodate poorly worded instructions and descriptions and still arrive at a reasonable assessment.

Threats to Construct Validity

These threats can skew or alter the perspective on the actual relationship between the experiment setting and the actual setting under study [201]. For users of this human-centered advisor, two threats to construct validity exist.

One threat is relying on a mono-method, specifically, a single description and set of supplied values to calculate time, effort and cost. To counter the mono-method, separate plug-in modules, specifically the estimator and the recommender, provide "sanity checks" of the human-centered advisor.

The other threat is that of users with various experience levels. Hopefully, the expert participants will accommodate users with various levels of experience and still arrive at a reasonable assessment.

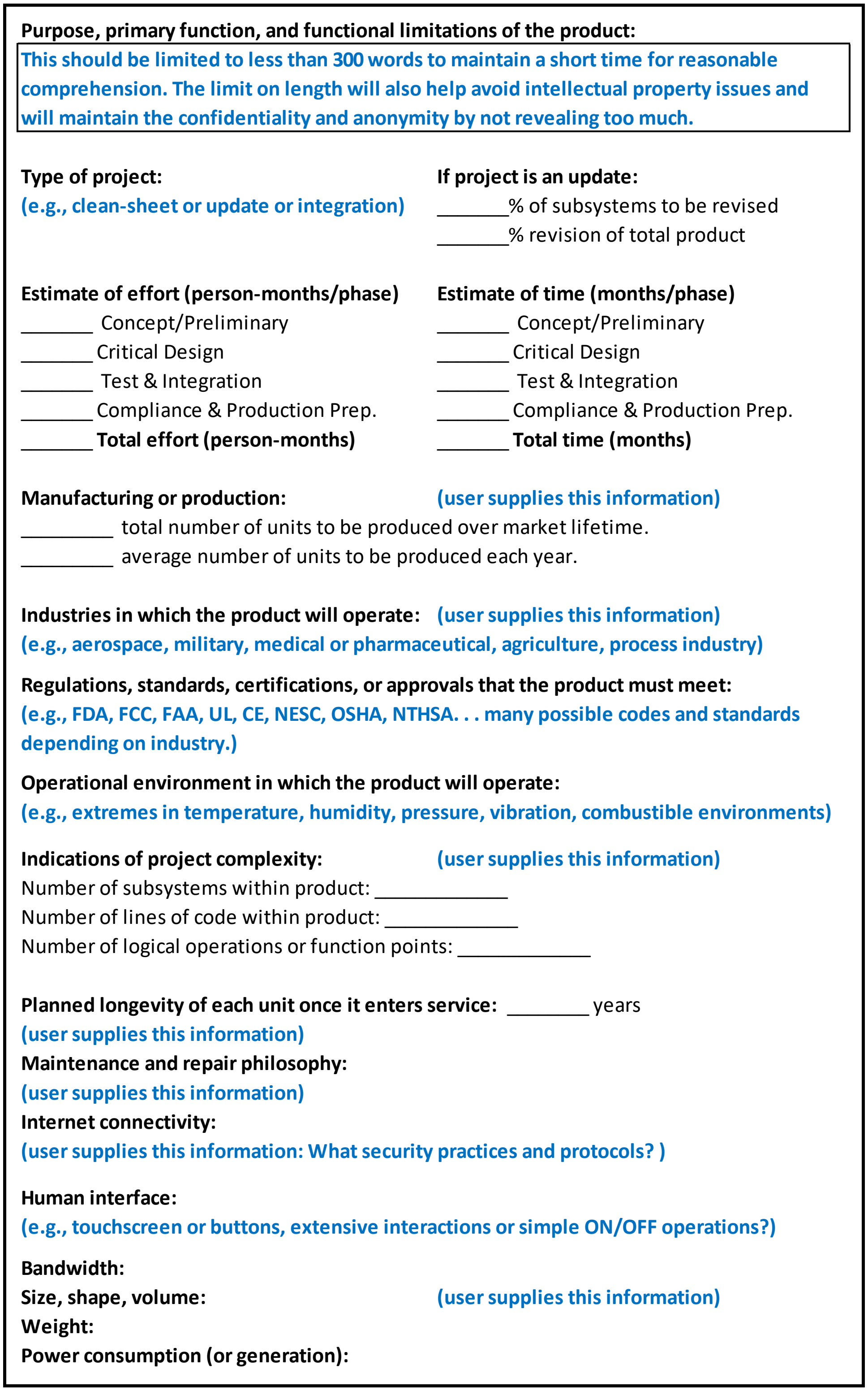
Threats to External Validity

These threats can alter the generalization of the experimental results to industrial practice [201]. For users of this human-centered advisor, two threats to external validity exist.

One threat is that the wrong people use the human-centered advisor and supply inappropriate or incorrect data. The check for this is when the data from the human-centered advisor is presented at the company for project contract negotiations, knowledgeable colleagues should spot the incorrect data. The estimator and the recommender modules could provide "sanity checks" of the human-centered advisor. Otherwise, separate plug-in modules, specifically the estimator and the recommender, provide "sanity checks" of the human-centered advisor.

Potential Format of a Sparse Description

Figure 6.9 illustrates one potential input page for delivering a sparse description to the framework. Users can fill out whatever amount of information that they wish to give. Appendix L has some other potential webpages for a tool that could host the framework.



Purpose, primary function, and functional limitations of the product:

This should be limited to less than 300 words to maintain a short time for reasonable comprehension. The limit on length will also help avoid intellectual property issues and will maintain the confidentiality and anonymity by not revealing too much.

Type of project:
(e.g., clean-sheet or update or integration)

If project is an update:
_______% of subsystems to be revised
_______% revision of total product

Estimate of effort (person-months/phase)
_______ Concept/Preliminary
_______ Critical Design
_______ Test & Integration
_______ Compliance & Production Prep.
_______ **Total effort (person-months)**

Estimate of time (months/phase)
_______ Concept/Preliminary
_______ Critical Design
_______ Test & Integration
_______ Compliance & Production Prep.
_______ **Total time (months)**

Manufacturing or production: (user supplies this information)
_________ total number of units to be produced over market lifetime.
_________ average number of units to be produced each year.

Industries in which the product will operate: (user supplies this information)
(e.g., aerospace, military, medical or pharmaceutical, agriculture, process industry)

Regulations, standards, certifications, or approvals that the product must meet:
(e.g., FDA, FCC, FAA, UL, CE, NESC, OSHA, NTHSA. . . many possible codes and standards depending on industry.)

Operational environment in which the product will operate:
(e.g., extremes in temperature, humidity, pressure, vibration, combustible environments)

Indications of project complexity: (user supplies this information)
Number of subsystems within product: _____________
Number of lines of code within product: _____________
Number of logical operations or function points: _____________

Planned longevity of each unit once it enters service: ________ years
(user supplies this information)
Maintenance and repair philosophy:
(user supplies this information)
Internet connectivity:
(user supplies this information: What security practices and protocols?)

Human interface:
(e.g., touchscreen or buttons, extensive interactions or simple ON/OFF operations?)

Bandwidth:
Size, shape, volume: (user supplies this information)
Weight:
Power consumption (or generation):

Figure 6.9 Potential input page for a sparse description of the embedded system.

Final Thoughts and Future Research

Other studies have compared crowdsourcing with the Delphi method. Most of them look for a single-point solution. There is no literature that compares the results of a crowdsource survey with the Delphi method for estimating time, effort, cost, and potential challenges for developing embedded systems.

The crowdsourcing appears not feasible when looking at outputs of different parameters simultaneously. Experienced people, on the other hand, tend to make multi-parameter decisions all the time and do so fairly well. The one concern in this study of the Delphi method is that people with expertise in different backgrounds in developing embedded systems participated. The immediate follow-up research would be to repeat this study but split the two case studies so that only experts in medical device development would work on Case Study 1 and only experts in military or maritime development would work on Case Study 2.

Chapter 7 - Conclusions

Introduction

This work shows that the framework should be east to use, not require learning new techniques, be inexpensive, be accessible, and motivate teams to pursue good practices. It should be extensible to receive new and different modules with specific, complementary functions. This work has also opened several questions; most importantly, how to demonstrate the trustworthiness of the boundary constraint. The work suggests several pilot studies and time-motions studies would be very useful in establishing the confidence in the boundary constraint.

There may be extensions from this work beyond developing embedded systems. Applications to appliances may be simple extensions to this work. Vehicles and smart buildings may demand considerably more work to extend the results in this work to their applications.

How Well the Framework Works

The framework is designed to accept plug-in modules to perform specific tasks: accept project inputs, an unbiased advisory panel (using the Delphi method), schedule and budget estimator, and advise the build-versus-buy decision.

Crowdsourcing surveys are not a reasonable tool for providing estimation of time, effort, and technical challenges. It is too slow and has too much variance in its answers.

The Delphi method might be a good tool for estimating time, effort, and technical challenges. It is was slow and the first study was low because not everyone was an expert in the technical fields for both case studies. The Delphi method bears further study with experts in the same technical field as the case study. Follow-on studies are being prepared at this writing.

I suggest that the Delphi method be used with paid consultants to speed up its results. One potential format would be for clients to pay a flat US$1,400 fee for five consultants to operate within the Delphi method. Each consultant would receive a $200 payment, as would the moderator and the hosting organization. A technical society would be well positioned for this type of work.

The modules in this work have been tested against case studies, 18 for advising the build-versus-buy decision and 10 for estimating schedule and budget. The models do not give single-point solutions, but rather a set of potential solutions, one of which clients may choose to match to their projects. These results are encouraging but not encompassing for the entire embedded systems field. More case studies should be identified and run against the models to confirm model utility, or adjust the models, or reject them outright for specific markets.

The framework is very much a work in progress and much more research can be done on developing its structure and function. It needs to be hosted as a website and run in a pilot study.

The "Nudge"

The clear need is for a method or tool that encourages a more rigorous approach to designing and developing products. Following the groundwork laid by Kahneman and Tversky, Atul Gawande, and Richard Thaler, any usable method must have these characteristics:

- Accounts for human biases [24], [27], [35], [38],
- Is easy to use and does not require learning new techniques [183],
- Fast and cheap, and
- Provides a "nudge" or "prompted choice" [199].

A new method or tool must have these characteristics to motivate use and to make it nearly irresistible to the engineering and business community. Part of the research in this work is to find or develop a "nudge" that makes using good practices something that development teams want to do. The "nudge" must provide value-added utility to each project.

Synergism Between the "Nudge" and the Framework

Using the tool and following its suggestions will significantly reduce the impact of human biases in developing embedded projects. Making the framework a web-based tool that is simple – fill-in boxes, answer questions, clear navigation – will make it fast and easy to use and it will not require learning any new techniques.

The last part is the challenge: cheap and a "prompted choice" or "nudge." I propose that a technical society host the web-based tool for free. The tool will accept a description of a project and generate a set of estimates, as done in the model plug-ins in Chapters 4 and 5. This allows the user to quickly decide between four potential outcomes which most closely matches their situation. It reduces liability to the technical society hosting the tool. The tool can learn and build its knowledge base with anonymous case studies that are useful but do not remove competitive advantage from the submitter's company because they will become available after the fact and only if the submitter’s company agrees. Clients would need to submit an anonymous debriefing of the tool’s utility in the project after the project finishes. Again, this debriefing would be simple and easy to use, taking little time from the client.

But why would someone still want to use it? I suggest that the technical society may authorize a seal of approval, much like a UL stamp or CE marking, that may accompany the product into the marketplace. It will indicate that the company implemented good practices, such as ISO 9000 or CMMI (Capability Maturity Model Integrated), and therefore, has a product more likely to function correctly and safely. It’s a reputation builder for the client that does not cost the company to apply it but requires the company to use the tool and indicate that it complied with known good practices.

Further Research with Pilot Studies

Pilot studies in the operation of the framework would establish basic bounds and confidence intervals in ease and timeliness. Time-motions studies would be very useful in establishing the confidence in the boundary constraint; these would be most useful in establishing baseline values for developing designs of circuit connections, mechanical linkages and connections, and enclosures. Appendix E suggests outlines of potential studies to set these values.

Other research includes surveys of more case studies for confirming or changing the recommender model.

Further Research into Sensitivity Analysis

While the eFAST algorithm adequately analyzes parameters for sensitivity, it is slow. My studies with Dr. Sanjoy Das may lead to faster methods to analyze for sensitivity. Currently, we are investigating Fast Fourier Transform (FFT) methods but results are too preliminary to be definitive.

References

[1] J. Ganssle and M. Barr, "Chapter E," in *Embedded Systems Dictionary*, CMP Books, 2003, pp. 90–91.

[2] "Electric Vehicle Recall." [Online]. Available: http://autorepair.about.com/library/recalls/ ; found in 2006, website no longer available.

[3] "National Highway Traffic Safety Administration." [Online]. Available: https://www-odi.nhtsa.dot.gov/.

[4] "Consumer Product Safety Commission." [Online]. Available: https://www.cpsc.gov/Recalls/2016/.

[5] P. Mozur, "Galaxy Note 7 Fires Caused by Battery and Design Flaws, Samsung Says," *New York Times, Business Day*, New York, NY, USA, p. B1, 22-Jan-2017.

[6] B. Vlasic and N. E. Boudette, "Self-Driving Tesla Was Involved in Fatal Crash, U.S. Says," *New York Times, Business Day*, New York, NY, USA, p. A1, 30-Jun-2016.

[7] "Ariane 5 Loss Avoidable with Complete Testing," *Aviation Week & Space Technology*, pp. 55–57, Sep-1996.

[8] "Ariane 5 Report Details Software Design Errors," *Aviation Week & Space Technology*, pp. 79–81, Sep-1996.

[9] N. G. Leveson and C. S. Turner, "An Investigation of the Therac-25 Accidents," *IEEE Comput.*, no. July, pp. 18–41, 1993.

[10] R. Quinnell, "Embedded development survey reveals trends, concerns for engineers," *EDN*, 2015. [Online]. Available: http://www.edn.com/electronics-blogs/now-hear-this/4439697/Embedded-development-survey-reveals-trends--concerns-for-engineers?_mc=NL_EDN_EDT_EDN_today_20150615&cid=NL_EDN_EDT_EDN_today_20150615&elq=1ef5cd79c21e494198627382bbdbef6f&elqCampaignId=23445&elqa. [Accessed: 15-Jun-2015].

[11] J. Ganssle, “Managing Embedded Projects,” *Embedded Systems Conference*. San Jose, CA, USA, 1998.

[12] K. Wiegers, “Karl Wiegers Describes 10 Requirements Traps to Avoid,” *Softw. Test. Qual. Eng.*, vol. 2, no. 1, pp. 1–8, 2000.

[13] D. Dorner, “Chapter 1: Some Examples,” in *The Logic of Failure, Recognizing and Avoiding Error in Complex Situations*, New York, NY, USA: Metropolitan Books, Henry Holt and Company, 1996, pp. 28–33.

[14] K. R. Fowler and C. L. Silver, “Interview with Jack Ganssle, Aug 2011,” in *Developing and Managing Embedded Systems and Products*, Oxford, UK: Newnes, an imprint of Elsevier, 2015, pp. 17–18.

[15] A. Gerson and M. Barr, “2016 Embedded Systems Safety & Security Survey,” Barr Group, Germantown, MD, USA, 2016.

[16] A. Gerson and M. Barr, “Embedded Systems Safety & Security Survey,” Barr Group, Germantown, MD, USA, 2017.

[17] H. Petroski, *To Engineer is Human, The Role of Failure in Successful Design*. New York, NY, USA: First Vintage Books Edition, 1992.

[18] “Brainscope Annouces FDA Clearance of AHEAD® 100 Device for Adjunctive Assessment of Traumatic Brain Injury,” 2014. [Online]. Available: http://brainscope.com/media/2015/8/19/brainscope-announces-fda-clearance-of-ahead-100-device-for-adjunctive-assessment-of-traumatic-brain-injury. [Accessed: 01-Jan-2017].

[19] J. Chiles, *Inviting Disaster, Lessons from the Edge of Technology*. New York, NY, USA: Harper Business, HarperCollins Publishers, Inc., 2002.

[20] T. H. McKaig, *Building Failures: Case Studies in Construction and Design*. New York, NY, USA: McGraw-Hill Inc., 1962.

[21] BKCASE Editorial Board, *Guide to the Systems Engineering Body of Knowledge*.

BKCASE, 2017.

[22] P. Bourque and R. E. Fairley, *Guide to the Software Engineering - Body of Knowledge.* IEEE Computer Society, 2014.

[23] B. Haskins, B. Dick, J. Stecklein, R. Lovell, G. Moroney, and J. Dabney, "Error Cost Escalation Through the Project Life Cycle," in *INCOSE-Annual Conference Symposium Proceedings- 14th Annual International Symposium*, 2004, pp. 1–8.

[24] D. Kahneman, "Chapters 10, 11, 13, 15, 23, 26, 30," in *Thinking, Fast and Slow*, New York, NY, USA: Farrar, Straus and Giroux, 2011, pp. 116–117, 120–126, 144, 159, 250–251, 283–284, 324.

[25] "List of cognitive biases," *Wikipedia*. pp. 1–22, 2016.

[26] "List of cognitive biases," *RationalWiki*. pp. 1–8, 2016.

[27] D. Kahneman and A. Tversky, "Prospect Theory: An Analysis of Decision under Risk," *Econom. J. Econom. Soc.*, vol. 47, no. 2, pp. 263–291, 1979.

[28] A. Tversky and D. Kahneman, "Belief in the Law of Small Numbers," *Psychol. Bull.*, vol. 76, no. 2, pp. 105–110, 1971.

[29] T. Gilovich, R. Vallone, and A. Tversky, "The Hot Hand in Basketball: On the Misperception of Random Sequences," *Cogn. Psychol.*, vol. 17, pp. 295–314, 1985.

[30] D. Ariely, "Chapters 2, 4, 6, 7, 10," in *Predictably Irrational, The Hidden Forces That Shape Our Decisions*, New York, NY, USA: Harper Perennial, 2010, pp. 25–53, 75–100, 127–129, 139–146, 216.

[31] T. Mussweiler and F. Strack, "The use of category and exemplar knowledge in the solution of anchoring tasks.," *J. Pers. Soc. Psychol.*, vol. 78, no. 6, pp. 1038–1052, 2000.

[32] G. B. Northcraft and M. A. Neale, "Experts, amateurs, and real estate: An anchoring-and-adjustment perspective on property pricing decisions," *Organ. Behav. Hum. Decis. Process.*, vol. 39, no. 1, pp. 84–97, 1987.

[33] A. D. Galinsky and T. Mussweiler, “First offers as anchors: the role of perspective-taking and negotiator focus,” *J. Pers. Soc. Psychol.*, vol. 81, no. 4, pp. 657–669, 2001.

[34] K. R. Fowler and H. B. Land, “System design that minimizes both missed detections and false alarms: A case study in arc fault detection,” in *Conference Record - IEEE Instrumentation and Measurement Technology Conference*, 2004, vol. 3, no. May, pp. 2213–2216.

[35] N. Novemsky and D. Kahneman, “The Boundaries of Loss Aversion,” *J. Mark. Res.*, vol. 42, no. 2, pp. 119–128, 2005.

[36] L. Levine and W. Novak, “Brooks’ Law,” *Acquisition Archetypes*, 2009. [Online]. Available: http://www.sei.cmu.edu/acquisition/research/%0Aarchetypes.cfm.

[37] F. P. Brooks, *The Mythical Man-Month: Essays on Software Engineering*. Addison-Wesley, 1975.

[38] A. Tversky and D. Kahneman, “Advances in Prospect-Theory - Cumulative Representation of Uncertainty,” *J. Risk Uncertain.*, vol. 5, no. 4, pp. 297–323, 1992.

[39] D. Prelec, “The Probability Weighting Function,” *Econometrica*, vol. 66, no. 3, pp. 497–527, 2012.

[40] G. Loewenstein, “Out of control: Visceral influences on behavior,” *Organ. Behav. Hum. Decis. Process.*, vol. 65, no. 3, pp. 272–292, 1996.

[41] D. Ariely and K. Wertenbroch, “Procrastination, deadlines, and performance: self-control by precommitment.,” *Psychol. Sci.*, vol. 13, no. 3, pp. 219–224, 2002.

[42] J. Surowiecki, *The Wisdom of Crowds*. New York, NY, USA: Anchor Books, A Division of Random House, Inc., 2005.

[43] C. Edwards, “Agreeing To Fail,” *Eng. Technol.*, no. April-May, pp. 74–76, 2009.

[44] C. Wagner and A. Suh, “The wisdom of crowds: Impact of collective size and expertise transfer on collective performance,” in *Proceedings of the Annual Hawaii International*

Conference on System Sciences, 2014, pp. 594–603.

[45] S. P. J. Medeiros, D. Schneider, J. M. De Souza, and M. G. P. Esteves, “Strategic planning in semantic crowdware large groups decision,” in *Proceedings of the 2012 IEEE 16th International Conference on Computer Supported Cooperative Work in Design*, 2012, pp. 649–654.

[46] L. Rosenberg and D. Baltaxe, “Setting Group Priorities - Swarms vs Votes,” in *Swarm/Human Blended Intelligence Workshop (SHBI)*, 2016.

[47] M. Pankowska, “Autopoiesis in recommender systems,” in *Information Society (i-Society), 2013 International Conference on*, 2013, pp. 124–129.

[48] C. Wagner, S. Zhao, C. Schneider, and H. Chen, “The wisdom of reluctant crowds,” in *Proceedings of the Annual Hawaii International Conference on System Sciences*, 2010, pp. 1–10.

[49] M. G. Stochel, “Reliability and accuracy of the estimation process: Wideband Delphi vs. Wisdom of Crowds,” in *Proceedings - International Computer Software and Applications Conference*, 2011, pp. 350–359.

[50] L. Rosenberg, D. Baltaxe, and N. Pescetelli, “Crowds vs Swarms, a Comparison of Intelligence,” in *Swarm/Human Blended Intelligence Workshop (SHBI)*, 2016.

[51] C. Ingram and E. Vaast, “Crowdfunding and the ‘Wisdom of the Crowds,’” in *Proceedings of the Annual Hawaii International Conference on System Sciences*, 2016, p. 3504.

[52] V. Du Nguyen and N. T. Nguyen, “The impact of diversity on the quality of collective prediction,” in *2017 IEEE International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, 2017, pp. 1–6.

[53] L. Xia, C. C. Cao, L. Chen, and Z. Chen, “C-DMr : Crowd-powered Decision Maker for Real World Knapsack Problems,” in *ICDE’14 Proceedings of IEEE 30th International Conference on Data Engineering*, 2014, pp. 1174–1177.

[54] L. Laird and Y. Yang, “Engaging software estimation education using LEGOs,” in *Proceedings of the 38th International Conference on Software Engineering Companion*, 2016, pp. 511–517.

[55] L. Rosenberg, “Artificial Swarm Intelligence vs human experts,” in *Proceedings of the International Joint Conference on Neural Networks*, 2016, pp. 2547–2551.

[56] S. Cooper *et al.*, “Predicting Protein Structures with a Multiplayer Online Game,” *Nature*, vol. 466, no. 7307, pp. 756–760, 2010.

[57] M. Hosseini, A. Shahri, K. Phalp, and R. Ali, “Recommendations on adapting crowdsourcing to problem types,” in *Proceedings - International Conference on Research Challenges in Information Science*, 2015, vol. 2015–June, no. June, pp. 423–433.

[58] M. Hosseini, A. Shahri, K. Phalp, J. Taylor, and R. Ali, “Crowdsourcing: A taxonomy and systematic mapping study,” *Comput. Sci. Rev.*, vol. 17, pp. 43–69, 2015.

[59] T. D. Latoza and A. Van Der Hoek, “Crowdsourcing in Software Engineering: Models , Opportunities , and Challenges,” *IEEE Softw.*, pp. 74–80, 2016.

[60] K. J. Stol, T. D. Latoza, and C. Bird, “Crowdsourcing for Software Engineering,” *IEEE Softw.*, vol. 34, no. 2, pp. 30–36, 2017.

[61] N. Hasteer, N. Nazir, A. Bansal, and B. K. Murthy, “Crowdsourcing Software Development: Many Benefits Many Concerns,” *Procedia Comput. Sci.*, vol. 78, pp. 48–54, 2016.

[62] G. J. Skulmoski, F. T. Hartman, and J. Krahn, “The Delphi Method for Graduate Research,” *J. Inf. Technol. Educ.*, vol. 6, no. 1, pp. 1–21, 2007.

[63] “Delphi method,” *Wikipedia*. [Online]. Available: https://en.wikipedia.org/wiki/Delphi_method.

[64] “Wideband Delphi,” *Wikipedia*. [Online]. Available: https://en.wikipedia.org/wiki/Wideband_delphi.

[65] L.-C. Huang, P.-T. Chang, and H.-J. Lin, “A study of fuzzy assessment model for managerial talent,” in *1996 IEEE International Conference on Systems, Man and Cybernetics. Information Intelligence and Systems*, 1996, vol. 1, pp. 400–405.

[66] M.-J. Wu, C.-Y. Huang, and Y.-H. Hung, “Capabilities-driven Curriculum Design for Hydrogen and Fuel Cell Technologies,” in *2011 IEEE Green Technologies Conference*, 2011, pp. 4–9.

[67] K. K. Lilja, K. Laakso, and J. Palomäki, “Using the Delphi Method,” in *2011 Proceedings of PICMET ’11: Technology Management in the Energy Smart World*, 2011, pp. 1–10.

[68] N. C. Dalkey and O. Helmer, “An Experimental Application of the Delphi Method to the Use of Experts,” *Manage. Sci.*, vol. 9, no. 3, pp. 458–468, 1963.

[69] G. Rowe and G. Wright, “The Delphi Technique as a Forecasting Tool: Issues and Analysis,” *Int. J. Forecast.*, vol. 15, no. 4, pp. 353–375, 1999.

[70] A. M. J. Skulimowski, “Expert Delphi Survey as a Cloud-Based Decision Support Service,” in *2017 IEEE 10th Conference on Service-Oriented Computing and Applications (SOCA)*, 2017, pp. 190–197.

[71] T.-S. Lan, P.-C. Chen, C.-W. Chiu, and Y.-M. Lai, “Developing a Maintenance Strategy of the Critical Components for the Environmental Control System of Aircraft,” in *2016 International Conference on Applied System Innovation (ICASI)*, 2016, pp. 1–4.

[72] Y. Qi, Azguli, Y. Xie, Q. Jia, B. Huang, and W. Fu, “The Research and Implementation of Electricity Bill Retrieving Risk Evaluation Model for Low- voltage Users,” in *2017 IEEE Conference on Energy Internet and Energy System Integration (EI2)*, 2017, pp. 1–4.

[73] X. Lu and H. Fan, “Study on the risk assessment approach of port facility security based on a comprehensive model of Delphi method, analytic hierarchy process, grey theory and fuzzy evaluation method,” in *2015 International Conference on Transportation Information and Safety (ICTIS)*, 2015, pp. 628–632.

[74] T. . Wang and P. . Yuan, “Technological economic study for ocean energy development in

China," in *IEEE International Conference on Industrial Engineering and Engineering Management*, 2011, pp. 610–614.

[75] S. L. Zhang, N. Zhou, and J. X. Wu, "The fuzzy integrated evaluation of embedded system security," in *Proceedings of The International Conference on Embedded Software and Systems, ICESS 2008*, 2008, pp. 157–162.

[76] B. Wu, X. Zhou, Q. Jin, F. Lin, and H. Leung, "Analyzing Social Roles Based on a Hierarchical Model and Data Mining for Collective Decision-Making Support," *IEEE Syst. J.*, vol. 11, no. 1, pp. 356–365, 2017.

[77] M. Adnan and M. Afzal, "Ontology based Multiagent Effort Estimation System for Scrum Agile Method," *IEEE Access*, pp. 25993–26005, 2017.

[78] C. Okoli and S. D. Pawlowski, "The Delphi Method as a Research Tool: An Example, Design Considerations and Applications," *Inf. Manag.*, vol. 42, no. 1, pp. 15–29, 2004.

[79] "Behavioral Game Theory," *Wikipedia*. 2017.

[80] K. Jhala, B. Natarajan, and A. Pahwa, "Prospect Theory based Active Consumer Behavior Under Variable Electricity Pricing," *IEEE Trans. Smart Grid*, pp. 1–8, 2018.

[81] G. V. Bhatia, "A game theory approach to negotiations in defense acquisitions in the context of value-driven design : an aircraft system case study," M.S. thesis, Aero. Eng., Iowa State University, Ames, IA, USA, 2016.

[82] D. Gale and L. S. Shapley, "College Admissions and the Stability of Marriage," *Am. Math. Mon.*, vol. 69, no. 1, pp. 9–15, 1962.

[83] "Stable Marriage Problem," *Wikipedia*. 2017.

[84] S. Gupta, K. Iwama, and S. Miyazaki, "Stable Nash Equilibria in the Gale-Shapley Matching Game," *arxiv.org*, pp. 1–11, 2015.

[85] V. Chankong and Y. Y. Haimes, *Multiobjective Decision Making, Theory and Methodology*. New York, NY, USA: North-Holland/Elsevier Science Publishing

Company, 1983.

[86] R. T. Clemen, T. Reilly, S. E. Bodily, and J. Guyse, *Making Hard Decisions with Decision Tools Suite*, 3rd ed. Mason, OH, USA: South-Western, 2013.

[87] G. Parmigiani and L. Inoue, *Decision Theory, Principles and Approaches*. Chichester, West Sussex, UK: John Wiley & Sons, Ltd., 2009.

[88] P. Driscoll and P. Kucik, “System Life Cycle,” in *Decision Making in Systems Engineering and Management*, 2nd ed., G. S. Parnell, P. J. Driscoll, and D. L. Henderson, Eds. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2011, pp. 65–93.

[89] T. Middelkoop, D. L. Pepyne, and A. Deshmukh, “Analysis of Negotiation Protocols for Distributed Design,” in *Decision Making in Engineering Design*, K. E. Lewis, W. Chen, and L. C. Schmidt, Eds. New York, NY, USA: ASME, 2006, pp. 265–279.

[90] The Committee for the Prize in Economic Sciences in Memory of Alfred Nobel, “OLIVER HART AND BENGT HOLMSTRÖM: CONTRACT THEORY, Scientific Background on the Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel 2016,” Stockholm, Sweden, 2016.

[91] H. Li and A. Kolotilin, “Explainer, contract theory, very brief explanation, 22 Oct 2017,” *The Conversation*, 2016. [Online]. Available: https://theconversation.com/explainer-what-is-contract-theory-and-why-it-deserved-a-nobel-prize-66826. [Accessed: 22-Oct-2017].

[92] A. H. Maslow, *Motivation and Personality*. New York, NY, USA: Harper & Brothers, 1954.

[93] F. Herzberg, B. Mausner, and B. B. Snyderman, *The Motivation to Work*. New York, NY, USA: John Wiley & Sons, Inc., 1959.

[94] G. L. Mangum, *Wage Incentive Systems*. Berkeley, CA, USA: University of California, Berkeley, 1964.

[95] G. F. Farris, “Rewards and retention of technical staff,” in *Proceedings of the 2000 IEEE*

Engineering Management Society, 2000, pp. 617–619.

[96] F. S. F. Soares, G. S. D. A. Júnior, and S. R. D. L. Meira, “Incentive systems in software organizations,” in *4th International Conference on Software Engineering Advances, ICSEA 2009*, 2009, pp. 93–99.

[97] V. Casey and I. Richardson, “The impact of fear on the operation of virtual teams,” in *Proceedings - 2008 3rd IEEE International Conference Global Software Engineering, ICGSE 2008*, 2008, pp. 163–172.

[98] L. Šteinberga and D. Šmite, “Towards understanding of software engineer motivation in globally distributed projects,” in *Proceedings - 2011 6th IEEE International Conference on Global Software Engineering Workshops, ICGSE Workshops 2011*, 2011, pp. 117–119.

[99] I. H. de F. Junior, L. Duarte, J. P. N. de Oliveira, A. R. N. Dantas, J. F. Barbosa, and H. P. de Moura, “Motivational Factors for Distributed Software Development Teams,” in *2012 IEEE Seventh International Conference on Global Software Engineering Workshops*, 2012, pp. 49–54.

[100] A. C. C. França, A. C. M. L. De Araújo, and F. Q. B. da Silva, “Motivation of Software Engineers: A Qualitative Case Study of a Research and Development Organisation,” in *6th International Workshop on Cooperative and Human Aspects of Software Engineering, CHASE 2013 - Proceedings*, 2013, pp. 9–16.

[101] S. Beecham, N. Baddoo, T. Hall, H. Robinson, and H. Sharp, “Motivation in Software Engineering: A Systematic Literature Review,” University of Hertfordshire, Hatfield, England, ACM Rep. 464, 2007.

[102] T. Hall, H. Sharp, S. Beecham, N. Baddoo, and H. Robinson, “What Do We Know about Developer Motivation,” *IEEE Software*, pp. 92–94, 2008.

[103] T. Hall, N. Baddoo, S. Beecham, H. Robinson, and H. Sharp, “A systematic review of theory use in studies investigating the motivations of software engineers,” *ACM Trans. Softw. Eng. Methodol.*, vol. 18, no. 3, pp. 1–29, 2009.

[104] H. Sharp and T. Hall, “An initial investigation of software practitioners’ motivation,” in *Proceedings of the 2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering, CHASE 2009*, 2009, pp. 84–91.

[105] M. Holle, L. Elsesser, M. Schuhmacher, and U. Lindemann, “How to Motivate External Open Innovation-Partners: Identifying Suitable Measures,” in *Proceedings of 2016 Portland International Conference on Management of Engineering and Technology (PICMET)*, 2016, pp. 902–913.

[106] K. R. Browne, “Unique Factors for Motivating Engineers : A Motivation Meta Theory Approach,” Doctor of Management (DMgt) dissertation, College of Business, George Fox University, Newberg, OR, USA, 2013.

[107] R. Mironov, “Motivating Development Teams 18,” 2014. [Online]. Available: http://www.mironov.com/motivating/.

[108] Y. Francino, “13 ways to motivate software engineers,” 2017. [Online]. Available: https://techbeacon.com/13-ways-motivate-software-engineers.

[109] K. Brown, “The Product Manager vs . The Engineering Manager,” 2014. [Online]. Available: http://blog.aha.io/the-product-manager-vs-the-engineering-manager/.

[110] S. Rekhi, “The Most Underrated Product Management Skill: Influence without Authority,” 2013. [Online]. Available: http://www.sachinrekhi.com/the-most-underrated-product-management-skill-influence-without-authority.

[111] S. Aboulafia, “11 Essential Non-Technical Product Management Skills,” 2016. [Online]. Available: http://community.uservoice.com/blog/non-technical-product-management-skills/.

[112] W. A. Hahn, “Applied Business Research,” *IRE Trans. Eng. Manag.*, vol. EM-9, no. 1, pp. 3–10, 1962.

[113] E. Wagner and S. A. Webber, “Creating a quality assurance function and a quality culture in a research and development organization,” in *IEEE International Conference on*

Communications, - Spanning the Universe, 1988, pp. 260–266.

[114] S. L. Brown and K. M. Eisenhardt, "Product Development : Past Research , Present Findings , and Future Directions," *Acad. Manag. Rev.*, vol. 20, no. 2, pp. 343–378, 1995.

[115] Q. Zhang and W. J. Doll, "The fuzzy front end and success of new product development: a causal model," *Eur. J. Innov. Manag.*, vol. 4, no. 2, pp. 95–112, 2001.

[116] C. Schwarz and C. Pothier, "Commercial ATS Program Management," *IEEE Aerosp. Electron. Syst. Mag.*, pp. 3–6, 2002.

[117] R. C. Moore, "Successful Space System Engineer," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 21, pp. 21–27, Mar-2000.

[118] L. Peters, "Managing Software Professionals," in *Engineering Management Conference, 2003. IEMC '03. Managing Technologically Driven Organizations: The Human Side of Innovation and Change*, 2003, pp. 61–66.

[119] R. P. Alexander and E. A. Mortlock, "Controlling and executing communications based train control (CBTC) installation & testing with a CBTC-ready vehicle," *Proc. 2005 ASME/IEEE Jt. Rail Conf. 2005.*, pp. 193–198, 2005.

[120] H. Peng and J. Xin, "Research on the synergistic work mechanism of R and D teams," in *Proceedings - 2008 International Seminar on Future Information Technology and Management Engineering, FITME 2008*, 2008, pp. 543–546.

[121] R. Sperry and A. Jetter, "Theoretical framework for managing the front end of innovation under uncertainty," in *PICMET: Portland International Center for Management of Engineering and Technology, Proceedings*, 2009, pp. 2021–2028.

[122] J. Hutchinson, M. Rouncefield, and J. Whittle, "Model-Driven Engineering Practices in Industry," in *33rd International Conference on Software Engineering (ICSE 2011)*, 2011, pp. 633–642.

[123] R. Pang, V. Kennedy, B. Armand, L. Mauch, and J. D. Fleming, "CHIRP program lessons

learned from the contractor program management team perspective," in *IEEE Aerospace Conference Proceedings*, 2012, pp. 1–7.

[124] I. R. Chiang and S. J. Wu, "Supplier Involvement and Contract Design During New Product Development," *IEEE Trans. Eng.*, vol. 63, no. 2, pp. 248–258, 2016.

[125] E. J. Young, "The Social and Cultural Dimensions in Engineering Management and Organization of Global Operations," in *Proceedings of the 1992 International Engineering Management Conference*, 1992, pp. 14–19.

[126] L. Jizhen and P. Xin, "IEEE, R&D Project Management Process, Cultural Adaptation of Lucent China, 2006.pdf," in *PICMET 2006 Proceedings*, 2006, pp. 744–747.

[127] C. A. Dorantes, C. Hallam, and Z. Gianluca, "Culture and Social Capital: effect on Growth of Small Medium Enterprise," in *2017 Proceedings of PICMET '17: Technology Management for Interconnected World Culture*, 2017.

[128] C. M. Chang, "Engineering Management in Developing Economies : The EMIDE Strategies to Meet the N ew Challenges," in *PICMET 2008 Proceedings*, 2008, pp. 1817–1826.

[129] L. Ji-zhen and L. Xing, "Cultural differences and process adaptation in international R&D project management," in *2009 International Conference on Management Science and Engineering - 16th Annual Conference Proceedings, ICMSE 2009*, 2009, pp. 1551–1558.

[130] X. Wen and C. Huang, "The Impact of Uncertainty Avoidance and Organizational Culture on Management Innovation," in *2012 International Conference on Information Management, Innovation Management and Industrial Engineering*, 2012, pp. 331–334.

[131] I. P. L. Png, B. C. Y. Tan, and K.-L. Wee, "Dimensions of national culture and corporate adoption of IT infrastructure," *IEEE Trans. Eng. Manag.*, vol. 48, no. 1, pp. 36–45, 2001.

[132] A. Onumo, A. Cullen, and I. Ullah-Awan, "An Empirical Study of Cultural Dimensions and Cybersecurity Development," in *2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud)*, 2017, pp. 70–76.

[133] G. Hofstede, “Dimensionalizing Cultures: The Hofstede Model in Context,” *Online Readings Psychol. Cult.*, vol. 2, no. 1, pp. 1–26, 2011.

[134] M. C. Aaron, *Negotiating Outcomes: Expert Solutions to Everyday Challenges (Pocket Mentor)*. Harvard Business Press, 2007.

[135] J. Nadler and S. Hackley, “Dispute resolution,” 2012.

[136] B. W. Tuckman and M. A. C. Jensen, “Stages of Small-Group Development Revisited,” *Gr. Organ. Manag.*, vol. 10, pp. 43–48, 419–427, 1977.

[137] B. Goldense, “Goldense lecture, 1996, Metrics for Measuring Product Development.” Goldense Group, Inc., Cambridge, MA, USA, 1996.

[138] K. L. Tyran, “MGMT 313, developing a team contract,” *Western Washington University webpage*. [Online]. Available: http://faculty.wwu.edu/tyrank/MGMT313/TeamContract313.htm. [Accessed: 15-Jan-2018].

[139] J. Ganssle, “private communications on team contracts.” Feb. 2018.

[140] M. Barr, “Barr RE team contracts in product development, personal communications.” 2018.

[141] R. C. Rassa, “Rassa RE team contracts in product development, personal communications.” 2018.

[142] M. Gard, “private communications on team contracts.” Feb. 2018.

[143] B. Boehm, P. Grünbacher, and R. O. Briggs, “Developing Groupware for Requirements Negotiation: Lessons Learned,” *IEEE Softw.*, no. May/June, pp. 46–55, 2001.

[144] A. Naumchev and B. Meyer, “Complete Contracts through Specification Drivers,” in *Proceedings - 10th International Symposium on Theoretical Aspects of Software Engineering, TASE 2016*, 2016, pp. 160–167.

[145] R. W. Floyd, “Assigning Meaning to Programs,” in *Proceedings of Symposium on Applied Mathematics*, 1967, pp. 19–32.

[146] C. A. R. Hoare, “Hoare Axiomatic Basis for Comp Prog.Pdf,” *Commun. ACM*, vol. 12, no. 10, pp. 576–580, 583, 1969.

[147] P. Nuzzo *et al.*, “A contract-based methodology for aircraft electric power system design,” *IEEE Access*, vol. 2, pp. 1–25, 2014.

[148] A. Benveniste, W. Damm, A. Sangiovanni-Vincentelli, D. Nickovic, R. Passerone, and P. Reinkemeier, “Contracts for the design of embedded systems, Part I: Methodology and use cases,” *Proc. IEEE, 2012*, vol. 2, no. 215543, pp. 1–30, 2012.

[149] A. Benveniste *et al.*, “Contracts for the Design of Embedded Systems - Part II,” *Proc. IEEE, 2012*, no. 215543, pp. 1–32, 2012.

[150] A. Benveniste *et al.*, “Contracts for Systems Design: Theory,” INRIA, Rennes, France, Rep. hal-01178467, 2015.

[151] A. Benveniste *et al.*, “Contracts for Systems Design: Methodology and Application Cases,” INRIA, Rennes, France, Rep. hal-01178469, 2015.

[152] A. Sutcliffe, “Analysing social computing requirements with small group theory,” in *2011 1st International Workshop on Requirements Engineering for Social Computing, RESC’11*, 2011, pp. 18–21.

[153] A. Sutcliffe, “Bridging users’ values and requirements to architecture,” in *2013 3rd International Workshop on the Twin Peaks of Requirements and Architecture, TwinPeaks 2013 - Proceedings*, 2013, pp. 15–21.

[154] A. Sutcliffe and P. Sawyer, “Requirements Elicitation:Towards the unknown unknowns,” in *21st IEEE International Requirements Engineering Conference (RE)*, 2013, pp. 92–104.

[155] J. Alstad, “COSATMO: Developing The Harmonized COSYSMO 3.0 Model Including the Coalesced Effort Multiplier Submodel,” University of Southern California, USA,

published by The Aerospace Corporation, 2015.

[156] "COCOMO," *Wikipedia*. [Online]. Available: https://en.wikipedia.org/wiki/COCOMO.

[157] I. Sommerville, "Software Cost Estimation," in *Software Engineering*, 8th ed., Boston, MA, USA: Addison-Wesley Longman, Inc., 2006, pp. 612–640.

[158] B. W. Boehm, D. J. Reifer, and R. Valerdi, "COSYSMO: A Systems Engineering Cost Model," in *1st Annual Conference on Systems Integration, 12-14 March*, 2003.

[159] P. N. Jeziorek, "Cost Estimation of Functional and Physical Changes Made to Complex Systems," M.S. thesis, Mech. Eng., MIT, Cambridge, MA, USA, 2005.

[160] R. Valerdi, G. J. Roedler, L. Martin, J. E. Rieff, and M. J. Wheaton, "Lessons Learned From Industrial Validation of COSYSMO," *Int. Counc. Syst. Eng.*, pp. 1–12, 2007.

[161] J. A. Lane, "Impacts of System of System Management Strategies on System of System Capability Engineering Effort," Ph.D. dissertation, Ind. and Sys. Eng., University of Southern California, Los Angeles, CA, 2009.

[162] M. Dabkowski, R. Valerdi, and J. Farr, "Exploiting Architectural Communities in Early Life Cycle Cost Estimation," in *Procedia Computer Science*, 2014, vol. 28, no. Cser, pp. 95–102.

[163] R. Valerdi, "The Costructive Systems Engineering Cost Model (COSYSMO)," Ph.D. dissertation, Ind. and Sys. Eng., University of Southern California, Los Angeles, CA, 2005.

[164] Jon K. Ilseng, "Systems Engineering Cost Estimation: System-of-Systems," M.S. thesis, Col. of Eng., Texas Tech University, Lubbock, TX, 2006.

[165] J. F. Holmes, "Designing a Cost Estimation Method for the Design of Prototype Systems," M.S. thesis, Mech. Eng., Georgia Institute of Technology, Atlanta, GA, 2012.

[166] Z. Zhai, P. Sempolinski, D. Thain, G. Madey, D. Wei, and A. Kareem, "Expert-citizen engineering: 'Crowdsourcing' skilled citizens," in *Proceedings - IEEE 9th International*

Conference on Dependable, Autonomic and Secure Computing, DASC 2011, 2011, pp. 879–886.

[167] Z. Zhai, D. Hachen, T. Kijewski-Correa, F. Shen, and G. Madey, “Citizen engineering: Methods for ‘crowdsourcing’ highly trustworthy results,” in *Proceedings of the Annual Hawaii International Conference on System Sciences*, 2012, pp. 3406–3415.

[168] P. A. Laplante, *Requirements Engineering for Software and Systems*, 2nd ed. Boca Raton, FL, USA: CRC Press, 2014.

[169] P. Ranjan and A. K. Misra, “An enhanced model for agent based requirement gathering and pre-system analysis,” in *Proceedings of the 13th Annual International Symposium and Workshop on Engineering of Computer Based Systems*, 2006, pp. 187–195.

[170] J. Savolainen, D. Hauksdóttir, and M. Mannion, “Challenges in balancing the amount of solution information in requirement specifications for embedded products,” in *2013 21st IEEE International Requirements Engineering Conference, RE 2013 - Proceedings*, 2013, pp. 256–260.

[171] J. Lockhart, C. Purdy, and P. Wilsey, “Formal methods for safety critical system specification,” in *Midwest Symposium on Circuits and Systems*, 2014, pp. 201–204.

[172] I. Hadar and A. Zamansky, “Cognitive factors in inconsistency management,” in *2015 IEEE 23rd International Requirements Engineering Conference, RE 2015 - Proceedings*, 2015, pp. 226–229.

[173] C. Larman and V. R. Basili, “Iterative and Incremental Development: A Brief History,” *Computer*, pp. 47–56, 2003.

[174] S. Shlaer and S. J. Mellor, “Recursive Design of an Application- Independent Architecture,” *IEEE Softw.*, no. 1, pp. 61–72, 1997.

[175] J. P. Bowen and M. G. Hinchey, “Seven More Myths of Formal Methods,” *IEEE Softw.*, vol. July, pp. 34–41, 1995.

[176] B. Larson, J. Hatcliff, and K. Fowler, “Illustrating the AADL Error Modeling Annex (v.2) Using a Simple Safety-Critical Medical Device,” in *HILT ’13*, 2013, pp. 65–82.

[177] J. M. Wing, “A specifier’s introduction to formal methods,” *Computer (Long. Beach. Calif).*, pp. 8–24, 1990.

[178] M. D’Inverno, G. R. R. Justo, and P. Howells, “A formal framework for specifying design methods,” in *System Sciences, 1996., Proceedings of the Twenty-Ninth Hawaii International Conference on,* 1996, vol. 1, pp. 741–750.

[179] C. Seidner and O. H. Roux, “Formal Methods for Systems Engineering Behavior Models,” *IEEE Trans. Ind. Informatics*, vol. 4, no. 4, pp. 280–291, 2008.

[180] M. Bordin, J. Guitton, J. Hugues, and L. Pautet, “Couverture: an Innovative Open Framework for Coverage Analysis of Safety Critical Applications,” *Ada User J.*, no. October 2016, pp. 1–14, 2009.

[181] J. Sifakis, “Methods and tools for component-based system design,” in *EDAA*, 2011, pp. 1–4.

[182] L. Marcil and M. Hawthornthwaite, “Realizing DO-178C’s value by using new technology: OOT, MBDV, TQC & FM,” in *IEEE Proceedings of the 31st Digital Avionics Systems Conference*, 2012, pp. 1–10.

[183] A. Gawande, *The Checklist Manifesto: How to Get Things Right*. New York, NY, USA: Picador, 2011.

[184] “Identifying and evaluating hazards in research laboratories, Guidelines Developed by the Hazard Identification and Evaluation Task Force of the American Chemical Society’s Committee on Chemical Safety,” American Chemical Society, Washington, DC, 2015.

[185] B. J. B. Wood, “Introduction,” in *Building Maintenance,* Wiley-Blackwell, 2009, p. 3.

[186] M. Scriven, “The logic and methodology of checklists,” The Evaluation Center, Western Michigan University, Kalamazoo, MI, USA, 2005.

[187] L. A. Wingate, “The Evaluation Checklist Project: The Inside Scoop on Content, Process, Policies, Impact, and Challenges,” in *American Evaluation Association Conference*, 2002, pp. 1–12.

[188] D. L. Stufflebeam, “Cipp evaluation model checklist,” The Evaluation Center, Western Michigan University, Kalamazoo, MI, USA, 2007.

[189] C. B. Hersman and K. R. Fowler, “Chapter 5: Best Practices in Spacecraft Development,” in *Mission-Critical and Safety-Critical Systems Handbook*, K. R. Fowler, Ed. Newnes, an imprint of Elsevier, 2010, pp. 269–460.

[190] K. Henriksen and J. Brady, “The pursuit of better diagnostic performance: A human factors perspective,” *BMJ Qual. Saf.*, vol. 22, no. SUPPL.2, pp. 1–5, 2013.

[191] J. Raman *et al.*, “When a checklist is not enough: How to improve them and what else is needed,” *J. Thorac. Cardiovasc. Surg.*, vol. 152, no. 2, pp. 585–592, 2016.

[192] M. A. Regan and J. H. Richardson, “Planning and implementing field operational tests of intelligent transport systems: a checklist derived from the EC FESTA project,” *IET Intell. Transp. Syst.*, vol. 3, no. 2, pp. 168–184, 2009.

[193] S. Nan *et al.*, “DCCSS : A Meta-model for Dynamic Clinical Checklist Support Systems,” in *3rd International Conference on Model-Driven Engineering and Software Development*, 2015, pp. 272–279.

[194] R. C. Hopkins, “A Systematic Procedure for System Development,” *IRE Trans. Eng. Manag.*, pp. 77–86, 1961.

[195] G. Belev, “Guidelines for Specification Development,” in *1989 Proceedings Annual RELIABILITY AND MAINTAINABILITY Symposium*, 1989, pp. 15–21.

[196] “Requirements Tools,” *Volere Requirements Resources*. [Online]. Available: http://www.volere.co.uk/tools.htm. [Accessed: 12-Feb-2018].

[197] E. E. Wierba, T. A. Finholt, and M. P. Steves, “Challenges to collaborative tool adoption

in a manufacturing engineering setting: A case study," in *Proceedings of the 35th Hawaii International Conference on System Sciences*, 2002, vol. 2002–Janua, no. c, pp. 3594–3603.

[198] "Multi-Board Systems Design." [Online]. Available: https://www.mentor.com/pcb/xpedition/systems-design/. [Accessed: 12-Feb-2018].

[199] R. H. Thaler, "Going Public," in *Misbehaving, The Making of Behavioral Economics*, New York, NY, USA: W. W. Norton & Company, Inc., 2015, pp. 326–329.

[200] K. R. Fowler, "Chapter 15, Summary Comparisons Across the 11 Case Studies," in *What Every Engineer Should Know About Developing Real-Time Embedded Products*, Boca Raton, FL, USA: CRC Press, 2008, pp. 377–392.

[201] C. Wohlin, P. Runeson, M. Host, M. C. Ohlsson, B. Regnell, and A. Wesslen, "8.7 Validity Evaluation," in *Experimentation in Software Engineering*, Second., New York, NY, USA: Springer, 2012, pp. 102–110.

[202] V. Kanabar and R. D. H. Warburton, "Leveraging the new practice standard for project estimating," in *2011 PMI Global Congress Proceedings – Dallas, TX*, 2011, no. October, pp. 1–21.

[203] *A guide to the project management body of knowledge (PMBOK guide)*, Sixth. Newtown Square, PA, USA: Project Management Institute, 2017.

[204] "Center for Systems and Software Engineering." [Online]. Available: http://csse.usc.edu/csse/research/COCOMO_suite/index.html. [Accessed: 05-Oct-2017].

[205] R. Valerdi and B. W. Boehm, "COSYSMO: A Systems Engineering Cost Model," *Genie Logiciel*, pp. 1–9, 2010.

[206] J. S. Busby, "The Problem with Design Reuse: An Investigation into Outcomes and Antecedents," *J. Eng. Des.*, vol. 10, no. 3, pp. 277–296, 1999.

[207] K. R. Fowler, "Chapter 15: Manufacturing," in *Developing and Managing Embedded*

Systems and Products, Newnes, an imprint of Elsevier, 2015, pp. 595–647.

[208] “Plastic Injection Molding FAQ,” *The Rodon Group*, 2018. [Online]. Available: https://www.rodongroup.com/blog/bid/64646/plastic-injection-molding-faq.

[209] “Design Guidelines: Plastic Injection Molding,” *Protolabs*, 2018. [Online]. Available: https://www.protolabs.com/services/injection-molding/plastic-injection-molding/design-guidelines/.

[210] C. Jones, “Software Quality in 2012: a Survey of the State of the Art,” Namcook Analytics LLC, RI, USA, (no report number), 2012.

[211] C. Jones, “Function points as a universal software metric,” Namcook Analytics LLC, Namcook Analytics LLC, RI, USA, (no report number), 2013.

[212] J. Voas, “Can Generic Software Be Assured?,” in *Proceedings. Twenty-Third Annual International Computer Software and Applications Conference*, 1999, pp. 94–95.

[213] “Points when making the build vs buy decision,” *TechRepublic*, 2002. [Online]. Available: https://www.techrepublic.com/article/buy-vs-build-six-steps-to-making-the-right-decision/. [Accessed: 22-Jul-2018].

[214] “Build Versus Buy: Which Is the Best Choice for Your Embedded Design?” National Instruments, 2012.

[215] National Instruments, “Understanding the Total Cost of Embedded Design,” 2012.

[216] “Build vs Buy Decision Matrix,” *Demand Metric*. [Online]. Available: https://www.demandmetric.com/content/build-vs-buy-decision-matrix. [Accessed: 22-Jul-2018].

[217] “Buy vs build, Six steps to making the right decision,” *TechRepublic*, 2002. [Online]. Available: https://www.techrepublic.com/article/buy-vs-build-six-steps-to-making-the-right-decision/. [Accessed: 22-Jul-2018].

[218] “The Build vs Buy Decision Tree,” *Scalyr blog*, 2018. [Online]. Available:

https://blog.scalyr.com/2018/05/build-vs-buy/. [Accessed: 22-Jul-2018].

[219] "The Embedded Software Beat," *VDC Research blog*, 2012. [Online]. Available: https://www.vdcresearch.com/News-events/iot-blog/2012/the-embedded-software-beat2.html. [Accessed: 22-Jul-2018].

[220] D. Strassberg, "Distributed Power: Taming the Dragons," *EDN*, pp. 40–41, 2003.

[221] I. Gorton and A. Liu, "Architectures and technologies for enterprise application integration," in *Proceedings. 26th International Conference on Software Engineering*, 2004, pp. 726–727.

[222] K. Fowler, "Build versus buy," *IEEE Instrumentation and Measurement Magazine*, vol. 7, no. 3, pp. 67–73, 2004.

[223] "Build vs buy in an SoC world," *Tech Design Forum*, 2007. [Online]. Available: http://www.techdesignforums.com/practice/technique/build-vs-buy-in-an-soc-world/. [Accessed: 22-Jul-2018].

[224] "Commercial Linux saves time and money," *Military Embedded Systems, Avionics Design Newsletter,* 2008. [Online]. Available: http://mil-embedded.com/article-id/?2873=. [Accessed: 22-Jul-2018].

[225] "PMP - Build Or Buy Decisions," *uCertify on YouTube*, 2012. [Online]. Available: https://www.youtube.com/watch?v=PvpaUwHynt8. [Accessed: 04-Apr-2018].

[226] T. Bell, "Part 4 - Relevant Costs for Decision Making - Make or Buy," *Tony Bell on YouTube*, 2013. [Online]. Available: https://www.youtube.com/watch?v=9ILpNPml17s. [Accessed: 04-Apr-2018].

[227] Y. Gan *et al.*, "A comprehensive evaluation of various sensitivity analysis methods: A case study with a hydrological model," *Environ. Model. Softw.*, vol. 51, pp. 269–285, 2014.

[228] B. Iooss and P. Lemaître, "A review on global sensitivity analysis methods," in

Uncertainty management in Simulation-Optimization of Complex Systems: Algorithms and Applications, C. Meloni and G. Dellino, Eds. Springer, 2015.

[229] A. Saltelli, S. Tarantola, and K. P.-S. Chan, "A Quantitative Model-Independent Method for Global Sensitivity Analysis of Model Output," *Technometrics*, vol. 41, no. 1, pp. 39–56, 1999.

[230] A. Saltelli, S. Tarantola, and F. Campolongo, "Sensitivity Analysis as an Ingredient of Modeling," *Stat. Sci.*, vol. 15, no. 4, pp. 377–395, 2000.

[231] C. Ma, X. Li, and S. Wang, "A Global Sensitivity Analysis of Soil Parameters Associated With Backscattering Using the Advanced Integral Equation Model," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 10, pp. 5613–5623, 2015.

[232] D. Li, R. Jin, J. Zhou, and J. Kang, "Analysis and Reduction of the Uncertainties in Soil Moisture Estimation With the L-MEB Model Using EFAST and Ensemble Retrieval," *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 6, pp. 1337–1341, 2015.

[233] M. Majidi and M. Etezadi-Amoli, "A New Fault Location Technique in Smart Distribution Networks Using Synchronized / Nonsynchronized Measurements," *IEEE Trans. Power Deliv.*, vol. 33, no. 3, pp. 1358–1368, 2018.

[234] T. Bdour and A. Reineix, "Global Sensitivity Analysis and Uncertainty Quantification of Radiated Susceptibility in PCB Using Nonintrusive Polynomial Chaos Expansions," *IEEE Trans. Electromagn. Compat.*, vol. 58, no. 3, pp. 939–942, 2016.

[235] T. T. Nguyen and S. Clenet, "Influence of Material and Geometric Parameters on the Sensor Based on Active Materials," *IEEE Trans. Magn.*, vol. 54, no. 3, 2018.

[236] A. J. Manders, D. M. Simpson, and S. L. Bell, "Objective Prediction of the Sound Quality of Music Processed by an Adaptive Feedback Canceller," *IEEE Trans. Audio. Speech. Lang. Processing*, vol. 20, no. 6, pp. 1734–1745, 2012.

[237] Y. Xiao, W. Zhao, D. Zhou, and H. Gong, "Sensitivity Analysis of Vegetation Reflectance to Biochemical and Biophysical Variables at Leaf, Canopy, and Regional Scales," *IEEE*

Trans. Geosci. Remote Sens., vol. 52, no. 7, pp. 4014–4024, 2014.

[238] X. Cheng and V. Monebhurrun, “Application of Different Methods to Quantify Uncertainty in Specific Absorption Rate Calculation Using a CAD-Based Mobile Phone Model,” *IEEE Trans. Electromagn. Compat.*, vol. 59, no. 1, pp. 14–23, 2017.

[239] A. Bartel, H. De Gersem, T. Hülsmann, U. Römer, S. Schöps, and T. Weiland, “Quantification of Uncertainty in the Field Quality ofMagnets Originating from Material Measurements,” *IEEE Trans. Instrum. Meas.*, vol. 49, no. 5, pp. 2367–2370, 2013.

[240] C. Schretter, I. Loris, A. Dooms, and P. Schelkens, “Total Variation Reconstruction From Quasi-Random Samples,” in *2nd international - Traveling Workshop on Interactions between Sparse models and Technology · iTWIST 2014*, 2014, no. March, pp. 2–4.

[241] K. R. Fowler, S. Das, and S. A. Dyer, “A Study of the Build-Versus-Buy Decision in Developing Embedded Systems,” *Submitt. Publ.*, 2018.

[242] D. C. Schmidt, “Why Software Reuse has Failed and How to Make It Work for You,” 2014. [Online]. Available: https://www.dre.vanderbilt.edu/~schmidt/reuse-lessons.html. [Accessed: 27-Aug-2018].

[243] K. R. Fowler and R. B. North, “Computer-Optimized Neurostimulation,” *APL Tech. Dig.*, vol. 12, no. 2, pp. 192–197, 1991.

[244] K. R. Fowler and R. B. North, “Computer-Optimized Neurological Stimulation,” in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Vol. 13, No. 4*, 1991, pp. 1692–1693.

[245] R. B. North, K. R. Fowler, D. J. Nigrin, and R. Szymanski, “Patient-Interactive, Computer-Controlled Neurological Stimulation System: Clinical Efficacy in Spinal Cord Stimulator Adjustment,” *J. Neurosurg.*, vol. 76, pp. 967–972, 1992.

[246] K. R. Fowler and H. B. Land, “System Design that Minimizes Both Missed Detections and False Alarms,” in *IEEE Instrumentation and Measurement Technical Conference*, 2004.

[247] S. Cooper *et al.*, “Predicting Protein Structures with a Multiplayer Online Game, Supplementary Information,” *Nature*, vol. 466, no. 7307, pp. 1–45, 2010.

[248] J. A. Aguilar, “Design Assurance Guide,” Aerospace, El Segundo, CA 90245-2808, 2009.

[249] A. Abran, D. St-Pierre, M. Maya, and J.-M. Desharnais, “Full Function Points for Embedded and Real-Time Software,” in *UKSMA Fall Conference*, 1998, pp. 1–14.

[250] L. Lavazza and C. Garavaglia, “Using Function Points to measure and estimate real-time and embedded software: Experiences and guidelines,” in *2009 3rd International Symposium on Empirical Software Engineering and Measurement, ESEM 2009*, 2009, pp. 100–110.

[251] L. M. Laird and M. C. Brennan, “Chapter 4: Measuring Size,” in *Software Measurement and Estimation: A Practical Approach*, First Edit., 2006, pp. 48–51.

[252] W. T. Ward, “Calculating the Real Cost of Software Defects,” *Hewlett-Packard J.*, no. October, pp. 55–58, 1991.

[253] F. W. Blakely and M. E. Boles, “A Case Study of Code Inspections,” *Hewlett-Packard J.*, no. October, pp. 58–63, 1991.

[254] P. Koopman, “Embedded Software Testing (lecture notes),” 2014.

[255] K. R. Fowler, “Chapter 1: Best Practices in Mission-Assured, Mission-Critical, and Safety-Critical Systems,” in *Mission-Critical and Safety-Critical Systems Handbook*, K. R. Fowler, Ed. Newnes, an imprint of Elsevier, 2010, pp. 66–82.

[256] K. R. Fowler, “Chapter 13: Case Study 10 - Programmer for Implanted Stimulators,” in *What Every Engineer Should Know About Developing Real-Time Embedded Products*, Boca Raton, FL, USA: CRC Press, 2008, pp. 303–331.

Chapter 8 - Appendices

This chapter contains the appendices, which provide support and additional materials to the main body of this dissertation. It contains definitions, proposed studies for future research, suggestions for tool webpages, simulations, data, and Matlab source code.

Appendix A - Definitions

Design assurance – "*a formal, systematic process that augments the design effort and increases the probability of product design conformance to requirements and mission needs* [248]."
Development –
Embedded system – "*a combination of computer hardware and software, and perhaps additional mechanical or other parts, designed to perform a dedicated function."* [Jack Ganssle and Michael Barr (2003). *Embedded Systems Dictionary*, CMP Books, pp. 90 – 91.] Or, in other words, a system that depends on a computer, its software, and its peripheral circuitry as a critical element in its function.

NRE – *non-recurring engineering* costs that accumulate over the design and development phases.
COGS – *cost of goods sold* during manufacturing and production.
COTS – *commercial off-the-shelf*; refers to components and subsystems that are readily available for purchase.
ROTS – *rugged off-the-shelf*; like COTS but for extreme environments.
FTE – *full-time equivalent*, which is the number of people needed to work a task at full-time effort.
Lifecycle – the entire duration for which the product exists, from concept to disposal.
Up-front lifetime buy – is a phrase that refers to stocking up components or subsystems to inventory over the lifetime of the product. It often is used when vendors announce that they will make components or subsystems obsolete; sometimes it is called an *obsolete-part buy*.
Premium – is the interest on a loan to buy and inventory COTS subsystems over the lifetime of the final product; this is one cost of product longevity.

Failure – The inability of a system, subsystem, or component to perform its required function.
Fault – Undesired anomaly in the functional operation of an equipment or system. the occurrence of an undesired state, which may be the result of a failure.
Hazard – A potentially unsafe condition resulting from failures, malfunctions, external events, errors, or a combination thereof (SAE ARP-4761).

Safety – The ability of a system to exclude certain undesired events (i.e., mishaps) during stated operation under stated conditions for a stated time. the ability of a system or product to operate with a known and accepted level of mishap risk. A built-in system characteristic.
Success – A system operates without failure or mishap and meets the requirements of the customer and designer(s).

These definitions derive from the 2013 edition of ISO Guide 51, "Safety aspects – Guidelines for their inclusion in standards," as follows:

> ***Safety** - freedom from risk which is not tolerable.*

and

> ***tolerable risk** - level of risk which is accepted in a given context based on the current values of society, (NOTE: For the purposes of this Guide, the terms "acceptable risk" and "tolerable risk" are considered to be synonymous.)*

where,

> ***Risk** - combination of the probability of occurrence of harm and the severity of that harm. NOTE: The probability of occurrence includes the exposure to a hazardous situation, the occurrence of a hazardous event, and the possibility to avoid or limit the harm.*

and

> ***harm** - injury or damage to the health of people, or damage to property or the environment,*

where

> ***hazard** - potential source of harm,*

and

> ***hazardous event** - event that can cause harm,*

based on a

> ***hazardous situation** - circumstance in which people, property or the environment are exposed to one or more hazards.*

Appendix B - Simulation of Latent Faults

This appendix is a large simulation to show that regardless of how vigilant we are, we cannot eliminate errors in design and development. While we can reduce the probability and impact of failure and errors, we cannot eliminate them entirely. More importantly by skipping the known good practices of reviews for modules, subsystems, and system and the equivalents in test, we are guaranteed to have errors.

I simulated a medium-sized system. I based the simulation on a small version of a satellite imaging system from which I had personal experience with development; it received images from up to eight cameras and could compress and multiplex the images into a serial data stream; it had about 17,000 lines of C code.

To model the system, I used two simple concepts. The first is an action, which is defined as a single operation, generally a single function. The second is an error, which is any mistake or misconception.

An action can be any one of the following items:

- Line of code (LOC)
- Component
- Circuit trace
- Signal (input or output)
- Command
- Human interaction

An action is simple like a line of code (LOC) but more general than describing only software. Actions are not function points (championed by Capers Jones) [211], [249], [250] nor feature points [251]. For the sake of this simulation, an action is a useful construct and the detail of a function point is unnecessary.

An error can be any one of the following:

- Design or concept mistake
- Ignorance of conditions or interactions
- Typo mistake
- Translation mistake in either understanding or requirement.

Figure 8.1 illustrates the simulated system. It covers every type of action defined above. The model contains 16,000 actions (LOC + signals + interactions).

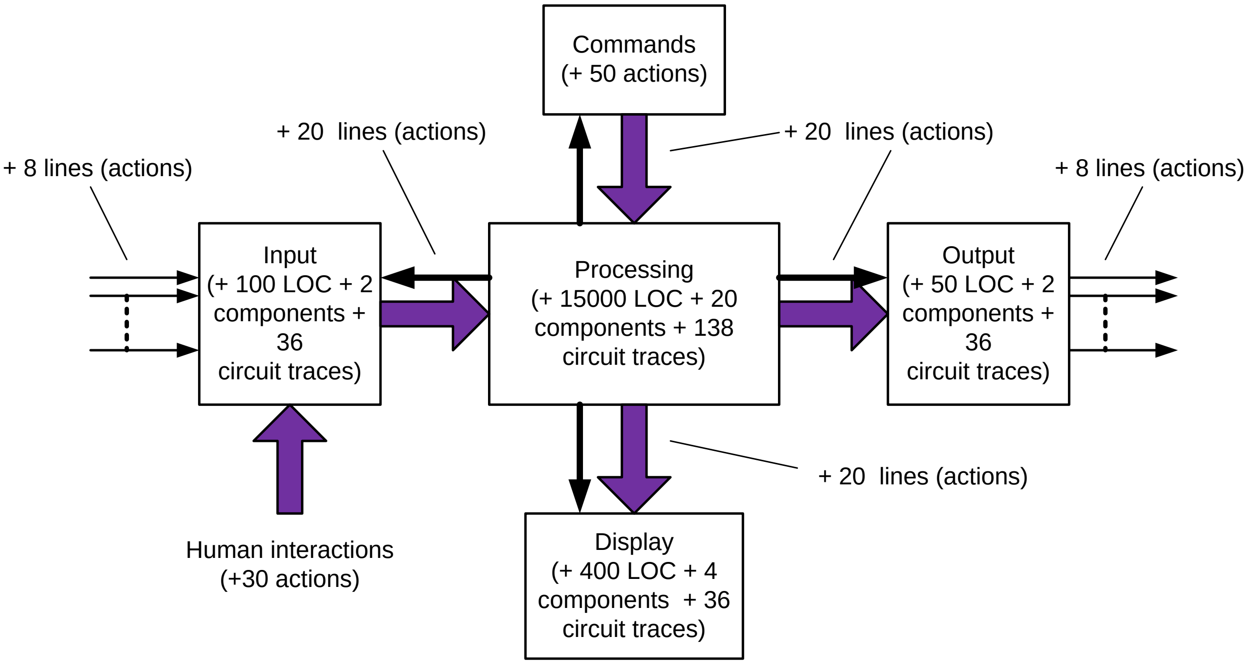

Commands
(+ 50 actions)
+ 20 lines (actions)
+ 20 lines (actions)
+ 8 lines (actions)
+ 8 lines (actions)
Input
(+ 100 LOC + 2 components + 36 circuit traces)
Processing
(+ 15000 LOC + 20 components + 138 circuit traces)
Output
(+ 50 LOC + 2 components + 36 circuit traces)
+ 20 lines (actions)
Human interactions
(+30 actions)
Display
(+ 400 LOC + 4 components + 36 circuit traces)

Figure 8.1 Example real-time, embedded system used in the simulations.

I prepared a Matlab program (see below) to run Monte Carlo simulations of randomly injected errors that were then randomly discovered and fixed. For the sake of simplicity and to run any sort of simulation within finite time, I conservatively assumed the best case that the errors were independent and uncorrelated, which they probably are not – particularly if the errors result in a cascade mode of failures. I also assumed that the reviews and tests that diagnosed the errors were randomly applied, independent, and uncorrelated, which they probably are not. Figure 8.2 illustrates the general form of the simulation program.

Each run of the simulation goes through a million iterations. Each run has 7 phases:

1: Conceptual

2: Preliminary Design

3: Critical Design

4: Test and Integration

5: Manufacturing and Production

6: Commissioning, Installation, Technical Evaluation

7: Operation

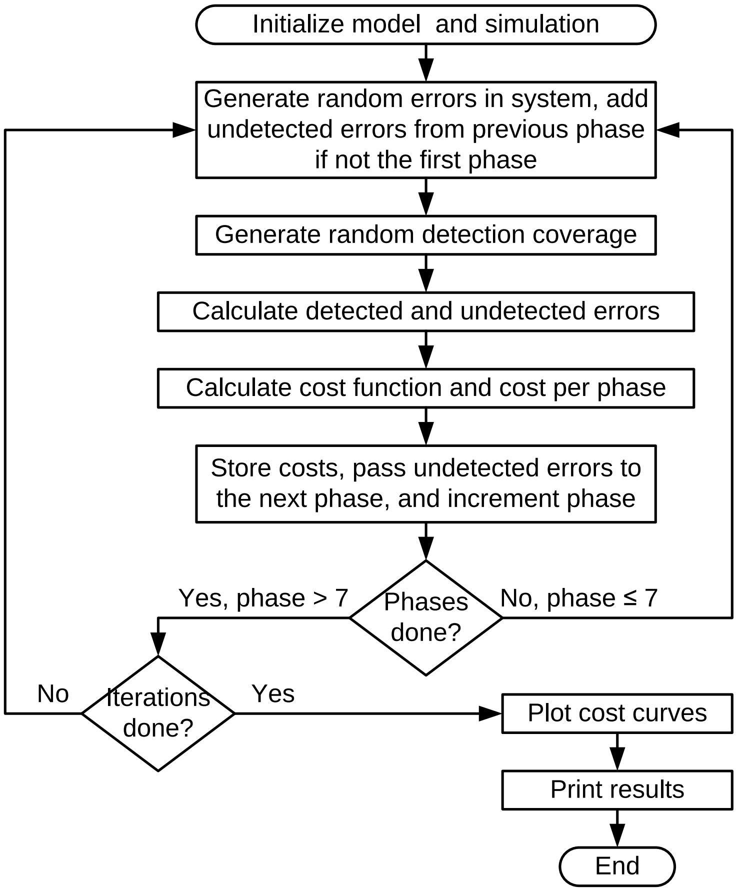

Initialize model and simulation
Generate random errors in system, add undetected errors from previous phase if not the first phase
Generate random detection coverage
Calculate detected and undetected errors
Calculate cost function and cost per phase
Store costs, pass undetected errors to the next phase, and increment phase
Yes, phase > 7
Phases done?
No, phase ≤ 7
No
Iterations done?
Yes
Plot cost curves
Print results
End

Figure 8.2 Algorithm used in the simulations.

The simulation is very optimistic for detecting errors. It assumes the following:

- 0.1% error insertion (1 in 1000 actions or LOC, many companies do not do this well)
- 80% coverage of errors by review [210], [252], [253]
- 50% coverage of errors by test [254]
- Independent distributions.

The cost function to repair errors randomly covers uncertainty in the type of error and its complexity; errors, and hence cost, are independent. I ran many simulations with coverage for review and test varying between 30% and 90%.

Figure 8.3 gives the mean cost to repair for a quadratic cost function. The simulation allowed for maturing interactions for the errors between phases; it also did not introduce any errors after Phase 5, which means that remaining errors could be discovered in Phases 6 and 7.

With average 1.78 errors remaining in Phase 5, the simulation's peak cost to fix is close to the results in the INCOSE report [23].

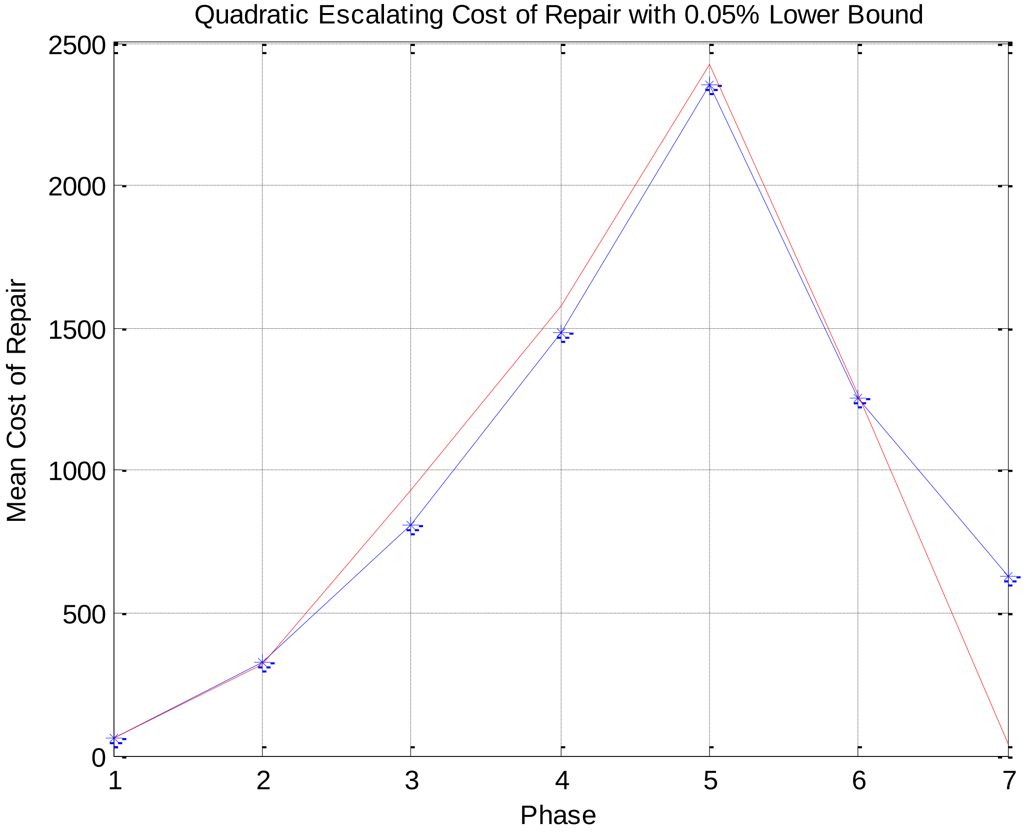


Figure 8.3 Simulated cost of detecting and fixing cumulative errors in a 16,000-action, example system. The blue line is the mean cost over 10^6 runs. The red line represents 5,000 of the lowest overall-cost runs; it is the lower bound. The assumption is that no more design errors are inserted after Phase 5 (i.e., none inserted in Phases 6 and 7). A quadratic cost function is close to the findings in the INCOSE report cited in the previous chapter for average, escalating cost.

Figure 8.4 provides the combined simulated costs to repair for linear, quadratic, and cubic cost functions.

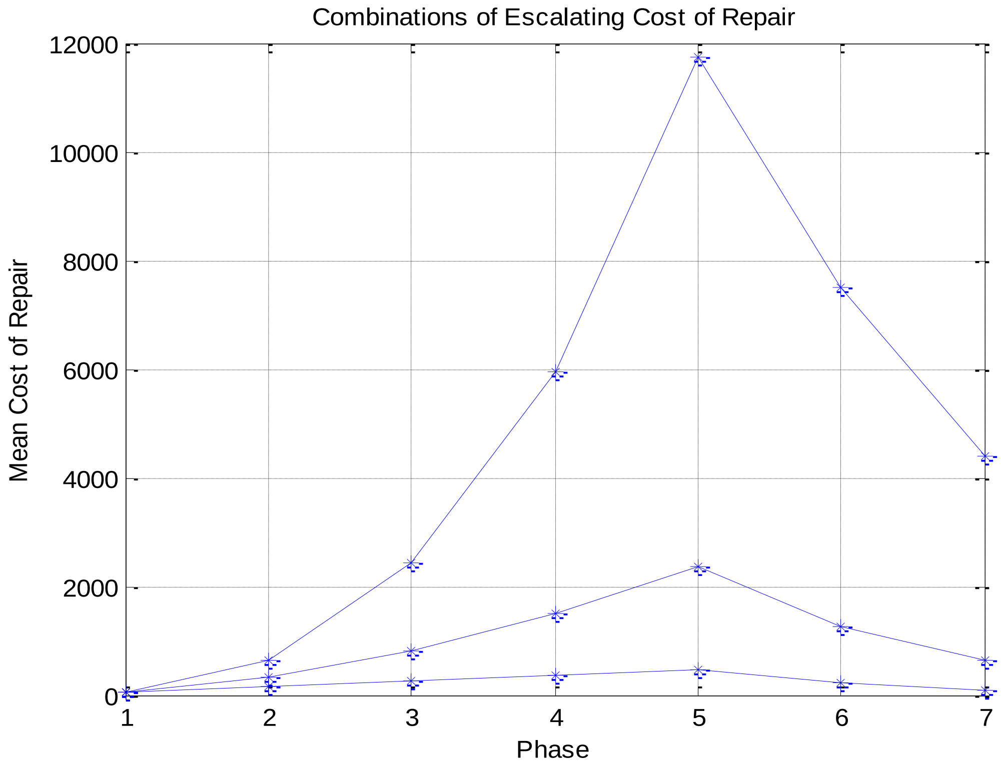


Figure 8.4 Simulated cost of detecting and fixing cumulative errors in a 16,000-action, example system. Each blue line is the mean cost over 10^6 runs. The lowest represents a linear cost function that increases with each phase. The middle one, which is represented in Figure 8.3, represents a quadratic cost function. The top line represents a cubic cost function. The assumption is that no more design errors are inserted after Phase 5 (i.e., none inserted in Phases 6 and 7).

The simulations had total repair costs as follows:

- Total linear escalating costs over phases = 1477
- Total quadratic escalating costs over phases = 5599
- Total cubic escalating costs over phases = 23733
- Total 4-order escalating costs over phases = 107577

Fourth-order costs were not illustrated in either figure. They were too large and did not compare well with the INCOSE report's results. The total errors injected/remaining, detected, and undetected are listed in Table 8.1.

Table 8.1 Errors in the System Per Phase, with Coverage of 80% for Thorough Review And 50% for Thorough Test.

Phase	Total Errors Injected or Remaining	Undetected Errors	Detected and Fixed Errors
1	16.00	1.60	14.40
2	17.60	1.76	15.84
3	17.76	1.77	15.99
4	17.77	1.78	15.99
5	17.77	1.78	15.99
6	1.78	0.18	1.60
7	0.18	0.02	0.16

Clearly, the simulation shows that even for very optimistic constraints, designing, developing, and producing an error-free system is essentially impossible. The sections that follow the Matlab code give more results.

Simulations for Cost of Error Detection and Repair

Matlab Code:

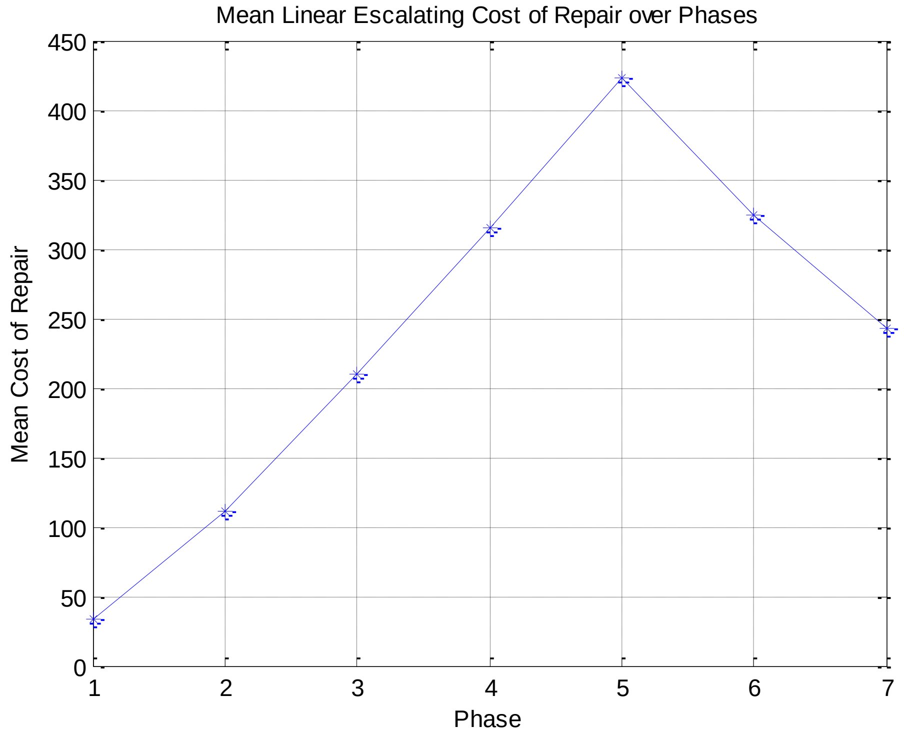
```
% ----------------------------------------------------------------------
%   This is test8d.m. By Kim R. Fowler, 2 September 2015
%
%   It simulates generation of random errors with remediation through
%   module review and unit test through 7 phases of product development.
%   Here are the phases:
%       1 = concept design
%       2 = preliminary design
%       3 = critical design
%       4 = test and integration
%       5 = production and manufacturing
%       6 = commissioning, evaluation, installation, early operation
%       7 = operation
%
%   It calculates repair cost for each error with a randomized function.
%   It uses 3 different orders of cost escalation tied to the phases:
%   linear, quadratic, cubic, and a flat cost that does not escalate.
%
%   It passes undetected errors to the next phase. Where it repeats the
%   the cost calculations.
%
%   The cost calculation is limited between 0.019 and 1000 times over.
% -----------------------------------------------------------------------
clf
sim = 1000;                 %number of iterations
phase = 7;                  %number of phases within development/iteration
N = 1000;                   %number of simulation runs in one iteration
LOC = 16000;                %number actions (operations/interactions/LOC)
review = .8;                %review coverage to uncover errors
test = .5;                   %test coverage to uncover errors
bug = .001;                 %error rate in entire system, 1 in 1000 actions
phcutoff = 6;               %phase where no more errors are introduced
norevortst = 0;             %marks early phases without review or test
tot1 = zeros(N,phase);      %total errors/run(or system)/phase
tot2 = zeros(N,phase);      %undetected errors/run(or system)/phase
tot3 = zeros(N,phase);      %detected errors/run(or system)/phase
addpherr = zeros(N,3,phase);    %errors per phase
addphcost = zeros(N,6,phase);   %cost of errors per phase
avgcost = zeros(phase,4,sim);   %average cost to fix errors/phase
stdcost = zeros(phase,4,sim);   %standard deviation in cost to fix errors
avgerr = zeros(phase,3,sim);    %average errors per phase: inserted,
                                %undetected, and detected
lowcnt = 500;                   %lower bound to display, this is 0.05%
satval = 1000000;
Y = satval*ones(lowcnt,6,phase);%saturate with large values to begin
                                %sorting on minimum values
X = zeros(lowcnt,6,phase);
% -----------------------------------------------------------------------
%   Run simulations of random generated errors with random detection
%   by review and test. These are parameter initializations for random
%   error generation, insertion, and detection.
%   No. of simulations/loop x no. of loops = total no. of simulations
% -----------------------------------------------------------------------
for indx = 1:sim
    Errnodet = zeros(LOC,N);    %initialize the undetected errors matrix
                                %that passes errors to the next phase
% -----------------------------------------------------------------------
%   Main loop is for simulating each phase of development.
% -----------------------------------------------------------------------
  for ph = 1:phase
    for i=1:N
        % ---------------------------------------------------------------
        %   Randomly seed system with errors at the specified rate. Then
        %   add in the undetected errors from the previous phase. In
        %   Phase 1, there are no previous errors, so Errnodet is
        %   initialized to 0. Assume design errors occur only in the
        %   first four phases. phcutoff = phase where no more errors are
        %   introduced.
        % ---------------------------------------------------------------
        if ph < phcutoff
            Syserr = (floor(bug + rand(LOC,1)))|Errnodet(1:LOC,i);
        else
            Syserr = Errnodet(1:LOC,i);
        end
        % ---------------------------------------------------------------
        %   Calculate the detected errors to correct in this system (run).
        % ---------------------------------------------------------------
        Revdt1 = floor(review + rand(LOC,1));   %review detects errors
        Tstdt1 = floor(test + rand(LOC,1));     %test detects errors
```

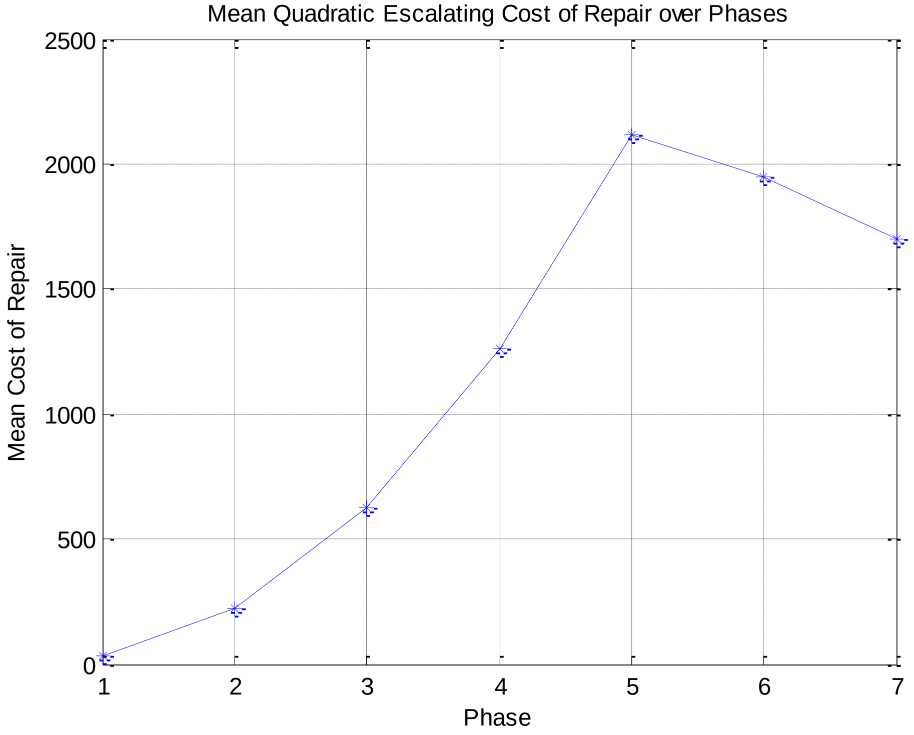
```
      if ph <= norevortst                  %skip early reviews and tests
          Errdet = zeros(LOC,1);              %no detected errors
      else
          Errdet = Syserr & (Revdt1|Tstdt1);  %detected errors
      end
      % --------------------------------------------------------------
      %   Calculate the undetected errors to pass to the next phase.
      % --------------------------------------------------------------
      Errnodet(1:LOC,i) = Syserr - Errdet;    %undetected errors
      % --------------------------------------------------------------
      %   Sum the errors in each system to prepare histogram display.
      % --------------------------------------------------------------
      tot1(i,ph) = sum(Syserr);               %inserted errors/system
      tot2(i,ph) = sum(Errnodet(1:LOC,i));    %undetected errors/system
      tot3(i,ph) = sum(Errdet);               %detected errors/system
      addpherr(i,:,ph) = [tot1(i,ph) tot2(i,ph) tot3(i,ph)];
      % --------------------------------------------------------------
      %   Calculate the escalating costs. This is a function of phase.
      %   Cost function is [1/X] - 1. It is limited to 1000 x by adding
      %   a very small increment of .001 to the random denominator:
      %   [1/(random X +.001)] - 0.98
      % --------------------------------------------------------------
      CX = (1./((rand(tot3(i,ph),1))+.001))-.98;%cost function, can be
      %a very small effort (.019) to very large (1000)
      fcost = sum(CX);                    %flat costs, no escalation
      lcost = ph * fcost;                 %linear escalating cost
      qcost = ph * lcost;                 %quadratic escalating cost
      ccost = ph * qcost;                 %cubic escalating cost
      addphcost(i,1:4,ph) = [fcost lcost qcost ccost];
  % record the running total of project costs (sum flat components)
      addphcost(i,5,ph) = addphcost(i,5,ph) + fcost;
  % record the running total of project costs (sum linear components)
      addphcost(i,6,ph) = addphcost(i,6,ph) + lcost;
  end
  % ------------------------------------------------------------------
  %   Store mean and standard deviation costs in each phase. These
  %   will be used in the plotting subroutine below. (min and max costs
  %   are not useful - min gives 0 and max is huge, which is expected
  %   for random generation.) Store the average errors per phase:
  %       inserted (or carry over), undetected, and detected
  % ------------------------------------------------------------------
  avgcost(ph,1:4,indx) = mean(addphcost(1:N,1:4,ph));
  %stdcost(ph,1:4,indx) = std(addphcost(1:N,1:4,ph));
  avgerr(ph,:,indx) = mean(addpherr(1:N,:,ph));
end
% --------------------------------------------------------------------
% Sort and select for the runs with lowest total values.
% --------------------------------------------------------------------
  for i = 1:lowcnt        %select lowest from this set of runs
      [minval,minind] = min(addphcost(:,5,phase));
      X(i,:,:) = addphcost(minind,:,:);
      addphcost(minind,5,phase) = satval;
  end
  Z = [X;Y];
  for i = 1:lowcnt        %select lowest thus far and save for next run
      [minval,minind] = min(Z(:,5,phase));
      Y(i,:,:) = Z(minind,:,:);
```

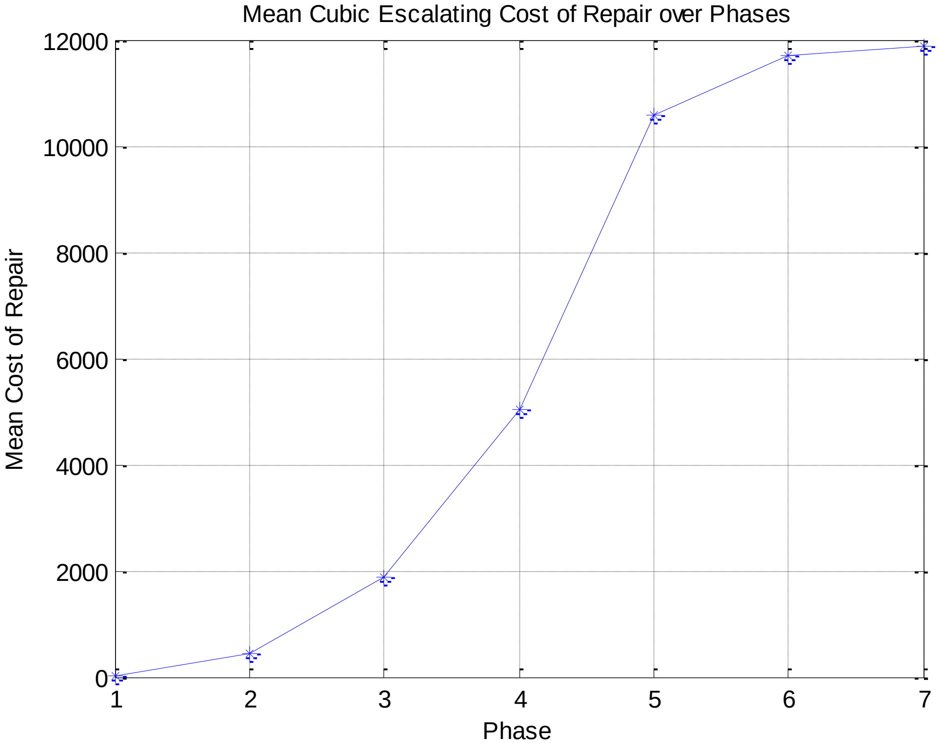
```
        Z(minind,5,phase) = satval;
    end
  % --------------------------------------------------------------------
  % Print out current iteration simulation to give an indication
  % how close to the end it is. (This script runs about 10 hours.)
  % --------------------------------------------------------------------
  if indx > 700
      fprintf(' %.f',indx)
  end
end
% ----------------------------------------------------------------------
%   Iterating through simulation loops gets around memory constraints of
%   too-large matrices.
% ----------------------------------------------------------------------
temp1 = zeros(phase,4);
%temp2 = zeros(phase,4);
temp3 = zeros(phase,3);
for indx = 1:sim
    temp1(:,:) = temp1(:,:) + avgcost(:,:,indx);
    %temp2(:,:) = temp2(:,:) + stdcost(:,:,indx);
    temp3(:,:) = temp3(:,:) + avgerr(:,:,indx);
end
cost = temp1(:,:)/sim;
%sdevcost = temp2(:,:)/sim;
%maxdev = cost + sdevcost;
toterr = temp3(:,:)/sim;
% ----------------------------------------------------------------------
%   Plot the aggregate costs over the phases.
% ----------------------------------------------------------------------
for j = 1:4
    plot(cost(1:phase,j),'b*-')
    ylabel('Mean Cost of Repair')
    xlabel('Phase')
    grid
    if j==1
        title('Mean Flat Cost of Repair over Phases')
    elseif j==2
        title('Mean Linear Escalating Cost of Repair over Phases')
    elseif j==3
        title('Mean Quadratic Escalating Cost of Repair over Phases')
    else
        title('Mean Cubic Escalating Cost of Repair over Phases')
    end
    fprintf('\n')
    fprintf('Press spacebar\n')
    pause
end
plot(cost(1:phase,3),'b*-')
hold on
plot(cost(1:phase,2),'b*-')
plot(cost(1:phase,1),'b*-')
ylabel('Mean Cost of Repair')
xlabel('Phase')
title('Combinations of Costs of Repair')
grid
fprintf('\n')
fprintf('Press spacebar\n')
```

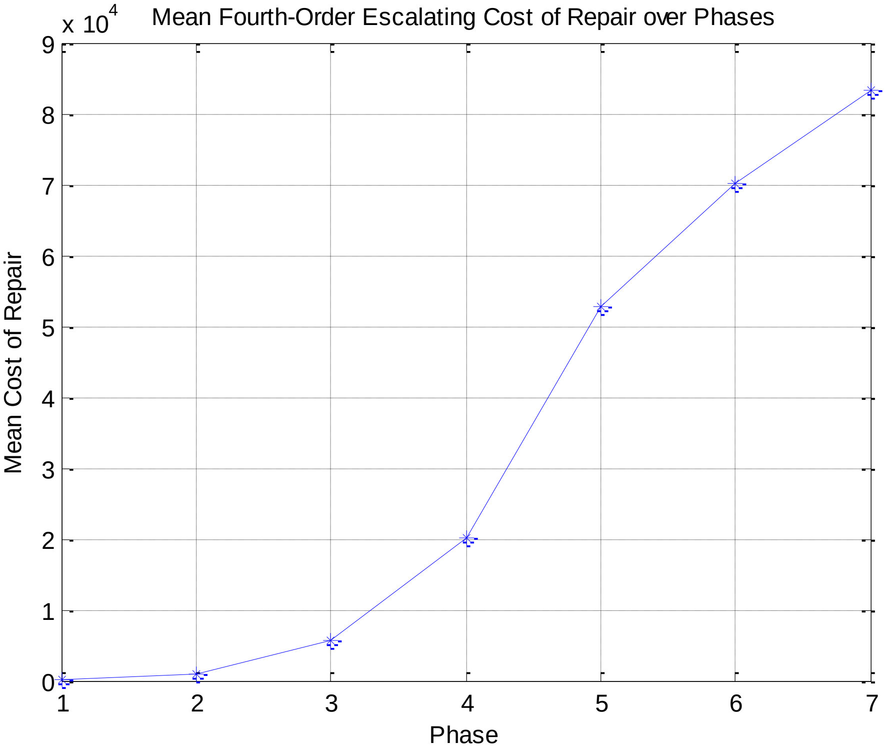
```
pause
hold off
% ----------------------------------------------------------------------
%   Plot the average linear costs over the phases with the lower bound.
% ----------------------------------------------------------------------
plot(cost(1:phase,1),'b*-')
hold on
for k = 1:phase
    Ylinplot(1,k) = mean(Y(1:lowcnt,1,k));
end
plot(Ylinplot(1,1:phase),'r-')
ylabel('Mean Cost of Repair')
xlabel('Phase')
title('Flat Cost of Repair with 0.05% Lower Bound')
grid
fprintf('\n')
fprintf('Press spacebar\n')
pause
hold off
% ----------------------------------------------------------------------
%   Plot the average quadratic costs over the phases with the lower bound.
% ----------------------------------------------------------------------
plot(cost(1:phase,2),'b*-')
hold on
for k = 1:phase
    Ylinplot(1,k) = mean(Y(1:lowcnt,2,k));
end
plot(Ylinplot(1,1:phase),'r-')
ylabel('Mean Cost of Repair')
xlabel('Phase')
title('Linear Escalating Cost of Repair with 0.05% Lower Bound')
grid
fprintf('\n')
fprintf('Press spacebar\n')
pause
hold off
% ----------------------------------------------------------------------
%   Print the parameters and aggregate costs over the phases.
% ----------------------------------------------------------------------
fprintf('\n')
totN = N * sim;
fprintf('Number of simulation runs  = %.f\n',totN)
fprintf('Number of operations/interactions/LOC  = %.f\n',LOC)
fprintf('Generated error rate  = %.4f\n',bug)
fprintf('Review coverage  = %.2f\n',review)
fprintf('Test coverage  = %.2f\n',test)
fprintf('Cut off for introduction of errors, phase = %.f\n',phcutoff)
fprintf('Lower bound, the guard rail = %.4f\n',lowcnt/totN)
fprintf('Phase that reviews and tests begin = %.f\n',norevortst+1)
fprintf('\n')
totcost = sum(cost);
fprintf('Total flat costs over phases = %.f\n',totcost(1))
fprintf('Total linear escalating costs over phases = %.f\n',totcost(2))
fprintf('Total quadratic escalating costs over phases = %.f\n',totcost(3))
fprintf('Total cubic escalating costs over phases = %.f\n',totcost(4))
fprintf('\n')
fprintf('Errors in the system per phase:\n')
fprintf('Phase     Total   Undetect   Detect\n')
fprintf(' 1:        %3.2f',toterr(1,1));fprintf('      %3.2f',toterr(1,2));
fprintf('      %3.2f\n',toterr(1,3))
fprintf(' 2:        %3.2f',toterr(2,1));fprintf('      %3.2f',toterr(2,2));
fprintf('      %3.2f\n',toterr(2,3))
fprintf(' 3:        %3.2f',toterr(3,1));fprintf('      %3.2f',toterr(3,2));
fprintf('      %3.2f\n',toterr(3,3))
fprintf(' 4:        %3.2f',toterr(4,1));fprintf('      %3.2f',toterr(4,2));
fprintf('      %3.2f\n',toterr(4,3))
fprintf(' 5:        %3.2f',toterr(5,1));fprintf('      %3.2f',toterr(5,2));
fprintf('      %3.2f\n',toterr(5,3))
fprintf(' 6:        %3.2f',toterr(6,1));fprintf('      %3.2f',toterr(6,2));
fprintf('      %3.2f\n',toterr(6,3))
fprintf(' 7:        %3.2f',toterr(7,1));fprintf('      %3.2f',toterr(7,2));
fprintf('      %3.2f\n',toterr(7,3))
fprintf('\n')
fprintf('The cost function is a random function of (1/X)-1,')
fprintf('where X is a random variable between 0 and 1; it leads to \n')
fprintf('cost values between 0.019 and 1000. This is script “test8d.m”. \n')
```

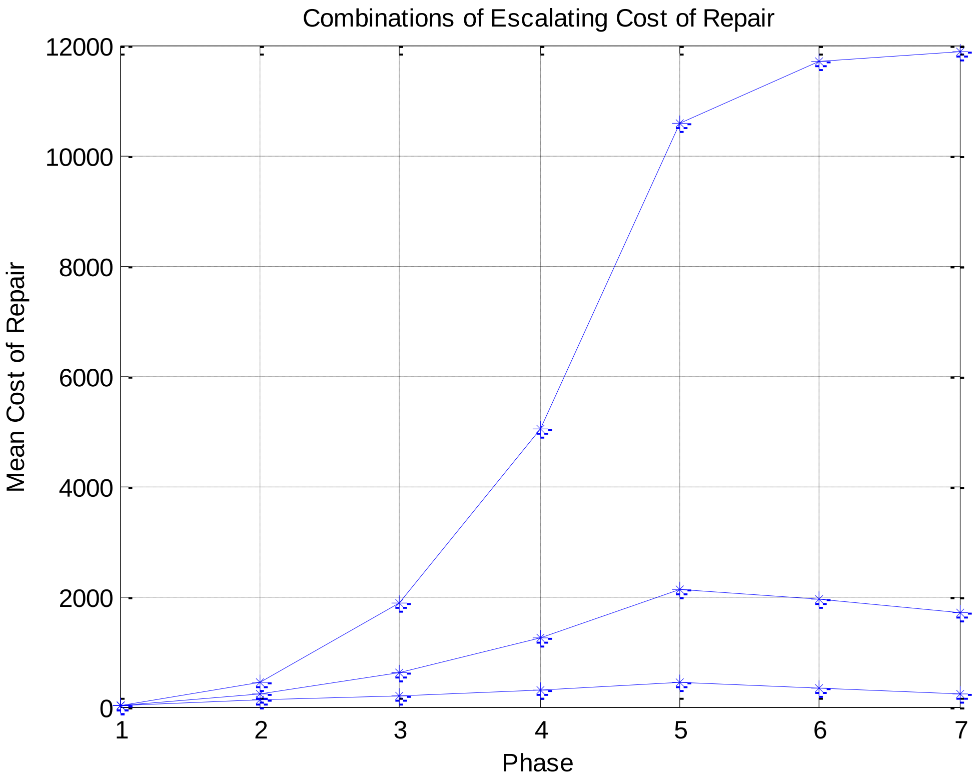
Results for Coverage of 20% by Review, 20% by Test, All Phases

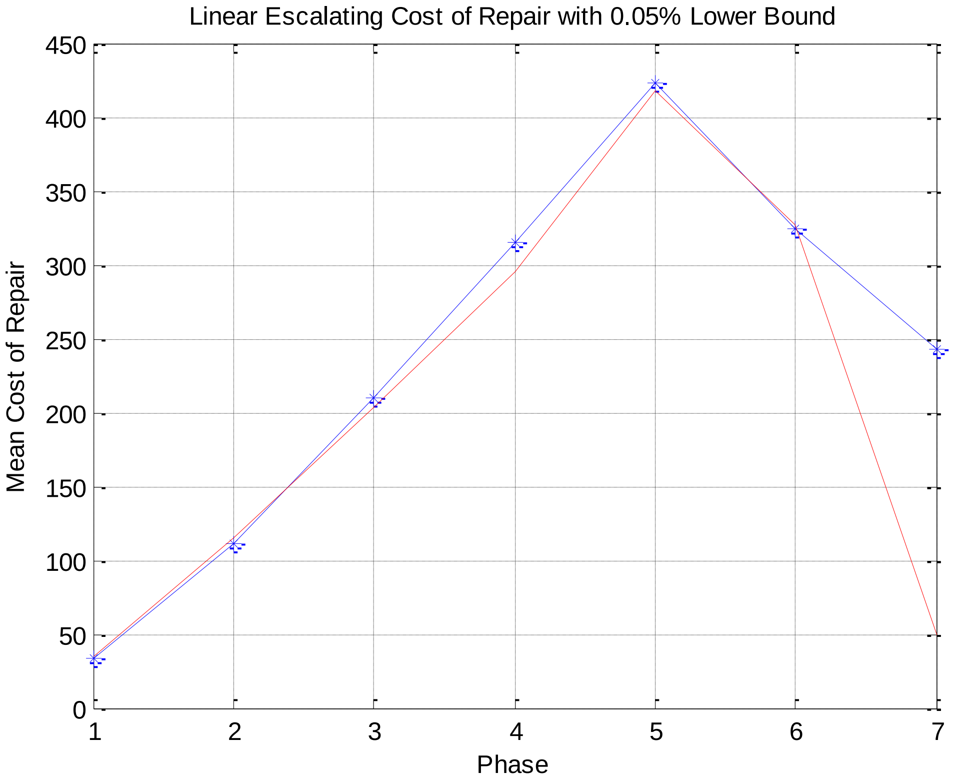


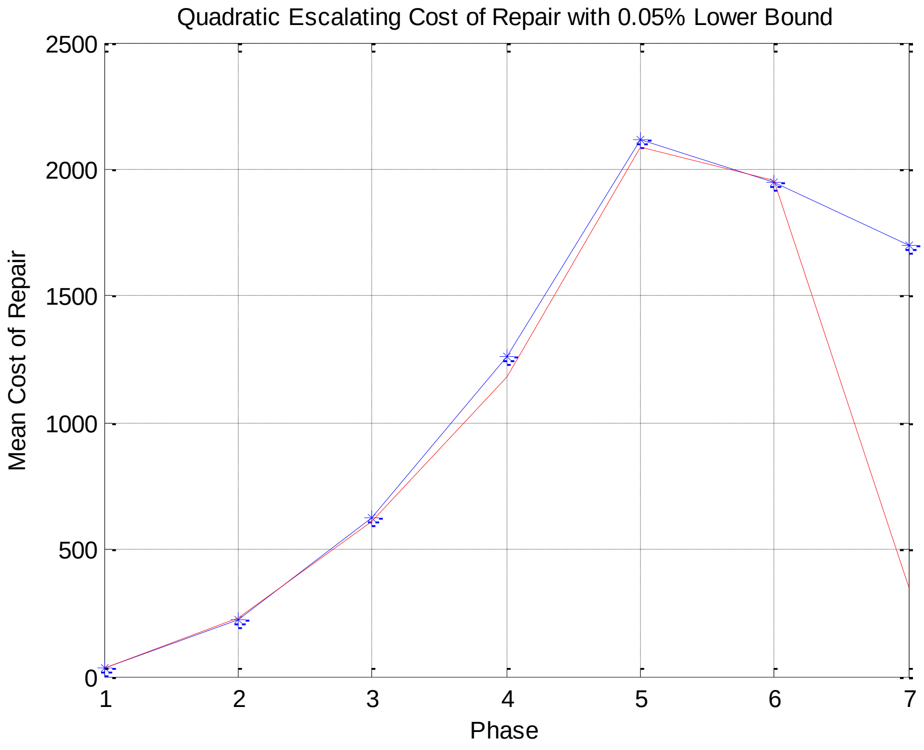



Mean Cubic Escalating Cost of Repair over Phases
12000
10000
8000
6000
4000
2000
0
Mean Cost of Repair
1
2
3
4
5
6
7
Phase


x 10^4
Mean Fourth-Order Escalating Cost of Repair over Phases
9
8
7
6
5
4
3
2
1
0
Mean Cost of Repair
1
2
3
4
5
6
7
Phase


Combinations of Escalating Cost of Repair
12000
10000
8000
6000
4000
2000
0
Mean Cost of Repair
1
2
3
4
5
6
7
Phase


Linear Escalating Cost of Repair with 0.05% Lower Bound
450
400
350
300
250
200
150
100
50
0
Mean Cost of Repair
1
2
3
4
5
6
7
Phase


Quadratic Escalating Cost of Repair with 0.05% Lower Bound
Mean Cost of Repair
2500
2000
1500
1000
500
0
1
2
3
4
5
6
7
Phase

Number of simulation runs = 1000000

Number of operations/interactions/LOC = 16000

Generated error rate = 0.0010

Review coverage = 0.20

Test coverage = 0.20

Cut off for introduction of errors, phase = 6, i.e., last errors are injected in Phase 5

Lower bound, the guard rail = 0.0005

Phase that reviews and tests begin = 1

Total linear escalating costs over phases = 1661

Total quadratic escalating costs over phases = 7909

Total cubic escalating costs over phases = 41565

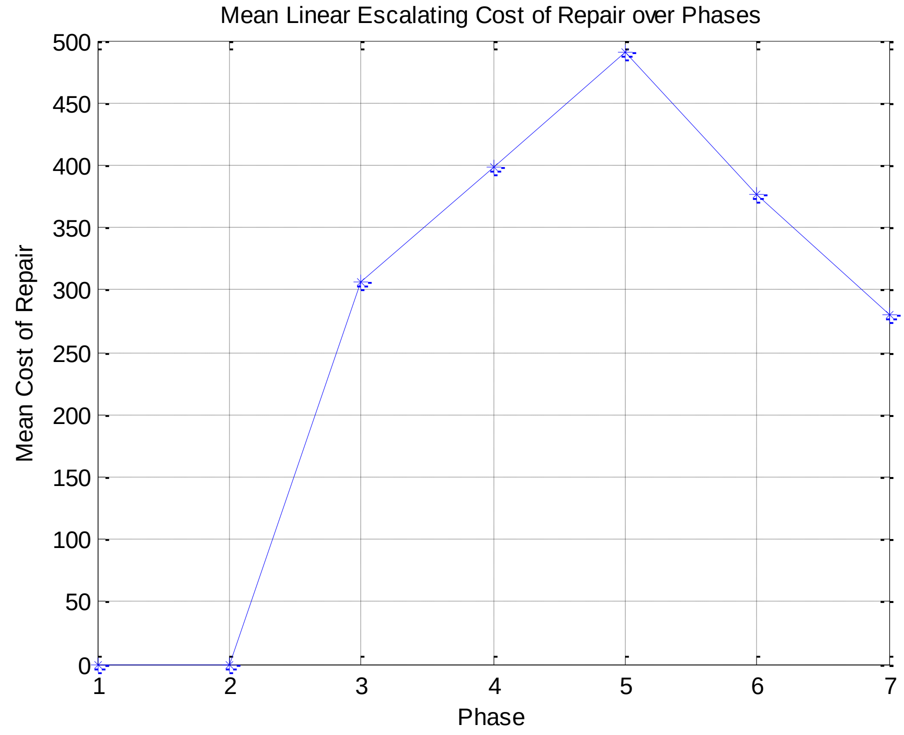
Total 4-order escalating costs over phases = 232989

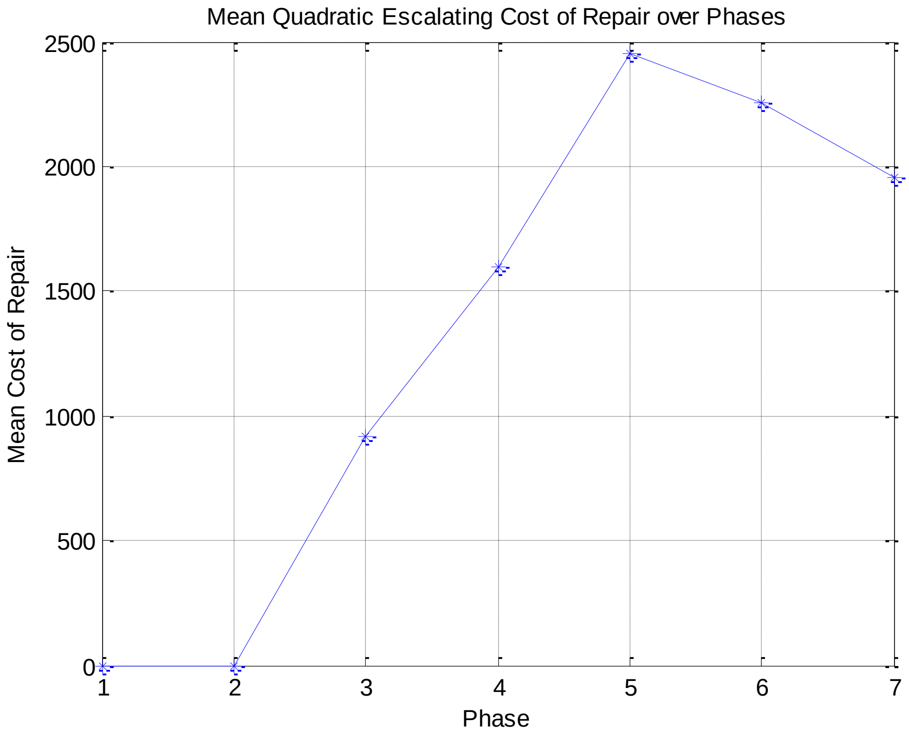
Errors in the system per phase:

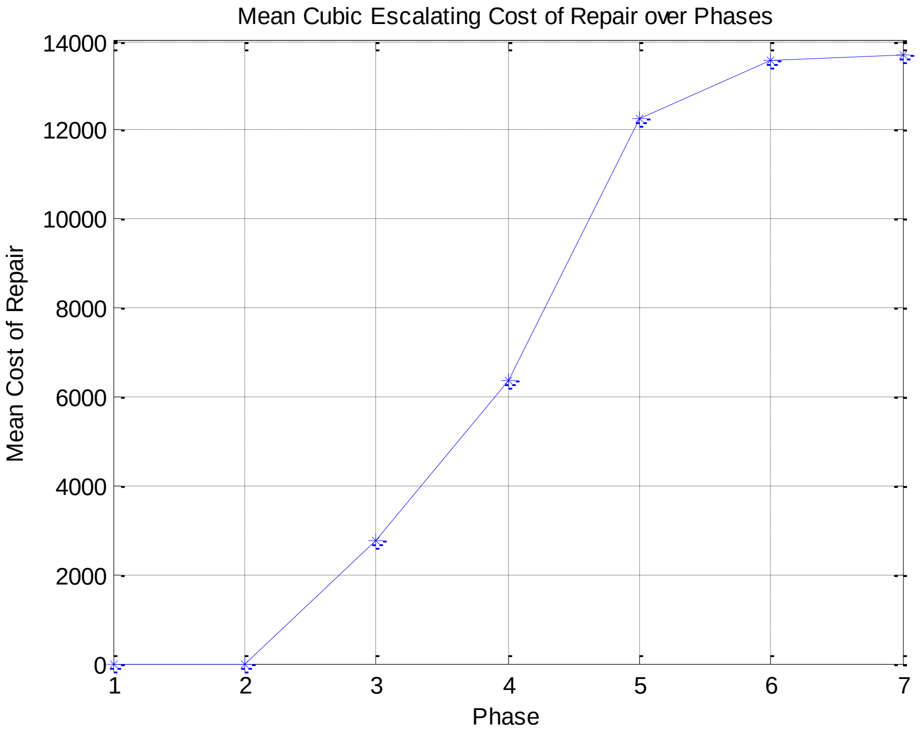
Phase	Total	Undetect	Detect
1:	16.00	10.24	5.76
2:	26.23	16.79	9.44
3:	32.77	20.98	11.79
4:	36.95	23.65	13.30
5:	39.63	25.36	14.26
6:	25.36	16.24	9.13
7:	16.24	10.39	5.84

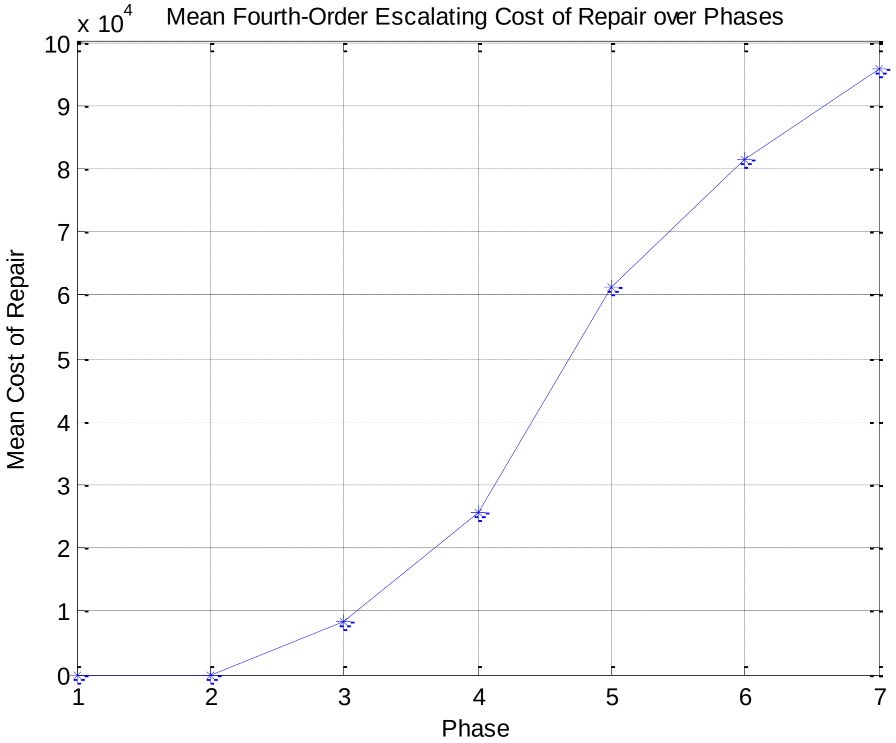
The cost function is a random function of (1/X)-1,where X is a random variable between 0 and 1; it leads to cost values between .019 and 1000. This is script “test8c.m”.

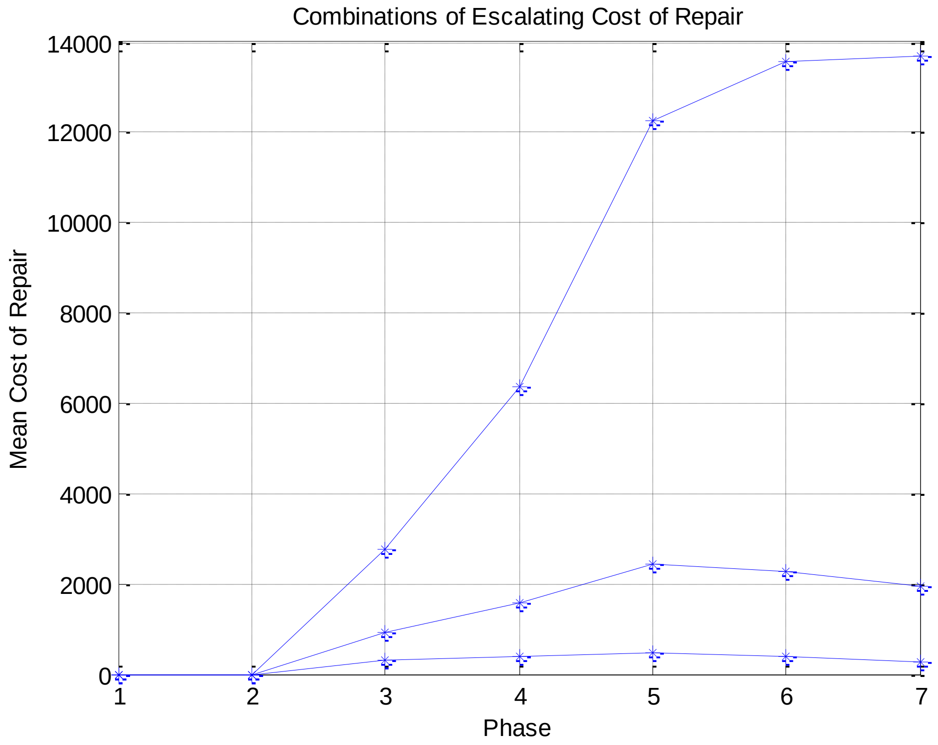
Results for Coverage of 20% by Review, 20% by Test, Phases 3 through 7

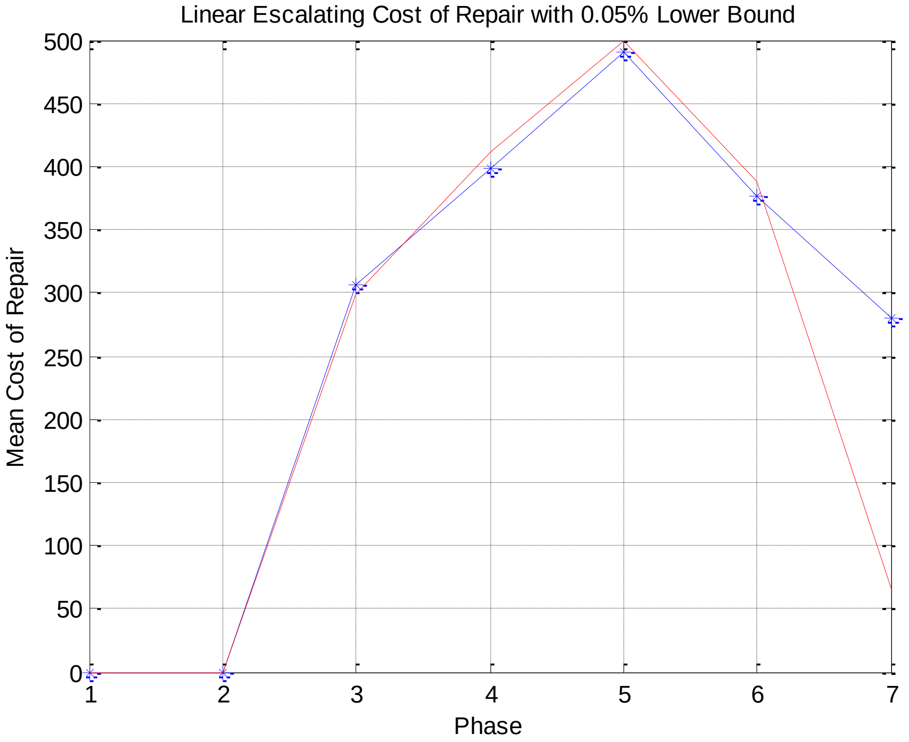


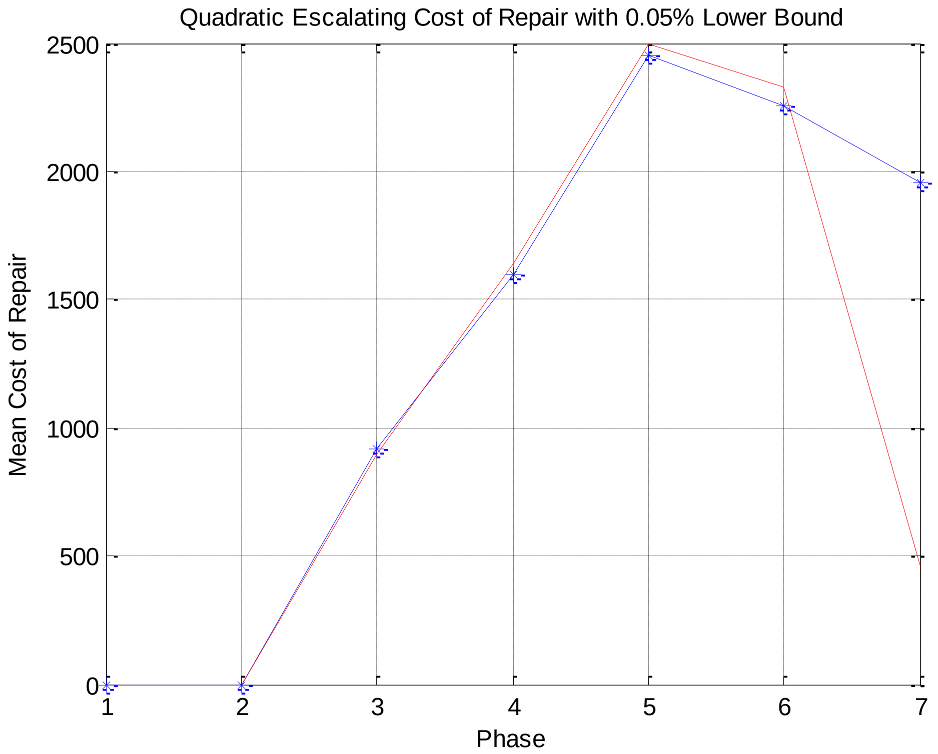



Mean Cubic Escalating Cost of Repair over Phases
14000
12000
10000
8000
6000
4000
2000
0
Mean Cost of Repair
1
2
3
4
5
6
7
Phase


x 10^4
Mean Fourth-Order Escalating Cost of Repair over Phases
10
9
8
7
6
5
4
3
2
1
0
Mean Cost of Repair
1
2
3
4
5
6
7
Phase


Combinations of Escalating Cost of Repair
Mean Cost of Repair
Phase
0
2000
4000
6000
8000
10000
12000
14000
1
2
3
4
5
6
7


Linear Escalating Cost of Repair with 0.05% Lower Bound
Mean Cost of Repair
Phase
0
50
100
150
200
250
300
350
400
450
500
1
2
3
4
5
6
7


Quadratic Escalating Cost of Repair with 0.05% Lower Bound
Mean Cost of Repair
2500
2000
1500
1000
500
0
1
2
3
4
5
6
7
Phase

Number of simulation runs = 1000000

Number of operations/interactions/LOC = 16000

Generated error rate = 0.0010

Review coverage = 0.20

Test coverage = 0.20

Cut off for introduction of errors, phase = 6, i.e., last errors are injected in Phase 5

Lower bound, the guard rail = 0.0005

Phase that reviews and tests begin = 3

Total linear escalating costs over phases = 1852

Total quadratic escalating costs over phases = 9183

Total cubic escalating costs over phases = 48651

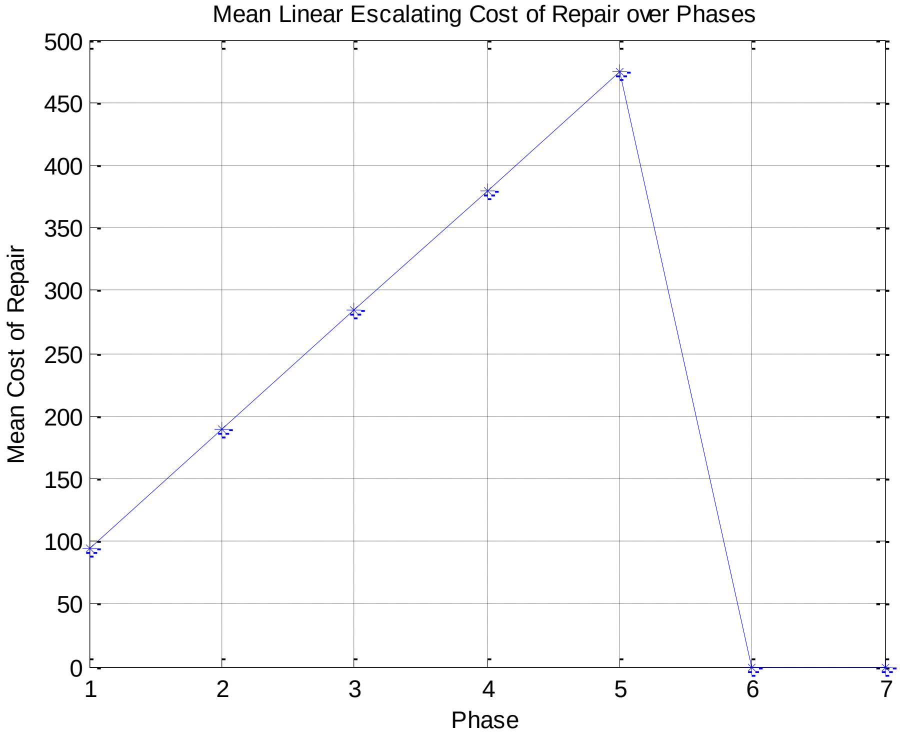
Total 4-order escalating costs over phases = 272294

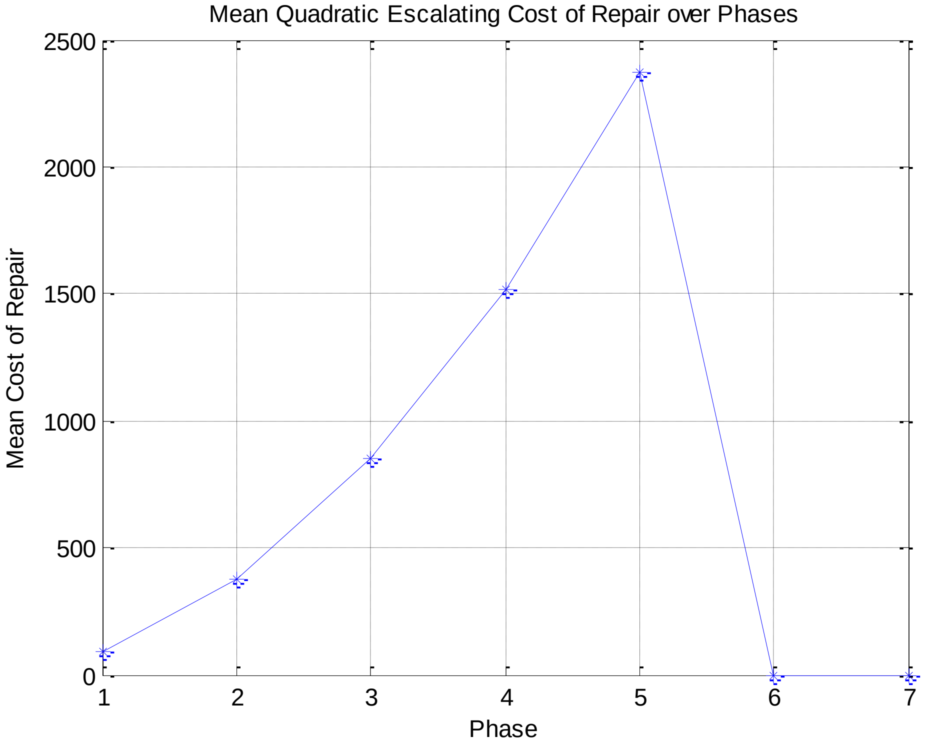
Errors in the system per phase:

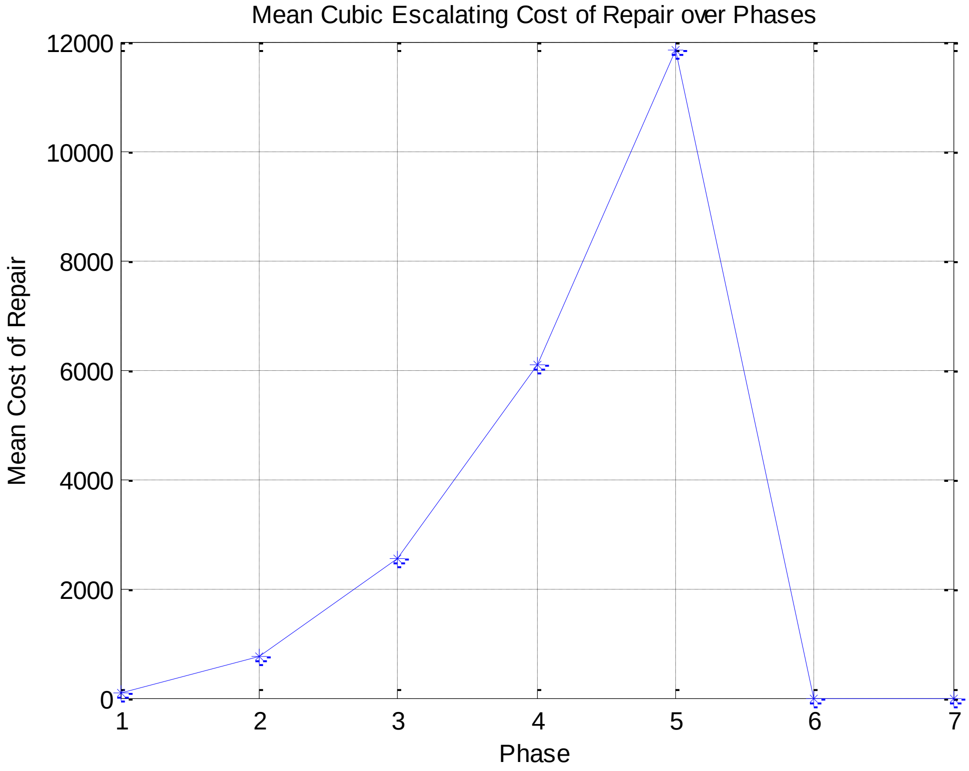
Phase	Total	Undetect	Detect
1:	16.00	16.00	0.00
2:	31.99	31.99	0.00
3:	47.96	30.69	17.27
4:	46.67	29.86	16.80
5:	45.83	29.33	16.50
6:	29.33	18.77	10.56
7:	18.77	12.01	6.76

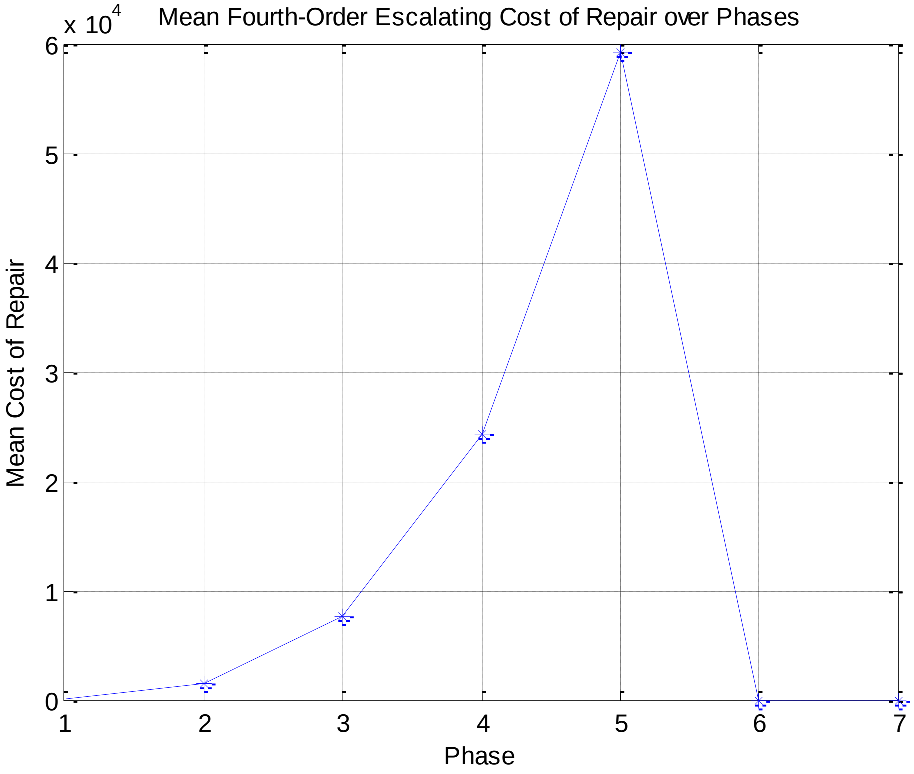
The cost function is a random function of (1/X)-1,where X is a random variable between 0 and 1; it leads to cost values between .019 and 1000. This is script “test8c.m”.

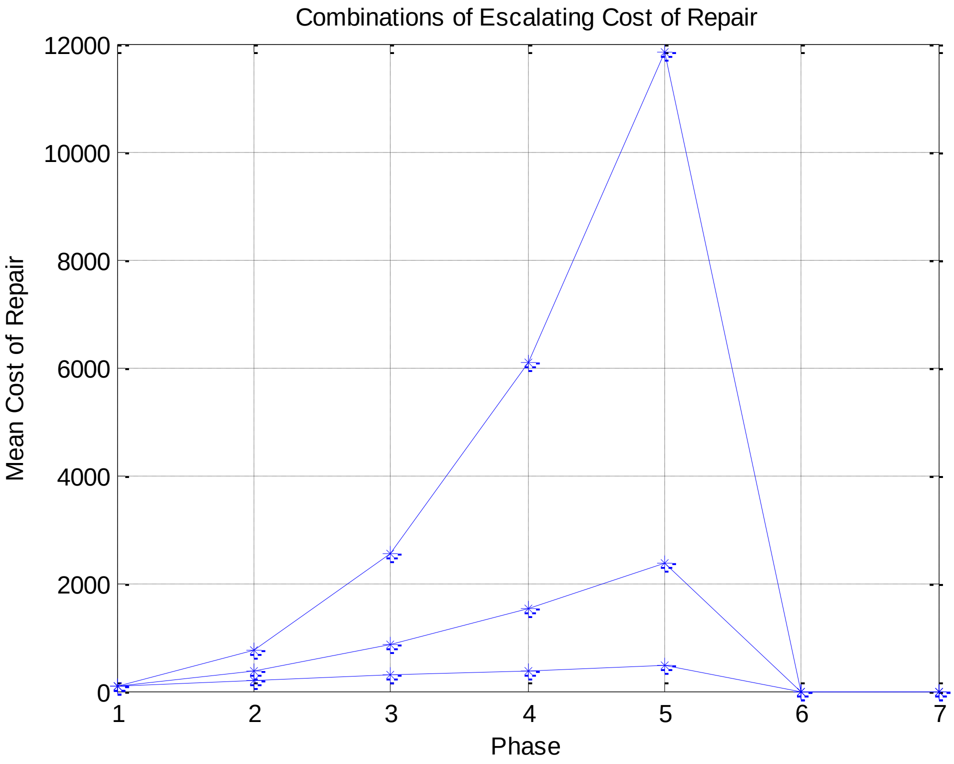
Results for Coverage of 100% by Review, 100% by Test, All Phases

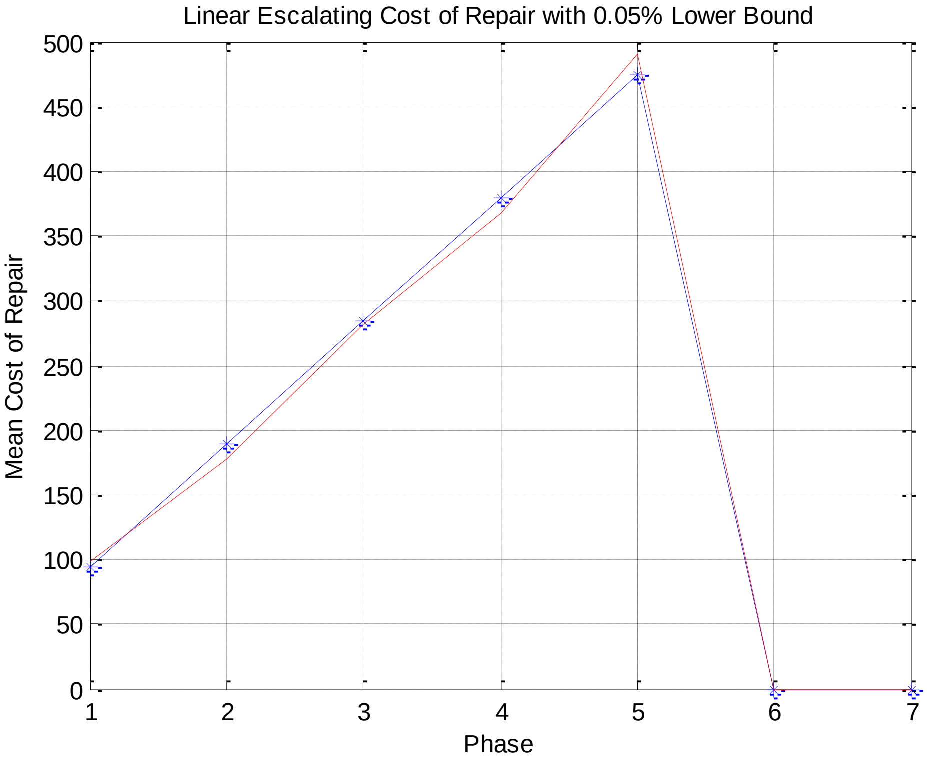


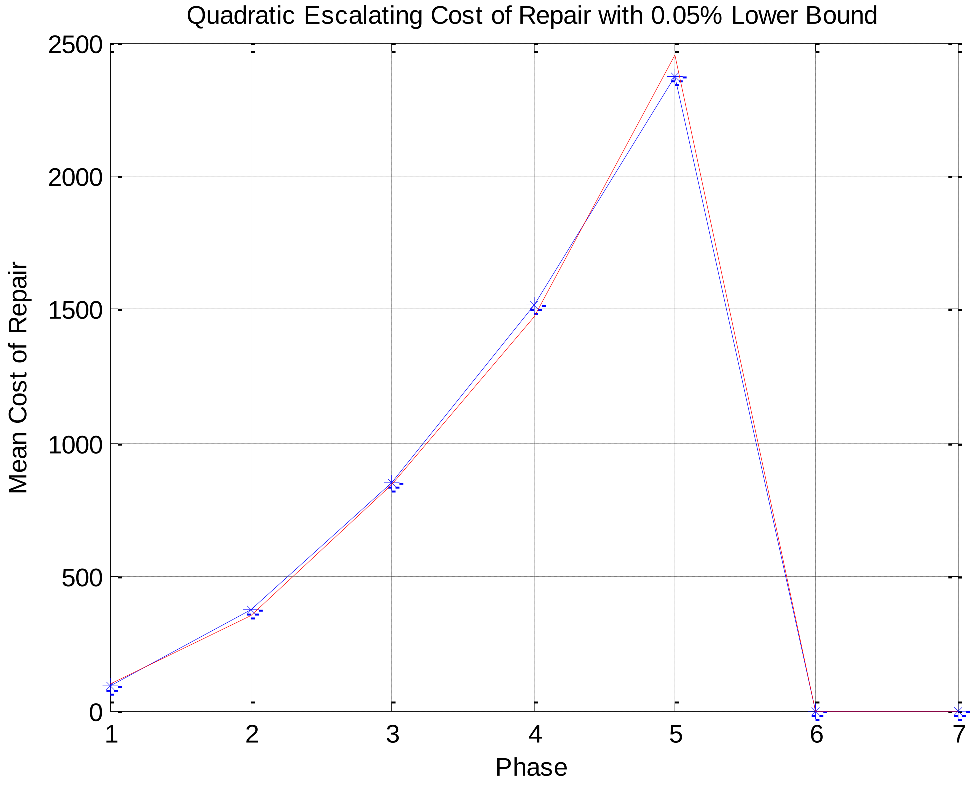



Mean Cubic Escalating Cost of Repair over Phases
Mean Cost of Repair
12000
10000
8000
6000
4000
2000
0
1
2
3
4
5
6
7
Phase


x 10^4
Mean Fourth-Order Escalating Cost of Repair over Phases
Mean Cost of Repair
6
5
4
3
2
1
0
1
2
3
4
5
6
7
Phase


Combinations of Escalating Cost of Repair
Mean Cost of Repair
Phase
0
2000
4000
6000
8000
10000
12000
1
2
3
4
5
6
7

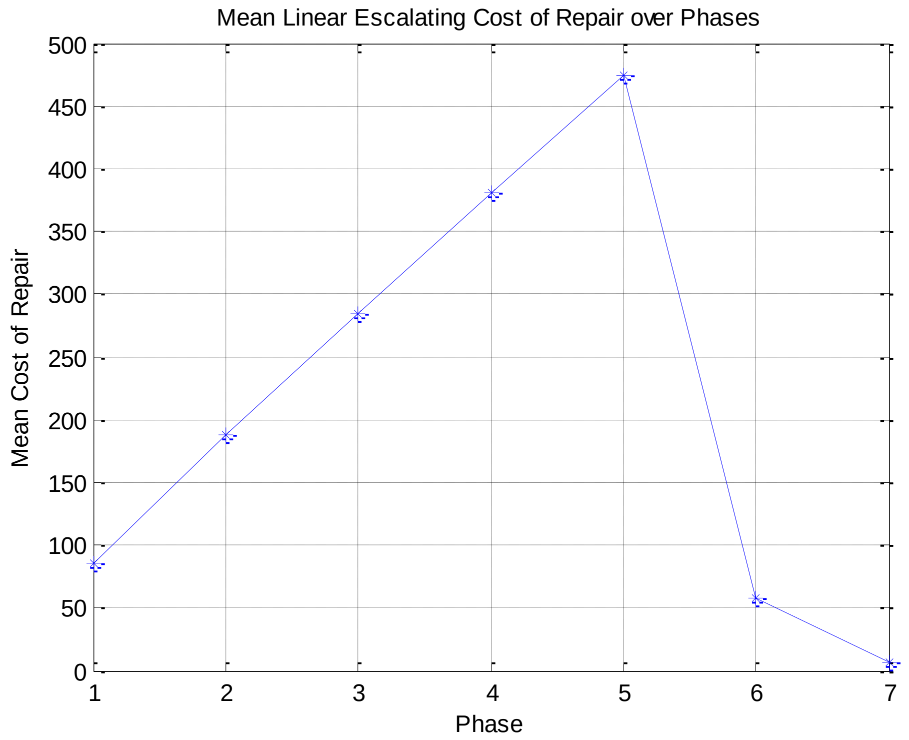

Linear Escalating Cost of Repair with 0.05% Lower Bound
Mean Cost of Repair
Phase
0
50
100
150
200
250
300
350
400
450
500
1
2
3
4
5
6
7

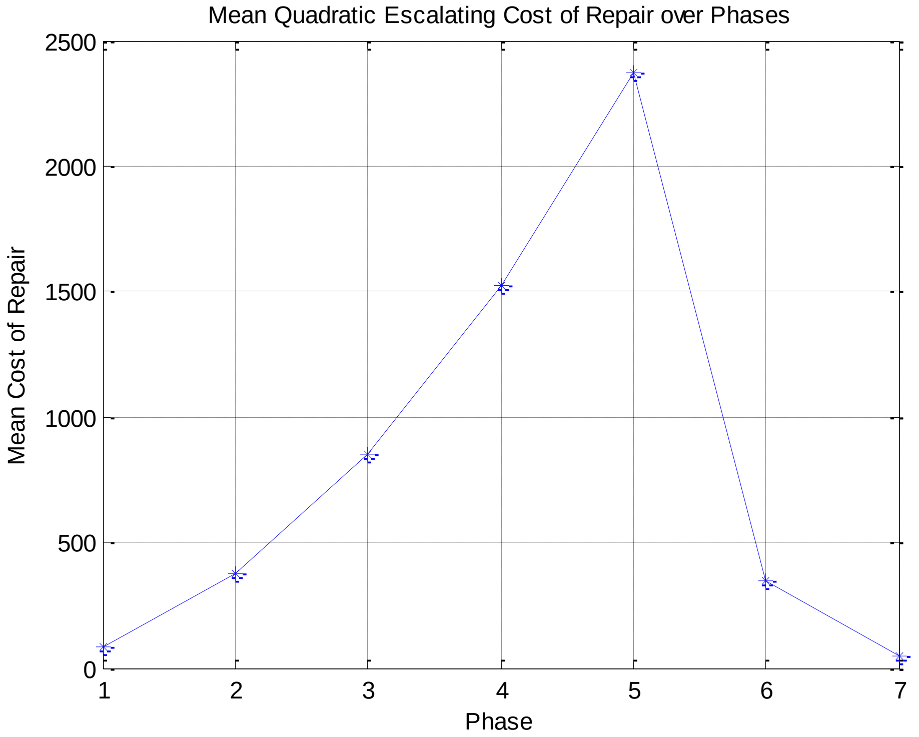

Quadratic Escalating Cost of Repair with 0.05% Lower Bound
2500
2000
1500
1000
500
0
Mean Cost of Repair
1
2
3
4
5
6
7
Phase

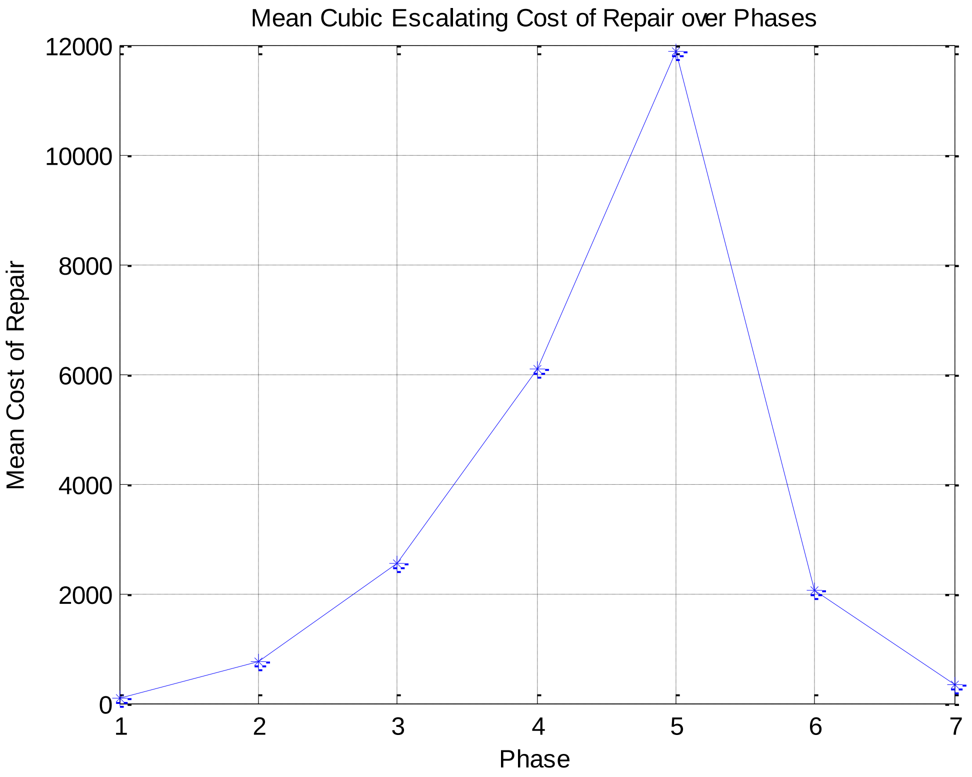
```
Number of simulation runs  = 1000000
Number of operations/interactions/LOC  = 16000
Generated error rate  = 0.0010
Review coverage  = 1.00
Test coverage  = 1.00
Cut off for introduction of errors, phase = 6, i.e., last errors are injected in Phase 5
Lower bound, the guard rail = 0.0005
Phase that reviews and tests begin = 1

Total linear escalating costs over phases = 1422
Total quadratic escalating costs over phases = 5215
Total cubic escalating costs over phases = 21334
Total 4-order escalating costs over phases = 92825

Errors in the system per phase:
Phase    Total   Undetect   Detect
 1:      16.00      0.00     16.00
 2:      16.00      0.00     16.00
 3:      16.00      0.00     16.00
 4:      16.00      0.00     16.00
 5:      16.00      0.00     16.00
 6:       0.00      0.00      0.00
 7:       0.00      0.00      0.00
```

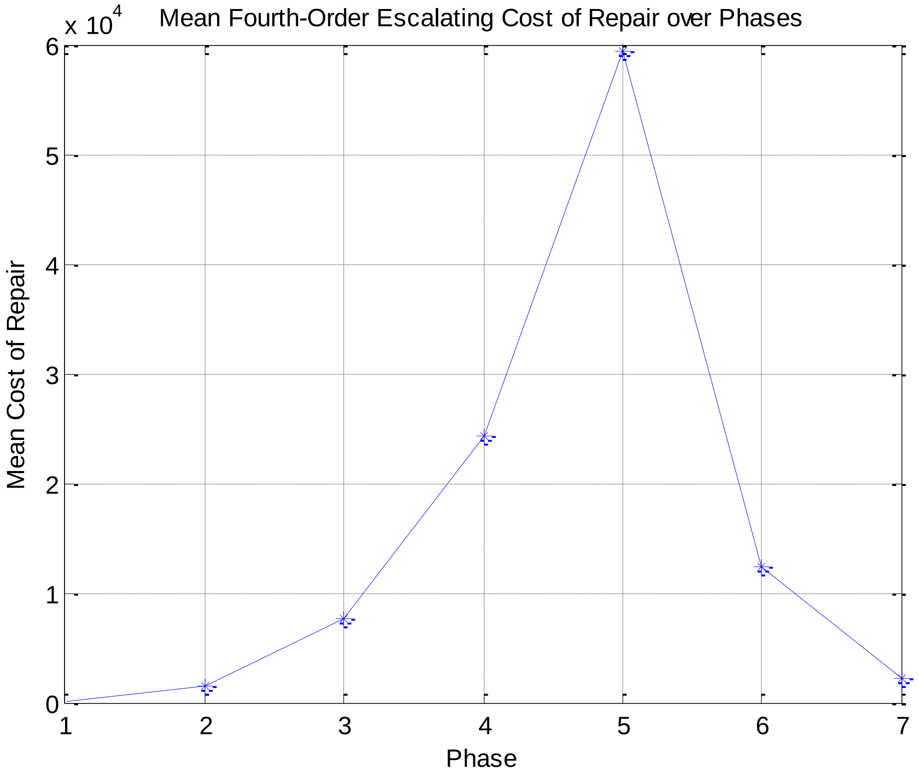
The cost function is a random function of (1/X)-1,where X is a random variable between 0 and 1; it leads to cost values between .019 and 1000. This is script “test8c.m”.

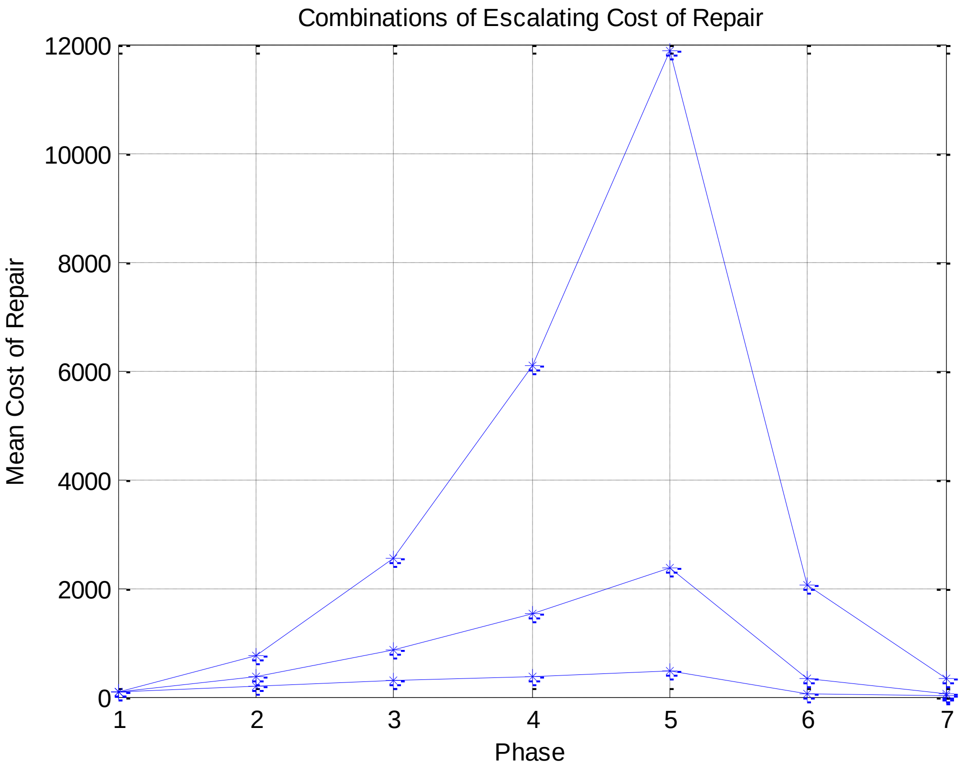
Results for Coverage of 80% by Review, 50% by Test, All Phases

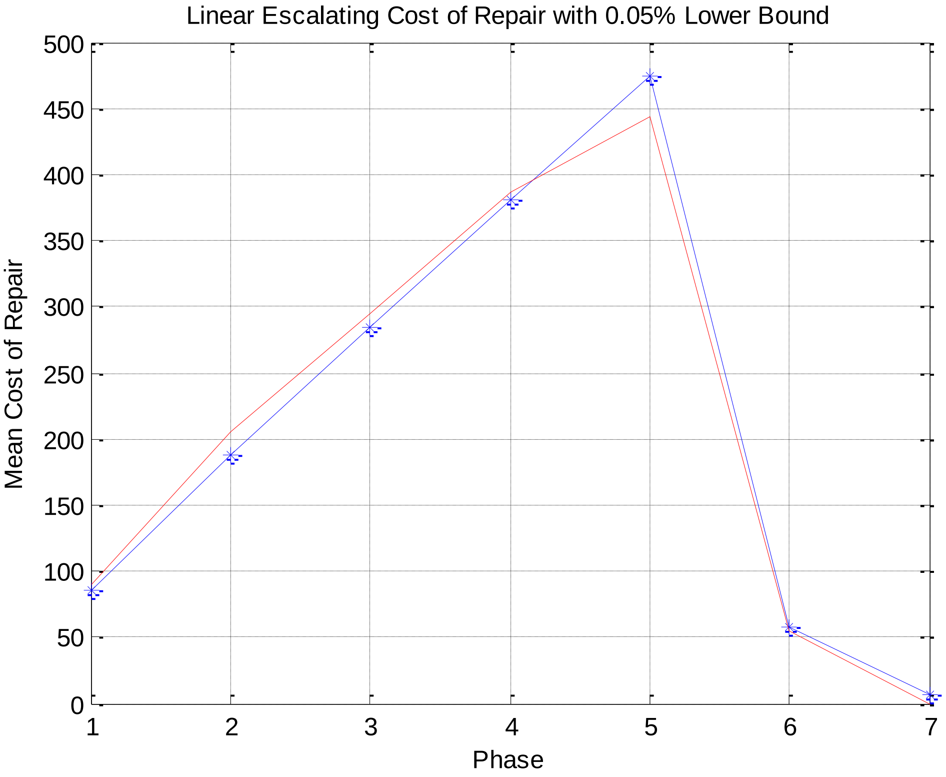


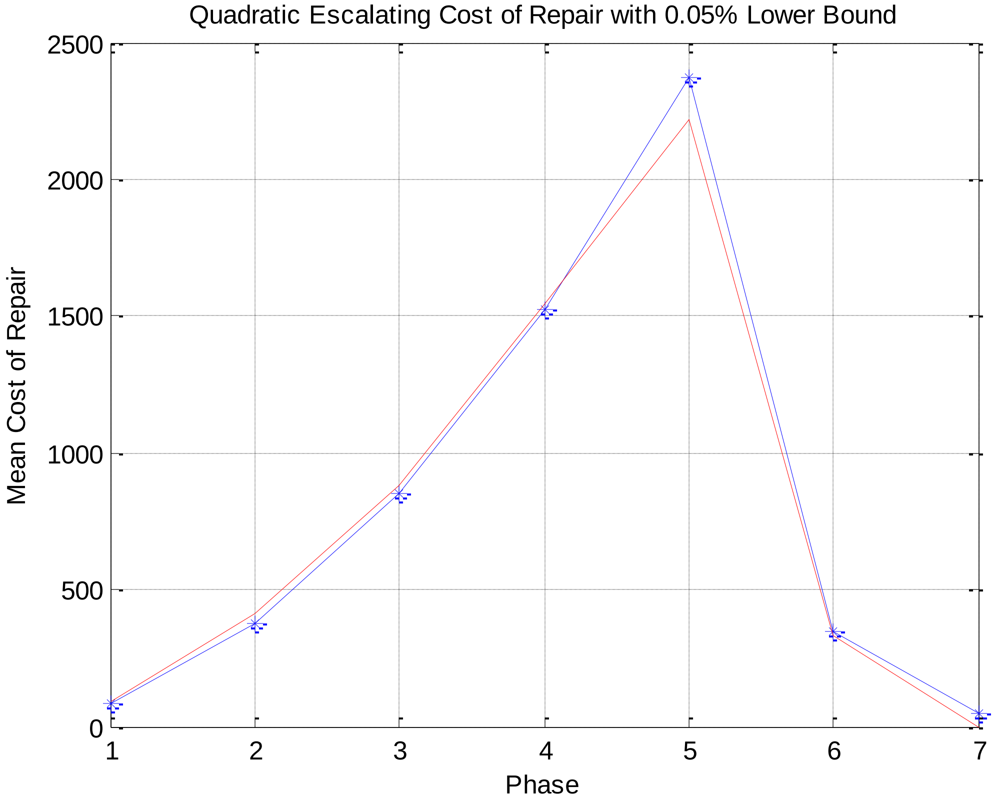



Mean Cubic Escalating Cost of Repair over Phases
Mean Cost of Repair
12000
10000
8000
6000
4000
2000
0
1
2
3
4
5
6
7
Phase


x 10^4
Mean Fourth-Order Escalating Cost of Repair over Phases
Mean Cost of Repair
6
5
4
3
2
1
0
1
2
3
4
5
6
7
Phase


Combinations of Escalating Cost of Repair
Mean Cost of Repair
Phase
0
2000
4000
6000
8000
10000
12000
1
2
3
4
5
6
7


Linear Escalating Cost of Repair with 0.05% Lower Bound
Mean Cost of Repair
Phase
0
50
100
150
200
250
300
350
400
450
500
1
2
3
4
5
6
7


Quadratic Escalating Cost of Repair with 0.05% Lower Bound
Mean Cost of Repair
Phase
0
500
1000
1500
2000
2500
1
2
3
4
5
6
7

Number of simulation runs = 1000000

Number of operations/interactions/LOC = 16000

Generated error rate = 0.0010

Review coverage = 0.80

Test coverage = 0.50

Cut off for introduction of errors, phase = 6, i.e., last errors are injected in Phase 5

Lower bound, the guard rail = 0.0005

Phase that reviews and tests begin = 1

Total linear escalating costs over phases = 1477

Total quadratic escalating costs over phases = 5599

Total cubic escalating costs over phases = 23733

Total 4-order escalating costs over phases = 107577

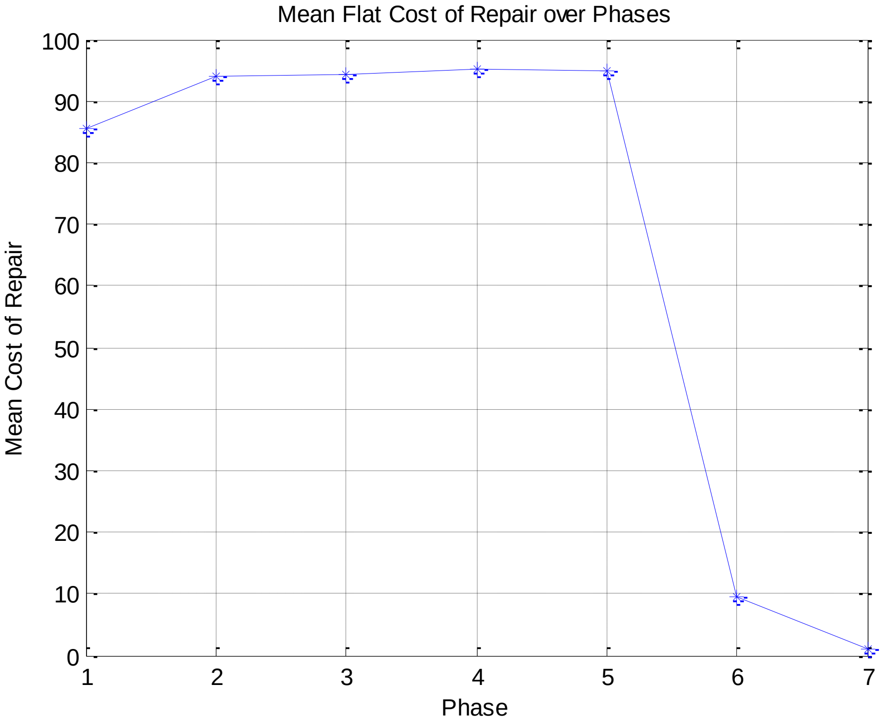
Errors in the system per phase:

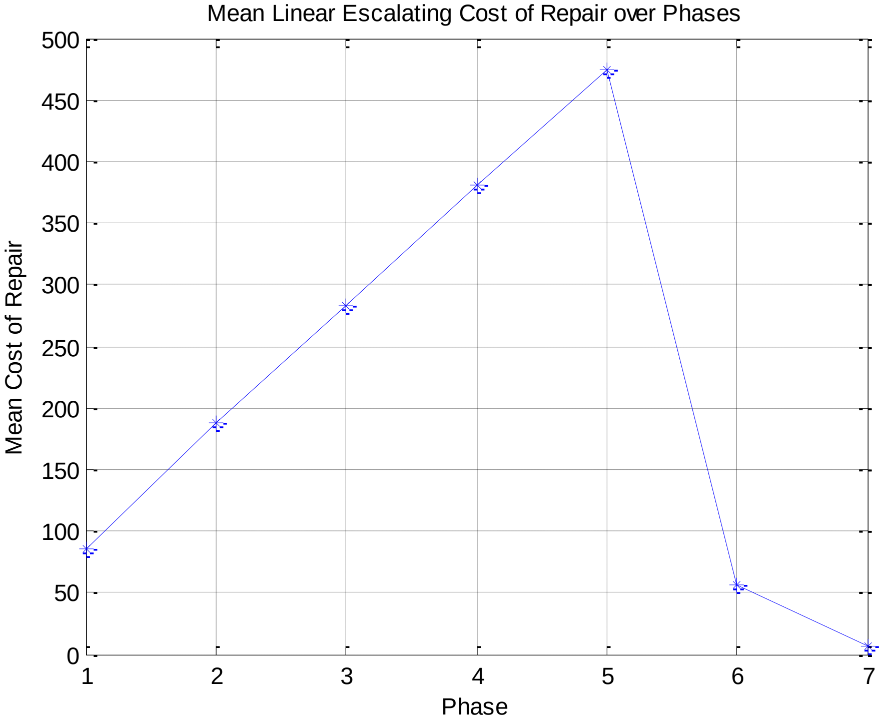
Phase	Total	Undetect	Detect
1:	16.00	1.60	14.40
2:	17.60	1.76	15.84
3:	17.76	1.77	15.99
4:	17.77	1.78	15.99
5:	17.77	1.78	15.99
6:	1.78	0.18	1.60
7:	0.18	0.02	0.16

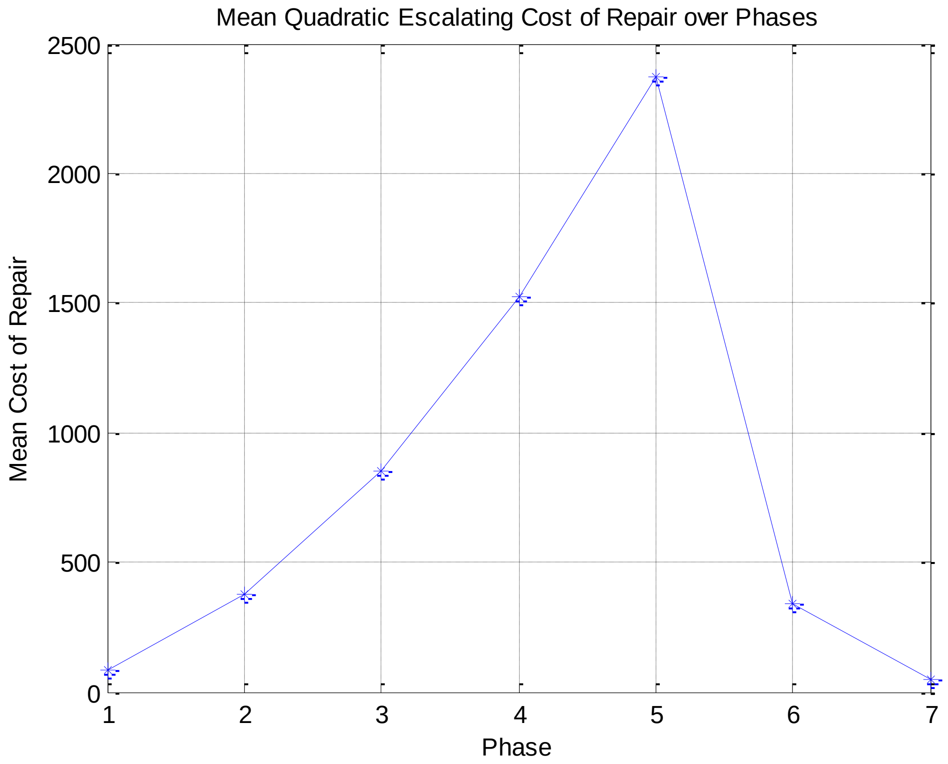
The cost function is a random function of (1/X)-1,where X is a random variable between 0 and 1; it leads to cost values between 0.019 and 1000. This is script “test8c.m”.

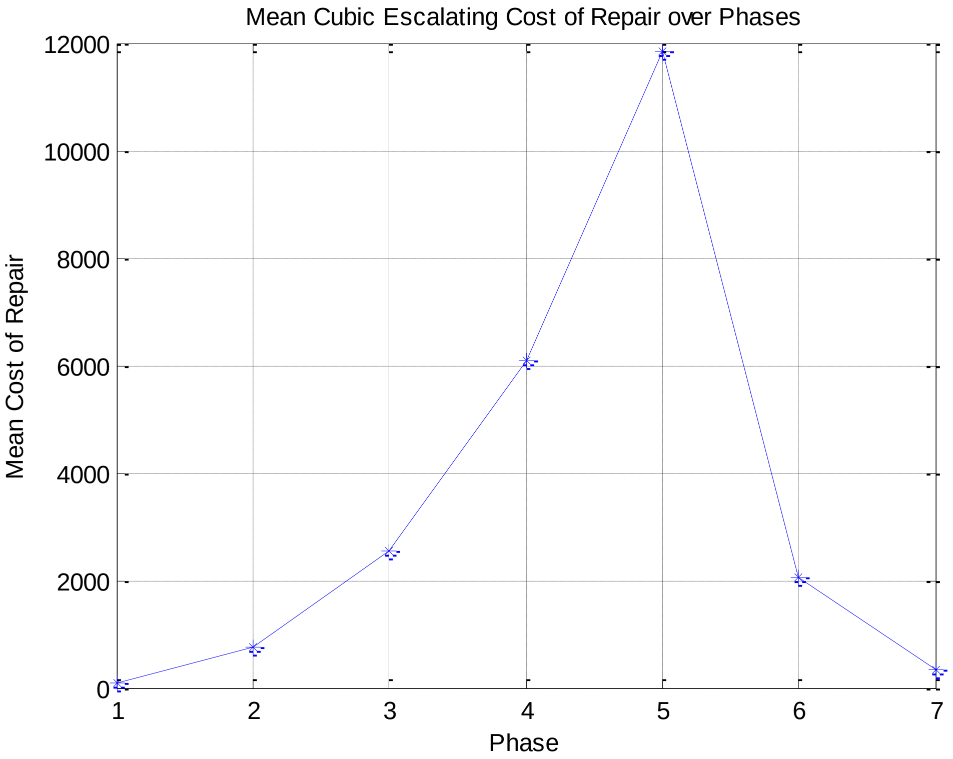
Results for Coverage of 80% by Review, 50% by Test, All Phases

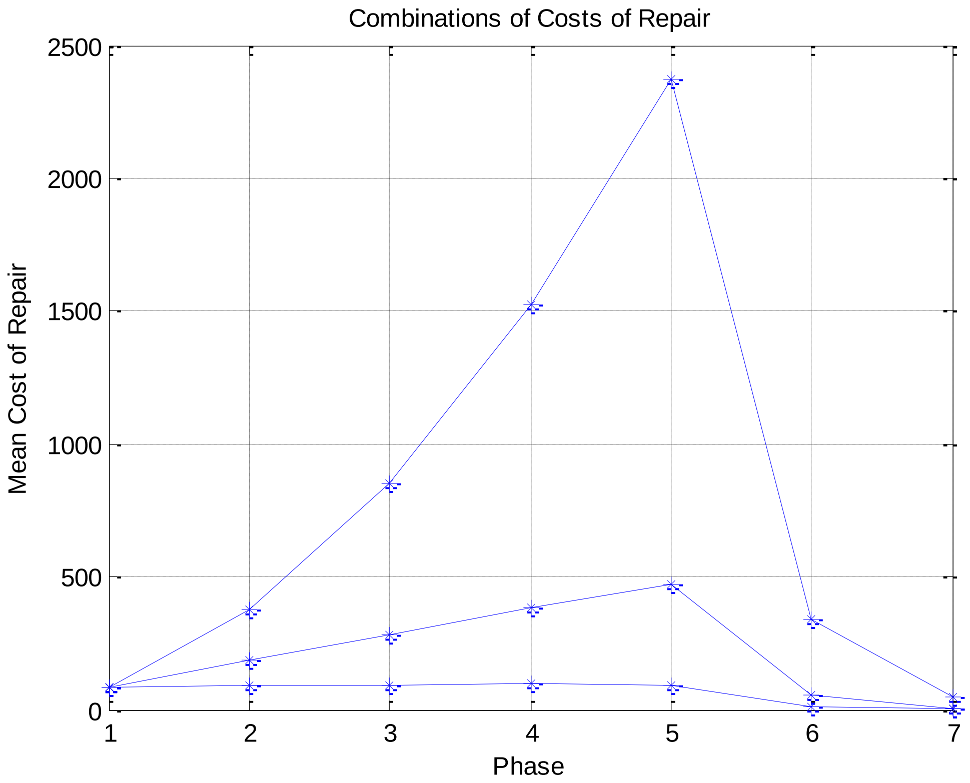
Flat Costs Thru Cubic (No 4th-Order) Escalating

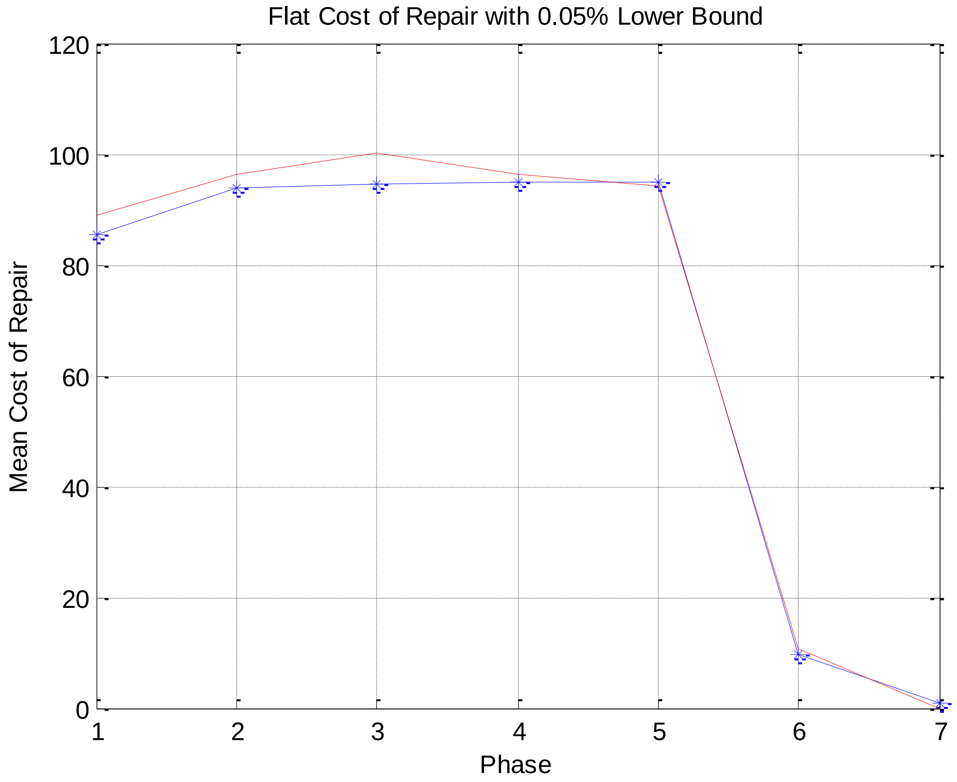


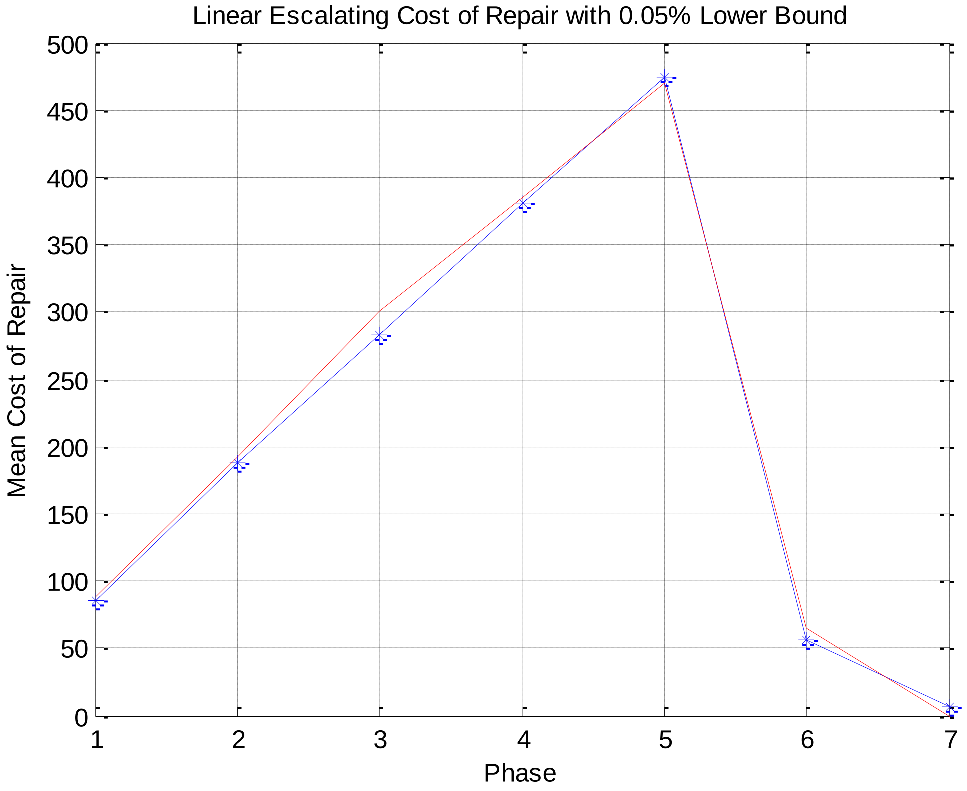



Mean Quadratic Escalating Cost of Repair over Phases
Mean Cost of Repair
0
500
1000
1500
2000
2500
1
2
3
4
5
6
7
Phase


Mean Cubic Escalating Cost of Repair over Phases
Mean Cost of Repair
0
2000
4000
6000
8000
10000
12000
1
2
3
4
5
6
7
Phase


Combinations of Costs of Repair
Mean Cost of Repair
Phase
0
500
1000
1500
2000
2500
1
2
3
4
5
6
7


Flat Cost of Repair with 0.05% Lower Bound
Mean Cost of Repair
Phase
0
20
40
60
80
100
120
1
2
3
4
5
6
7


Linear Escalating Cost of Repair with 0.05% Lower Bound
Mean Cost of Repair
500
450
400
350
300
250
200
150
100
50
0
1
2
3
4
5
6
7
Phase

Number of simulation runs = 1000000

Number of operations/interactions/LOC = 16000

Generated error rate = 0.0010

Review coverage = 0.80

Test coverage = 0.50

Cut off for introduction of errors, phase = 6, i.e., last errors are injected in Phase 5

Lower bound, the guard rail = 0.0005

Phase that reviews and tests begin = 1

Total flat costs over phases = 474

Total linear escalating costs over phases = 1475

Total quadratic escalating costs over phases = 5591

Total cubic escalating costs over phases = 23702

Errors in the system per phase:

Phase	Total	Undetect	Detect
1:	16.00	1.60	14.40
2:	17.60	1.76	15.84
3:	17.76	1.78	15.98
4:	17.77	1.78	15.99
5:	17.77	1.77	16.00
6:	1.77	0.18	1.60
7:	0.18	0.02	0.16

The cost function is a random function of (1/X)-1,where X is a random variable between 0 and 1; it leads to cost values between 0.019 and 1000. This is script “test8d.m”.

Appendix C - Potential Plans and Checklists

Background Information

Checklists should be tailored for the market and environment. These examples only focus on embedded systems and not entire vehicles, such as an aircraft, automobile, industrial plant, or military vehicle, in their markets. Each of these checklists is only a starting point; they should be revised and tailored for a specific product.

The three following sections contain general outlines of potential checklists:

- A general outline is given here that nearly all projects should follow to some degree.
- A general listing of documents that many projects should prepare.
- Listings of potential standard organizations that may affect specific markets.

Some checklists for specific markets then make up the remaining sections of this appendix.

General Outline of Activities within a Project

- Project Tracking – Engineering Concerns
  - Visibility
  - Accountability
  - Configuration Management
  - Version Control
  - Database Implementation
  - Engineering Change Notice
  - Feedback
  - Improvement
  - Closeout
  - Archiving
  - Outsourcing
  - Qualification of Vendors
  - Templates – plans, documents, reports, action item, design documents, test results
- General Engineering Processes
  - Power and power flows
  - Material flows

  - Requirements
    - Input – customer, company, standards
    - Analysis – business feasibility
    - Producing Metrics
    - Modification and Change
    - Compliance Matrix
  - Analysis
    - Thermal
    - Worst case stress
    - Shock and vibration
    - Humidity
    - Radiation
    - Corrosion
    - Pressure
    - Outgassing
  - Design
  - Reviews
    - Peer Review
    - Control Board Review
    - Design Review – CoDR, PDR, CDR, PRR, etc.
  - Development
  - Test
  - Integration
  - Acceptance
  - Delivery
- Systems engineering
  - Standards
  - Concept of operations (CONOPS)
  - Tools
  - Tool Certification
  - Design

        - Schematics
        - Mechanical Layout
    - Analysis
        - Event Tree Analysis (ETA)
        - Failure Mode Effects Analysis (FMEA)
        - Fault Tree Analysis (FTA)
        - System Theoretic Process Analysis (STPA)
        - Probabilistic Risk Assessment (PRA)
        - Safety and safety instrumented levels (SIL)
        - Robust dependability – reliability, availability
        - Radiation
        - Operations research
        - Interface Inspections
        - Integration Coverage – functional, physical, environmental
- Electronic and electrical engineering
    - Tools
    - Tool Certification
    - Design
        - Schematics
        - Circuit board layout
        - Chassis layout
        - Wire harness and cabling
        - Assembly
    - Analysis
        - Signal integrity
        - Electromagnetic Compatibility (EMC)
        - Power consumption
        - Thermal
        - Worst case stress
        - Shock and vibration
        - Humidity

        - Radiation
        - Corrosion
        - Pressure
        - Outgassing
- Software engineering
    - Tools
    - Tool certification
    - Style guide
    - Source listing
    - Compilation and transfer to hardware
- Mechanical engineering
    - Tools
    - Tool Certification
    - Design
        - Schematics
        - Mechanical layout
        - Assembly
    - Analysis
        - Mass, Size and Volume
        - Materials
        - Structural model
        - Thermal
        - Worst case stress
        - Shock and vibration
        - Humidity
        - Radiation
        - Corrosion
        - Pressure
        - Outgassing
- Specialty engineering and services (per market)
    - Aerospace and space sciences

    - Optics
    - Acoustics, ultrasound
    - RF, radar
    - Communications, networks
    - Materials
    - Chemical
    - Transportation
    - Food
    - Mining
    - Ocean and marine
    - Pharmaceutical
    - Medical and bioengineering
    - Operations research
    - Safety
    - Life support
    - Ergonomics, human interface, industrial psychology
- Test and integration
    - bench tests
    - mockup and fit checks
    - unit and module tests
    - fault insertion tests
    - verification and validation
    - integration
    - alignment checks and calibration
    - field tests
    - environmental tests
    - security tests
    - stress test and highly accelerated life test (HALT)
    - commissioning
- Compliance (per industry and market)
- Manufacturing

  - Volume
  - Specialty operations
  - In-house
  - Contract
  - Test
- Logistics and distribution
- Metrology
  - Qualification of Vendors
  - Calibration
  - Certification of Measurements
  - Archiving
- Sales, marketing, and service support
  - Warranties
  - Installation
  - Repair
  - Maintenance
  - Help desk
  - Training
- Disposal

Potential Documents Generated within a Project

There are many potential documents that a project can generate, and they can form a picture of the design and development. No one set of documents fit all projects, but a good checklist of documents can help. An example of two general sets of documents may be found in reference [255].

Documents take many different forms. There can be business-related documents, company general procedures, project-specific business, and technical documents. Examples of business-related procedures and marketing may be catalogs, brochures, and schedule-budget plans. Examples of company technical procedures and templates may be quality management system, infrastructure plan, vendor qualification template, and manufacturing/production template. Examples of documents specific to a customer or project may be business feasibility, work break-down structure, state of work, and product acceptance plan. Examples of project-specific technical documents are the electrical design description, software design description, CONOPs, and logistics and delivery plan.

Documents are important to design assurance. They prepare the groundwork for design and development. They record and archive what was done. They can, and should, provide rationale for what was done. They serve as the basis for understanding the process of design and development.

Standards Organizations

- International Organization for Standardization (ISO) (http://www.iso.ch).
- International Electrotechnical Commission (IEC) (http://www.iec.ch). The electrical/electronic section of ISO.
- US National Institute of Standards and Technology (NIST) (http://www.nist.org). Co-ordinates the production of American standards.
- European Committee for Standardisation (CEN). European standards are prefixed EN ("Euro Norm").
- European Committee for Electrotechnical Standardization (CENELEC).
- European Commission (EC). The EC issues directives related to compliance with standards.
- Underwriters Laboratory (UL) (https://ulstandards.ul.com/).
- British Standards Institution (BSI) (http://www.bsi.org.uk). The BSI co-ordinates the production of UK standards. British standards are prefixed BS.
- The UK Accreditation Service (UKAS) (http://www.ukas.com) is responsible for the accreditation of laboratories offering measurement and calibration services. UKAS also provides accreditation for ISO9000 certification organisations.
- German Association for Electronics and Information Technology (VDE) (http://www.vde.de). VDE produces standards for electronics and IT, provides accreditation services, and operates test laboratories. The VDE mark is affixed to equipment which complies with relevant VDE standards.

Industry-Based Standards Organizations

Mechanical, Systems

- American Society of Mechanical Engineers (ASME) (http://www.asme.com)
- American Society of Testing and Materials (ASTM) (http:www.astm.com)

Electrical, Electronic

- Institute of Electrical and Electronic Engineers (IEEE) (http://www.ieee.org).
- Telcordia Technologies (was Bellcore). Telcordia standards are applied in the telecommunications industries worldwide. http://www.telcordia.com

Military Standards Organizations

- US Department of Defense - Military standards, handbooks and specifications (MIL-STDs, MIL-HDBKs, MIL-SPECs).
- North Atlantic Treaty Organization - NATO Standards (STANAGS).
- UK Ministry of Defence - Defence Standards (DEFSTANs).

Aviation, Aerospace Standards Organizations

- International Civil Aviation Organization (ICAO).
- Air Radio Inc. (ARINC). Produces standards for aviation electronic systems (avionics).
- National Astronautics and Space Administration (NASA).
- Federal Aviation Administration (FAA) (USA).
- Civil Aviation Administration (CAA) (UK).
- European Space Agency (ESA).

Best Practices for Spacecraft

Each spacecraft is a custom development, regardless, there are some common procedures and activities associated with each. An example of best practices for spacecraft may be found in reference [189].

Satellite Instrument

Checklists for designing and developing a satellite instrument should be tailored for the environment, reliability, and autonomy. Instruments must endure the high-vibration of launch, the very cold when facing away from the sun, the very hot when in the sunlight, no pressure, and radiation. Reliability and autonomy are extremely important; the instrument must work without most types of invention; repair is certainly not an option, except maybe on the International Space Station.

Mechanically and electrically, the instrument must be small, low-mass, and low-power consumption. Often the power supply derives from solar panels and batteries, which power other parts of the spacecraft, too.

The entire set of documents in reference [255] are appropriate for most satellite instruments. The parts of best practices in reference [189] are also appropriate for most satellite instruments.

Medical Device

Checklists for designing and developing a medical device should be tailored for safety and efficacy. Dependability, reliability, and robustness are extremely important; the device must operate in its designed environment without failure. External (to the human body) support equipment often has an expected lifetime of more than 10 years and sometimes beyond 20 years. Implanted devices often must operate without intervention for two to 10 years.

The human interface needs careful consideration in its design. The patients and medical staff using the device must understand its basic purpose and function and how to operate it.

Mechanically and electrically, the instrument must be safe. It must not harm the patient. Electrical isolation is very important to prevent potentially dangerous leakage currents. It needs appropriate mechanical attachment to avoid falling on or hitting a patient or other equipment. Specialty compliance testing, such as full EMC tests, are often a part of the FDA approval.

The device must undergo clinical trials, which often take two years or longer, to demonstrate safety and efficacy. Clinical specialties are needed to run the studies. You must work closely with the FDA to run your studies, submit the results, and gain approval. These studies take time and effort.

Military Equipment

Checklists for designing and developing military equipment should be tailored for dependability, environment, and longevity. This type of equipment must operate dependably in extreme environments, such as artic cold or desert heat with salt spray or sand blowing on it.

Since longevity can be 20 to 50 years or even longer, the equipment often needs an architecture that can be repaired and maintained.

Like a medical device, the human interface needs careful consideration in its design. The military personnel using the device must understand its basic purpose and function and how to operate it. They are not always kind or gentle in handling equipment, either.

Military equipment must undergo compliance testing for EMC and environmental assault. Military equipment typically undergoes years of trials to demonstrate utility and operational effectiveness. You must work closely with the government customer to run the evaluations, submit the results, and gain approval. These studies take time and effort; fighter aircraft, for instance, can take decades of evaluations before inclusion in the military inventory.

Industrial Equipment

Checklists for designing and developing industrial equipment are similar to military equipment. Compliance testing and certification are specific to the industry.

Consumer Appliance

Checklists for designing and developing consumer appliances should be tailored for low cost, customer acceptance, human operation, safety, and, to some extent, reliability. Appliances need to be low-cost as most have very low profit margins, hence, quantity sales are important to make enough money for the company. Part of making the sales is that the appliance attracts customers, which often means that the human interface and operation are very important and require proportionately more time in design and development.

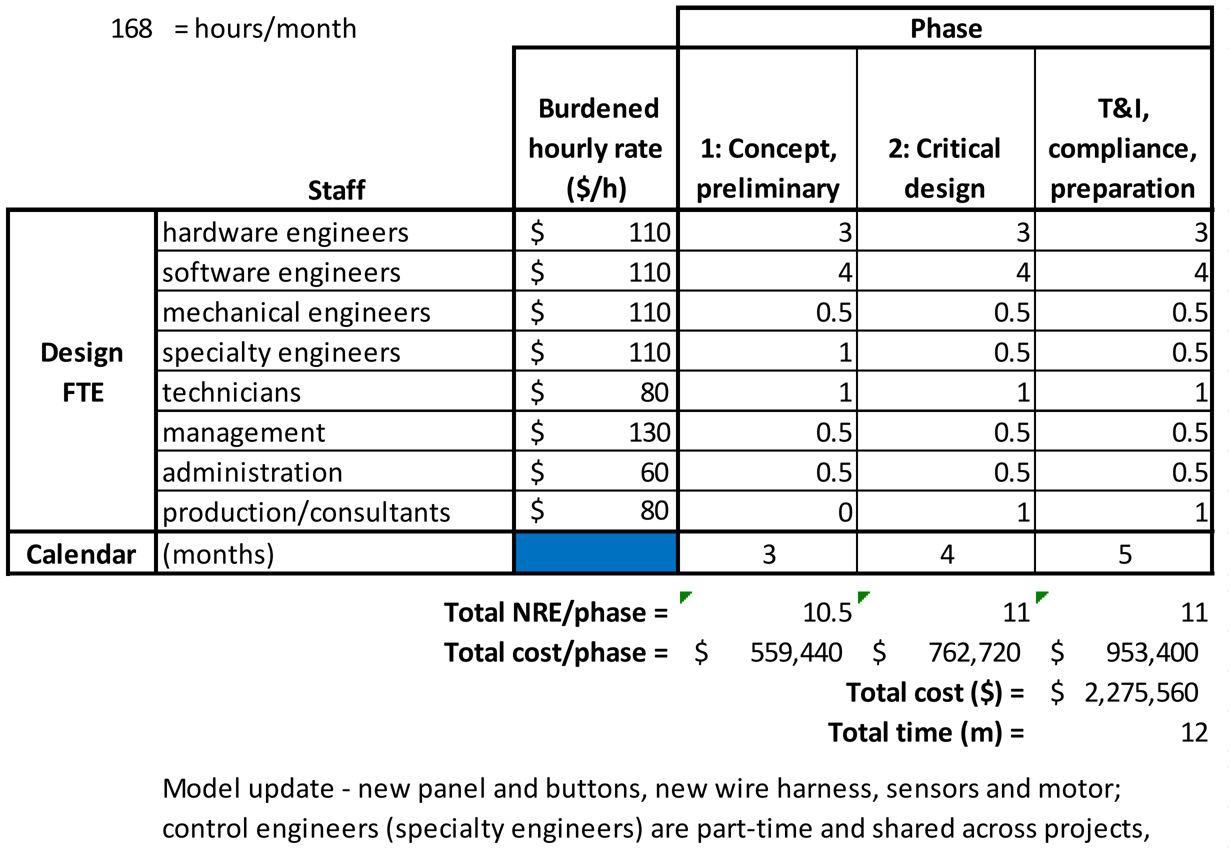
Appliances generally are not repaired nor maintained. Consequently, a dependable life span of operation is important. Smaller appliances, such as coffee makers, may not have the expected lifetime as larger appliances, such as refrigerators.

Consumer appliances must undergo compliance testing for safety, particularly electrical shock and fire, to receive UL approval or the CE mark. You must work closely with the certifying body to run the evaluations, submit the results, and gain approval. These studies take time and effort.

Appendix D - Case Studies for Effort and Cost

The case studies listed here are from industry sources [200]. The literature does not give loaded, or burdened, average salaries. These case studies use salary values that may be found in industry now; this is not a critical factor as these are spreadsheets and the loaded, average salaries may by easily changed. These case studies could be called by the framework should a user request to see them, by clicking the button in Figure 3.2 that requests the case studies.

Clothes Washer – One-Year Development

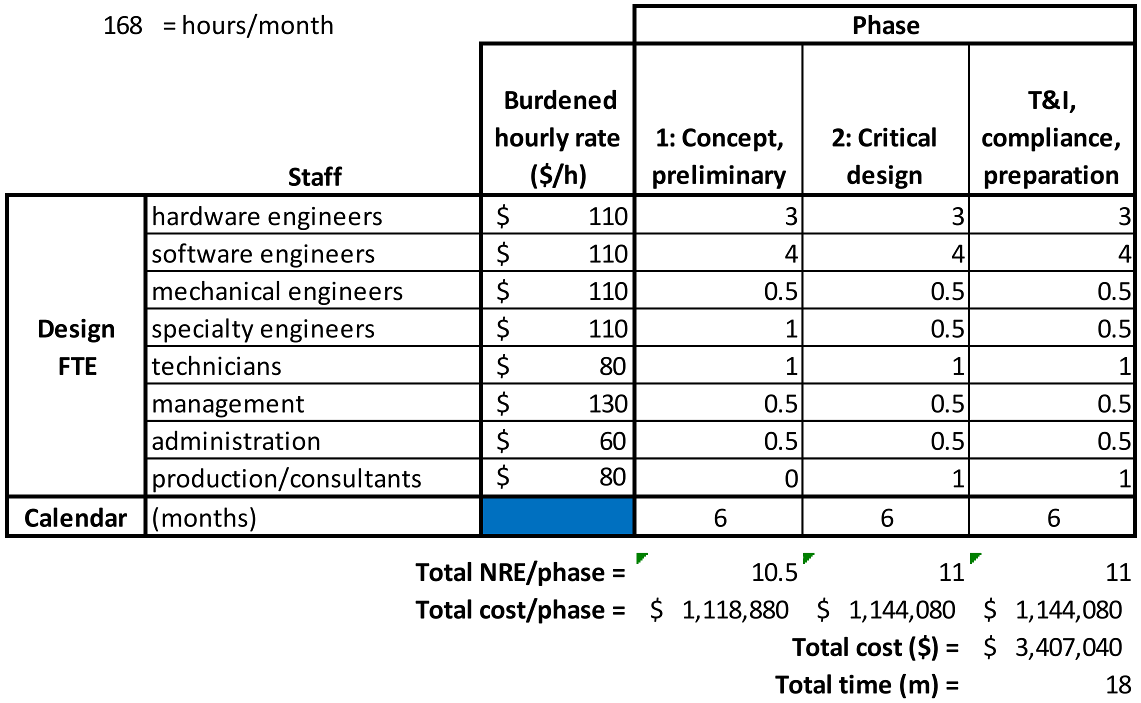


168 = hours/month

			Phase		
	Staff	**Burdened hourly rate ($/h)**	**1: Concept, preliminary**	**2: Critical design**	**T&I, compliance, preparation**
Design FTE	hardware engineers	$ 110	3	3	3
	software engineers	$ 110	4	4	4
	mechanical engineers	$ 110	0.5	0.5	0.5
	specialty engineers	$ 110	1	0.5	0.5
	technicians	$ 80	1	1	1
	management	$ 130	0.5	0.5	0.5
	administration	$ 60	0.5	0.5	0.5
	production/consultants	$ 80	0	1	1
Calendar	(months)	🟦	3	4	5
		Total NRE/phase =	10.5	11	11
		Total cost/phase =	$ 559,440	$ 762,720	$ 953,400
				Total cost ($) =	$ 2,275,560
				Total time (m) =	12

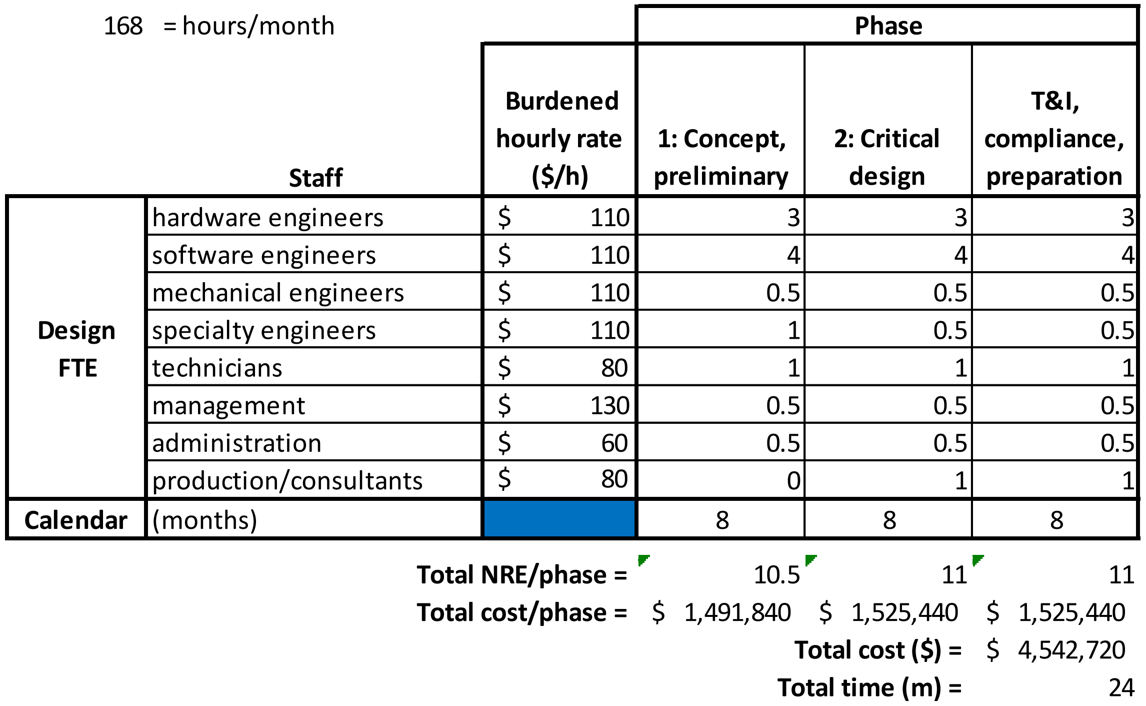
Model update - new panel and buttons, new wire harness, sensors and motor; control engineers (specialty engineers) are part-time and shared across projects,

Clothes Washer – One-and-a-Half Year Development



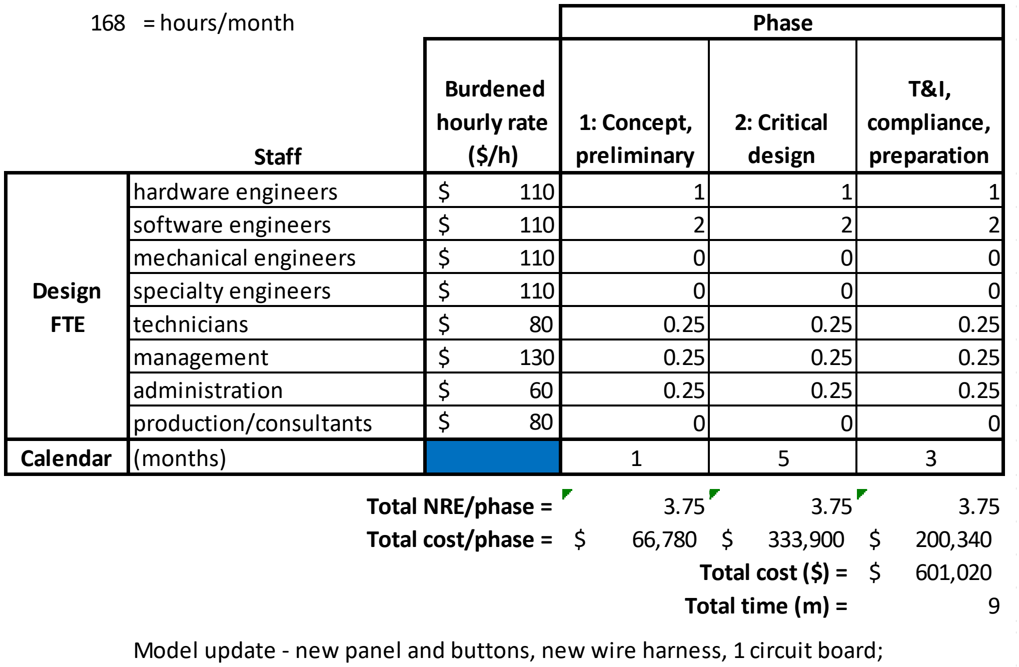
168	= hours/month		Phase		
	Staff	Burdened hourly rate ($/h)	1: Concept, preliminary	2: Critical design	T&I, compliance, preparation
Design FTE	hardware engineers	$ 110	3	3	3
	software engineers	$ 110	4	4	4
	mechanical engineers	$ 110	0.5	0.5	0.5
	specialty engineers	$ 110	1	0.5	0.5
	technicians	$ 80	1	1	1
	management	$ 130	0.5	0.5	0.5
	administration	$ 60	0.5	0.5	0.5
	production/consultants	$ 80	0	1	1
Calendar	(months)		6	6	6
		Total NRE/phase =	10.5	11	11
		Total cost/phase =	$ 1,118,880	$ 1,144,080	$ 1,144,080
				Total cost ($) =	$ 3,407,040
				Total time (m) =	18

Clothes Washer – Two-Year Development



168	= hours/month		Phase		
	Staff	Burdened hourly rate ($/h)	1: Concept, preliminary	2: Critical design	T&I, compliance, preparation
Design FTE	hardware engineers	$ 110	3	3	3
	software engineers	$ 110	4	4	4
	mechanical engineers	$ 110	0.5	0.5	0.5
	specialty engineers	$ 110	1	0.5	0.5
	technicians	$ 80	1	1	1
	management	$ 130	0.5	0.5	0.5
	administration	$ 60	0.5	0.5	0.5
	production/consultants	$ 80	0	1	1
Calendar	(months)		8	8	8
		Total NRE/phase =	10.5	11	11
		Total cost/phase =	$ 1,491,840	$ 1,525,440	$ 1,525,440
				Total cost ($) =	$ 4,542,720
				Total time (m) =	24

Office Data Telecom Development



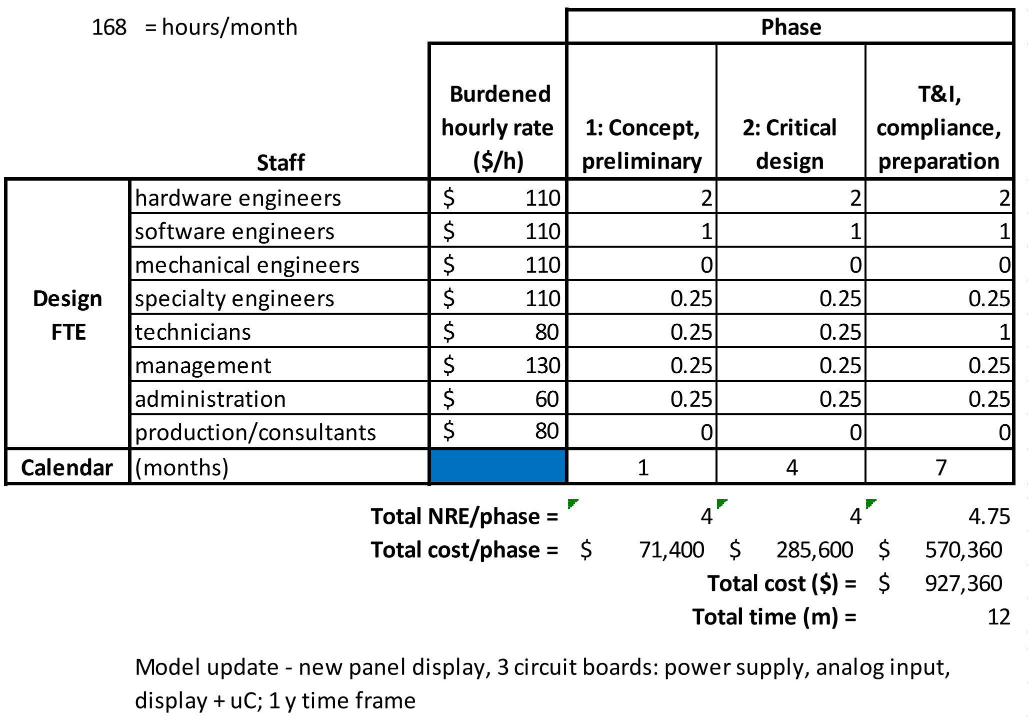
168 = hours/month

	Staff	Burdened hourly rate ($/h)	Phase 1: Concept, preliminary	 2: Critical design	 T&I, compliance, preparation
Design FTE	hardware engineers	$ 110	1	1	1
	software engineers	$ 110	2	2	2
	mechanical engineers	$ 110	0	0	0
	specialty engineers	$ 110	0	0	0
	technicians	$ 80	0.25	0.25	0.25
	management	$ 130	0.25	0.25	0.25
	administration	$ 60	0.25	0.25	0.25
	production/consultants	$ 80	0	0	0
Calendar	(months)		1	5	3

Total NRE/phase =	3.75	3.75	3.75
Total cost/phase =	$ 66,780	$ 333,900	$ 200,340
		Total cost ($) =	$ 601,020
		Total time (m) =	9

Model update - new panel and buttons, new wire harness, 1 circuit board;

Digital Multimeter Development



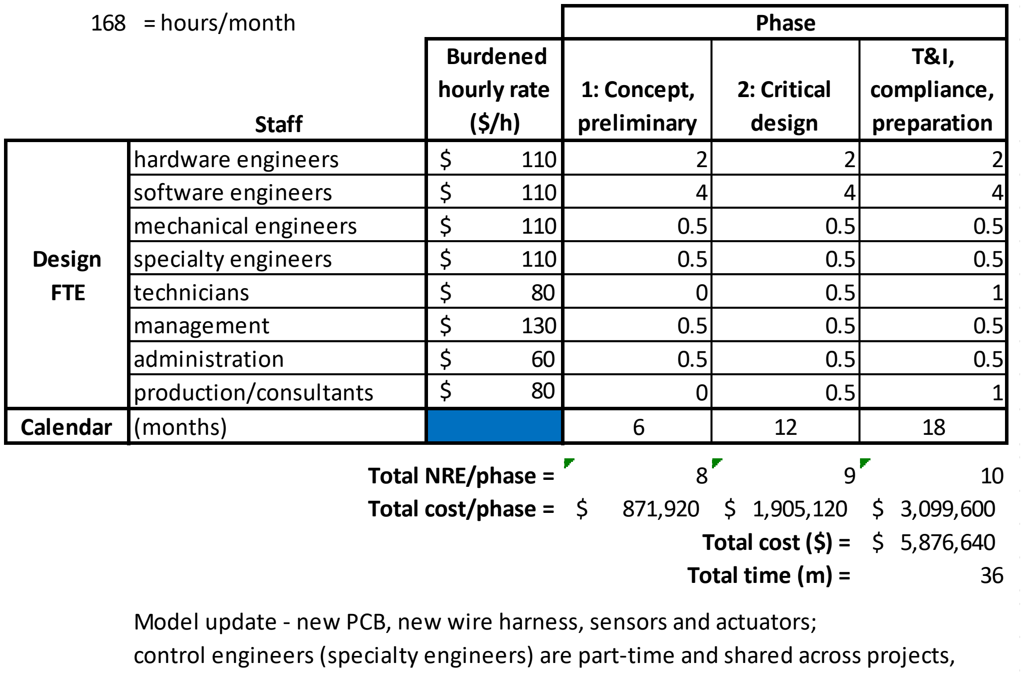
168 = hours/month

	Staff	Burdened hourly rate ($/h)	Phase 1: Concept, preliminary	 2: Critical design	 T&I, compliance, preparation
Design FTE	hardware engineers	$ 110	2	2	2
	software engineers	$ 110	1	1	1
	mechanical engineers	$ 110	0	0	0
	specialty engineers	$ 110	0.25	0.25	0.25
	technicians	$ 80	0.25	0.25	1
	management	$ 130	0.25	0.25	0.25
	administration	$ 60	0.25	0.25	0.25
	production/consultants	$ 80	0	0	0
Calendar	(months)		1	4	7

Total NRE/phase =	4	4	4.75
Total cost/phase =	$ 71,400	$ 285,600	$ 570,360
		Total cost ($) =	$ 927,360
		Total time (m) =	12

Model update - new panel display, 3 circuit boards: power supply, analog input, display + uC; 1 y time frame

Automobile Engine Control Module Development



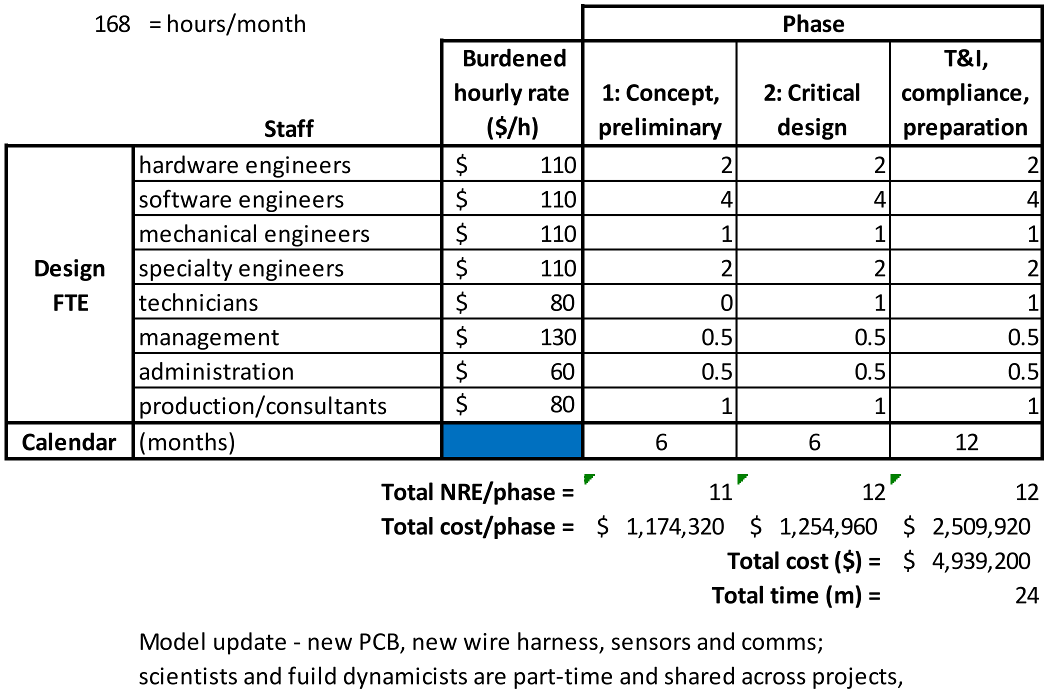
168 = hours/month

			Phase		
	Staff	Burdened hourly rate ($/h)	1: Concept, preliminary	2: Critical design	T&I, compliance, preparation
Design FTE	hardware engineers	$ 110	2	2	2
	software engineers	$ 110	4	4	4
	mechanical engineers	$ 110	0.5	0.5	0.5
	specialty engineers	$ 110	0.5	0.5	0.5
	technicians	$ 80	0	0.5	1
	management	$ 130	0.5	0.5	0.5
	administration	$ 60	0.5	0.5	0.5
	production/consultants	$ 80	0	0.5	1
Calendar	(months)	🟦	6	12	18

Total NRE/phase =	8	9	10
Total cost/phase =	$ 871,920	$ 1,905,120	$ 3,099,600
		Total cost ($) =	$ 5,876,640
		Total time (m) =	36

Model update - new PCB, new wire harness, sensors and actuators;
control engineers (specialty engineers) are part-time and shared across projects,

Oilfield Flowmeter Development



168 = hours/month

			Phase		
	Staff	Burdened hourly rate ($/h)	1: Concept, preliminary	2: Critical design	T&I, compliance, preparation
Design FTE	hardware engineers	$ 110	2	2	2
	software engineers	$ 110	4	4	4
	mechanical engineers	$ 110	1	1	1
	specialty engineers	$ 110	2	2	2
	technicians	$ 80	0	1	1
	management	$ 130	0.5	0.5	0.5
	administration	$ 60	0.5	0.5	0.5
	production/consultants	$ 80	1	1	1
Calendar	(months)	🟦	6	6	12

Total NRE/phase =	11	12	12
Total cost/phase =	$ 1,174,320	$ 1,254,960	$ 2,509,920
		Total cost ($) =	$ 4,939,200
		Total time (m) =	24

Model update - new PCB, new wire harness, sensors and comms;
scientists and fuild dynamicists are part-time and shared across projects,

Military Radar Development

168 = hours/month

			Phase		
	Staff	Burdened hourly rate ($/h)	1: Concept, preliminary	2: Critical design	T&I, compliance, preparation
Design FTE	hardware engineers	$ 110	2	2	2
	software engineers	$ 110	2	2	2
	mechanical engineers	$ 110	0.5	0.5	0.5
	specialty engineers	$ 110	1	1	1
	technicians	$ 80	1	2	2
	management	$ 130	0.5	0.5	0.5
	administration	$ 60	0.5	0.5	0.5
	production/consultants	$ 80	0	0	0
Calendar	(months)		6	12	30

Total NRE/phase =	7.5	8.5	8.5
Total cost/phase =	$ 786,240	$ 1,733,760	$ 4,334,400
		Total cost ($) =	$ 6,854,400
		Total time (m) =	48

clean sheet - new enclosure, display, new wire harness, sensors and comms;
radar engineer is specialty; both digital and analog PCBs are used

Spacecraft Imaging and Data Acquisition Development

168 = hours/month

			Phase		
	Staff	Burdened hourly rate ($/h)	1: Concept, preliminary	2: Critical design	T&I, compliance, preparation
Design FTE	hardware engineers	$ 110	2	2	18
	software engineers	$ 110	2	3	2
	mechanical engineers	$ 110	2	2	0.5
	specialty engineers	$ 110	0.5	0.5	1
	technicians	$ 80	2	2	2
	management	$ 130	0.5	0.5	0.5
	administration	$ 60	1	1	0.5
	production/consultants	$ 80	0.5	0.5	0
Calendar	(months)		12	12	24

Total NRE/phase =	10.5	11.5	24.5
Total cost/phase =	$ 2,096,640	$ 2,318,400	$ 10,563,840
		Total cost ($) =	$ 14,978,880
		Total time (m) =	48

Appendix E - Suggested Studies for the Estimator

Statistical Studies for Baselining the Sanity Checker

Outline of Studies

The sanity checker provides a baseline constraint that indicates the combined minimum of cost, time and effort for developing any embedded system. The baseline constraint is a completely optimistic boundary for a perfect development that minimizes latent errors, faults and failures in the design.

Problem: The problem is to determine the optimistic, minimum time and effort to design and develop a circuit board with a reasonable confidence level and low margin of error. Cost is assumed directly calculated from the time and effort; if significant resource costs arise during design and development, they can be added to the baseline constraint.

Approach: Two means are available to find these minimum values for time, effort and cost. The first type of study runs time-motion studies of volunteers performing each task. Time-motion studies are intensive and costly tasks to run. The second type of study is to conduct surveys of practitioners in industry. This appendix focuses on the surveys. The study will send surveys to designers who work in the specific sections defined in the sanity checker: software, circuit boards, cables, mechanisms, and enclosures.

Each section of the sanity checker (software, circuit boards, cables, mechanisms, and enclosures) requires a study to establish the statistical floor for its development. Each study will be a survey that has two or three example situations; respondents will fill in boxes to estimate the time and effort to complete development of each example situation. Examining the mean estimates and their variances will help establish levels of confidence, as seen in Table 8.2.

Once the results are completed, I will divide the total time by the number of individual components, much like LOC in software, to calculate a rate of development. For the circuit boards and cables, the individual components will be the conductor connections to electrical components on the board or in the cable. For the mechanical devices, the individual components will be the basic gears, pulleys, belts, bearings, and actuators. For the enclosures, the individual components will be the number of penetrations of the enclosure.

Table 8.2 Level of Confidence and the Margin of Error for Various Sample Sizes.

Population size	Confidence Level (%)	Margin of error (%)	Sample size needed
100,000	95	3	1,056
100,000	95	5	383
100,000	95	8	150
100,000	95	8.2	143
100,000	95	10	96
100,000	90	3	742
100,000	90	5	269
100,000	90	8	105
100,000	90	10	68
100,000	80	8	64
100,000	80	10	41

Format: The format will be the same for all four surveys: software, circuit boards and cables, mechanisms, and enclosures. The first part of the survey will determine the years of experience and what industry or industries for each respondent. It will also determine if they are directly involved in developing embedded systems. 0illustrates a reasonable model to follow for the surveys.

Following the initial identification section, the next section will determine how many different designs with which the respondents have experience or involvement. Then it will ask if they, or their companies, have any metrics for time or effort on delivered designs to production.

The third section will present two or three hypothetical cases. The survey will ask respondents to estimate the values of time and effort for each case.

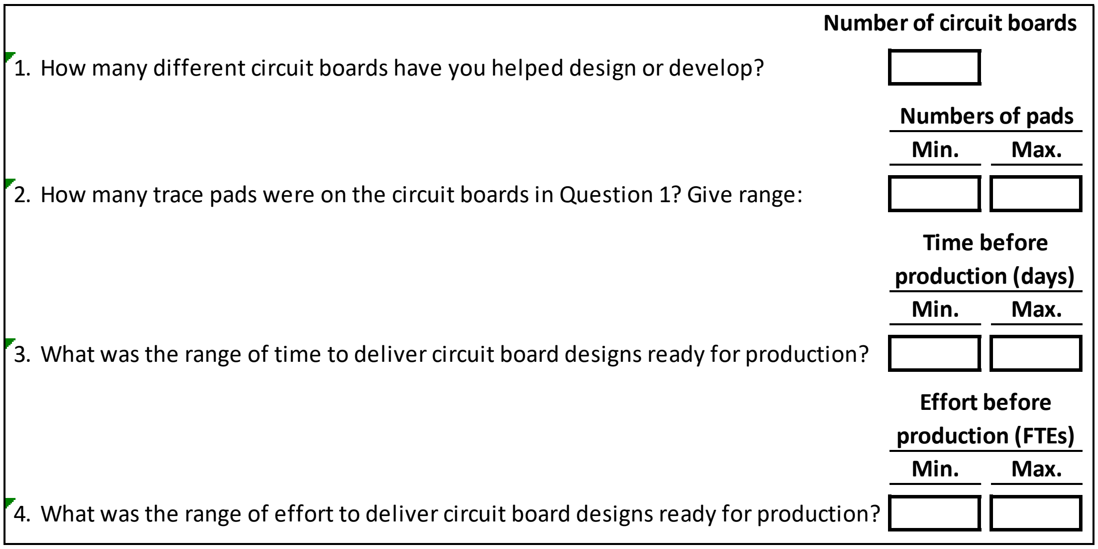
The fourth and final section will ask respondents to view and select a range of acceptable numbers for time and effort for each case. The four sub headed sections that follow will outline the individual surveys that follow the identification section.

Software Design and Development Baseline Study

This survey will be very short. Its primary purpose is to confirm the industry best of two LOC/h or to provide another more compelling number. It will ask for the company's accepted rate for LOC/h for delivered code that has been carefully developed internally.

Circuit Board Design and Development Baseline Study

Figure 8.5 illustrates a potential first page for this section of the survey. It attempts to glean experience and boundaries on circuit board design.



Number of circuit boards

1. How many different circuit boards have you helped design or develop? □

Numbers of pads
Min. Max.

2. How many trace pads were on the circuit boards in Question 1? Give range: □ □

Time before production (days)
Min. Max.

3. What was the range of time to deliver circuit board designs ready for production? □ □

Effort before production (FTEs)
Min. Max.

4. What was the range of effort to deliver circuit board designs ready for production? □ □

Figure 8.5 Potential opening page for specifics of circuit board development.

The next page in the tool is for the first case study for a clean-sheet design of a circuit board. Figure 8.6 illustrates a potential outline of an embedded system, which will be displayed to the respondent. Figure 8.7 gives a potential accompanying page to Figure 8.6.

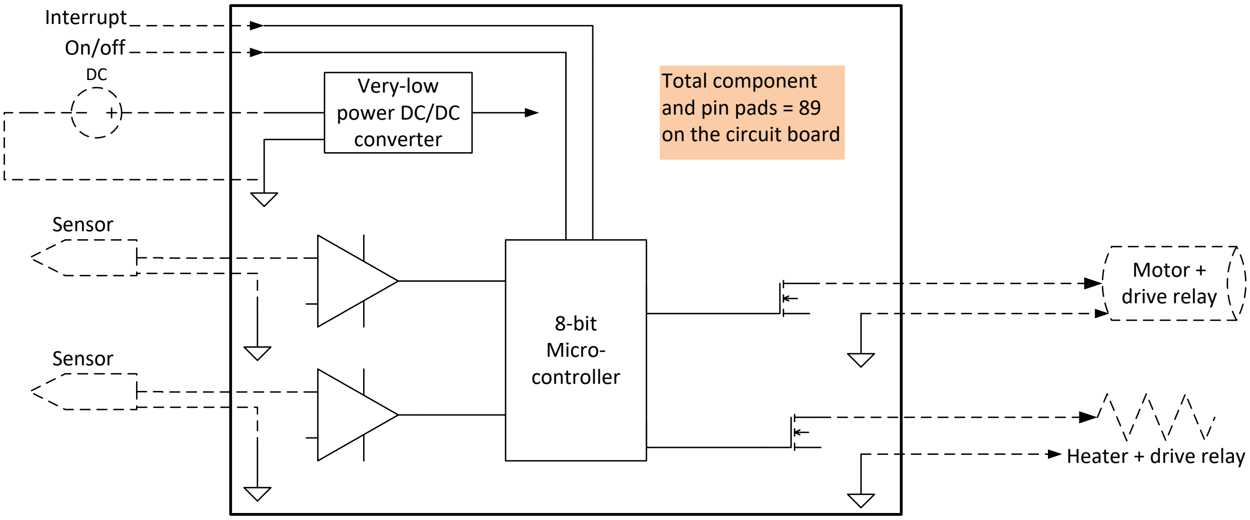
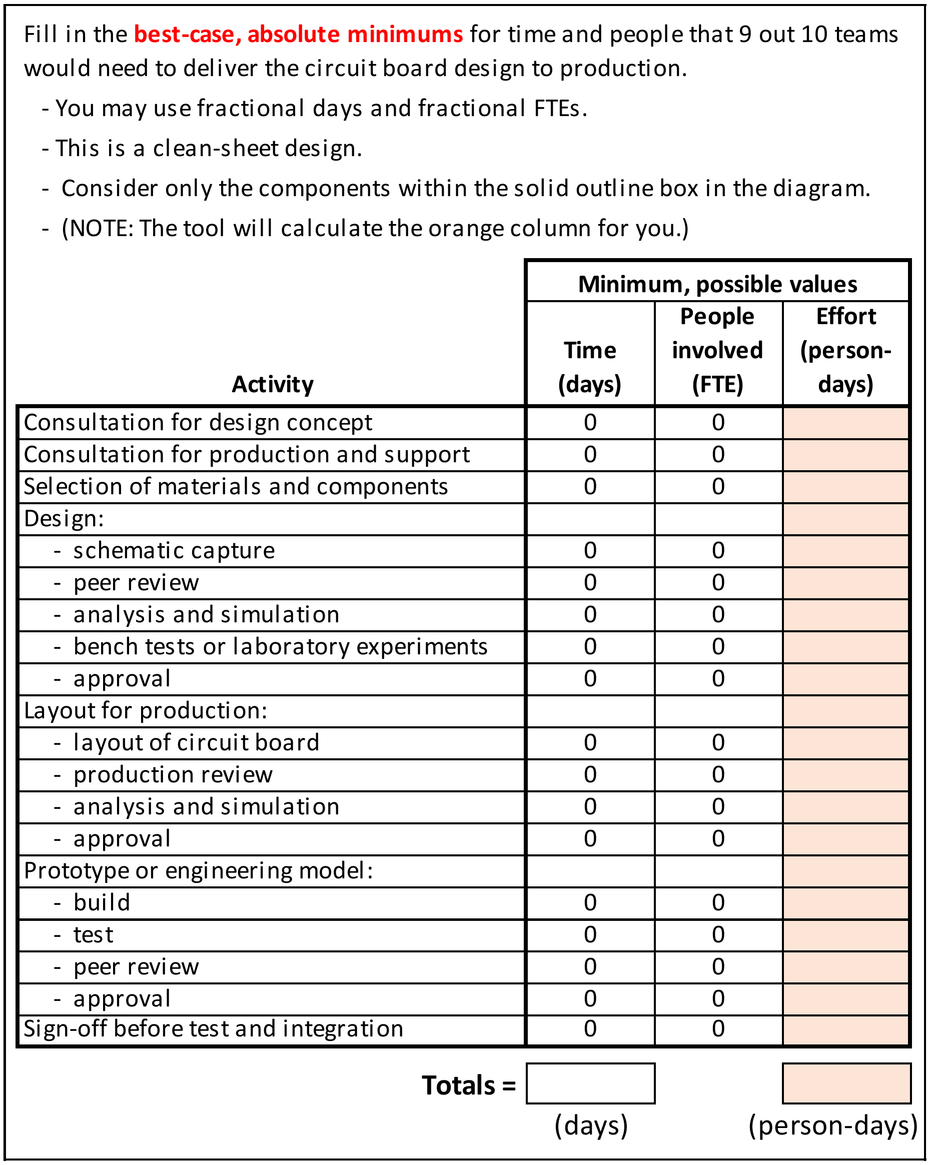


Figure 8.6 First case of a simple embedded system. The respondent will estimate the time and effort to develop the circuit board only within the solid lines.



Fill in the **best-case, absolute minimums** for time and people that 9 out 10 teams would need to deliver the circuit board design to production.

- You may use fractional days and fractional FTEs.
- This is a clean-sheet design.
- Consider only the components within the solid outline box in the diagram.
- (NOTE: The tool will calculate the orange column for you.)

	Minimum, possible values		
Activity	**Time (days)**	**People involved (FTE)**	**Effort (person-days)**
Consultation for design concept	0	0	
Consultation for production and support	0	0	
Selection of materials and components	0	0	
Design:			
- schematic capture	0	0	
- peer review	0	0	
- analysis and simulation	0	0	
- bench tests or laboratory experiments	0	0	
- approval	0	0	
Layout for production:			
- layout of circuit board	0	0	
- production review	0	0	
- analysis and simulation	0	0	
- approval	0	0	
Prototype or engineering model:			
- build	0	0	
- test	0	0	
- peer review	0	0	
- approval	0	0	
Sign-off before test and integration	0	0	
Totals =	(days)		(person-days)

Figure 8.7 Potential page for estimating the time and effort to develop the circuit board in each case study.

The second case study of a circuit board will follow the first. Figure 8.8 illustrates a potential outline of an embedded system, which will be displayed to the respondent. Figure 8.7 gives a potential accompanying page to Figure 8.8.

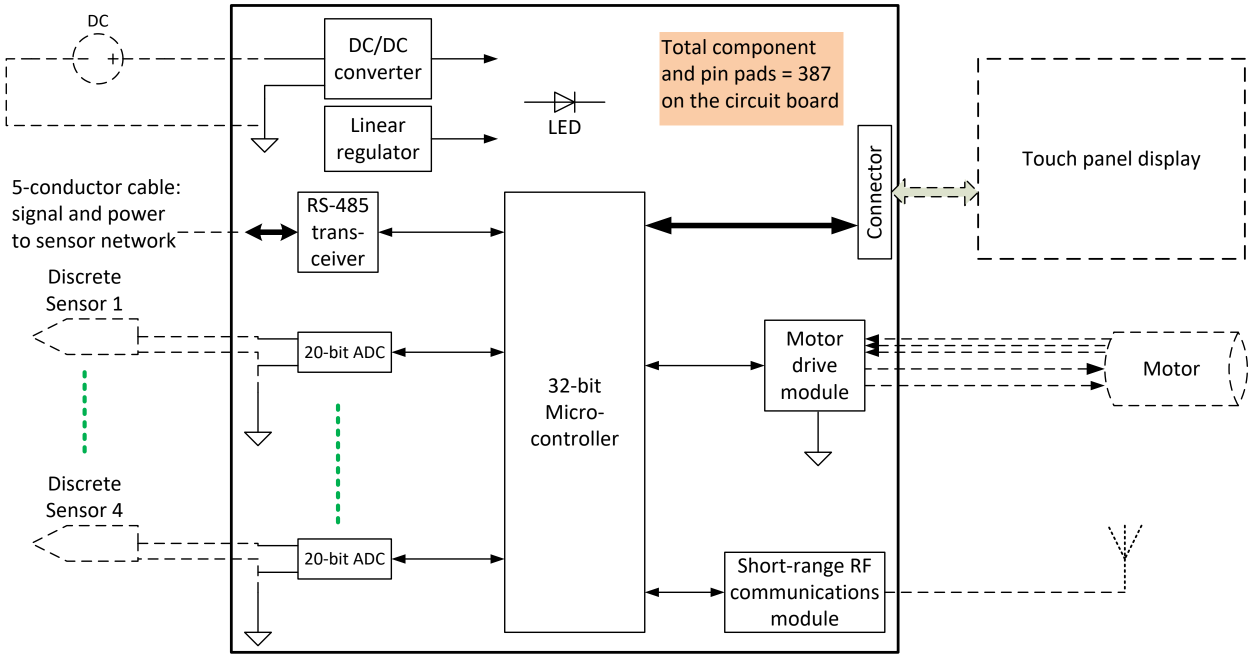
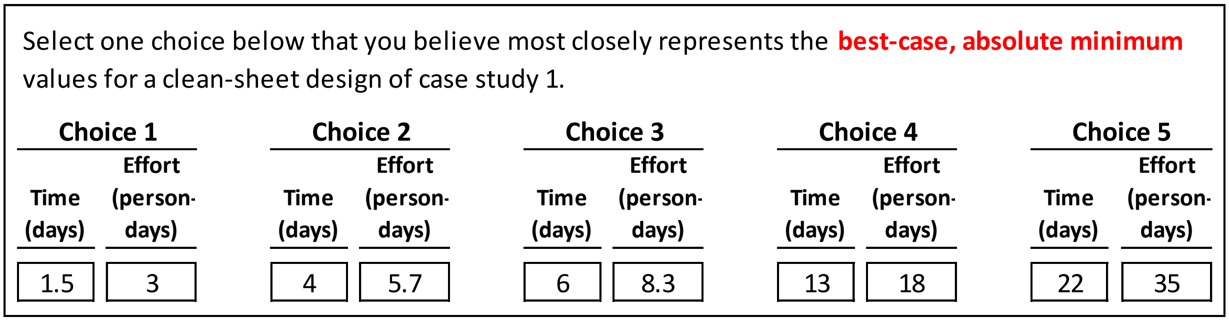


Figure 8.8 Second case of an embedded system. The respondent will estimate the time and effort to develop the circuit board only within the solid lines.

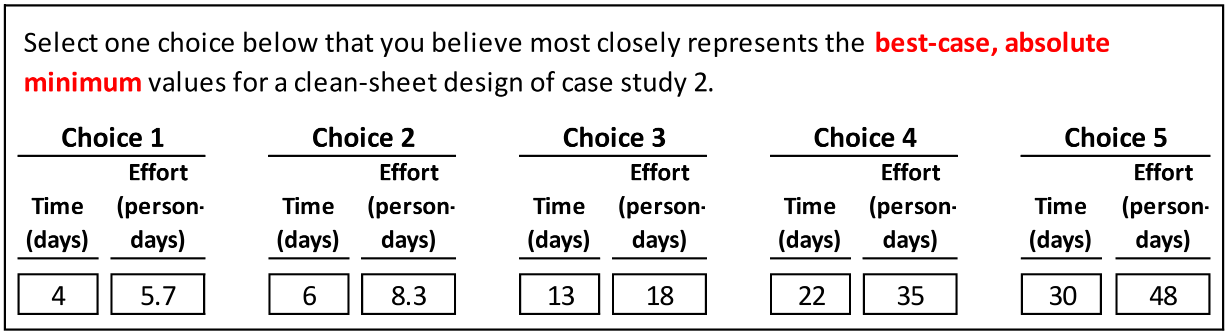
After the second case, the fourth section will ask respondents to view and select a range of acceptable numbers for time and effort for each case of clean-sheet design and development. It will repeat the diagrams in Figure 8.6 and Figure 8.8, and then for each diagram, it will ask for the respondent to select one of the columns in either Figure 8.9 or Figure 8.10.



Select one choice below that you believe most closely represents the **best-case, absolute minimum** values for a clean-sheet design of case study 1.

Choice 1		Choice 2		Choice 3		Choice 4		Choice 5	
Time (days)	Effort (person-days)	Time (days)	Effort (person-days)	Time (days)	Effort (person-days)	Time (days)	Effort (person-days)	Time (days)	Effort (person-days)
1.5	3	4	5.7	6	8.3	13	18	22	35

Figure 8.9 Potential page for checking the estimate for the time and effort to develop the circuit board in Case Study 1.



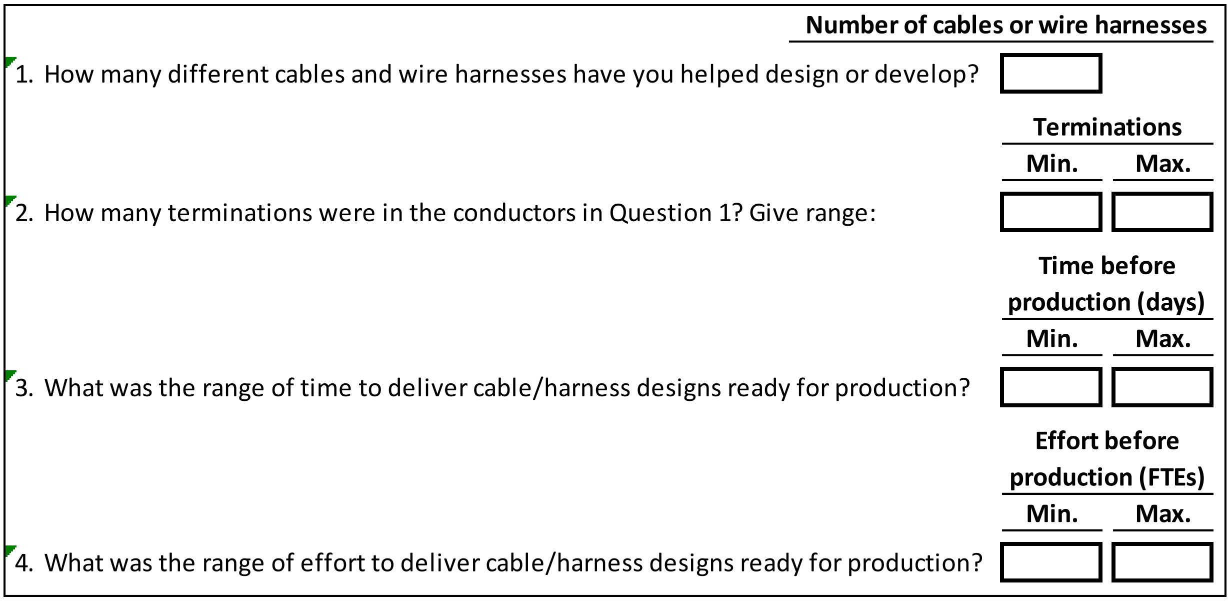
Select one choice below that you believe most closely represents the **best-case, absolute minimum** values for a clean-sheet design of case study 2.

Choice 1		Choice 2		Choice 3		Choice 4		Choice 5	
Time (days)	Effort (person-days)	Time (days)	Effort (person-days)	Time (days)	Effort (person-days)	Time (days)	Effort (person-days)	Time (days)	Effort (person-days)
4	5.7	6	8.3	13	18	22	35	30	48

Figure 8.10 Potential page for checking the estimate for the time and effort to develop the circuit board in Case Study 2.

Cable Design and Development Baseline Study

Figure 8.11 illustrates a potential first page for this section of the survey. It attempts to glean experience and boundaries on the design of cables and wire harnesses.



Number of cables or wire harnesses

1. How many different cables and wire harnesses have you helped design or develop? □

Terminations

Min. **Max.**

2. How many terminations were in the conductors in Question 1? Give range: □ □

Time before production (days)

Min. **Max.**

3. What was the range of time to deliver cable/harness designs ready for production? □ □

Effort before production (FTEs)

Min. **Max.**

4. What was the range of effort to deliver cable/harness designs ready for production? □ □

Figure 8.11 Potential opening page for specifics of development of cables and wire harnesses.

The next page in the tool is for the first case study for a clean-sheet design of a set of cables/wire harness. Figure 8.12 illustrates a potential outline of an embedded system, which will be displayed to the respondent. Figure 8.13 gives a potential accompanying page to Figure 8.12.

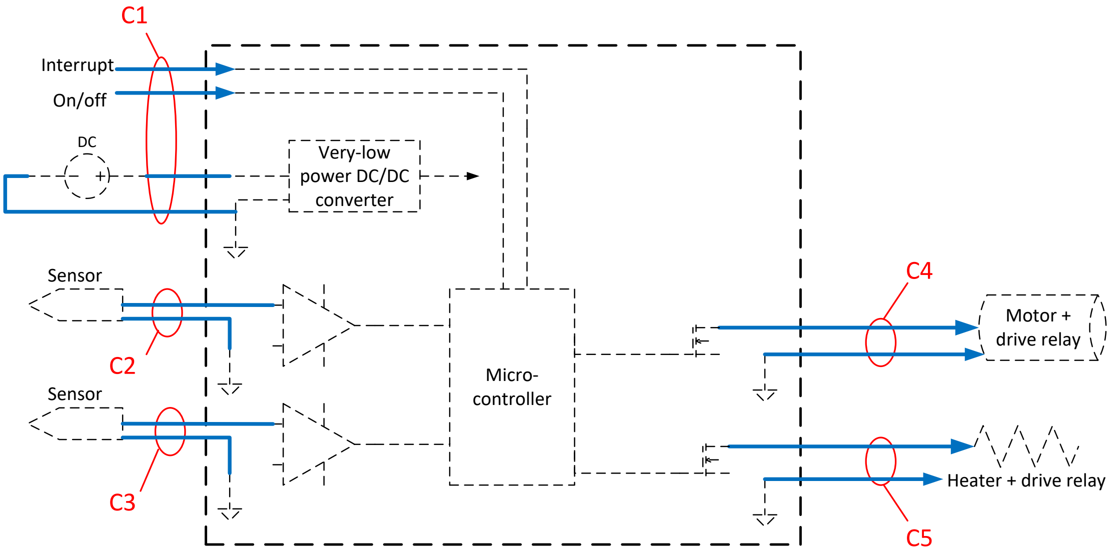
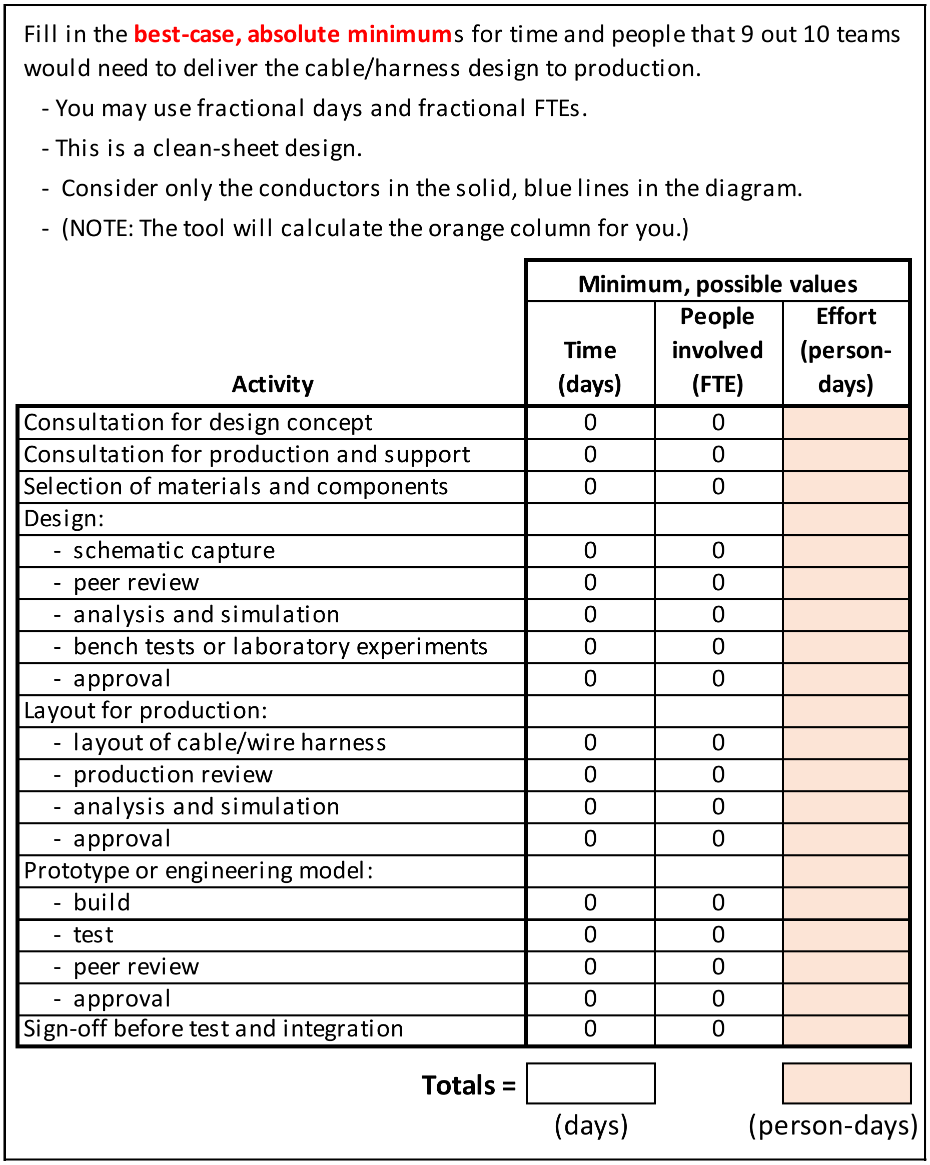


Figure 8.12 First case of a simple embedded system. The respondent will estimate the time and effort to develop the cables or wire harnesses in the solid, blue lines.

The second case study of cables and wire harness will follow the first. Figure 8.14 illustrates a potential outline of an embedded system, which will be displayed to the respondent. Figure 8.13 gives a potential accompanying page to Figure 8.14.

After the second case, the fourth section will ask respondents to view and select a range of acceptable numbers for time and effort for each case of clean-sheet design and development. It will repeat the diagrams in Figure 8.12 and Figure 8.14, and then for each diagram, it will ask for the respondent to select one of the columns in either Figure 8.15 or Figure 8.16.



Fill in the **best-case, absolute minimum**s for time and people that 9 out 10 teams would need to deliver the cable/harness design to production.

- You may use fractional days and fractional FTEs.
- This is a clean-sheet design.
- Consider only the conductors in the solid, blue lines in the diagram.
- (NOTE: The tool will calculate the orange column for you.)

	Minimum, possible values		
Activity	**Time (days)**	**People involved (FTE)**	**Effort (person-days)**
Consultation for design concept	0	0	
Consultation for production and support	0	0	
Selection of materials and components	0	0	
Design:			
- schematic capture	0	0	
- peer review	0	0	
- analysis and simulation	0	0	
- bench tests or laboratory experiments	0	0	
- approval	0	0	
Layout for production:			
- layout of cable/wire harness	0	0	
- production review	0	0	
- analysis and simulation	0	0	
- approval	0	0	
Prototype or engineering model:			
- build	0	0	
- test	0	0	
- peer review	0	0	
- approval	0	0	
Sign-off before test and integration	0	0	

Totals = □ (days) □ (person-days)

Figure 8.13 Potential page for estimating the time and effort to develop the cables or wire harnesses in each case study.

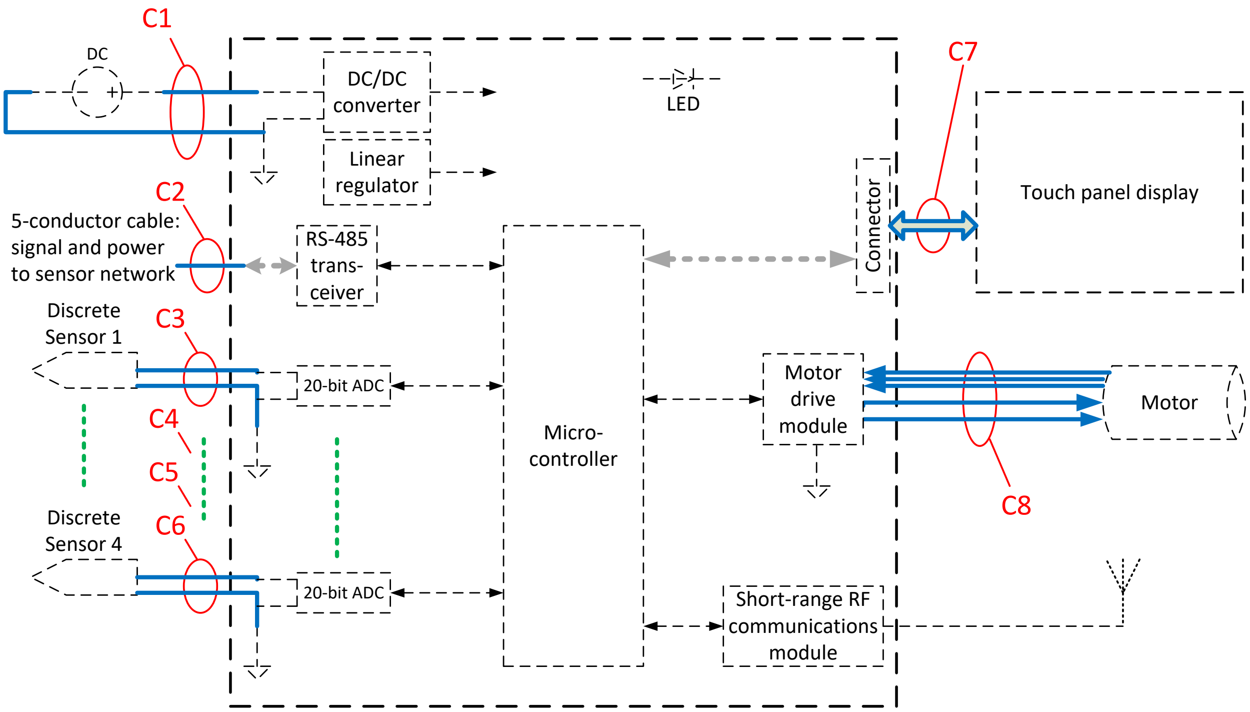
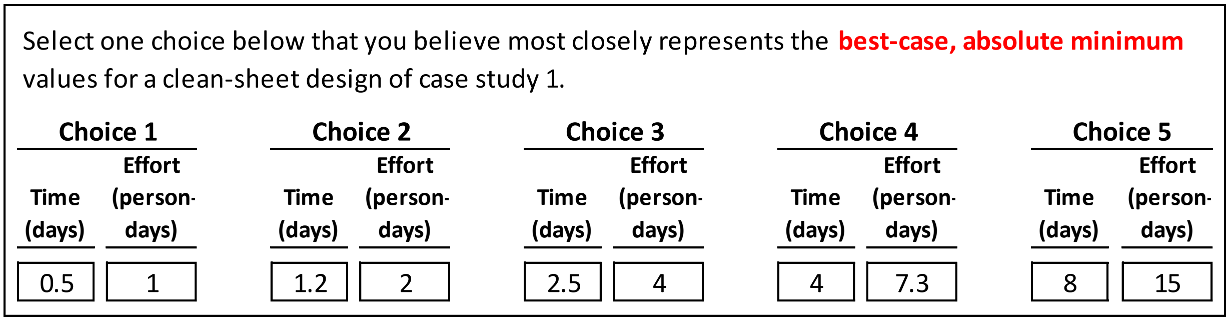


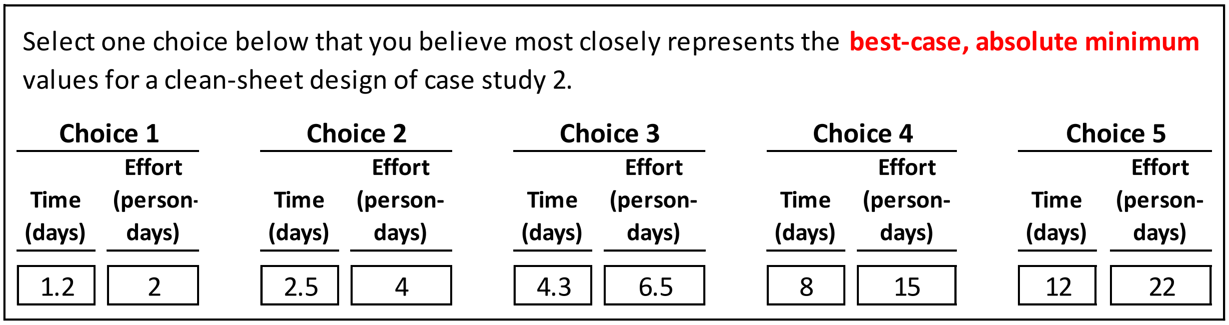
Figure 8.14 Second case of an embedded system. The respondent will estimate the time and effort to develop the cables or wire harnesses in the solid, blue lines.



Select one choice below that you believe most closely represents the **best-case, absolute minimum** values for a clean-sheet design of case study 1.

Choice 1		Choice 2		Choice 3		Choice 4		Choice 5	
Time (days)	Effort (person-days)	Time (days)	Effort (person-days)	Time (days)	Effort (person-days)	Time (days)	Effort (person-days)	Time (days)	Effort (person-days)
0.5	1	1.2	2	2.5	4	4	7.3	8	15

Figure 8.15 Potential page for checking the estimate for the time and effort to develop the cables or wire harness in Case Study 1.



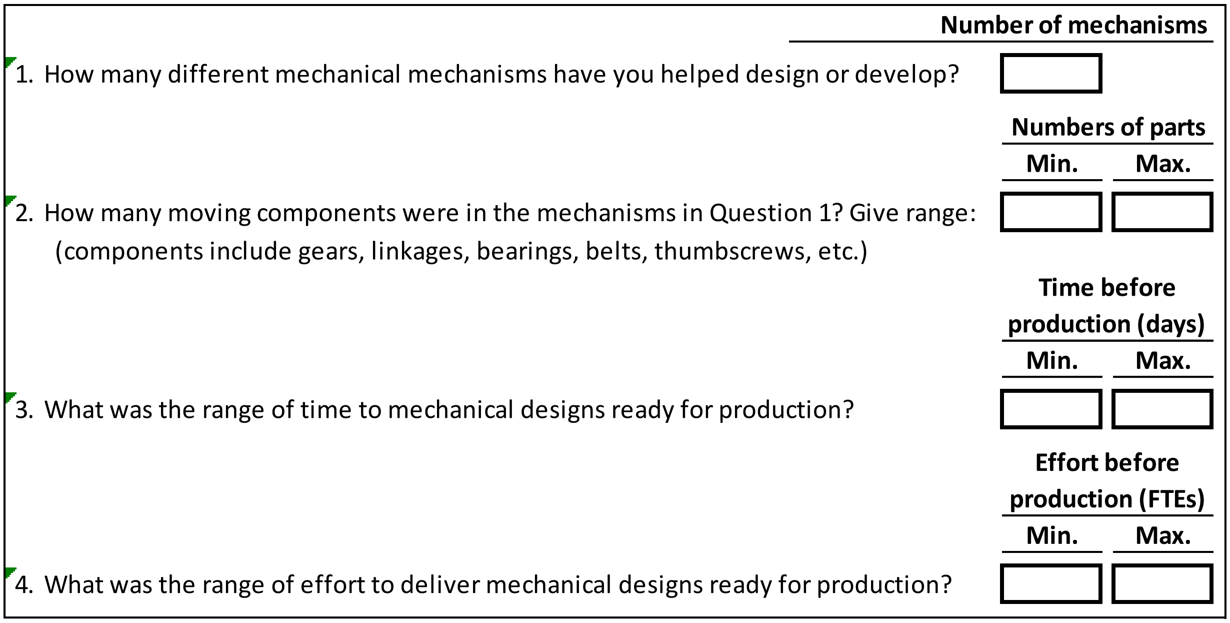
Select one choice below that you believe most closely represents the **best-case, absolute minimum** values for a clean-sheet design of case study 2.

Choice 1		Choice 2		Choice 3		Choice 4		Choice 5	
Time (days)	Effort (person-days)	Time (days)	Effort (person-days)	Time (days)	Effort (person-days)	Time (days)	Effort (person-days)	Time (days)	Effort (person-days)
1.2	2	2.5	4	4.3	6.5	8	15	12	22

Figure 8.16 Potential page for checking the estimate for the time and effort to develop the cables or wire harness in Case Study 2.

Mechanism Design and Development Baseline Study

Figure 8.17 illustrates a potential first page for this section of the survey. It attempts to glean experience and boundaries on the design of mechanics and mechanisms.



Number of mechanisms

1. How many different mechanical mechanisms have you helped design or develop? □

Numbers of parts — **Min.** **Max.**

2. How many moving components were in the mechanisms in Question 1? Give range: □ □
   (components include gears, linkages, bearings, belts, thumbscrews, etc.)

Time before production (days) — **Min.** **Max.**

3. What was the range of time to mechanical designs ready for production? □ □

Effort before production (FTEs) — **Min.** **Max.**

4. What was the range of effort to deliver mechanical designs ready for production? □ □

Figure 8.17 Potential opening page for specifics of development of mechanisms.

The next page in the tool is for the first case study for a clean-sheet design of a mechanism. Figure 8.18 illustrates a potential outline of a mechanism, which will be displayed to the respondent. Figure 8.19 gives a potential accompanying page to Figure 8.18.

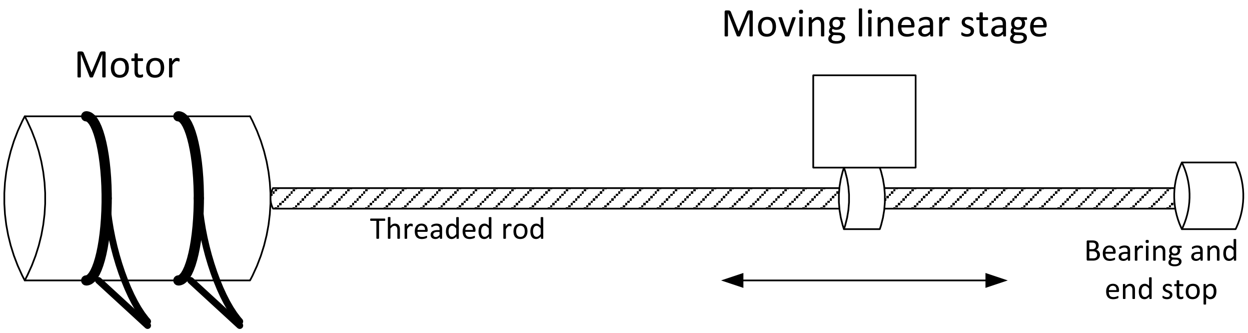
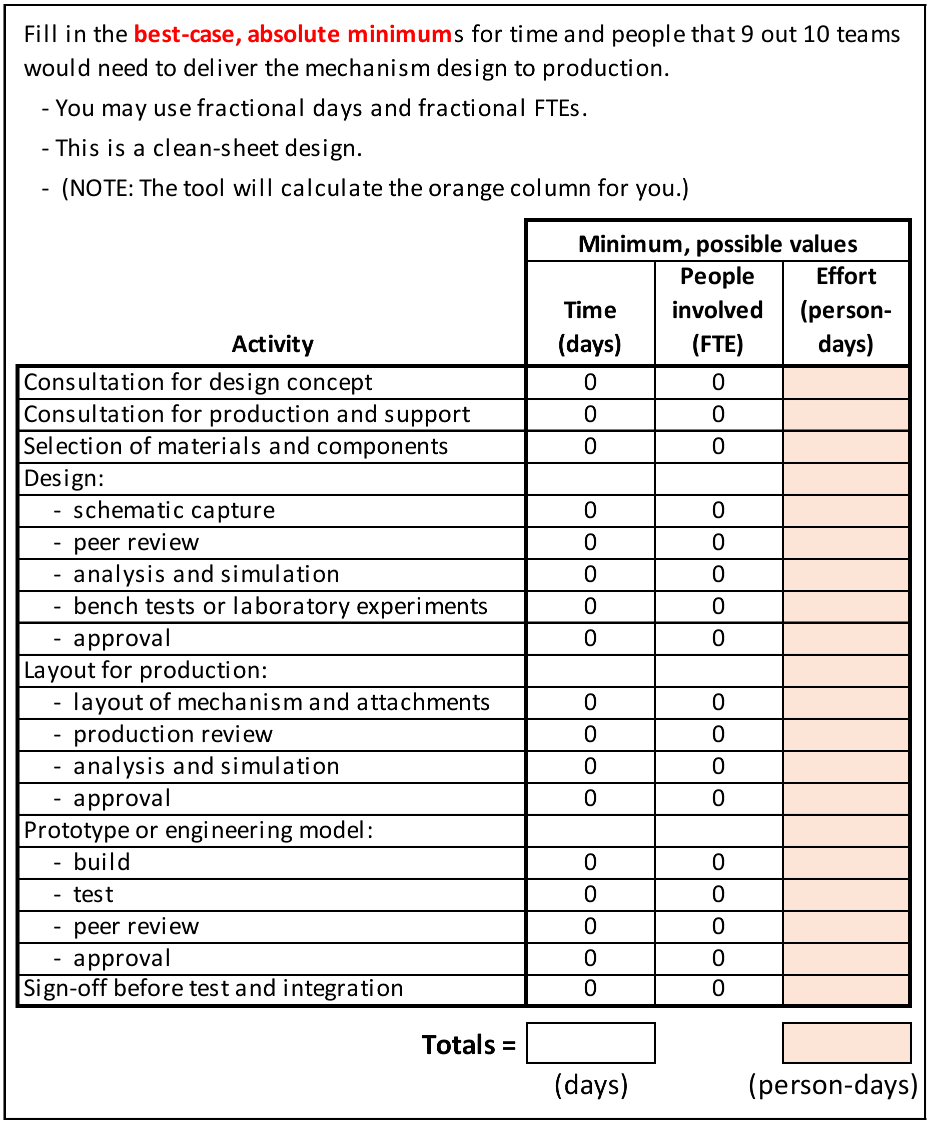


Figure 8.18 First case of a simple mechanism. The respondent will estimate the time and effort to develop the mechanism.

The second case study of mechanisms will follow the first. Figure 8.20 illustrates a potential outline of a mechanism, which will be displayed to the respondent. Figure 8.19 gives a potential accompanying page to Figure 8.20.

The third case study of mechanisms will follow the second. Figure 8.21 illustrates a potential outline of a mechanism, which will be displayed to the respondent. Figure 8.19 gives a potential accompanying page to Figure 8.21.

After the third case, the fourth section will ask respondents to view and select a range of acceptable numbers for time and effort for each case of clean-sheet design and development. It will repeat the diagrams in Figure 8.18, Figure 8.20 and Figure 8.21, and then for each diagram, it will ask for the respondent to select one of the columns in Figure 8.22, Figure 8.23 or Figure 8.24.



Fill in the **best-case, absolute minimum**s for time and people that 9 out 10 teams would need to deliver the mechanism design to production.

- You may use fractional days and fractional FTEs.
- This is a clean-sheet design.
- (NOTE: The tool will calculate the orange column for you.)

	Minimum, possible values		
Activity	**Time (days)**	**People involved (FTE)**	**Effort (person-days)**
Consultation for design concept	0	0	
Consultation for production and support	0	0	
Selection of materials and components	0	0	
Design:			
- schematic capture	0	0	
- peer review	0	0	
- analysis and simulation	0	0	
- bench tests or laboratory experiments	0	0	
- approval	0	0	
Layout for production:			
- layout of mechanism and attachments	0	0	
- production review	0	0	
- analysis and simulation	0	0	
- approval	0	0	
Prototype or engineering model:			
- build	0	0	
- test	0	0	
- peer review	0	0	
- approval	0	0	
Sign-off before test and integration	0	0	

Totals = □ □

(days) (person-days)

Figure 8.19 Potential page for estimating the time and effort to develop the mechanisms in each case study.

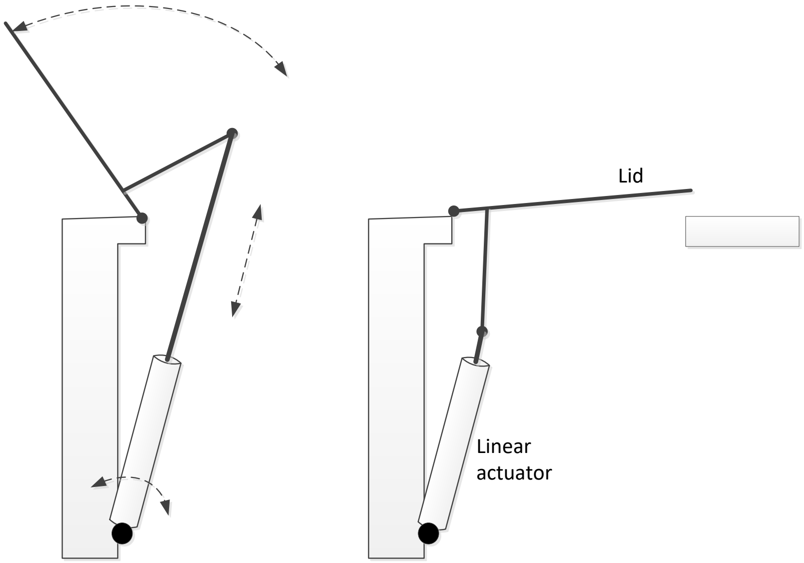


Figure 8.20 Second case of a mechanism. The respondent will estimate the time and effort to develop the mechanism.

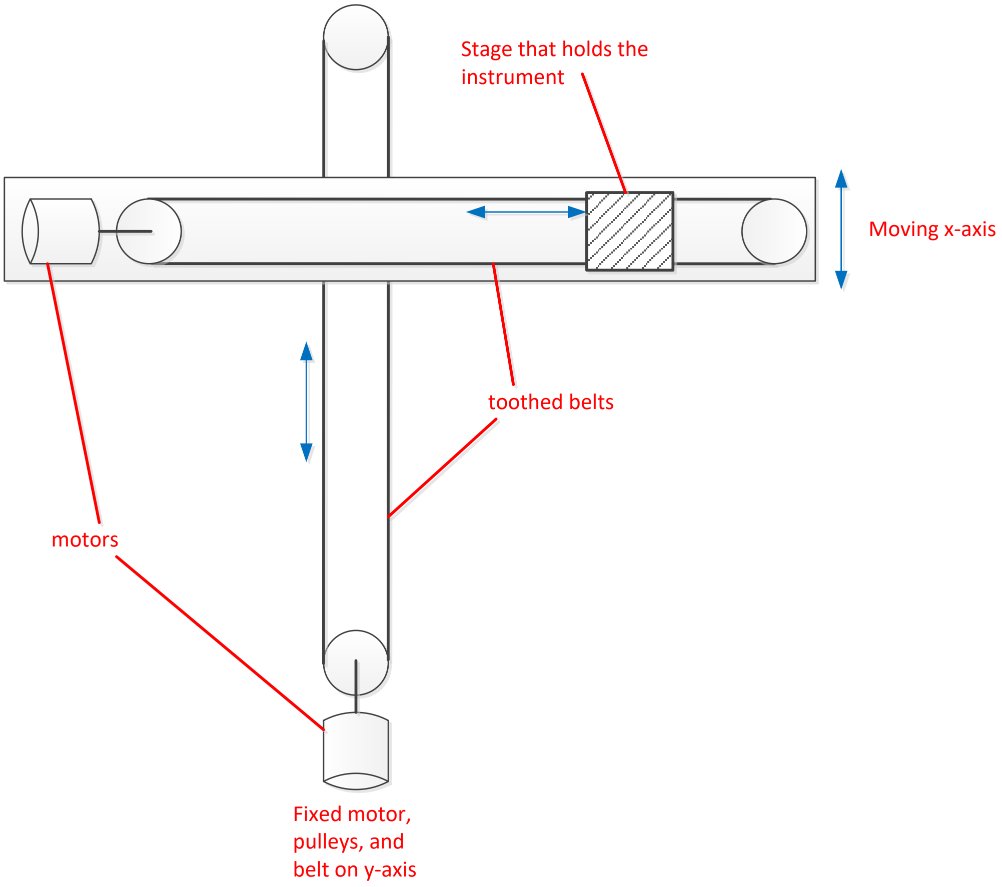
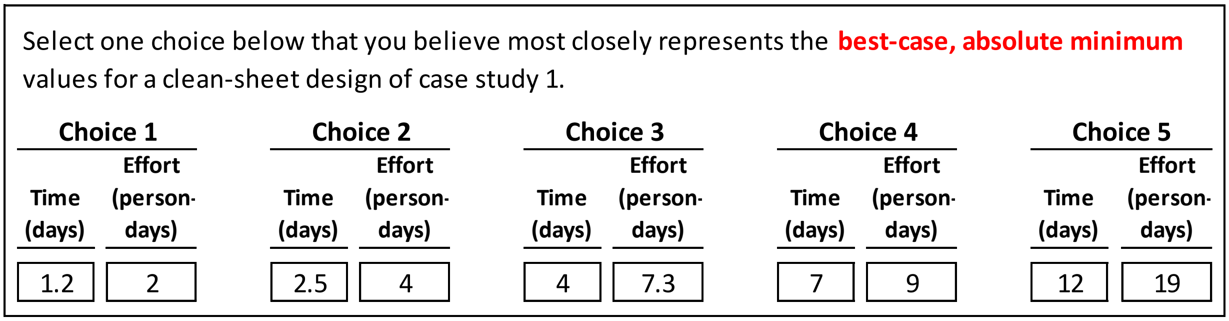


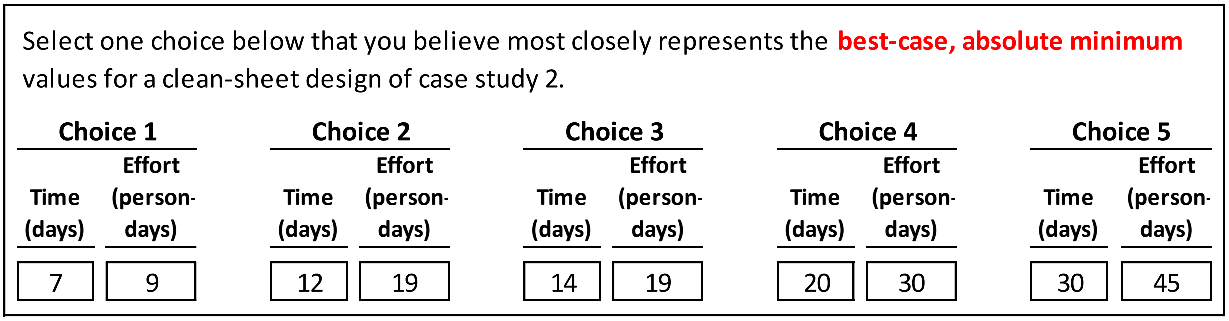
Figure 8.21 Third case of a mechanism. The respondent will estimate the time and effort to develop the mechanism.



Select one choice below that you believe most closely represents the **best-case, absolute minimum** values for a clean-sheet design of case study 1.

Choice 1		Choice 2		Choice 3		Choice 4		Choice 5	
Time (days)	Effort (person-days)	Time (days)	Effort (person-days)	Time (days)	Effort (person-days)	Time (days)	Effort (person-days)	Time (days)	Effort (person-days)
1.2	2	2.5	4	4	7.3	7	9	12	19

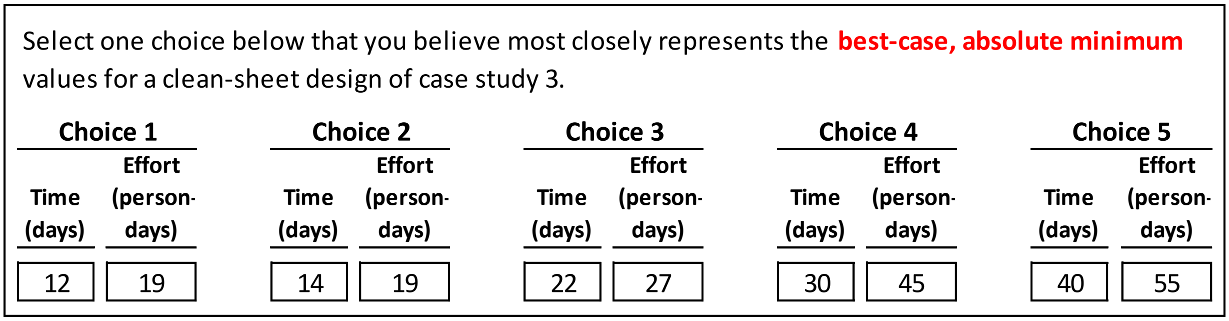
Figure 8.22 Potential page for checking the estimate for the time and effort to develop the mechanism in case study 1.



Select one choice below that you believe most closely represents the **best-case, absolute minimum** values for a clean-sheet design of case study 2.

Choice 1		Choice 2		Choice 3		Choice 4		Choice 5	
Time (days)	Effort (person-days)	Time (days)	Effort (person-days)	Time (days)	Effort (person-days)	Time (days)	Effort (person-days)	Time (days)	Effort (person-days)
7	9	12	19	14	19	20	30	30	45

Figure 8.23 Potential page for checking the estimate for the time and effort to develop the mechanism in case study 2.



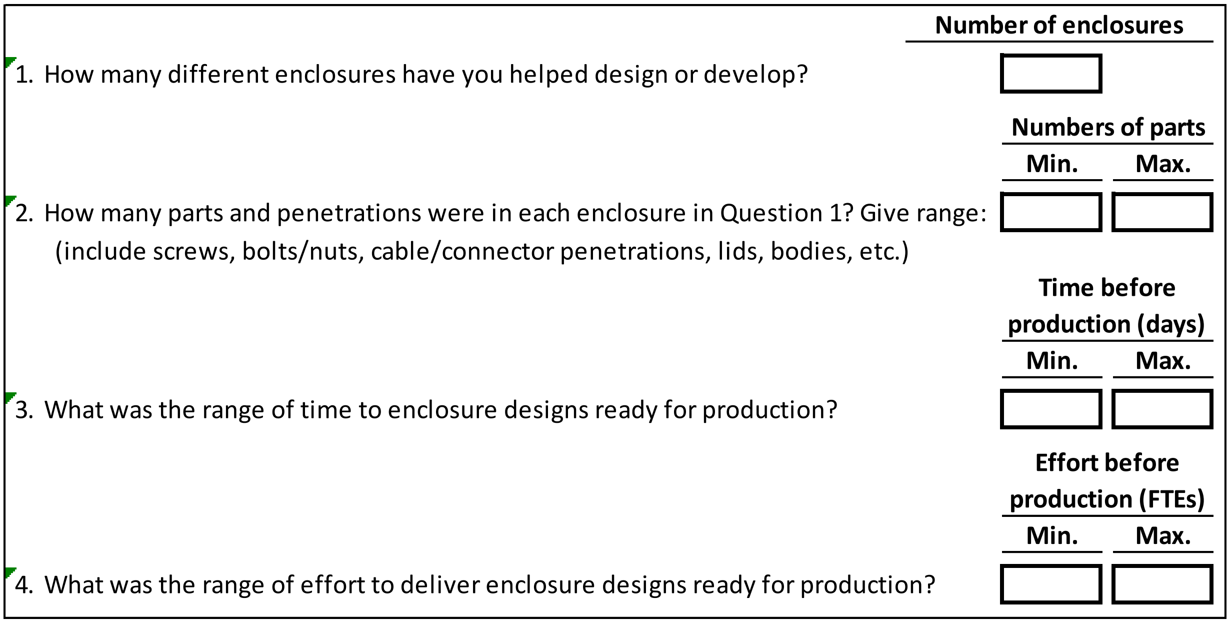
Select one choice below that you believe most closely represents the **best-case, absolute minimum** values for a clean-sheet design of case study 3.

Choice 1		Choice 2		Choice 3		Choice 4		Choice 5	
Time (days)	Effort (person-days)	Time (days)	Effort (person-days)	Time (days)	Effort (person-days)	Time (days)	Effort (person-days)	Time (days)	Effort (person-days)
12	19	14	19	22	27	30	45	40	55

Figure 8.24 Potential page for checking the estimate for the time and effort to develop the mechanism in case study 3.

Enclosure Design and Development Baseline Study

Figure 8.25 illustrates a potential first page for this section of the survey. It attempts to glean experience and boundaries on the design of enclosures.



Number of enclosures

1. How many different enclosures have you helped design or develop? □

Numbers of parts

Min. **Max.**

2. How many parts and penetrations were in each enclosure in Question 1? Give range: □ □
   (include screws, bolts/nuts, cable/connector penetrations, lids, bodies, etc.)

Time before production (days)

Min. **Max.**

3. What was the range of time to enclosure designs ready for production? □ □

Effort before production (FTEs)

Min. **Max.**

4. What was the range of effort to deliver enclosure designs ready for production? □ □

Figure 8.25 Potential opening page for specifics of development of enclosures.

The next page in the tool is for the first case study for a clean-sheet design of an enclosure. Figure 8.26 illustrates a potential outline of an enclosure, which will be displayed to the respondent. Figure 8.27 gives a potential accompanying page to Figure 8.26.

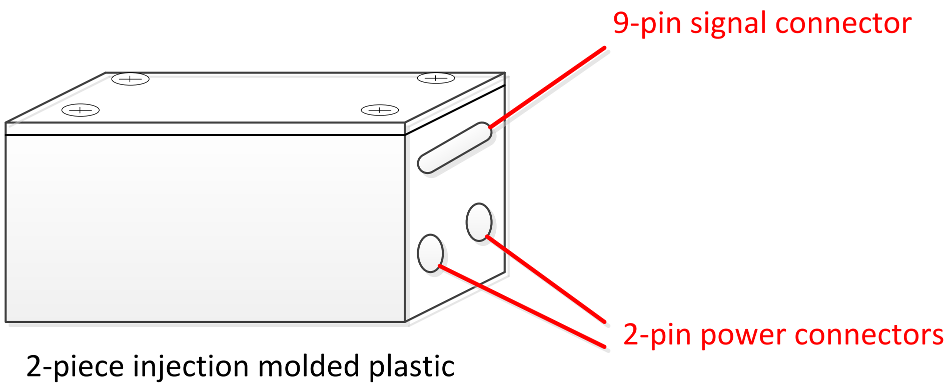
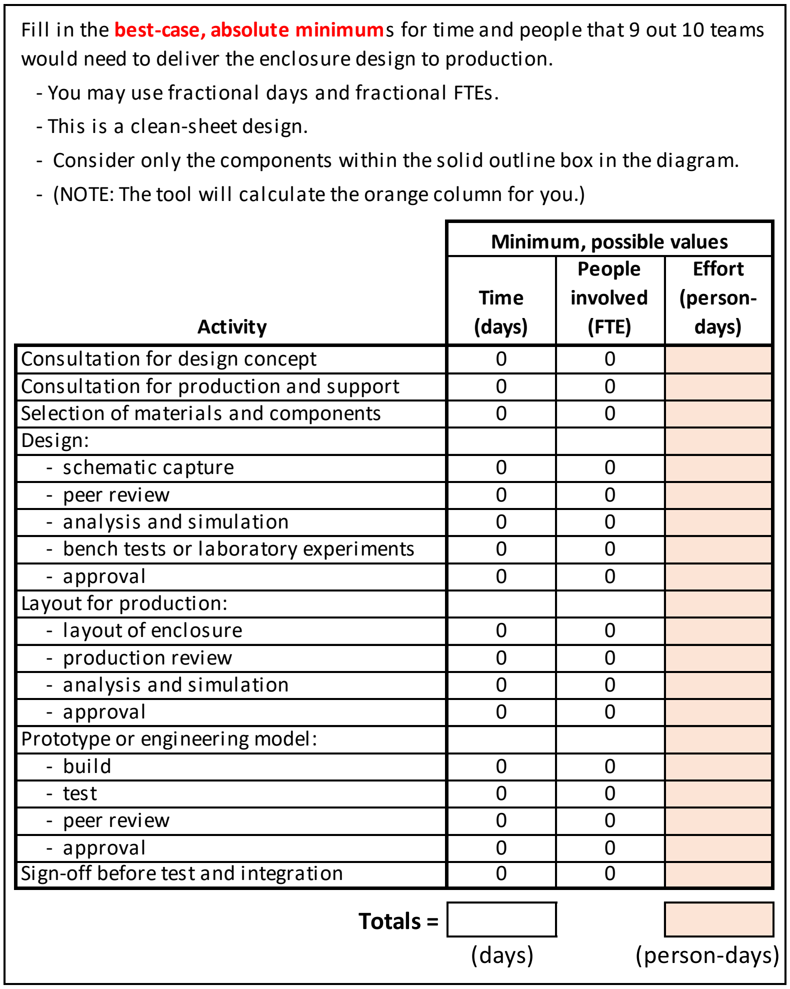


Figure 8.26 First case of a simple enclosure. The respondent will estimate the time and effort to develop the enclosure.

The second case study of an enclosure will follow the first. Figure 8.28 illustrates a potential outline of an enclosure, which will be displayed to the respondent. Figure 8.27 gives a potential accompanying page to Figure 8.28.

The third case study of an enclosure will follow the second. Figure 8.29 illustrates a potential outline of an enclosure, which will be displayed to the respondent. Figure 8.27 gives a potential accompanying page to Figure 8.29.

After the third case, the fourth section will ask respondents to select a range of acceptable numbers for time and effort for each case of clean-sheet design and development. It will repeat the diagrams in Figure 8.26, Figure 8.28 and Figure 8.29, and then for each diagram, it will ask for the respondent to select one of the columns in Figure 8.30, Figure 8.31 or Figure 8.32.



Fill in the **best-case, absolute minimum**s for time and people that 9 out 10 teams would need to deliver the enclosure design to production.

- You may use fractional days and fractional FTEs.
- This is a clean-sheet design.
- Consider only the components within the solid outline box in the diagram.
- (NOTE: The tool will calculate the orange column for you.)

	Minimum, possible values		
Activity	Time (days)	People involved (FTE)	Effort (person-days)
Consultation for design concept	0	0	
Consultation for production and support	0	0	
Selection of materials and components	0	0	
Design:			
- schematic capture	0	0	
- peer review	0	0	
- analysis and simulation	0	0	
- bench tests or laboratory experiments	0	0	
- approval	0	0	
Layout for production:			
- layout of enclosure	0	0	
- production review	0	0	
- analysis and simulation	0	0	
- approval	0	0	
Prototype or engineering model:			
- build	0	0	
- test	0	0	
- peer review	0	0	
- approval	0	0	
Sign-off before test and integration	0	0	

Totals = □ (days) □ (person-days)

Figure 8.27 Potential page for estimating the time and effort to develop the enclosures in each case study.

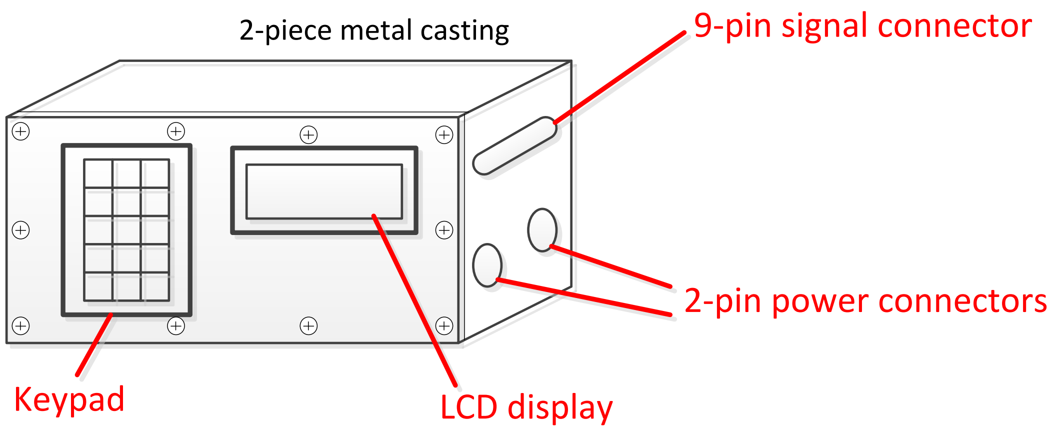


Figure 8.28 Second case of an enclosure. The respondent will estimate the time and effort to develop the enclosure.

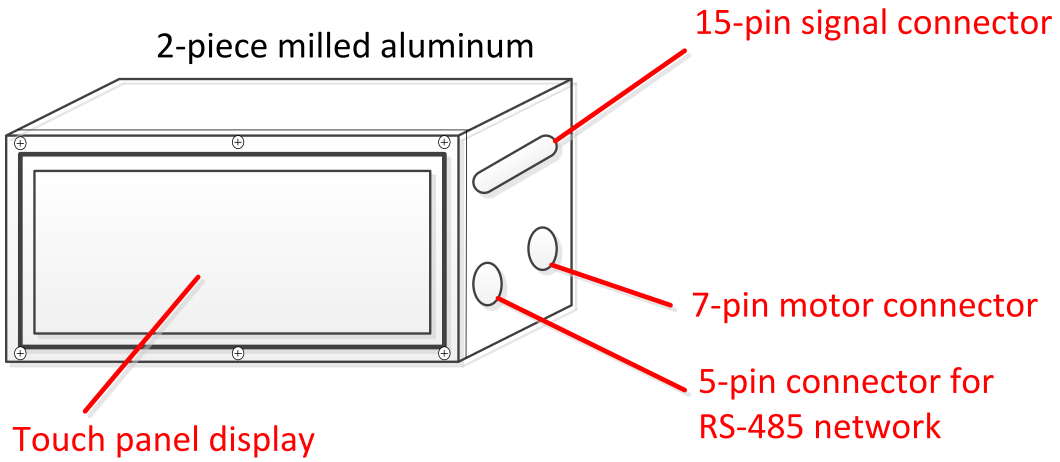
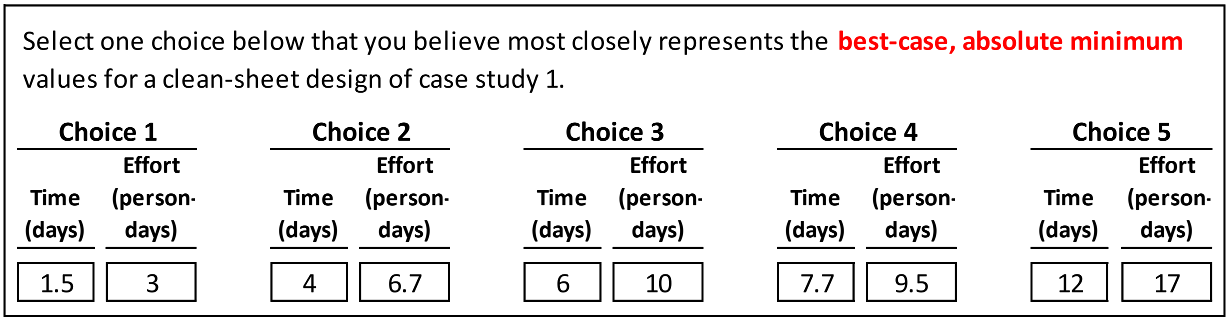


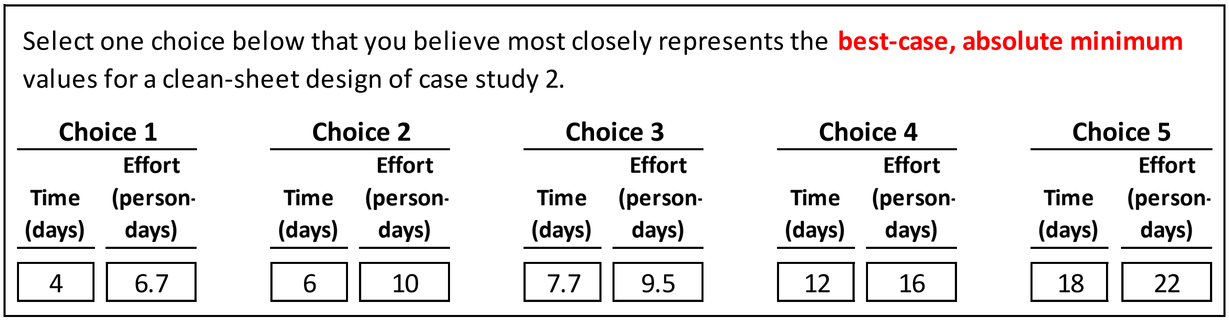
Figure 8.29 Third case of an enclosure. The respondent will estimate the time and effort to develop the enclosure.



Select one choice below that you believe most closely represents the **best-case, absolute minimum** values for a clean-sheet design of case study 1.

Choice 1		Choice 2		Choice 3		Choice 4		Choice 5	
Time (days)	Effort (person-days)	Time (days)	Effort (person-days)	Time (days)	Effort (person-days)	Time (days)	Effort (person-days)	Time (days)	Effort (person-days)
1.5	3	4	6.7	6	10	7.7	9.5	12	17

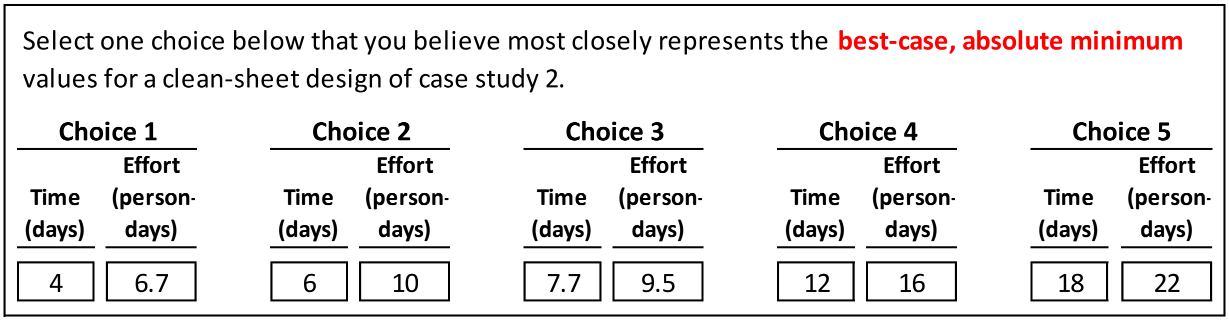
Figure 8.30 Potential page for checking the estimate for the time and effort to develop the enclosure in Case Study 1.



Select one choice below that you believe most closely represents the **best-case, absolute minimum** values for a clean-sheet design of case study 2.

Choice 1		Choice 2		Choice 3		Choice 4		Choice 5	
Time (days)	Effort (person-days)	Time (days)	Effort (person-days)	Time (days)	Effort (person-days)	Time (days)	Effort (person-days)	Time (days)	Effort (person-days)
4	6.7	6	10	7.7	9.5	12	16	18	22

Figure 8.31 Potential page for checking the estimate for the time and effort to develop the enclosure in Case Study 2.



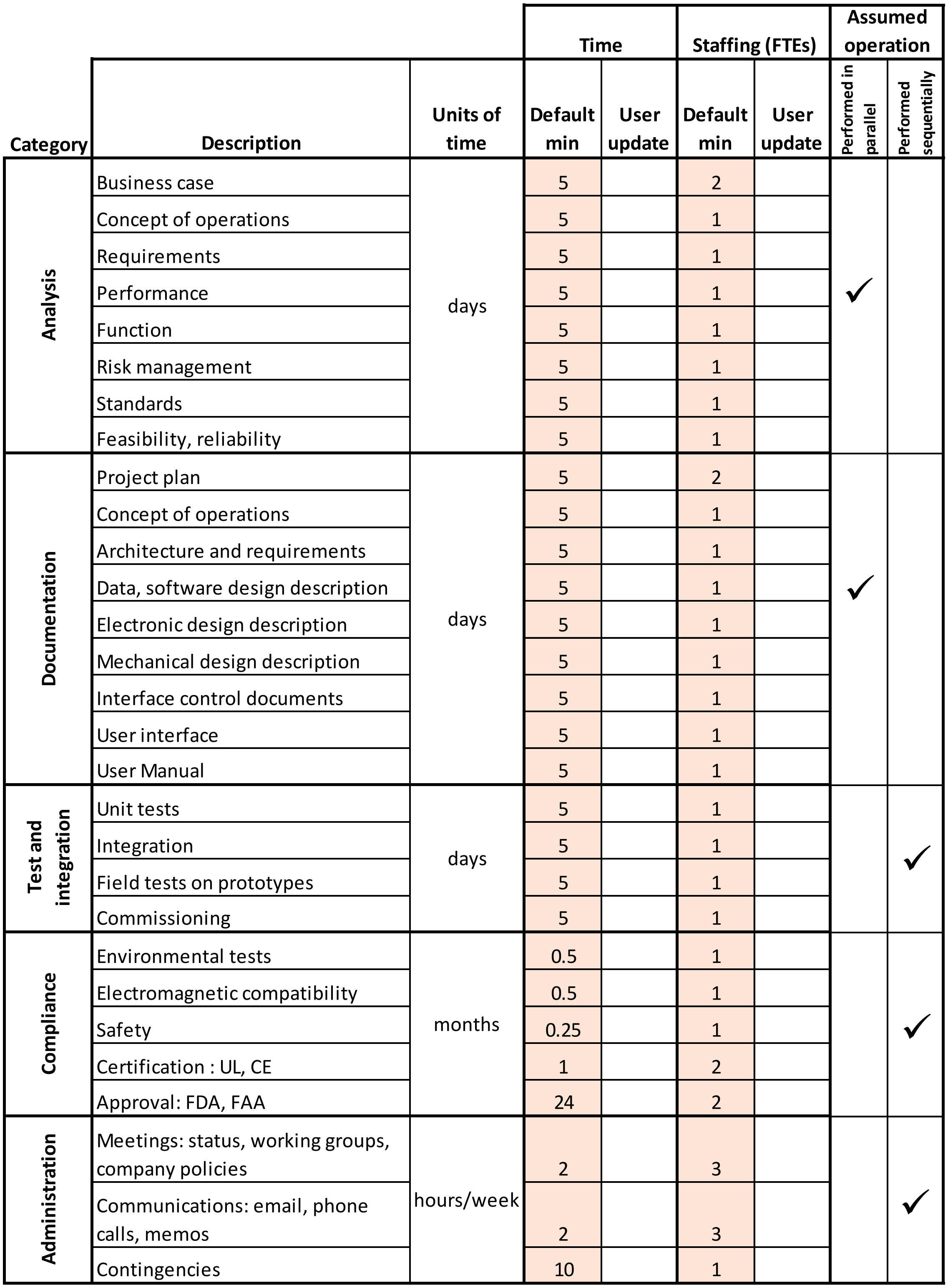
Select one choice below that you believe most closely represents the **best-case, absolute minimum** values for a clean-sheet design of case study 2.

Choice 1		Choice 2		Choice 3		Choice 4		Choice 5	
Time (days)	Effort (person-days)	Time (days)	Effort (person-days)	Time (days)	Effort (person-days)	Time (days)	Effort (person-days)	Time (days)	Effort (person-days)
4	6.7	6	10	7.7	9.5	12	16	18	22

Figure 8.32 Potential page for checking the estimate for the time and effort to develop the enclosure in Case Study 3.

Auxiliary Activities Baseline Study

Besides direct design activities, there are many auxiliary activities associated with project development. Figure 4.9, repeated below, illustrates a potential webpage that might capture many of these activities. Surveys of experienced, embedded-system developers in different industries can provide a basis for acceptable default values in Figure 4.9.



			Time		Staffing (FTEs)		Assumed operation	
Category	Description	Units of time	Default min	User update	Default min	User update	Performed in parallel	Performed sequentially
Analysis	Business case	days	5		2		✓	
	Concept of operations		5		1			
	Requirements		5		1			
	Performance		5		1			
	Function		5		1			
	Risk management		5		1			
	Standards		5		1			
	Feasibility, reliability		5		1			
Documentation	Project plan	days	5		2		✓	
	Concept of operations		5		1			
	Architecture and requirements		5		1			
	Data, software design description		5		1			
	Electronic design description		5		1			
	Mechanical design description		5		1			
	Interface control documents		5		1			
	User interface		5		1			
	User Manual		5		1			
Test and integration	Unit tests	days	5		1			✓
	Integration		5		1			
	Field tests on prototypes		5		1			
	Commissioning		5		1			
Compliance	Environmental tests	months	0.5		1			✓
	Electromagnetic compatibility		0.5		1			
	Safety		0.25		1			
	Certification : UL, CE		1		2			
	Approval: FDA, FAA		24		2			
Administration	Meetings: status, working groups, company policies	hours/week	2		3			✓
	Communications: email, phone calls, memos		2		3			
	Contingencies		10		1			

(Figure 4.9 repeated for reader's convenience) Potential auxiliary activities during development; surveying developers of embedded systems can set the orange default values.

Pilot Studies for Measuring the Speed of the Estimator Screens

The estimator tool in Chapter 4 needs two pilot studies of its operation to measure how long it takes to run the tool. The first study is for the sanity checker. The second is for the other screens within the estimator.

The sanity checker has five screens that need to be set up to indicate the number of purchased subsystems, the total number of subsystems to be developed, the industry, and the default values of specific activities. Following these five screens are five more screens of default values of the best case, absolute minimums for designing and developing sections of the embedded system; each user should view them and either confirm them acceptable or change values to increase the defaults for a specific project.

Next the estimator needs to be run and studied for speed and ease of operation. There are four opening screens that initiate operation of the estimator for each, new embedded project. The study will measure the time to complete the sequence of all four screens. Then depending on the level chosen, the study will measure the time to fill in values of time and FTEs.

All recorded times and observations, from both the sanity checker and the estimating matrices will be combined to determine the mean time and standard deviation to operate the entire tool. The goal will be for the tool to take less than 30 minutes for the first-time user.

Appendix F - Build-versus-Buy Simulation Results

Overview of Simulation Operations

I programmed two primary routines. The first routine was a Matlab m-file script and it provided the basic input values for all 57 parameters; it had the general format for its function call of `init_param_[case study name]`. The second routine was a Matlab m-file script and it used the 57 parameters in a Monte Carlo simulation that usually ran for a million iterations.

I used an Excel spreadsheet file to set and record most of the parameters before feeding them into the first routine. These Excel files follow.

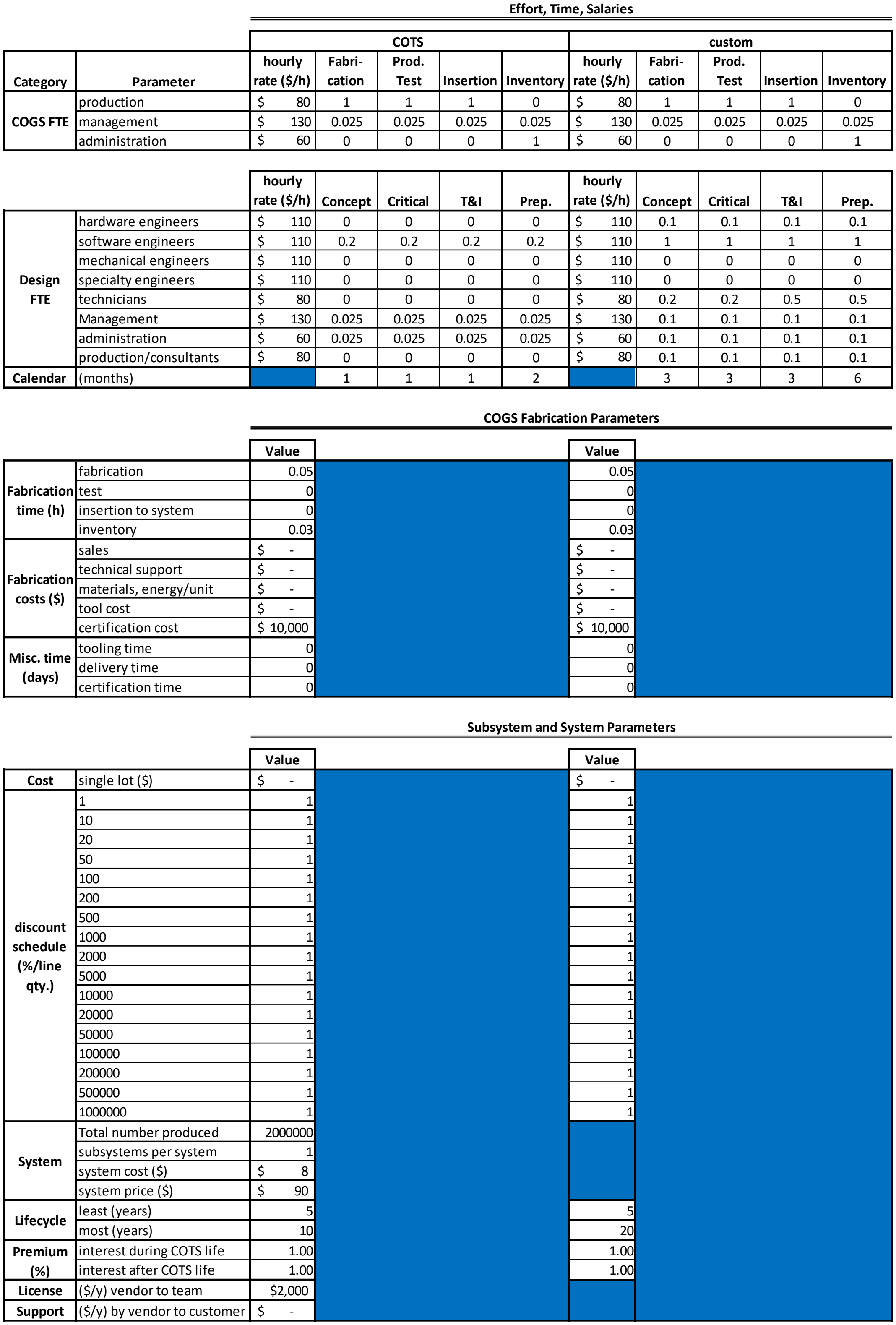
The second routine then would print out values and plot several distributions. It also generated variables and placed them in the workspace for other routines to analyze or plot specific values or distributions. An example of the output follows the parameter files.

Finally, I have printed the source code for the multiple different routines that run the simulation and display the results.

Parameter Inputs to the Simulations

(See following pages.)

Case Study 1: Consumer Appliance RTOS



Effort, Time, Salaries

		COTS					custom				
Category	Parameter	hourly rate ($/h)	Fabrication	Prod. Test	Insertion	Inventory	hourly rate ($/h)	Fabrication	Prod. Test	Insertion	Inventory
COGS FTE	production	$ 80	1	1	1	0	$ 80	1	1	1	0
	management	$ 130	0.025	0.025	0.025	0.025	$ 130	0.025	0.025	0.025	0.025
	administration	$ 60	0	0	0	1	$ 60	0	0	0	1
		hourly rate ($/h)	Concept	Critical	T&I	Prep.	hourly rate ($/h)	Concept	Critical	T&I	Prep.
Design FTE	hardware engineers	$ 110	0	0	0	0	$ 110	0.1	0.1	0.1	0.1
	software engineers	$ 110	0.2	0.2	0.2	0.2	$ 110	1	1	1	1
	mechanical engineers	$ 110	0	0	0	0	$ 110	0	0	0	0
	specialty engineers	$ 110	0	0	0	0	$ 110	0	0	0	0
	technicians	$ 80	0	0	0	0	$ 80	0.2	0.2	0.5	0.5
	Management	$ 130	0.025	0.025	0.025	0.025	$ 130	0.1	0.1	0.1	0.1
	administration	$ 60	0.025	0.025	0.025	0.025	$ 60	0.1	0.1	0.1	0.1
	production/consultants	$ 80	0	0	0	0	$ 80	0.1	0.1	0.1	0.1
Calendar	(months)		1	1	1	2		3	3	3	6

COGS Fabrication Parameters

		Value	Value
Fabrication time (h)	fabrication	0.05	0.05
	test	0	0
	insertion to system	0	0
	inventory	0.03	0.03
Fabrication costs ($)	sales	$ -	$ -
	technical support	$ -	$ -
	materials, energy/unit	$ -	$ -
	tool cost	$ -	$ -
	certification cost	$ 10,000	$ 10,000
Misc. time (days)	tooling time	0	0
	delivery time	0	0
	certification time	0	0

Subsystem and System Parameters

		Value	Value
Cost	single lot ($)	$ -	$ -
discount schedule (%/line qty.)	1	1	1
	10	1	1
	20	1	1
	50	1	1
	100	1	1
	200	1	1
	500	1	1
	1000	1	1
	2000	1	1
	5000	1	1
	10000	1	1
	20000	1	1
	50000	1	1
	100000	1	1
	200000	1	1
	500000	1	1
	1000000	1	1
System	Total number produced	2000000	
	subsystems per system	1	
	system cost ($)	$ 8	
	system price ($)	$ 90	
Lifecycle	least (years)	5	5
	most (years)	10	20
Premium (%)	interest during COTS life	1.00	1.00
	interest after COTS life	1.00	1.00
License	($/y) vendor to team	$2,000	
Support	($/y) by vendor to customer	$ -	

Case Study 2: Laboratory Instrument RTOS

Effort, Time, Salaries

		COTS					custom				
Category	Parameter	hourly rate ($/h)	Fabri-cation	Prod. Test	Insertion	Inventory	hourly rate ($/h)	Fabri-cation	Prod. Test	Insertion	Inventory
COGS FTE	production	$ 80	1	1	1	0	$ 80	1	1	1	0
	management	$ 130	0.025	0.025	0.025	0.025	$ 130	0.025	0.025	0.025	0.025
	administration	$ 60	0	0	0	1	$ 60	0	0	0	1

		hourly rate ($/h)	Concept	Critical	T&I	Prep.	hourly rate ($/h)	Concept	Critical	T&I	Prep.
Design FTE	hardware engineers	$ 110	0	0	0	0	$ 110	0.1	0.1	0.1	0.1
	software engineers	$ 110	0.2	0.2	0.2	0.2	$ 110	1	1	1	1
	mechanical engineers	$ 110	0	0	0	0	$ 110	0	0	0	0
	specialty engineers	$ 110	0	0	0	0	$ 110	0	0	0	0
	technicians	$ 80	0	0	0	0	$ 80	0.2	0.2	0.5	0.5
	Management	$ 130	0.025	0.025	0.025	0.025	$ 130	0.1	0.1	0.1	0.1
	administration	$ 60	0.025	0.025	0.025	0.025	$ 60	0.1	0.1	0.1	0.1
	production/consultants	$ 80	0	0	0	0	$ 80	0.1	0.1	0.1	0.1
Calendar	(months)		1	1	2	4		3	3	6	8

COGS Fabrication Parameters

		Value	Value
Fabrication time (h)	fabrication	0.05	0.05
	test	0	0
	insertion to system	0	0
	inventory	0.1	0.1
Fabrication costs ($)	sales	$ -	$ -
	technical support	$ -	$ -
	materials, energy/unit	$ -	$ -
	tool cost	$ -	$ -
	certification cost	$ 10,000	$ 10,000
Misc. time (days)	tooling time	0	0
	delivery time	0	0
	certification time	0	0

Subsystem and System Parameters

		Value	Value
Cost	single lot ($)	$20	$ -
discount schedule (%/line qty.)	1	1	1
	10	1	1
	20	1	1
	50	1	1
	100	1	1
	200	1	1
	500	1	1
	1000	1	1
	2000	1	1
	5000	1	1
	10000	1	1
	20000	1	1
	50000	1	1
	100000	1	1
	200000	1	1
	500000	1	1
	1000000	1	1
System	Total number produced	20000	
	subsystems per system	1	
	system cost ($)	$ 1,000	
	system price ($)	$ 1,500	
Lifecycle	least (years)	5	5
	most (years)	10	20
Premium (%)	interest during COTS life	1.00	1.00
	interest after COTS life	1.00	1.00
License	($/y) vendor to team	$2,000	
Support	($/y) by vendor to customer	$ -	

Case Study 3: Medical Device RTOS

Effort, Time, Salaries

		COTS					custom				
Category	Parameter	hourly rate ($/h)	Fabri-cation	Prod. Test	Insertion	Inventory	hourly rate ($/h)	Fabri-cation	Prod. Test	Insertion	Inventory
COGS FTE	production	$ 80	1	1	1	0	$ 80	1	1	1	0
	management	$ 130	0.025	0.025	0.025	0.025	$ 130	0.025	0.025	0.025	0.025
	administration	$ 60	0	0	0	1	$ 60	0	0	0	1

		hourly rate ($/h)	Concept	Critical	T&I	Prep.	hourly rate ($/h)	Concept	Critical	T&I	Prep.
Design FTE	hardware engineers	$ 110	0	0	0	0	$ 110	0.1	0.1	0.1	0.1
	software engineers	$ 110	1	1	1	1	$ 110	1.5	3	2	1
	mechanical engineers	$ 110	0	0	0	0	$ 110	0	0	0	0
	specialty engineers	$ 110	0.5	0.5	0.5	0.5	$ 110	0.5	0.5	0.5	0.5
	technicians	$ 80	0	0	0	0	$ 80	0.2	0.2	0.5	0.5
	Management	$ 130	0.1	0.1	0.1	0.1	$ 130	0.1	0.1	0.1	0.1
	administration	$ 60	0.1	0.1	0.1	0.1	$ 60	0.1	0.1	0.1	0.1
	production/consultants	$ 80	0	0	0	0	$ 80	0.1	0.1	0.1	0.1
Calendar	(months)		3	3	3	6		6	12	12	18

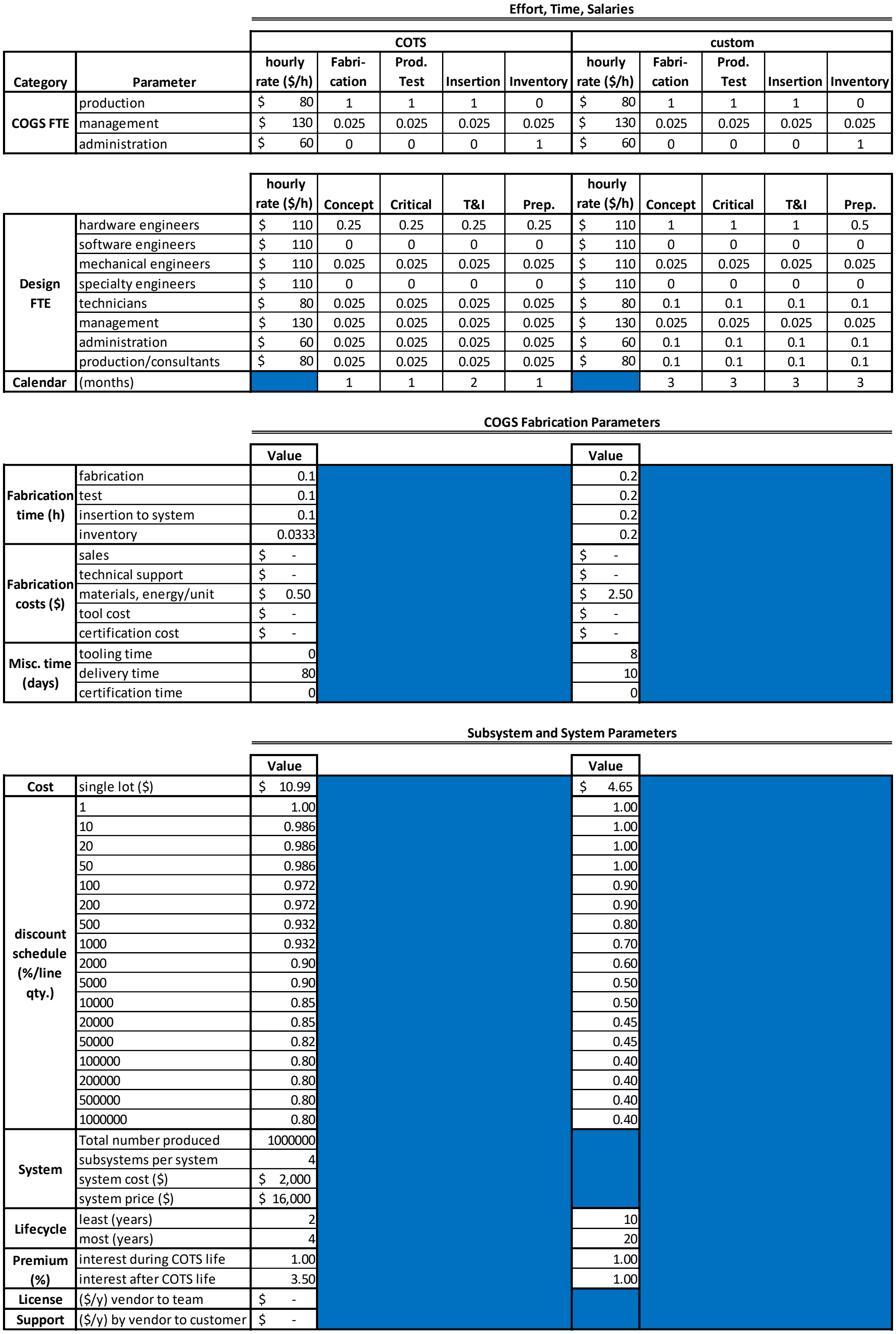
COGS Fabrication Parameters

		Value	Value
Fabrication time (h)	fabrication	0.05	0.05
	test	0	0
	insertion to system	0	0
	inventory	0.5	0.5
Fabrication costs ($)	sales	$ -	$ -
	technical support	$ -	$ -
	materials, energy/unit	$ -	$ -
	tool cost	$ -	$ -
	certification cost	$ 30,000	$ 30,000
Misc. time (days)	tooling time	0	0
	delivery time	0	0
	certification time	0	0

Subsystem and System Parameters

		Value	Value
Cost	single lot ($)	$500	$ -
discount schedule (%/line qty.)	1	1	1
	10	1	1
	20	1	1
	50	1	1
	100	1	1
	200	1	1
	500	1	1
	1000	1	1
	2000	1	1
	5000	1	1
	10000	1	1
	20000	1	1
	50000	1	1
	100000	1	1
	200000	1	1
	500000	1	1
	1000000	1	1
System	Total number produced	5000	
	subsystems per system	1	
	system cost ($)	$ 3,000	
	system price ($)	$ 30,000	
Lifecycle	least (years)	5	5
	most (years)	10	20
Premium (%)	interest during COTS life	1.00	1.00
	interest after COTS life	1.00	1.00
License	($/y) vendor to team	$10,000	
Support	($/y) by vendor to customer	$ -	

Case Study 4: Point-of-Load Converter (POL)



Effort, Time, Salaries

		COTS					custom				
Category	Parameter	hourly rate ($/h)	Fabri-cation	Prod. Test	Insertion	Inventory	hourly rate ($/h)	Fabri-cation	Prod. Test	Insertion	Inventory
COGS FTE	production	$ 80	1	1	1	0	$ 80	1	1	1	0
	management	$ 130	0.025	0.025	0.025	0.025	$ 130	0.025	0.025	0.025	0.025
	administration	$ 60	0	0	0	1	$ 60	0	0	0	1

		hourly rate ($/h)	Concept	Critical	T&I	Prep.	hourly rate ($/h)	Concept	Critical	T&I	Prep.
Design FTE	hardware engineers	$ 110	0.25	0.25	0.25	0.25	$ 110	1	1	1	0.5
	software engineers	$ 110	0	0	0	0	$ 110	0	0	0	0
	mechanical engineers	$ 110	0.025	0.025	0.025	0.025	$ 110	0.025	0.025	0.025	0.025
	specialty engineers	$ 110	0	0	0	0	$ 110	0	0	0	0
	technicians	$ 80	0.025	0.025	0.025	0.025	$ 80	0.1	0.1	0.1	0.1
	management	$ 130	0.025	0.025	0.025	0.025	$ 130	0.025	0.025	0.025	0.025
	administration	$ 60	0.025	0.025	0.025	0.025	$ 60	0.1	0.1	0.1	0.1
	production/consultants	$ 80	0.025	0.025	0.025	0.025	$ 80	0.1	0.1	0.1	0.1
Calendar	(months)		1	1	2	1		3	3	3	3

COGS Fabrication Parameters

		Value	Value
Fabrication time (h)	fabrication	0.1	0.2
	test	0.1	0.2
	insertion to system	0.1	0.2
	inventory	0.0333	0.2
Fabrication costs ($)	sales	$ -	$ -
	technical support	$ -	$ -
	materials, energy/unit	$ 0.50	$ 2.50
	tool cost	$ -	$ -
	certification cost	$ -	$ -
Misc. time (days)	tooling time	0	8
	delivery time	80	10
	certification time	0	0

Subsystem and System Parameters

		Value	Value
Cost	single lot ($)	$ 10.99	$ 4.65
discount schedule (%/line qty.)	1	1.00	1.00
	10	0.986	1.00
	20	0.986	1.00
	50	0.986	1.00
	100	0.972	0.90
	200	0.972	0.90
	500	0.932	0.80
	1000	0.932	0.70
	2000	0.90	0.60
	5000	0.90	0.50
	10000	0.85	0.50
	20000	0.85	0.45
	50000	0.82	0.45
	100000	0.80	0.40
	200000	0.80	0.40
	500000	0.80	0.40
	1000000	0.80	0.40
System	Total number produced	1000000	
	subsystems per system	4	
	system cost ($)	$ 2,000	
	system price ($)	$ 16,000	
Lifecycle	least (years)	2	10
	most (years)	4	20
Premium (%)	interest during COTS life	1.00	1.00
	interest after COTS life	3.50	1.00
License	($/y) vendor to team	$ -	
Support	($/y) by vendor to customer	$ -	

Notes for the POL parameters:

https://www.digikey.com/product-detail/en/OKI-T-36W-W40P-C/811-3132-1-ND/ . . . . .

Murata Power Solutions, POL, 36W, 5-15V $ 10.99

https://power.sager.com/oki-t-36w-w40p-c-2493551.html?

1	$11.79	**100%**
10	$11.62	**0.985581**
100	$11.46	**0.97201**
500	$10.99	**0.932146**

http://www.futureelectronics.com/en/Technologies/Product.aspx? . . . .

ProductID=OKIT36WW40PCMURATAPOWERSOLUTIONS3096603&IM=0

250+ $9.95

inductor =	$	0.10
control IC =	$	1.75
discretes =	$	0.30
PCB =	$	2.50
	$	4.65

Case Study 5: NSS Touchscreen Using Rugged Off-the-Shelf (ROTS)

Effort, Time, Salaries

		ROTS					custom				
Category	Parameter	hourly rate ($/h)	Fabri-cation	Prod. Test	Assembly	Inventory	hourly rate ($/h)	Fabri-cation	Prod. Test	Assembly	Inventory
COGS FTE	production	$ 80	3	1	1	0	$ 80	3	2	1	0
	management	$ 130	0.05	0.05	0.05	0.05	$ 130	0.05	0.05	0.05	0.05
	administration	$ 60	0.1	0.1	0.1	1	$ 60	0.1	0.1	0.1	1

		hourly rate ($/h)	Concept	Critical	T&I	Prep.	hourly rate ($/h)	Concept	Critical	T&I	Prep.
Design FTE	hardware engineers	$ 110	0.5	0.5	0.5	0.5	$ 110	1	1	1	1
	software engineers	$ 110	1	3	3	3	$ 110	2	4	4	4
	mechanical engineers	$ 110	0.5	0.5	0.5	0.5	$ 110	1	1	1	1
	specialty engineers	$ 110	0.5	0.5	0.5	0.5	$ 110	0.5	0.5	0.5	0.5
	technicians	$ 80	1	1	2	2	$ 80	1	1	2	2
	Management	$ 130	0.5	0.5	0.5	0.5	$ 130	0.5	0.5	0.5	0.5
	administration	$ 60	1	1	1	1	$ 60	1	1	1	1
	production/consultants	$ 80	1	1	1	1	$ 80	1	1	1	1
Calendar	(months)		6	12	12	12		12	12	12	12

COGS Fabrication Parameters

		Value	Value
Fabrication time (h)	fabrication	0.2	0.4
	test	0.3	0.4
	assembly to system	0.05	0.1
	inventory	0.5	0.5
Fabrication costs ($)	sales	$ -	$ -
	technical support	$ -	$ -
	materials, energy/unit	$ -	$ -
	tool cost	$ -	$ -
	certification cost	$ 30,000	$ 30,000
Misc. time (days)	tooling time	5	5
	delivery time	14	14
	certification time	42	42

Subsystem and System Parameters

		Value	Value
Cost	single lot ($)	$2,530	$ 230.00
discount schedule (%/line qty.)	1	1	1.00
	10	1	1.00
	20	1	1.00
	50	0.98	1.00
	100	0.95	0.90
	200	0.95	0.90
	500	0.9	0.80
	1000	0.9	0.70
	2000	0.88	0.60
	5000	0.85	0.50
	10000	0.83	0.50
	20000	0.8	0.45
	50000	0.8	0.45
	100000	0.8	0.40
	200000	0.8	0.40
	500000	0.8	0.40
	1000000	0.8	0.40
System	Total number produced	10000	
	subsystems per system	1	
	system cost ($)	$ 2,700	$ 600
	system price ($)	$ 5,400	$ 1,200
Lifecycle	least (years)	5	5
	most (years)	10	10
Premium (%)	interest during COTS life	1.00	1.00
	interest after COTS life	1.00	1.00
License	($/y) vendor to team	$ -	
Support	($/y) by vendor to customer	$ -	

Notes for the ROTS Touchscreen parameters:

COTS uses ROTS + uC + xcvr

custom uses touchscreen with computer + xcvr + batteries + protective case

neither includes the USB antenna, recharge station and cables, nor misc. manuals in calculations

assumes contract manufacturing according to GMP and FDA guidelines

assumes some of the burden for FDA approval, primarily in the calendar times

assumes that price does not include revenue generated from data mining

Case Study 6: NSS Touchscreen Using iPad

Effort, Time, Salaries

		COTS					custom				
Category	**Parameter**	**hourly rate ($/h)**	**Fabri-cation**	**Prod. Test**	**Assembly**	**Inventory**	**hourly rate ($/h)**	**Fabri-cation**	**Prod. Test**	**Assembly**	**Inventory**
COGS FTE	production	$ 80	3	1	1	0	$ 80	3	2	1	0
	management	$ 130	0.05	0.05	0.05	0.05	$ 130	0.05	0.05	0.05	0.05
	administration	$ 60	0.1	0.1	0.1	1	$ 60	0.1	0.1	0.1	1

		hourly rate ($/h)	**Concept**	**Critical**	**T&I**	**Prep.**	**hourly rate ($/h)**	**Concept**	**Critical**	**T&I**	**Prep.**
Design FTE	hardware engineers	$ 110	0.5	0.5	0.5	0.5	$ 110	1	1	1	1
	software engineers	$ 110	1	3	3	3	$ 110	2	4	4	4
	mechanical engineers	$ 110	0.5	0.5	0.5	0.5	$ 110	1	1	1	1
	specialty engineers	$ 110	0.5	0.5	0.5	0.5	$ 110	0.5	0.5	0.5	0.5
	technicians	$ 80	1	1	2	2	$ 80	1	1	2	2
	Management	$ 130	0.5	0.5	0.5	0.5	$ 130	0.5	0.5	0.5	0.5
	administration	$ 60	1	1	1	1	$ 60	1	1	1	1
	production/consultants	$ 80	1	1	1	1	$ 80	1	1	1	1
Calendar	(months)		6	12	12	12		12	12	12	12

COGS Fabrication Parameters

		Value	**Value**
Fabrication time (h)	fabrication	0.2	0.4
	test	0.3	0.4
	assembly to system	0.05	0.1
	inventory	0.5	0.5
Fabrication costs ($)	sales	$ -	$ -
	technical support	$ -	$ -
	materials, energy/unit	$ -	$ -
	tool cost	$ -	$ -
	certification cost	$ 30,000	$ 30,000
Misc. time (days)	tooling time	5	5
	delivery time	14	14
	certification time	42	42

Subsystem and System Parameters

		Value	**Value**
Cost	single lot ($)	$430	$ 230.00
discount schedule (%/line qty.)	1	1	1.00
	10	1	1.00
	20	1	1.00
	50	0.98	1.00
	100	0.95	0.90
	200	0.95	0.90
	500	0.9	0.80
	1000	0.9	0.70
	2000	0.88	0.60
	5000	0.85	0.50
	10000	0.83	0.50
	20000	0.8	0.45
	50000	0.8	0.45
	100000	0.8	0.40
	200000	0.8	0.40
	500000	0.8	0.40
	1000000	0.8	0.40
System	Total number produced	10000	
	subsystems per system	1	
	system cost ($)	$ 600	
	system price ($)	$ 1,200	
Lifecycle	least (years)	0.4	5
	most (years)	1	10
Premium (%)	interest during COTS life	1.00	1.00
	interest after COTS life	2.00	1.00
License	($/y) vendor to team	$ -	
Support	($/y) by vendor to customer	$ -	

Notes for the iPad Touchscreen parameters:

COTS uses iPad + uC + xcvr

custom uses touchscreen with computer + xcvr + batteries + protective case

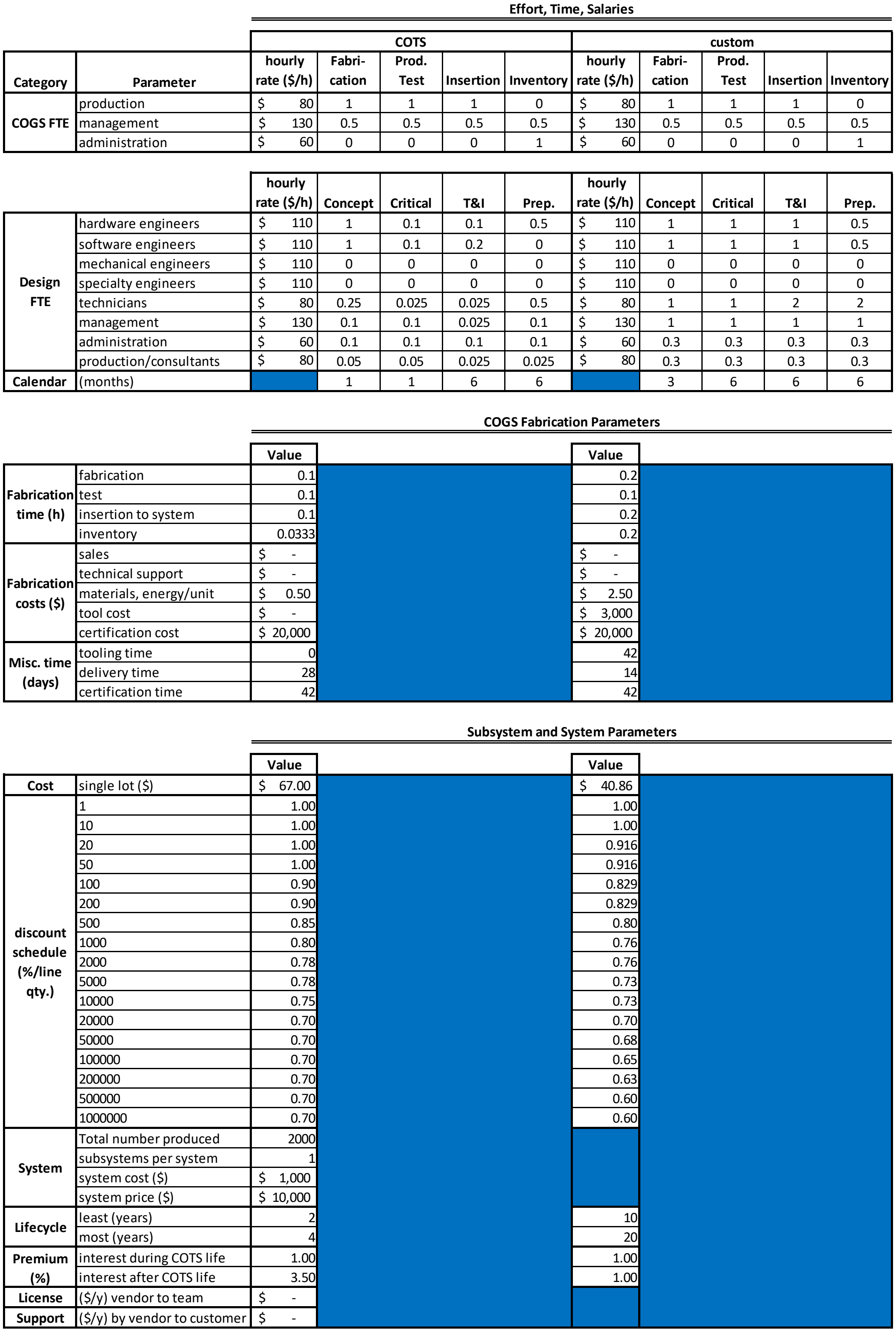
neither includes the USB antenna, recharge station and cables, nor misc. manuals in calculations

assumes contract manufacturing according to GMP and FDA guidelines

assumes some of the burden for FDA approval, primarily in the calendar times

assumes that price does not include revenue generated from data mining

Case Study 7: Arc-Fault Detector (AFD) Using a PIC Microcontroller



Effort, Time, Salaries

		COTS					custom				
Category	Parameter	hourly rate ($/h)	Fabrication	Prod. Test	Insertion	Inventory	hourly rate ($/h)	Fabrication	Prod. Test	Insertion	Inventory
COGS FTE	production	$ 80	1	1	1	0	$ 80	1	1	1	0
	management	$ 130	0.5	0.5	0.5	0.5	$ 130	0.5	0.5	0.5	0.5
	administration	$ 60	0	0	0	1	$ 60	0	0	0	1

		hourly rate ($/h)	Concept	Critical	T&I	Prep.	hourly rate ($/h)	Concept	Critical	T&I	Prep.
Design FTE	hardware engineers	$ 110	1	0.1	0.1	0.5	$ 110	1	1	1	0.5
	software engineers	$ 110	1	0.1	0.2	0	$ 110	1	1	1	0.5
	mechanical engineers	$ 110	0	0	0	0	$ 110	0	0	0	0
	specialty engineers	$ 110	0	0	0	0	$ 110	0	0	0	0
	technicians	$ 80	0.25	0.025	0.025	0.5	$ 80	1	1	2	2
	management	$ 130	0.1	0.1	0.025	0.1	$ 130	1	1	1	1
	administration	$ 60	0.1	0.1	0.1	0.1	$ 60	0.3	0.3	0.3	0.3
	production/consultants	$ 80	0.05	0.05	0.025	0.025	$ 80	0.3	0.3	0.3	0.3
Calendar	(months)		1	1	6	6		3	6	6	6

COGS Fabrication Parameters

		Value	Value
Fabrication time (h)	fabrication	0.1	0.2
	test	0.1	0.1
	insertion to system	0.1	0.2
	inventory	0.0333	0.2
Fabrication costs ($)	sales	$ -	$ -
	technical support	$ -	$ -
	materials, energy/unit	$ 0.50	$ 2.50
	tool cost	$ -	$ 3,000
	certification cost	$ 20,000	$ 20,000
Misc. time (days)	tooling time	0	42
	delivery time	28	14
	certification time	42	42

Subsystem and System Parameters

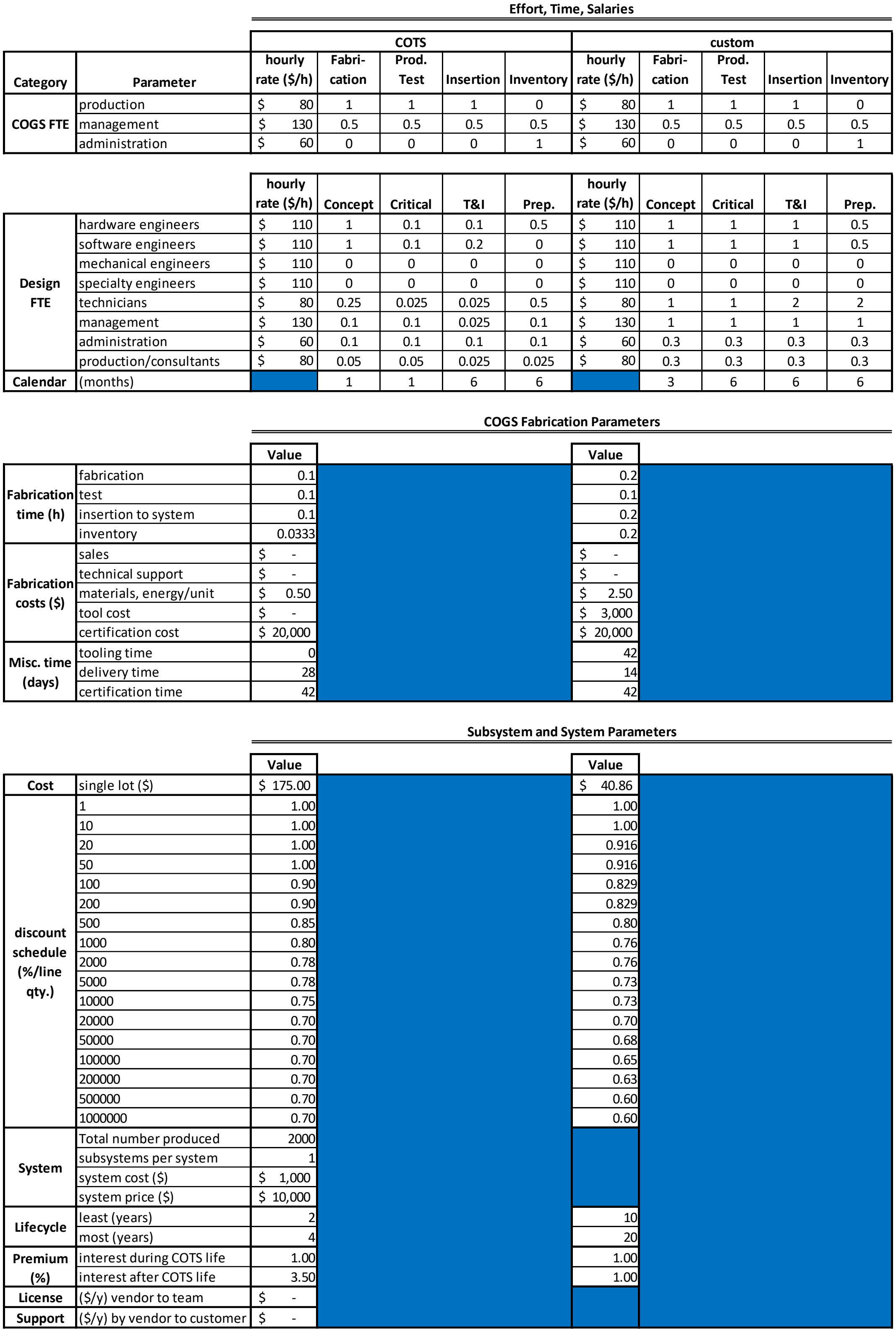
		Value	Value
Cost	single lot ($)	$ 67.00	$ 40.86
discount schedule (%/line qty.)	1	1.00	1.00
	10	1.00	1.00
	20	1.00	0.916
	50	1.00	0.916
	100	0.90	0.829
	200	0.90	0.829
	500	0.85	0.80
	1000	0.80	0.76
	2000	0.78	0.76
	5000	0.78	0.73
	10000	0.75	0.73
	20000	0.70	0.70
	50000	0.70	0.68
	100000	0.70	0.65
	200000	0.70	0.63
	500000	0.70	0.60
	1000000	0.70	0.60
System	Total number produced	2000	
	subsystems per system	1	
	system cost ($)	$ 1,000	
	system price ($)	$ 10,000	
Lifecycle	least (years)	2	10
	most (years)	4	20
Premium (%)	interest during COTS life	1.00	1.00
	interest after COTS life	3.50	1.00
License	($/y) vendor to team	$ -	
Support	($/y) by vendor to customer	$ -	

Notes for the AFD using a PIC Microcontroller:

http://microcontrollershop.com/product_info.php?cPath=346_0_112_160_204&products_id= . . .

sPIC33FJ128GP802 Board, XBee socket, RS485, I2C	$42.00
Xbee transceiver @ 900 MHz + encryption	$17.00
900 MHz Sparkfun transceiver	$10.00

Case Study 8: AFD Using an ARM Microcontroller



Effort, Time, Salaries

		COTS					custom				
Category	**Parameter**	**hourly rate ($/h)**	**Fabri-cation**	**Prod. Test**	**Insertion**	**Inventory**	**hourly rate ($/h)**	**Fabri-cation**	**Prod. Test**	**Insertion**	**Inventory**
COGS FTE	production	$ 80	1	1	1	0	$ 80	1	1	1	0
	management	$ 130	0.5	0.5	0.5	0.5	$ 130	0.5	0.5	0.5	0.5
	administration	$ 60	0	0	0	1	$ 60	0	0	0	1
		hourly rate ($/h)	**Concept**	**Critical**	**T&I**	**Prep.**	**hourly rate ($/h)**	**Concept**	**Critical**	**T&I**	**Prep.**
Design FTE	hardware engineers	$ 110	1	0.1	0.1	0.5	$ 110	1	1	1	0.5
	software engineers	$ 110	1	0.1	0.2	0	$ 110	1	1	1	0.5
	mechanical engineers	$ 110	0	0	0	0	$ 110	0	0	0	0
	specialty engineers	$ 110	0	0	0	0	$ 110	0	0	0	0
	technicians	$ 80	0.25	0.025	0.025	0.5	$ 80	1	1	2	2
	management	$ 130	0.1	0.1	0.025	0.1	$ 130	1	1	1	1
	administration	$ 60	0.1	0.1	0.1	0.1	$ 60	0.3	0.3	0.3	0.3
	production/consultants	$ 80	0.05	0.05	0.025	0.025	$ 80	0.3	0.3	0.3	0.3
Calendar	(months)		1	1	6	6		3	6	6	6

COGS Fabrication Parameters

		Value	**Value**
Fabrication time (h)	fabrication	0.1	0.2
	test	0.1	0.1
	insertion to system	0.1	0.2
	inventory	0.0333	0.2
Fabrication costs ($)	sales	$ -	$ -
	technical support	$ -	$ -
	materials, energy/unit	$ 0.50	$ 2.50
	tool cost	$ -	$ 3,000
	certification cost	$ 20,000	$ 20,000
Misc. time (days)	tooling time	0	42
	delivery time	28	14
	certification time	42	42

Subsystem and System Parameters

		Value	**Value**
Cost	single lot ($)	$ 175.00	$ 40.86
discount schedule (%/line qty.)	1	1.00	1.00
	10	1.00	1.00
	20	1.00	0.916
	50	1.00	0.916
	100	0.90	0.829
	200	0.90	0.829
	500	0.85	0.80
	1000	0.80	0.76
	2000	0.78	0.76
	5000	0.78	0.73
	10000	0.75	0.73
	20000	0.70	0.70
	50000	0.70	0.68
	100000	0.70	0.65
	200000	0.70	0.63
	500000	0.70	0.60
	1000000	0.70	0.60
System	Total number produced	2000	
	subsystems per system	1	
	system cost ($)	$ 1,000	
	system price ($)	$ 10,000	
Lifecycle	least (years)	2	10
	most (years)	4	20
Premium (%)	interest during COTS life	1.00	1.00
	interest after COTS life	3.50	1.00
License	($/y) vendor to team	$ -	
Support	($/y) by vendor to customer	$ -	

Notes for the AFD using an ARM Microcontroller:

Xbee transceiver @ 900 MHz + encryption	$17.00
900 MHz Sparkfun transceiver	$10.00

Case Studies 9-12: AFD Using an Atom Processor

Effort, Time, Salaries

		COTS					custom				
Category	Parameter	hourly rate ($/h)	Fabri-cation	Prod. Test	Insertion	Inventory	hourly rate ($/h)	Fabri-cation	Prod. Test	Insertion	Inventory
COGS FTE	production	$ 80	1	1	1	0	$ 80	1	1	1	0
	management	$ 130	0.5	0.5	0.5	0.5	$ 130	0.5	0.5	0.5	0.5
	administration	$ 60	0	0	0	1	$ 60	0	0	0	1

		hourly rate ($/h)	Concept	Critical	T&I	Prep.	hourly rate ($/h)	Concept	Critical	T&I	Prep.
Design FTE	hardware engineers	$ 110	1	0.1	0.1	0.5	$ 110	1	1	1	0.5
	software engineers	$ 110	1	0.1	0.2	0	$ 110	1	1	1	0.5
	mechanical engineers	$ 110	0	0	0	0	$ 110	0	0	0	0
	specialty engineers	$ 110	0	0	0	0	$ 110	0	0	0	0
	technicians	$ 80	0.25	0.025	0.025	0.5	$ 80	1	1	2	2
	management	$ 130	0.1	0.1	0.025	0.1	$ 130	1	1	1	1
	administration	$ 60	0.1	0.1	0.1	0.1	$ 60	0.3	0.3	0.3	0.3
	production/consultants	$ 80	0.05	0.05	0.025	0.025	$ 80	0.3	0.3	0.3	0.3
Calendar	(months)		1	1	6	6		3	6	6	6

COGS Fabrication Parameters

		Value	Value
Fabrication time (h)	fabrication	0.1	0.2
	test	0.1	0.1
	insertion to system	0.1	0.2
	inventory	0.0333	0.2
Fabrication costs ($)	sales	$ -	$ -
	technical support	$ -	$ -
	materials, energy/unit	$ 0.50	$ 2.50
	tool cost	$ -	$ 3,000
	certification cost	$ 20,000	$ 20,000
Misc. time (days)	tooling time	0	42
	delivery time	28	14
	certification time	42	42

Subsystem and System Parameters

		Value	Value
Cost	single lot ($)	$ 1,700	$ 400
discount schedule (%/line qty.)	1	1.00	1.00
	10	1.00	1.00
	20	1.00	0.916
	50	1.00	0.916
	100	0.90	0.829
	200	0.90	0.829
	500	0.85	0.80
	1000	0.80	0.76
	2000	0.78	0.76
	5000	0.78	0.73
	10000	0.75	0.73
	20000	0.70	0.70
	50000	0.70	0.68
	100000	0.70	0.65
	200000	0.70	0.63
	500000	0.70	0.60
	1000000	0.70	0.60
System	Total number produced	2000	
	subsystems per system	1	
	system cost ($)	$ 1,000	
	system price ($)	$ 10,000	
Lifecycle	least (years)	2	10
	most (years)	4	20
Premium (%)	interest during COTS life	1.00	1.00
	interest after COTS life	3.50	1.00
License	($/y) vendor to team	$ -	
Support	($/y) by vendor to customer	$ -	

Case Study 13: Storage Grain Bin Controller

Effort, Time, Salaries

		COTS					custom				
		hourly	Fabri-	Prod.			hourly	Fabri-	Prod.		
Category	Parameter	rate ($/h)	cation	Test	Insertion	Inventory	rate ($/h)	cation	Test	Insertion	Inventory
COGS FTE	production	$ 80	1	1	1	0	$ 80	1.5	1	1	0
	management	$ 130	0.5	0.5	0.5	0.5	$ 130	0.5	0.5	0.5	0.5
	administration	$ 60	0	0	0	1	$ 60	0	0	0	1

		hourly					hourly				
		rate ($/h)	Concept	Critical	T&I	Prep.	rate ($/h)	Concept	Critical	T&I	Prep.
Design FTE	hardware engineers	$ 110	1	0.1	0.1	0.5	$ 110	1	1	1	0.5
	software engineers	$ 110	1	0.1	0.2	0	$ 110	1	1	1	0.5
	mechanical engineers	$ 110	0.1	0.5	0.2	0.1	$ 110	0.1	0.5	0.2	0.1
	specialty engineers	$ 110	0.1	0.5	0.2	0.1	$ 110	0.1	0.5	0.2	0.1
	technicians	$ 80	0.25	0.025	0.025	0.5	$ 80	1	1	2	2
	management	$ 130	0.1	0.1	0.025	0.1	$ 130	1	1	1	1
	administration	$ 60	0.1	0.1	0.1	0.1	$ 60	0.3	0.3	0.3	0.3
	production/consultants	$ 80	0.05	0.05	0.025	0.025	$ 80	0.3	0.3	0.3	0.3
Calendar	(months)		2	2	6	6		3	6	6	6

COGS Fabrication Parameters

		Value	Value
Fabrication time (h)	fabrication	0.1	0.3
	test	0.1	0.1
	insertion to system	0.1	0.1
	inventory	0.2	0.2
Fabrication costs ($)	sales	$ -	$ -
	technical support	$ -	$ -
	materials, energy/unit	$ 0.50	$ 2.50
	tool cost	$ -	$ 3,000
	certification cost	$ 20,000	$ 20,000
Misc. time (days)	tooling time	0	42
	delivery time	28	14
	certification time	42	42

Subsystem and System Parameters

		Value	Value
Cost	single lot ($)	$ 175.00	$ 40.86
discount schedule (%/line qty.)	1	1.00	1.00
	10	1.00	1.00
	20	1.00	0.916
	50	1.00	0.916
	100	0.90	0.829
	200	0.90	0.829
	500	0.85	0.80
	1000	0.80	0.76
	2000	0.78	0.76
	5000	0.78	0.73
	10000	0.75	0.73
	20000	0.70	0.70
	50000	0.70	0.68
	100000	0.70	0.65
	200000	0.70	0.63
	500000	0.70	0.60
	1000000	0.70	0.60
System	Total number produced	40000	
	subsystems per system	1	
	system cost ($)	$ 3,000	
	system price ($)	$ 15,000	
Lifecycle	least (years)	2	10
	most (years)	4	30
Premium (%)	interest during COTS life	1.00	1.00
	interest after COTS life	6.00	1.00
License	($/y) vendor to team	$ -	
Support	($/y) by vendor to customer	$ -	

Notes for the Storage Grain Bin Controller:

Xbee transceiver @ 900 MHz + encryption	$17.00
900 MHz Sparkfun transceiver	$10.00

Case Study 14: Volatile Organic Compound Sensors Controller

Effort, Time, Salaries

		COTS					custom				
Category	Parameter	hourly rate ($/h)	Fabri-cation	Prod. Test	Insertion	Inventory	hourly rate ($/h)	Fabri-cation	Prod. Test	Insertion	Inventory
COGS FTE	production	$ 80	1	1	1	0	$ 80	2	1	1	0
	management	$ 130	0.5	0.5	0.5	0.5	$ 130	0.5	0.5	0.5	0.5
	administration	$ 60	0	0	0	1	$ 60	0	0	0	1

		hourly rate ($/h)	Concept	Critical	T&I	Prep.	hourly rate ($/h)	Concept	Critical	T&I	Prep.
Design FTE	hardware engineers	$ 110	1	1	1	1	$ 110	1.5	1.5	1	1
	software engineers	$ 110	1	1	1	1	$ 110	2	2	2	1
	mechanical engineers	$ 110	0.5	0.5	0.5	0.5	$ 110	1	1	1	1
	specialty engineers	$ 110	0.5	0.5	0.5	0.5	$ 110	0.5	0.5	0.5	0.5
	technicians	$ 80	0.25	0.1	0.1	0.5	$ 80	1	1	2	2
	management	$ 130	0.5	0.5	0.5	0.5	$ 130	1	1	1	1
	administration	$ 60	0.5	0.5	0.5	0.5	$ 60	0.5	0.5	0.5	0.5
	production/consultants	$ 80	1	1	0.5	0.5	$ 80	1	1	0.5	0.5
Calendar	(months)		6	6	6	6		6	6	12	12

COGS Fabrication Parameters

		Value	Value
Fabrication time (h)	fabrication	0.1	0.3
	test	0.1	0.1
	assembly into system	0.1	0.1
	inventory	0.4	0.4
Fabrication costs ($)	sales	$ -	$ -
	technical support	$ -	$ -
	materials, energy/unit	$ 0.50	$ 2.50
	tool cost	$ -	$ 3,000
	certification cost	$ 40,000	$ 40,000
Misc. time (days)	tooling time	0	42
	delivery time	28	14
	certification time	42	42

Subsystem and System Parameters

		Value	Value
Cost	single lot ($)	$ 9,375	$ 704.86
discount schedule (%/line qty.)	1	1.00	1.00
	10	1.00	1.00
	20	1.00	0.916
	50	1.00	0.916
	100	0.90	0.829
	200	0.90	0.829
	500	0.85	0.80
	1000	0.80	0.76
	2000	0.78	0.76
	5000	0.78	0.73
	10000	0.75	0.73
	20000	0.70	0.70
	50000	0.70	0.68
	100000	0.70	0.65
	200000	0.70	0.63
	500000	0.70	0.60
	1000000	0.70	0.60
System	Total number produced	5000	
	subsystems per system	1	
	system cost ($)	$150,000	
	system price ($)	$250,000	
Lifecycle	least (years)	5	10
	most (years)	15	30
Premium (%)	interest during COTS life	1.00	1.00
	interest after COTS life	2.00	1.00
License	($/y) vendor to team	$ -	
Support	($/y) by vendor to customer	$ -	

Notes for the Volatile Organic Compound Sensors Controller:

This is an industrial controller for a VOC sensor system for refineries. It communicates with a base station.

Xbee transceiver @ 900 MHz + encryption

900 MHz Sparkfun transceiver

Custom circuit board and enclosure

processor and components = 5.86+25+15

PCB = $15

connectors = $4

contract fab & assembly cost of PCB = $40

Gas-tight enclosure = $400

cable glands = $200

Case Study 15: Single Processor Board

Effort, Time, Salaries

		COTS					custom				
Category	Parameter	hourly rate ($/h)	Fabri-cation	Prod. Test	Insertion	Inventory	hourly rate ($/h)	Fabri-cation	Prod. Test	Insertion	Inventory
COGS FTE	production	$ 80	1	1	1	0	$ 80	1	1	1	0
	management	$ 130	0.5	0.5	0.5	0.5	$ 130	0.5	0.5	0.5	0.5
	administration	$ 60	0	0	0	1	$ 60	0	0	0	1

		hourly rate ($/h)	Concept	Critical	T&I	Prep.	hourly rate ($/h)	Concept	Critical	T&I	Prep.
Design FTE	hardware engineers	$ 110	1	0.1	0.1	0.5	$ 110	1	1	1	0.25
	software engineers	$ 110	1	0.1	0.2	0	$ 110	2	2	2	1
	mechanical engineers	$ 110	0	0	0	0	$ 110	0	0	0	0
	specialty engineers	$ 110	0	0	0	0	$ 110	0	0	0	0
	technicians	$ 80	0.25	0.025	0.025	0.5	$ 80	1	1	2	2
	Management	$ 130	0.1	0.1	0.025	0.1	$ 130	1	1	1	1
	administration	$ 60	0.1	0.1	0.1	0.1	$ 60	0.5	0.5	0.5	0.5
	production/consultants	$ 80	0.05	0.05	0.025	0.025	$ 80	0.5	0.5	0.5	0.5
Calendar	(months)		1	1	2	2		6	12	6	6

COGS Fabrication Parameters

		Value	Value
Fabrication time (h)	fabrication	0.1	0.5
	test	0.1	0.3
	insertion to system	0.1	0.2
	inventory	0.0333	0.2
Fabrication costs ($)	sales	$ -	$ -
	technical support	$ -	$ -
	materials, energy/unit	$ 0.50	$ 2.50
	tool cost	$ -	$ 3,000
	certification cost	$ 20,000	$ -
Misc. time (days)	tooling time	0	5
	delivery time	28	14
	certification time	42	0

Subsystem and System Parameters

		Value	Value
Cost	single lot ($)	$ 6,000	$ 900
discount schedule (%/line qty.)	1	1.00	1.00
	10	0.95	1.00
	20	0.90	1.00
	50	0.85	1.00
	100	0.80	0.90
	200	0.70	0.90
	500	0.60	0.80
	1000	0.50	0.70
	2000	0.50	0.60
	5000	0.50	0.50
	10000	0.50	0.50
	20000	0.50	0.45
	50000	0.50	0.45
	100000	0.50	0.40
	200000	0.50	0.40
	500000	0.50	0.40
	1000000	0.50	0.40
System	Total number produced	10000	
	subsystems per system	1	
	system cost ($)	$ 10,000	
	system price ($)	$ 6,000	
Lifecycle	least (years)	2	4
	most (years)	4	8
Premium (%)	interest during COTS life	1.00	1.00
	interest after COTS life	1.00	1.00
License	($/y) vendor to team	$ 25,000	
Support	($/y) by vendor to customer	$ 25,000	

Case Study 16: Multi-Processor Board

Effort, Time, Salaries

		COTS					custom				
Category	**Parameter**	**hourly rate ($/h)**	**Fabri-cation**	**Prod. Test**	**Insertion**	**Inventory**	**hourly rate ($/h)**	**Fabri-cation**	**Prod. Test**	**Insertion**	**Inventory**
COGS FTE	production	$ 80	1	1	1	0	$ 80	1	1	1	0
	management	$ 130	0.5	0.5	0.5	0.5	$ 130	0.5	0.5	0.5	0.5
	administration	$ 60	0	0	0	1	$ 60	0	0	0	1

		hourly rate ($/h)	**Concept**	**Critical**	**T&I**	**Prep.**	**hourly rate ($/h)**	**Concept**	**Critical**	**T&I**	**Prep.**
Design FTE	hardware engineers	$ 110	1	0.1	0.1	0.5	$ 110	1	1	1	0.25
	software engineers	$ 110	1	0.1	0.2	0	$ 110	2	2	2	1
	mechanical engineers	$ 110	0	0	0	0	$ 110	0	0	0	0
	specialty engineers	$ 110	0	0	0	0	$ 110	0	0	0	0
	technicians	$ 80	0.25	0.025	0.025	0.5	$ 80	1	1	2	2
	Management	$ 130	0.1	0.1	0.025	0.1	$ 130	1	1	1	1
	administration	$ 60	0.1	0.1	0.1	0.1	$ 60	0.5	0.5	0.5	0.5
	production/consultants	$ 80	0.05	0.05	0.025	0.025	$ 80	0.5	0.5	0.5	0.5
Calendar	(months)		1	1	2	2		6	12	6	6

COGS Fabrication Parameters

		Value	**Value**
Fabrication time (h)	fabrication	0.1	0.5
	test	0.1	0.3
	insertion to system	0.1	0.2
	inventory	0.0333	0.2
Fabrication costs ($)	sales	$ -	$ -
	technical support	$ -	$ -
	materials, energy/unit	$ 0.50	$ 2.50
	tool cost	$ -	$ 3,000
	certification cost	$ 20,000	$ -
Misc. time (days)	tooling time	0	5
	delivery time	28	14
	certification time	42	0

Subsystem and System Parameters

		Value	**Value**
Cost	single lot ($)	$ 25,000	$ 4,000
discount schedule (%/line qty.)	1	1.00	1.00
	10	0.95	1.00
	20	0.90	1.00
	50	0.85	1.00
	100	0.80	0.90
	200	0.70	0.90
	500	0.60	0.80
	1000	0.50	0.70
	2000	0.50	0.60
	5000	0.50	0.50
	10000	0.50	0.50
	20000	0.50	0.45
	50000	0.50	0.45
	100000	0.50	0.40
	200000	0.50	0.40
	500000	0.50	0.40
	1000000	0.50	0.40
System	Total number produced	10000	
	subsystems per system	1	
	system cost ($)	$ 100,000	
	system price ($)	$ 60,000	
Lifecycle	least (years)	2	4
	most (years)	4	8
Premium (%)	interest during COTS life	1.00	1.00
	interest after COTS life	1.00	1.00
License	($/y) vendor to team	$ 25,000	
Support	($/y) by vendor to customer	$ 25,000	

Case Studies 17 and 18: Space-Qualified Data Acquisition System

Effort, Time, Salaries

		COTS					custom				
Category	Parameter	hourly rate ($/h)	Fabri-cation	Prod. Test	Assembly	Inventory	hourly rate ($/h)	Fabri-cation	Prod. Test	Assembly	Inventory
COGS FTE	production	$ 80	7	2	2	1	$ 80	7	2	2	1
	management	$ 130	0.2	0.2	0.2	0.2	$ 130	0.2	0.2	0.2	0.2
	administration	$ 60	0.1	0.1	0.1	1	$ 60	0.1	0.1	0.1	1

		hourly rate ($/h)	Concept	Critical	T&I	Prep.	hourly rate ($/h)	Concept	Critical	T&I	Prep.
Design FTE	hardware engineers	$ 110	1	1	2	1	$ 110	2	2	2	1
	software engineers	$ 110	1	2	2	1	$ 110	2	3	3	2
	mechanical engineers	$ 110	0.5	0.5	0.5	0.5	$ 110	2	2	2	1
	specialty engineers	$ 110	0.5	0.5	0.5	0.5	$ 110	0.5	0.5	0.5	0.5
	technicians	$ 80	1	1	2	2	$ 80	2	2	2	3
	Management	$ 130	0.1	0.1	0.1	0.1	$ 130	0.5	0.5	0.5	0.5
	administration	$ 60	0.1	0.1	0.1	0.1	$ 60	1	1	1	1
	production/consultants	$ 80	0.1	0.1	0.1	0.1	$ 80	0.5	0.5	0.5	1
Calendar	(months)		3	6	6	3		12	18	18	12

COGS Fabrication Parameters

		Value	Value
Fabrication time (h)	fabrication	20	160
	test	40	80
	assembly to system	80	80
	inventory	10	10
Fabrication costs ($)	sales	$ -	$ -
	technical support	$ -	$ -
	materials, energy/unit	$ -	$ -
	tool cost	$ -	$ -
	certification cost	$ -	$ -
Misc. time (days)	tooling time	0	0
	delivery time	105	105
	certification time	0	0

Subsystem and System Parameters

		Value	Value
Cost	single lot ($)	$150,000	$ 43,900
discount schedule (%/line qty.)	1	1	1.00
	10	0.9	0.95
	20	0.85	0.90
	50	0.8	0.85
	100	0.75	0.85
	200	0.7	0.85
	500	0.65	0.80
	1000	0.6	0.70
	2000	0.6	0.60
	5000	0.6	0.50
	10000	0.6	0.50
	20000	0.6	0.45
	50000	0.6	0.45
	100000	0.6	0.40
	200000	0.6	0.40
	500000	0.6	0.40
	1000000	0.6	0.40
System	Total number produced	10000	
	subsystems per system	1	
	system cost ($)	$ 100,000	
	system price ($)	$ 150,000	
Lifecycle	least (years)	5	5
	most (years)	10	10
Premium (%)	interest during COTS life	1.00	1.00
	interest after COTS life	1.00	1.00
License	($/y) vendor to team	$ -	
Support	($/y) by vendor to customer	$ -	

Notes for the Volatile Organic Compound Sensors Controller:

Assumes only basic programming for the board support package (BSP), not application programming.

enclosure = $8000
5 cameras = 5 * $2000
5 cameras enclosures = 5 * $1500
7 cables = 7 * $2500
components = $900

Example Simulation Outputs

(See following pages.)

Corrected Results: Plotting of parameters for $25K COTS v $4K Custom boards

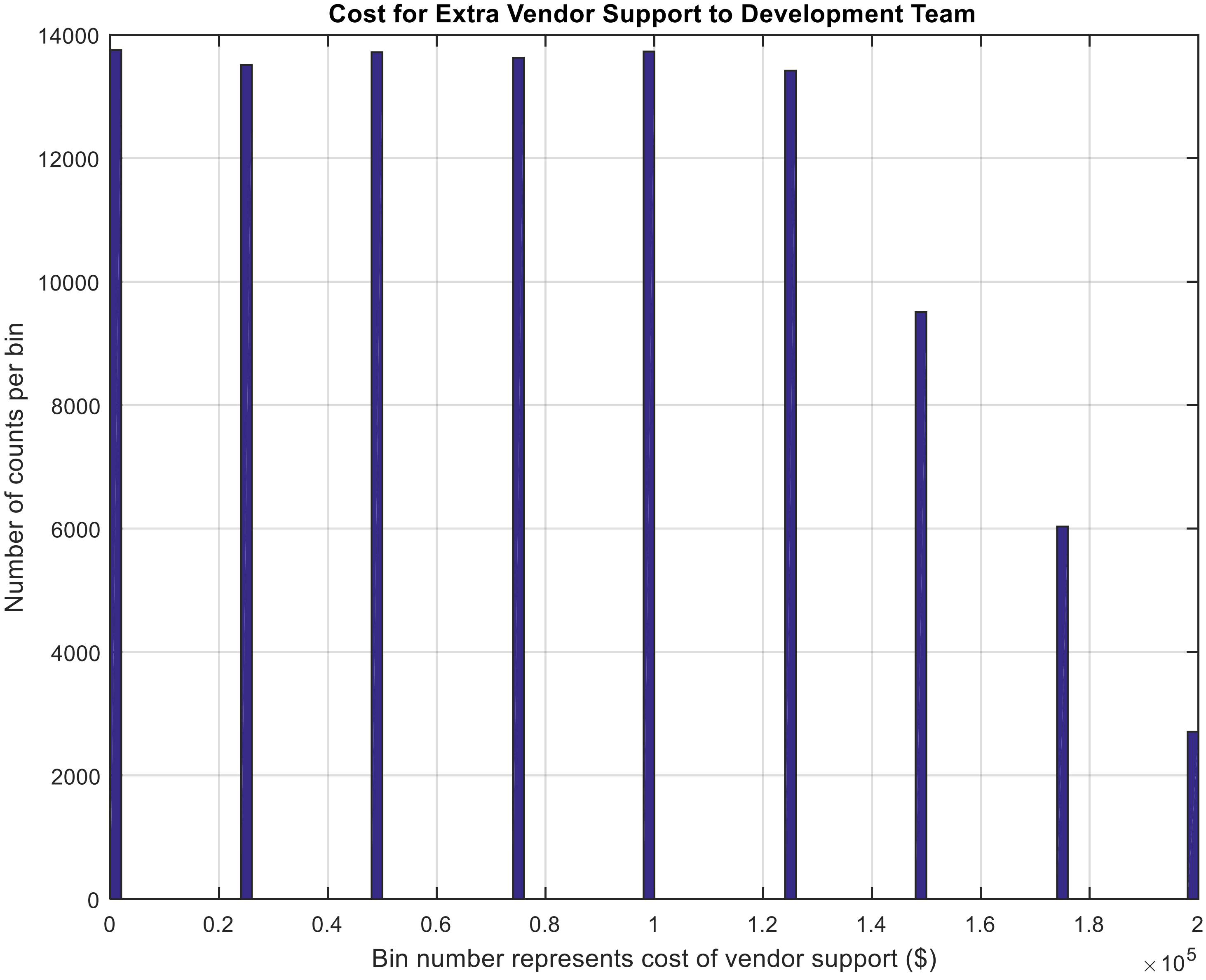
July 16, 2018

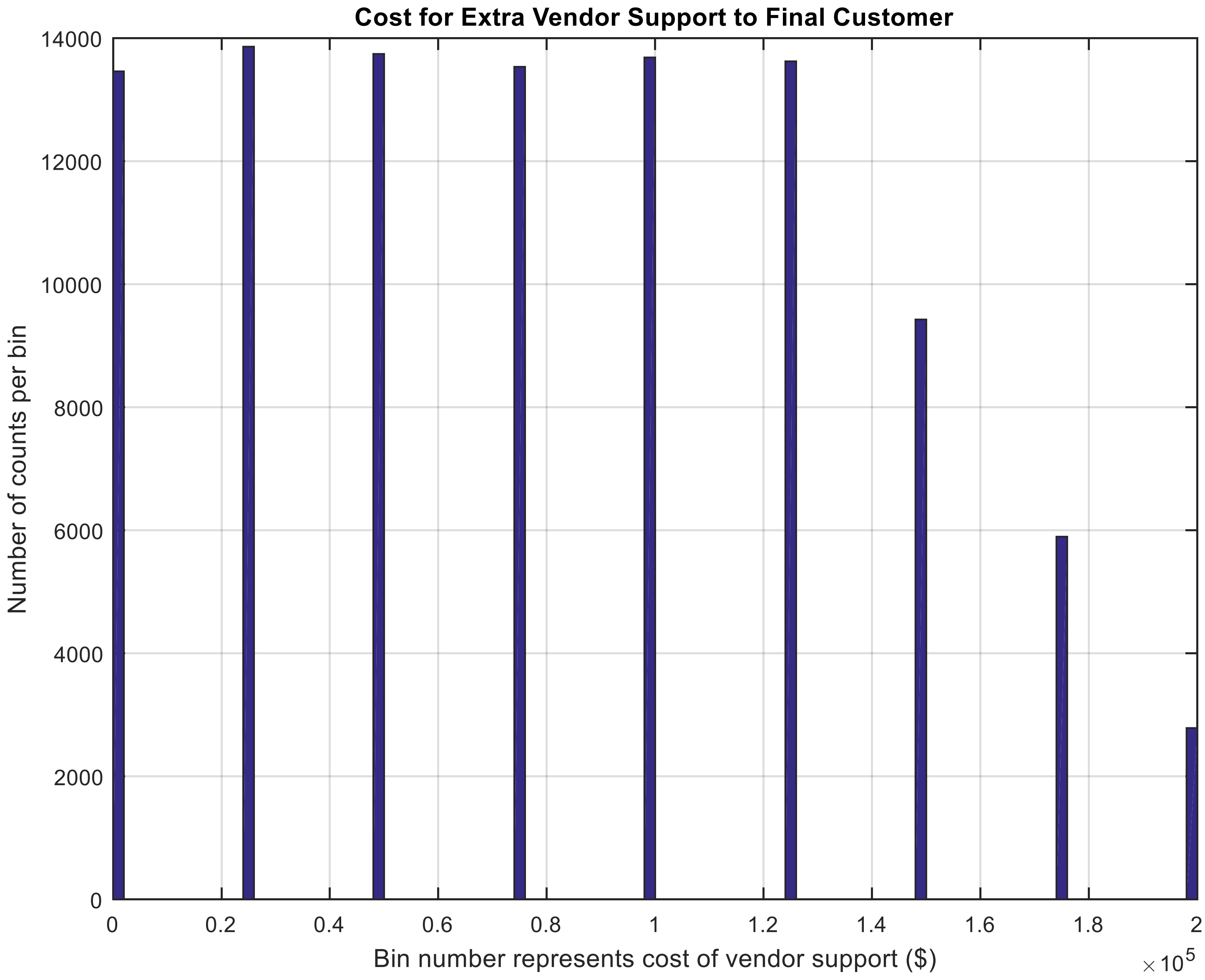
This is looking at how cross over distributions change with COTS correction factors: specifications, quality of support, effort to receive support, and effort to provide customer support. All are set to a flat, random distribution that varies 40% from either 60% to 100% or 0% to 40%.

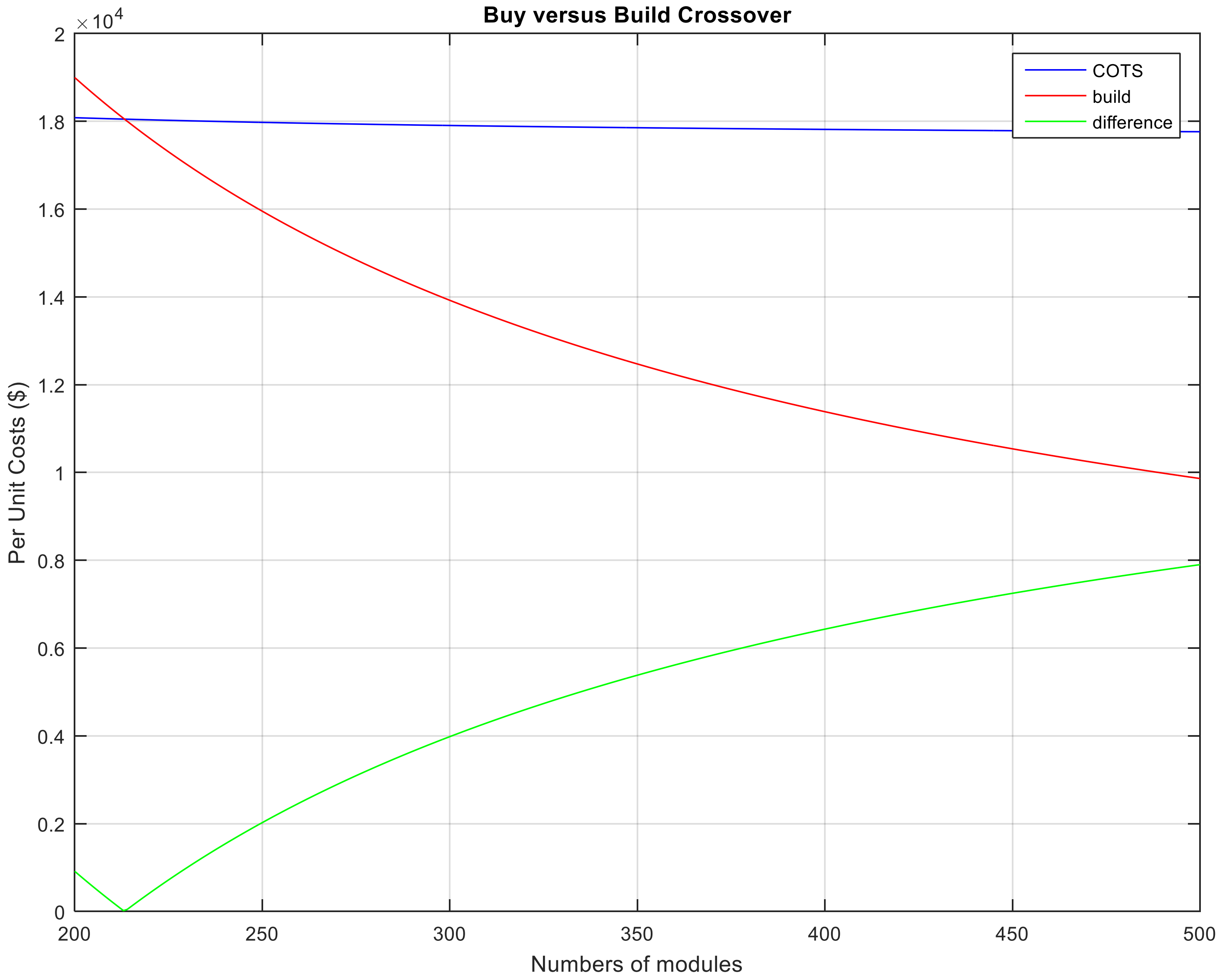
Premium = 1.
COTS single lot = $25,000
Components single lot = $4,000
Development team expertise: 60% to 100%
Resources available to team: 60% to 100%
Yearly team license support: $25,000/y
Yearly final customer support: $25,000/y
Each bin number in the 3D plots represents the 22.27 nearest bins of crossover points. So, 80 bins represents cross overs between 1795 and 1782.

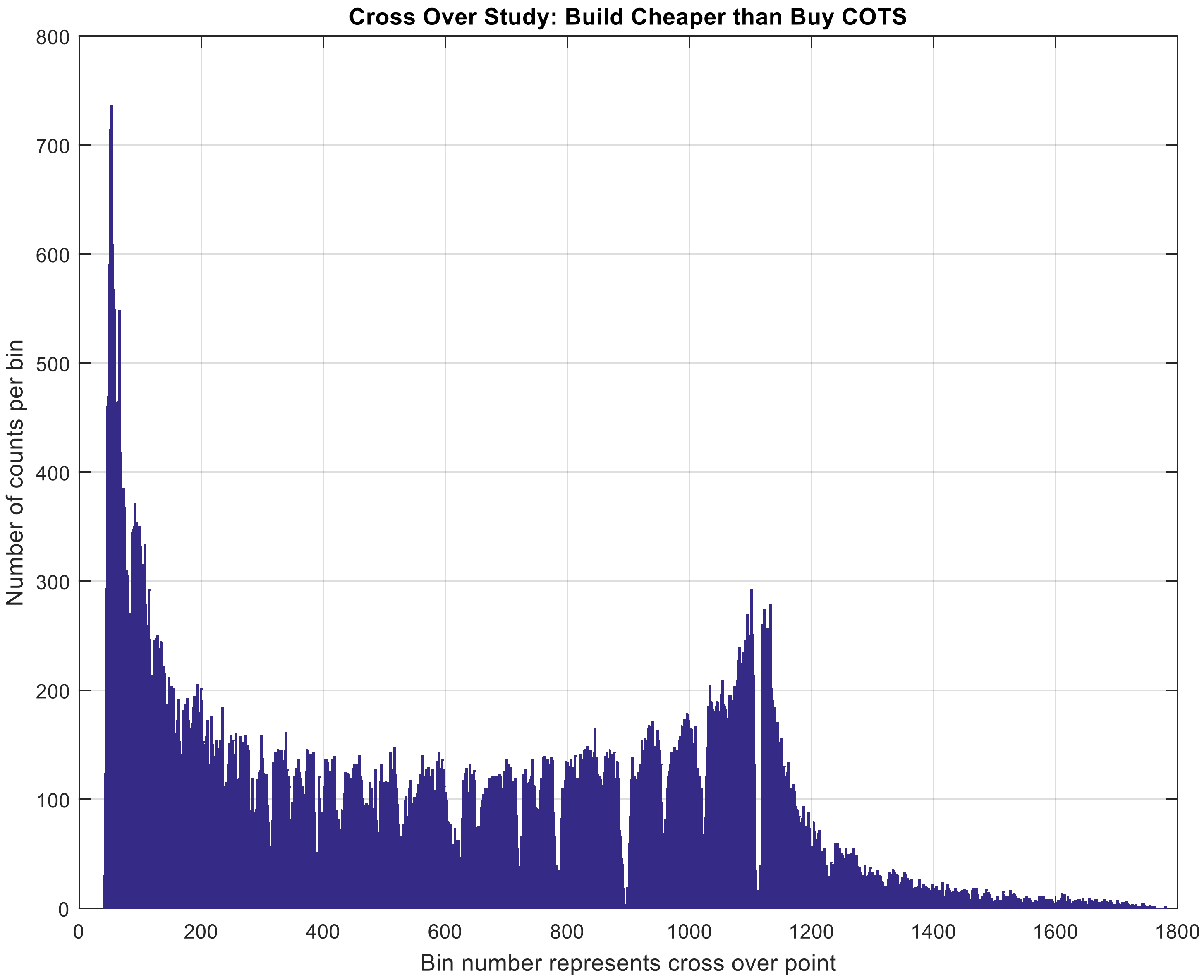
```
Histogram bin width = 22.27
Single-lot price for COTS module = $ 25000
Single-lot price for build components = $ 4000
COTS features range in Monte Carlo from 0.60 to 1.00
COTS market life ranges in Monte Carlo from 2.0 to 4.0
System market life ranges in Monte Carlo from 4.0 to 8.0
Premium to inventory COTS = 1.0

Number of systems planned = 10000
Number of modules per system = 1

Average loaded hourly rates for staff:
Production staff = $ 80
Technicians = $ 80
Engineers & developers = $ 110
Management = $ 130
Administrative support = $ 60
Consulting & production staff = $ 80

COTS support and documentation for particular vendor:
Licensing support cost = $ 25000
```

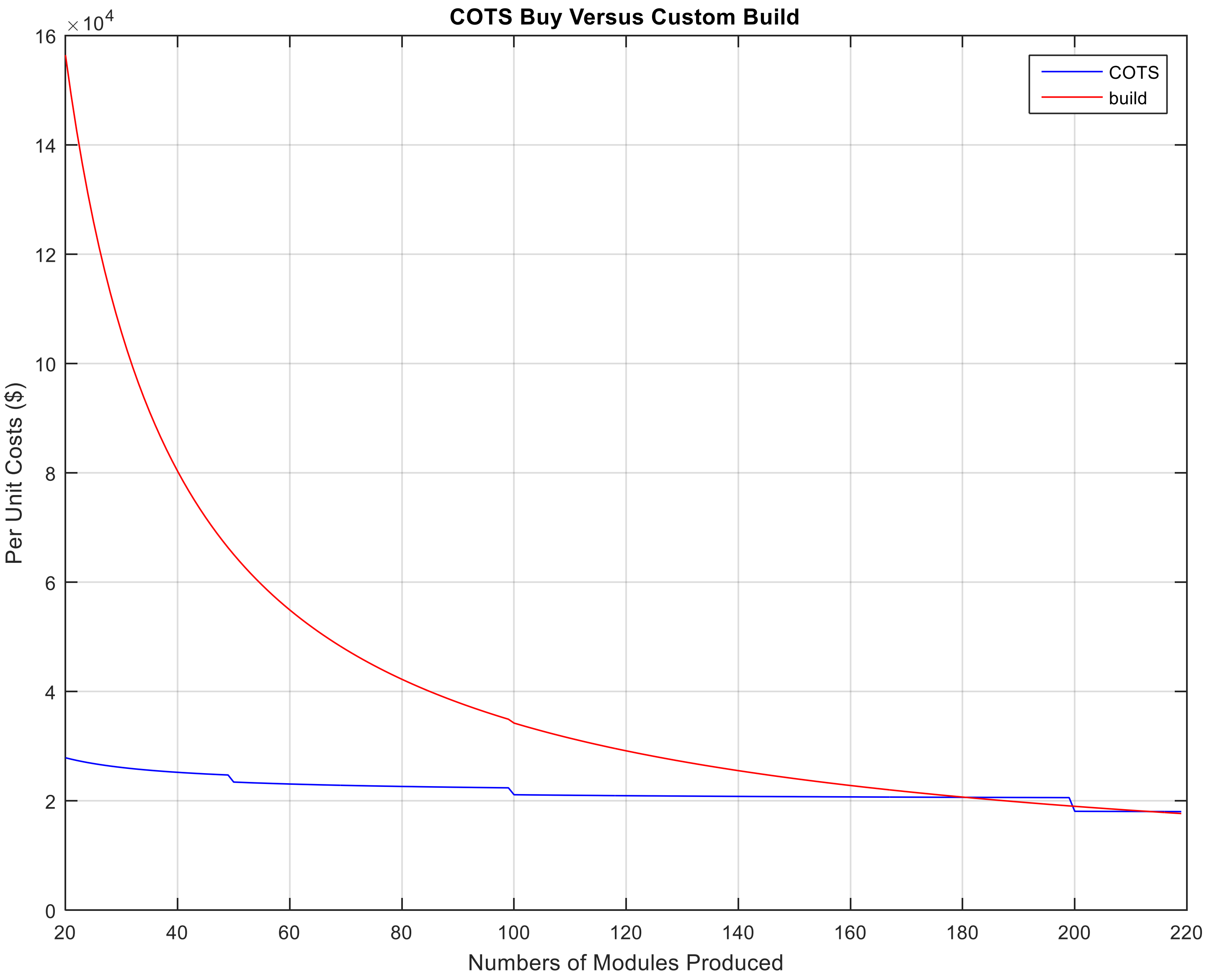
```
Customer support cost = $ 25000
Team NRE for vendor support ranges from 0.00 to 0.40
Team NRE for customer support ranges from 0.00 to 0.40
Quality of doc & support ranges in Monte Carlo from 0.60 to 1.00
Customer interaction directly with module = 1.00

Time of NRE for COTS (months) = 6.00
Time of NRE for custom (months) = 30.00
Cost of NRE for COTS = $ 106596
Cost of NRE for custom = $ 3044640

Lost opportunity cost = $ 1920000
Total system revenues over lifecycle = $ 400000000
Lost opportunity cost to total revenues = 0.0048

Crossover point (number of modules) = 213
Crossover COTS module cost = $ 18049
Crossover custom-build module cost = $ 18067
Press spacebar

# of simulation runs = 100000

Elapsed time is 243.768236 seconds.
```

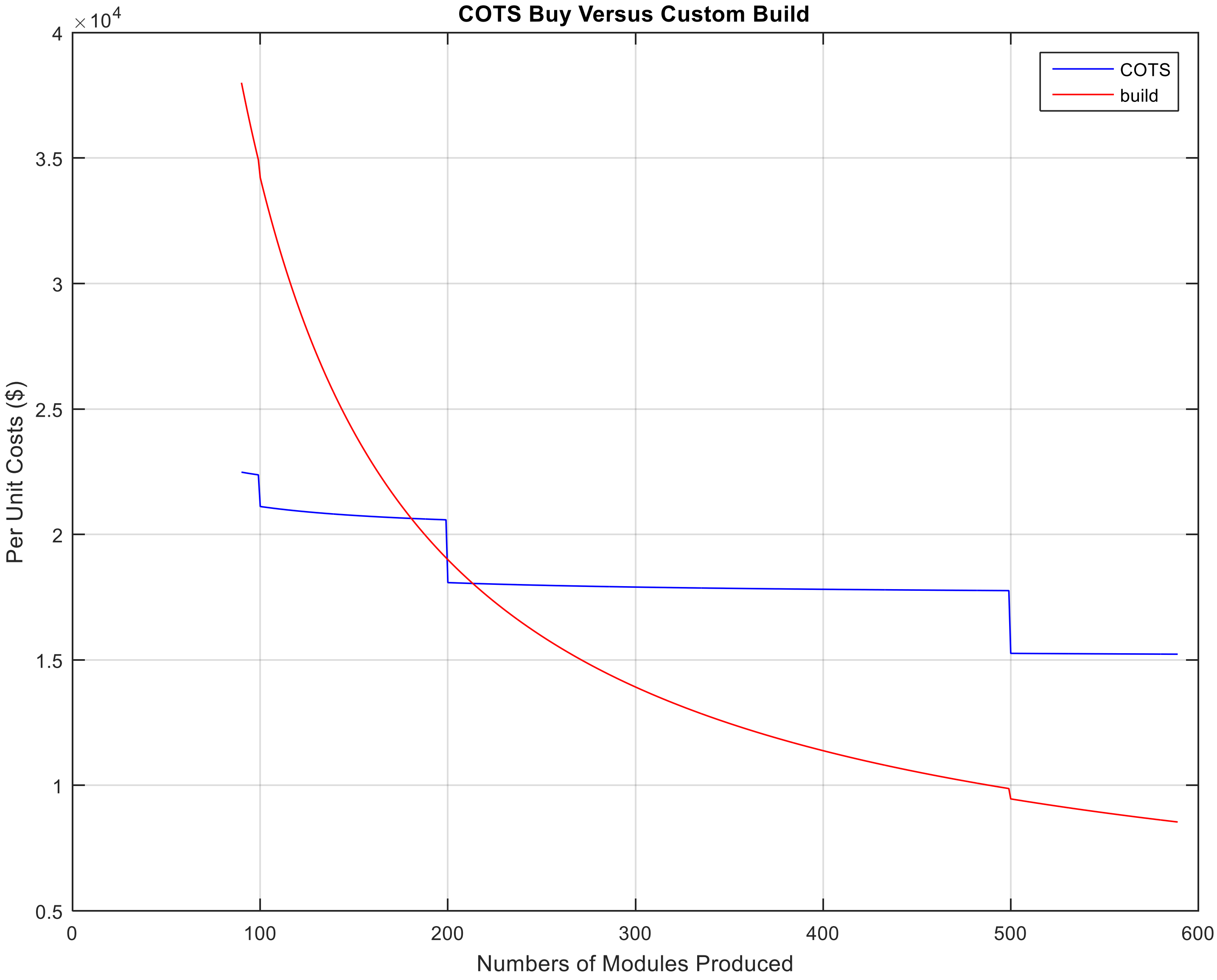

Cost for Extra Vendor Support to Development Team
Number of counts per bin
0
2000
4000
6000
8000
10000
12000
14000
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
Bin number represents cost of vendor support ($)
× 10^5

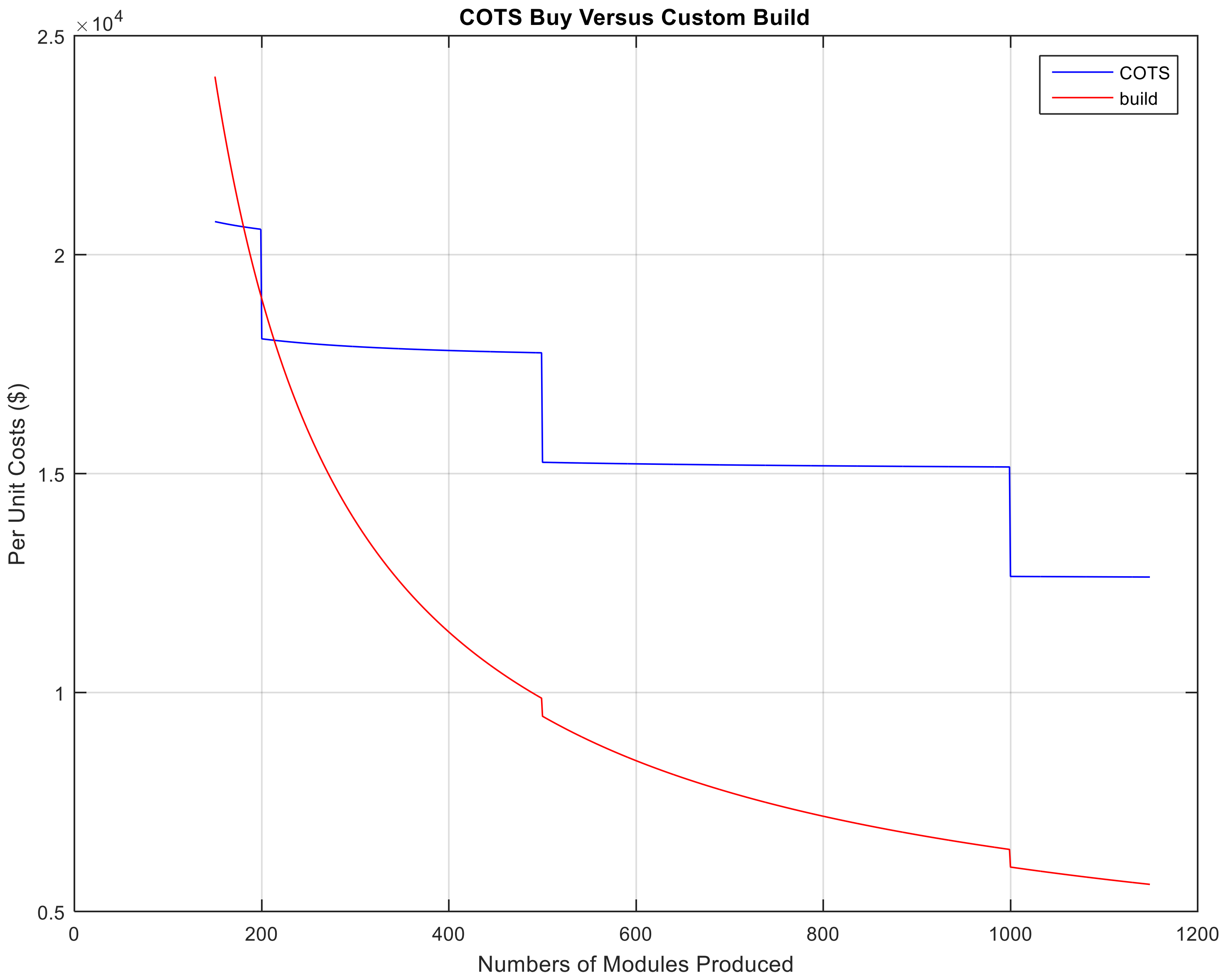

Cost for Extra Vendor Support to Final Customer
14000
12000
10000
8000
6000
4000
2000
0
Number of counts per bin
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
Bin number represents cost of vendor support ($)
$\times 10^5$

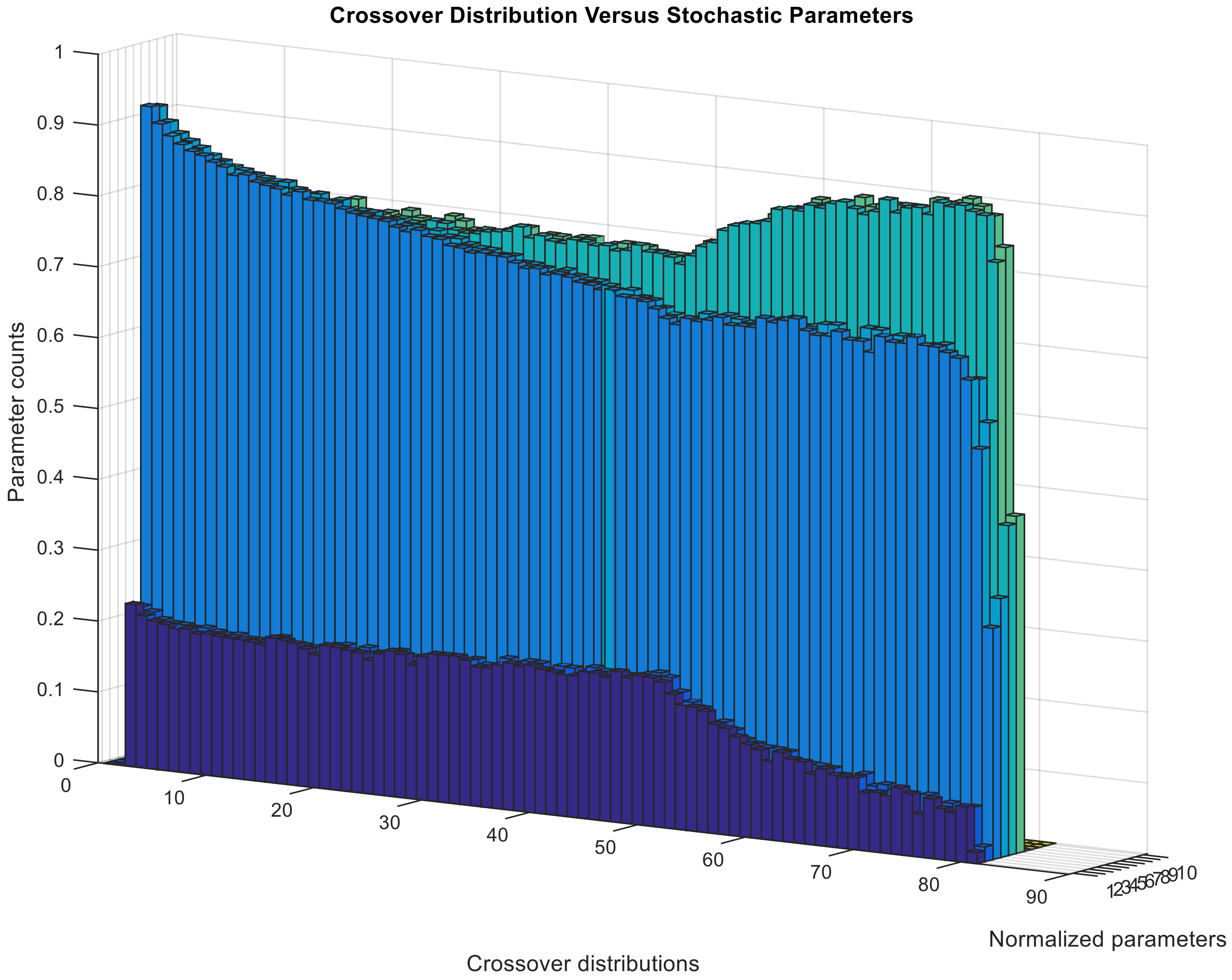

Buy versus Build Crossover
×10^4
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
Per Unit Costs ($)
200
250
300
350
400
450
500
Numbers of modules
COTS
build
difference

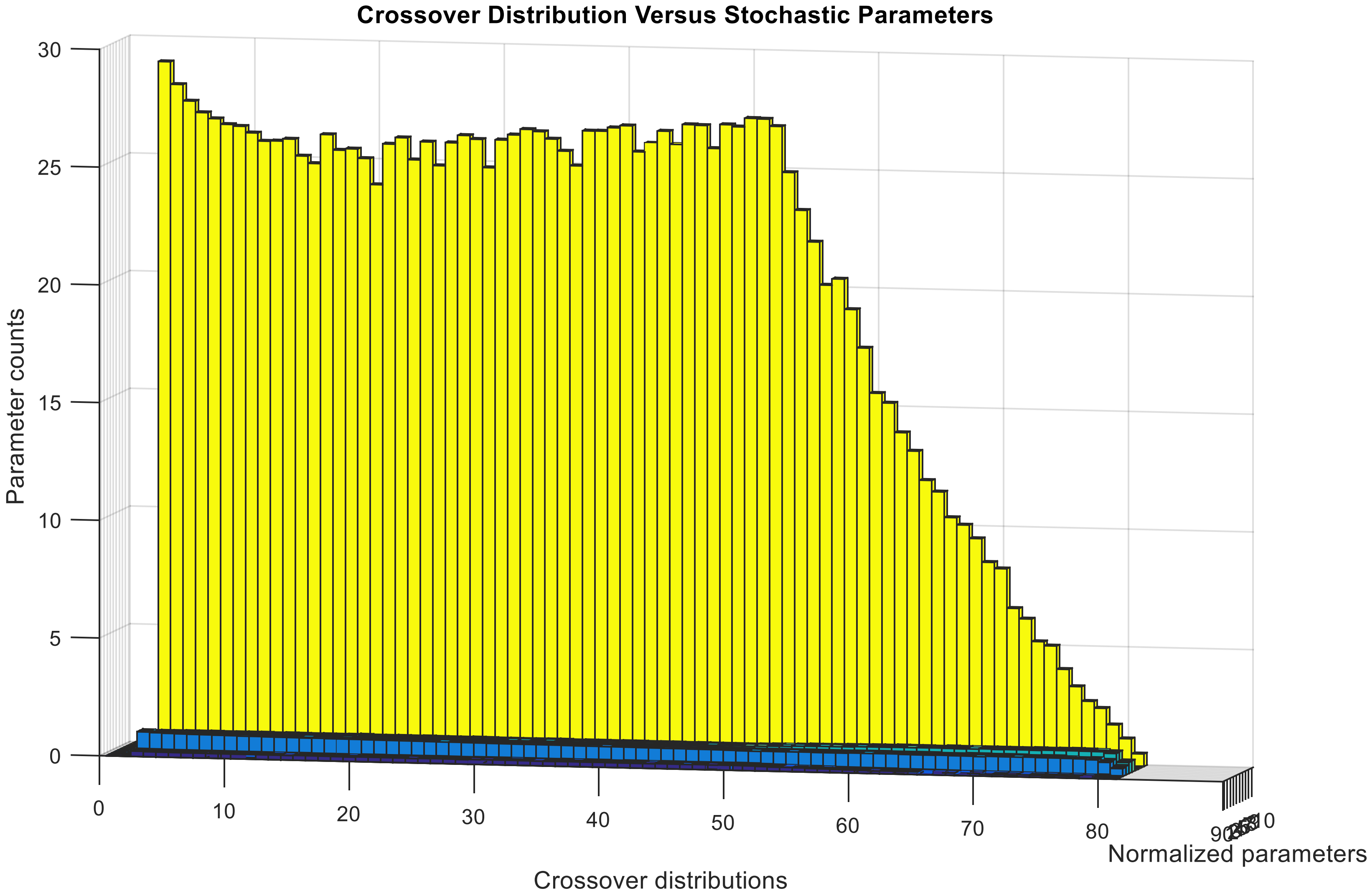

Cross Over Study: Build Cheaper than Buy COTS
800
700
600
500
400
300
200
100
0
Number of counts per bin
0
200
400
600
800
1000
1200
1400
1600
1800
Bin number represents cross over point

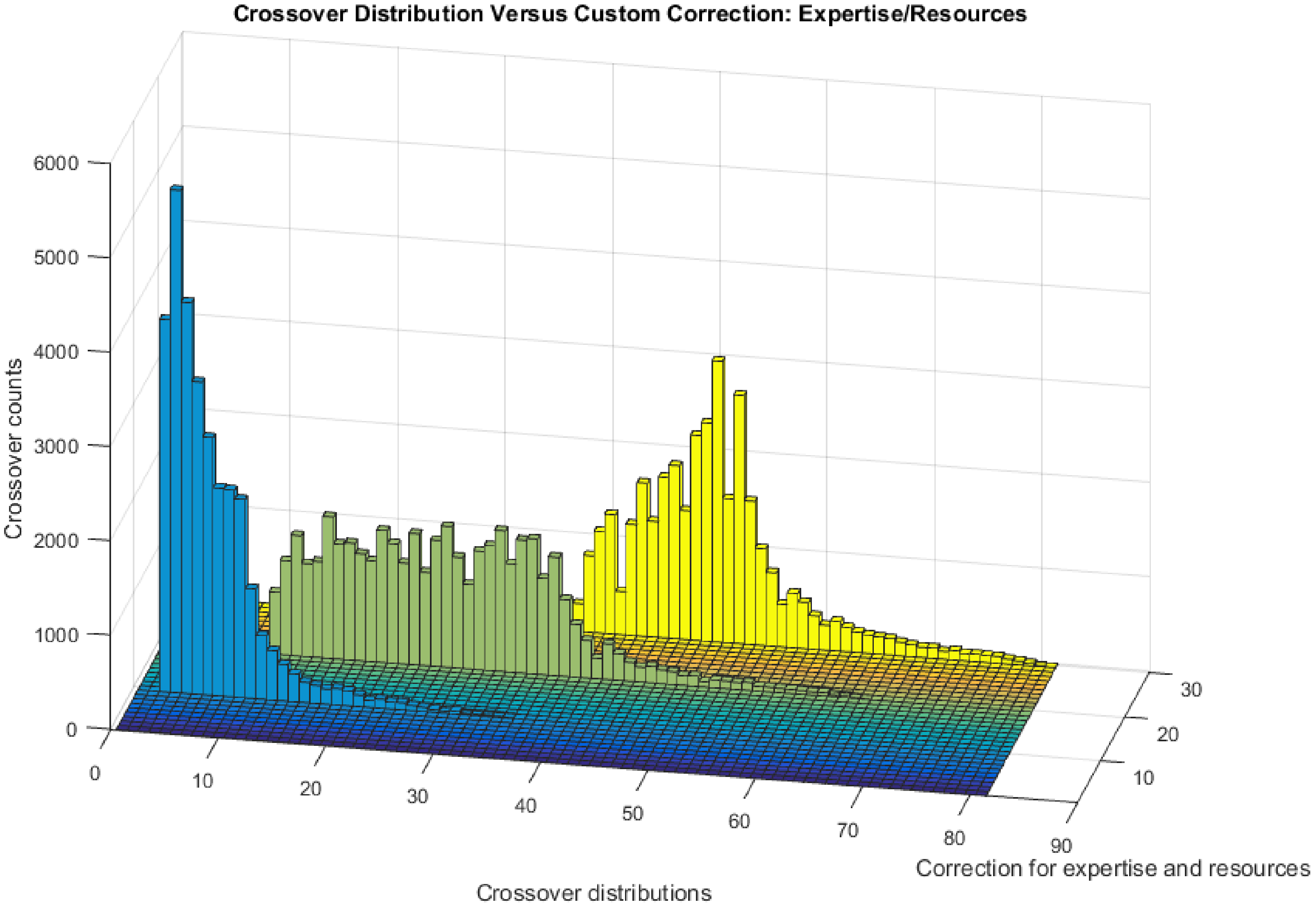

COTS Buy Versus Custom Build
×10^4
16
14
12
10
8
6
4
2
0
Per Unit Costs ($)
20
40
60
80
100
120
140
160
180
200
220
Numbers of Modules Produced
COTS
build

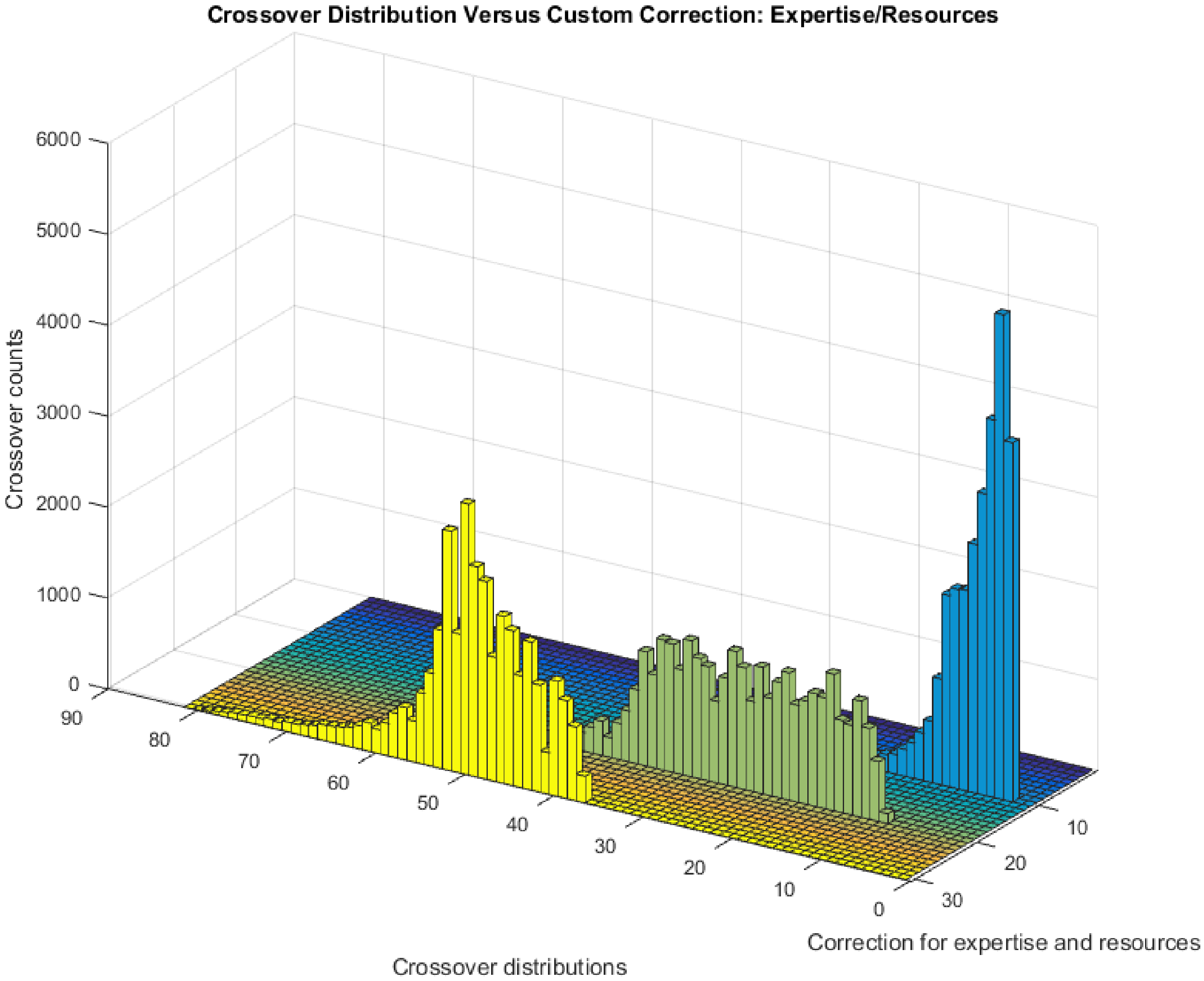

COTS Buy Versus Custom Build
×10^4
4
3.5
3
2.5
2
1.5
1
0.5
Per Unit Costs ($)
0
100
200
300
400
500
600
Numbers of Modules Produced
COTS
build

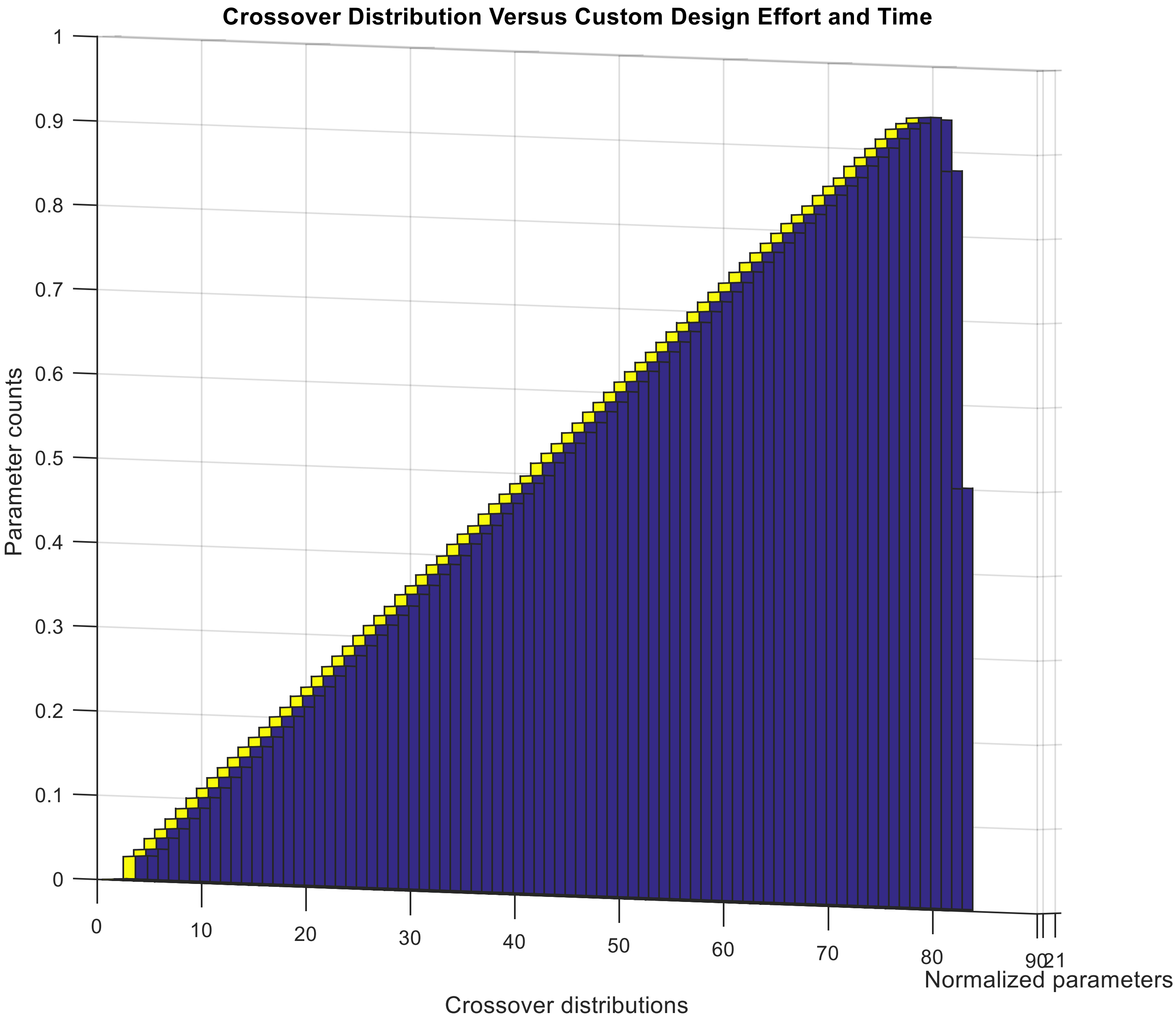

COTS Buy Versus Custom Build
×10^4
2.5
2
1.5
1
0.5
0
200
400
600
800
1000
1200
Per Unit Costs ($)
Numbers of Modules Produced
COTS
build

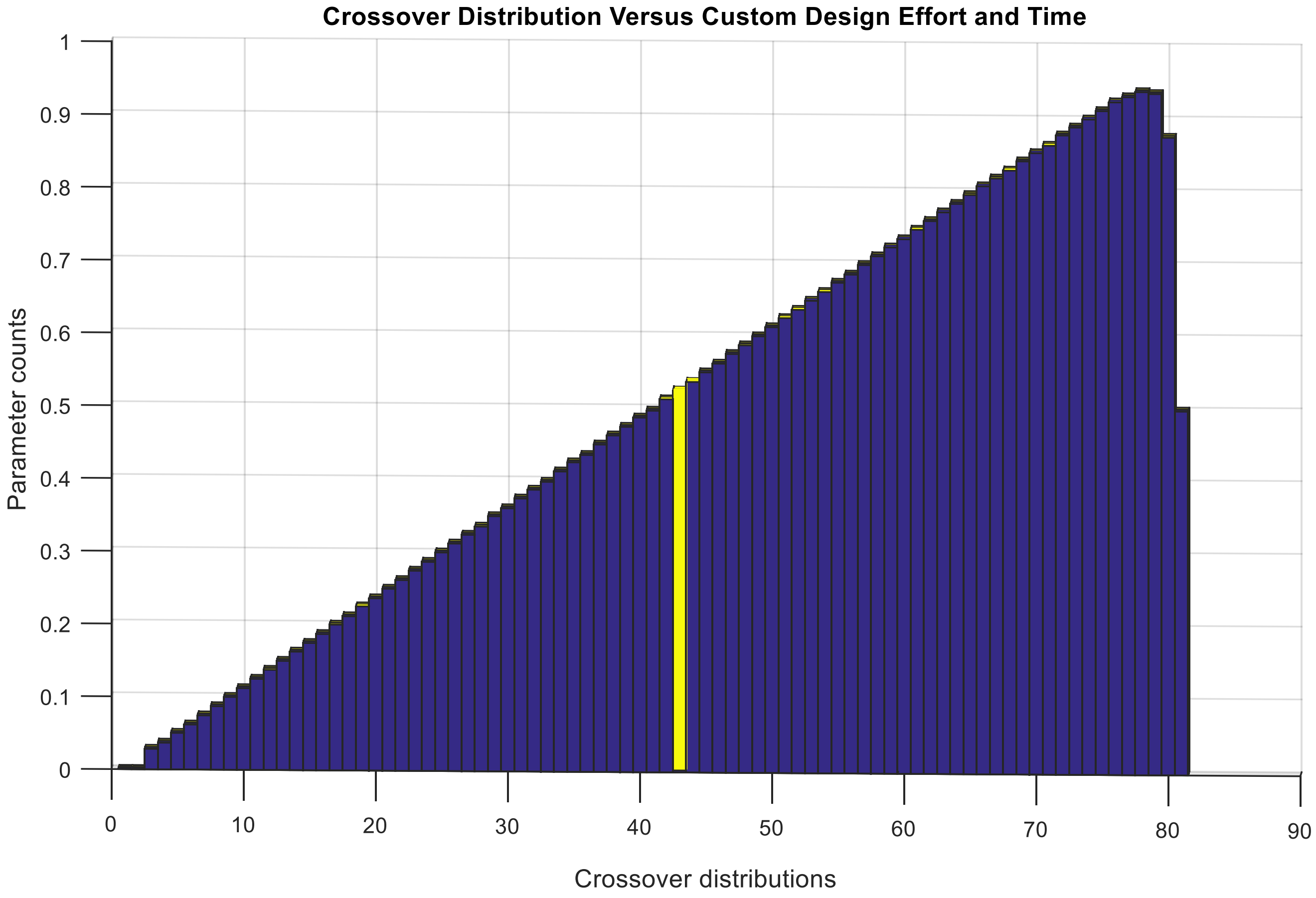

Crossover Distribution Versus Stochastic Parameters
Parameter counts
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
10
20
30
40
50
60
70
80
90
Crossover distributions
1
2
3
4
5
6
7
8
9
10
Normalized parameters


Crossover Distribution Versus Stochastic Parameters
Parameter counts
30
25
20
15
10
5
0
0
10
20
30
40
50
60
70
80
90
Crossover distributions
Normalized parameters

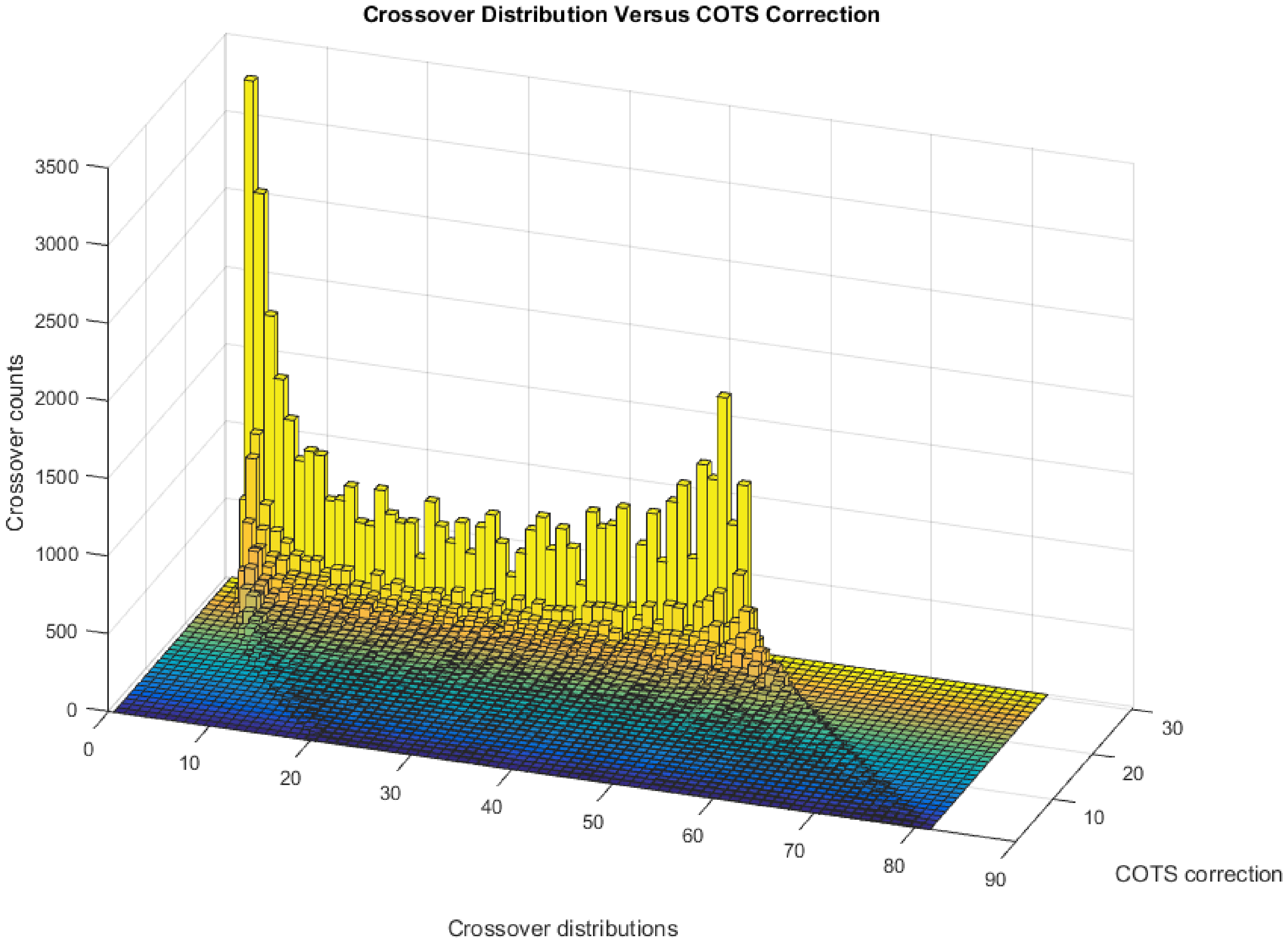

Crossover Distribution Versus Custom Correction: Expertise/Resources
Crossover counts
6000
5000
4000
3000
2000
1000
0
0
10
20
30
40
50
60
70
80
90
Crossover distributions
10
20
30
Correction for expertise and resources


Crossover Distribution Versus Custom Correction: Expertise/Resources
Crossover counts
6000
5000
4000
3000
2000
1000
0
90
80
70
60
50
40
30
20
10
0
Crossover distributions
10
20
30
Correction for expertise and resources

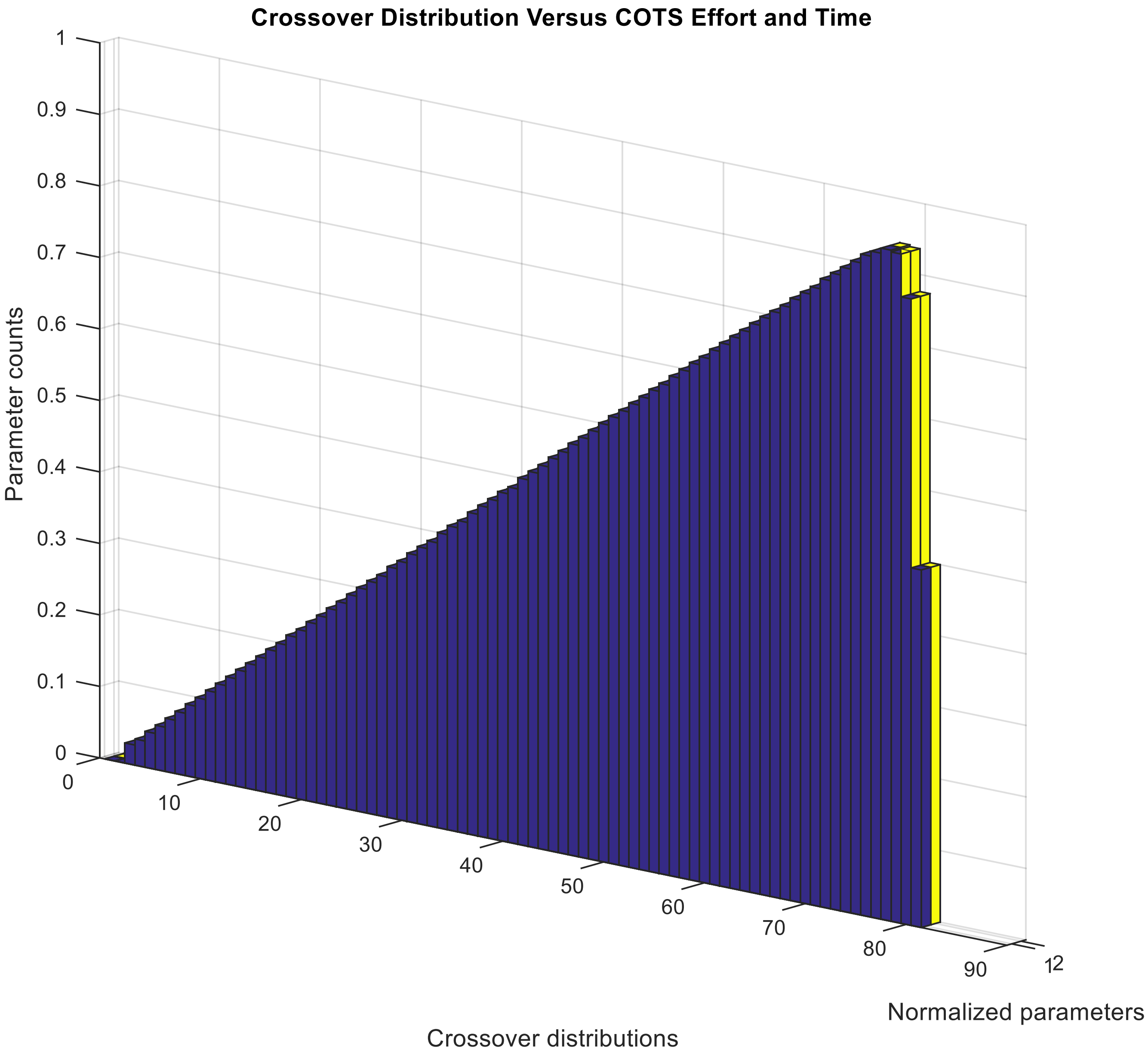

Crossover Distribution Versus Custom Design Effort and Time
Parameter counts
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
10
20
30
40
50
60
70
80
90
1
2
Crossover distributions
Normalized parameters

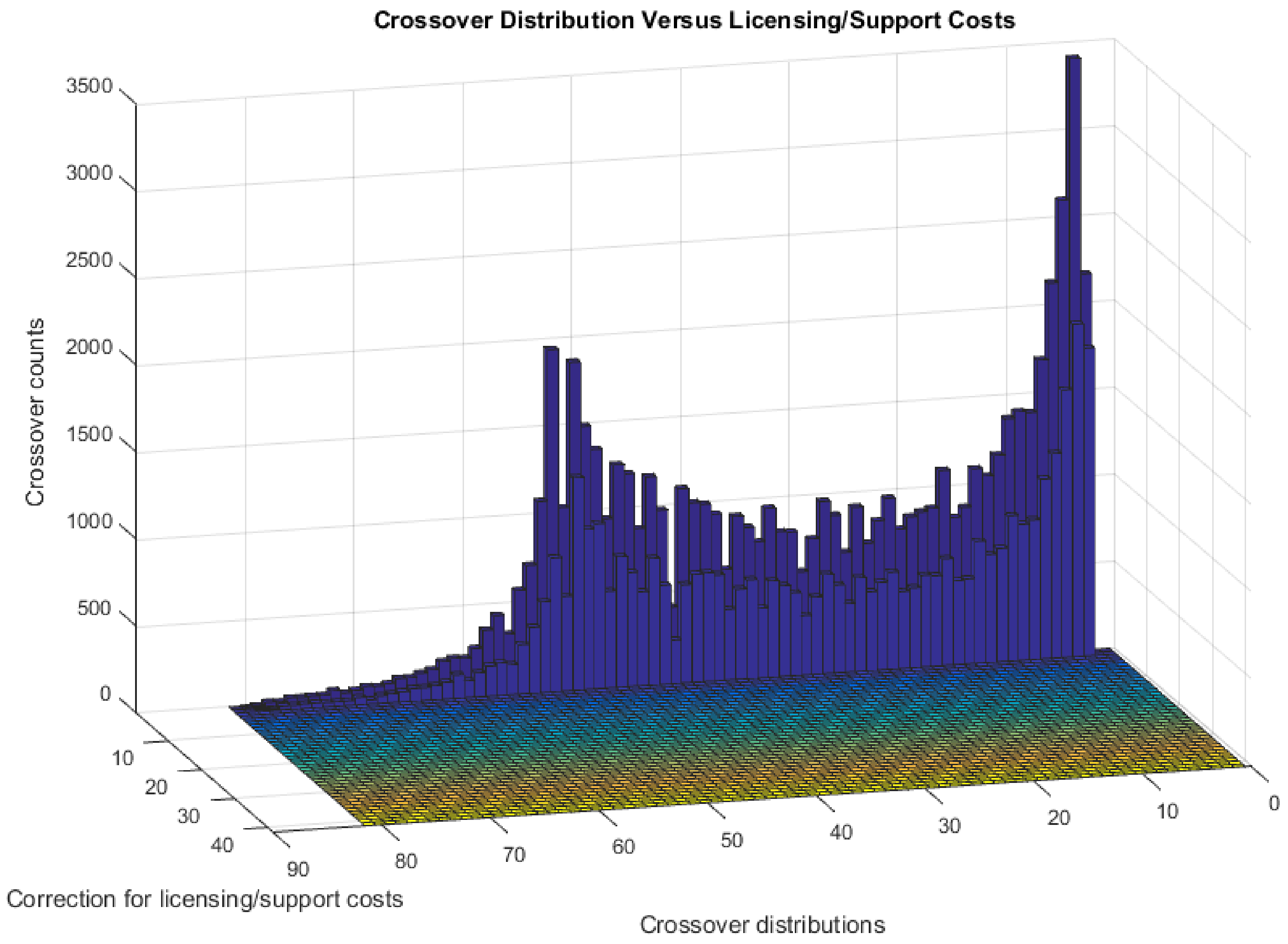


Vertical, yellow bar is a Matlab plotting artifact.



Crossover Distribution Versus COTS Correction
Crossover counts
3500
3000
2500
2000
1500
1000
500
0
0
10
20
30
40
50
60
70
80
90
Crossover distributions
30
20
10
COTS correction


Crossover Distribution Versus COTS Effort and Time
Parameter counts
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
10
20
30
40
50
60
70
80
90
1
2
Crossover distributions
Normalized parameters


Crossover Distribution Versus Licensing/Support Costs
Crossover counts
3500
3000
2500
2000
1500
1000
500
0
10
20
30
40
Correction for licensing/support costs
90
80
70
60
50
40
30
20
10
0
Crossover distributions

Matlab Source Code

Source Code for Parameter Initialization

June 23, 2018

```
function [d, Q1, Q2, Q3, Q4] = init_param_25K()
% --------------------------------------------------------------------------
%   This is init_param. By Kim R. Fowler.
%   Created on 23 June 2018.
%
%   This script file prepares a data matrix with initialization values for
%   simulations of the buy versus build model. It sets up of all the
%   variables for either COTS or custom-build. It also echo prints the
%   variables to screen set. - this is for 25K versus 4K boards
% --------------------------------------------------------------------------
COTS_price = 25000;          % Single-lot price for COTS module
Comp_price = 4000;           % Single-lot price for custom-build components

% --------------------------------------------------------------------------
%   Discount schedules at the following quantity breaks, N
% --------------------------------------------------------------------------
N = [1 10 20 50 100 200 500 1000 2000 5000 10000 20000 50000 100000];
COTS_discount = [1 0.95 0.9 0.85 0.8 0.7 0.6 0.5 0.5 0.5 0.5 0.5 0.5 0.5];
Comp_discount = [1 1 1 1 0.9 0.9 0.8 0.7 0.6 0.5 0.5 0.45 0.45 0.4];

% --------------------------------------------------------------------------
%   Percent of COTS features compared to the system specifications.
%   Make sure that the values stay between 0 and 1.
% --------------------------------------------------------------------------
COTS_spec_lo = 0.80;    % Lower limit on the Monte Carlo simulation
COTS_spec_hi = 1;       % Upper limit on the Monte Carlo simulation

% --------------------------------------------------------------------------
%   Make sure that the values of the market lifecycles (yrs) are > 0.
%   Also make sure that the HI > LO.
% --------------------------------------------------------------------------
% Market lifecycle (yrs) of COTS module, measured in years
COTS_life_lo = 2.0;     % Lower limit on the Monte Carlo simulation
COTS_life_hi = 4.0;     % Upper limit on the Monte Carlo simulation
% Market lifecycle (yrs) of system, measured in years
sys_life_lo = 4.0;      % Lower limit on the Monte Carlo simulation
sys_life_hi = 8.0;      % Upper limit on the Monte Carlo simulation

% --------------------------------------------------------------------------
%   Set premium for inventory for custom design or for COTS or to buy ROTS
%   if system life is greater than COTS module’s life. Premium can vary
%   between 1 and 5x.
% --------------------------------------------------------------------------
inven_prem1 = 1;        % Premium when system life < COTS life, usually = 1
inven_prem2 = 1;        % Premium when system life < COTS life, vary 1 to 5
```

```
% --------------------------------------------------------------------------
%   Some system numbers that are planned by development team.
% --------------------------------------------------------------------------
N_sys = 10000;                   % Number of systems planned over lifetime
mod_persys = 1;                 % Number of modules per system, default=1
Sys_price = 10000;                 % Planned unit price of a system
Sys_cost = 6000;                   % Planned unit cost of a system

% --------------------------------------------------------------------------
%   Print values of variables
% --------------------------------------------------------------------------
fprintf('\n')
fprintf('Single-lot price for COTS module = $ %.f\n', COTS_price)
fprintf('Single-lot price for build components = $ %.f\n', Comp_price)
fprintf('COTS features range in Monte Carlo from %1.2f', COTS_spec_lo)
fprintf(' to %1.2f\n',COTS_spec_hi)
fprintf('COTS market life ranges in Monte Carlo from %2.1f', COTS_life_lo)
fprintf(' to %2.1f\n',COTS_life_hi)
fprintf('System market life ranges in Monte Carlo from %2.1f', sys_life_lo)
fprintf(' to %2.1f\n',sys_life_hi)
fprintf('Premium to inventory COTS = %1.1f\n', inven_prem2)

fprintf('\n')
fprintf('Number of systems planned = %.f\n', N_sys)
fprintf('Number of modules per system = %.f\n', mod_persys)

% --------------------------------------------------------------------------
%   Labor costs in $/hr
% --------------------------------------------------------------------------
Prod_labor = 80;        % Average loaded salary of production staff
Tech_labor = 80;        % Average loaded salary of technicians
Eng_labor = 110;        % Average loaded salary of engineers & developers
Manag_labor = 130;      % Average loaded salary of management
Admin_labor = 60;       % Average loaded salary of admin & support staff
Consult_labor = 80;     % Average loaded salary of consulting & prod. staff

fprintf('\n')
fprintf('Average loaded hourly rates for staff: \n')
fprintf('Production staff = $ %.f\n', Prod_labor)
fprintf('Technicians = $ %.f\n', Tech_labor)
fprintf('Engineers & developers = $ %.f\n', Eng_labor)
fprintf('Management = $ %.f\n', Manag_labor)
fprintf('Administrative support = $ %.f\n', Admin_labor)
fprintf('Consulting & production staff = $ %.f\n', Consult_labor)

% --------------------------------------------------------------------------
%   Variable initialization for COGS calculations per unit COTS module.
% --------------------------------------------------------------------------
% Maxtrix FTE for COTS in production
COTS_COGS_FTE = [1 1 1 0; 0.5 0.5 0.5 0.5; 0 0 0 1];

COTS_fab_t = 0.1;       % Time (hrs) to fabricate and assemble a module,
                        % this includes time to load software, default=2min
COTS_test_t = 0.1;      % Time (hrs) to test a module, default=1 min
```

```
COTS_insert_t = 0.1;    % Time (hrs) in production to insert module
COTS_inventory =0.0333; % Time to inventory a module, default=2 min
COTS_sales = 0;         % Cost ($) of sales and marketing per module
COTS_techsup = 0;       % Cost ($) of technical support per module
COTS_mat_en = 0.5;      % Cost ($) for assets, materials, and energy / unit

% --------------------------------------------------------------------------
%   Variable initialization for COGS calculations per unit custom module.
% --------------------------------------------------------------------------
% Maxtrix FTE for custom-build in production
Cust_COGS_FTE = [1 1 1 0; 0.5 0.5 0.5 0.5; 0 0 0 1];
Cust_fab_t = 0.5;       % Time (hrs) to fabricate and assemble a module,
                        % this includes time to load software
Cust_test_t = 0.3;      % Time (hrs) to test a module
Cust_insert_t = 0.2;    % Time (hrs) in production to insert module
Cust_inventory =0.2;    % Time to inventory a module
Cust_sales = 0;         % Cost ($) of sales and marketing per module
Cust_techsup = 0;       % Cost ($) of technical support per module
Cust_mat_en = 2.5;      % Cost ($) for assets, materials, and energy / unit

% --------------------------------------------------------------------------
%   Variable initialization for COTS NRE calculations for development of
%   module incorporation in the system.
% --------------------------------------------------------------------------
COTS_tool_cost = 0;     % One time cost ($) to tool up for COTS production
                        % default = $0
COTS_tool_t = 0;        % Time (hrs) for tooling up for COTS production
                        % default = 0 hrs
COTS_deliv_t = 28;      % Delivery time (days) for first COTS modules
                        % default = 14 days
COTS_cert_cost = 20000; % Cost ($) for certification of COTS module
COTS_expert = 1;        % Percent expertise required by project,
                        % default = 1; reserved for future decision model
COTS_resource = 1;      % Percent resources, tools, facilities required by
                        % project, default = 1; reserved for future
                        % decision model
COTS_team_sup = 25000;      % Yearly cost for vendor supporting dev. team
Final_customer_sup = 25000; % Yearly cost for vendor supporting final
customer

% --------------------------------------------------------------------------
%   Percent of COTS vendor support needed by development team.
%   Make sure that the values stay between 0 and 1.
% --------------------------------------------------------------------------
COTS_ve_su_lo = 0;      % Lower limit on the Monte Carlo simulation
COTS_ve_su_hi = 0.2;    % Upper limit on the Monte Carlo simulation

% --------------------------------------------------------------------------
%   Percent quality of COTS vendor support provided to development team.
%   Make sure that the values stay between 0 and 1. 1 = best, 0 = none.
% --------------------------------------------------------------------------
Qual_doc_lo = 0.8;     % Lower limit on the Monte Carlo simulation
Qual_doc_hi = 1;        % Upper limit on the Monte Carlo simulation

% --------------------------------------------------------------------------
%   Percent COTS vendor support provided to customers; make sure that the
%   values stay between 0 and 1. Also vendor charge for extra support.
% -------------------------------------------------------------------------
COTS_te_su_lo = 0;      % Lower limit on the Monte Carlo simulation
COTS_te_su_hi = 0.2;    % Upper limit on the Monte Carlo simulation

COTS_interact = 1;      % Does customer interact directly with module?
                        % no = 0, yes = 1

fprintf('\n')
fprintf('COTS support and documentation for particular vendor: \n')
fprintf('Licensing support cost = $ %.f\n', COTS_team_sup)
fprintf('Customer support cost = $ %.f\n', Final_customer_sup)
fprintf('Team NRE for vendor support ranges from %1.2f', COTS_ve_su_lo)
fprintf(' to %1.2f\n', COTS_ve_su_hi)
fprintf('Team NRE for customer support ranges from %1.2f', COTS_te_su_lo)
fprintf(' to %1.2f\n', COTS_te_su_hi)
fprintf('Quality of doc & support ranges in Monte Carlo from %1.2f',
Qual_doc_lo)
fprintf(' to %1.2f\n', Qual_doc_hi)
fprintf('Customer interaction directly with module = %1.2f\n', COTS_interact)

% -------------------------------------------------------------------------
%   COTS NRE calculations for effort and time. The full time equivalent for
%   staffing is a 8 x 4 matrix; the rows represent specific types of
%   staff members; the columns represent development phases.
%
%   Row 1 = hardware engineers
%   Row 2 = software engineers and developers
%   Row 3 = mechanical engineers
%   Row 4 = specialty engineers (optics, chemical, materials, OR)
%   Row 5 = technicians
%   Row 6 = management
%   Row 7 = admin support staff
%   Row 8 = production staff and other consultants
%
%   Column 1 = Concept & Preliminary phase
%   Column 2 = Critical Design phase
%   Column 3 = Test & Integration phase
%   Column 4 = Compliance & Production Preparation phase
% -------------------------------------------------------------------------
COTS_NRE_FTE = [1 .1 .1 .5; 1 .1 .2 0; 0 0 0 0; 0 0 0 0; .25 .025 .025 .5; .1
.1 .025 .1; .1 .1 .1 .1; .05 .05 .025 .025];
COTS_Calendar = [1; 1; 2; 2];       %NRE calendar time per phase (months)

% -------------------------------------------------------------------------
%   Variable initialization for NRE calculations for development of a
%   custom-build module in the system.
% -------------------------------------------------------------------------
Cust_tool_cost = 3000;  % One time cost ($) to tool up for production
                        % default = $0
Cust_tool_t = 40;       % Time (hrs) for tooling up for production
                        % default = 0 hrs
Cust_deliv_t = 14;      % Delivery time (days) for first component lots
                        % default = 14 days
Cust_cert_cost = 0;     % Cost ($) for certification of custom-built module
```

```
% ----------------------------------------------------------------------
%   Percent of the necessary expertise of the development team.
%   Make sure that the values stay between 0 and 1. 1 = best, 0 = none.
% ----------------------------------------------------------------------
Cust_ex_lo = 0.6;       % Lower limit on the Monte Carlo simulation
Cust_ex_hi = 1;         % Upper limit on the Monte Carlo simulation

% ----------------------------------------------------------------------
%   Percent of the necessary resources available to the development team.
%   Make sure that the values stay between 0 and 1. 1 = best, 0 = none.
% ----------------------------------------------------------------------
Cust_re_lo = 0.6;       % Lower limit on the Monte Carlo simulation
Cust_re_hi = 1;         % Upper limit on the Monte Carlo simulation

% ----------------------------------------------------------------------
%   NRE calculations for effort and time. The full time equivalent for
%   staffing is a 8 x 4 matrix; the rows represent specific types of
%   staff members; the columns represent development phases.
%
%   Row 1 = hardware engineers
%   Row 2 = software engineers and developers
%   Row 3 = mechanical engineers
%   Row 4 = specialty engineers (optics, chemical, materials, OR)
%   Row 5 = technicians
%   Row 6 = management
%   Row 7 = admin support staff
%   Row 8 = production staff and other consultants
%
%   Column 1 = Concept & Preliminary phase
%   Column 2 = Critical Design phase
%   Column 3 = Test & Integration phase
%   Column 4 = Compliance & Production Preparation phase
% ----------------------------------------------------------------------
Cust_NRE_FTE = [1 1 1 .25; 2 2 2 1; 0 0 0 0; 0 0 0 0; 1 1 2 2; 1 1 1 1; .5 .5
.5 .5; .5 .5 .5 .5];
Cust_Calendar = [6; 12; 6; 6];     %NRE calendar time per phase (months)

% //////////////////////////////////////////////////////////////////////
% ----------------------------------------------------------------------
%   Store parameter and data initializations in matrices
% ----------------------------------------------------------------------
% //////////////////////////////////////////////////////////////////////

d = zeros(75,1);            % Initialize scalar data to store in a matrix

d(1,1) = COTS_price;
d(2,1) = Comp_price;

d(3,1) = COTS_spec_lo;
d(4,1) = COTS_spec_hi;
d(5,1) = COTS_life_lo;
d(6,1) = COTS_life_hi;
d(7,1) = sys_life_lo;
d(8,1) = sys_life_hi;
```

```
d(9,1) = inven_prem1;
d(10,1) = inven_prem2;
d(11,1) = N_sys;
d(12,1) = mod_persys;
d(13,1) = Sys_price;
d(14,1) = Sys_cost;

d(15,1) = Prod_labor;
d(16,1) = Tech_labor;
d(17,1) = Eng_labor;
d(18,1) = Manag_labor;
d(19,1) = Admin_labor;
d(20,1) = Consult_labor;

d(21,1) = COTS_fab_t;
d(22,1) = COTS_test_t;
d(23,1) = COTS_insert_t;
d(24,1) = COTS_inventory;
d(25,1) = COTS_sales;
d(26,1) = COTS_techsup;
d(27,1) = COTS_mat_en;

d(28,1) = Cust_fab_t;
d(29,1) = Cust_test_t;
d(30,1) = Cust_insert_t;
d(31,1) = Cust_inventory;
d(32,1) = Cust_sales;
d(33,1) = Cust_techsup;
d(34,1) = Cust_mat_en;

d(35,1) = COTS_tool_cost;
d(36,1) = COTS_tool_t;
d(37,1) = COTS_deliv_t;
d(38,1) = COTS_cert_cost;
d(39,1) = COTS_expert;
d(40,1) = COTS_resourc;
d(41,1) = COTS_team_sup;
d(42,1) = Final_customer_sup;

d(43,1) = COTS_ve_su_lo;
d(44,1) = COTS_ve_su_hi;
d(45,1) = Qual_doc_lo;
d(46,1) = Qual_doc_hi;
d(47,1) = COTS_te_su_lo;
d(48,1) = COTS_te_su_hi;
d(49,1) = COTS_interact;

d(50,1) = Cust_tool_cost;
d(51,1) = Cust_tool_t;
d(52,1) = Cust_deliv_t;
d(53,1) = Cust_cert_cost;

d(54,1) = Cust_ex_lo;
d(55,1) = Cust_ex_hi;
d(56,1) = Cust_re_lo;
```

```
d(57,1) = Cust_re_hi;


Q1 = zeros(3,14);            % Initialize the discount schedules
Q1(1,:) = N;
Q1(2,:) = COTS_discount;
Q1(3,:) = Comp_discount;

Q2 = zeros(3,4,2);           % Initialize the COGS FTE
Q2(:,:,1) = COTS_COGS_FTE;
Q2(:,:,2) = Cust_COGS_FTE;

Q3 = zeros(8,4,2);           % Initialize the development NRE
Q3(:,:,1) = COTS_NRE_FTE;
Q3(:,:,2) = Cust_NRE_FTE;

Q4 = zeros(4,1,2);           % Initialize the development calendar
Q4(:,:,1) = COTS_Calendar;
Q4(:,:,2) = Cust_Calendar;
```

Bvb24 with “importance sampling”

July 14, 2018

```
function [data_mat, O1, O2, U1, U2, N] = bvb24(d, Q1, Q2, Q3, Q4)
% --------------------------------------------------------------------------
%   This is bvbt23. By Kim R. Fowler.
%   Created on 23 June 2018.
%   Modified on 14 July 2018 for importance sampling.
%
%   These simulations only target embedded systems and they only refer to
%   one type of module for a particular system. The module under
%   consideration may have multiple copies within a single system.
%
%   This code includes insufficient COTS specifications, insufficient
%   vendor support, finite vendor consulting support, or lifecycle that
%   increase the COTS NRE and thus lower crossover point. It includes lower
%   expertise and resources that can increase the NRE of custom design.
%
%   Input is from data matrices with initialization values from the script
%   file, "init_param".
% --------------------------------------------------------------------------
clf                         % Clear figures from previous simulation run
clc                         % Clear commands from command window

% //////////////////////////////////////////////////////////////////////////
% --------------------------------------------------------------------------
%   Setup of all variables for either COTS or custom-build
% --------------------------------------------------------------------------
% //////////////////////////////////////////////////////////////////////////

COTS_price = d(1,1);        % Single-lot price for COTS module
Comp_price = d(2,1);        % Single-lot price for custom-build components

% --------------------------------------------------------------------------
%   Discount schedules at the following quantity breaks in N
% --------------------------------------------------------------------------
N = Q1(1,:);
N_indx = size(N);
COTS_discount = Q1(2,:);
Comp_discount = Q1(3,:);

% --------------------------------------------------------------------------
%   Percent of COTS features compared to the system specifications.
%   Make sure that the values stay between 0 and 1.
% --------------------------------------------------------------------------
COTS_spec_lo = d(3,1);      % Lower limit on the Monte Carlo simulation
COTS_spec_hi = d(4,1);      % Upper limit on the Monte Carlo simulation

% --------------------------------------------------------------------------
%   Make sure that the values of the market lifecycles (yrs) are > 0.
%   Also make sure that the HI > LO.
% --------------------------------------------------------------------------
% Market lifecycle (yrs) of COTS module, measured in years
COTS_life_lo = d(5,1);      % Lower limit on the Monte Carlo simulation
```

```
COTS_life_hi = d(6,1);      % Upper limit on the Monte Carlo simulation
% Market lifecycle (yrs) of system, measured in years
sys_life_lo = d(7,1);       % Lower limit on the Monte Carlo simulation
sys_life_hi = d(8,1);       % Upper limit on the Monte Carlo simulation

% -------------------------------------------------------------------------
%   Set premium for inventory for custom design or for COTS or to buy ROTS
%   if system life is greater than COTS module’s life. Premium can vary
%   between 1 and 5x.
% -------------------------------------------------------------------------
inven_prem1 = d(9,1);   % Premium when system life < COTS life, usually = 1
inven_prem2 = d(10,1);  % Premium when system life < COTS life, vary 1 to 5

% -------------------------------------------------------------------------
%   Some system numbers planned
% -------------------------------------------------------------------------
N_sys = d(11,1);               % Number of systems planned over lifetime
mod_sys = d(12,1);             % Number of modules per system, default=1
mod_total = N_sys * mod_sys;   % Total modules planned for product lifetime

Sys_price = d(13,1);                    % Planned unit price of a system
Sys_cost = d(14,1);                     % Planned unit cost of a system
Sys_revenue = Sys_price - Sys_cost;     % Planned revenue per system

Early_sales_rate = 2;           % Average early sales per month of systems

% -------------------------------------------------------------------------
%   Labor costs in $/hr
% -------------------------------------------------------------------------
Prod_labor = d(15,1);    % Average loaded salary of production staff
Tech_labor = d(16,1);    % Average loaded salary of technicians
Eng_labor = d(17,1);     % Average loaded salary of engineers & developers
Manag_labor = d(18,1);   % Average loaded salary of management
Admin_labor = d(19,1);   % Average loaded salary of admin & support staff
Consult_labor = d(20,1);% Average loaded salary of consulting & prod. staff

% COGS - vector of staff costs for production
COGS_labor(1,1) = Prod_labor;       % col.1 = production staff
COGS_labor(1,2) = Manag_labor;      % col.2 = management
COGS_labor(1,3) = Admin_labor;      % col.3 = admin/support staff

% NRE - vector of staff costs for development
Dev_labor(1,1) = Eng_labor;         % col.1 = H/W engineer
Dev_labor(1,2) = Eng_labor;         % col.2 = S/W engineer
Dev_labor(1,3) = Eng_labor;         % col.3 = Mechanical engineer
Dev_labor(1,4) = Eng_labor;         % col.4 = specialty engineer
Dev_labor(1,5) = Tech_labor;        % col.5 = technician
Dev_labor(1,6) = Manag_labor;       % col.6 = management
Dev_labor(1,7) = Admin_labor;       % col.7 = admin/support staff
Dev_labor(1,8) = Consult_labor;     % col.8 = production & misc consulting

% -------------------------------------------------------------------------
%   Variable initialization for COGS calculations per unit COTS module.
% -------------------------------------------------------------------------
% Maxtrix FTE for COTS in production
```

```
COTS_COGS_FTE = Q2(:,:,1);

COTS_fab_t = d(21,1);       % Time (hrs) to fabricate and assemble a module
COTS_test_t = d(22,1);      % Time (hrs) to test a module
COTS_insert_t = d(23,1);    % Time (hrs) in production to insert module
COTS_inventory = d(24,1);   % Time to inventory a module
COTS_sales = d(25,1);       % Cost ($) of sales and marketing per module
COTS_techsup = d(26,1);     % Cost ($) of technical support per module
COTS_mat_en = d(27,1);      % Cost ($) for assets, materials, energy/unit

% ---------------------------------------------------------------------------
%   Variable initialization for COGS calculations per unit custom module.
% ---------------------------------------------------------------------------
% Maxtrix FTE for custom-build in production
Cust_COGS_FTE = Q2(:,:,2);

Cust_fab_t = d(28,1);       % Time (hrs) to fabricate and assemble a module
Cust_test_t = d(29,1);      % Time (hrs) to test a module
Cust_insert_t = d(30,1);    % Time (hrs) in production to insert module
Cust_inventory = d(31,1);   % Time to inventory a module
Cust_sales = d(32,1);       % Cost ($) of sales and marketing per module
Cust_techsup = d(33,1);     % Cost ($) of technical support per module
Cust_mat_en = d(34,1);      % Cost ($) for assets, materials, energy/unit

% ---------------------------------------------------------------------------
%   Variable initialization for COTS NRE calculations for development of
%   module incorporation in the system.
% ---------------------------------------------------------------------------
COTS_tool_cost = d(35,1);   % One time cost ($) to tool up for COTS
production
COTS_tool_t = d(36,1);      % Time (hrs) for tooling up for COTS production
COTS_deliv_t = d(37,1);     % Delivery time (days) for first COTS modules
COTS_cert_cost = d(38,1);   % Cost ($) for certification of COTS module

COTS_team_sup = d(41,1);        % Yearly cost ($) for vendor supporting
                                % development team
Final_customer_sup = d(42,1);   % Yearly cost ($) for vendor supporting
                                % final customer
% ---------------------------------------------------------------------------
%   Percent of COTS vendor support needed by development team.
%   Make sure that the values stay between 0 and 1.
% ---------------------------------------------------------------------------
COTS_ve_su_lo = d(43,1);    % Lower limit on the Monte Carlo simulation
COTS_ve_su_hi = d(44,1);    % Upper limit on the Monte Carlo simulation

% ---------------------------------------------------------------------------
%   Percent quality of COTS vendor support provided to development team.
%   Make sure that the values stay between 0 and 1. 1 = best, 0 = none.
% ---------------------------------------------------------------------------
Qual_doc_lo = d(45,1);      % Lower limit on the Monte Carlo simulation
Qual_doc_hi = d(46,1);      % Upper limit on the Monte Carlo simulation

% ---------------------------------------------------------------------------
%   Percent COTS vendor support provided to customers; make sure that the
%   values stay between 0 and 1. Also vendor charge for extra support.
% ----------------------------------------------------------------------------
COTS_te_su_lo = d(47,1);    % Lower limit on the Monte Carlo simulation
COTS_te_su_hi = d(48,1);    % Upper limit on the Monte Carlo simulation

COTS_interact = d(49,1);    % Does customer interact directly with module?
                            % no = 0, yes = 1
Final_customer_sup_cost = Final_customer_sup * COTS_interact;

% ----------------------------------------------------------------------------
%   COTS NRE calculations for effort and time. The full time equivalent for
%   staffing is a 8 x 4 matrix; the rows represent specific types of
%   staff members; the columns represent development phases.
%
%   Row 1 = hardware engineers
%   Row 2 = software engineers and developers
%   Row 3 = mechanical engineers
%   Row 4 = specialty engineers (optics, chemical, materials, OR)
%   Row 5 = technicians
%   Row 6 = management
%   Row 7 = admin support staff
%   Row 8 = production staff and other consultants
%
%   Column 1 = Concept & Preliminary phase
%   Column 2 = Critical Design phase
%   Column 3 = Test & Integration phase
%   Column 4 = Compliance & Production Preparation phase
% ----------------------------------------------------------------------------
COTS_NRE_FTE = Q3(:,:,1);
COTS_Calendar = Q4(:,:,1);  %NRE calendar time per phase (months)

% ----------------------------------------------------------------------------
%   Variable initialization for NRE calculations for development of a
%   custom-build module in the system.
% ----------------------------------------------------------------------------
Cust_tool_cost = d(50,1);   % One time cost ($) to tool up for production
Cust_tool_t = d(51,1);      % Time (hrs) for tooling up for production
Cust_deliv_t = d(52,1);     % Delivery time (days) for first component lots
Cust_cert_cost = d(53,1);   % Cost($) to certify custom-built module

% ----------------------------------------------------------------------------
%   Percent of the necessary expertise of the development team.
%   Make sure that the values stay between 0 and 1. 1 = best, 0 = none.
% ----------------------------------------------------------------------------
Cust_ex_lo = d(54,1);       % Lower limit on the Monte Carlo simulation
Cust_ex_hi = d(55,1);       % Upper limit on the Monte Carlo simulation

% ----------------------------------------------------------------------------
%   Percent of the necessary resources available to the development team.
%   Make sure that the values stay between 0 and 1. 1 = best, 0 = none.
% ----------------------------------------------------------------------------
Cust_re_lo = d(56,1);       % Lower limit on the Monte Carlo simulation
Cust_re_hi = d(57,1);       % Upper limit on the Monte Carlo simulation

% ----------------------------------------------------------------------------
%   NRE calculations for effort and time. The full time equivalent for
%   staffing is a 8 x 4 matrix; the rows represent specific types of
%   staff members; the columns represent development phases.
%
%   Row 1 = hardware engineers
%   Row 2 = software engineers and developers
%   Row 3 = mechanical engineers
%   Row 4 = specialty engineers (optics, chemical, materials, OR)
%   Row 5 = technicians
%   Row 6 = management
%   Row 7 = admin support staff
%   Row 8 = production staff and other consultants
%
%   Column 1 = Concept & Preliminary phase
%   Column 2 = Critical Design phase
%   Column 3 = Test & Integration phase
%   Column 4 = Compliance & Production Preparation phase
% -------------------------------------------------------------------------
Cust_NRE_FTE = Q3(:,:,2);
Cust_Calendar = Q4(:,:,2);     %NRE calendar time per phase (months)

% /////////////////////////////////////////////////////////////////////////
% -------------------------------------------------------------------------
%   Perform deterministic calculations, assuming exact estimation of
%   effort, FTE, and calendar time.
% -------------------------------------------------------------------------
% /////////////////////////////////////////////////////////////////////////

% -------------------------------------------------------------------------
%   COGS calculations per unit COTS module. This simulation assumes
%   no learning curve to shorten time in production.
% -------------------------------------------------------------------------
% Vector time and total time for COTS in production
COTS_COGS_t = [COTS_fab_t; COTS_test_t; COTS_insert_t; COTS_inventory];
COTS_COGS_tot_t = sum(COTS_COGS_t);     % Total time (hrs) per module
% Total cost ($) of labor, material, and energy per module
COTS_COGS_tot_cost = (COGS_labor*(COTS_COGS_FTE*COTS_COGS_t)) + COTS_sales +
COTS_techsup + COTS_mat_en;

% -------------------------------------------------------------------------
%   COGS calculations per unit custom-built module. This simulation assumes
%   no learning curve to shorten time in production.
% -------------------------------------------------------------------------
% Total time and total time for custom-build in production
Cust_COGS_t = [Cust_fab_t; Cust_test_t; Cust_insert_t; Cust_inventory];
Cust_COGS_tot_t = sum(Cust_COGS_t);     % Total time (hrs) per module
% Total cost ($) of labor, material, and energy per module
Cust_COGS_tot_cost =(COGS_labor*(Cust_COGS_FTE* Cust_COGS_t)) + Cust_sales +
Cust_techsup + Cust_mat_en;

% -------------------------------------------------------------------------
%   COTS NRE calculations for development of module incorporation in the
%   system. The NRE calculations then results in cost per unit COTS module.
% -------------------------------------------------------------------------
COTS_person_month = COTS_NRE_FTE * COTS_Calendar;
COTS_Tot_effort = sum(COTS_person_month);
    %Total NRE labor cost for project ($), assume 168 hrs/month
COTS_Tot_labor = Dev_labor * COTS_person_month * 168;
```

```
COTS_Tot_time = sum(COTS_Calendar); %Total calendar time for NRE (months)

% Calculate total COTS costs from NRE, tooling, and compliance/certification
COTS_NRE_cost = COTS_Tot_labor + COTS_tool_cost + COTS_cert_cost;
O2 = COTS_NRE_cost;

% Calculate prices for COTS modules with discounts and inventory premium
COTS_mod_p = COTS_discount * COTS_price * inven_prem2;

% Set up COTS cost vector from total COGS cost
COTS_COGS_cost = ones(1,N_indx(1,2)) * COTS_COGS_tot_cost;

% Calculate vector of COTS unit prices at the cost breakpoints
COTS_unit_cost = COTS_COGS_cost + COTS_mod_p + (COTS_NRE_cost ./N);
O1 = COTS_unit_cost;

% -----------------------------------------------------------------------
%   NRE calculations for development of a custom-build module in the
%   system. The NRE calculations then results in cost per unit module.
% -----------------------------------------------------------------------
Cust_person_month = Cust_NRE_FTE * Cust_Calendar;
Cust_Tot_effort = sum(Cust_person_month);
%Total NRE labor cost for project ($), assume 168 hrs/month
Cust_Tot_labor = Dev_labor * Cust_person_month * 168;
Cust_Tot_time = sum(Cust_Calendar); %Total calendar time for NRE (months)

% Calculate build costs from NRE, tooling, and compliance/certification
Cust_NRE_cost = Cust_Tot_labor + Cust_tool_cost + Cust_cert_cost;
U2 = Cust_NRE_cost;

% Calculate prices for custom-built modules with discounts and inventory
Cust_mod_p = Comp_discount * Comp_price * inven_prem1;

% Set up build cost vector from total COGS cost
Cust_COGS_cost = ones(1,N_indx(1,2)) * Cust_COGS_tot_cost;

% Calculate the lost opportunity costs
lost_opp_cost = (Cust_Tot_time - COTS_Tot_time) * Early_sales_rate *
Sys_revenue;

fprintf('\n')
fprintf('Time of NRE for COTS (months) = %3.2f\n',COTS_Tot_time)
fprintf('Time of NRE for custom (months) = %3.2f\n',Cust_Tot_time)
fprintf('Cost of NRE for COTS = $ %.f\n',COTS_NRE_cost)
fprintf('Cost of NRE for custom = $ %.f\n',Cust_NRE_cost)

Sys_Tot_rev = Sys_revenue * N_sys;
Per_lost_opp = lost_opp_cost/Sys_Tot_rev;
fprintf('\n')
fprintf('Lost opportunity cost = $ %.f\n',lost_opp_cost)
fprintf('Total system revenues over lifecycle = $ %.f\n',Sys_Tot_rev)
fprintf('Lost opportunity cost to total revenues = %.4f\n',Per_lost_opp)

% Calculate vector of COTS unit prices at the cost breakpoints
```

```
Cust_unit_cost = Cust_COGS_cost + Cust_mod_p + (Cust_NRE_cost ./N);
U1 = Cust_unit_cost;

% --------------------------------------------------------------------------
%   Find the crossover point in unit costs between COTS and custom-build
% --------------------------------------------------------------------------
find_crossover = Cust_unit_cost - COTS_unit_cost;   % First order crossover
find_cross1 = zeros(1,N_indx(1,2));
find_cross2 = zeros(1,N_indx(1,2));
find_cross3 = zeros(1,N_indx(1,2));
% Find interval where crossover occurs, depends on whether custom-build
% is decreasing or increasing
cross_point = diff(sign(find_crossover));
find_cross1(1,1:(N_indx(1,2)-1)) = cross_point;
find_cross2(1,2:N_indx(1,2)) = cross_point; % Interval end if decreasing
cross_point(1:1) = [];
find_cross3(1,1:(N_indx(1,2)-2)) = cross_point;% Interval end if increasing
check_cross = sum(find_cross1);             % Does crossover exist?

if check_cross == 0                         % =0, then no crossover
    fprintf('No crossover point found\n')
    crossover_i = [0 0];
    iCOTSmodp = max(COTS_mod_p);
    iCOTSCcost = max(COTS_COGS_cost);
    iCustmodp = max(Cust_mod_p);
    iCustCcost = max(Cust_COGS_cost);
else                                        % Find crossover indices
    crossover_i = [0 0];
    select_i1 = N.*(find_cross1/check_cross);
    select_i2 = N.*(find_cross2/check_cross);
    select_i3 = N.*(find_cross3/check_cross);
    if check_cross < 0              % - means custom-build is decreasing
        crossover_i(1,1) = max(select_i1);
        crossover_i(1,2) = max(select_i2);
        % Calculate the module and COGS costs for this decreasing interval
        iCOTSmodp = max(COTS_mod_p.*(select_i1/crossover_i(1,1)));
        iCOTSCcost = max(COTS_COGS_cost.*(select_i1/crossover_i(1,1)));
        iCustmodp = max(Cust_mod_p.*(select_i1/crossover_i(1,1)));
        iCustCcost = max(Cust_COGS_cost.*(select_i1/crossover_i(1,1)));
    else                            % + means custom-build is increasing
        crossover_i(1,1) = max(select_i3);
        crossover_i(1,2) = max(select_i1);
        % Calculate the module and COGS costs for this increasing interval
        iCOTSmodp = max(COTS_mod_p.*(select_i3/crossover_i(1,1)));
        iCOTSCcost = max(COTS_COGS_cost.*(select_i3/crossover_i(1,1)));
        iCustmodp = max(Cust_mod_p.*(select_i3/crossover_i(1,1)));
        iCustCcost = max(Cust_COGS_cost.*(select_i3/crossover_i(1,1)));
    end
end

% --------------------------------------------------------------------------
%   Calcualte the crossover point in the selected interval
% --------------------------------------------------------------------------
M = crossover_i(1,1):1:crossover_i(1,2);    % Interval with the crossover
% Form vectors of the module and COGS costs for both COTS and custom-build
iCOTS_mod_p = ones(size(M))*iCOTSmodp;
```

```
iCOTS_COGS_cost = ones(size(M))*iCOTSCcost;
iCust_mod_p = ones(size(M))*iCustmodp;
iCust_COGS_cost = ones(size(M))*iCustCcost;
% Calculate the unit costs and then find the difference between
% COTS and custom-build
iCOTS_unit_cost = iCOTS_COGS_cost + iCOTS_mod_p + (COTS_NRE_cost./M);
iCust_unit_cost = iCust_COGS_cost + iCust_mod_p + (Cust_NRE_cost./M);

idiff_unit_cost = iCust_unit_cost - iCOTS_unit_cost;
cost_diff = abs(idiff_unit_cost);
icross_point = zeros(size(M));
icross_point(1:length(M)-1) = diff(sign(idiff_unit_cost));
icross_val = sum(icross_point);
break_point = sum(M.*(icross_point/icross_val));
COTS_cross_mod_cost = sum(iCOTS_unit_cost.*(icross_point/icross_val));
Cust_cross_mod_cost = sum(iCust_unit_cost.*(icross_point/icross_val));

% -------------------------------------------------------------------------
%   Print crossover point and module costs
% -------------------------------------------------------------------------
fprintf('\n')
fprintf('Crossover point (number of modules) = %.f\n',break_point)
fprintf('Crossover COTS module cost = $ %.f\n',COTS_cross_mod_cost)
fprintf('Crossover custom-build module cost = $ %.f\n',Cust_cross_mod_cost)

plot(M,iCOTS_unit_cost,'b-')
hold on
plot(M,iCust_unit_cost,'r-')
plot(M,cost_diff,'g-')
ylabel('Per Unit Costs ($)')
xlabel('Numbers of modules')
title('Buy versus Build Crossover')
legend('COTS','build','difference')
grid
fprintf('\n')
fprintf('Press spacebar\n')
pause
hold off

% /////////////////////////////////////////////////////////////////////////
% *************************************************************************
%   This begins the actual Monte Carlo simulation section
% *************************************************************************
% /////////////////////////////////////////////////////////////////////////
sim_run = 100000;                % Sets the number of Monte Carlo runs
fprintf('# of simulation runs = %.f\n',sim_run)

data_mat = zeros(sim_run, 25);  % initialize the data matrix for analysis

% -------------------------------------------------------------------------
%   Add random dither to set up values within vectors
% -------------------------------------------------------------------------

COTS_spec_diff = COTS_spec_hi - COTS_spec_lo; % Vary specification
iCOTS_spec = COTS_spec_lo + (COTS_spec_diff .* rand(sim_run,1));
```

```
Qual_doc_diff = Qual_doc_hi - Qual_doc_lo;    % Vary quality vendor support
iCOTS_vend_doc = Qual_doc_lo + (Qual_doc_diff .* rand(sim_run,1));

COTS_ve_su_diff = COTS_ve_su_hi - COTS_ve_su_lo;    % Vary vendor support
iCOTS_vend_sup = COTS_ve_su_lo + (COTS_ve_su_diff .* rand(sim_run,1));

COTS_te_su_diff = COTS_te_su_hi - COTS_te_su_lo;    % Vary customer support
iCOTS_tech_sup = COTS_te_su_lo + (COTS_te_su_diff .* rand(sim_run,1));

Cust_ex_diff = Cust_ex_hi - Cust_ex_lo;             % Vary expertise
iCust_expert = Cust_ex_lo + (Cust_ex_diff .* rand(sim_run,1));

Cust_re_diff = Cust_re_hi - Cust_re_lo;             % Vary resources
iCust_resource = Cust_re_lo + (Cust_re_diff .* rand(sim_run,1));

sys_life_diff = sys_life_hi - sys_life_lo;          % Vary system life
iSystem_life = sys_life_lo + (sys_life_diff .* rand(sim_run,1));

COTS_life_diff = COTS_life_hi - COTS_life_lo;       % Vary COTS life
iCOTS_life = COTS_life_lo + (COTS_life_diff .* rand(sim_run,1));

% vary vendor charges from 0 to max. life of system for both development
% team support and for support to final customer
iCOTS_team_cost = zeros(sim_run, 1);    % Initialize development team supp.
iCOTS_fi_cu_cost = zeros(sim_run, 1);   % Initialize final customer support
cap = ceil(iSystem_life);   % Discretize life into years for max. support
for i = 1:sim_run
    iCOTS_team_cost(i,1) = (randi([0,cap(i,1)])) * COTS_team_sup;
    iCOTS_fi_cu_cost(i,1) = (randi([0,cap(i,1)])) * Final_customer_sup_cost;
end

hist(iCOTS_spec(:,1),100)
grid
ylabel('Number of counts per bin')
xlabel('Bin number represents fraction of specs covered by COTS')
title('COTS Coverage of Specifications')
fprintf('\n')
fprintf('Press spacebar\n')
pause

hist(iCOTS_vend_doc(:,1),100)
grid
ylabel('Number of counts per bin')
xlabel('Bin number represents fraction of quality of COTS support')
title('Quality of Vendor Support')
fprintf('\n')
fprintf('Press spacebar\n')
pause

hist(iCOTS_vend_sup(:,1),100)
grid
ylabel('Number of counts per bin')
xlabel('Bin number represents fraction of support to team (team NRE)')
title('Fraction of Necessary Vendor-Supplied Support to Team')
```

```
fprintf('\n')
fprintf('Press spacebar\n')
pause

hist(iCOTS_tech_sup(:,1),100)
grid
ylabel('Number of counts per bin')
xlabel('Bin number represents fraction of support to customer (team NRE)')
title('Fraction of Necessary Vendor Support to Final Customer')
fprintf('\n')
fprintf('Press spacebar\n')
pause

hist(iCOTS_team_cost(:,1),100)
grid
ylabel('Number of counts per bin')
xlabel('Bin number represents cost of vendor support ($)')
title('Cost for Extra Vendor Support to Development Team')
fprintf('\n')
fprintf('Press spacebar\n')
pause

hist(iCOTS_fi_cu_cost(:,1),100)
grid
ylabel('Number of counts per bin')
xlabel('Bin number represents cost of vendor support ($)')
title('Cost for Extra Vendor Support to Final Customer')
fprintf('\n')
fprintf('Press spacebar\n')
pause

hist(iCust_expert(:,1),100)
grid
ylabel('Number of counts per bin')
xlabel('Bin number represents fraction of necessary expertise')
title('Spread of Expertise for Custom Design')
fprintf('\n')
fprintf('Press spacebar\n')
pause

hist(iCust_resource(:,1),100)
grid
ylabel('Number of counts per bin')
xlabel('Bin number represents fraction of necessary resources')
title('Spread of Resources for Custom Design')
fprintf('\n')
fprintf('Press spacebar\n')
pause

hist(iSystem_life(:,1),100)
grid
ylabel('Number of counts per bin')
xlabel('Bin number represents system life')
title('System Lifetime')
fprintf('\n')
fprintf('Press spacebar\n')
```

```
pause

hist(iCOTS_life(:,1),100)
grid
ylabel('Number of counts per bin')
xlabel('Bin number represents COTS life')
title('COTS lifetime')
fprintf('\n')
fprintf('Press spacebar\n')
pause

fprintf('\n')
if (sim_run <= 1000001) && (sim_run > 100000)
    fprintf('       !!!Now WAIT between 4 and 40 minutes!!!\n')
    fprintf('\n')
end
if (sim_run <= 100000) && (sim_run > 40000)
    fprintf('       !!!Now WAIT between 1.5 and 4 minutes!!!\n')
    fprintf('\n')
end
if sim_run <= 40000
    fprintf('       !!!Now WAIT less than 90 seconds!!!\n')
    fprintf('\n')
end

tic                                  % Start timer
% ------------------------------------------------------------------------
%   Multiply, element by element, the vector of insufficient or incorrect
%   vendor support or documentation with the vector for insufficient COTS
%   specifications. This operation calculates the factor of increased
%   COTS cost when inserted into the sigmoid Logistic Function.
% ------------------------------------------------------------------------
filter_spec1 = iCOTS_spec .* iCOTS_vend_doc;
filter_spec2 = (1 - iCOTS_vend_sup) .* (1 - iCOTS_tech_sup);
filter_spec = 1 - (filter_spec1 .* filter_spec2);

% ------------------------------------------------------------------------
%   Use the sigmoid curve of the Logistic Function to calculate when
%   both insufficient COTS specifications and COTS support will increase
%   COTS cost and when both support to the team and the customer are
%   required in some measure.
% ------------------------------------------------------------------------

% L factor for sigmoid, it scales the final output to not exceed the
% custom effort.
L_factor = Cust_Tot_effort/COTS_Tot_effort;

% Sets steepness of the sigmoid
k_stp_lo = 7;                       % Sets lower limit on steepness
k_stp_hi = 25;                      % Sets upper limit on steepness
k_diff = k_stp_hi - k_stp_lo;       % Sets the difference between limits
k_steep = k_stp_lo + (k_diff .* rand(sim_run,1));

% Sets midpoint of the sigmoid
x0sig_lo = 0.2;                     % Sets lower limit on midpoint
```

```
x0sig_hi = 0.5;                 % Sets upper limit on midpoint
x0_diff = x0sig_hi - x0sig_lo;  % Sets the difference between limits
x0sigmoid = x0sig_lo + (x0_diff .* rand(sim_run,1));

% Calculate the offset vector when x = 0
filter_zero = L_factor./(1 + exp(k_steep .* x0sigmoid));

% Calculate vector of corrections to the COTS NRE effort to address
% insufficient specifications
COTS_cor = (L_factor./(1 + exp(k_steep .* (x0sigmoid-filter_spec)))) -
filter_zero + 1;
% Calculate vector of time corrections; assume that the corrected effort is
% split evening between the staff FTEs and the calendar time
COTS_time_cor = sqrt(COTS_cor);

% -------------------------------------------------------------------------
%   Multiply, element by element, the vector of team expertise and
%   available company resources. This operation calculates the factor of
%   increased custom design cost when inserted into the Logistic Function.
% -------------------------------------------------------------------------
filter_spec = 1 - (iCust_expert .* iCust_resource);

% -------------------------------------------------------------------------
%   Use the sigmoid curve of the Logistic Function to calculate when
%   both the expertise of the development team and available resources
%   will increase the NRE design cost and time.
% -------------------------------------------------------------------------

% L factor scales the final output to not exceed a 3 times (see +1 below).
L_factor = 2;

% Sets steepness of the sigmoid between 7 and 25 from COTS curve parameters
k_steep = k_stp_lo + (k_diff .* rand(sim_run,1));

% Sets midpoint of the sigmoid between 0.2 and 0.5 from COTS parameters
x0sigmoid = x0sig_lo + (x0_diff .* rand(sim_run,1));

% Calculate the offset vector when x = 0
filter_zero = L_factor./(1 + exp(k_steep .* x0sigmoid));

% Calculate vector of corrections to the COTS NRE effort to address
% insufficient specifications
Cust_cor = (L_factor./(1 + exp(k_steep .* (x0sigmoid-filter_spec)))) -
filter_zero + 1;
% Calculate vector of time corrections; assume that the corrected effort is
% split evening between the staff FTEs and the calendar time
Cust_time_cor = sqrt(Cust_cor);

% -------------------------------------------------------------------------
%   Set up "importance sampling" for the two Rayleigh random variables
% -------------------------------------------------------------------------
u = [0.1 0.3 0.5 0.7 0.9];
v = [0.1 0.3 0.5 0.7 0.9];
mlp = numel(u);                             % loop index 1
nlp = numel(v);                             % loop index 2
```

```
a = 1;                                      % the Rayleigh variance
b = 1;                                      % the Rayleigh variance
x = a .* sqrt(-2 .* log(u));                % samples of random variable, x
y = b .* sqrt(-2 .* log(v));                % samples of random variable, y
p = (x./(a^2)).*(exp(-(x.^2)./(2*(a^2))));  % probability of x
q = (y./(b^2)).*(exp(-(y.^2)./(2*(b^2))));  % probability of y
off_set = 0.8;                              % Rayleigh offset
x_offset = x + off_set;
y_offset = y + off_set;

% -------------------------------------------------------------------------
%   Prepare filter mask for different inventory premiums: none, premium,
%   or buy ROTS.
% -------------------------------------------------------------------------
ilife_ratio = iCOTS_life ./ iSystem_life;
mat_a = sign(ilife_ratio - 1);       % Find all values >= 1.0
mat_b = round((mat_a + 1.6)/2);      % Normalize to 1
mat_c = (mat_b - 1)/-1;              % Complement the mask

mat_b = mat_b .* inven_prem1;
mat_c = mat_c .* inven_prem2;

inven_adj = mat_b + mat_c;

% -------------------------------------------------------------------------
%   Prepare an adjustment vector for additional costs charged by COTS
%   vendor to support the development team or the final customer or both.
% -------------------------------------------------------------------------
COTS_adjust = iCOTS_team_cost + iCOTS_fi_cu_cost;

% /////////////////////////////////////////////////////////////////////////
% -------------------------------------------------------------------------
%   Begin simulation runs
% -------------------------------------------------------------------------
% /////////////////////////////////////////////////////////////////////////

no_break = 0;   % Initialize variable to record number of no cross overs

for i = 1:sim_run

% -------------------------------------------------------------------------
%   Begin dual, embedded loops for "importance sampling"
% -------------------------------------------------------------------------
    tot_sum = zeros(9,1);   % initialize the importance sampling sum
    norm_sum = 0;           % initialize the normalizing sum
    Y = zeros(9,1);         % initialize the crossover point variables
    z = zeros(mlp,nlp,9);   % initialize the matrix for collecting partial
                            % values from model
    for ilp = 1:mlp
        for jlp = 1:nlp
            % These two lines use Rayleigh distributed random variables to
            % adjust the value of the estimated time for design, in both
            % COTS and custom design.
            iCOTS_Calendar = COTS_Calendar * x_offset(ilp);
            iCust_Calendar = Cust_Calendar * y_offset(jlp);
```

```
% --------------------------------------------------------------------------
%   First, calculate the COTS time, staff effort, and unit costs
% --------------------------------------------------------------------------
COTS_person_month = (COTS_NRE_FTE * iCOTS_Calendar).* COTS_cor(i,1);
COTS_Tot_effort = sum(COTS_person_month);
% Total NRE labor cost for project ($), assume 168 hrs/month
COTS_Tot_labor = Dev_labor * COTS_person_month * 168;
% Total calendar time for NRE (months)
COTS_Tot_time = (sum(iCOTS_Calendar)) * COTS_time_cor(i,1);

% Calculate total COTS costs from NRE, tooling, compliance/certification
% and vendor suppor for the team and final customer
COTS_NRE_cost = COTS_Tot_labor + COTS_tool_cost + COTS_cert_cost +
COTS_adjust(i,1);

% Calculate prices for COTS modules with discounts and inventory premium
COTS_mod_p = COTS_discount * COTS_price * inven_adj(i,1);

% Set up COTS cost vector from total COGS cost
COTS_COGS_cost = ones(1,N_indx(1,2)) * COTS_COGS_tot_cost;

% Calculate vector of COTS unit prices at the cost breakpoints
COTS_unit_cost = COTS_COGS_cost + COTS_mod_p + (COTS_NRE_cost ./N);

% --------------------------------------------------------------------------
%   Second, calculate the custom-build time, staff effort, and unit costs
% --------------------------------------------------------------------------
Cust_person_month = (Cust_NRE_FTE * iCust_Calendar) .* Cust_cor(i,1);
Cust_Tot_effort = sum(Cust_person_month);
% Total NRE labor cost for project ($), assume 168 hrs/month
Cust_Tot_labor = Dev_labor * Cust_person_month * 168;
% Total calendar time for NRE (months)
Cust_Tot_time = (sum(iCust_Calendar)) * Cust_time_cor(i,1);

% Calculate build costs from NRE, tooling, and compliance/certification
Cust_NRE_cost = Cust_Tot_labor + Cust_tool_cost + Cust_cert_cost;

% Calculate prices for custom-built modules with discounts and inventory
Cust_mod_p = Comp_discount * Comp_price * inven_prem1;

% Set up custom-build cost vector from total COGS cost
Cust_COGS_cost = ones(1,N_indx(1,2)) * Cust_COGS_tot_cost;

% Calculate the lost opportunity costs
lost_opp_cost = (Cust_Tot_time - COTS_Tot_time) * Early_sales_rate *
Sys_revenue;

% Calculate vector of custom unit prices at the cost breakpoints
Cust_unit_cost = Cust_COGS_cost + Cust_mod_p + (Cust_NRE_cost ./N);

% --------------------------------------------------------------------------
%   Find the crossover point in unit costs between COTS and custom-build
% --------------------------------------------------------------------------
```

```
find_crossover = Cust_unit_cost - COTS_unit_cost;   % First order crossover
find_cross1 = zeros(1,N_indx(1,2));
find_cross2 = zeros(1,N_indx(1,2));
cross_point = diff(sign(find_crossover));
find_cross1(1,1:(N_indx(1,2)-1)) = cross_point;
find_cross2(1,2:N_indx(1,2)) = cross_point;    % Interval end if decreasing
check_cross = sum(find_cross1);         % Does crossover exist?

if check_cross == 0                 % =0, then no crossover
    no_break = no_break + 1;        % increment count of no crossovers
    break_point = 0;                % zero out indication of crossover
    if find_crossover(1,1) <= 0     % COTS and custom costs to first unit
        COTS_cross_mod_cost = COTS_unit_cost(1,1);
        Cust_cross_mod_cost = Cust_unit_cost(1,1);
    else                            % COTS and custom costs to last unit
        COTS_cross_mod_cost = COTS_unit_cost(1,N_indx(1,2));
        Cust_cross_mod_cost = Cust_unit_cost(1,N_indx(1,2));
    end
else                                % Find crossover indices
crossover_i = [0 0];
select_i1 = N.*(find_cross1/check_cross);
select_i2 = N.*(find_cross2/check_cross);
crossover_i(1,1) = max(select_i1);
crossover_i(1,2) = max(select_i2);

% Calculate the module and COGS costs for this decreasing interval
iCOTSmodp = max(COTS_mod_p.*(select_i1/crossover_i(1,1)));
iCOTSCcost = max(COTS_COGS_cost.*(select_i1/crossover_i(1,1)));
iCustmodp = max(Cust_mod_p.*(select_i1/crossover_i(1,1)));
iCustCcost = max(Cust_COGS_cost.*(select_i1/crossover_i(1,1)));

iendCOTSmodp = max(COTS_mod_p.*(select_i2/crossover_i(1,2)));
iendCOTSCcost = max(COTS_COGS_cost.*(select_i2/crossover_i(1,2)));
iendCustmodp = max(Cust_mod_p.*(select_i2/crossover_i(1,2)));
iendCustCcost = max(Cust_COGS_cost.*(select_i2/crossover_i(1,2)));

% -------------------------------------------------------------------------
%   Calculate the crossover point in the selected interval
% -------------------------------------------------------------------------
M = crossover_i(1,1):1:crossover_i(1,2);    % Interval with the crossover
% Form vectors of the module and COGS costs for both COTS and custom-build.
% Must add discount break at last point in each vector
iCOTS_mod_p = ones(size(M))*iCOTSmodp;
iCOTS_mod_p(1,length(M)) = iendCOTSmodp;    % COTS end with discount break
iCOTS_COGS_cost = ones(size(M))*iCOTSCcost;
iCOTS_COGS_cost(1,length(M)) = iendCOTSCcost;
iCust_mod_p = ones(size(M))*iCustmodp;
iCust_mod_p(1,length(M)) = iendCustmodp;    % Custom end with discount break
iCust_COGS_cost = ones(size(M))*iCustCcost;
iCust_COGS_cost(1,length(M)) = iendCustCcost;
% Calculate the unit costs and then find the difference between
% COTS and custom-build
iCOTS_unit_cost = iCOTS_COGS_cost + iCOTS_mod_p + (COTS_NRE_cost./M);
iCust_unit_cost = iCust_COGS_cost + iCust_mod_p + (Cust_NRE_cost./M);
```

```
idiff_unit_cost = iCust_unit_cost - iCOTS_unit_cost;
icross_point = zeros(size(M));
icross_point(1:length(M)-1) = diff(sign(idiff_unit_cost));
icross_val = sum(icross_point);

unity_cross = icross_point/icross_val;
break_point = sum(M.*(unity_cross));
COTS_cross_mod_cost = sum(iCOTS_unit_cost.*(unity_cross));
Cust_cross_mod_cost = sum(iCust_unit_cost.*(unity_cross));
end     % this is the end of the if-statement to check for no cross over

% Adjust these 9 variable outputs to a uniform sampling from the Rayleigh
% distribution and two random variables
        z(ilp,jlp,1) = break_point;         % Crossover number of modules
        z(ilp,jlp,2) = COTS_Tot_effort;     % COTS effort at crossover
        z(ilp,jlp,3) = Cust_Tot_effort;     % Custom effort at crossover
        z(ilp,jlp,4) = COTS_cross_mod_cost; % COTS module cost at crossover
        z(ilp,jlp,5) = Cust_cross_mod_cost; % Custom module cost at crossover
        z(ilp,jlp,6) = COTS_Tot_time;       % Total NRE time for COTS
        z(ilp,jlp,7) = Cust_Tot_time;       % Total NRE time for custom
        z(ilp,jlp,8) = lost_opp_cost;       % Lost opportunity cost
        z(ilp,jlp,9) = lost_opp_cost/Sys_Tot_rev;

        tot_sum(:,1) = tot_sum(:,1) + ((p(ilp)*q(jlp)).*z(ilp,jlp));
        norm_sum = norm_sum + (p(ilp)*q(jlp));
    end
end

Y(:,1) = tot_sum(:,1) ./ norm_sum;

% Add data to the data matrix for later analysis in another function
data_mat(i,1:9) = Y(:,1);

end         % This is the end of the big sim_run loop

% Store vectors of results in data matrix for later analysis
data_mat(:,10) = ilife_ratio;      % COTS to custom life ratio
data_mat(:,11) = inven_adj;        % adjustment to COTS inventory premium
data_mat(:,12) = COTS_cor;         % correction to COTS effort and time

data_mat(:,13) = iCOTS_spec;       % percent COTS covering required specs
data_mat(:,14) = iCOTS_vend_doc;   % quality of vendor support/documentation
data_mat(:,15) = iCOTS_vend_sup;   % percent of vendor support needed by
                                   % development team (team NRE)
data_mat(:,16) = iCOTS_tech_sup;   % percent of vendor support needed to
                                   % supply to final customer (team NRE)
data_mat(:,17) = iCOTS_team_cost;  % cost of vendor support needed to
                                   % supply to development team ($)
data_mat(:,18) = iCOTS_fi_cu_cost; % cost of vendor support needed to
                                   % supply to final customer ($)
data_mat(:,19) = COTS_adjust;      % Final cost of vendor support ($)

data_mat(:,20) = Cust_cor;         % correction to custom effort and time
data_mat(:,21) = iCust_expert;     % percent of necessary expertise
                                   % available for custom design
```

```
data_mat(:,22) = iCust_resource;    % percent of necessary resources
                                    % available for custom design

toc             % Stop timer and display compute time

hist(data_mat(:,1),1000)
grid
ylabel('Number of counts per bin')
xlabel('Bin number represents cross over point')
title('Cross Over Study: Build Cheaper than Buy COTS')
fprintf('\n')
fprintf('Press spacebar\n')
pause
```

Appendix G - Build-versus-Buy Parameter Sensitivity Results

Overview of Simulation Operations

I programmed two primary routines. The first routine was a Matlab m-file script and it provided the basic input values for all 57 parameters; it had the general format for its function call of `init_param_(case study name)`. These initial parameter routines are described in Appendix F.

The second routine was a Matlab m-file script and it used the 57 parameters in a nearly identical model which only passed data through in one pass; it did not run a Monte Carlo simulation. A sensitivity analysis program called eFAST, which is iterative, calls the model in each iteration. eFAST usually ran for 10,000 iterations; it was then repeated for a total of 50 runs of the routine for a total of 500,000 iterations overall. For each iteration, eFAST generates a random value for each of the 14 parameters in the model that calculates the crossover point between custom built units and COTS units. The parameters are:

X(1) = COTS specification

X(2) = COTS quality of support adjustment

X(3) = COTS vendor supplied support to development team requiring team NRE

X(4) = COTS vendor supplied support to final customer requiring team NRE

X(5) = team expertise

X(6) = team resources

X(7) = system life

X(8) = COTS life

X(9) = cost of COTS vendor supplied support to development team

X(10) = cost of COTS vendor supplied support to final customer

X(11) = steepness of sigmoid to set effort for deficient COTS

X(12) = midpoint of sigmoid to set effort for deficient COTS

X(13) = steepness of sigmoid to set custom effort

X(14) = midpoint of sigmoid to set custom effort

After the eFAST algorithm finished its iterations and runs, the second routine then prints out values, generate variables, and placed them in the workspace for other routines to analyze or plot specific values. The outputs for the 18 case studies follow this section.

Finally, I have printed the different routines used in the sensitivity analyses.

Simulation Results for Sensitivity

Table 8.3 contains the sensitivity analyses for Case Studies 2 through 18. Case Study 1 did not have any crossover points, so it had no parameter analysis.

Table 8.3 Compilation of sensitivity results from 17 case studies.

Parameter Sensitivity - mean

	Appliance RTOS	Lab RTOS	Medical RTOS	6K v 900	25K v 4K	POL
X1	N/A	0.0084386519	0.0079716885	0.0075895779	0.0074166860	0.0021747341
X2	N/A	0.0081589508	0.0076508551	0.0076104986	0.0074706208	0.0021006934
X3	N/A	0.0081738973	0.0078253697	0.0077137210	0.0074559616	0.0021812357
X4	N/A	0.0082924049	0.0079805974	0.0075932426	0.0074096331	0.0021696324
X5	N/A	0.2903033419	0.2915014281	0.2964112082	0.2908847812	0.2045516010
X6	N/A	0.2929528340	0.2917973668	0.2954782901	0.2911463303	0.2040786259
X7	N/A	0.0000125265	0.0000350224	0.0000386417	0.0000367325	0.0000500116
X8	N/A	0.0000000501	0.0000000527	0.0000000518	0.0000001025	0.0000476491
X9	N/A	0.0001256035	0.0001053433	0.0001172476	0.0001115301	0.0000436923
X10	N/A	0.0000000603	0.0000000439	0.0001178547	0.0001121687	0.0000478616
X11	N/A	0.0050990418	0.0047157923	0.0048445163	0.0043975463	0.0000743806
X12	N/A	0.0082986964	0.0075749823	0.0079163185	0.0076761721	0.0021033166
X13	N/A	0.0080063679	0.0090061226	0.0067117124	0.0108502706	0.0155710437
X14	N/A	0.2137791910	0.2174446178	0.2154878253	0.2161523426	0.1562630342

N/A = no crossover for the parameters in this case study

Parameter Sensitivity - standard deviation

	Appliance RTOS	Lab RTOS	Medical RTOS	6K v 900	25K v 4K	POL
X1	N/A	0.0007351522	0.0006131867	0.0006126059	0.0006236028	0.0009567924
X2	N/A	0.0006588940	0.0006273840	0.0006677844	0.0005495094	0.0012483959
X3	N/A	0.0005650675	0.0006182655	0.0005662531	0.0005848508	0.0009861214
X4	N/A	0.0005674327	0.0006094615	0.0005897167	0.0006008657	0.0009421100
X5	N/A	0.0048146164	0.0044714143	0.0050007300	0.0054263555	0.0086230774
X6	N/A	0.0055794571	0.0061972743	0.0056127235	0.0066570947	0.0057061803
X7	N/A	0.0000010456	0.0000020375	0.0000019428	0.0000020472	0.0000202629
X8	N/A	0.0000000335	0.0000000364	0.0000000390	0.0000000683	0.0000278588
X9	N/A	0.0000041228	0.0000034967	0.0000043138	0.0000048483	0.0000248419
X10	N/A	0.0000000370	0.0000000344	0.0000041536	0.0000054217	0.0000296923
X11	N/A	0.0001935703	0.0002104093	0.0001823183	0.0001889821	0.0000487135
X12	N/A	0.0015955581	0.0014507962	0.0014985445	0.0013333840	0.0013131257
X13	N/A	0.0001969063	0.0001992431	0.0001505890	0.0002362302	0.0013040747
X14	N/A	0.0051466662	0.0043716206	0.0050453752	0.0059656661	0.0110813990

Parameter Total Sensitivity - mean

	Appliance RTOS	Lab RTOS	Medical RTOS	6K v 900	25K v 4K	POL
X1	N/A	0.0256658705	0.0249105024	0.0230785841	0.0241121497	0.1253430567
X2	N/A	0.0252099929	0.0241874187	0.0230794358	0.0241333215	0.1251737107
X3	N/A	0.0251623470	0.0246533207	0.0234038933	0.0240326945	0.1248671268
X4	N/A	0.0254079532	0.0250229729	0.0230465947	0.0239651144	0.1260149017
X5	N/A	0.3798046484	0.3831443479	0.3869294004	0.3862954485	0.5190784007
X6	N/A	0.3834000424	0.3834326447	0.3857472807	0.3864888820	0.5198367638
X7	N/A	0.0019511942	0.0020366109	0.0019888046	0.0022479618	0.0660173014
X8	N/A	0.0019555690	0.0019809033	0.0019364829	0.0021794249	0.0662487649
X9	N/A	0.0020819519	0.0021181027	0.0020339855	0.0022953721	0.0653066304
X10	N/A	0.0019434355	0.0020098741	0.0020605636	0.0023019293	0.0669382459
X11	N/A	0.0093169713	0.0090736536	0.0088048746	0.0089814620	0.0957577448
X12	N/A	0.0254905667	0.0238304984	0.0237201391	0.0244251757	0.1259137934
X13	N/A	0.0392604350	0.0406952545	0.0372074036	0.0429261663	0.1160125615
X14	N/A	0.3003275309	0.3076092815	0.3014593921	0.3097015443	0.4411827890

Parameter Total Sensitivity - standard deviation

	Appliance RTOS	Lab RTOS	Medical RTOS	6K v 900	25K v 4K	POL
X1	N/A	0.0016766859	0.0014076420	0.0013159053	0.0014361531	0.0076479811
X2	N/A	0.0014878196	0.0014795847	0.0015439976	0.0013991932	0.0086129283
X3	N/A	0.0012276102	0.0013971138	0.0011970538	0.0013066565	0.0076785537
X4	N/A	0.0012398834	0.0014326959	0.0013985605	0.0013833085	0.0073329383
X5	N/A	0.0063061365	0.0060752588	0.0064858108	0.0071985494	0.0127986785
X6	N/A	0.0073680753	0.0081483610	0.0073415649	0.0089285064	0.0093096900
X7	N/A	0.0001094614	0.0001216113	0.0001016885	0.0001324399	0.0035285757
X8	N/A	0.0001205884	0.0000989945	0.0001217385	0.0001556980	0.0032795877
X9	N/A	0.0001031413	0.0001069896	0.0000931897	0.0001298372	0.0035591225
X10	N/A	0.0001089144	0.0001295812	0.0000880032	0.0001255070	0.0039089499
X11	N/A	0.0003375678	0.0004172184	0.0003002855	0.0004068205	0.0054384794
X12	N/A	0.0038921423	0.0035072426	0.0034788142	0.0034536976	0.0093941595
X13	N/A	0.0011167738	0.0011745052	0.0009202626	0.0011343105	0.0051746485
X14	N/A	0.0071041915	0.0062519336	0.0068844840	0.0083503973	0.0188874982

Normalized Mean for Total Sensitivity

	Appliance RTOS	Lab RTOS	Medical RTOS	6K v 900	25K v 4K	POL
X1	N/A	0.0206	0.0199	0.0185	0.0191	0.0485
X2	N/A	0.0202	0.0193	0.0185	0.0191	0.0484
X3	N/A	0.0202	0.0196	0.0188	0.0190	0.0483
X4	N/A	0.0204	0.0199	0.0185	0.0190	0.0488
X5	N/A	0.3046	0.3054	0.3109	0.3056	0.2009
X6	N/A	0.3075	0.3056	0.3100	0.3057	0.2012
X7	N/A	0.0016	0.0016	0.0016	0.0018	0.0256
X8	N/A	0.0016	0.0016	0.0016	0.0017	0.0256
X9	N/A	0.0017	0.0017	0.0016	0.0018	0.0253
X10	N/A	0.0016	0.0016	0.0017	0.0018	0.0259
X11	N/A	0.0075	0.0072	0.0071	0.0071	0.0371
X12	N/A	0.0204	0.0190	0.0191	0.0193	0.0487
X13	N/A	0.0315	0.0324	0.0299	0.0340	0.0449
X14	N/A	0.2408	0.2452	0.2422	0.2450	0.1708

N/A = no crossover for the parameters in this case study

Parameter Sensitivity - mean

	NSS ROTS	NSS iPad	AFD PIC	AFD ARM	AFD Atom - 40%	AFD Atom - 20/40%
X1	0.0023707733	0.0005371396	0.0074352028	0.0073566036	0.0074610708	0.0164583542
X2	0.0022518513	0.0005618568	0.0072908526	0.0074440532	0.0074509087	0.0165411769
X3	0.0020652932	0.0005428490	0.0074156551	0.0074726759	0.0073540246	0.0165080588
X4	0.0023897429	0.0005398224	0.0072449024	0.0074344431	0.0072856108	0.0164724419
X5	0.1256415368	0.0344851250	0.2942620884	0.2925564034	0.2927663461	0.2700361825
X6	0.1244911829	0.0340243245	0.2950267465	0.2927683536	0.2939065684	0.2697266067
X7	0.0000240971	0.0002035451	0.0000000529	0.0000000607	0.0000000572	0.0000000447
X8	0.0000253474	0.0001902451	0.0000000441	0.0000000504	0.0000000646	0.0000000391
X9	0.0000245557	0.0001937177	0.0000000446	0.0000000589	0.0000000606	0.0000000379
X10	0.0000234003	0.0002013599	0.0000000520	0.0000000584	0.0000000668	0.0000000456
X11	0.0028006926	0.0016050054	0.0045757589	0.0045416527	0.0044745814	0.0005092934
X12	0.0025317707	0.0004934076	0.0075786211	0.0076701310	0.0072630986	0.0708244112
X13	0.0475193217	0.0513959683	0.0079423863	0.0096928707	0.0091663949	0.0063072833
X14	0.0616855726	0.0304573872	0.2156877939	0.2167191952	0.2163374149	0.1970566023

Parameter Sensitivity - standard deviation

	NSS ROTS	NSS iPad	AFD PIC	AFD ARM	AFD Atom - 40%	AFD Atom - 20/40%
X1	0.0008715419	0.0003046572	0.0005859513	0.0005079945	0.0005324501	0.0003733787
X2	0.0009335914	0.0003613173	0.0004811073	0.0005601652	0.0005762155	0.0004573348
X3	0.0007762308	0.0003607580	0.0005048646	0.0005536650	0.0006096992	0.0003854201
X4	0.0009196077	0.0003662349	0.0005513655	0.0005306543	0.0005998423	0.0004347887
X5	0.0091717716	0.0029879456	0.0052668245	0.0067134696	0.0067055307	0.0046429609
X6	0.0080332625	0.0033965048	0.0056301381	0.0055324023	0.0052393874	0.0044813655
X7	0.0000158916	0.0001180108	0.0000000349	0.0000000410	0.0000000354	0.0000000311
X8	0.0000133770	0.0001135695	0.0000000276	0.0000000367	0.0000000444	0.0000000350
X9	0.0000117723	0.0001170011	0.0000000281	0.0000000437	0.0000000414	0.0000000298
X10	0.0000114339	0.0001217235	0.0000000318	0.0000000428	0.0000000418	0.0000000358
X11	0.0007229564	0.0006474723	0.0001630797	0.0001854826	0.0001945729	0.0000465358
X12	0.0010167764	0.0004049989	0.0012380962	0.0014614495	0.0014666277	0.0028666345
X13	0.0031012301	0.0048806873	0.0001892310	0.0001956965	0.0002147521	0.0001655896
X14	0.0070324377	0.0055824327	0.0043666935	0.0044928921	0.0044470292	0.0030160340

Parameter Total Sensitivity - mean

	NSS ROTS	NSS iPad	AFD PIC	AFD ARM	AFD Atom - 40%	AFD Atom - 20/40%
X1	0.1096790943	0.3637307675	0.0229971568	0.0233200038	0.0232647390	0.0231579947
X2	0.1070001704	0.3612622973	0.0227056859	0.0235158475	0.0232766097	0.0233274618
X3	0.1032219846	0.3618552700	0.0229201594	0.0235890318	0.0230666639	0.0233065747
X4	0.1059539693	0.3615880887	0.0225601368	0.0234311554	0.0229748658	0.0232375793
X5	0.5979304844	0.6946950229	0.3856754640	0.3833271123	0.3844689125	0.3511017853
X6	0.5982328228	0.6937105033	0.3868473821	0.3836173684	0.3857926604	0.3507432956
X7	0.0406732130	0.2730541857	0.0019378279	0.0019670639	0.0019873608	0.0016830351
X8	0.0404563566	0.2756344962	0.0019657924	0.0019546176	0.0019802340	0.0017043113
X9	0.0408962472	0.2753223976	0.0019480704	0.0019910747	0.0019841344	0.0016844494
X10	0.0403495858	0.2727943254	0.0019379208	0.0019757359	0.0019831625	0.0016937909
X11	0.0580350271	0.3141957619	0.0086151465	0.0087337037	0.0086102031	0.0093857138
X12	0.1083706423	0.3611005912	0.0233107460	0.0237608555	0.0227686202	0.0868553395
X13	0.1143210918	0.3922759449	0.0389763493	0.0423825147	0.0411188367	0.0353521846
X14	0.4529637189	0.6310137963	0.3038923524	0.3056413039	0.3056193317	0.2742891764

Parameter Total Sensitivity - standard deviation

	NSS ROTS	NSS iPad	AFD PIC	AFD ARM	AFD Atom - 40%	AFD Atom - 20/40%
X1	0.0090166858	0.0137717366	0.0013194004	0.0011503399	0.0012181949	0.0005239576
X2	0.0093432309	0.0138070732	0.0010282867	0.0011864927	0.0012567852	0.0006190611
X3	0.0078089712	0.0143820025	0.0011027886	0.0012057127	0.0013695863	0.0005790707
X4	0.0096940880	0.0141723131	0.0012704467	0.0011692106	0.0014244404	0.0006808576
X5	0.0119917295	0.0165287690	0.0068461172	0.0087038778	0.0088254335	0.0058656017
X6	0.0116226563	0.0145313317	0.0073549411	0.0072046040	0.0069112823	0.0057588230
X7	0.0021676687	0.0099883701	0.0000942735	0.0000944034	0.0001105907	0.0000932050
X8	0.0021655430	0.0111791130	0.0001004629	0.0001090997	0.0001187969	0.0000934431
X9	0.0026909484	0.0104414352	0.0000917358	0.0001049021	0.0001146983	0.0000779227
X10	0.0024725834	0.0126337185	0.0001025755	0.0001126876	0.0001011528	0.0000937015
X11	0.0039432790	0.0165799434	0.0003350087	0.0003439391	0.0003521849	0.0003344971
X12	0.0136924405	0.0235386040	0.0029660457	0.0035024958	0.0036462905	0.0037182712
X13	0.0047283531	0.0174606470	0.0011781624	0.0010162627	0.0012034148	0.0014291850
X14	0.0161957040	0.0330392015	0.0060775149	0.0064307016	0.0062906707	0.0041118387

Normalized Mean for Total Sensitivity

	NSS ROTS	NSS iPad	AFD PIC	AFD ARM	AFD Atom - 40%	AFD Atom - 20/40%
X1	0.0436	0.0646	0.0185	0.0187	0.0186	0.0192
X2	0.0425	0.0641	0.0182	0.0188	0.0186	0.0193
X3	0.0410	0.0642	0.0184	0.0189	0.0185	0.0193
X4	0.0421	0.0642	0.0181	0.0188	0.0184	0.0192
X5	0.2375	0.1233	0.3095	0.3069	0.3078	0.2908
X6	0.2376	0.1232	0.3104	0.3071	0.3089	0.2905
X7	0.0162	0.0485	0.0016	0.0016	0.0016	0.0014
X8	0.0161	0.0489	0.0016	0.0016	0.0016	0.0014
X9	0.0162	0.0489	0.0016	0.0016	0.0016	0.0014
X10	0.0160	0.0484	0.0016	0.0016	0.0016	0.0014
X11	0.0230	0.0558	0.0069	0.0070	0.0069	0.0078
X12	0.0430	0.0641	0.0187	0.0190	0.0182	0.0719
X13	0.0454	0.0696	0.0313	0.0339	0.0329	0.0293
X14	0.1799	0.1120	0.2438	0.2447	0.2447	0.2272

Parameter Sensitivity - standard deviation

	AFD Atom - 10/40%	AFD Atom - 10%	Grain Bin	VOC controller	Space DAQ - 40%	Space DAQ - 10%
X1	0.0000673341	0.0020154507	0.0007989857	0.0011096910	0.0006202973	0.0020826545
X2	0.0000531484	0.0017737928	0.0008966197	0.0009323491	0.0006183694	0.0020484635
X3	0.0000588920	0.0023572392	0.0009599529	0.0010656789	0.0006050095	0.0019178680
X4	0.0000570277	0.0016568471	0.0008716398	0.0009199240	0.0006429342	0.0020614076
X5	0.0024011156	0.0013562373	0.0047780764	0.0412052478	0.0056176336	0.0014221055
X6	0.0022547290	0.0015103653	0.0053670572	0.0385995900	0.0067075252	0.0013768668
X7	0.0000000297	0.0000000224	0.0002828827	0.0043232326	0.0000000377	0.0000000366
X8	0.0000000331	0.0000000172	0.0002918638	0.0018122704	0.0000000415	0.0000000414
X9	0.0000000233	0.0000000277	0.0002716688	0.0002242074	0.0000000502	0.0000000459
X10	0.0000000305	0.0000000247	0.0002561057	0.0002171770	0.0000000489	0.0000000448
X11	0.0000226516	0.0007852964	0.0005861154	0.0014876430	0.0001942834	0.0007165546
X12	0.0002609281	0.0040737451	0.0010673367	0.0026524167	0.0013481527	0.0034267483
X13	0.0001022317	0.0021719483	0.0008689635	0.0024656734	0.0002496436	0.0019160514
X14	0.0014382969	0.0045771211	0.0037348745	0.0308867583	0.0048967322	0.0049018403

Parameter Total Sensitivity - mean

	AFD Atom - 10/40%	AFD Atom - 10%	Grain Bin	VOC controller	Space DAQ - 40%	Space DAQ - 10%
X1	0.0058145848	0.0719977111	0.5979792186	0.0592848017	0.0254043099	0.0715791417
X2	0.0058441247	0.0719210192	0.5986435537	0.0587035362	0.0253675675	0.0715045485
X3	0.0058576061	0.0711889640	0.5949060194	0.0591289819	0.0256200094	0.0718814484
X4	0.0058336587	0.0717957482	0.5948181850	0.0574679435	0.0252289416	0.0712667642
X5	0.3967436961	0.0875554451	0.7131393241	0.3340537991	0.3824154248	0.0849623326
X6	0.3971225946	0.0873061476	0.7168301831	0.3431483432	0.3829269868	0.0854223677
X7	0.0019023008	0.0083229915	0.5898337226	0.3657014082	0.0020422008	0.0082717602
X8	0.0018915161	0.0081608457	0.5921769558	0.2955089946	0.0020598653	0.0082651591
X9	0.0019024591	0.0081540307	0.5887806234	0.0493954097	0.0020316802	0.0083207738
X10	0.0018995325	0.0085442559	0.5896790402	0.0498189956	0.0020332451	0.0083339886
X11	0.0048552574	0.0552696163	0.5922691621	0.0533881279	0.0092256281	0.0558162873
X12	0.0317522229	0.4629979901	0.5975889008	0.0665375686	0.0260463743	0.4715009102
X13	0.0390698874	0.0751455185	0.6023811418	0.0832705997	0.0427783994	0.0723240963
X14	0.3083845288	0.2351844235	0.6881675892	0.2773833457	0.3052177916	0.2311588787

Parameter Total Sensitivity - mean

	AFD Atom - 10/40%	AFD Atom - 10%	Grain Bin	VOC controller	Space DAQ - 40%	Space DAQ - 10%
X1	0.0058145848	0.0719977111	0.5979792186	0.0592848017	0.0254043099	0.0715791417
X2	0.0058441247	0.0719210192	0.5986435537	0.0587035362	0.0253675675	0.0715045485
X3	0.0058576061	0.0711889640	0.5949060194	0.0591289819	0.0256200094	0.0718814484
X4	0.0058336587	0.0717957482	0.5948181850	0.0574679435	0.0252289416	0.0712667642
X5	0.3967436961	0.0875554451	0.7131393241	0.3340537991	0.3824154248	0.0849623326
X6	0.3971225946	0.0873061476	0.7168301831	0.3431483432	0.3829269868	0.0854223677
X7	0.0019023008	0.0083229915	0.5898337226	0.3657014082	0.0020422008	0.0082717602
X8	0.0018915161	0.0081608457	0.5921769558	0.2955089946	0.0020598653	0.0082651591
X9	0.0019024591	0.0081540307	0.5887806234	0.0493954097	0.0020316802	0.0083207738
X10	0.0018995325	0.0085442559	0.5896790402	0.0498189956	0.0020332451	0.0083339886
X11	0.0048552574	0.0552696163	0.5922691621	0.0533881279	0.0092256281	0.0558162873
X12	0.0317522229	0.4629979901	0.5975889008	0.0665375686	0.0260463743	0.4715009102
X13	0.0390698874	0.0751455185	0.6023811418	0.0832705997	0.0427783994	0.0723240963
X14	0.3083845288	0.2351844235	0.6881675892	0.2773833457	0.3052177916	0.2311588787

Parameter Total Sensitivity - standard deviation

	AFD Atom - 10/40%	AFD Atom - 10%	Grain Bin	VOC controller	Space DAQ - 40%	Space DAQ - 10%
X1	0.0001538910	0.0044382653	0.0077077510	0.0070485109	0.0012710224	0.0044879274
X2	0.0001442524	0.0040969755	0.0072528285	0.0057767037	0.0014579474	0.0043964958
X3	0.0001574311	0.0048073109	0.0062737216	0.0075941385	0.0013944946	0.0039218813
X4	0.0001562295	0.0036547002	0.0068304158	0.0058278583	0.0012885600	0.0043074216
X5	0.0030194118	0.0033208776	0.0095397121	0.0713304779	0.0076073351	0.0036197423
X6	0.0028369092	0.0037111937	0.0099390320	0.0663696336	0.0088575886	0.0033911375
X7	0.0000914754	0.0010135453	0.0078570242	0.0110308226	0.0001123552	0.0012190936
X8	0.0000888656	0.0011955444	0.0081582878	0.0083392756	0.0001031262	0.0012577594
X9	0.0001046105	0.0012221182	0.0080283535	0.0063241400	0.0001060350	0.0011520564
X10	0.0000922953	0.0012429628	0.0071222463	0.0069893122	0.0001105516	0.0013523476
X11	0.0001245846	0.0021671577	0.0075334310	0.0072390380	0.0004699794	0.0021219485
X12	0.0006178563	0.0077785265	0.0077155180	0.0092401667	0.0035448258	0.0067072253
X13	0.0007786836	0.0037845739	0.0094388451	0.0095638252	0.0012597553	0.0032937361
X14	0.0019165110	0.0070996086	0.0075251434	0.0594550582	0.0068524891	0.0076279115

Normalized Mean for Total Sensitivity

	AFD Atom - 10/40%	AFD Atom - 10%	Grain Bin	VOC controller	Space DAQ - 40%	Space DAQ - 10%
X1	0.0048	0.0544	0.0691	0.0275	0.0202	0.0542
X2	0.0048	0.0543	0.0691	0.0273	0.0202	0.0541
X3	0.0048	0.0538	0.0687	0.0275	0.0204	0.0544
X4	0.0048	0.0542	0.0687	0.0267	0.0200	0.0540
X5	0.3282	0.0662	0.0824	0.1552	0.3039	0.0643
X6	0.3285	0.0660	0.0828	0.1594	0.3043	0.0647
X7	0.0016	0.0063	0.0681	0.1699	0.0016	0.0063
X8	0.0016	0.0062	0.0684	0.1373	0.0016	0.0063
X9	0.0016	0.0062	0.0680	0.0229	0.0016	0.0063
X10	0.0016	0.0065	0.0681	0.0231	0.0016	0.0063
X11	0.0040	0.0418	0.0684	0.0248	0.0073	0.0423
X12	0.0263	0.3498	0.0690	0.0309	0.0207	0.3570
X13	0.0323	0.0568	0.0696	0.0387	0.0340	0.0548
X14	0.2551	0.1777	0.0795	0.1288	0.2425	0.1750

Matlab Source Code

Original eFAST

```
function [Sens TotSens X Y] = eFAST(P, funchd, N1, invcdfhd)
% P         :  Number of parameters in model
% funchd    :  Function handle to model
% N1        :  Desired number of sample runs of the model
% Example usage:
%              [S T] = eFAST(3, @ishigami, 20000);
% invcfdhd  :  Optional handle to the inverse Cum. Distr. Fn. (CDF)
%              Default: the parameters X are uniformly distributed in [0 1]
%              When invcdf is provided, invcdf(X) follows another pdf.
%              For example if P is in [0 1]
%                     X = betainv(P,A,B)
%              then X follows the beta distribution.
%              Here invcdfhd can be a handle to the function betainv
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  REFERENCE                           %
% Andrea Saltelli, Stefano Tarantola and Karen Chan,  %
% "A quantitative model-independent method for        %
%    global sensitivity analysis of model output"     %
% Technometrics Vol. 41, pp. 39-56, 1999.             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% STEP-0: INITIALIZE
M = 4;                              % Max no. of harmonics
wi = floor(N1/M/2);                 % High frequency (the Mth harmonic will
                                    % be sampled at Nyquist rate)
if mod(wi,2) ~= 0
    wi = wi + 1;
end
N = 2*M*wi + 1;                     % No. of samples

for pn = 1 : P                      % Loop though each parameter number

    % STEP-1: DETERMINE PARAMETER FREQUENCIES
    wc = setfreq(P-1, wi/(2*M)-1);  % Freqs of complementary variables
    pc = 1:P;                       % Complementary parameter set
    pc(pn) = [];
    w = zeros(1,P);                 % All parameter frequencies
    w(pc) = wc;
    w(pn) = wi;

    % STEP-2: DETERMINE MODEL PARAMETER VALUES FOR EACH SAMPLE RUN
    ph = rand(1,P)*2*pi;         % 1xP random phase shifts
    s = pi*(2*(1:N)-N-1)/N;      % 1XN vector, -pi*(1-1/N) < s < pi*(1-1/N)
    phmat = repmat(ph',[1 N]);   % PxN random phase matrix
    theta = w'*s + phmat;
    X = 0.5+asin(sin(theta'))/pi; % PxN matrix of parameters

    % STEP-3: Transform X to another pdf (if known) and scale it
    if nargin > 3
        X = invcdfhd(X);
    end

    % STEP-4: RUN MODEL WITH EACH SAMPLE PARAMETER VECTOR
    Y = zeros(N,1);
    for j = 1 : N
        Y(j) = funchd(X(j,:)');
    end

    % STEP-5: FOURIER COEFFICIENTS FOR COMPLEMENTARY PARAMETER
    Y = Y - mean(Y);                  % Remove DC value
    mp = (N+1)/2;                     % Mid point of vector Y
    Yp = Y(mp-1:-1:1) + Y(mp+1:1:N);
    Ym = -Y(mp-1:-1:1) + Y(mp+1:1:N);
    for j = 1 : wi/2
        angle = j*2*(1:mp-1)*pi/N;  % Angles from 0 thru pi
        CosV = cos(angle);
        SinV = sin(angle);
        Acoeff(j) = (Y(mp)+Yp'*CosV')/N;
        Bcoeff(j) = Ym'*SinV'/N;
    end
    % Complementary variance
    Vci(pn) = 2*sum(Acoeff(1:wi/2).^2 + Bcoeff(1:wi/2).^2);

    % STEP-6: FOURIER COEFFICIENTS FOR PARAMETER OF INTEREST (pn)
    for j = wi : wi : M*wi
        angle = j*2*(1:mp-1)*pi/N;  % Angles from 0 thru pi
        CosV = cos(angle);
        SinV = sin(angle);
        Acoeff(j/wi) = (Y(mp)+Yp'*CosV')/N;
        Bcoeff(j/wi) = Ym'*SinV'/N;
    end
    % Parameter pn variance
    Vi(pn) = 2*sum(Acoeff(1:M).^2 + Bcoeff(1:M).^2);
    VY(pn) = Y'*Y/N;

    % STEP-7 COMPUTING SENSITIVITIES
    Sens(pn) = Vi(pn)/VY(pn);
    TotSens(pn) = 1 - (Vci(pn)/VY(pn));

end

function wc = setfreq(P, wmax)
% P    : The number of complementary parameters
% wmax : Maximum allowed frequency of complementary set
% Implemented recursively as described in:
% Appendix of "Sensitivity Analysis"
% [Saltelli et al.]
if P == 1
    wc = 1;
elseif wmax == 1
    wc = ones(1,P);
else
    infd = min([P, wmax]);
    istep = round((wmax-1)/(infd-1));
    if(wmax == 1)
        istep = 0;
    end
    otmp = 1:istep:infd*istep;
    fl_infd = floor(infd);
    for i=1:P
        j = mod(i-1,fl_infd)+1;
        wc(i) = otmp(j);
    end
end
```

Printing routine for eFAST output

```
function [S_r S_ave S_std S_norm T_r T_ave T_std T_norm] =
an_sensitivity(Sens_mat, TotSens_mat)
% -------------------------------------------------------------------------
%   This is an_sensitivity. By Kim R. Fowler
%   Created on 15 July 2018.
%
%   It analyses the sensitivity of the parameters produced by the eFAST
%   function. The original data derives from initial parameters used
%   in the buy-versus-build decision. Input is from data matrices with
%   initialization values from the script file, "eFAST_bvb".
%
%   These simulations only target embedded systems and they only refer to
%   one type of module for a particular system. The module under
%   consideration may have multiple copies within a single system.
% -------------------------------------------------------------------------

Sindex = size(Sens_mat);        % matrix size determines loop index

idx = Sindex(1,1);              % shorter index and variable names
S_mat = Sens_mat;
T_mat = TotSens_mat;

S_r = zeros(idx, 1);            % initialize vectors of statistics
S_ave = zeros(idx, 1);
S_std = zeros(idx, 1);
T_r = zeros(idx, 1);
T_ave = zeros(idx, 1);
T_std = zeros(idx, 1);

for i = 1:idx
    S_r(i) = (max(S_mat(i,:)) - min(S_mat(i,:))) / max(S_mat(i,:));
    S_ave(i) = mean(S_mat(i,:));
    S_std(i) = std(S_mat(i,:));
    T_r(i) = (max(T_mat(i,:)) - min(T_mat(i,:))) / max(T_mat(i,:));
    T_ave(i) = mean(T_mat(i,:));
    T_std(i) = std(T_mat(i,:));
end
```

```
S_tot_sum = sum(S_ave);
T_tot_sum = sum(T_ave);
S_norm = S_ave / S_tot_sum;
T_norm = T_ave / T_tot_sum;

fprintf('\n')
fprintf('                Sensitivity\n')
fprintf('--------------------------------------------\n')
fprintf('       percent\n')
fprintf('      delta in                                    Normalized\n')
fprintf('        range       Mean          Std. Dev.        weights\n')
fprintf('--------------------------------------------       --------\n')
for i = 1:idx
    fprintf('X(%2.f',i);fprintf(') ')
    if S_r(i) < 0.1
        fprintf('   %2.1f',100*S_r(i))
    else
        fprintf('  %2.1f',100*S_r(i))
    end
    fprintf('    %1.10f',S_ave(i))
    fprintf('    %1.10f',S_std(i))
    fprintf('        %.4f\n',S_norm(i))
end

fprintf('\n');fprintf('\n')
fprintf('              Total Sensitivity\n')
fprintf('--------------------------------------------\n')
fprintf('       percent\n')
fprintf('      delta in                                    Normalized\n')
fprintf('        range       Mean          Std. Dev.        weights\n')
fprintf('--------------------------------------------       --------\n')
for i = 1:idx
    fprintf('X(%2.f',i);fprintf(') ')
    if T_r(i) < 0.1
        fprintf('   %2.1f',100*T_r(i))
    else
        fprintf('  %2.1f',100*T_r(i))
    end
    fprintf('    %1.10f',T_ave(i))
    fprintf('    %1.10f',T_std(i))
    fprintf('        %.4f\n',T_norm(i))
end
```

eFAST inside loop for multiple runs

```
function [Sens_mat TotSens_mat] = eFAST_loop(P, funchd, N1, N2)
% /////////////////////////////////////////////////////////////////////////
% -------------------------------------------------------------------------
% Created 15 July 2018 by Kim R. Fowler.
%
% P             :  Number of parameters in model
% funchd        :  Function handle to model
% N1            :  Desired number of sample runs of the model in eFAST
% N2            :  Number of overall runs of eFAST
% Sens_mat      :  Sensitivity data matrix
% Totsens_mat   :  Total sensitivity data matrix
% --------------------------------------------------------------------------
% //////////////////////////////////////////////////////////////////////////

Sens_mat = zeros(P,N2);
TotSens_mat = zeros(P,N2);

for i = 1:N2
% //////////////////////////////////////////////////////////////////////////
% --------------------------------------------------------------------------
%   This is eFAST from Sanjoy Das.
%
% P         :  Number of parameters in model
% funchd    :  Function handle to model
% N1        :  Desired number of sample runs of the model
% Sens      :  Sensitivity
% Totsens    :  Total sensitivity
% d, Q1-4   :  parameter matrices needed by bvb
% Example usage:
%              [S T] = eFAST(3, @ishigami, 20000);
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  REFERENCE                          %
% Andrea Saltelli, Stefano Tarantola and Karen Chan,  %
% "A quantitative model-independent method for        %
%    global sensitivity analysis of model output"     %
% Technometrics Vol. 41, pp. 39-56, 1999.             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% --------------------------------------------------------------------------
% //////////////////////////////////////////////////////////////////////////

% STEP-1: INITIALIZE
M = 4;                              % Max no. of harmonics
wi = floor(N1/M/2);                 % High frequency (the Mth harmonic will
                                    % be sampled at Nyquist rate)
if mod(wi,2) ~= 0
    wi = wi + 1;
end
N = 2*M*wi + 1;                     % No. of samples

for pn = 1 : P                      % Loop though each parameter number

    % STEP-2: DETERMINE PARAMETER FREQUENCIES
    wc = setfreq(P-1, wi/(2*M)-1);  % Freqs of complementary variables
    pc = 1:P;                       % Complementary parameter set
    pc(pn) = [];
    w = zeros(1,P);                 % All parameter frequencies
    w(pc) = wc;
    w(pn) = wi;

    % STEP-3: DETERMINE MODEL PARAMETER VALUES FOR EACH SAMPLE RUN
    ph = rand(1,P)*2*pi;         % 1xP random phase shifts
    s = pi*(2*(1:N)-N-1)/N;      % 1XN vector, -pi*(1-1/N) < s < pi*(1-1/N)
    phmat = repmat(ph',[1 N]);   % PxN random phase matrix
    theta = w'*s + phmat;
    X = 0.5+asin(sin(theta'))/pi; % PxN matrix of parameters

    % STEP-4: RUN MODEL WITH EACH SAMPLE PARAMETER VECTOR
    Y = zeros(N,1);
    for j = 1 : N
        Y(j) = funchd(X(j,:)');
    end

    % STEP-5: FOURIER COEFFICIENTS FOR COMPLEMENTARY PARAMETER
    Y = Y - mean(Y);                % Remove DC value
    mp = (N+1)/2;                   % Mid point of vector Y
    Yp = Y(mp-1:-1:1) + Y(mp+1:1:N);
    Ym = -Y(mp-1:-1:1) + Y(mp+1:1:N);
    for j = 1 : wi/2
        angle = j*2*(1:mp-1)*pi/N;  % Angles from 0 thru pi
        CosV = cos(angle);
        SinV = sin(angle);
        Acoeff(j) = (Y(mp)+Yp'*CosV')/N;
        Bcoeff(j) = Ym'*SinV'/N;
    end
    % Complementary variance
    Vci(pn) = 2*sum(Acoeff(1:wi/2).^2 + Bcoeff(1:wi/2).^2);

    % STEP-6: FOURIER COEFFICIENTS FOR PARAMETER OF INTEREST (pn)
    for j = wi : wi : M*wi
        angle = j*2*(1:mp-1)*pi/N;  % Angles from 0 thru pi
        CosV = cos(angle);
        SinV = sin(angle);
        Acoeff(j/wi) = (Y(mp)+Yp'*CosV')/N;
        Bcoeff(j/wi) = Ym'*SinV'/N;
    end
    % Parameter pn variance
    Vi(pn) = 2*sum(Acoeff(1:M).^2 + Bcoeff(1:M).^2);
    VY(pn) = Y'*Y/N;

    % STEP-7 COMPUTING SENSITIVITIES
    Sens(pn) = Vi(pn)/VY(pn);
    TotSens(pn) = 1 - (Vci(pn)/VY(pn));

end

% Save the sensitivity data to a data matrix and continue loop
    Sens_mat(:,i) = Sens(1,:)';
    TotSens_mat(:,i) = TotSens(1,:)';
    fprintf('Run = %.f\n',i)
end

function wc = setfreq(P, wmax)
% P    : The number of complementary parameters
% wmax : Maximum allowed frequency of complementary set
% Implemented recursively as described in:
% Appendix of "Sensitivity Analysis"
% [Saltelli et al.]
if P == 1
    wc = 1;
elseif wmax == 1
    wc = ones(1,P);
else
    infd = min([P, wmax]);
    istep = round((wmax-1)/(infd-1));
    if(wmax == 1)
        istep = 0;
    end
    otmp = 1:istep:infd*istep;
    fl_infd = floor(infd);
    for i=1:P
        j = mod(i-1,fl_infd)+1;
        wc(i) = otmp(j);
    end
end
```

One Example Model used by eFAST

```
function [Y] = bvb_grain_controller(X)
% -------------------------------------------------------------------------
%   This is bvb_grain_controller.
%   By Kim R. Fowler.
%   Created on 3 July 2018.
%
%   It calculates a single crossover point. It is used by either the
%   eFAST_bvb or eFAST_loop algorithm to determine parameter sensitivities.
%
%   These simulations only target embedded systems and they only refer to
%   one type of module for a particular system. The module under
%   consideration may have multiple copies within a single system.
%
%   This code includes insufficient COTS specifications, insufficient
%   vendor support, finite vendor consulting support, or lifecycle that
%   increase the COTS NRE and thus lower crossover point. It includes lower
%   expertise and resources that can increase the NRE of custom design.
% -------------------------------------------------------------------------

% /////////////////////////////////////////////////////////////////////////
% -------------------------------------------------------------------------
%   This section originated from init_param_VOC_control. Instead of
%   rewriting bvbt23 to shorten the code (and introduce errors), this
%   section initializes all the code.
%
%   It simulates buy versus build for a controller for a network
%   of grain bin sensors.
%
%   Rayleigh parameters: factor = 0.7, offset = 0.8
%
%   This script file prepares a data matrix with initialization values for
%   simulations of the buy versus build model. It sets up of all the
%   variables for either COTS or custom-build. It also echo prints the
%   variables to screen set.
% -------------------------------------------------------------------------
COTS_price = 175;       % Single-lot price for COTS module
Comp_price = 41;        % Single-lot price for custom-build components

% -------------------------------------------------------------------------
%   Discount schedules at the following quantity breaks, N
% -------------------------------------------------------------------------
N = [1 10 20 50 100 200 500 1000 2000 5000 10000 20000 50000];
COTS_discount = [1 1 1 1 .9 .9 .85 .8 .78 .78 .75 .7 .7];
Comp_discount = [1 1 .916 .916 .829 .829 .8 .76 .76 .73 .73 .7 .68];

% -------------------------------------------------------------------------
%   Percent of COTS features compared to the system specifications.
%   Make sure that the values stay between 0 and 1.
% -------------------------------------------------------------------------
COTS_spec_lo = 0.60;    % Lower limit on the Monte Carlo simulation
COTS_spec_hi = 1;       % Upper limit on the Monte Carlo simulation

% -------------------------------------------------------------------------
%   Make sure that the values of the market lifecycles (yrs) are > 0.
%   Also make sure that the HI > LO.
% -------------------------------------------------------------------------
% Market lifecycle (yrs) of COTS module, measured in years
COTS_life_lo = 2;     % Lower limit on the Monte Carlo simulation
COTS_life_hi = 4;    % Upper limit on the Monte Carlo simulation
% Market lifecycle (yrs) of system, measured in years
sys_life_lo = 10;      % Lower limit on the Monte Carlo simulation
sys_life_hi = 30;     % Upper limit on the Monte Carlo simulation

% -------------------------------------------------------------------------
%   Set premium for inventory for custom design or for COTS or to buy ROTS
%   if system life is greater than COTS module’s life. Premium can vary
%   between 1 and 5x.
% -------------------------------------------------------------------------
inven_prem1 = 1;        % Premium when system life < COTS life, usually = 1
inven_prem2 = 6;        % Premium when system life < COTS life, vary 1 to 5

% -------------------------------------------------------------------------
%   Some system numbers that are planned by development team.
% -------------------------------------------------------------------------
N_sys = 40000;          % Number of systems planned over lifetime
mod_persys = 1;         % Number of modules per system, default=1
Sys_cost = 3000;        % Planned unit cost of a system
Sys_price = 15000;      % Planned unit price of a system

% -------------------------------------------------------------------------
%   Labor costs in $/hr
% -------------------------------------------------------------------------
Prod_labor = 80;        % Average loaded salary of production staff
Tech_labor = 80;        % Average loaded salary of technicians
Eng_labor = 110;        % Average loaded salary of engineers & developers
Manag_labor = 130;      % Average loaded salary of management
Admin_labor = 60;       % Average loaded salary of admin & support staff
```

```
Consult_labor = 80;     % Average loaded salary of consulting & prod. staff

% --------------------------------------------------------------------------
%   Variable initialization for COGS calculations per unit COTS module.
% --------------------------------------------------------------------------
% Maxtrix FTE for COTS in production
COTS_COGS_FTE = [1 1 1 0; .05 .05 .05 .05; 0 0 0 1];

COTS_fab_t = 0.1;       % Time (hrs) to fabricate and assemble a module,
                        % this includes time to load software, default=2min
COTS_test_t = 0.1;      % Time (hrs) to test a module, default=1 min
COTS_insert_t = 0.1;   % Time (hrs) in production to insert module
COTS_inventory =0.2;    % Time to inventory a module, default=2 min
COTS_sales = 0;         % Cost ($) of sales and marketing per module
COTS_techsup = 0;       % Cost ($) of technical support per module
COTS_mat_en = 0.5;        % Cost ($) for assets, materials, and energy / unit

% --------------------------------------------------------------------------
%   Variable initialization for COGS calculations per unit custom module.
% --------------------------------------------------------------------------
% Maxtrix FTE for custom-build in production
Cust_COGS_FTE = [1.5 1 1 0; .05 .05 .05 .05; 0 0 0 1];
Cust_fab_t = 0.3;       % Time (hrs) to fabricate and assemble a module,
                        % this includes time to load software
Cust_test_t = 0.1;      % Time (hrs) to test a module
Cust_insert_t = 0.1;    % Time (hrs) in production to insert module
Cust_inventory =0.2;    % Time to inventory a module
Cust_sales = 0;         % Cost ($) of sales and marketing per module
Cust_techsup = 0;       % Cost ($) of technical support per module
Cust_mat_en = 2.5;        % Cost ($) for assets, materials, and energy / unit

% --------------------------------------------------------------------------
%   Variable initialization for COTS NRE calculations for development of
%   module incorporation in the system.
% --------------------------------------------------------------------------
COTS_tool_cost = 0;     % One time cost ($) to tool up for COTS production
                        % default = $0
COTS_tool_t = 0;        % Time (hrs) for tooling up for COTS production
                        % default = 0 days
COTS_deliv_t = 28;      % Delivery time (days) for first COTS modules
                        % default = 14 days
COTS_cert_cost = 20000; % Cost ($) for certification of COTS module

COTS_team_sup = 0;      % Yearly cost for vendor supporting dev. team
Final_customer_sup = 0; % Yearly cost for vendor supporting final customer

% --------------------------------------------------------------------------
%   Percent of COTS vendor support needed by development team.
%   Make sure that the values stay between 0 and 1.
% --------------------------------------------------------------------------
COTS_ve_su_lo = 0;      % Lower limit on the Monte Carlo simulation
COTS_ve_su_hi = 0.4;    % Upper limit on the Monte Carlo simulation

% --------------------------------------------------------------------------
%   Percent quality of COTS vendor support provided to development team.
%   Make sure that the values stay between 0 and 1. 1 = best, 0 = none.
% -------------------------------------------------------------------------
Qual_doc_lo = 0.6;      % Lower limit on the Monte Carlo simulation
Qual_doc_hi = 1;        % Upper limit on the Monte Carlo simulation

% -------------------------------------------------------------------------
%   Percent COTS vendor support provided to customers; make sure that the
%   values stay between 0 and 1. Also vendor charge for extra support.
% -------------------------------------------------------------------------
COTS_te_su_lo = 0;      % Lower limit on the Monte Carlo simulation
COTS_te_su_hi = 0.4;    % Upper limit on the Monte Carlo simulation

COTS_interact = 0;      % Does customer interact directly with module?
                        % no = 0, yes = 1

% -------------------------------------------------------------------------
%   COTS NRE calculations for effort and time. The full time equivalent for
%   staffing is a 8 x 4 matrix; the rows represent specific types of
%   staff members; the columns represent development phases.
%
%   Row 1 = hardware engineers
%   Row 2 = software engineers and developers
%   Row 3 = mechanical engineers
%   Row 4 = specialty engineers (optics, chemical, materials, OR)
%   Row 5 = technicians
%   Row 6 = management
%   Row 7 = admin support staff
%   Row 8 = production staff and other consultants
%
%   Column 1 = Concept & Preliminary phase
%   Column 2 = Critical Design phase
%   Column 3 = Test & Integration phase
%   Column 4 = Compliance & Production Preparation phase
% -------------------------------------------------------------------------
COTS_NRE_FTE = [1 .1 .1 .5; 1 .1 .2 0; .1 .5 .2 .1; .1 .5 .2 .1; .25 .025
.025 .5; .1 .1 .025 .1; .1 .1 .1 .1; .05 .05 .025 .025];
COTS_Calendar = [2; 2; 6; 6];       %NRE calendar time per phase (months)

% -------------------------------------------------------------------------
%   Variable initialization for NRE calculations for development of a
%   custom-build module in the system.
% -------------------------------------------------------------------------
Cust_tool_cost = 3000;     % One time cost ($) to tool up for production
                        % default = $0
Cust_tool_t = 42;        % Time (hrs) for tooling up for production
                        % default = 0 days
Cust_deliv_t = 14;      % Delivery time (days) for first component lots
                        % default = 14 days
Cust_cert_cost = 40000; % Cost ($) for certification of custom-built module

% -------------------------------------------------------------------------
%   Percent of the necessary expertise of the development team.
%   Make sure that the values stay between 0 and 1. 1 = best, 0 = none.
% -------------------------------------------------------------------------
Cust_ex_lo = 0.6;       % Lower limit on the Monte Carlo simulation
Cust_ex_hi = 1;         % Upper limit on the Monte Carlo simulation
```

```
% -------------------------------------------------------------------------
%   Percent of the necessary resources available to the development team.
%   Make sure that the values stay between 0 and 1. 1 = best, 0 = none.
% -------------------------------------------------------------------------
Cust_re_lo = 0.6;       % Lower limit on the Monte Carlo simulation
Cust_re_hi = 1;         % Upper limit on the Monte Carlo simulation

% -------------------------------------------------------------------------
%   NRE calculations for effort and time. The full time equivalent for
%   staffing is a 8 x 4 matrix; the rows represent specific types of
%   staff members; the columns represent development phases.
%
%   Row 1 = hardware engineers
%   Row 2 = software engineers and developers
%   Row 3 = mechanical engineers
%   Row 4 = specialty engineers (optics, chemical, materials, OR)
%   Row 5 = technicians
%   Row 6 = management
%   Row 7 = admin support staff
%   Row 8 = production staff and other consultants
%
%   Column 1 = Concept & Preliminary phase
%   Column 2 = Critical Design phase
%   Column 3 = Test & Integration phase
%   Column 4 = Compliance & Production Preparation phase
% -------------------------------------------------------------------------
Cust_NRE_FTE = [1 1 1 .5; 1 1 1 .5; .1 .5 .2 .1; .1 .5 .2 .1; 1 1 2 2; 1 1 1
1; .3 .3 .3 .3; .3 .3 .3 .3];
Cust_Calendar = [3; 6; 6; 6];     %NRE calendar time per phase (months)

% -------------------------------------------------------------------------
%   Store parameter and data initializations in matrices
% -------------------------------------------------------------------------

d = zeros(75,1);            % Initialize scalar data to store in a matrix

d(1,1) = COTS_price;
d(2,1) = Comp_price;

d(3,1) = COTS_spec_lo;
d(4,1) = COTS_spec_hi;
d(5,1) = COTS_life_lo;
d(6,1) = COTS_life_hi;
d(7,1) = sys_life_lo;
d(8,1) = sys_life_hi;

d(9,1) = inven_prem1;
d(10,1) = inven_prem2;
d(11,1) = N_sys;
d(12,1) = mod_persys;
d(13,1) = Sys_price;
d(14,1) = Sys_cost;

d(15,1) = Prod_labor;
```

```
d(16,1) = Tech_labor;
d(17,1) = Eng_labor;
d(18,1) = Manag_labor;
d(19,1) = Admin_labor;
d(20,1) = Consult_labor;

d(21,1) = COTS_fab_t;
d(22,1) = COTS_test_t;
d(23,1) = COTS_insert_t;
d(24,1) = COTS_inventory;
d(25,1) = COTS_sales;
d(26,1) = COTS_techsup;
d(27,1) = COTS_mat_en;

d(28,1) = Cust_fab_t;
d(29,1) = Cust_test_t;
d(30,1) = Cust_insert_t;
d(31,1) = Cust_inventory;
d(32,1) = Cust_sales;
d(33,1) = Cust_techsup;
d(34,1) = Cust_mat_en;

d(35,1) = COTS_tool_cost;
d(36,1) = COTS_tool_t;
d(37,1) = COTS_deliv_t;
d(38,1) = COTS_cert_cost;
%d(39,1) = COTS_expert;
%d(40,1) = COTS_resource;
d(41,1) = COTS_team_sup;
d(42,1) = Final_customer_sup;

d(43,1) = COTS_ve_su_lo;
d(44,1) = COTS_ve_su_hi;
d(45,1) = Qual_doc_lo;
d(46,1) = Qual_doc_hi;
d(47,1) = COTS_te_su_lo;
d(48,1) = COTS_te_su_hi;
d(49,1) = COTS_interact;

d(50,1) = Cust_tool_cost;
d(51,1) = Cust_tool_t;
d(52,1) = Cust_deliv_t;
d(53,1) = Cust_cert_cost;

d(54,1) = Cust_ex_lo;
d(55,1) = Cust_ex_hi;
d(56,1) = Cust_re_lo;
d(57,1) = Cust_re_hi;

index = size(N);
Q1 = zeros(3,index(1,2));    % Initialize the discount schedules
Q1(1,:) = N;
Q1(2,:) = COTS_discount;
Q1(3,:) = Comp_discount;
```

```
Q2 = zeros(3,4,2);          % Initialize the COGS FTE
Q2(:,:,1) = COTS_COGS_FTE;
Q2(:,:,2) = Cust_COGS_FTE;

Q3 = zeros(8,4,2);          % Initialize the development NRE
Q3(:,:,1) = COTS_NRE_FTE;
Q3(:,:,2) = Cust_NRE_FTE;

Q4 = zeros(4,1,2);          % Initialize the development calendar
Q4(:,:,1) = COTS_Calendar;
Q4(:,:,2) = Cust_Calendar;

% //////////////////////////////////////////////////////////////////////////
% --------------------------------------------------------------------------
%   This is the beginning of the calculation section. It setups all
%   variables for either COTS or custom-build.
% --------------------------------------------------------------------------
% //////////////////////////////////////////////////////////////////////////

COTS_price = d(1,1);        % Single-lot price for COTS module
Comp_price = d(2,1);        % Single-lot price for custom-build components

% --------------------------------------------------------------------------
%   Discount schedules at the following quantity breaks in N
% --------------------------------------------------------------------------
N = Q1(1,:);
N_indx = size(N);
COTS_discount = Q1(2,:);
Comp_discount = Q1(3,:);

% --------------------------------------------------------------------------
%   Percent of COTS features compared to the system specifications.
%   Make sure that the values stay between 0 and 1.
% --------------------------------------------------------------------------
COTS_spec_lo = d(3,1);      % Lower limit on the Monte Carlo simulation
COTS_spec_hi = d(4,1);      % Upper limit on the Monte Carlo simulation

% --------------------------------------------------------------------------
%   Make sure that the values of the market lifecycles (yrs) are > 0.
%   Also make sure that the HI > LO.
% --------------------------------------------------------------------------
% Market lifecycle (yrs) of COTS module, measured in years
COTS_life_lo = d(5,1);      % Lower limit on the Monte Carlo simulation
COTS_life_hi = d(6,1);      % Upper limit on the Monte Carlo simulation
% Market lifecycle (yrs) of system, measured in years
sys_life_lo = d(7,1);       % Lower limit on the Monte Carlo simulation
sys_life_hi = d(8,1);       % Upper limit on the Monte Carlo simulation

% --------------------------------------------------------------------------
%   Set premium for inventory for custom design or for COTS or to buy ROTS
%   if system life is greater than COTS module’s life. Premium can vary
%   between 1 and 5x.
% --------------------------------------------------------------------------
inven_prem1 = d(9,1);   % Premium when system life < COTS life, usually = 1
inven_prem2 = d(10,1);  % Premium when system life < COTS life, vary 1 to 5
```

```
% -------------------------------------------------------------------------
%   Some system numbers planned
% -------------------------------------------------------------------------
N_sys = d(11,1);               % Number of systems planned over lifetime
mod_sys = d(12,1);             % Number of modules per system, default=1
mod_total = N_sys * mod_sys;   % Total modules planned for product lifetime

Sys_price = d(13,1);                    % Planned unit price of a system
Sys_cost = d(14,1);                     % Planned unit cost of a system
Sys_revenue = Sys_price - Sys_cost;     % Planned revenue per system

Early_sales_rate = 2;           % Average early sales per month of systems

% -------------------------------------------------------------------------
%   Labor costs in $/hr
% -------------------------------------------------------------------------
Prod_labor = d(15,1);   % Average loaded salary of production staff
Tech_labor = d(16,1);   % Average loaded salary of technicians
Eng_labor = d(17,1);    % Average loaded salary of engineers & developers
Manag_labor = d(18,1);  % Average loaded salary of management
Admin_labor = d(19,1);  % Average loaded salary of admin & support staff
Consult_labor = d(20,1);% Average loaded salary of consulting & prod. staff

% COGS - vector of staff costs for production
COGS_labor(1,1) = Prod_labor;       % col.1 = production staff
COGS_labor(1,2) = Manag_labor;      % col.2 = management
COGS_labor(1,3) = Admin_labor;      % col.3 = admin/support staff

% NRE - vector of staff costs for development
Dev_labor(1,1) = Eng_labor;         % col.1 = H/W engineer
Dev_labor(1,2) = Eng_labor;         % col.2 = S/W engineer
Dev_labor(1,3) = Eng_labor;         % col.3 = Mechanical engineer
Dev_labor(1,4) = Eng_labor;         % col.4 = specialty engineer
Dev_labor(1,5) = Tech_labor;        % col.5 = technician
Dev_labor(1,6) = Manag_labor;       % col.6 = management
Dev_labor(1,7) = Admin_labor;       % col.7 = admin/support staff
Dev_labor(1,8) = Consult_labor;     % col.8 = production & misc consulting

% -------------------------------------------------------------------------
%   Variable initialization for COGS calculations per unit COTS module.
% -------------------------------------------------------------------------
% Maxtrix FTE for COTS in production
COTS_COGS_FTE = Q2(:,:,1);

COTS_fab_t = d(21,1);       % Time (hrs) to fabricate and assemble a module
COTS_test_t = d(22,1);      % Time (hrs) to test a module
COTS_insert_t = d(23,1);    % Time (hrs) in production to insert module
COTS_inventory = d(24,1);   % Time to inventory a module
COTS_sales = d(25,1);       % Cost ($) of sales and marketing per module
COTS_techsup = d(26,1);     % Cost ($) of technical support per module
COTS_mat_en = d(27,1);      % Cost ($) for assets, materials, energy/unit

% -------------------------------------------------------------------------
%   Variable initialization for COGS calculations per unit custom module.
% -------------------------------------------------------------------------
% Maxtrix FTE for custom-build in production
Cust_COGS_FTE = Q2(:,:,2);

Cust_fab_t = d(28,1);       % Time (hrs) to fabricate and assemble a module
Cust_test_t = d(29,1);      % Time (hrs) to test a module
Cust_insert_t = d(30,1);    % Time (hrs) in production to insert module
Cust_inventory = d(31,1);   % Time to inventory a module
Cust_sales = d(32,1);       % Cost ($) of sales and marketing per module
Cust_techsup = d(33,1);     % Cost ($) of technical support per module
Cust_mat_en = d(34,1);      % Cost ($) for assets, materials, energy/unit

% -------------------------------------------------------------------------
%   Variable initialization for COTS NRE calculations for development of
%   module incorporation in the system.
% -------------------------------------------------------------------------
COTS_tool_cost = d(35,1);   % One time cost ($) to tool up for COTS
production
COTS_tool_t = d(36,1);      % Time (hrs) for tooling up for COTS production
COTS_deliv_t = d(37,1);     % Delivery time (days) for first COTS modules
COTS_cert_cost = d(38,1);   % Cost ($) for certification of COTS module

COTS_team_sup = d(41,1);        % Yearly cost ($) for vendor supporting
                                % development team
Final_customer_sup = d(42,1);   % Yearly cost ($) for vendor supporting
                                % final customer
% -------------------------------------------------------------------------
%   Percent of COTS vendor support needed by development team.
%   Make sure that the values stay between 0 and 1.
% -------------------------------------------------------------------------
COTS_ve_su_lo = d(43,1);    % Lower limit on the Monte Carlo simulation
COTS_ve_su_hi = d(44,1);    % Upper limit on the Monte Carlo simulation

% -------------------------------------------------------------------------
%   Percent quality of COTS vendor support provided to development team.
%   Make sure that the values stay between 0 and 1. 1 = best, 0 = none.
% -------------------------------------------------------------------------
Qual_doc_lo = d(45,1);      % Lower limit on the Monte Carlo simulation
Qual_doc_hi = d(46,1);      % Upper limit on the Monte Carlo simulation

% -------------------------------------------------------------------------
%   Percent COTS vendor support provided to customers; make sure that the
%   values stay between 0 and 1. Also vendor charge for extra support.
% -------------------------------------------------------------------------
COTS_te_su_lo = d(47,1);    % Lower limit on the Monte Carlo simulation
COTS_te_su_hi = d(48,1);    % Upper limit on the Monte Carlo simulation

COTS_interact = d(49,1);    % Does customer interact directly with module?
                            % no = 0, yes = 1
Final_customer_sup_cost = Final_customer_sup * COTS_interact;

% -------------------------------------------------------------------------
%   COTS NRE calculations for effort and time. The full time equivalent for
%   staffing is a 8 x 4 matrix; the rows represent specific types of
%   staff members; the columns represent development phases.
```

```
%
%   Row 1 = hardware engineers
%   Row 2 = software engineers and developers
%   Row 3 = mechanical engineers
%   Row 4 = specialty engineers (optics, chemical, materials, OR)
%   Row 5 = technicians
%   Row 6 = management
%   Row 7 = admin support staff
%   Row 8 = production staff and other consultants
%
%   Column 1 = Concept & Preliminary phase
%   Column 2 = Critical Design phase
%   Column 3 = Test & Integration phase
%   Column 4 = Compliance & Production Preparation phase
% --------------------------------------------------------------------------
COTS_NRE_FTE = Q3(:,:,1);
COTS_Calendar = Q4(:,:,1);  %NRE calendar time per phase (months)

% --------------------------------------------------------------------------
%   Variable initialization for NRE calculations for development of a
%   custom-build module in the system.
% --------------------------------------------------------------------------
Cust_tool_cost = d(50,1);   % One time cost ($) to tool up for production
Cust_tool_t = d(51,1);      % Time (hrs) for tooling up for production
Cust_deliv_t = d(52,1);     % Delivery time (days) for first component lots
Cust_cert_cost = d(53,1);   % Cost($) to certify custom-built module

% --------------------------------------------------------------------------
%   Percent of the necessary expertise of the development team.
%   Make sure that the values stay between 0 and 1. 1 = best, 0 = none.
% --------------------------------------------------------------------------
Cust_ex_lo = d(54,1);       % Lower limit on the Monte Carlo simulation
Cust_ex_hi = d(55,1);       % Upper limit on the Monte Carlo simulation

% --------------------------------------------------------------------------
%   Percent of the necessary resources available to the development team.
%   Make sure that the values stay between 0 and 1. 1 = best, 0 = none.
% --------------------------------------------------------------------------
Cust_re_lo = d(56,1);       % Lower limit on the Monte Carlo simulation
Cust_re_hi = d(57,1);       % Upper limit on the Monte Carlo simulation

% --------------------------------------------------------------------------
%   NRE calculations for effort and time. The full time equivalent for
%   staffing is a 8 x 4 matrix; the rows represent specific types of
%   staff members; the columns represent development phases.
%
%   Row 1 = hardware engineers
%   Row 2 = software engineers and developers
%   Row 3 = mechanical engineers
%   Row 4 = specialty engineers (optics, chemical, materials, OR)
%   Row 5 = technicians
%   Row 6 = management
%   Row 7 = admin support staff
%   Row 8 = production staff and other consultants
%
%   Column 1 = Concept & Preliminary phase
%   Column 2 = Critical Design phase
%   Column 3 = Test & Integration phase
%   Column 4 = Compliance & Production Preparation phase
% -------------------------------------------------------------------------
Cust_NRE_FTE = Q3(:,:,2);
Cust_Calendar = Q4(:,:,2);     %NRE calendar time per phase (months)

% /////////////////////////////////////////////////////////////////////////
% -------------------------------------------------------------------------
%   Perform deterministic calculations, assuming exact estimation of
%   effort, FTE, and calendar time.
% -------------------------------------------------------------------------
% /////////////////////////////////////////////////////////////////////////

% -------------------------------------------------------------------------
%   COGS calculations per unit COTS module. This simulation assumes
%   no learning curve to shorten time in production.
% -------------------------------------------------------------------------
% Vector time and total time for COTS in production
COTS_COGS_t = [COTS_fab_t; COTS_test_t; COTS_insert_t; COTS_inventory];
COTS_COGS_tot_t = sum(COTS_COGS_t);     % Total time (hrs) per module
% Total cost ($) of labor, material, and energy per module
COTS_COGS_tot_cost = (COGS_labor*(COTS_COGS_FTE*COTS_COGS_t)) + COTS_sales +
COTS_techsup + COTS_mat_en;

% -------------------------------------------------------------------------
%   COGS calculations per unit custom-built module. This simulation assumes
%   no learning curve to shorten time in production.
% -------------------------------------------------------------------------
% Total time and total time for custom-build in production
Cust_COGS_t = [Cust_fab_t; Cust_test_t; Cust_insert_t; Cust_inventory];
Cust_COGS_tot_t = sum(Cust_COGS_t);     % Total time (hrs) per module
% Total cost ($) of labor, material, and energy per module
Cust_COGS_tot_cost =(COGS_labor*(Cust_COGS_FTE* Cust_COGS_t)) + Cust_sales +
Cust_techsup + Cust_mat_en;

% -------------------------------------------------------------------------
%   COTS NRE calculations for development of module incorporation in the
%   system. The NRE calculations then results in cost per unit COTS module.
% -------------------------------------------------------------------------
COTS_person_month = COTS_NRE_FTE * COTS_Calendar;
COTS_Tot_effort = sum(COTS_person_month);

% -------------------------------------------------------------------------
%   NRE calculations for development of a custom-build module in the
%   system. The NRE calculations then results in cost per unit module.
% -------------------------------------------------------------------------
Cust_person_month = Cust_NRE_FTE * Cust_Calendar;
Cust_Tot_effort = sum(Cust_person_month);

% /////////////////////////////////////////////////////////////////////////
% ************************************************************************
%   This begins the actual calculation section
% ************************************************************************
% /////////////////////////////////////////////////////////////////////////
```

```
sig_fac = 0.7;                  % Sigma factor in Rayleigh distributions
off_set = 0.8;                  % Offset for Rayleigh distributions

% -------------------------------------------------------------------------
%   Add random dither to set up values within vectors
% -------------------------------------------------------------------------

COTS_spec_diff = COTS_spec_hi - COTS_spec_lo; % Vary specification
iCOTS_spec = COTS_spec_lo + (COTS_spec_diff * X(1));

Qual_doc_diff = Qual_doc_hi - Qual_doc_lo;    % Vary quality vendor support
iCOTS_vend_doc = Qual_doc_lo + (Qual_doc_diff * X(2));

COTS_ve_su_diff = COTS_ve_su_hi - COTS_ve_su_lo;    % Vary vendor support
iCOTS_vend_sup = COTS_ve_su_lo + (COTS_ve_su_diff * X(3));

COTS_te_su_diff = COTS_te_su_hi - COTS_te_su_lo;    % Vary customer support
iCOTS_tech_sup = COTS_te_su_lo + (COTS_te_su_diff * X(4));

Cust_ex_diff = Cust_ex_hi - Cust_ex_lo;             % Vary expertise
iCust_expert = Cust_ex_lo + (Cust_ex_diff * X(5));

Cust_re_diff = Cust_re_hi - Cust_re_lo;             % Vary resources
iCust_resource = Cust_re_lo + (Cust_re_diff * X(6));

sys_life_diff = sys_life_hi - sys_life_lo;          % Vary system life
iSystem_life = sys_life_lo + (sys_life_diff * X(7));

COTS_life_diff = COTS_life_hi - COTS_life_lo;       % Vary COTS life
iCOTS_life = COTS_life_lo + (COTS_life_diff * X(8));

% vary vendor charges from 0 to max. life of system for both development
% team support and for support to final customer
cap = ceil(iSystem_life);   % Discretize life into years for max. support
iCOTS_team_cost = (round(X(9)*cap)) * COTS_team_sup;
iCOTS_fi_cu_cost = (round(X(10)*cap)) * Final_customer_sup_cost;

% -------------------------------------------------------------------------
%   Multiply, element by element, the vector of insufficient or incorrect
%   vendor support or documentation with the vector for insufficient COTS
%   specifications. This operation calculates the factor of increased
%   COTS cost when inserted into the sigmoid Logistic Function.
% -------------------------------------------------------------------------
filter_spec1 = iCOTS_spec * iCOTS_vend_doc;
filter_spec2 = (1 - iCOTS_vend_sup) * (1 - iCOTS_tech_sup);
filter_spec = 1 - (filter_spec1 * filter_spec2);

% -------------------------------------------------------------------------
%   Use the sigmoid curve of the Logistic Function to calculate when
%   both insufficient COTS specifications and COTS support will increase
%   COTS cost and when both support to the team and the customer are
%   required in some measure.
% -------------------------------------------------------------------------
% L factor for sigmoid, it scales the final output to not exceed the
% custom effort.
L_factor = Cust_Tot_effort/COTS_Tot_effort;

% Sets steepness of the sigmoid
k_stp_lo = 7;                   % Sets lower limit on steepness
k_stp_hi = 25;                  % Sets upper limit on steepness
k_diff = k_stp_hi - k_stp_lo;   % Sets the difference between limits
k_steep = k_stp_lo + (k_diff * X(11));

% Sets midpoint of the sigmoid
x0sig_lo = 0.2;                 % Sets lower limit on midpoint
x0sig_hi = 0.5;                 % Sets upper limit on midpoint
x0_diff = x0sig_hi - x0sig_lo;  % Sets the difference between limits
x0sigmoid = x0sig_lo + (x0_diff * X(12));

% Calculate the offset vector when x = 0
filter_zero = L_factor/(1 + exp(k_steep * x0sigmoid));

% Calculate vector of corrections to the COTS NRE effort to address
% insufficient specifications, quality of support, or required NRE
COTS_cor = (L_factor/(1 + exp(k_steep * (x0sigmoid-filter_spec)))) -
filter_zero + 1;
% Calculate vector of time corrections; assume that the corrected effort is
% split evenly between the staff FTEs and the calendar time
COTS_time_cor = sqrt(COTS_cor);

% -------------------------------------------------------------------------
%   Multiply, element by element, the vector of team expertise and
%   available company resources. This operation calculates the factor of
%   increased custom design cost when inserted into the Logistic Function.
% -------------------------------------------------------------------------
filter_spec = 1 - (iCust_expert * iCust_resource);

% -------------------------------------------------------------------------
%   Use the sigmoid curve of the Logistic Function to calculate when
%   both the expertise of the development team and available resources
%   will increase the NRE design cost and time.
% -------------------------------------------------------------------------

% L factor scales the final output to not exceed a 3 times (see +1 below).
L_factor = 2;

% Sets steepness of the sigmoid between 7 and 25
k_steep = k_stp_lo + (k_diff * X(13));

% Sets midpoint of the sigmoid between 0.2 and 0.5
x0sigmoid = x0sig_lo + (x0_diff * X(14));

% Calculate the offset vector when x = 0
filter_zero = L_factor/(1 + exp(k_steep * x0sigmoid));

% Calculate vector of corrections to the custom NRE effort
```

```
Cust_cor = (L_factor/(1 + exp(k_steep * (x0sigmoid-filter_spec)))) -
filter_zero + 1;
% Calculate vector of time corrections; assume that the corrected effort is
% split evenly between the staff FTEs and the calendar time
Cust_time_cor = sqrt(Cust_cor);

% --------------------------------------------------------------------------
%   Add random dither to FTE and calendar for both COTS and custom
% --------------------------------------------------------------------------
iCOTS_NRE_FTE = COTS_NRE_FTE * (raylrnd(sig_fac,1,1)+off_set);
iCOTS_Calendar = COTS_Calendar * (raylrnd(sig_fac,1,1)+off_set);
iCust_NRE_FTE = Cust_NRE_FTE * (raylrnd(sig_fac,1,1)+off_set);
iCust_Calendar = Cust_Calendar * (raylrnd(sig_fac,1,1)+off_set);

% --------------------------------------------------------------------------
%   Prepare filter mask for different inventory premiums: none, premium,
%   interest, or ROTS.
% --------------------------------------------------------------------------
inven_adj = inven_prem1;
ilife_ratio = iCOTS_life / iSystem_life;
if ilife_ratio < 1
    inven_adj = inven_prem2;
end

% --------------------------------------------------------------------------
%   Prepare an adjustment vector for additional costs charged by COTS
%   vendor to support the development team or the final customer or both.
% --------------------------------------------------------------------------
COTS_adjust = iCOTS_team_cost + iCOTS_fi_cu_cost;

% --------------------------------------------------------------------------
%   First, calculate the COTS time, staff effort, and unit costs
% --------------------------------------------------------------------------
COTS_person_month = (iCOTS_NRE_FTE * iCOTS_Calendar) * COTS_cor;
COTS_Tot_effort = sum(COTS_person_month);
% Total NRE labor cost for project ($), assume 168 hrs/month
COTS_Tot_labor = Dev_labor * COTS_person_month * 168;
% Total calendar time for NRE (months)
COTS_Tot_time = (sum(iCOTS_Calendar)) * COTS_time_cor;

% Calculate total COTS costs from NRE, tooling, compliance/certification
% and vendor suppor for the team and final customer
COTS_NRE_cost = COTS_Tot_labor + COTS_tool_cost + COTS_cert_cost +
COTS_adjust;

% Calculate prices for COTS modules with discounts and inventory premium
COTS_mod_p = COTS_discount * COTS_price * inven_adj;

% Set up COTS cost vector from total COGS cost
COTS_COGS_cost = ones(1,N_indx(1,2)) * COTS_COGS_tot_cost;

% Calculate vector of COTS unit prices at the cost breakpoints
COTS_unit_cost = COTS_COGS_cost + COTS_mod_p + (COTS_NRE_cost ./N);

% --------------------------------------------------------------------------
%   Second, calculate the custom-build time, staff effort, and unit costs
% -------------------------------------------------------------------------
Cust_person_month = (iCust_NRE_FTE * iCust_Calendar) * Cust_cor;
Cust_Tot_effort = sum(Cust_person_month);
% Total NRE labor cost for project ($), assume 168 hrs/month
Cust_Tot_labor = Dev_labor * Cust_person_month * 168;
% Total calendar time for NRE (months)
Cust_Tot_time = (sum(iCust_Calendar)) * Cust_time_cor;

% Calculate build costs from NRE, tooling, and compliance/certification
Cust_NRE_cost = Cust_Tot_labor + Cust_tool_cost + Cust_cert_cost;

% Calculate prices for custom-built modules with discounts and inventory
Cust_mod_p = Comp_discount * Comp_price * inven_prem1;

% Set up custom-build cost vector from total COGS cost
Cust_COGS_cost = ones(1,N_indx(1,2)) * Cust_COGS_tot_cost;

% Calculate the lost opportunity costs
lost_opp_cost = (Cust_Tot_time - COTS_Tot_time) * Early_sales_rate *
Sys_revenue;

% Calculate vector of custom unit prices at the cost breakpoints
Cust_unit_cost = Cust_COGS_cost + Cust_mod_p + (Cust_NRE_cost ./N);

% -------------------------------------------------------------------------
%   Find the crossover point in unit costs between COTS and custom-build
% -------------------------------------------------------------------------

find_crossover = Cust_unit_cost - COTS_unit_cost;    % First order crossover
find_cross1 = zeros(1,N_indx(1,2));
find_cross2 = zeros(1,N_indx(1,2));
cross_point = diff(sign(find_crossover));
find_cross1(1,1:(N_indx(1,2)-1)) = cross_point;
find_cross2(1,2:N_indx(1,2)) = cross_point;     % Interval end if decreasing
check_cross = sum(find_cross1);         % Does crossover exist?

if check_cross == 0                     % =0, then no crossover
    break_point = 0;                    % set output variable
    if find_crossover(1,1) <= 0
        COTS_cross_mod_cost = COTS_unit_cost(1,1);  % COTS cost to first unit
        Cust_cross_mod_cost = Cust_unit_cost(1,1);  % Custom cost to first
unit
    else
        COTS_cross_mod_cost = COTS_unit_cost(1,N_indx(1,2));    % COTS cost
to last unit
        Cust_cross_mod_cost = Cust_unit_cost(1,N_indx(1,2));    % Custom cost
to last unit
    end
else                                    % Find crossover indices
    crossover_i = [0 0];
    select_i1 = N.*(find_cross1/check_cross);
    select_i2 = N.*(find_cross2/check_cross);
    crossover_i(1,1) = max(select_i1);
    crossover_i(1,2) = max(select_i2);
```

```
% Calculate the module and COGS costs for this decreasing interval
    iCOTSmodp = max(COTS_mod_p.*(select_i1/crossover_i(1,1)));
    iCOTSCcost = max(COTS_COGS_cost.*(select_i1/crossover_i(1,1)));
    iCustmodp = max(Cust_mod_p.*(select_i1/crossover_i(1,1)));
    iCustCcost = max(Cust_COGS_cost.*(select_i1/crossover_i(1,1)));

    iendCOTSmodp = max(COTS_mod_p.*(select_i2/crossover_i(1,2)));
    iendCOTSCcost = max(COTS_COGS_cost.*(select_i2/crossover_i(1,2)));
    iendCustmodp = max(Cust_mod_p.*(select_i2/crossover_i(1,2)));
    iendCustCcost = max(Cust_COGS_cost.*(select_i2/crossover_i(1,2)));

% --------------------------------------------------------------------------
%   Calcualte the crossover point in the selected interval
% --------------------------------------------------------------------------
    M = crossover_i(1,1):1:crossover_i(1,2);    % Interval with the crossover
% Form vectors of the module and COGS costs for both COTS and custom-build.
% Must add discount break at last point in each vector
    iCOTS_mod_p = ones(size(M))*iCOTSmodp;
    iCOTS_mod_p(1,length(M)) = iendCOTSmodp;    % COTS end with discount
break
    iCOTS_COGS_cost = ones(size(M))*iCOTSCcost;
    iCOTS_COGS_cost(1,length(M)) = iendCOTSCcost;
    iCust_mod_p = ones(size(M))*iCustmodp;
    iCust_mod_p(1,length(M)) = iendCustmodp;    % Custom end with discount
break
    iCust_COGS_cost = ones(size(M))*iCustCcost;
    iCust_COGS_cost(1,length(M)) = iendCustCcost;
% Calculate the unit costs and then find the difference between
% COTS and custom-build
    iCOTS_unit_cost = iCOTS_COGS_cost + iCOTS_mod_p + (COTS_NRE_cost./M);
    iCust_unit_cost = iCust_COGS_cost + iCust_mod_p + (Cust_NRE_cost./M);

    idiff_unit_cost = iCust_unit_cost - iCOTS_unit_cost;
    icross_point = zeros(size(M));
    icross_point(1:length(M)-1) = diff(sign(idiff_unit_cost));
    icross_val = sum(icross_point);
    unity_cross = icross_point/icross_val;
    break_point = sum(M.*(unity_cross));
    COTS_cross_mod_cost = sum(iCOTS_unit_cost.*(unity_cross));
    Cust_cross_mod_cost = sum(iCust_unit_cost.*(unity_cross));
end

Y = break_point;

return;
```

Appendix H - Model Results for Recommending Build-versus-Buy

Overview of Model Operations

I programmed two primary routines. The first routine was a Matlab m-file script and it provided the basic input values for all 57 parameters; it had the general format for its function call of `init_param_(case study name)`. These initial parameter routines are described in Appendix F.

The second routine is a Matlab m-file script and it used the 57 parameters in a nearly identical model which only passed data through in one pass; it did not run a Monte Carlo simulation. It uses the sensitivity results from Appendix G to calculate crossover values for four different boundary cases. I ran the script on 15 different case studies that were basically identical to those in Appendix G. Case Study 3 split into two studies; one where the software costs $200 per unit, the other where the software costs $500 per unit. Case Studies 9 through 12 combined into one since I did not have to account for different ranges of random variables; otherwise, they setup identically. The same is true for Case Studies 17 and 18; I combined them into one since I did not have to account for different ranges of random variables; otherwise, they setup identically.

The second routine then prints out values, generate variables, and placed them in the workspace for other routines to analyze or plot specific values. The outputs for the 15 case studies follow this section.

Finally, I have printed the different routines used in the sensitivity analyses.

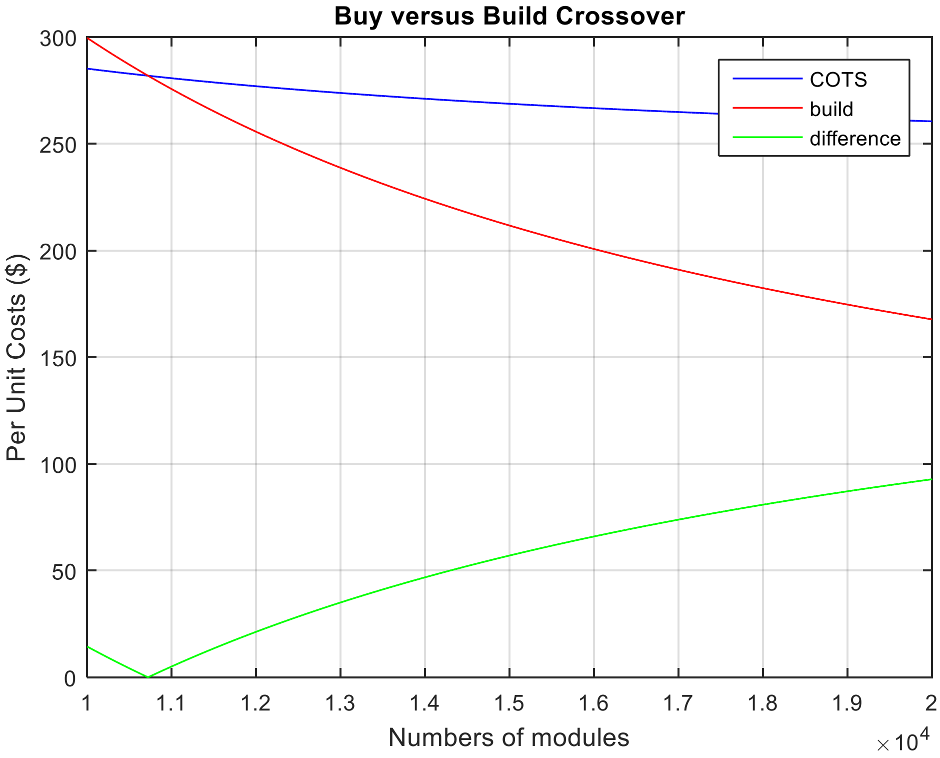
Recommender Model Raw Results

Table 5.3 already summarizes the recommendations for the 15 case studies. The remaining results are on following pages.

Model Results: $200 RTOS, Medical Device

August 27, 2018

```
>> [d, Q1, Q2, Q3, Q4] = init_param_Med_RTOS();

Single-lot price for COTS module = $ 200
Single-lot price for build components = $ 0
COTS features range in Monte Carlo from 0.60 to 1.00
COTS market life ranges in Monte Carlo from 5.0 to 10.0
System market life ranges in Monte Carlo from 5.0 to 20.0
Premium to inventory COTS = 1.0

Number of systems planned = 5000
Number of modules per system = 1

Average loaded hourly rates for staff:
Production staff = $ 80
Technicians = $ 80
Engineers & developers = $ 110
Management = $ 130
Administrative support = $ 60
Consulting & production staff = $ 80

COTS support and documentation for particular vendor:
Licensing support cost = $ 10000
Customer support cost = $ 0
Team NRE for vendor support ranges from 0.00 to 0.40
Team NRE for customer support ranges from 0.00 to 0.40
Quality of doc & support ranges in Monte Carlo from 0.60 to 1.00
Customer interaction directly with module = 0.00

>> [St1, St2, St3, St4] = bvb_model(d, Q1, Q2, Q3, Q4);

     STAGE 1
-----------------
Time of NRE for COTS (months) = 15.00
Time of NRE for custom (months) = 48.00
Cost of NRE for COTS = $ 493680
Cost of NRE for custom = $ 2637696

Lost opportunity cost = $ 1782000
Total system revenues over lifecycle = $ 135000000
Lost opportunity cost to total revenues = 0.0132

Crossover point (number of modules) = 10720
Crossover COTS module cost = $ 282
Crossover custom-build module cost = $ 282
----------------------------------------------

Press spacebar
```



```
##############################################
##############################################

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 16.20
Time of NRE for custom (months) = 51.84
Cost of NRE for COTS = $ 530774
Cost of NRE for custom = $ 2846312

Crossover point (number of modules) = 11577
Crossover COTS module cost = $ 282
Crossover custom-build module cost = $ 282

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 21.00
Time of NRE for custom (months) = 67.20
Cost of NRE for COTS = $ 679152
Cost of NRE for custom = $ 3680774

Crossover point (number of modules) = 15008
Crossover COTS module cost = $ 281
Crossover custom-build module cost = $ 281

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 30.00
Time of NRE for custom (months) = 96.00
Cost of NRE for COTS = $ 957360
Cost of NRE for custom = $ 5245392

Crossover point (number of modules) = 21440
Crossover COTS module cost = $ 280
Crossover custom-build module cost = $ 280
```

```
----------------------------------------------
     STAGE 2: Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 45.00
Time of NRE for custom (months) = 144.00
Cost of NRE for COTS = $ 1421040
Cost of NRE for custom = $ 7853088

Crossover point (number of modules) = 32160
Crossover COTS module cost = $ 280
Crossover custom-build module cost = $ 280

##############################################
##############################################

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 27.68
Time of NRE for custom (months) = 77.76
Cost of NRE for COTS = $ 885735
Cost of NRE for custom = $ 4254468
COTS sigmoid adjustment = 1.71
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 16843
Crossover COTS module cost = $ 288
Crossover custom-build module cost = $ 288

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 35.89
Time of NRE for custom (months) = 100.80
Cost of NRE for COTS = $ 1139286
Cost of NRE for custom = $ 5506162
COTS sigmoid adjustment = 1.71
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 21834
Crossover COTS module cost = $ 288
Crossover custom-build module cost = $ 288

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 51.26
Time of NRE for custom (months) = 144.00
Cost of NRE for COTS = $ 1614695
Cost of NRE for custom = $ 7853088
COTS sigmoid adjustment = 1.71
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 31191
Crossover COTS module cost = $ 288
Crossover custom-build module cost = $ 288

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 76.90
Time of NRE for custom (months) = 216.00
Cost of NRE for COTS = $ 2407042
Cost of NRE for custom = $ 11764632
COTS sigmoid adjustment = 1.71
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 46787
Crossover COTS module cost = $ 287
Crossover custom-build module cost = $ 287

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 83.05
Time of NRE for custom (months) = 129.60
Cost of NRE for COTS = $ 2597205
Cost of NRE for custom = $ 7070779
COTS sigmoid adjustment = 5.13
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 22367
Crossover COTS module cost = $ 352
Crossover custom-build module cost = $ 352

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 107.66
Time of NRE for custom (months) = 168.00
Cost of NRE for COTS = $ 3357859
Cost of NRE for custom = $ 9156936
COTS sigmoid adjustment = 5.13
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 28995
Crossover COTS module cost = $ 352
Crossover custom-build module cost = $ 352

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 153.79
Time of NRE for custom (months) = 240.00
Cost of NRE for COTS = $ 4784084
Cost of NRE for custom = $ 13068480
COTS sigmoid adjustment = 5.13
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 41421
Crossover COTS module cost = $ 351
```

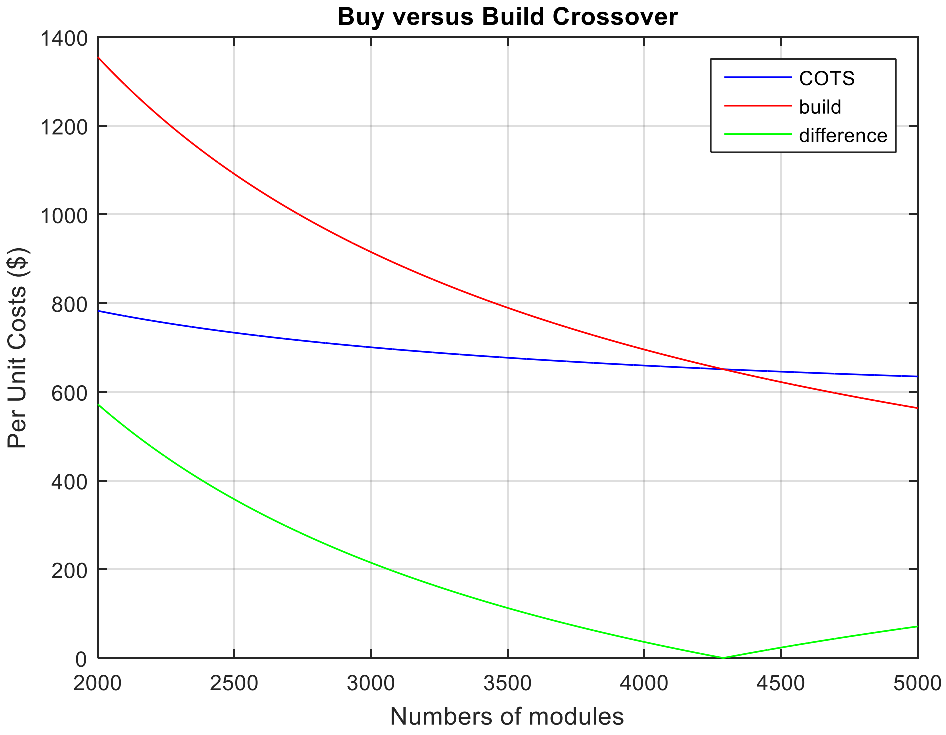
```
Crossover custom-build module cost = $ 351

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 230.69
Time of NRE for custom (months) = 360.00
Cost of NRE for COTS = $ 7161126
Cost of NRE for custom = $ 19587720
COTS sigmoid adjustment = 5.13
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 62132
Crossover COTS module cost = $ 351
Crossover custom-build module cost = $ 351

##############################################
##############################################

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 55.37
Time of NRE for custom (months) = 77.76
Cost of NRE for COTS = $ 1741470
Cost of NRE for custom = $ 4254468
COTS sigmoid adjustment (add 0.25 factor) = 3.42
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 12564
Crossover COTS module cost = $ 374
Crossover custom-build module cost = $ 374

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 71.77
Time of NRE for custom (months) = 100.80
Cost of NRE for COTS = $ 2248572
Cost of NRE for custom = $ 5506162
COTS sigmoid adjustment (add 0.25 factor) = 3.42
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 16287
Crossover COTS module cost = $ 374
Crossover custom-build module cost = $ 374

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 102.53
Time of NRE for custom (months) = 144.00
Cost of NRE for COTS = $ 3199389
Cost of NRE for custom = $ 7853088
COTS sigmoid adjustment (add 0.25 factor) = 3.42
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 23268
Crossover COTS module cost = $ 373
Crossover custom-build module cost = $ 373

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 153.79
Time of NRE for custom (months) = 216.00
Cost of NRE for COTS = $ 4784084
Cost of NRE for custom = $ 11764632
COTS sigmoid adjustment (add 0.25 factor) = 3.42
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 34902
Crossover COTS module cost = $ 373
Crossover custom-build module cost = $ 373

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 110.73
Time of NRE for custom (months) = 129.60
Cost of NRE for COTS = $ 3452940
Cost of NRE for custom = $ 7070779
COTS sigmoid adjustment (add 0.25 factor) = 6.84
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 18089
Crossover COTS module cost = $ 427
Crossover custom-build module cost = $ 427

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 143.54
Time of NRE for custom (months) = 168.00
Cost of NRE for COTS = $ 4467145
Cost of NRE for custom = $ 9156936
COTS sigmoid adjustment (add 0.25 factor) = 6.84
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 23448
Crossover COTS module cost = $ 426
Crossover custom-build module cost = $ 426

----------------------------------------------
```

```
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 205.06
Time of NRE for custom (months) = 240.00
Cost of NRE for COTS = $ 6368778
Cost of NRE for custom = $ 13068480
COTS sigmoid adjustment (add 0.25 factor) = 6.84
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 33498
Crossover COTS module cost = $ 426
Crossover custom-build module cost = $ 426

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 307.59
Time of NRE for custom (months) = 360.00
Cost of NRE for COTS = $ 9538168
Cost of NRE for custom = $ 19587720
COTS sigmoid adjustment (add 0.25 factor) = 6.84
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 50247
Crossover COTS module cost = $ 426
Crossover custom-build module cost = $ 426

##############################################
##############################################
```

Model Results: $500 RTOS, Medical Device

August 27, 2018

```
>> [d, Q1, Q2, Q3, Q4] = init_param_Med_RTOS();

Single-lot price for COTS module = $ 500
Single-lot price for build components = $ 0
COTS features range in Monte Carlo from 0.60 to 1.00
COTS market life ranges in Monte Carlo from 5.0 to 10.0
System market life ranges in Monte Carlo from 5.0 to 20.0
Premium to inventory COTS = 1.0

Number of systems planned = 5000
Number of modules per system = 1

Average loaded hourly rates for staff:
Production staff = $ 80
Technicians = $ 80
Engineers & developers = $ 110
Management = $ 130
Administrative support = $ 60
Consulting & production staff = $ 80

COTS support and documentation for particular vendor:
Licensing support cost = $ 10000
Customer support cost = $ 0
Team NRE for vendor support ranges from 0.00 to 0.40
Team NRE for customer support ranges from 0.00 to 0.40
Quality of doc & support ranges in Monte Carlo from 0.60 to 1.00
Customer interaction directly with module = 0.00


>> [St1, St2, St3, St4] = bvb_model(d, Q1, Q2, Q3, Q4);

     STAGE 1
-----------------
Time of NRE for COTS (months) = 15.00
Time of NRE for custom (months) = 48.00
Cost of NRE for COTS = $ 493680
Cost of NRE for custom = $ 2637696

Lost opportunity cost = $ 1782000
Total system revenues over lifecycle = $ 135000000
Lost opportunity cost to total revenues = 0.0132

Crossover point (number of modules) = 4288
Crossover COTS module cost = $ 651
Crossover custom-build module cost = $ 651
----------------------------------------------

Press spacebar
```



```
##############################################
##############################################

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 16.20
Time of NRE for custom (months) = 51.84
Cost of NRE for COTS = $ 530774
Cost of NRE for custom = $ 2846312

Crossover point (number of modules) = 4631
Crossover COTS module cost = $ 650
Crossover custom-build module cost = $ 650

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 21.00
Time of NRE for custom (months) = 67.20
Cost of NRE for COTS = $ 679152
Cost of NRE for custom = $ 3680774

Crossover point (number of modules) = 6003
Crossover COTS module cost = $ 649
Crossover custom-build module cost = $ 649

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 30.00
Time of NRE for custom (months) = 96.00
Cost of NRE for COTS = $ 957360
Cost of NRE for custom = $ 5245392

Crossover point (number of modules) = 8576
Crossover COTS module cost = $ 647
Crossover custom-build module cost = $ 647

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 3.00
----------------------------------------------
```

```
Time of NRE for COTS (months) = 45.00
Time of NRE for custom (months) = 144.00
Cost of NRE for COTS = $ 1421040
Cost of NRE for custom = $ 7853088

Crossover point (number of modules) = 12864
Crossover COTS module cost = $ 646
Crossover custom-build module cost = $ 646

##############################################
##############################################

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 27.68
Time of NRE for custom (months) = 77.76
Cost of NRE for COTS = $ 885735
Cost of NRE for custom = $ 4254468
COTS sigmoid adjustment = 1.71
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 6737
Crossover COTS module cost = $ 667
Crossover custom-build module cost = $ 667

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 35.89
Time of NRE for custom (months) = 100.80
Cost of NRE for COTS = $ 1139286
Cost of NRE for custom = $ 5506162
COTS sigmoid adjustment = 1.71
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 8733
Crossover COTS module cost = $ 666
Crossover custom-build module cost = $ 666
```

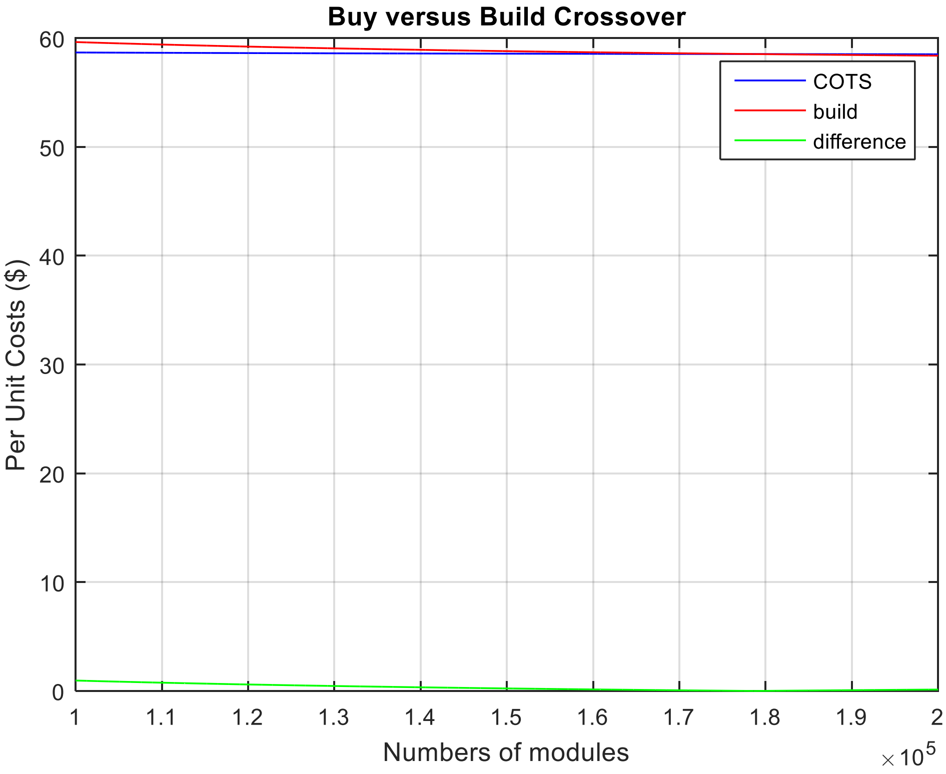
```
----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 51.26
Time of NRE for custom (months) = 144.00
Cost of NRE for COTS = $ 1614695
Cost of NRE for custom = $ 7853088
COTS sigmoid adjustment = 1.71
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 12476
Crossover COTS module cost = $ 665
Crossover custom-build module cost = $ 665

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 76.90
Time of NRE for custom (months) = 216.00
Cost of NRE for COTS = $ 2407042
Cost of NRE for custom = $ 11764632
COTS sigmoid adjustment = 1.71
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 18715
Crossover COTS module cost = $ 664
Crossover custom-build module cost = $ 664

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 83.05
Time of NRE for custom (months) = 129.60
Cost of NRE for COTS = $ 2597205
Cost of NRE for custom = $ 7070779
COTS sigmoid adjustment = 5.13
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 8947
Crossover COTS module cost = $ 826
Crossover custom-build module cost = $ 826

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 107.66
Time of NRE for custom (months) = 168.00
Cost of NRE for COTS = $ 3357859
Cost of NRE for custom = $ 9156936
COTS sigmoid adjustment = 5.13
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 11598
Crossover COTS module cost = $ 825
Crossover custom-build module cost = $ 825

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 153.79
Time of NRE for custom (months) = 240.00
Cost of NRE for COTS = $ 4784084
Cost of NRE for custom = $ 13068480
COTS sigmoid adjustment = 5.13
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 16568
Crossover COTS module cost = $ 825
Crossover custom-build module cost = $ 825

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 230.69
Time of NRE for custom (months) = 360.00
Cost of NRE for COTS = $ 7161126
Cost of NRE for custom = $ 19587720
COTS sigmoid adjustment = 5.13
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 24853
Crossover COTS module cost = $ 824
Crossover custom-build module cost = $ 824

##############################################
##############################################

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 55.37
Time of NRE for custom (months) = 77.76
Cost of NRE for COTS = $ 1741470
Cost of NRE for custom = $ 4254468
COTS sigmoid adjustment (add 0.25 factor) = 3.42
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 5025
Crossover COTS module cost = $ 882
Crossover custom-build module cost = $ 882

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 71.77
Time of NRE for custom (months) = 100.80
Cost of NRE for COTS = $ 2248572
Cost of NRE for custom = $ 5506162
COTS sigmoid adjustment (add 0.25 factor) = 3.42
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 6515
Crossover COTS module cost = $ 881
Crossover custom-build module cost = $ 881

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 102.53
Time of NRE for custom (months) = 144.00
Cost of NRE for COTS = $ 3199389
Cost of NRE for custom = $ 7853088
COTS sigmoid adjustment (add 0.25 factor) = 3.42
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 9307
Crossover COTS module cost = $ 880
Crossover custom-build module cost = $ 880

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 153.79
Time of NRE for custom (months) = 216.00
Cost of NRE for COTS = $ 4784084
Cost of NRE for custom = $ 11764632
COTS sigmoid adjustment (add 0.25 factor) = 3.42
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 13961
Crossover COTS module cost = $ 878
Crossover custom-build module cost = $ 878

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.08
----------------------------------------------
```

```
Time of NRE for COTS (months) = 110.73
Time of NRE for custom (months) = 129.60
Cost of NRE for COTS = $ 3452940
Cost of NRE for custom = $ 7070779
COTS sigmoid adjustment (add 0.25 factor) = 6.84
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 7235
Crossover COTS module cost = $ 1013
Crossover custom-build module cost = $ 1013

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 143.54
Time of NRE for custom (months) = 168.00
Cost of NRE for COTS = $ 4467145
Cost of NRE for custom = $ 9156936
COTS sigmoid adjustment (add 0.25 factor) = 6.84
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 9379
Crossover COTS module cost = $ 1012
Crossover custom-build module cost = $ 1012

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 205.06
Time of NRE for custom (months) = 240.00
Cost of NRE for COTS = $ 6368778
Cost of NRE for custom = $ 13068480
COTS sigmoid adjustment (add 0.25 factor) = 6.84
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 13399
Crossover COTS module cost = $ 1011
Crossover custom-build module cost = $ 1011

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 307.59
Time of NRE for custom (months) = 360.00
Cost of NRE for COTS = $ 9538168
Cost of NRE for custom = $ 19587720
COTS sigmoid adjustment (add 0.25 factor) = 6.84
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 20099
Crossover COTS module cost = $ 1010
Crossover custom-build module cost = $ 1010

##############################################
##############################################
```

Model Results: Point of Load Converter

August 27, 2018

```
>> [d, Q1, Q2, Q3, Q4] = init_param_POL();

Single-lot price for COTS module = $ 11
Single-lot price for build components = $ 5
COTS features range in Monte Carlo from 0.60 to 1.00
COTS market life ranges in Monte Carlo from 2.0 to 4.0
System market life ranges in Monte Carlo from 10.0 to 20.0
Premium to inventory COTS = 3.5

Number of systems planned = 1000000
Number of modules per system = 4

Average loaded hourly rates for staff:
Production staff = $ 80
Technicians = $ 80
Engineers & developers = $ 110
Management = $ 130
Administrative support = $ 60
Consulting & production staff = $ 80

COTS support and documentation for particular vendor:
Licensing support cost = $ 0
Customer support cost = $ 0
Team NRE for vendor support ranges from 0.00 to 0.40
Team NRE for customer support ranges from 0.00 to 0.40
Quality of doc & support ranges in Monte Carlo from 0.60 to 1.00
Customer interaction directly with module = 0.00

>> [St1, St2, St3, St4] = bvb_model(d, Q1, Q2, Q3, Q4);

     STAGE 1
-----------------
Time of NRE for COTS (months) = 5.00
Time of NRE for custom (months) = 12.00
Cost of NRE for COTS = $ 32760
Cost of NRE for custom = $ 250488

Lost opportunity cost = $ 196000
Total system revenues over lifecycle = $ 14000000000
Lost opportunity cost to total revenues = 0.0000

Crossover point (number of modules) = 178725
Crossover COTS module cost = $ 59
Crossover custom-build module cost = $ 59
----------------------------------------------

Press spacebar
```



```
##############################################
##############################################

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 5.40
Time of NRE for custom (months) = 12.96
Cost of NRE for COTS = $ 35381
Cost of NRE for custom = $ 270527

Crossover point (number of modules) = 193023
Crossover COTS module cost = $ 59
Crossover custom-build module cost = $ 59

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 7.00
Time of NRE for custom (months) = 16.80
Cost of NRE for COTS = $ 45864
Cost of NRE for custom = $ 350683

Crossover point (number of modules) = 250215
Crossover COTS module cost = $ 59
Crossover custom-build module cost = $ 59

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 10.00
Time of NRE for custom (months) = 24.00
Cost of NRE for COTS = $ 65520
Cost of NRE for custom = $ 500976

Crossover point (number of modules) = 357451
Crossover COTS module cost = $ 59
Crossover custom-build module cost = $ 59

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 3.00
```

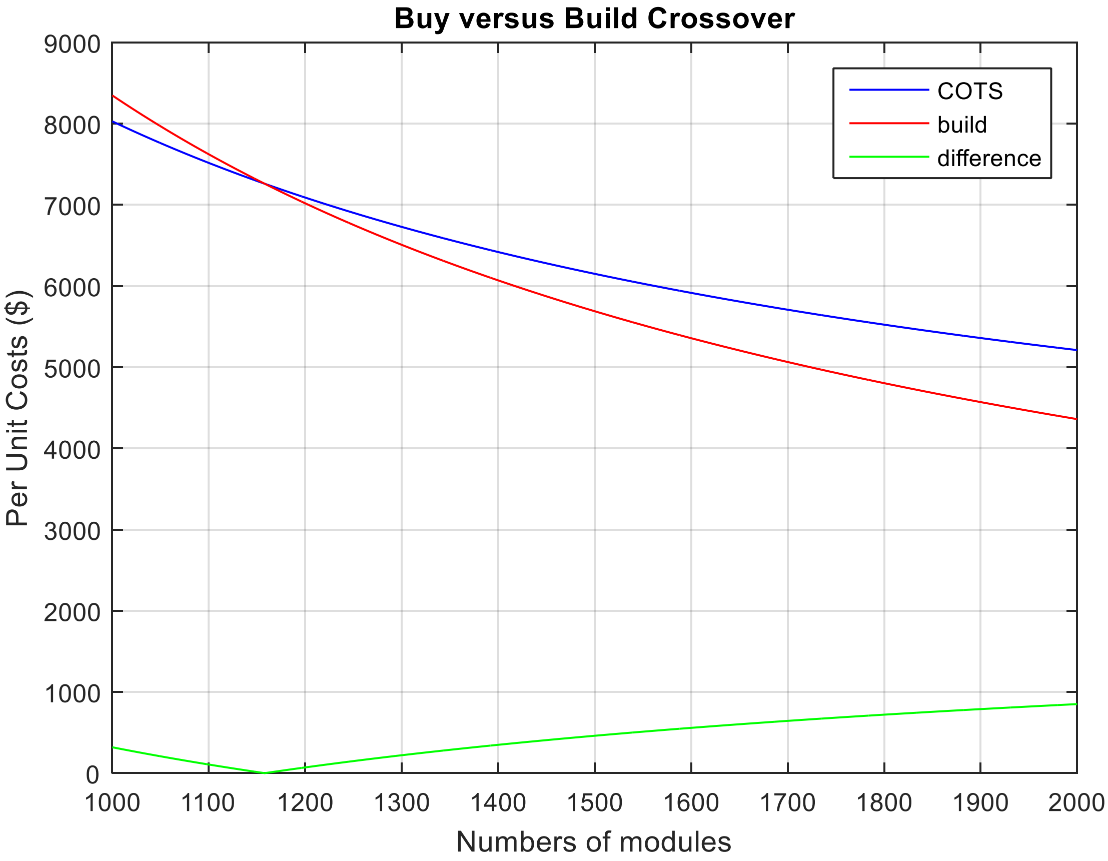
```
----------------------------------------------
Time of NRE for COTS (months) = 15.00
Time of NRE for custom (months) = 36.00
Cost of NRE for COTS = $ 98280
Cost of NRE for custom = $ 751464

Crossover point (number of modules) = 536176
Crossover COTS module cost = $ 59
Crossover custom-build module cost = $ 59

##############################################
##############################################

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 11.93
Time of NRE for custom (months) = 19.44
Cost of NRE for COTS = $ 78192
Cost of NRE for custom = $ 405791
COTS sigmoid adjustment = 2.21
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 268915
Crossover COTS module cost = $ 59
Crossover custom-build module cost = $ 59

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 15.47
Time of NRE for custom (months) = 25.20
Cost of NRE for COTS = $ 101359
Cost of NRE for custom = $ 526025
COTS sigmoid adjustment = 2.21
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 348593
Crossover COTS module cost = $ 59
Crossover custom-build module cost = $ 59

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 22.10
Time of NRE for custom (months) = 36.00
Cost of NRE for COTS = $ 144799
Cost of NRE for custom = $ 751464
COTS sigmoid adjustment = 2.21
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 497990
Crossover COTS module cost = $ 59
Crossover custom-build module cost = $ 59

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 33.15
Time of NRE for custom (months) = 54.00
Cost of NRE for COTS = $ 217199
Cost of NRE for custom = $ 1127196
COTS sigmoid adjustment = 2.21
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 746986
Crossover COTS module cost = $ 59
Crossover custom-build module cost = $ 59

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 35.80
Time of NRE for custom (months) = 32.40
Cost of NRE for COTS = $ 234575
Cost of NRE for custom = $ 676318
COTS sigmoid adjustment = 6.63
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 362611
Crossover COTS module cost = $ 59
Crossover custom-build module cost = $ 59

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 46.41
Time of NRE for custom (months) = 42.00
Cost of NRE for COTS = $ 304078
Cost of NRE for custom = $ 876708
COTS sigmoid adjustment = 6.63
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 470052
Crossover COTS module cost = $ 59
Crossover custom-build module cost = $ 59

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 66.30
Time of NRE for custom (months) = 60.00
Cost of NRE for COTS = $ 434398
Cost of NRE for custom = $ 1252440
COTS sigmoid adjustment = 6.63
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 671503
Crossover COTS module cost = $ 59
Crossover custom-build module cost = $ 59

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 99.45
Time of NRE for custom (months) = 90.00
Cost of NRE for COTS = $ 651596
Cost of NRE for custom = $ 1878660
COTS sigmoid adjustment = 6.63
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 1007255
Crossover COTS module cost = $ 59
Crossover custom-build module cost = $ 59

##############################################
##############################################

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 23.87
Time of NRE for custom (months) = 19.44
Cost of NRE for COTS = $ 156383
Cost of NRE for custom = $ 405791
COTS sigmoid adjustment (add 0.25 factor) = 4.42
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 204730
Crossover COTS module cost = $ 59
Crossover custom-build module cost = $ 59

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 30.94
Time of NRE for custom (months) = 25.20
Cost of NRE for COTS = $ 202719
Cost of NRE for custom = $ 526025
COTS sigmoid adjustment (add 0.25 factor) = 4.42
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 265390
Crossover COTS module cost = $ 59
Crossover custom-build module cost = $ 59

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 44.20
Time of NRE for custom (months) = 36.00
Cost of NRE for COTS = $ 289598
Cost of NRE for custom = $ 751464
COTS sigmoid adjustment (add 0.25 factor) = 4.42
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 379129
Crossover COTS module cost = $ 59
Crossover custom-build module cost = $ 59

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 66.30
Time of NRE for custom (months) = 54.00
Cost of NRE for COTS = $ 434398
Cost of NRE for custom = $ 1127196
COTS sigmoid adjustment (add 0.25 factor) = 4.42
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 568694
Crossover COTS module cost = $ 59
Crossover custom-build module cost = $ 59

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 47.74
Time of NRE for custom (months) = 32.40
Cost of NRE for COTS = $ 312766
Cost of NRE for custom = $ 676318
COTS sigmoid adjustment (add 0.25 factor) = 8.84
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 298427
Crossover COTS module cost = $ 59
Crossover custom-build module cost = $ 59

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 61.88
Time of NRE for custom (months) = 42.00
Cost of NRE for COTS = $ 405438
Cost of NRE for custom = $ 876708
COTS sigmoid adjustment (add 0.25 factor) = 8.84
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 386849
Crossover COTS module cost = $ 59
Crossover custom-build module cost = $ 59

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
```

```
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 88.40
Time of NRE for custom (months) = 60.00
Cost of NRE for COTS = $ 579197
Cost of NRE for custom = $ 1252440
COTS sigmoid adjustment (add 0.25 factor) = 8.84
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 552642
Crossover COTS module cost = $ 59
Crossover custom-build module cost = $ 59

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 132.60
Time of NRE for custom (months) = 90.00
Cost of NRE for COTS = $ 868795
Cost of NRE for custom = $ 1878660
COTS sigmoid adjustment (add 0.25 factor) = 8.84
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 828964
Crossover COTS module cost = $ 59
Crossover custom-build module cost = $ 59

##############################################
##############################################
```

Model Results: ROTS Touchscreen, Med Dev

August 27, 2018

```
>> [d, Q1, Q2, Q3, Q4] = init_param_NSS_ROTS();

Single-lot price for COTS module = $ 2530
Single-lot price for build components = $ 230
COTS features range in Monte Carlo from 0.80 to 1.00
COTS market life ranges in Monte Carlo from 5.0 to 10.0
System market life ranges in Monte Carlo from 5.0 to 10.0
Premium to inventory COTS = 1.0

Number of systems planned = 10000
Number of modules per system = 1

Average loaded hourly rates for staff:
Production staff = $ 80
Technicians = $ 80
Engineers & developers = $ 110
Management = $ 130
Administrative support = $ 60
Consulting & production staff = $ 80

COTS support and documentation for particular vendor:
Licensing support cost = $ 0
Customer support cost = $ 0
Team NRE for vendor support ranges from 0.00 to 0.20
Team NRE for customer support ranges from 0.00 to 0.20
Quality of doc & support ranges in Monte Carlo from 0.80 to 1.00
Customer interaction directly with module = 0.00

>> [St1, St2, St3, St4] = bvb_model(d, Q1, Q2, Q3, Q4);

     STAGE 1
-----------------
Time of NRE for COTS (months) = 42.00
Time of NRE for custom (months) = 48.00
Cost of NRE for COTS = $ 5634480
Cost of NRE for custom = $ 7973040

Lost opportunity cost = $ 32400
Total system revenues over lifecycle = $ 27000000
Lost opportunity cost to total revenues = 0.0012

Crossover point (number of modules) = 1157
Crossover COTS module cost = $ 7263
Crossover custom-build module cost = $ 7265
----------------------------------------------

Press spacebar
```



```
##############################################
##############################################

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 45.36
Time of NRE for custom (months) = 51.84
Cost of NRE for COTS = $ 6082838
Cost of NRE for custom = $ 8608483

Crossover point (number of modules) = 1250
Crossover COTS module cost = $ 7259
Crossover custom-build module cost = $ 7260

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 58.80
Time of NRE for custom (months) = 67.20
Cost of NRE for COTS = $ 7876272
Cost of NRE for custom = $ 11150256

Crossover point (number of modules) = 1621
Crossover COTS module cost = $ 7252
Crossover custom-build module cost = $ 7252

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 84.00
Time of NRE for custom (months) = 96.00
Cost of NRE for COTS = $ 11238960
Cost of NRE for custom = $ 15916080

Crossover point (number of modules) = 2347
Crossover COTS module cost = $ 7131
Crossover custom-build module cost = $ 7132

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 126.00
Time of NRE for custom (months) = 144.00
Cost of NRE for COTS = $ 16843440
Cost of NRE for custom = $ 23859120

Crossover point (number of modules) = 3521
Crossover COTS module cost = $ 7126
Crossover custom-build module cost = $ 7127

##############################################
##############################################

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 26.98
Time of NRE for custom (months) = 77.76
Cost of NRE for COTS = $ 3630395
Cost of NRE for custom = $ 12897725
COTS sigmoid adjustment = 0.59
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 4652
Crossover COTS module cost = $ 3123
Crossover custom-build module cost = $ 3123

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 34.98
Time of NRE for custom (months) = 100.80
Cost of NRE for COTS = $ 4697179
Cost of NRE for custom = $ 16710384
COTS sigmoid adjustment = 0.59
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 6195
Crossover COTS module cost = $ 3025
Crossover custom-build module cost = $ 3025

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 49.97
Time of NRE for custom (months) = 144.00
Cost of NRE for COTS = $ 6697399
Cost of NRE for custom = $ 23859120
COTS sigmoid adjustment = 0.59
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 8850
Crossover COTS module cost = $ 3023
Crossover custom-build module cost = $ 3023

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 74.95
Time of NRE for custom (months) = 216.00
Cost of NRE for COTS = $ 10031098
Cost of NRE for custom = $ 35773680
COTS sigmoid adjustment = 0.59
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 13631
Crossover COTS module cost = $ 2952
Crossover custom-build module cost = $ 2952

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 80.94
Time of NRE for custom (months) = 129.60
Cost of NRE for COTS = $ 10831186
Cost of NRE for custom = $ 21476208
COTS sigmoid adjustment = 1.78
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 5489
Crossover COTS module cost = $ 4240
Crossover custom-build module cost = $ 4240

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 104.93
Time of NRE for custom (months) = 168.00
Cost of NRE for COTS = $ 14031537
Cost of NRE for custom = $ 27830640
COTS sigmoid adjustment = 1.78
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 7116
Crossover COTS module cost = $ 4238
Crossover custom-build module cost = $ 4238

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 149.90
Time of NRE for custom (months) = 240.00
Cost of NRE for COTS = $ 20032196
Cost of NRE for custom = $ 39745200
COTS sigmoid adjustment = 1.78
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 10438
Crossover COTS module cost = $ 4135
Crossover custom-build module cost = $ 4135

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 224.84
Time of NRE for custom (months) = 360.00
Cost of NRE for COTS = $ 30033294
Cost of NRE for custom = $ 59602800
COTS sigmoid adjustment = 1.78
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 15657
Crossover COTS module cost = $ 4134
Crossover custom-build module cost = $ 4134

##############################################
##############################################

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 53.96
Time of NRE for custom (months) = 77.76
Cost of NRE for COTS = $ 7230791
Cost of NRE for custom = $ 12897725
COTS sigmoid adjustment (add 0.25 factor) = 1.19
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 2844
Crossover COTS module cost = $ 4885
Crossover custom-build module cost = $ 4886

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 69.95
Time of NRE for custom (months) = 100.80
Cost of NRE for COTS = $ 9364358
Cost of NRE for custom = $ 16710384
COTS sigmoid adjustment (add 0.25 factor) = 1.19
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 3687
Crossover COTS module cost = $ 4882
Crossover custom-build module cost = $ 4883

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 99.93
Time of NRE for custom (months) = 144.00
Cost of NRE for COTS = $ 13364797
Cost of NRE for custom = $ 23859120
COTS sigmoid adjustment (add 0.25 factor) = 1.19
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 5411
Crossover COTS module cost = $ 4737
Crossover custom-build module cost = $ 4737

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 149.90
Time of NRE for custom (months) = 216.00
Cost of NRE for COTS = $ 20032196
Cost of NRE for custom = $ 35773680
COTS sigmoid adjustment (add 0.25 factor) = 1.19
Custom sigmoid adjustment = 1.50
```

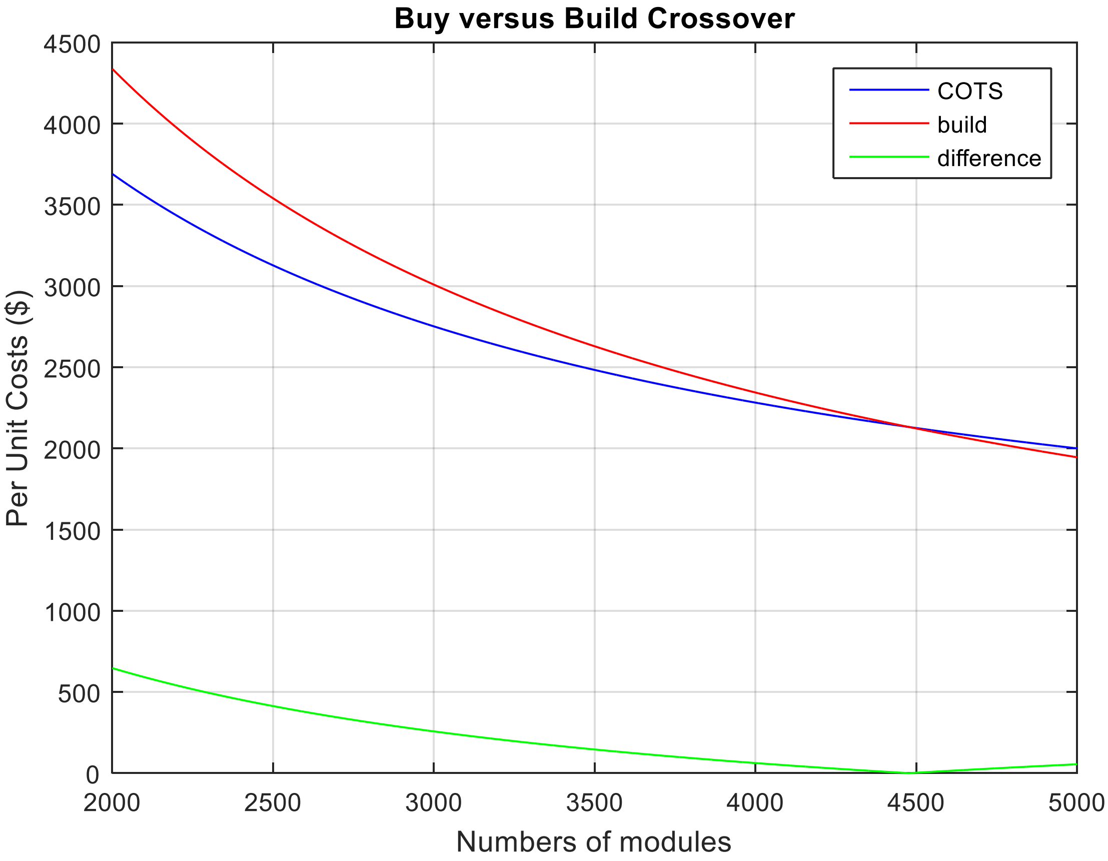
```
Crossover point (number of modules) = 8117
Crossover COTS module cost = $ 4735
Crossover custom-build module cost = $ 4735

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 107.93
Time of NRE for custom (months) = 129.60
Cost of NRE for COTS = $ 14431581
Cost of NRE for custom = $ 21476208
COTS sigmoid adjustment (add 0.25 factor) = 2.38
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 3536
Crossover COTS module cost = $ 6424
Crossover custom-build module cost = $ 6424

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 139.90
Time of NRE for custom (months) = 168.00
Cost of NRE for COTS = $ 18698716
Cost of NRE for custom = $ 27830640
COTS sigmoid adjustment (add 0.25 factor) = 2.38
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 4584
Crossover COTS module cost = $ 6422
Crossover custom-build module cost = $ 6422
```

```
----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 199.86
Time of NRE for custom (months) = 240.00
Cost of NRE for COTS = $ 26699594
Cost of NRE for custom = $ 39745200
COTS sigmoid adjustment (add 0.25 factor) = 2.38
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 6727
Crossover COTS module cost = $ 6236
Crossover custom-build module cost = $ 6236

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 299.79
Time of NRE for custom (months) = 360.00
Cost of NRE for COTS = $ 40034392
Cost of NRE for custom = $ 59602800
COTS sigmoid adjustment (add 0.25 factor) = 2.38
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 10361
Crossover COTS module cost = $ 6080
Crossover custom-build module cost = $ 6080

##############################################
##############################################
```

Model Results: iPad Touchscreen, Med Dev

August 27, 2018

```
>> [d, Q1, Q2, Q3, Q4] = init_param_NSS_iPad();

Single-lot price for COTS module = $ 430
Single-lot price for build components = $ 230
COTS features range in Monte Carlo from 0.80 to 1.00
COTS market life ranges in Monte Carlo from 0.4 to 1.0
System market life ranges in Monte Carlo from 5.0 to 10.0
Premium to inventory COTS = 2.0

Number of systems planned = 10000
Number of modules per system = 1

Average loaded hourly rates for staff:
Production staff = $ 80
Technicians = $ 80
Engineers & developers = $ 110
Management = $ 130
Administrative support = $ 60
Consulting & production staff = $ 80

COTS support and documentation for particular vendor:
Licensing support cost = $ 0
Customer support cost = $ 0
Team NRE for vendor support ranges from 0.00 to 0.20
Team NRE for customer support ranges from 0.00 to 0.20
Quality of doc & support ranges in Monte Carlo from 0.80 to 1.00
Customer interaction directly with module = 0.00

>> [St1, St2, St3, St4] = bvb_model(d, Q1, Q2, Q3, Q4);

     STAGE 1
-----------------
Time of NRE for COTS (months) = 42.00
Time of NRE for custom (months) = 48.00
Cost of NRE for COTS = $ 5634480
Cost of NRE for custom = $ 7973040

Lost opportunity cost = $ 7200
Total system revenues over lifecycle = $ 6000000
Lost opportunity cost to total revenues = 0.0012

Crossover point (number of modules) = 4476
Crossover COTS module cost = $ 2132
Crossover custom-build module cost = $ 2132
----------------------------------------------

Press spacebar
```



```
##############################################
##############################################

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 45.36
Time of NRE for custom (months) = 51.84
Cost of NRE for COTS = $ 6082838
Cost of NRE for custom = $ 8608483

Crossover point (number of modules) = 4834
Crossover COTS module cost = $ 2131
Crossover custom-build module cost = $ 2131

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 58.80
Time of NRE for custom (months) = 67.20
Cost of NRE for COTS = $ 7876272
Cost of NRE for custom = $ 11150256

Crossover point (number of modules) = 6300
Crossover COTS module cost = $ 2097
Crossover custom-build module cost = $ 2097

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 84.00
Time of NRE for custom (months) = 96.00
Cost of NRE for COTS = $ 11238960
Cost of NRE for custom = $ 15916080

Crossover point (number of modules) = 9000
Crossover COTS module cost = $ 2096
Crossover custom-build module cost = $ 2096

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 126.00
Time of NRE for custom (months) = 144.00
Cost of NRE for COTS = $ 16843440
Cost of NRE for custom = $ 23859120

Crossover point (number of modules) = 13963
Crossover COTS module cost = $ 2036
Crossover custom-build module cost = $ 2036

##############################################
##############################################

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 26.98
Time of NRE for custom (months) = 77.76
Cost of NRE for COTS = $ 3630395
Cost of NRE for custom = $ 12897725
COTS sigmoid adjustment = 0.59
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 18445
Crossover COTS module cost = $ 1027
Crossover custom-build module cost = $ 1027

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 34.98
Time of NRE for custom (months) = 100.80
Cost of NRE for COTS = $ 4697179
Cost of NRE for custom = $ 16710384
COTS sigmoid adjustment = 0.59
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 24610
Crossover COTS module cost = $ 995
Crossover custom-build module cost = $ 995

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 49.97
Time of NRE for custom (months) = 144.00
Cost of NRE for COTS = $ 6697399
Cost of NRE for custom = $ 23859120
COTS sigmoid adjustment = 0.59
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 35158
Crossover COTS module cost = $ 995
Crossover custom-build module cost = $ 995

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 74.95
Time of NRE for custom (months) = 216.00
Cost of NRE for COTS = $ 10031098
Cost of NRE for custom = $ 35773680
COTS sigmoid adjustment = 0.59
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 52737
Crossover COTS module cost = $ 994
Crossover custom-build module cost = $ 994

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 80.94
Time of NRE for custom (months) = 129.60
Cost of NRE for COTS = $ 10831186
Cost of NRE for custom = $ 21476208
COTS sigmoid adjustment = 1.78
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 21807
Crossover COTS module cost = $ 1301
Crossover custom-build module cost = $ 1301

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 104.93
Time of NRE for custom (months) = 168.00
Cost of NRE for COTS = $ 14031537
Cost of NRE for custom = $ 27830640
COTS sigmoid adjustment = 1.78
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 28269
Crossover COTS module cost = $ 1300
Crossover custom-build module cost = $ 1300

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 149.90
Time of NRE for custom (months) = 240.00
Cost of NRE for COTS = $ 20032196
Cost of NRE for custom = $ 39745200
COTS sigmoid adjustment = 1.78
Custom sigmoid adjustment = 2.50
```

```
Crossover point (number of modules) = 40385
Crossover COTS module cost = $ 1300
Crossover custom-build module cost = $ 1300

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 224.84
Time of NRE for custom (months) = 360.00
Cost of NRE for COTS = $ 30033294
Cost of NRE for custom = $ 59602800
COTS sigmoid adjustment = 1.78
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 60577
Crossover COTS module cost = $ 1300
Crossover custom-build module cost = $ 1300

##############################################
##############################################

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 53.96
Time of NRE for custom (months) = 77.76
Cost of NRE for COTS = $ 7230791
Cost of NRE for custom = $ 12897725
COTS sigmoid adjustment (add 0.25 factor) = 1.19
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 11279
Crossover COTS module cost = $ 1471
Crossover custom-build module cost = $ 1471

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 69.95
Time of NRE for custom (months) = 100.80
Cost of NRE for COTS = $ 9364358
Cost of NRE for custom = $ 16710384
COTS sigmoid adjustment (add 0.25 factor) = 1.19
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 14621
Crossover COTS module cost = $ 1470
Crossover custom-build module cost = $ 1470

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 99.93
Time of NRE for custom (months) = 144.00
Cost of NRE for COTS = $ 13364797
Cost of NRE for custom = $ 23859120
COTS sigmoid adjustment (add 0.25 factor) = 1.19
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 21499
Crossover COTS module cost = $ 1426
Crossover custom-build module cost = $ 1426

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 149.90
Time of NRE for custom (months) = 216.00
Cost of NRE for COTS = $ 20032196
Cost of NRE for custom = $ 35773680
COTS sigmoid adjustment (add 0.25 factor) = 1.19
Custom sigmoid adjustment = 1.50
```

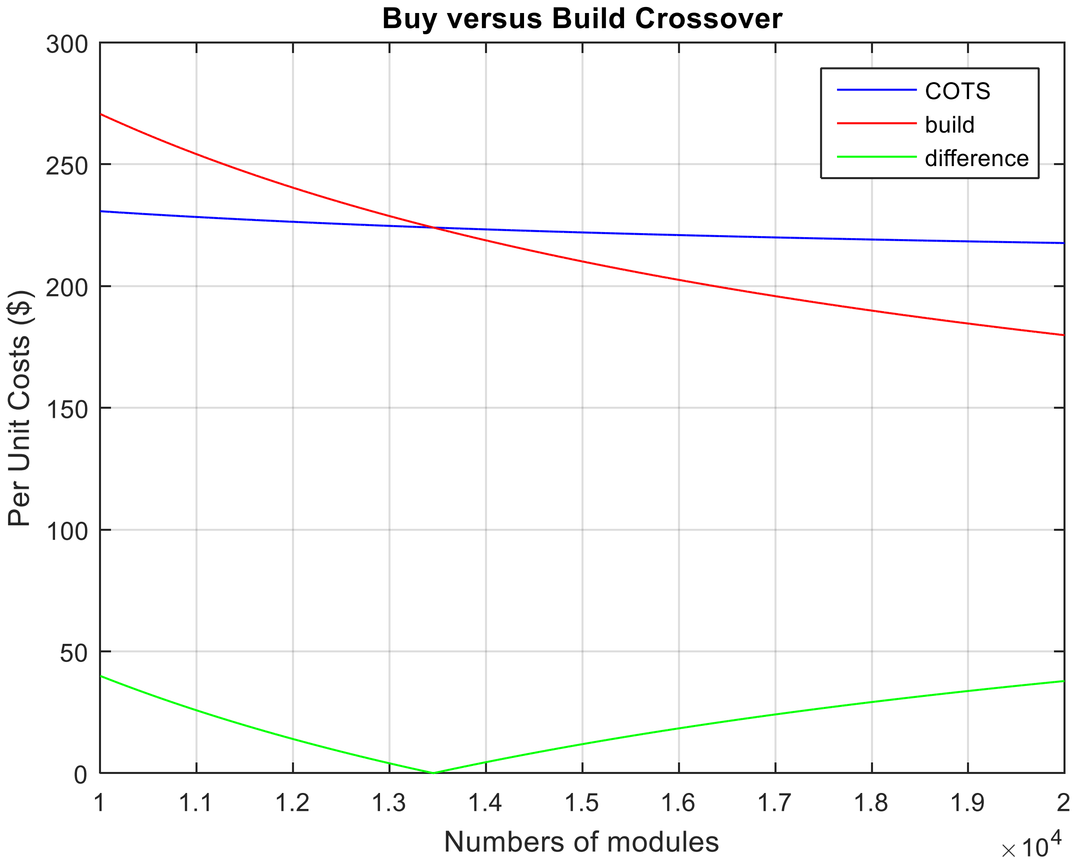
```
Crossover point (number of modules) = 32248
Crossover COTS module cost = $ 1425
Crossover custom-build module cost = $ 1425

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 107.93
Time of NRE for custom (months) = 129.60
Cost of NRE for COTS = $ 14431581
Cost of NRE for custom = $ 21476208
COTS sigmoid adjustment (add 0.25 factor) = 2.38
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 14021
Crossover COTS module cost = $ 1859
Crossover custom-build module cost = $ 1859

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 139.90
Time of NRE for custom (months) = 168.00
Cost of NRE for COTS = $ 18698716
Cost of NRE for custom = $ 27830640
COTS sigmoid adjustment (add 0.25 factor) = 2.38
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 18175
Crossover COTS module cost = $ 1859
Crossover custom-build module cost = $ 1859
```

```
----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 199.86
Time of NRE for custom (months) = 240.00
Cost of NRE for COTS = $ 26699594
Cost of NRE for custom = $ 39745200
COTS sigmoid adjustment (add 0.25 factor) = 2.38
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 26725
Crossover COTS module cost = $ 1803
Crossover custom-build module cost = $ 1803

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 299.79
Time of NRE for custom (months) = 360.00
Cost of NRE for COTS = $ 40034392
Cost of NRE for custom = $ 59602800
COTS sigmoid adjustment (add 0.25 factor) = 2.38
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 40088
Crossover COTS module cost = $ 1803
Crossover custom-build module cost = $ 1803

##############################################
##############################################
```

Model Results: AFD – PIC

August 27, 2018

```
>> [d, Q1, Q2, Q3, Q4] = init_param_AFD_PIC();

Single-lot price for COTS module = $ 67
Single-lot price for build components = $ 41
COTS features range in Monte Carlo from 0.60 to 1.00
COTS market life ranges in Monte Carlo from 2.0 to 4.0
System market life ranges in Monte Carlo from 10.0 to 20.0
Premium to inventory COTS = 3.5

Number of systems planned = 2000
Number of modules per system = 1

Average loaded hourly rates for staff:
Production staff = $ 80
Technicians = $ 80
Engineers & developers = $ 110
Management = $ 130
Administrative support = $ 60
Consulting & production staff = $ 80

COTS support and documentation for particular vendor:
Licensing support cost = $ 0
Customer support cost = $ 0
Team NRE for vendor support ranges from 0.00 to 0.40
Team NRE for customer support ranges from 0.00 to 0.40
Quality of doc & support ranges in Monte Carlo from 0.60 to 1.00
Customer interaction directly with module = 0.00

>> [St1, St2, St3, St4] = bvb_model(d, Q1, Q2, Q3, Q4);

     STAGE 1
-----------------
Time of NRE for COTS (months) = 14.00
Time of NRE for custom (months) = 21.00
Cost of NRE for COTS = $ 261500
Cost of NRE for custom = $ 1816232

Lost opportunity cost = $ 126000
Total system revenues over lifecycle = $ 18000000
Lost opportunity cost to total revenues = 0.0070

Crossover point (number of modules) = 13453
Crossover COTS module cost = $ 224
Crossover custom-build module cost = $ 224
----------------------------------------------

Press spacebar
```



```
##############################################
##############################################

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 15.12
Time of NRE for custom (months) = 22.68
Cost of NRE for COTS = $ 280820
Cost of NRE for custom = $ 1959691

Crossover point (number of modules) = 14528
Crossover COTS module cost = $ 224
Crossover custom-build module cost = $ 224

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 19.60
Time of NRE for custom (months) = 29.40
Cost of NRE for COTS = $ 358100
Cost of NRE for custom = $ 2533525

Crossover point (number of modules) = 20705
Crossover COTS module cost = $ 210
Crossover custom-build module cost = $ 210

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 28.00
Time of NRE for custom (months) = 42.00
Cost of NRE for COTS = $ 503000
Cost of NRE for custom = $ 3609464

Crossover point (number of modules) = 29567
Crossover COTS module cost = $ 210
Crossover custom-build module cost = $ 210

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 42.00
Time of NRE for custom (months) = 63.00
Cost of NRE for COTS = $ 744500
Cost of NRE for custom = $ 5402696

Crossover point (number of modules) = 44336
Crossover COTS module cost = $ 210
Crossover custom-build module cost = $ 210

##############################################
##############################################

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 31.48
Time of NRE for custom (months) = 34.02
Cost of NRE for COTS = $ 563002
Cost of NRE for custom = $ 2928036
COTS sigmoid adjustment = 2.08
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 22510
Crossover COTS module cost = $ 218
Crossover custom-build module cost = $ 218

----------------------------------------------
```

```
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 40.81
Time of NRE for custom (months) = 44.10
Cost of NRE for COTS = $ 723892
Cost of NRE for custom = $ 3788787
COTS sigmoid adjustment = 2.08
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 29171
Crossover COTS module cost = $ 218
Crossover custom-build module cost = $ 218

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 58.29
Time of NRE for custom (months) = 63.00
Cost of NRE for COTS = $ 1025560
Cost of NRE for custom = $ 5402696
COTS sigmoid adjustment = 2.08
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 41661
Crossover COTS module cost = $ 217
Crossover custom-build module cost = $ 217

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 87.44
Time of NRE for custom (months) = 94.50
Cost of NRE for COTS = $ 1528339
Cost of NRE for custom = $ 8092544
COTS sigmoid adjustment = 2.08
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 61994
Crossover COTS module cost = $ 217
Crossover custom-build module cost = $ 217

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 94.44
Time of NRE for custom (months) = 56.70
Cost of NRE for COTS = $ 1649007
Cost of NRE for custom = $ 4864726
COTS sigmoid adjustment = 6.25
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 30607
Crossover COTS module cost = $ 247
Crossover custom-build module cost = $ 247

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 122.42
Time of NRE for custom (months) = 73.50
Cost of NRE for COTS = $ 2131675
Cost of NRE for custom = $ 6299312
COTS sigmoid adjustment = 6.25
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 39667
Crossover COTS module cost = $ 247
Crossover custom-build module cost = $ 247

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 174.88
Time of NRE for custom (months) = 105.00
Cost of NRE for COTS = $ 3036679
```

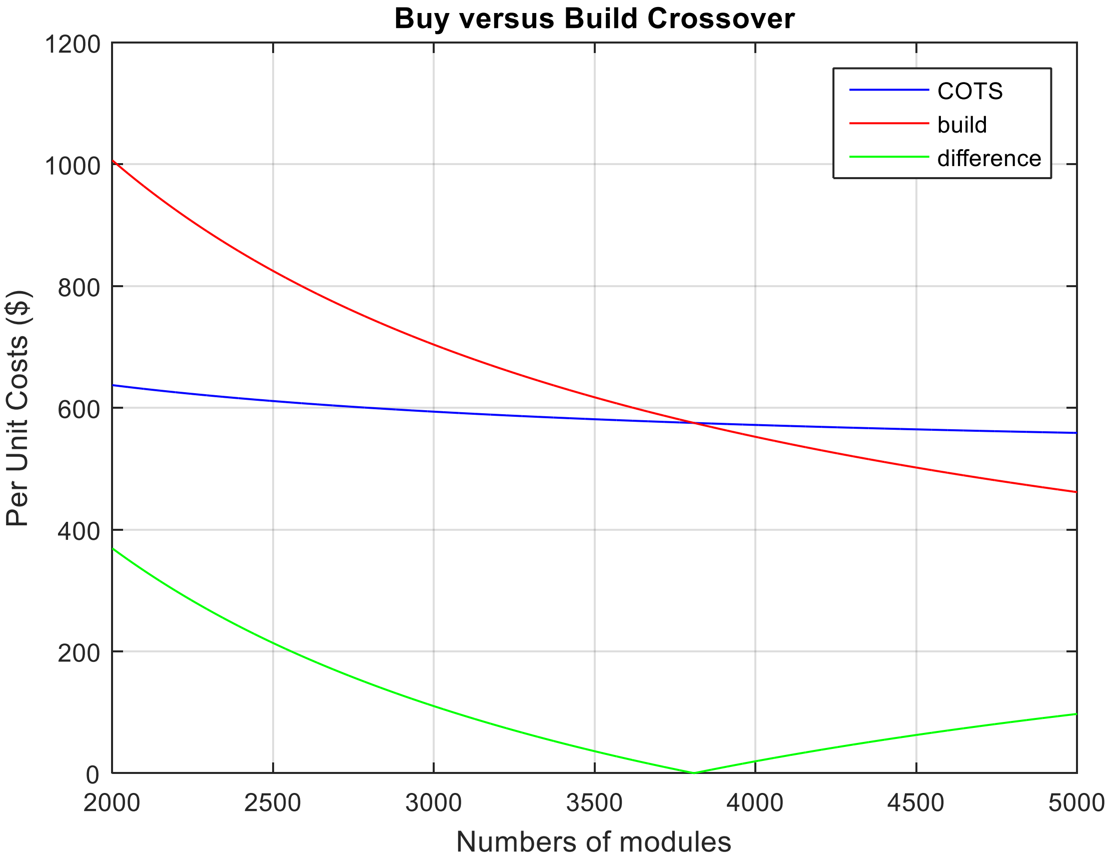
```
Cost of NRE for custom = $ 8989160
COTS sigmoid adjustment = 6.25
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 56216
Crossover COTS module cost = $ 247
Crossover custom-build module cost = $ 247

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 262.32
Time of NRE for custom (months) = 157.50
Cost of NRE for COTS = $ 4545018
Cost of NRE for custom = $ 13472240
COTS sigmoid adjustment = 6.25
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 84310
Crossover COTS module cost = $ 247
Crossover custom-build module cost = $ 247

##############################################
##############################################

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 62.96
Time of NRE for custom (months) = 34.02
Cost of NRE for COTS = $ 1106004
Cost of NRE for custom = $ 2928036
COTS sigmoid adjustment (add 0.25 factor) = 4.16
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 15767
Crossover COTS module cost = $ 275
Crossover custom-build module cost = $ 275

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 81.61
Time of NRE for custom (months) = 44.10
Cost of NRE for COTS = $ 1427783
Cost of NRE for custom = $ 3788787
COTS sigmoid adjustment (add 0.25 factor) = 4.16
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 22471
Crossover COTS module cost = $ 256
Crossover custom-build module cost = $ 256

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 116.59
Time of NRE for custom (months) = 63.00
Cost of NRE for COTS = $ 2031119
Cost of NRE for custom = $ 5402696
COTS sigmoid adjustment (add 0.25 factor) = 4.16
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 32090
Crossover COTS module cost = $ 256
Crossover custom-build module cost = $ 256

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 174.88
Time of NRE for custom (months) = 94.50
```

```
Cost of NRE for COTS = $ 3036679
Cost of NRE for custom = $ 8092544
COTS sigmoid adjustment (add 0.25 factor) = 4.16
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 48121
Crossover COTS module cost = $ 256
Crossover custom-build module cost = $ 256

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 125.91
Time of NRE for custom (months) = 56.70
Cost of NRE for COTS = $ 2192009
Cost of NRE for custom = $ 4864726
COTS sigmoid adjustment (add 0.25 factor) = 8.33
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 25438
Crossover COTS module cost = $ 279
Crossover custom-build module cost = $ 279

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 163.22
Time of NRE for custom (months) = 73.50
Cost of NRE for COTS = $ 2835567
Cost of NRE for custom = $ 6299312
COTS sigmoid adjustment (add 0.25 factor) = 8.33
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 32967
Crossover COTS module cost = $ 279
Crossover custom-build module cost = $ 279

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 233.17
Time of NRE for custom (months) = 105.00
Cost of NRE for COTS = $ 4042238
Cost of NRE for custom = $ 8989160
COTS sigmoid adjustment (add 0.25 factor) = 8.33
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 47084
Crossover COTS module cost = $ 279
Crossover custom-build module cost = $ 279

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 349.76
Time of NRE for custom (months) = 157.50
Cost of NRE for COTS = $ 6053358
Cost of NRE for custom = $ 13472240
COTS sigmoid adjustment (add 0.25 factor) = 8.33
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 70065
Crossover COTS module cost = $ 279
Crossover custom-build module cost = $ 279

##############################################
##############################################
```

Model Results: AFD – ARM

August 27, 2018

```
>> [d, Q1, Q2, Q3, Q4] = init_param_AFD_ARM();

Single-lot price for COTS module = $ 175
Single-lot price for build components = $ 41
COTS features range in Monte Carlo from 0.60 to 1.00
COTS market life ranges in Monte Carlo from 2.0 to 4.0
System market life ranges in Monte Carlo from 10.0 to 20.0
Premium to inventory COTS = 3.5

Number of systems planned = 2000
Number of modules per system = 1

Average loaded hourly rates for staff:
Production staff = $ 80
Technicians = $ 80
Engineers & developers = $ 110
Management = $ 130
Administrative support = $ 60
Consulting & production staff = $ 80

COTS support and documentation for particular vendor:
Licensing support cost = $ 0
Customer support cost = $ 0
Team NRE for vendor support ranges from 0.00 to 0.40
Team NRE for customer support ranges from 0.00 to 0.40
Quality of doc & support ranges in Monte Carlo from 0.60 to 1.00
Customer interaction directly with module = 0.00

>> [St1, St2, St3, St4] = bvb_model(d, Q1, Q2, Q3, Q4);

     STAGE 1
-----------------
Time of NRE for COTS (months) = 14.00
Time of NRE for custom (months) = 21.00
Cost of NRE for COTS = $ 261500
Cost of NRE for custom = $ 1816232

Lost opportunity cost = $ 126000
Total system revenues over lifecycle = $ 18000000
Lost opportunity cost to total revenues = 0.0070

Crossover point (number of modules) = 3808
Crossover COTS module cost = $ 575
Crossover custom-build module cost = $ 575
----------------------------------------------

Press spacebar
```



```
##############################################
##############################################

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 15.12
Time of NRE for custom (months) = 22.68
Cost of NRE for COTS = $ 280820
Cost of NRE for custom = $ 1959691

Crossover point (number of modules) = 4112
Crossover COTS module cost = $ 575
Crossover custom-build module cost = $ 575

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 19.60
Time of NRE for custom (months) = 29.40
Cost of NRE for COTS = $ 358100
Cost of NRE for custom = $ 2533525

Crossover point (number of modules) = 5313
Crossover COTS module cost = $ 574
Crossover custom-build module cost = $ 574

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 28.00
Time of NRE for custom (months) = 42.00
Cost of NRE for COTS = $ 503000
Cost of NRE for custom = $ 3609464
Crossover point (number of modules) = 7587
Crossover COTS module cost = $ 573
Crossover custom-build module cost = $ 573

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 42.00
Time of NRE for custom (months) = 63.00
Cost of NRE for COTS = $ 744500
Cost of NRE for custom = $ 5402696

Crossover point (number of modules) = 11911
Crossover COTS module cost = $ 551
Crossover custom-build module cost = $ 551

##############################################
##############################################

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 31.48
Time of NRE for custom (months) = 34.02
Cost of NRE for COTS = $ 563002
Cost of NRE for custom = $ 2928036
COTS sigmoid adjustment = 2.08
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 5776
Crossover COTS module cost = $ 604
Crossover custom-build module cost = $ 604

----------------------------------------------
```

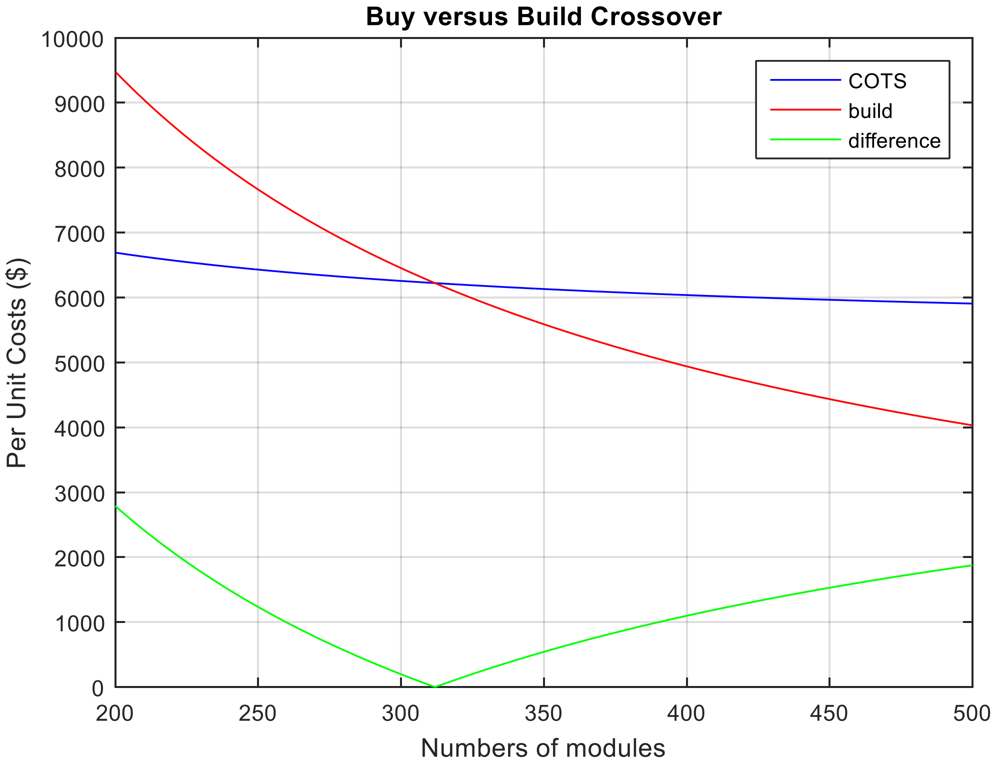
```
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 40.81
Time of NRE for custom (months) = 44.10
Cost of NRE for COTS = $ 723892
Cost of NRE for custom = $ 3788787
COTS sigmoid adjustment = 2.08
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 7485
Crossover COTS module cost = $ 603
Crossover custom-build module cost = $ 603

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 58.29
Time of NRE for custom (months) = 63.00
Cost of NRE for COTS = $ 1025560
Cost of NRE for custom = $ 5402696
COTS sigmoid adjustment = 2.08
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 11193
Crossover COTS module cost = $ 580
Crossover custom-build module cost = $ 580

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 87.44
Time of NRE for custom (months) = 94.50
Cost of NRE for COTS = $ 1528339
Cost of NRE for custom = $ 8092544
COTS sigmoid adjustment = 2.08
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 16785
Crossover COTS module cost = $ 579
Crossover custom-build module cost = $ 579

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 94.44
Time of NRE for custom (months) = 56.70
Cost of NRE for COTS = $ 1649007
Cost of NRE for custom = $ 4864726
COTS sigmoid adjustment = 6.25
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 7854
Crossover COTS module cost = $ 716
Crossover custom-build module cost = $ 716

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 122.42
Time of NRE for custom (months) = 73.50
Cost of NRE for COTS = $ 2131675
Cost of NRE for custom = $ 6299312
COTS sigmoid adjustment = 6.25
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 10657
Crossover COTS module cost = $ 688
Crossover custom-build module cost = $ 688

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 174.88
Time of NRE for custom (months) = 105.00
Cost of NRE for COTS = $ 3036679
Cost of NRE for custom = $ 8989160
COTS sigmoid adjustment = 6.25
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 15221
Crossover COTS module cost = $ 688
Crossover custom-build module cost = $ 688

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 262.32
Time of NRE for custom (months) = 157.50
Cost of NRE for COTS = $ 4545018
Cost of NRE for custom = $ 13472240
COTS sigmoid adjustment = 6.25
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 24683
Crossover COTS module cost = $ 642
Crossover custom-build module cost = $ 642

##############################################
##############################################

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 62.96
Time of NRE for custom (months) = 34.02
Cost of NRE for COTS = $ 1106004
Cost of NRE for custom = $ 2928036
COTS sigmoid adjustment (add 0.25 factor) = 4.16
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 4463
Crossover COTS module cost = $ 754
Crossover custom-build module cost = $ 754

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 81.61
Time of NRE for custom (months) = 44.10
Cost of NRE for COTS = $ 1427783
Cost of NRE for custom = $ 3788787
COTS sigmoid adjustment (add 0.25 factor) = 4.16
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 5766
Crossover COTS module cost = $ 754
Crossover custom-build module cost = $ 754

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 116.59
Time of NRE for custom (months) = 63.00
Cost of NRE for COTS = $ 2031119
Cost of NRE for custom = $ 5402696
COTS sigmoid adjustment (add 0.25 factor) = 4.16
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 8234
Crossover COTS module cost = $ 753
Crossover custom-build module cost = $ 753

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 174.88
Time of NRE for custom (months) = 94.50
```

```
Cost of NRE for COTS = $ 3036679
Cost of NRE for custom = $ 8092544
COTS sigmoid adjustment (add 0.25 factor) = 4.16
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 12928
Crossover COTS module cost = $ 723
Crossover custom-build module cost = $ 723

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 125.91
Time of NRE for custom (months) = 56.70
Cost of NRE for COTS = $ 2192009
Cost of NRE for custom = $ 4864726
COTS sigmoid adjustment (add 0.25 factor) = 8.33
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 6527
Crossover COTS module cost = $ 842
Crossover custom-build module cost = $ 842

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 163.22
Time of NRE for custom (months) = 73.50
Cost of NRE for COTS = $ 2835567
Cost of NRE for custom = $ 6299312
COTS sigmoid adjustment (add 0.25 factor) = 8.33
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 8459
Crossover COTS module cost = $ 842
Crossover custom-build module cost = $ 842

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 233.17
Time of NRE for custom (months) = 105.00
Cost of NRE for COTS = $ 4042238
Cost of NRE for custom = $ 8989160
COTS sigmoid adjustment (add 0.25 factor) = 8.33
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 12650
Crossover COTS module cost = $ 808
Crossover custom-build module cost = $ 808

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 349.76
Time of NRE for custom (months) = 157.50
Cost of NRE for COTS = $ 6053358
Cost of NRE for custom = $ 13472240
COTS sigmoid adjustment (add 0.25 factor) = 8.33
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 20513
Crossover COTS module cost = $ 753
Crossover custom-build module cost = $ 753

##############################################
##############################################
```

Model Results: AFD – Atom

August 27, 2018

```
>> [d, Q1, Q2, Q3, Q4] = init_param_AFD_Atom();

Single-lot price for COTS module = $ 1700
Single-lot price for build components = $ 400
COTS features range in Monte Carlo from 0.90 to 1.00
COTS market life ranges in Monte Carlo from 2.0 to 4.0
System market life ranges in Monte Carlo from 10.0 to 20.0
Premium to inventory COTS = 3.5

Number of systems planned = 2000
Number of modules per system = 1

Average loaded hourly rates for staff:
Production staff = $ 80
Technicians = $ 80
Engineers & developers = $ 110
Management = $ 130
Administrative support = $ 60
Consulting & production staff = $ 80

COTS support and documentation for particular vendor:
Licensing support cost = $ 0
Customer support cost = $ 0
Team NRE for vendor support ranges from 0.00 to 0.10
Team NRE for customer support ranges from 0.00 to 0.10
Quality of doc & support ranges in Monte Carlo from 0.90 to 1.00
Customer interaction directly with module = 0.00

>> [St1, St2, St3, St4] = bvb_model(d, Q1, Q2, Q3, Q4);

     STAGE 1
-----------------
Time of NRE for COTS (months) = 14.00
Time of NRE for custom (months) = 21.00
Cost of NRE for COTS = $ 261500
Cost of NRE for custom = $ 1816232

Lost opportunity cost = $ 126000
Total system revenues over lifecycle = $ 18000000
Lost opportunity cost to total revenues = 0.0070

Crossover point (number of modules) = 311
Crossover COTS module cost = $ 6225
Crossover custom-build module cost = $ 6239
----------------------------------------------

Press spacebar
```



```
##############################################
##############################################

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 15.12
Time of NRE for custom (months) = 22.68
Cost of NRE for COTS = $ 280820
Cost of NRE for custom = $ 1959691

Crossover point (number of modules) = 336
Crossover COTS module cost = $ 6219
Crossover custom-build module cost = $ 6231

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 19.60
Time of NRE for custom (months) = 29.40
Cost of NRE for COTS = $ 358100
Cost of NRE for custom = $ 2533525

Crossover point (number of modules) = 436
Crossover COTS module cost = $ 6205
Crossover custom-build module cost = $ 6209

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 28.00
Time of NRE for custom (months) = 42.00
Cost of NRE for COTS = $ 503000
Cost of NRE for custom = $ 3609464

Crossover point (number of modules) = 661
Crossover COTS module cost = $ 5847
Crossover custom-build module cost = $ 5848

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 42.00
Time of NRE for custom (months) = 63.00
Cost of NRE for COTS = $ 744500
Cost of NRE for custom = $ 5402696

Crossover point (number of modules) = 1054
Crossover COTS module cost = $ 5495
Crossover custom-build module cost = $ 5497

##############################################
##############################################

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 31.48
Time of NRE for custom (months) = 34.02
Cost of NRE for COTS = $ 563002
Cost of NRE for custom = $ 2928036
COTS sigmoid adjustment = 2.08
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 503
Crossover COTS module cost = $ 6205
Crossover custom-build module cost = $ 6208

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 40.81
Time of NRE for custom (months) = 44.10
Cost of NRE for COTS = $ 723892
Cost of NRE for custom = $ 3788787
COTS sigmoid adjustment = 2.08
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 652
Crossover COTS module cost = $ 6196
Crossover custom-build module cost = $ 6198
```

```
----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 58.29
Time of NRE for custom (months) = 63.00
Cost of NRE for COTS = $ 1025560
Cost of NRE for custom = $ 5402696
COTS sigmoid adjustment = 2.08
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 931
Crossover COTS module cost = $ 6188
Crossover custom-build module cost = $ 6190

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 87.44
Time of NRE for custom (months) = 94.50
Cost of NRE for COTS = $ 1528339
Cost of NRE for custom = $ 8092544
COTS sigmoid adjustment = 2.08
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 1485
Crossover COTS module cost = $ 5818
Crossover custom-build module cost = $ 5821

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 94.44
Time of NRE for custom (months) = 56.70
Cost of NRE for COTS = $ 1649007
Cost of NRE for custom = $ 4864726
COTS sigmoid adjustment = 6.25
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 684
Crossover COTS module cost = $ 7497
Crossover custom-build module cost = $ 7499

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 122.42
Time of NRE for custom (months) = 73.50
Cost of NRE for COTS = $ 2131675
Cost of NRE for custom = $ 6299312
COTS sigmoid adjustment = 6.25
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 886
Crossover COTS module cost = $ 7492
Crossover custom-build module cost = $ 7497

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 174.88
Time of NRE for custom (months) = 105.00
Cost of NRE for COTS = $ 3036679
Cost of NRE for custom = $ 8989160
COTS sigmoid adjustment = 6.25
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 1347
Crossover COTS module cost = $ 7043
Crossover custom-build module cost = $ 7045

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 262.32
Time of NRE for custom (months) = 157.50
Cost of NRE for COTS = $ 4545018
Cost of NRE for custom = $ 13472240
COTS sigmoid adjustment = 6.25
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 2076
Crossover COTS module cost = $ 6859
Crossover custom-build module cost = $ 6861

##############################################
##############################################

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 62.96
Time of NRE for custom (months) = 34.02
Cost of NRE for COTS = $ 1106004
Cost of NRE for custom = $ 2928036
COTS sigmoid adjustment (add 0.25 factor) = 4.16
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 365
Crossover COTS module cost = $ 8414
Crossover custom-build module cost = $ 8421

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 81.61
Time of NRE for custom (months) = 44.10
Cost of NRE for COTS = $ 1427783
Cost of NRE for custom = $ 3788787
COTS sigmoid adjustment (add 0.25 factor) = 4.16
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 502
Crossover COTS module cost = $ 7930
Crossover custom-build module cost = $ 7934

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 116.59
Time of NRE for custom (months) = 63.00
Cost of NRE for COTS = $ 2031119
Cost of NRE for custom = $ 5402696
COTS sigmoid adjustment (add 0.25 factor) = 4.16
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 717
Crossover COTS module cost = $ 7919
Crossover custom-build module cost = $ 7922

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 174.88
Time of NRE for custom (months) = 94.50
Cost of NRE for COTS = $ 3036679
Cost of NRE for custom = $ 8092544
COTS sigmoid adjustment (add 0.25 factor) = 4.16
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 1144
Crossover COTS module cost = $ 7443
Crossover custom-build module cost = $ 7445

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.08
----------------------------------------------
```

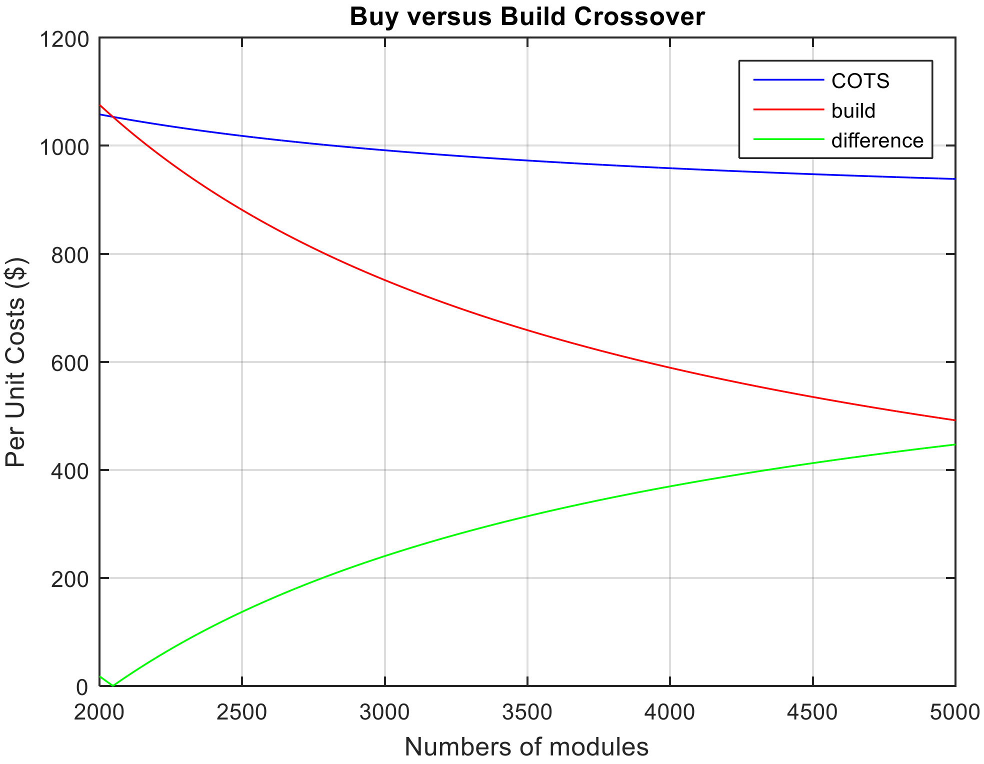
```
Time of NRE for COTS (months) = 125.91
Time of NRE for custom (months) = 56.70
Cost of NRE for COTS = $ 2192009
Cost of NRE for custom = $ 4864726
COTS sigmoid adjustment (add 0.25 factor) = 8.33
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 568
Crossover COTS module cost = $ 8945
Crossover custom-build module cost = $ 8952

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 163.22
Time of NRE for custom (months) = 73.50
Cost of NRE for COTS = $ 2835567
Cost of NRE for custom = $ 6299312
COTS sigmoid adjustment (add 0.25 factor) = 8.33
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 737
Crossover COTS module cost = $ 8934
Crossover custom-build module cost = $ 8934

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 2.00
```

```
----------------------------------------------
Time of NRE for COTS (months) = 233.17
Time of NRE for custom (months) = 105.00
Cost of NRE for COTS = $ 4042238
Cost of NRE for custom = $ 8989160
COTS sigmoid adjustment (add 0.25 factor) = 8.33
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 1119
Crossover COTS module cost = $ 8401
Crossover custom-build module cost = $ 8404

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 349.76
Time of NRE for custom (months) = 157.50
Cost of NRE for COTS = $ 6053358
Cost of NRE for custom = $ 13472240
COTS sigmoid adjustment (add 0.25 factor) = 8.33
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 1679
Crossover COTS module cost = $ 8394
Crossover custom-build module cost = $ 8395

##############################################
##############################################
```

Model Results: Grain Bin Controller

August 27, 2018

```
>> [d, Q1, Q2, Q3, Q4] = init_param_grain_bin();

Single-lot price for COTS module = $ 175
Single-lot price for build components = $ 41
COTS features range in Monte Carlo from 0.60 to 1.00
COTS market life ranges in Monte Carlo from 2.0 to 4.0
System market life ranges in Monte Carlo from 10.0 to 30.0
Premium to inventory COTS = 6.0

Number of systems planned = 40000
Number of modules per system = 1

Average loaded hourly rates for staff:
Production staff = $ 80
Technicians = $ 80
Engineers & developers = $ 110
Management = $ 130
Administrative support = $ 60
Consulting & production staff = $ 80

COTS support and documentation for particular vendor:
Licensing support cost = $ 0
Customer support cost = $ 0
Team NRE for vendor support ranges from 0.00 to 0.40
Team NRE for customer support ranges from 0.00 to 0.40
Quality of doc & support ranges in Monte Carlo from 0.60 to 1.00
Customer interaction directly with module = 0.00

>> [St1, St2, St3, St4] = bvb_model(d, Q1, Q2, Q3, Q4);

     STAGE 1
-----------------
Time of NRE for COTS (months) = 16.00
Time of NRE for custom (months) = 21.00
Cost of NRE for COTS = $ 398588
Cost of NRE for custom = $ 1947112

Lost opportunity cost = $ 120000
Total system revenues over lifecycle = $ 480000000
Lost opportunity cost to total revenues = 0.0003

Crossover point (number of modules) = 2046
Crossover COTS module cost = $ 1054
Crossover custom-build module cost = $ 1054
----------------------------------------------

Press spacebar
```



```
##############################################
##############################################

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 17.28
Time of NRE for custom (months) = 22.68
Cost of NRE for COTS = $ 428875
Cost of NRE for custom = $ 2099441

Crossover point (number of modules) = 2208
Crossover COTS module cost = $ 1053
Crossover custom-build module cost = $ 1053

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 22.40
Time of NRE for custom (months) = 29.40
Cost of NRE for COTS = $ 550023
Cost of NRE for custom = $ 2708757

Crossover point (number of modules) = 2853
Crossover COTS module cost = $ 1052
Crossover custom-build module cost = $ 1052

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 32.00
Time of NRE for custom (months) = 42.00
Cost of NRE for COTS = $ 777176
Cost of NRE for custom = $ 3851224

Crossover point (number of modules) = 4063
Crossover COTS module cost = $ 1050
Crossover custom-build module cost = $ 1050
```

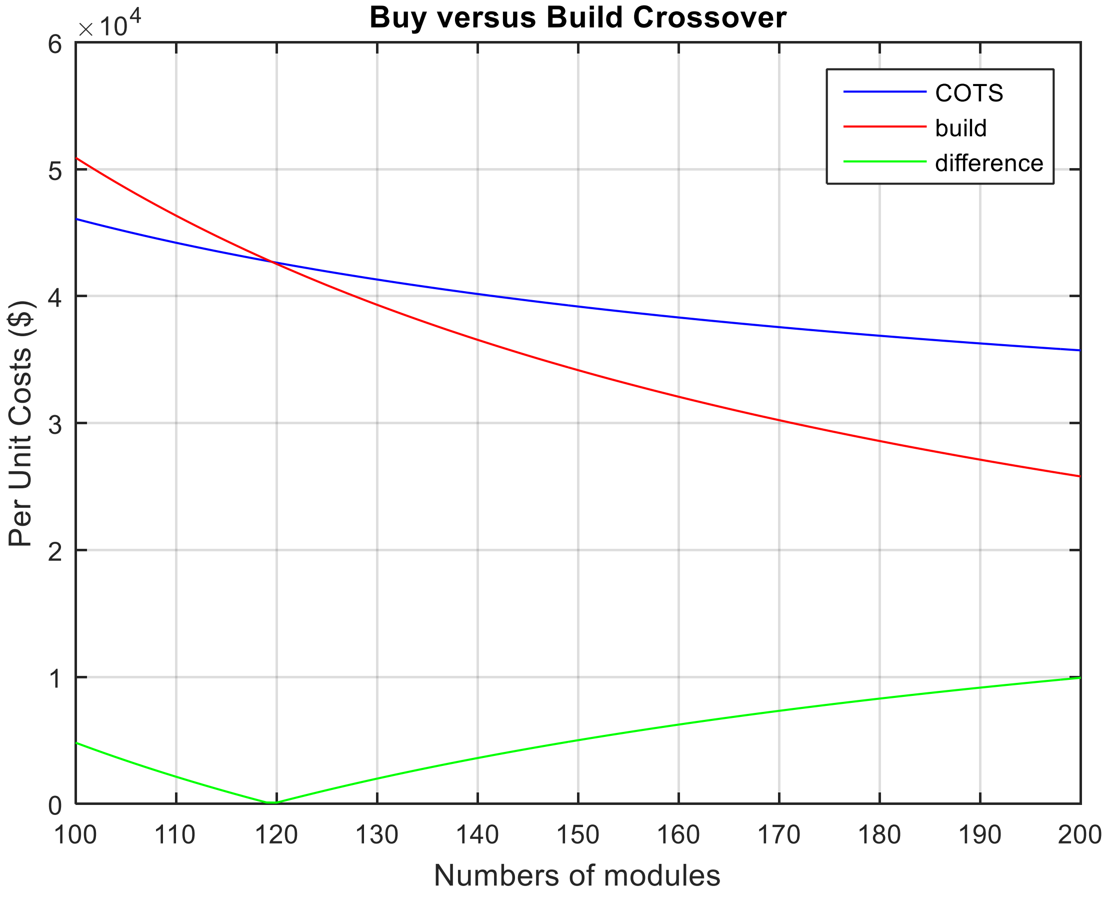
```
----------------------------------------------
     STAGE 2: Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 48.00
Time of NRE for custom (months) = 63.00
Cost of NRE for COTS = $ 1155764
Cost of NRE for custom = $ 5755336

Crossover point (number of modules) = 6069
Crossover COTS module cost = $ 1049
Crossover custom-build module cost = $ 1049

##############################################
##############################################

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 26.32
Time of NRE for custom (months) = 34.02
Cost of NRE for COTS = $ 642773
Cost of NRE for custom = $ 3127661
COTS sigmoid adjustment = 1.52
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 3284
Crossover COTS module cost = $ 1054
Crossover custom-build module cost = $ 1055

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 34.12
Time of NRE for custom (months) = 44.10
Cost of NRE for COTS = $ 827298
Cost of NRE for custom = $ 4041635
COTS sigmoid adjustment = 1.52
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 4248
Crossover COTS module cost = $ 1054
Crossover custom-build module cost = $ 1054

---------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 2.00
---------------------------------------------
Time of NRE for COTS (months) = 48.74
Time of NRE for custom (months) = 63.00
Cost of NRE for COTS = $ 1173283
Cost of NRE for custom = $ 5755336
COTS sigmoid adjustment = 1.52
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 6046
Crossover COTS module cost = $ 1053
Crossover custom-build module cost = $ 1053

---------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 3.00
---------------------------------------------
Time of NRE for COTS (months) = 73.11
Time of NRE for custom (months) = 94.50
Cost of NRE for COTS = $ 1749925
Cost of NRE for custom = $ 8611504
COTS sigmoid adjustment = 1.52
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 9054
Crossover COTS module cost = $ 1052
Crossover custom-build module cost = $ 1052

---------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.08
---------------------------------------------
Time of NRE for COTS (months) = 78.96
Time of NRE for custom (months) = 56.70
Cost of NRE for COTS = $ 1888319
Cost of NRE for custom = $ 5184102
COTS sigmoid adjustment = 4.57
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 4356
Crossover COTS module cost = $ 1292
Crossover custom-build module cost = $ 1292

---------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.40
---------------------------------------------
Time of NRE for COTS (months) = 102.35
Time of NRE for custom (months) = 73.50
Cost of NRE for COTS = $ 2441895
Cost of NRE for custom = $ 6707392
COTS sigmoid adjustment = 4.57
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 5629
Crossover COTS module cost = $ 1293
Crossover custom-build module cost = $ 1293

---------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 2.00
---------------------------------------------
Time of NRE for COTS (months) = 146.22
Time of NRE for custom (months) = 105.00
Cost of NRE for COTS = $ 3479850
Cost of NRE for custom = $ 9563560
COTS sigmoid adjustment = 4.57
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 8028
Crossover COTS module cost = $ 1292
Crossover custom-build module cost = $ 1292

---------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 3.00
---------------------------------------------
Time of NRE for COTS (months) = 219.33
Time of NRE for custom (months) = 157.50
Cost of NRE for COTS = $ 5209775
Cost of NRE for custom = $ 14323840
COTS sigmoid adjustment = 4.57
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 12549
Crossover COTS module cost = $ 1242
Crossover custom-build module cost = $ 1242

#############################################
#############################################

---------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.08
---------------------------------------------
Time of NRE for COTS (months) = 52.64
Time of NRE for custom (months) = 34.02
Cost of NRE for COTS = $ 1265546
Cost of NRE for custom = $ 3127661
COTS sigmoid adjustment (add 0.25 factor) = 3.05
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 2461
Crossover COTS module cost = $ 1373
Crossover custom-build module cost = $ 1373

---------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.40
---------------------------------------------
Time of NRE for COTS (months) = 68.24
Time of NRE for custom (months) = 44.10
Cost of NRE for COTS = $ 1634597
Cost of NRE for custom = $ 4041635
COTS sigmoid adjustment (add 0.25 factor) = 3.05
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 3181
Crossover COTS module cost = $ 1373
Crossover custom-build module cost = $ 1373

---------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 2.00
---------------------------------------------
Time of NRE for COTS (months) = 97.48
Time of NRE for custom (months) = 63.00
Cost of NRE for COTS = $ 2326567
Cost of NRE for custom = $ 5755336
COTS sigmoid adjustment (add 0.25 factor) = 3.05
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 4532
Crossover COTS module cost = $ 1372
Crossover custom-build module cost = $ 1372

---------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 3.00
---------------------------------------------
Time of NRE for COTS (months) = 146.22
Time of NRE for custom (months) = 94.50
Cost of NRE for COTS = $ 3479850
Cost of NRE for custom = $ 8611504
COTS sigmoid adjustment (add 0.25 factor) = 3.05
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 6772
Crossover COTS module cost = $ 1373
Crossover custom-build module cost = $ 1373

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 105.28
Time of NRE for custom (months) = 56.70
Cost of NRE for COTS = $ 2511092
Cost of NRE for custom = $ 5184102
COTS sigmoid adjustment (add 0.25 factor) = 6.09
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 3533
Crossover COTS module cost = $ 1570
Crossover custom-build module cost = $ 1570

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 136.47
Time of NRE for custom (months) = 73.50
Cost of NRE for COTS = $ 3249193
Cost of NRE for custom = $ 6707392
COTS sigmoid adjustment (add 0.25 factor) = 6.09
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 4571
Crossover COTS module cost = $ 1570
Crossover custom-build module cost = $ 1570
```

```
----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 194.96
Time of NRE for custom (months) = 105.00
Cost of NRE for COTS = $ 4633133
Cost of NRE for custom = $ 9563560
COTS sigmoid adjustment (add 0.25 factor) = 6.09
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 6506
Crossover COTS module cost = $ 1571
Crossover custom-build module cost = $ 1571

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 292.44
Time of NRE for custom (months) = 157.50
Cost of NRE for COTS = $ 6939700
Cost of NRE for custom = $ 14323840
COTS sigmoid adjustment (add 0.25 factor) = 6.09
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 10167
Crossover COTS module cost = $ 1510
Crossover custom-build module cost = $ 1510

##############################################
##############################################
```

Model Results: VOC Sensor Controller

August 27, 2018

```
>> [d, Q1, Q2, Q3, Q4] = init_param_VOC_control();

Single-lot price for COTS module = $ 9375
Single-lot price for build components = $ 705
COTS features range in Monte Carlo from 0.60 to 1.00
COTS market life ranges in Monte Carlo from 5.0 to 15.0
System market life ranges in Monte Carlo from 10.0 to 30.0
Premium to inventory COTS = 3.0

Number of systems planned = 5000
Number of modules per system = 1

Average loaded hourly rates for staff:
Production staff = $ 80
Technicians = $ 80
Engineers & developers = $ 110
Management = $ 130
Administrative support = $ 60
Consulting & production staff = $ 80

COTS support and documentation for particular vendor:
Licensing support cost = $ 0
Customer support cost = $ 0
Team NRE for vendor support ranges from 0.00 to 0.40
Team NRE for customer support ranges from 0.00 to 0.40
Quality of doc & support ranges in Monte Carlo from 0.60 to 1.00
Customer interaction directly with module = 0.00

>> [St1, St2, St3, St4] = bvb_model(d, Q1, Q2, Q3, Q4);

     STAGE 1
-----------------
Time of NRE for COTS (months) = 24.00
Time of NRE for custom (months) = 36.00
Cost of NRE for COTS = $ 2072128
Cost of NRE for custom = $ 5022520

Lost opportunity cost = $ 2400000
Total system revenues over lifecycle = $ 500000000
Lost opportunity cost to total revenues = 0.0048

Crossover point (number of modules) = 119
Crossover COTS module cost = $ 42778
Crossover custom-build module cost = $ 42887
----------------------------------------------

Press spacebar
```



```
##############################################
##############################################

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 25.92
Time of NRE for custom (months) = 38.88
Cost of NRE for COTS = $ 2234698
Cost of NRE for custom = $ 5420882

Crossover point (number of modules) = 129
Crossover COTS module cost = $ 42689
Crossover custom-build module cost = $ 42703

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 33.60
Time of NRE for custom (months) = 50.40
Cost of NRE for COTS = $ 2884979
Cost of NRE for custom = $ 7014328

Crossover point (number of modules) = 167
Crossover COTS module cost = $ 42641
Crossover custom-build module cost = $ 42683

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 48.00
Time of NRE for custom (months) = 72.00
Cost of NRE for COTS = $ 4104256
Cost of NRE for custom = $ 10002040

Crossover point (number of modules) = 238
Crossover COTS module cost = $ 42610
Crossover custom-build module cost = $ 42706

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 72.00
Time of NRE for custom (months) = 108.00
Cost of NRE for COTS = $ 6136384
Cost of NRE for custom = $ 14981560

Crossover point (number of modules) = 358
Crossover COTS module cost = $ 42506
Crossover custom-build module cost = $ 42529

##############################################
##############################################

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 22.40
Time of NRE for custom (months) = 58.32
Cost of NRE for COTS = $ 1936296
Cost of NRE for custom = $ 8109822
COTS sigmoid adjustment = 0.86
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 250
Crossover COTS module cost = $ 33111
Crossover custom-build module cost = $ 33120

----------------------------------------------
```

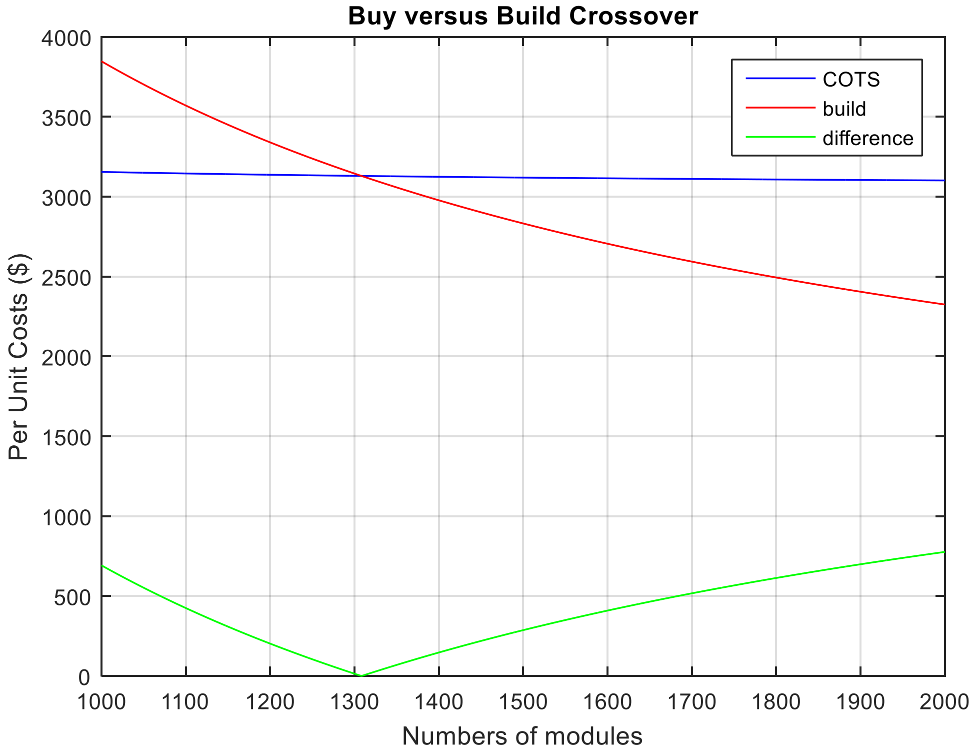
```
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 29.03
Time of NRE for custom (months) = 75.60
Cost of NRE for COTS = $ 2498162
Cost of NRE for custom = $ 10499992
COTS sigmoid adjustment = 0.86
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 324
Crossover COTS module cost = $ 33076
Crossover custom-build module cost = $ 33088

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 41.47
Time of NRE for custom (months) = 108.00
Cost of NRE for COTS = $ 3551660
Cost of NRE for custom = $ 14981560
COTS sigmoid adjustment = 0.86
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 463
Crossover COTS module cost = $ 33037
Crossover custom-build module cost = $ 33038

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 62.21
Time of NRE for custom (months) = 162.00
Cost of NRE for COTS = $ 5307490
Cost of NRE for custom = $ 22450840
COTS sigmoid adjustment = 0.86
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 735
Crossover COTS module cost = $ 31180
Crossover custom-build module cost = $ 31206

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 67.19
Time of NRE for custom (months) = 97.20
Cost of NRE for COTS = $ 5728889
Cost of NRE for custom = $ 13487704
COTS sigmoid adjustment = 2.59
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 314
Crossover COTS module cost = $ 43610
Crossover custom-build module cost = $ 43635

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 87.09
Time of NRE for custom (months) = 126.00
Cost of NRE for COTS = $ 7414486
Cost of NRE for custom = $ 17471320
COTS sigmoid adjustment = 2.59
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 407
Crossover COTS module cost = $ 43583
Crossover custom-build module cost = $ 43608

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 124.42
Time of NRE for custom (months) = 180.00
Cost of NRE for COTS = $ 10574979
Cost of NRE for custom = $ 24940600
COTS sigmoid adjustment = 2.59
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 616
Crossover COTS module cost = $ 41126
Crossover custom-build module cost = $ 41148

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 186.63
Time of NRE for custom (months) = 270.00
Cost of NRE for COTS = $ 15842469
Cost of NRE for custom = $ 37389400
COTS sigmoid adjustment = 2.59
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 924
Crossover COTS module cost = $ 41105
Crossover custom-build module cost = $ 41125

##############################################
##############################################

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 44.79
Time of NRE for custom (months) = 58.32
Cost of NRE for COTS = $ 3832593
Cost of NRE for custom = $ 8109822
COTS sigmoid adjustment (add 0.25 factor) = 1.73
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 173
Crossover COTS module cost = $ 47519
Crossover custom-build module cost = $ 47558

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 58.06
Time of NRE for custom (months) = 75.60
Cost of NRE for COTS = $ 4956324
Cost of NRE for custom = $ 10499992
COTS sigmoid adjustment (add 0.25 factor) = 1.73
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 224
Crossover COTS module cost = $ 47492
Crossover custom-build module cost = $ 47556

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 82.95
Time of NRE for custom (months) = 108.00
Cost of NRE for COTS = $ 7063320
Cost of NRE for custom = $ 14981560
COTS sigmoid adjustment (add 0.25 factor) = 1.73
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 320
Crossover COTS module cost = $ 47438
Crossover custom-build module cost = $ 47498

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 124.42
Time of NRE for custom (months) = 162.00
```

```
Cost of NRE for COTS = $ 10574979
Cost of NRE for custom = $ 22450840
COTS sigmoid adjustment (add 0.25 factor) = 1.73
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 509
Crossover COTS module cost = $ 44735
Crossover custom-build module cost = $ 44768

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 89.58
Time of NRE for custom (months) = 97.20
Cost of NRE for COTS = $ 7625185
Cost of NRE for custom = $ 13487704
COTS sigmoid adjustment (add 0.25 factor) = 3.46
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 237
Crossover COTS module cost = $ 57539
Crossover custom-build module cost = $ 57591

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 116.13
Time of NRE for custom (months) = 126.00
Cost of NRE for COTS = $ 9872647
Cost of NRE for custom = $ 17471320
COTS sigmoid adjustment (add 0.25 factor) = 3.46
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 307
Crossover COTS module cost = $ 57524
Crossover custom-build module cost = $ 57591

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 165.89
Time of NRE for custom (months) = 180.00
Cost of NRE for COTS = $ 14086639
Cost of NRE for custom = $ 24940600
COTS sigmoid adjustment (add 0.25 factor) = 3.46
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 439
Crossover COTS module cost = $ 57454
Crossover custom-build module cost = $ 57493

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 248.84
Time of NRE for custom (months) = 270.00
Cost of NRE for COTS = $ 21109959
Cost of NRE for custom = $ 37389400
COTS sigmoid adjustment (add 0.25 factor) = 3.46
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 698
Crossover COTS module cost = $ 54203
Crossover custom-build module cost = $ 54227

##############################################
##############################################
```

Model Results: $6K v $900 Processor Board

August 27, 2018

```
>> [d, Q1, Q2, Q3, Q4] = init_param_6K_v_9();

Single-lot price for COTS module = $ 6000
Single-lot price for build components = $ 900
COTS features range in Monte Carlo from 0.60 to 1.00
COTS market life ranges in Monte Carlo from 2.0 to 4.0
System market life ranges in Monte Carlo from 4.0 to 8.0
Premium to inventory COTS = 1.0

Number of systems planned = 10000
Number of modules per system = 1

Average loaded hourly rates for staff:
Production staff = $ 80
Technicians = $ 80
Engineers & developers = $ 110
Management = $ 130
Administrative support = $ 60
Consulting & production staff = $ 80

COTS support and documentation for particular vendor:
Licensing support cost = $ 25000
Customer support cost = $ 25000
Team NRE for vendor support ranges from 0.00 to 0.40
Team NRE for customer support ranges from 0.00 to 0.40
Quality of doc & support ranges in Monte Carlo from 0.60 to 1.00
Customer interaction directly with module = 1.00

>> [St1, St2, St3, St4] = bvb_model(d, Q1, Q2, Q3, Q4);

     STAGE 1
-----------------
Time of NRE for COTS (months) = 6.00
Time of NRE for custom (months) = 30.00
Cost of NRE for COTS = $ 106596
Cost of NRE for custom = $ 3044640

Lost opportunity cost = $ 1920000
Total system revenues over lifecycle = $ 400000000
Lost opportunity cost to total revenues = 0.0048

Crossover point (number of modules) = 1308
Crossover COTS module cost = $ 3130
Crossover custom-build module cost = $ 3130
----------------------------------------------

Press spacebar
```



```
##############################################
##############################################

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 6.48
Time of NRE for custom (months) = 32.40
Cost of NRE for COTS = $ 115124
Cost of NRE for custom = $ 3287971

Crossover point (number of modules) = 1412
Crossover COTS module cost = $ 3130
Crossover custom-build module cost = $ 3131

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 8.40
Time of NRE for custom (months) = 42.00
Cost of NRE for COTS = $ 149234
Cost of NRE for custom = $ 4261296

Crossover point (number of modules) = 1831
Crossover COTS module cost = $ 3130
Crossover custom-build module cost = $ 3130

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 12.00
Time of NRE for custom (months) = 60.00
Cost of NRE for COTS = $ 213192
Cost of NRE for custom = $ 6086280

Crossover point (number of modules) = 2514
Crossover COTS module cost = $ 3133
Crossover custom-build module cost = $ 3133

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 18.00
Time of NRE for custom (months) = 90.00
Cost of NRE for COTS = $ 319788
Cost of NRE for custom = $ 9127920

Crossover point (number of modules) = 3771
Crossover COTS module cost = $ 3133
Crossover custom-build module cost = $ 3133

##############################################
##############################################

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 47.74
Time of NRE for custom (months) = 48.60
Cost of NRE for COTS = $ 848191
Cost of NRE for custom = $ 4930457
COTS sigmoid adjustment = 7.37
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 1817
Crossover COTS module cost = $ 3515
Crossover custom-build module cost = $ 3516

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 61.89
Time of NRE for custom (months) = 63.00
Cost of NRE for COTS = $ 1099506
Cost of NRE for custom = $ 6390444
COTS sigmoid adjustment = 7.37
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 2265
Crossover COTS module cost = $ 3534
Crossover custom-build module cost = $ 3534
```

```
----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 88.41
Time of NRE for custom (months) = 90.00
Cost of NRE for COTS = $ 1570723
Cost of NRE for custom = $ 9127920
COTS sigmoid adjustment = 7.37
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 3235
Crossover COTS module cost = $ 3534
Crossover custom-build module cost = $ 3534

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 132.62
Time of NRE for custom (months) = 135.00
Cost of NRE for COTS = $ 2356085
Cost of NRE for custom = $ 13690380
COTS sigmoid adjustment = 7.37
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 4852
Crossover COTS module cost = $ 3534
Crossover custom-build module cost = $ 3534

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 143.23
Time of NRE for custom (months) = 81.00
Cost of NRE for COTS = $ 2544572
Cost of NRE for custom = $ 8215428
COTS sigmoid adjustment = 22.10
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 2427
Crossover COTS module cost = $ 4097
Crossover custom-build module cost = $ 4098

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 185.66
Time of NRE for custom (months) = 105.00
Cost of NRE for COTS = $ 3298519
Cost of NRE for custom = $ 10648740
COTS sigmoid adjustment = 22.10
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 3146
Crossover COTS module cost = $ 4097
Crossover custom-build module cost = $ 4097

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 265.24
Time of NRE for custom (months) = 150.00
Cost of NRE for COTS = $ 4712170
Cost of NRE for custom = $ 15211200
COTS sigmoid adjustment = 22.10
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 4495
Crossover COTS module cost = $ 4096
Crossover custom-build module cost = $ 4097

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 397.85
Time of NRE for custom (months) = 225.00
Cost of NRE for COTS = $ 7068255
Cost of NRE for custom = $ 22815300
COTS sigmoid adjustment = 22.10
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 6491
Crossover COTS module cost = $ 4137
Crossover custom-build module cost = $ 4137

##############################################
##############################################

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 95.48
Time of NRE for custom (months) = 48.60
Cost of NRE for COTS = $ 1696381
Cost of NRE for custom = $ 4930457
COTS sigmoid adjustment (add 0.25 factor) = 14.74
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 1440
Crossover COTS module cost = $ 4226
Crossover custom-build module cost = $ 4226

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 123.78
Time of NRE for custom (months) = 63.00
Cost of NRE for COTS = $ 2199013
Cost of NRE for custom = $ 6390444
COTS sigmoid adjustment (add 0.25 factor) = 14.74
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 1866
Crossover COTS module cost = $ 4227
Crossover custom-build module cost = $ 4227

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 176.82
Time of NRE for custom (months) = 90.00
Cost of NRE for COTS = $ 3141447
Cost of NRE for custom = $ 9127920
COTS sigmoid adjustment (add 0.25 factor) = 14.74
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 2563
Crossover COTS module cost = $ 4274
Crossover custom-build module cost = $ 4274

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 265.24
Time of NRE for custom (months) = 135.00
Cost of NRE for COTS = $ 4712170
Cost of NRE for custom = $ 13690380
COTS sigmoid adjustment (add 0.25 factor) = 14.74
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 3843
Crossover COTS module cost = $ 4274
Crossover custom-build module cost = $ 4275

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.08
----------------------------------------------
```

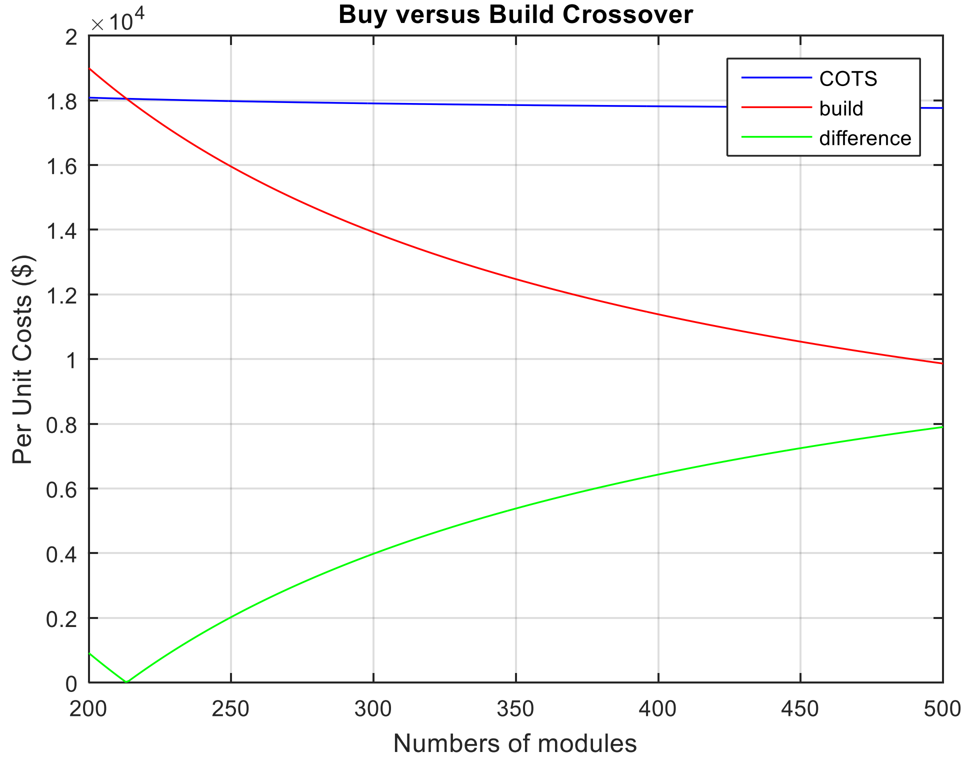
```
Time of NRE for COTS (months) = 190.97
Time of NRE for custom (months) = 81.00
Cost of NRE for COTS = $ 3392763
Cost of NRE for custom = $ 8215428
COTS sigmoid adjustment (add 0.25 factor) = 29.47
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 2064
Crossover COTS module cost = $ 4692
Crossover custom-build module cost = $ 4693

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 247.55
Time of NRE for custom (months) = 105.00
Cost of NRE for COTS = $ 4398026
Cost of NRE for custom = $ 10648740
COTS sigmoid adjustment (add 0.25 factor) = 29.47
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 2676
Crossover COTS module cost = $ 4692
Crossover custom-build module cost = $ 4692

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 2.00
```

```
----------------------------------------------
Time of NRE for COTS (months) = 353.65
Time of NRE for custom (months) = 150.00
Cost of NRE for COTS = $ 6282894
Cost of NRE for custom = $ 15211200
COTS sigmoid adjustment (add 0.25 factor) = 29.47
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 3822
Crossover COTS module cost = $ 4692
Crossover custom-build module cost = $ 4692

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 530.47
Time of NRE for custom (months) = 225.00
Cost of NRE for COTS = $ 9424340
Cost of NRE for custom = $ 22815300
COTS sigmoid adjustment (add 0.25 factor) = 29.47
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 5520
Crossover COTS module cost = $ 4755
Crossover custom-build module cost = $ 4756

##############################################
##############################################
```

Model Results for $25K v $4K boards

August 27, 2018

```
>> [d, Q1, Q2, Q3, Q4] = init_param_25K_v_4K();

Single-lot price for COTS module = $ 25000
Single-lot price for build components = $ 4000
COTS features range in Monte Carlo from 0.60 to 1.00
COTS market life ranges in Monte Carlo from 2.0 to 4.0
System market life ranges in Monte Carlo from 4.0 to 8.0
Premium to inventory COTS = 1.0

Number of systems planned = 10000
Number of modules per system = 1

Average loaded hourly rates for staff:
Production staff = $ 80
Technicians = $ 80
Engineers & developers = $ 110
Management = $ 130
Administrative support = $ 60
Consulting & production staff = $ 80

COTS support and documentation for particular vendor:
Licensing support cost = $ 25000
Customer support cost = $ 25000
Team NRE for vendor support ranges from 0.00 to 0.40
Team NRE for customer support ranges from 0.00 to 0.40
Quality of doc & support ranges in Monte Carlo from 0.60 to 1.00
Customer interaction directly with module = 1.00

>> [St1, St2, St3, St4] = bvb_model(d, Q1, Q2, Q3, Q4);

     STAGE 1
-----------------
Time of NRE for COTS (months) = 6.00
Time of NRE for custom (months) = 30.00
Cost of NRE for COTS = $ 106596
Cost of NRE for custom = $ 3044640

Lost opportunity cost = $ 1920000
Total system revenues over lifecycle = $ 400000000
Lost opportunity cost to total revenues = 0.0048

Crossover point (number of modules) = 213
Crossover COTS module cost = $ 18049
Crossover custom-build module cost = $ 18067
---------------------------------------------

Press spacebar
```



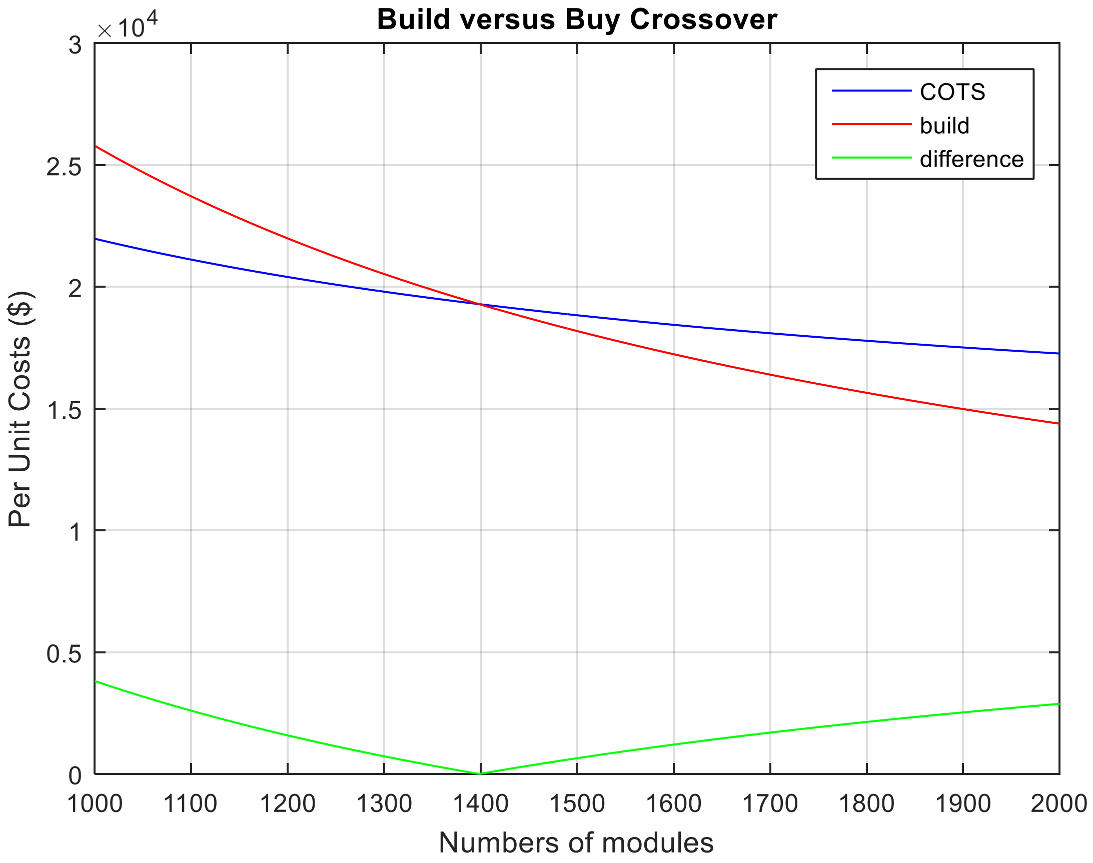
```
##############################################
##############################################

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 6.48
Time of NRE for custom (months) = 32.40
Cost of NRE for COTS = $ 115124
Cost of NRE for custom = $ 3287971

Crossover point (number of modules) = 230
Crossover COTS module cost = $ 18049
Crossover custom-build module cost = $ 18068

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 8.40
Time of NRE for custom (months) = 42.00
Cost of NRE for COTS = $ 149234
Cost of NRE for custom = $ 4261296

Crossover point (number of modules) = 298
Crossover COTS module cost = $ 18049
Crossover custom-build module cost = $ 18072

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 12.00
Time of NRE for custom (months) = 60.00
Cost of NRE for COTS = $ 213192
Cost of NRE for custom = $ 6086280

Crossover point (number of modules) = 503
Crossover COTS module cost = $ 15472
Crossover custom-build module cost = $ 15472

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 3.00
----------------------------------------------
```

```
Time of NRE for COTS (months) = 18.00
Time of NRE for custom (months) = 90.00
Cost of NRE for COTS = $ 319788
Cost of NRE for custom = $ 9127920

Crossover point (number of modules) = 754
Crossover COTS module cost = $ 15472
Crossover custom-build module cost = $ 15478

##############################################
##############################################

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 47.74
Time of NRE for custom (months) = 48.60
Cost of NRE for COTS = $ 848191
Cost of NRE for custom = $ 4930457
COTS sigmoid adjustment = 7.37
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 296
Crossover COTS module cost = $ 20414
Crossover custom-build module cost = $ 20429

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 61.89
Time of NRE for custom (months) = 63.00
Cost of NRE for COTS = $ 1099506
Cost of NRE for custom = $ 6390444
COTS sigmoid adjustment = 7.37
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 384
Crossover COTS module cost = $ 20411
Crossover custom-build module cost = $ 20414
```

```
----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 88.41
Time of NRE for custom (months) = 90.00
Cost of NRE for COTS = $ 1570723
Cost of NRE for custom = $ 9127920
COTS sigmoid adjustment = 7.37
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 647
Crossover COTS module cost = $ 17476
Crossover custom-build module cost = $ 17481

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 132.62
Time of NRE for custom (months) = 135.00
Cost of NRE for COTS = $ 2356085
Cost of NRE for custom = $ 13690380
COTS sigmoid adjustment = 7.37
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 1183
Crossover COTS module cost = $ 14540
Crossover custom-build module cost = $ 14545

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 143.23
Time of NRE for custom (months) = 81.00
Cost of NRE for COTS = $ 2544572
Cost of NRE for custom = $ 8215428
COTS sigmoid adjustment = 22.10
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 411
Crossover COTS module cost = $ 23739
Crossover custom-build module cost = $ 23761

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 185.66
Time of NRE for custom (months) = 105.00
Cost of NRE for COTS = $ 3298519
Cost of NRE for custom = $ 10648740
COTS sigmoid adjustment = 22.10
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 629
Crossover COTS module cost = $ 20292
Crossover custom-build module cost = $ 20302

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 265.24
Time of NRE for custom (months) = 150.00
Cost of NRE for COTS = $ 4712170
Cost of NRE for custom = $ 15211200
COTS sigmoid adjustment = 22.10
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 1096
Crossover COTS module cost = $ 16848
Crossover custom-build module cost = $ 16851

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 397.85
Time of NRE for custom (months) = 225.00
Cost of NRE for COTS = $ 7068255
Cost of NRE for custom = $ 22815300
COTS sigmoid adjustment = 22.10
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 1644
Crossover COTS module cost = $ 16848
Crossover custom-build module cost = $ 16850

##############################################
##############################################

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 95.48
Time of NRE for custom (months) = 48.60
Cost of NRE for COTS = $ 1696381
Cost of NRE for custom = $ 4930457
COTS sigmoid adjustment (add 0.25 factor) = 14.74
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 234
Crossover COTS module cost = $ 24798
Crossover custom-build module cost = $ 24843

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 123.78
Time of NRE for custom (months) = 63.00
Cost of NRE for COTS = $ 2199013
Cost of NRE for custom = $ 6390444
COTS sigmoid adjustment (add 0.25 factor) = 14.74
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 304
Crossover COTS module cost = $ 24782
Crossover custom-build module cost = $ 24794

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 176.82
Time of NRE for custom (months) = 90.00
Cost of NRE for COTS = $ 3141447
Cost of NRE for custom = $ 9127920
COTS sigmoid adjustment (add 0.25 factor) = 14.74
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 512
Crossover COTS module cost = $ 21184
Crossover custom-build module cost = $ 21200

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 265.24
Time of NRE for custom (months) = 135.00
Cost of NRE for COTS = $ 4712170
Cost of NRE for custom = $ 13690380
COTS sigmoid adjustment (add 0.25 factor) = 14.74
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 768
Crossover COTS module cost = $ 21184
Crossover custom-build module cost = $ 21199

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.08
----------------------------------------------
```

```
Time of NRE for COTS (months) = 190.97
Time of NRE for custom (months) = 81.00
Cost of NRE for COTS = $ 3392763
Cost of NRE for custom = $ 8215428
COTS sigmoid adjustment (add 0.25 factor) = 29.47
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 350
Crossover COTS module cost = $ 27242
Crossover custom-build module cost = $ 27245

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 247.55
Time of NRE for custom (months) = 105.00
Cost of NRE for COTS = $ 4398026
Cost of NRE for custom = $ 10648740
COTS sigmoid adjustment (add 0.25 factor) = 29.47
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 535
Crossover COTS module cost = $ 23269
Crossover custom-build module cost = $ 23277

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 2.00
```

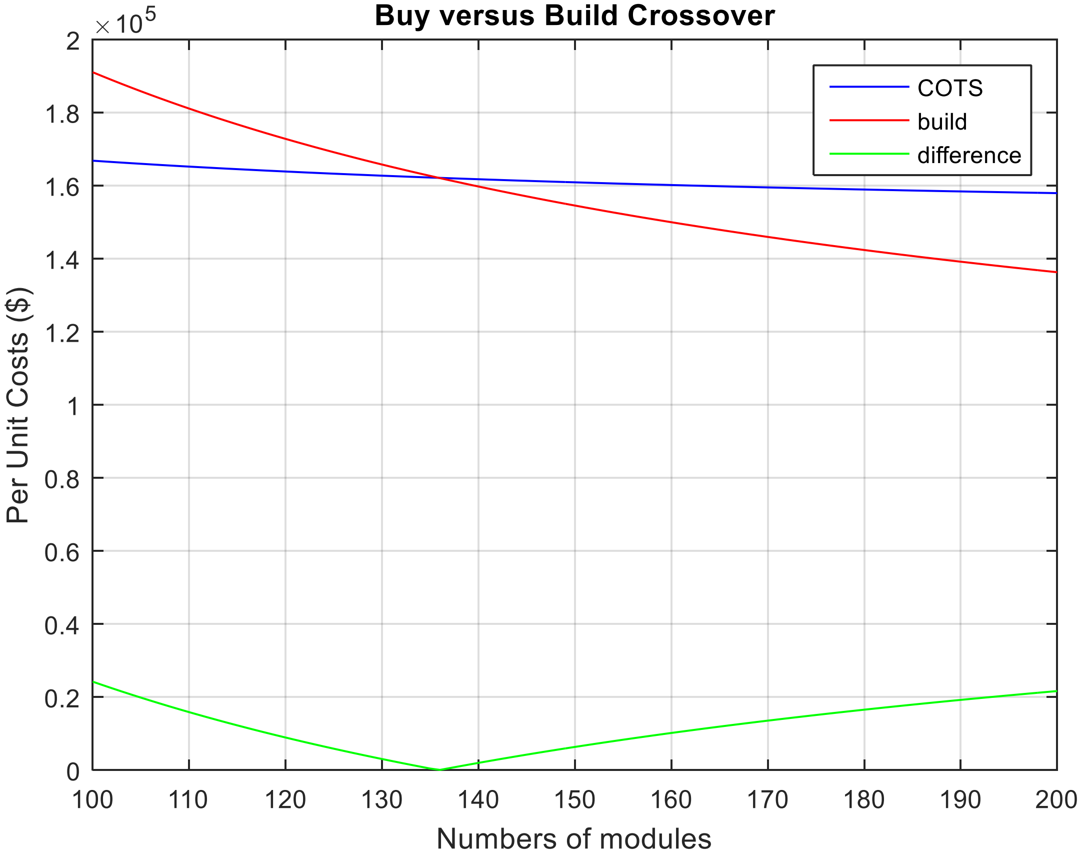
```
----------------------------------------------
Time of NRE for COTS (months) = 353.65
Time of NRE for custom (months) = 150.00
Cost of NRE for COTS = $ 6282894
Cost of NRE for custom = $ 15211200
COTS sigmoid adjustment (add 0.25 factor) = 29.47
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 764
Crossover COTS module cost = $ 23272
Crossover custom-build module cost = $ 23282

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 530.47
Time of NRE for custom (months) = 225.00
Cost of NRE for COTS = $ 9424340
Cost of NRE for custom = $ 22815300
COTS sigmoid adjustment (add 0.25 factor) = 29.47
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 1398
Crossover COTS module cost = $ 19289
Crossover custom-build module cost = $ 19292

##############################################
##############################################
```



Model Results: Space Qualified DAQ

August 27, 2018

```
>> [d, Q1, Q2, Q3, Q4] = init_param_Sp_DAQ();

Single-lot price for COTS module = $ 150000
Single-lot price for build components = $ 43900
COTS features range in Monte Carlo from 0.60 to 1.00
COTS market life ranges in Monte Carlo from 5.0 to 10.0
System market life ranges in Monte Carlo from 5.0 to 10.0
Premium to inventory COTS = 1.0

Number of systems planned = 500
Number of modules per system = 1

Average loaded hourly rates for staff:
Production staff = $ 80
Technicians = $ 80
Engineers & developers = $ 110
Management = $ 130
Administrative support = $ 60
Consulting & production staff = $ 80

COTS support and documentation for particular vendor:
Licensing support cost = $ 0
Customer support cost = $ 0
Team NRE for vendor support ranges from 0.00 to 0.40
Team NRE for customer support ranges from 0.00 to 0.40
Quality of doc & support ranges in Monte Carlo from 0.60 to 1.00
Customer interaction directly with module = 0.00

>> [St1, St2, St3, St4] = bvb_model(d, Q1, Q2, Q3, Q4);

     STAGE 1
-----------------
Time of NRE for COTS (months) = 18.00
Time of NRE for custom (months) = 60.00
Cost of NRE for COTS = $ 1775088
Cost of NRE for custom = $ 10946880

Lost opportunity cost = $ 4200000
Total system revenues over lifecycle = $ 25000000
Lost opportunity cost to total revenues = 0.1680

Crossover point (number of modules) = 135
Crossover COTS module cost = $ 162189
Crossover custom-build module cost = $ 162623
----------------------------------------------
```



```
##############################################
##############################################

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 19.44
Time of NRE for custom (months) = 64.80
Cost of NRE for COTS = $ 1917095
Cost of NRE for custom = $ 11822630

Crossover point (number of modules) = 146
Crossover COTS module cost = $ 162171
Crossover custom-build module cost = $ 162512

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 25.20
Time of NRE for custom (months) = 84.00
Cost of NRE for COTS = $ 2485123
Cost of NRE for custom = $ 15325632

Crossover point (number of modules) = 213
Crossover COTS module cost = $ 153207
Crossover custom-build module cost = $ 153486

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 36.00
Time of NRE for custom (months) = 120.00
Cost of NRE for COTS = $ 3550176
Cost of NRE for custom = $ 21893760
```

```
Crossover point (number of modules) = 305
Crossover COTS module cost = $ 153180
Crossover custom-build module cost = $ 153318

----------------------------------------------
     STAGE 2: Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 54.00
Time of NRE for custom (months) = 180.00
Cost of NRE for COTS = $ 5325264
Cost of NRE for custom = $ 32840640

Crossover point (number of modules) = 503
Crossover COTS module cost = $ 144627
Crossover custom-build module cost = $ 144630

##############################################
##############################################

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 35.58
Time of NRE for custom (months) = 97.20
Cost of NRE for COTS = $ 3509165
Cost of NRE for custom = $ 17733946
COTS sigmoid adjustment = 1.83
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 237
Crossover COTS module cost = $ 156347
Crossover custom-build module cost = $ 156362
```

```
----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 46.13
Time of NRE for custom (months) = 126.00
Cost of NRE for COTS = $ 4548918
Cost of NRE for custom = $ 22988448
COTS sigmoid adjustment = 1.83
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 307
Crossover COTS module cost = $ 156357
Crossover custom-build module cost = $ 156416

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 65.90
Time of NRE for custom (months) = 180.00
Cost of NRE for COTS = $ 6498454
Cost of NRE for custom = $ 32840640
COTS sigmoid adjustment = 1.83
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 438
Crossover COTS module cost = $ 156377
Crossover custom-build module cost = $ 156514

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.25
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 98.84
Time of NRE for custom (months) = 270.00
Cost of NRE for COTS = $ 9747682
Cost of NRE for custom = $ 49260960
COTS sigmoid adjustment = 1.83
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 722
Crossover COTS module cost = $ 147541
Crossover custom-build module cost = $ 147568

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 106.75
Time of NRE for custom (months) = 162.00
Cost of NRE for COTS = $ 10527496
Cost of NRE for custom = $ 29556576
COTS sigmoid adjustment = 5.49
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 317
Crossover COTS module cost = $ 174750
Crossover custom-build module cost = $ 174773

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 138.38
Time of NRE for custom (months) = 210.00
Cost of NRE for COTS = $ 13646754
Cost of NRE for custom = $ 38314080
COTS sigmoid adjustment = 5.49
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 411
Crossover COTS module cost = $ 174744
Crossover custom-build module cost = $ 174757

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 197.69
Time of NRE for custom (months) = 300.00
Cost of NRE for COTS = $ 19495363
Cost of NRE for custom = $ 54734400
COTS sigmoid adjustment = 5.49
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 644
Crossover COTS module cost = $ 164312
Crossover custom-build module cost = $ 164331

----------------------------------------------
     STAGE 3: Sigmoid adjustment = 0.75
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 296.53
Time of NRE for custom (months) = 450.00
Cost of NRE for COTS = $ 29243045
Cost of NRE for custom = $ 82101600
COTS sigmoid adjustment = 5.49
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 1024
Crossover COTS module cost = $ 155098
Crossover custom-build module cost = $ 155127

##############################################
##############################################

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 71.17
Time of NRE for custom (months) = 97.20
Cost of NRE for COTS = $ 7018331
Cost of NRE for custom = $ 17733946
COTS sigmoid adjustment (add 0.25 factor) = 3.66
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 158
Crossover COTS module cost = $ 193460
Crossover custom-build module cost = $ 193775

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 92.26
Time of NRE for custom (months) = 126.00
Cost of NRE for COTS = $ 9097836
Cost of NRE for custom = $ 22988448
COTS sigmoid adjustment (add 0.25 factor) = 3.66
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 231
Crossover COTS module cost = $ 180925
Crossover custom-build module cost = $ 181052

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 131.79
Time of NRE for custom (months) = 180.00
Cost of NRE for COTS = $ 12996909
Cost of NRE for custom = $ 32840640
COTS sigmoid adjustment (add 0.25 factor) = 3.66
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 330
Crossover COTS module cost = $ 180925
Crossover custom-build module cost = $ 181052

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.25
              Rayleigh adjustment = 3.00
----------------------------------------------
```

```
Time of NRE for COTS (months) = 197.69
Time of NRE for custom (months) = 270.00
Cost of NRE for COTS = $ 19495363
Cost of NRE for custom = $ 49260960
COTS sigmoid adjustment (add 0.25 factor) = 3.66
Custom sigmoid adjustment = 1.50

Crossover point (number of modules) = 544
Crossover COTS module cost = $ 169877
Crossover custom-build module cost = $ 169893

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.08
----------------------------------------------
Time of NRE for COTS (months) = 142.34
Time of NRE for custom (months) = 162.00
Cost of NRE for COTS = $ 14036661
Cost of NRE for custom = $ 29556576
COTS sigmoid adjustment (add 0.25 factor) = 7.32
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 258
Crossover COTS module cost = $ 195946
Crossover custom-build module cost = $ 196095

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 1.40
----------------------------------------------
Time of NRE for COTS (months) = 184.51
Time of NRE for custom (months) = 210.00
Cost of NRE for COTS = $ 18195672
Cost of NRE for custom = $ 38314080
COTS sigmoid adjustment (add 0.25 factor) = 7.32
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 335
Crossover COTS module cost = $ 195855
Crossover custom-build module cost = $ 195905

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 2.00
----------------------------------------------
Time of NRE for COTS (months) = 263.59
Time of NRE for custom (months) = 300.00
Cost of NRE for COTS = $ 25993817
Cost of NRE for custom = $ 54734400
COTS sigmoid adjustment (add 0.25 factor) = 7.32
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 525
Crossover COTS module cost = $ 183552
Crossover custom-build module cost = $ 183596

----------------------------------------------
     STAGE 4: COTS deficiencies = 0.25
              Sigmoid adjustment = 0.75
              Rayleigh adjustment = 3.00
----------------------------------------------
Time of NRE for COTS (months) = 395.38
Time of NRE for custom (months) = 450.00
Cost of NRE for COTS = $ 38990726
Cost of NRE for custom = $ 82101600
COTS sigmoid adjustment (add 0.25 factor) = 7.32
Custom sigmoid adjustment = 2.50

Crossover point (number of modules) = 788
Crossover COTS module cost = $ 183521
Crossover custom-build module cost = $ 183530

##############################################
##############################################
```

Matlab Source Code

Bvb_Model, rev2

August 26, 2018

```
function [St1, St2, St3, St4] = bvb_model(d, Q1, Q2, Q3, Q4)
% -------------------------------------------------------------------------
%   This is bvb_model. By Kim R. Fowler.
%   Created on 25 August 2018.
%   Modified from bvb25 that was created on 30 July 2018.
%
%   This model only targets embedded systems and it only refers to
%   one type of module for a particular system. The module under
%   consideration may have multiple copies within a single system.
%
%   This code includes insufficient COTS specifications, insufficient
%   vendor support, finite vendor consulting support, or lifecycle that
%   increase the COTS NRE and thus lower crossover point. It includes lower
%   expertise and resources that can increase the NRE of custom design.
%
%   Input is from data matrices with initialization values from the script
%   file, "init_param".
% -------------------------------------------------------------------------
clf                         % Clear figures from previous simulation run
%clc                         % Clear commands from command window

% /////////////////////////////////////////////////////////////////////////
% -------------------------------------------------------------------------
%   Setup of all variables for either COTS or custom-build
% -------------------------------------------------------------------------
% /////////////////////////////////////////////////////////////////////////

COTS_price = d(1,1);        % Single-lot price for COTS module
Comp_price = d(2,1);        % Single-lot price for custom-build components

% -------------------------------------------------------------------------
%   Discount schedules at the following quantity breaks in N
% -------------------------------------------------------------------------
N = Q1(1,:);
N_indx = size(N);
COTS_discount = Q1(2,:);
Comp_discount = Q1(3,:);

% -------------------------------------------------------------------------
%   Set premium for inventory for custom design or for COTS or to buy ROTS
%   if system life is greater than COTS module’s life. Premium can vary
%   between 1 and 5x.
% -------------------------------------------------------------------------
inven_prem1 = d(9,1);   % Premium when system life < COTS life, usually = 1
inven_prem2 = d(10,1);  % Premium when system life < COTS life, vary 1 to 5

% -------------------------------------------------------------------------
%   Some system numbers planned
% -------------------------------------------------------------------------
N_sys = d(11,1);               % Number of systems planned over lifetime
mod_sys = d(12,1);             % Number of modules per system, default=1
mod_total = N_sys * mod_sys;   % Total modules planned for product lifetime

Sys_price = d(13,1);                    % Planned unit price of a system
Sys_cost = d(14,1);                     % Planned unit cost of a system
Sys_revenue = Sys_price - Sys_cost;     % Planned revenue per system

Early_sales_rate = 2;           % Average early sales per month of systems

% -------------------------------------------------------------------------
%   Labor costs in $/hr
% -------------------------------------------------------------------------
Prod_labor = d(15,1);   % Average loaded salary of production staff
Tech_labor = d(16,1);   % Average loaded salary of technicians
Eng_labor = d(17,1);    % Average loaded salary of engineers & developers
Manag_labor = d(18,1);  % Average loaded salary of management
Admin_labor = d(19,1);  % Average loaded salary of admin & support staff
Consult_labor = d(20,1);% Average loaded salary of consulting & prod. staff

% COGS - vector of staff costs for production
COGS_labor(1,1) = Prod_labor;       % col.1 = production staff
COGS_labor(1,2) = Manag_labor;      % col.2 = management
COGS_labor(1,3) = Admin_labor;      % col.3 = admin/support staff

% NRE - vector of staff costs for development
Dev_labor(1,1) = Eng_labor;         % col.1 = H/W engineer
Dev_labor(1,2) = Eng_labor;         % col.2 = S/W engineer
Dev_labor(1,3) = Eng_labor;         % col.3 = Mechanical engineer
Dev_labor(1,4) = Eng_labor;         % col.4 = specialty engineer
Dev_labor(1,5) = Tech_labor;        % col.5 = technician
Dev_labor(1,6) = Manag_labor;       % col.6 = management
Dev_labor(1,7) = Admin_labor;       % col.7 = admin/support staff
Dev_labor(1,8) = Consult_labor;     % col.8 = production & misc consulting

% -------------------------------------------------------------------------
%   Variable initialization for COGS calculations per unit COTS module.
% -------------------------------------------------------------------------
% Maxtrix FTE for COTS in production
COTS_COGS_FTE = Q2(:,:,1);

COTS_fab_t = d(21,1);       % Time (hrs) to fabricate and assemble a module
COTS_test_t = d(22,1);      % Time (hrs) to test a module
COTS_insert_t = d(23,1);    % Time (hrs) in production to insert module
COTS_inventory = d(24,1);   % Time to inventory a module
COTS_sales = d(25,1);       % Cost ($) of sales and marketing per module
COTS_techsup = d(26,1);     % Cost ($) of technical support per module
COTS_mat_en = d(27,1);      % Cost ($) for assets, materials, energy/unit

% -------------------------------------------------------------------------
%   Variable initialization for COGS calculations per unit custom module.
% -------------------------------------------------------------------------
% Maxtrix FTE for custom-build in production
Cust_COGS_FTE = Q2(:,:,2);
```

```
Cust_fab_t = d(28,1);       % Time (hrs) to fabricate and assemble a module
Cust_test_t = d(29,1);      % Time (hrs) to test a module
Cust_insert_t = d(30,1);    % Time (hrs) in production to insert module
Cust_inventory = d(31,1);   % Time to inventory a module
Cust_sales = d(32,1);       % Cost ($) of sales and marketing per module
Cust_techsup = d(33,1);     % Cost ($) of technical support per module
Cust_mat_en = d(34,1);      % Cost ($) for assets, materials, energy/unit

% -------------------------------------------------------------------------
%   Variable initialization for COTS NRE calculations for development of
%   module incorporation in the system.
% -------------------------------------------------------------------------
COTS_tool_cost = d(35,1);   % One time cost ($) to tool up for COTS
production
COTS_tool_t = d(36,1);      % Time (hrs) for tooling up for COTS production
COTS_deliv_t = d(37,1);     % Delivery time (days) for first COTS modules
COTS_cert_cost = d(38,1);   % Cost ($) for certification of COTS module

COTS_team_sup = d(41,1);        % Yearly cost ($) for vendor supporting
                                % development team
Final_customer_sup = d(42,1);   % Yearly cost ($) for vendor supporting
                                % final customer

COTS_interact = d(49,1);    % Does customer interact directly with module?
                            % no = 0, yes = 1
Final_customer_sup_cost = Final_customer_sup * COTS_interact;

% -------------------------------------------------------------------------
%   COTS NRE calculations for effort and time. The full time equivalent for
%   staffing is a 8 x 4 matrix; the rows represent specific types of
%   staff members; the columns represent development phases.
%
%   Row 1 = hardware engineers
%   Row 2 = software engineers and developers
%   Row 3 = mechanical engineers
%   Row 4 = specialty engineers (optics, chemical, materials, OR)
%   Row 5 = technicians
%   Row 6 = management
%   Row 7 = admin support staff
%   Row 8 = production staff and other consultants
%
%   Column 1 = Concept & Preliminary phase
%   Column 2 = Critical Design phase
%   Column 3 = Test & Integration phase
%   Column 4 = Compliance & Production Preparation phase
% -------------------------------------------------------------------------
COTS_NRE_FTE = Q3(:,:,1);
COTS_Calendar = Q4(:,:,1);  %NRE calendar time per phase (months)

% -------------------------------------------------------------------------
%   Variable initialization for NRE calculations for development of a
%   custom-build module in the system.
% -------------------------------------------------------------------------
Cust_tool_cost = d(50,1);   % One time cost ($) to tool up for production
Cust_tool_t = d(51,1);      % Time (hrs) for tooling up for production
```

```
Cust_deliv_t = d(52,1);      % Delivery time (days) for first component lots
Cust_cert_cost = d(53,1);    % Cost($) to certify custom-built module

% --------------------------------------------------------------------------
%   NRE calculations for effort and time. The full time equivalent for
%   staffing is a 8 x 4 matrix; the rows represent specific types of
%   staff members; the columns represent development phases.
%
%   Row 1 = hardware engineers
%   Row 2 = software engineers and developers
%   Row 3 = mechanical engineers
%   Row 4 = specialty engineers (optics, chemical, materials, OR)
%   Row 5 = technicians
%   Row 6 = management
%   Row 7 = admin support staff
%   Row 8 = production staff and other consultants
%
%   Column 1 = Concept & Preliminary phase
%   Column 2 = Critical Design phase
%   Column 3 = Test & Integration phase
%   Column 4 = Compliance & Production Preparation phase
% --------------------------------------------------------------------------
Cust_NRE_FTE = Q3(:,:,2);
Cust_Calendar = Q4(:,:,2);      %NRE calendar time per phase (months)

% --------------------------------------------------------------------------
%   Set up COGS to perform deterministic calculations.
% --------------------------------------------------------------------------

% --------------------------------------------------------------------------
%   COGS calculations per unit COTS module. This simulation assumes
%   no learning curve to shorten time in production.
% --------------------------------------------------------------------------
% Vector time and total time for COTS in production
COTS_COGS_t = [COTS_fab_t; COTS_test_t; COTS_insert_t; COTS_inventory];
COTS_COGS_tot_t = sum(COTS_COGS_t);      % Total time (hrs) per module
% Total cost ($) of labor, material, and energy per module
COTS_COGS_tot_cost = (COGS_labor*(COTS_COGS_FTE*COTS_COGS_t)) + COTS_sales +
COTS_techsup + COTS_mat_en;

% --------------------------------------------------------------------------
%   COGS calculations per unit custom-built module. This simulation assumes
%   no learning curve to shorten time in production.
% --------------------------------------------------------------------------
% Total time and total time for custom-build in production
Cust_COGS_t = [Cust_fab_t; Cust_test_t; Cust_insert_t; Cust_inventory];
Cust_COGS_tot_t = sum(Cust_COGS_t);      % Total time (hrs) per module
% Total cost ($) of labor, material, and energy per module
Cust_COGS_tot_cost =(COGS_labor*(Cust_COGS_FTE* Cust_COGS_t)) + Cust_sales +
Cust_techsup + Cust_mat_en;

% //////////////////////////////////////////////////////////////////////////
% --------------------------------------------------------------------------
%   STAGE 1
% --------------------------------------------------------------------------
% //////////////////////////////////////////////////////////////////////////
% -------------------------------------------------------------------------
%   COTS NRE calculations for development of module incorporation in the
%   system. The NRE calculations then results in cost per unit COTS module.
%   These calculations assume exact estimation of effort, FTE, and
%   calendar time in this stage 1.
% -------------------------------------------------------------------------
COTS_person_month = COTS_NRE_FTE * COTS_Calendar;
COTS_Tot_effort = sum(COTS_person_month);
    %Total NRE labor cost for project ($), assume 168 hrs/month
COTS_Tot_labor = Dev_labor * COTS_person_month * 168;
COTS_Tot_time = sum(COTS_Calendar); %Total calendar time for NRE (months)

% Calculate total COTS costs from NRE, tooling, and compliance/certification
COTS_NRE_cost = COTS_Tot_labor + COTS_tool_cost + COTS_cert_cost;

% Calculate prices for COTS modules with discounts and inventory premium
COTS_mod_p = COTS_discount * COTS_price * inven_prem2;

% Set up COTS cost vector from total COGS cost
COTS_COGS_cost = ones(1,N_indx(1,2)) * COTS_COGS_tot_cost;

% Calculate vector of COTS unit prices at the cost breakpoints
COTS_unit_cost = COTS_COGS_cost + COTS_mod_p + (COTS_NRE_cost ./N);

% -------------------------------------------------------------------------
%   NRE calculations for development of a custom-build module in the
%   system. The NRE calculations then results in cost per unit module.
% -------------------------------------------------------------------------
Cust_person_month = Cust_NRE_FTE * Cust_Calendar;
Cust_Tot_effort = sum(Cust_person_month);
%Total NRE labor cost for project ($), assume 168 hrs/month
Cust_Tot_labor = Dev_labor * Cust_person_month * 168;
Cust_Tot_time = sum(Cust_Calendar); %Total calendar time for NRE (months)

% Calculate build costs from NRE, tooling, and compliance/certification
Cust_NRE_cost = Cust_Tot_labor + Cust_tool_cost + Cust_cert_cost;

% Calculate prices for custom-built modules with discounts and inventory
Cust_mod_p = Comp_discount * Comp_price * inven_prem1;

% Set up build cost vector from total COGS cost
Cust_COGS_cost = ones(1,N_indx(1,2)) * Cust_COGS_tot_cost;

% Calculate the lost opportunity costs
lost_opp_cost = (Cust_Tot_time - COTS_Tot_time) * Early_sales_rate *
Sys_revenue;

fprintf('\n');fprintf('     STAGE 1     \n');fprintf('-----------------\n')
fprintf('Time of NRE for COTS (months) = %3.2f\n',COTS_Tot_time)
fprintf('Time of NRE for custom (months) = %3.2f\n',Cust_Tot_time)
fprintf('Cost of NRE for COTS = $ %.f\n',COTS_NRE_cost)
fprintf('Cost of NRE for custom = $ %.f\n',Cust_NRE_cost)

Sys_Tot_rev = Sys_revenue * N_sys;
Per_lost_opp = lost_opp_cost/Sys_Tot_rev;
```

```
fprintf('\n')
fprintf('Lost opportunity cost = $ %.f\n',lost_opp_cost)
fprintf('Total system revenues over lifecycle = $ %.f\n',Sys_Tot_rev)
fprintf('Lost opportunity cost to total revenues = %.4f\n',Per_lost_opp)

% Calculate vector of COTS unit prices at the cost breakpoints
Cust_unit_cost = Cust_COGS_cost + Cust_mod_p + (Cust_NRE_cost ./N);

% Calculate the initial ratio of custom effort to COTS effort for later use
C_fac = (Cust_Tot_effort/COTS_Tot_effort) + 1;

% -------------------------------------------------------------------------
%   Find the crossover point in unit costs between COTS and custom-build
% -------------------------------------------------------------------------
find_crossover = Cust_unit_cost - COTS_unit_cost;   % First order crossover
find_cross1 = zeros(1,N_indx(1,2));
find_cross2 = zeros(1,N_indx(1,2));
find_cross3 = zeros(1,N_indx(1,2));
% Find interval where crossover occurs, depends on whether custom-build
% is decreasing or increasing
cross_point = diff(sign(find_crossover));
find_cross1(1,1:(N_indx(1,2)-1)) = cross_point;
find_cross2(1,2:N_indx(1,2)) = cross_point; % Interval end if decreasing
cross_point(1:1) = [];
find_cross3(1,1:(N_indx(1,2)-2)) = cross_point;% Interval end if increasing
check_cross = sum(find_cross1);             % Does crossover exist?

if check_cross == 0                         % =0, then no crossover
    fprintf('No crossover point found\n')
    crossover_i = [0 0];
    iCOTSmodp = max(COTS_mod_p);
    iCOTSCcost = max(COTS_COGS_cost);
    iCustmodp = max(Cust_mod_p);
    iCustCcost = max(Cust_COGS_cost);
else                                        % Find crossover indices
    crossover_i = [0 0];
    select_i1 = N.*(find_cross1/check_cross);
    select_i2 = N.*(find_cross2/check_cross);
    select_i3 = N.*(find_cross3/check_cross);
    if check_cross < 0              % - means custom-build is decreasing
        crossover_i(1,1) = max(select_i1);
        crossover_i(1,2) = max(select_i2);
        % Calculate the module and COGS costs for this decreasing interval
        iCOTSmodp = max(COTS_mod_p.*(select_i1/crossover_i(1,1)));
        iCOTSCcost = max(COTS_COGS_cost.*(select_i1/crossover_i(1,1)));
        iCustmodp = max(Cust_mod_p.*(select_i1/crossover_i(1,1)));
        iCustCcost = max(Cust_COGS_cost.*(select_i1/crossover_i(1,1)));
    else                            % + means custom-build is increasing
        crossover_i(1,1) = max(select_i3);
        crossover_i(1,2) = max(select_i1);
        % Calculate the module and COGS costs for this increasing interval
        iCOTSmodp = max(COTS_mod_p.*(select_i3/crossover_i(1,1)));
        iCOTSCcost = max(COTS_COGS_cost.*(select_i3/crossover_i(1,1)));
        iCustmodp = max(Cust_mod_p.*(select_i3/crossover_i(1,1)));
        iCustCcost = max(Cust_COGS_cost.*(select_i3/crossover_i(1,1)));
    end
```

```
end

% -------------------------------------------------------------------------
%   Calcualte the crossover point in the selected interval
% -------------------------------------------------------------------------
M = crossover_i(1,1):1:crossover_i(1,2);    % Interval with the crossover
% Form vectors of the module and COGS costs for both COTS and custom-build
iCOTS_mod_p = ones(size(M))*iCOTSmodp;
iCOTS_COGS_cost = ones(size(M))*iCOTSCcost;
iCust_mod_p = ones(size(M))*iCustmodp;
iCust_COGS_cost = ones(size(M))*iCustCcost;
% Calculate the unit costs and then find the difference between
% COTS and custom-build
iCOTS_unit_cost = iCOTS_COGS_cost + iCOTS_mod_p + (COTS_NRE_cost./M);
iCust_unit_cost = iCust_COGS_cost + iCust_mod_p + (Cust_NRE_cost./M);

idiff_unit_cost = iCust_unit_cost - iCOTS_unit_cost;
cost_diff = abs(idiff_unit_cost);
icross_point = zeros(size(M));
icross_point(1:length(M)-1) = diff(sign(idiff_unit_cost));
icross_val = sum(icross_point);
break_point = sum(M.*(icross_point/icross_val));
COTS_cross_mod_cost = sum(iCOTS_unit_cost.*(icross_point/icross_val));
Cust_cross_mod_cost = sum(iCust_unit_cost.*(icross_point/icross_val));

% -------------------------------------------------------------------------
%   Print crossover point and module costs
% -------------------------------------------------------------------------
fprintf('\n')
fprintf('Crossover point (number of modules) = %.f\n',break_point)
fprintf('Crossover COTS module cost = $ %.f\n',COTS_cross_mod_cost)
fprintf('Crossover custom-build module cost = $ %.f\n',Cust_cross_mod_cost)
fprintf('----------------------------------------------\n')

plot(M,iCOTS_unit_cost,'b-')
hold on
plot(M,iCust_unit_cost,'r-')
plot(M,cost_diff,'g-')
ylabel('Per Unit Costs ($)')
xlabel('Numbers of modules')
title('Buy versus Build Crossover')
legend('COTS','build','difference')
grid
fprintf('\n')
fprintf('Press spacebar\n')
fprintf('\n')
pause
hold off
fprintf('##############################################\n')
fprintf('##############################################\n')
fprintf('\n')

St1 = zeros(3,1);           % initialize output vector for Stage 1
St1(1) = break_point;       % crossover in Stage 1
St1(2) = mod_total;         % number of modules planned over production run
St1(3) = (COTS_cross_mod_cost + Cust_cross_mod_cost)/2; % ave mod unit cost
```

```
% /////////////////////////////////////////////////////////////////////////
% -------------------------------------------------------------------------
%   STAGE 2
% -------------------------------------------------------------------------
% /////////////////////////////////////////////////////////////////////////
% -------------------------------------------------------------------------
%   COTS NRE calculations for development of module incorporation in the
%   system. The NRE calculations then results in cost per unit COTS module.
%   These calculations assume exact estimation of effort, FTE, but multiply
%   the calendar time according to four (4) different Rayleigh medians to
%   predict various Overruns.
% -------------------------------------------------------------------------
Rayl_adj = [1.08 1.4 2.0 3.0];  % four different Rayleigh medians
St2 = zeros(8,1);               % initialize output vector for Stage 2

for i = 1:4
    COTS_Cal_adj = COTS_Calendar * Rayl_adj(i);
COTS_person_month = COTS_NRE_FTE * COTS_Cal_adj;
COTS_Tot_effort = sum(COTS_person_month);
    %Total NRE labor cost for project ($), assume 168 hrs/month
COTS_Tot_labor = Dev_labor * COTS_person_month * 168;
COTS_Tot_time = sum(COTS_Cal_adj);  %Total calendar time for NRE (months)

% Calculate total COTS costs from NRE, tooling, and compliance/certification
COTS_NRE_cost = COTS_Tot_labor + COTS_tool_cost + COTS_cert_cost;

% Calculate prices for COTS modules with discounts and inventory premium
COTS_mod_p = COTS_discount * COTS_price * inven_prem2;

% Set up COTS cost vector from total COGS cost
COTS_COGS_cost = ones(1,N_indx(1,2)) * COTS_COGS_tot_cost;

% Calculate vector of COTS unit prices at the cost breakpoints
COTS_unit_cost = COTS_COGS_cost + COTS_mod_p + (COTS_NRE_cost ./N);
% -------------------------------------------------------------------------
%   NRE calculations for development of a custom-build module in the
%   system. The NRE calculations then results in cost per unit module.
% -------------------------------------------------------------------------
    Cust_Cal_adj = Cust_Calendar * Rayl_adj(i);
Cust_person_month = Cust_NRE_FTE * Cust_Cal_adj;
Cust_Tot_effort = sum(Cust_person_month);
%Total NRE labor cost for project ($), assume 168 hrs/month
Cust_Tot_labor = Dev_labor * Cust_person_month * 168;
Cust_Tot_time = sum(Cust_Cal_adj); %Total calendar time for NRE (months)

% Calculate build costs from NRE, tooling, and compliance/certification
Cust_NRE_cost = Cust_Tot_labor + Cust_tool_cost + Cust_cert_cost;

% Calculate prices for custom-built modules with discounts and inventory
Cust_mod_p = Comp_discount * Comp_price * inven_prem1;

% Set up build cost vector from total COGS cost
Cust_COGS_cost = ones(1,N_indx(1,2)) * Cust_COGS_tot_cost;
```

```
% Calculate the lost opportunity costs
lost_opp_cost = (Cust_Tot_time - COTS_Tot_time) * Early_sales_rate *
Sys_revenue;

fprintf('----------------------------------------------\n')
fprintf('     STAGE 2: Rayleigh adjustment = %1.2f\n',Rayl_adj(i))
fprintf('----------------------------------------------\n')
fprintf('Time of NRE for COTS (months) = %3.2f\n',COTS_Tot_time)
fprintf('Time of NRE for custom (months) = %3.2f\n',Cust_Tot_time)
fprintf('Cost of NRE for COTS = $ %.f\n',COTS_NRE_cost)
fprintf('Cost of NRE for custom = $ %.f\n',Cust_NRE_cost)

% Calculate vector of COTS unit prices at the cost breakpoints
Cust_unit_cost = Cust_COGS_cost + Cust_mod_p + (Cust_NRE_cost ./N);

% --------------------------------------------------------------------------
%   Find the crossover point in unit costs between COTS and custom-build
% --------------------------------------------------------------------------
find_crossover = Cust_unit_cost - COTS_unit_cost;   % First order crossover
find_cross1 = zeros(1,N_indx(1,2));
find_cross2 = zeros(1,N_indx(1,2));
find_cross3 = zeros(1,N_indx(1,2));
% Find interval where crossover occurs, depends on whether custom-build
% is decreasing or increasing
cross_point = diff(sign(find_crossover));
find_cross1(1,1:(N_indx(1,2)-1)) = cross_point;
find_cross2(1,2:N_indx(1,2)) = cross_point; % Interval end if decreasing
cross_point(1:1) = [];
find_cross3(1,1:(N_indx(1,2)-2)) = cross_point;% Interval end if increasing
check_cross = sum(find_cross1);             % Does crossover exist?

if check_cross == 0                         % =0, then no crossover
    fprintf('No crossover point found\n')
    crossover_i = [0 0];
    iCOTSmodp = max(COTS_mod_p);
    iCOTSCcost = max(COTS_COGS_cost);
    iCustmodp = max(Cust_mod_p);
    iCustCcost = max(Cust_COGS_cost);
else                                        % Find crossover indices
    crossover_i = [0 0];
    select_i1 = N.*(find_cross1/check_cross);
    select_i2 = N.*(find_cross2/check_cross);
    select_i3 = N.*(find_cross3/check_cross);
    if check_cross < 0              % - means custom-build is decreasing
        crossover_i(1,1) = max(select_i1);
        crossover_i(1,2) = max(select_i2);
        % Calculate the module and COGS costs for this decreasing interval
        iCOTSmodp = max(COTS_mod_p.*(select_i1/crossover_i(1,1)));
        iCOTSCcost = max(COTS_COGS_cost.*(select_i1/crossover_i(1,1)));
        iCustmodp = max(Cust_mod_p.*(select_i1/crossover_i(1,1)));
        iCustCcost = max(Cust_COGS_cost.*(select_i1/crossover_i(1,1)));
    else                            % + means custom-build is increasing
        crossover_i(1,1) = max(select_i3);
        crossover_i(1,2) = max(select_i1);
        % Calculate the module and COGS costs for this increasing interval
        iCOTSmodp = max(COTS_mod_p.*(select_i3/crossover_i(1,1)));
        iCOTSCcost = max(COTS_COGS_cost.*(select_i3/crossover_i(1,1)));
        iCustmodp = max(Cust_mod_p.*(select_i3/crossover_i(1,1)));
        iCustCcost = max(Cust_COGS_cost.*(select_i3/crossover_i(1,1)));
    end
end

% -------------------------------------------------------------------------
%   Calcualte the crossover point in the selected interval
% -------------------------------------------------------------------------
M = crossover_i(1,1):1:crossover_i(1,2);    % Interval with the crossover
% Form vectors of the module and COGS costs for both COTS and custom-build
iCOTS_mod_p = ones(size(M))*iCOTSmodp;
iCOTS_COGS_cost = ones(size(M))*iCOTSCcost;
iCust_mod_p = ones(size(M))*iCustmodp;
iCust_COGS_cost = ones(size(M))*iCustCcost;
% Calculate the unit costs and then find the difference between
% COTS and custom-build
iCOTS_unit_cost = iCOTS_COGS_cost + iCOTS_mod_p + (COTS_NRE_cost./M);
iCust_unit_cost = iCust_COGS_cost + iCust_mod_p + (Cust_NRE_cost./M);

idiff_unit_cost = iCust_unit_cost - iCOTS_unit_cost;
cost_diff = abs(idiff_unit_cost);
icross_point = zeros(size(M));
icross_point(1:length(M)-1) = diff(sign(idiff_unit_cost));
icross_val = sum(icross_point);
break_point = sum(M.*(icross_point/icross_val));
COTS_cross_mod_cost = sum(iCOTS_unit_cost.*(icross_point/icross_val));
Cust_cross_mod_cost = sum(iCust_unit_cost.*(icross_point/icross_val));

% -------------------------------------------------------------------------
%   Print crossover point and module costs
% -------------------------------------------------------------------------
fprintf('\n')
fprintf('Crossover point (number of modules) = %.f\n',break_point)
fprintf('Crossover COTS module cost = $ %.f\n',COTS_cross_mod_cost)
fprintf('Crossover custom-build module cost = $ %.f\n',Cust_cross_mod_cost)
fprintf('\n')

indx = (2 * i) - 1;         % set index into vector for output
St2(indx) = break_point;    % store a crossover from Stage 2 in the vector
end
fprintf('##############################################\n')
fprintf('##############################################\n')
fprintf('\n')

% /////////////////////////////////////////////////////////////////////////
% -------------------------------------------------------------------------
%   STAGE 3
% -------------------------------------------------------------------------
% /////////////////////////////////////////////////////////////////////////
% -------------------------------------------------------------------------
%   COTS NRE calculations for development of module incorporation in the
%   system. The NRE calculations then results in cost per unit COTS module.
%   These calculations assume exact estimation of effort, FTE, but multiply
%   the calendar time according to four (4) different Rayleigh medians to
%   predict various Overruns and according to for two (2) different
%   sigmoid adjustments.
% -------------------------------------------------------------------------
COTS_sig = [(C_fac * 0.25) (C_fac * 0.75)];
Cust_sig = [1.5 2.5];
St3 = zeros(12,1);               % initialize output vector for Stage 3

for j = 1:2
for i = 1:4
    COTS_Cal_adj = COTS_Calendar * Rayl_adj(i) * COTS_sig(j);
COTS_person_month = COTS_NRE_FTE * COTS_Cal_adj;
COTS_Tot_effort = sum(COTS_person_month);
    %Total NRE labor cost for project ($), assume 168 hrs/month
COTS_Tot_labor = Dev_labor * COTS_person_month * 168;
COTS_Tot_time = sum(COTS_Cal_adj);  %Total calendar time for NRE (months)

% Calculate total COTS costs from NRE, tooling, and compliance/certification
COTS_NRE_cost = COTS_Tot_labor + COTS_tool_cost + COTS_cert_cost;

% Calculate prices for COTS modules with discounts and inventory premium
COTS_mod_p = COTS_discount * COTS_price * inven_prem2;

% Set up COTS cost vector from total COGS cost
COTS_COGS_cost = ones(1,N_indx(1,2)) * COTS_COGS_tot_cost;

% Calculate vector of COTS unit prices at the cost breakpoints
COTS_unit_cost = COTS_COGS_cost + COTS_mod_p + (COTS_NRE_cost ./N);
% -------------------------------------------------------------------------
%   NRE calculations for development of a custom-build module in the
%   system. The NRE calculations then results in cost per unit module.
% -------------------------------------------------------------------------
    Cust_Cal_adj = Cust_Calendar * Rayl_adj(i) * Cust_sig(j);
Cust_person_month = Cust_NRE_FTE * Cust_Cal_adj;
Cust_Tot_effort = sum(Cust_person_month);
%Total NRE labor cost for project ($), assume 168 hrs/month
Cust_Tot_labor = Dev_labor * Cust_person_month * 168;
Cust_Tot_time = sum(Cust_Cal_adj); %Total calendar time for NRE (months)

% Calculate build costs from NRE, tooling, and compliance/certification
Cust_NRE_cost = Cust_Tot_labor + Cust_tool_cost + Cust_cert_cost;

% Calculate prices for custom-built modules with discounts and inventory
Cust_mod_p = Comp_discount * Comp_price * inven_prem1;

% Set up build cost vector from total COGS cost
Cust_COGS_cost = ones(1,N_indx(1,2)) * Cust_COGS_tot_cost;

% Calculate the lost opportunity costs
lost_opp_cost = (Cust_Tot_time - COTS_Tot_time) * Early_sales_rate *
Sys_revenue;

fprintf('----------------------------------------------\n')
if j == 1
    fprintf('     STAGE 3: Sigmoid adjustment = 0.25\n')
else
    fprintf('     STAGE 3: Sigmoid adjustment = 0.75\n')
```

```
end
fprintf('              Rayleigh adjustment = %1.2f\n',Rayl_adj(i))
fprintf('----------------------------------------------\n')
fprintf('Time of NRE for COTS (months) = %3.2f\n',COTS_Tot_time)
fprintf('Time of NRE for custom (months) = %3.2f\n',Cust_Tot_time)
fprintf('Cost of NRE for COTS = $ %.f\n',COTS_NRE_cost)
fprintf('Cost of NRE for custom = $ %.f\n',Cust_NRE_cost)
fprintf('COTS sigmoid adjustment = %3.2f\n',COTS_sig(j))
fprintf('Custom sigmoid adjustment = %3.2f\n',Cust_sig(j))

% Calculate vector of COTS unit prices at the cost breakpoints
Cust_unit_cost = Cust_COGS_cost + Cust_mod_p + (Cust_NRE_cost ./N);

% -------------------------------------------------------------------------
%   Find the crossover point in unit costs between COTS and custom-build
% -------------------------------------------------------------------------
find_crossover = Cust_unit_cost - COTS_unit_cost;   % First order crossover
find_cross1 = zeros(1,N_indx(1,2));
find_cross2 = zeros(1,N_indx(1,2));
find_cross3 = zeros(1,N_indx(1,2));
% Find interval where crossover occurs, depends on whether custom-build
% is decreasing or increasing
cross_point = diff(sign(find_crossover));
find_cross1(1,1:(N_indx(1,2)-1)) = cross_point;
find_cross2(1,2:N_indx(1,2)) = cross_point; % Interval end if decreasing
cross_point(1:1) = [];
find_cross3(1,1:(N_indx(1,2)-2)) = cross_point;% Interval end if increasing
check_cross = sum(find_cross1);             % Does crossover exist?

if check_cross == 0                         % =0, then no crossover
    fprintf('No crossover point found\n')
    crossover_i = [0 0];
    iCOTSmodp = max(COTS_mod_p);
    iCOTSCcost = max(COTS_COGS_cost);
    iCustmodp = max(Cust_mod_p);
    iCustCcost = max(Cust_COGS_cost);
else                                        % Find crossover indices
    crossover_i = [0 0];
    select_i1 = N.*(find_cross1/check_cross);
    select_i2 = N.*(find_cross2/check_cross);
    select_i3 = N.*(find_cross3/check_cross);
    if check_cross < 0              % - means custom-build is decreasing
        crossover_i(1,1) = max(select_i1);
        crossover_i(1,2) = max(select_i2);
        % Calculate the module and COGS costs for this decreasing interval
        iCOTSmodp = max(COTS_mod_p.*(select_i1/crossover_i(1,1)));
        iCOTSCcost = max(COTS_COGS_cost.*(select_i1/crossover_i(1,1)));
        iCustmodp = max(Cust_mod_p.*(select_i1/crossover_i(1,1)));
        iCustCcost = max(Cust_COGS_cost.*(select_i1/crossover_i(1,1)));
    else                            % + means custom-build is increasing
        crossover_i(1,1) = max(select_i3);
        crossover_i(1,2) = max(select_i1);
        % Calculate the module and COGS costs for this increasing interval
        iCOTSmodp = max(COTS_mod_p.*(select_i3/crossover_i(1,1)));
        iCOTSCcost = max(COTS_COGS_cost.*(select_i3/crossover_i(1,1)));
        iCustmodp = max(Cust_mod_p.*(select_i3/crossover_i(1,1)));
        iCustCcost = max(Cust_COGS_cost.*(select_i3/crossover_i(1,1)));
    end
end

% -------------------------------------------------------------------------
%   Calcualte the crossover point in the selected interval
% -------------------------------------------------------------------------
M = crossover_i(1,1):1:crossover_i(1,2);    % Interval with the crossover
% Form vectors of the module and COGS costs for both COTS and custom-build
iCOTS_mod_p = ones(size(M))*iCOTSmodp;
iCOTS_COGS_cost = ones(size(M))*iCOTSCcost;
iCust_mod_p = ones(size(M))*iCustmodp;
iCust_COGS_cost = ones(size(M))*iCustCcost;
% Calculate the unit costs and then find the difference between
% COTS and custom-build
iCOTS_unit_cost = iCOTS_COGS_cost + iCOTS_mod_p + (COTS_NRE_cost./M);
iCust_unit_cost = iCust_COGS_cost + iCust_mod_p + (Cust_NRE_cost./M);

idiff_unit_cost = iCust_unit_cost - iCOTS_unit_cost;
cost_diff = abs(idiff_unit_cost);
icross_point = zeros(size(M));
icross_point(1:length(M)-1) = diff(sign(idiff_unit_cost));
icross_val = sum(icross_point);
break_point = sum(M.*(icross_point/icross_val));
COTS_cross_mod_cost = sum(iCOTS_unit_cost.*(icross_point/icross_val));
Cust_cross_mod_cost = sum(iCust_unit_cost.*(icross_point/icross_val));

% -------------------------------------------------------------------------
%   Print crossover point and module costs
% -------------------------------------------------------------------------
fprintf('\n')
fprintf('Crossover point (number of modules) = %.f\n',break_point)
fprintf('Crossover COTS module cost = $ %.f\n',COTS_cross_mod_cost)
fprintf('Crossover custom-build module cost = $ %.f\n',Cust_cross_mod_cost)
fprintf('\n')

if j == 1                   % set index into vector for output
    indx = (i * 3) - 2;
else
    indx = (i * 3) - 1;
end
St3(indx) = break_point;    % store a crossover from Stage 3 in the vector

end
end
fprintf('##############################################\n')
fprintf('##############################################\n')
fprintf('\n')

% /////////////////////////////////////////////////////////////////////////
% -------------------------------------------------------------------------
%   STAGE 4
% -------------------------------------------------------------------------
% /////////////////////////////////////////////////////////////////////////
% -------------------------------------------------------------------------
%   COTS NRE calculations for development of module incorporation in the
%   system. The NRE calculations then results in cost per unit COTS module.
%   These calculations assume exact estimation of effort, FTE, but multiply
%   the calendar time according to four (4) different Rayleigh medians to
%   predict various Overruns and according to for two (2) different
%   sigmoid adjustments. They finish with adjusting COTS effort for 25%
%   deficiencies in specifications, quality of support and documentation.
% -------------------------------------------------------------------------
COTS_sig = [(C_fac * 0.5) C_fac];
St4 = zeros(12,1);              % initialize output vector for Stage 4

for j = 1:2
for i = 1:4
    COTS_Cal_adj = COTS_Calendar * Rayl_adj(i) * COTS_sig(j);
COTS_person_month = COTS_NRE_FTE * COTS_Cal_adj;
COTS_Tot_effort = sum(COTS_person_month);
    %Total NRE labor cost for project ($), assume 168 hrs/month
COTS_Tot_labor = Dev_labor * COTS_person_month * 168;
COTS_Tot_time = sum(COTS_Cal_adj);  %Total calendar time for NRE (months)

% Calculate total COTS costs from NRE, tooling, and compliance/certification
COTS_NRE_cost = COTS_Tot_labor + COTS_tool_cost + COTS_cert_cost;

% Calculate prices for COTS modules with discounts and inventory premium
COTS_mod_p = COTS_discount * COTS_price * inven_prem2;

% Set up COTS cost vector from total COGS cost
COTS_COGS_cost = ones(1,N_indx(1,2)) * COTS_COGS_tot_cost;

% Calculate vector of COTS unit prices at the cost breakpoints
COTS_unit_cost = COTS_COGS_cost + COTS_mod_p + (COTS_NRE_cost ./N);
% -------------------------------------------------------------------------
%   NRE calculations for development of a custom-build module in the
%   system. The NRE calculations then results in cost per unit module.
% -------------------------------------------------------------------------
    Cust_Cal_adj = Cust_Calendar * Rayl_adj(i) * Cust_sig(j);
Cust_person_month = Cust_NRE_FTE * Cust_Cal_adj;
Cust_Tot_effort = sum(Cust_person_month);
%Total NRE labor cost for project ($), assume 168 hrs/month
Cust_Tot_labor = Dev_labor * Cust_person_month * 168;
Cust_Tot_time = sum(Cust_Cal_adj); %Total calendar time for NRE (months)

% Calculate build costs from NRE, tooling, and compliance/certification
Cust_NRE_cost = Cust_Tot_labor + Cust_tool_cost + Cust_cert_cost;

% Calculate prices for custom-built modules with discounts and inventory
Cust_mod_p = Comp_discount * Comp_price * inven_prem1;

% Set up build cost vector from total COGS cost
Cust_COGS_cost = ones(1,N_indx(1,2)) * Cust_COGS_tot_cost;

% Calculate the lost opportunity costs
lost_opp_cost = (Cust_Tot_time - COTS_Tot_time) * Early_sales_rate *
Sys_revenue;

fprintf('----------------------------------------------\n')
fprintf('     STAGE 4: COTS deficiencies = 0.25\n')
if j == 1
    fprintf('              Sigmoid adjustment = 0.25\n')
else
    fprintf('              Sigmoid adjustment = 0.75\n')
end
fprintf('              Rayleigh adjustment = %1.2f\n',Rayl_adj(i))
fprintf('----------------------------------------------\n')
fprintf('Time of NRE for COTS (months) = %3.2f\n',COTS_Tot_time)
fprintf('Time of NRE for custom (months) = %3.2f\n',Cust_Tot_time)
fprintf('Cost of NRE for COTS = $ %.f\n',COTS_NRE_cost)
fprintf('Cost of NRE for custom = $ %.f\n',Cust_NRE_cost)
fprintf('COTS sigmoid adjustment (add 0.25 factor) = %3.2f\n',COTS_sig(j))
fprintf('Custom sigmoid adjustment = %3.2f\n',Cust_sig(j))

% Calculate vector of COTS unit prices at the cost breakpoints
Cust_unit_cost = Cust_COGS_cost + Cust_mod_p + (Cust_NRE_cost ./N);

% --------------------------------------------------------------------------
%   Find the crossover point in unit costs between COTS and custom-build
% --------------------------------------------------------------------------
find_crossover = Cust_unit_cost - COTS_unit_cost;   % First order crossover
find_cross1 = zeros(1,N_indx(1,2));
find_cross2 = zeros(1,N_indx(1,2));
find_cross3 = zeros(1,N_indx(1,2));
% Find interval where crossover occurs, depends on whether custom-build
% is decreasing or increasing
cross_point = diff(sign(find_crossover));
find_cross1(1,1:(N_indx(1,2)-1)) = cross_point;
find_cross2(1,2:N_indx(1,2)) = cross_point; % Interval end if decreasing
cross_point(1:1) = [];
find_cross3(1,1:(N_indx(1,2)-2)) = cross_point;% Interval end if increasing
check_cross = sum(find_cross1);             % Does crossover exist?

if check_cross == 0                         % =0, then no crossover
    fprintf('No crossover point found\n')
    crossover_i = [0 0];
    iCOTSmodp = max(COTS_mod_p);
    iCOTSCcost = max(COTS_COGS_cost);
    iCustmodp = max(Cust_mod_p);
    iCustCcost = max(Cust_COGS_cost);
else                                        % Find crossover indices
    crossover_i = [0 0];
    select_i1 = N.*(find_cross1/check_cross);
    select_i2 = N.*(find_cross2/check_cross);
    select_i3 = N.*(find_cross3/check_cross);
    if check_cross < 0              % - means custom-build is decreasing
        crossover_i(1,1) = max(select_i1);
        crossover_i(1,2) = max(select_i2);
        % Calculate the module and COGS costs for this decreasing interval
        iCOTSmodp = max(COTS_mod_p.*(select_i1/crossover_i(1,1)));
        iCOTSCcost = max(COTS_COGS_cost.*(select_i1/crossover_i(1,1)));
        iCustmodp = max(Cust_mod_p.*(select_i1/crossover_i(1,1)));
        iCustCcost = max(Cust_COGS_cost.*(select_i1/crossover_i(1,1)));
    else                            % + means custom-build is increasing
        crossover_i(1,1) = max(select_i3);
        crossover_i(1,2) = max(select_i1);
        % Calculate the module and COGS costs for this increasing interval
        iCOTSmodp = max(COTS_mod_p.*(select_i3/crossover_i(1,1)));
        iCOTSCcost = max(COTS_COGS_cost.*(select_i3/crossover_i(1,1)));
        iCustmodp = max(Cust_mod_p.*(select_i3/crossover_i(1,1)));
        iCustCcost = max(Cust_COGS_cost.*(select_i3/crossover_i(1,1)));
    end
end

% -------------------------------------------------------------------------
%   Calcualte the crossover point in the selected interval
% -------------------------------------------------------------------------
M = crossover_i(1,1):1:crossover_i(1,2);    % Interval with the crossover
% Form vectors of the module and COGS costs for both COTS and custom-build
iCOTS_mod_p = ones(size(M))*iCOTSmodp;
iCOTS_COGS_cost = ones(size(M))*iCOTSCcost;
iCust_mod_p = ones(size(M))*iCustmodp;
iCust_COGS_cost = ones(size(M))*iCustCcost;
% Calculate the unit costs and then find the difference between
% COTS and custom-build
iCOTS_unit_cost = iCOTS_COGS_cost + iCOTS_mod_p + (COTS_NRE_cost./M);
iCust_unit_cost = iCust_COGS_cost + iCust_mod_p + (Cust_NRE_cost./M);

idiff_unit_cost = iCust_unit_cost - iCOTS_unit_cost;
cost_diff = abs(idiff_unit_cost);
icross_point = zeros(size(M));
icross_point(1:length(M)-1) = diff(sign(idiff_unit_cost));
icross_val = sum(icross_point);
break_point = sum(M.*(icross_point/icross_val));
COTS_cross_mod_cost = sum(iCOTS_unit_cost.*(icross_point/icross_val));
Cust_cross_mod_cost = sum(iCust_unit_cost.*(icross_point/icross_val));

% -------------------------------------------------------------------------
%   Print crossover point and module costs
% -------------------------------------------------------------------------
fprintf('\n')
fprintf('Crossover point (number of modules) = %.f\n',break_point)
fprintf('Crossover COTS module cost = $ %.f\n',COTS_cross_mod_cost)
fprintf('Crossover custom-build module cost = $ %.f\n',Cust_cross_mod_cost)
fprintf('\n')

if j == 1                   % set index into vector for output
    indx = (i * 3) - 2;
else
    indx = (i * 3) - 1;
end
St4(indx) = break_point;    % store a crossover from Stage 4 in the vector

end
end
fprintf('##############################################\n')
fprintf('##############################################\n')
fprintf('\n')
```

Appendix I - Actual Effort for the Two Case Studies

Background Information for the Neurostimulator Programmer

This project was part of a medical company that I co-founded in 1998 and sold in 2003. This information may be found in reference [256].

Chronic pain in the lower back and legs may be treated in several different ways. Since the 1980s, electrical stimulation of the spinal cord, which blocks pain signals traveling up the spinal cord to the brain, has proven effective. A small but important market has grown for implanting these stimulators in patients with chronic pain. After release from the hospital, a patient may program the desired stimulation through a transmitter that encodes parameters and then couples them through radio frequency (RF) to the implant.

Adjusting the stimulator for each patient, however, is extremely tedious because there are many possible combinations of parameter values. Just eight electrodes represent 9200 combinations of polarity while 16 electrodes represent over 100 million possible combinations. A sophisticated programmer/transmitter, for use by medical staffs and patients, can ease the burden of adjustment by selecting appropriate subsets of stimulation parameters from the wide variety available. The programmer that my company designed was a pentop computer that allows patients to tailor their own treatment by drawing simple lines and touching screen buttons. Figure 8.33 shows a prototype of such a programmer.

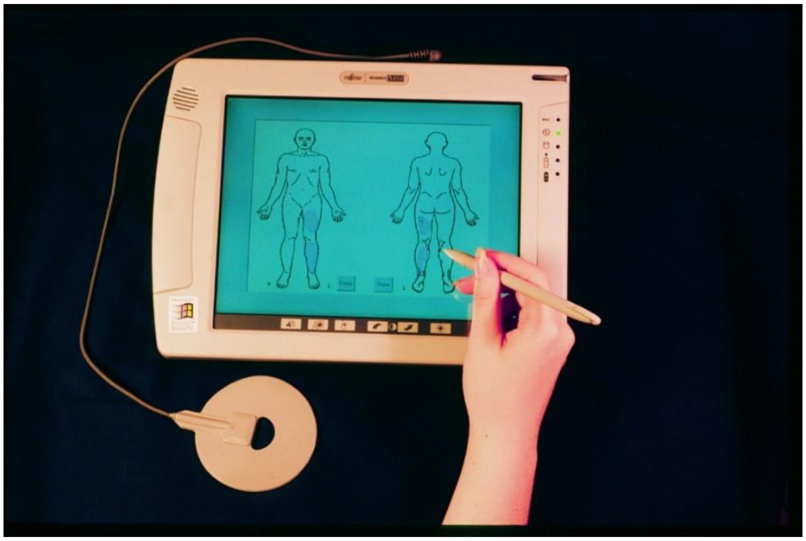


Figure 8.33 Prototype of a programmer tablet for a neurostimulator. (From the personal collection of Kim Fowler. Used with permission.)

The programmer was to be used primarily by patients in physicians' offices. A physician or assistant taped the programmer's antenna over the site of the implant, set up the session on the programmer, and then handed the programmer to the patient. Patients responded to simple instructions and answered questions posed on the screen; they also drew outlines of where the felt both pain and stimulation effects.

I helped develop the concept for this device over 13 years, between 1985 and 1998, with partners in university medicine at The Johns Hopkins Hospital, Baltimore, Maryland. In 1998 I co-founded Stimsoft in Maryland, USA with two other partners to develop the programmer into a commercial device.

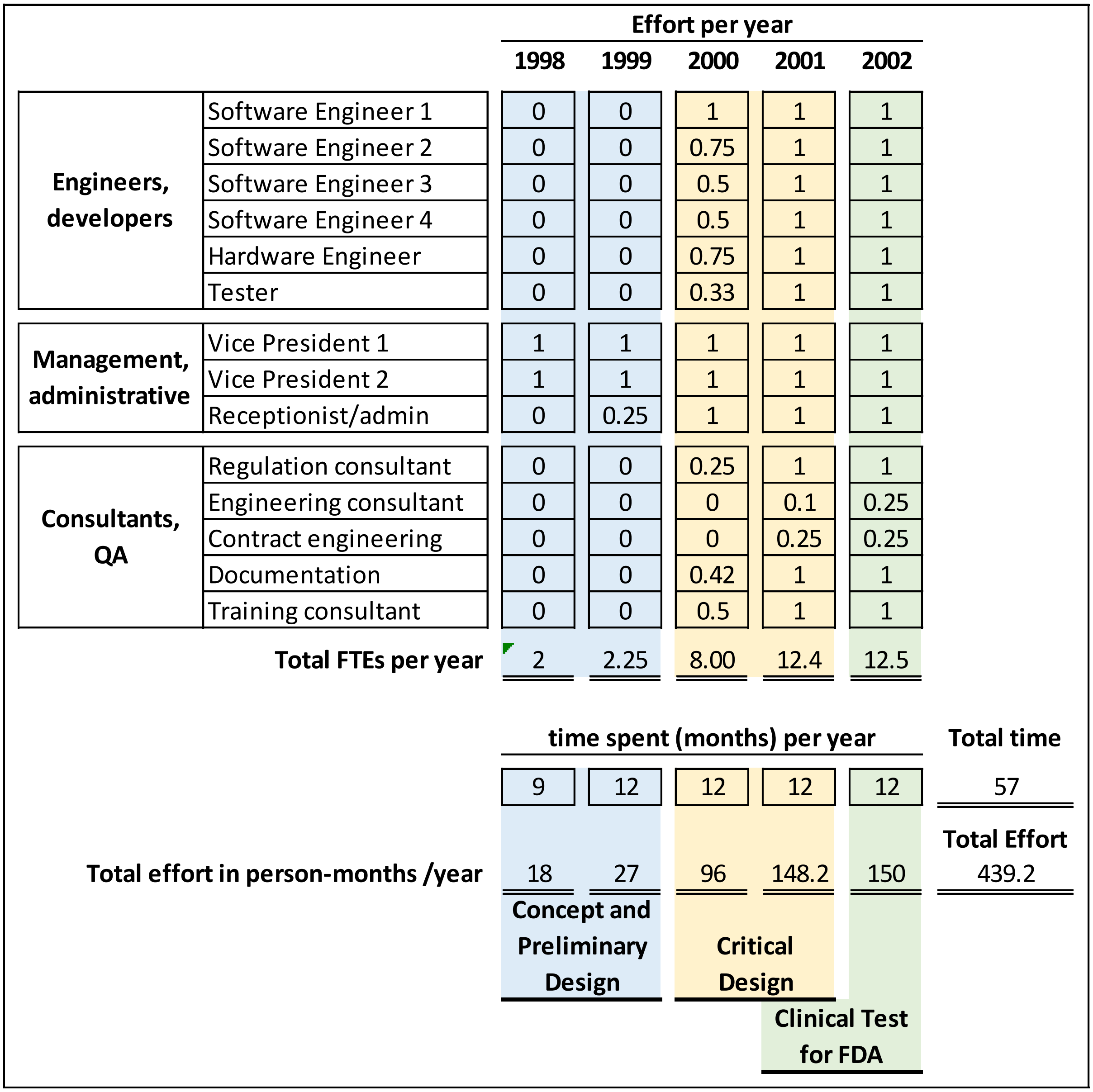
Stimsoft eventually had a team of 12 full time employees to design and develop the programmer. That team comprised four software engineers, one hardware engineer who also wrote software, a software tester, an FDA process and regulation specialist, a training specialist, a specialist for documentation and training, and an office administrator who doubled as receptionist. The two vice presidents also directed design of the architecture and reviewed technical progress. There was also some contract and consulting engineering time of about one quarter of a year.

Clinical testing for FDA approval required participation by three or more medical centers outside of the Johns Hopkins Hospital system. At least one physician and two or three physician assistants in each center participated in the study of the programmer with patients, but they are not included in the estimation of effort by the design/development team. The total time for clinical testing was approximately six months, during which the design/development team would have been active. The device did not receive FDA approval and the company ceased operations one-third of the way into 2002.

Compilation of Effort and Time for the Neurostimulator Programmer

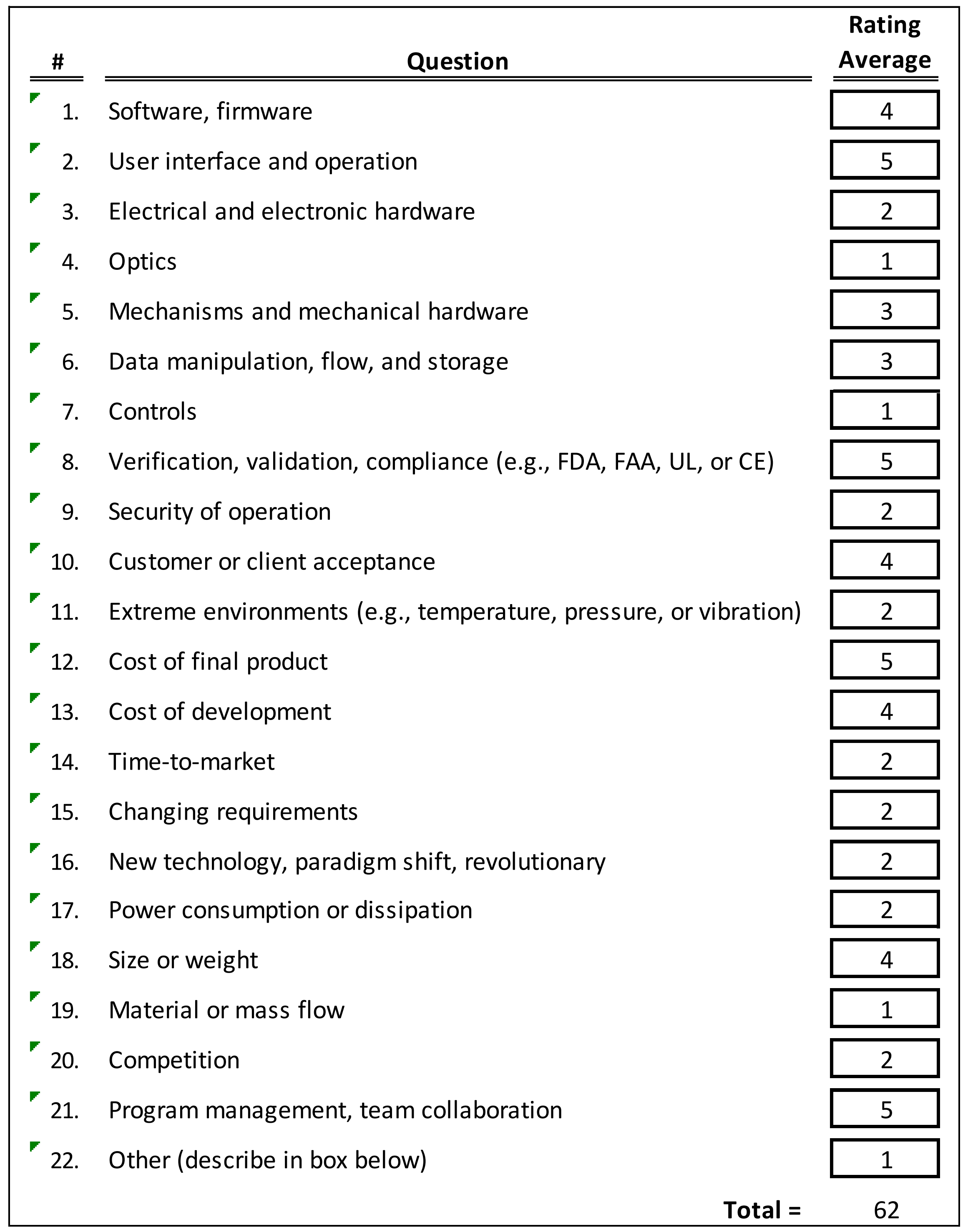
Assuming the company had gone on to complete the FDA approval and we had completed 2002, then the effort would have been about 440 person-months total for a period of just about five years. This would NOT include going into production or marketing and sales, which would have added two calendar years and a sizable effort. Table 8.4 shows the results of completing the project.

Table 8.4 The total effort and time to develop the neurostimulator programmer.



		Effort per year					
		1998	1999	2000	2001	2002	
Engineers, developers	Software Engineer 1	0	0	1	1	1	
	Software Engineer 2	0	0	0.75	1	1	
	Software Engineer 3	0	0	0.5	1	1	
	Software Engineer 4	0	0	0.5	1	1	
	Hardware Engineer	0	0	0.75	1	1	
	Tester	0	0	0.33	1	1	
Management, administrative	Vice President 1	1	1	1	1	1	
	Vice President 2	1	1	1	1	1	
	Receptionist/admin	0	0.25	1	1	1	
Consultants, QA	Regulation consultant	0	0	0.25	1	1	
	Engineering consultant	0	0	0	0.1	0.25	
	Contract engineering	0	0	0	0.25	0.25	
	Documentation	0	0	0.42	1	1	
	Training consultant	0	0	0.5	1	1	
	Total FTEs per year	2	2.25	8.00	12.4	12.5	
		time spent (months) per year					Total time
		9	12	12	12	12	57
							Total Effort
	Total effort in person-months /year	18	27	96	148.2	150	439.2
		Concept and Preliminary Design		Critical Design			
					Clinical Test for FDA		

Table 8.5 Challenges encountered by the neurostimulator programmer.



#	Question	Rating Average
1.	Software, firmware	4
2.	User interface and operation	5
3.	Electrical and electronic hardware	2
4.	Optics	1
5.	Mechanisms and mechanical hardware	3
6.	Data manipulation, flow, and storage	3
7.	Controls	1
8.	Verification, validation, compliance (e.g., FDA, FAA, UL, or CE)	5
9.	Security of operation	2
10.	Customer or client acceptance	4
11.	Extreme environments (e.g., temperature, pressure, or vibration)	2
12.	Cost of final product	5
13.	Cost of development	4
14.	Time-to-market	2
15.	Changing requirements	2
16.	New technology, paradigm shift, revolutionary	2
17.	Power consumption or dissipation	2
18.	Size or weight	4
19.	Material or mass flow	1
20.	Competition	2
21.	Program management, team collaboration	5
22.	Other (describe in box below)	1
	Total =	62

The rating scale was:

1 = **Low.** Effort, cost, and technology are understood and controlled; e.g., system upgrade to a servomechanism that does not require software changes and only minor differences in the motor and drive.

2 = Effort, cost, and technology are between Low and Medium.

3 = **Medium.** Effort, cost, and technology are understood; e.g., clean-sheet design of a servomechanism that uses available components.

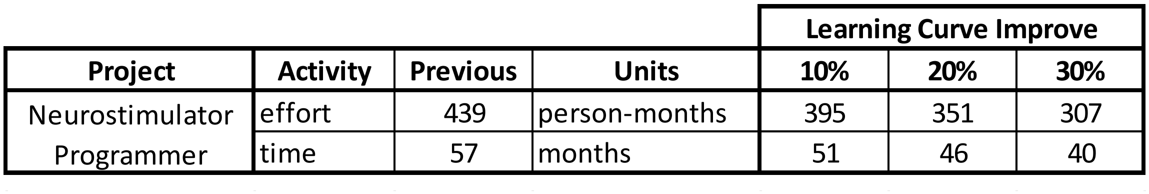
4 = Effort, cost, and technology are between Medium and High.

5 = **High.** Effort, cost, or technology are new to the team; e.g., clean-sheet design of a spacecraft instrument with new software and autonomy.

Adjustment to Realistic Estimates for the Neurostimulator Programmer

The record in Table 8.4 is skewed to more effort than an experienced and competent team might take. The team in Table 8.4 was a new company and all the employees were new to developing medical products; also, all worked on the one project, there were no follow-on projects. Consequently, I would expect the team to have moved down the learning curve somewhat, so that fewer people and a lower effort are expended in a follow-on project. Table 8.6 gives several revised views for completing the project before production. I would expect the team to have moved down the learning curve somewhat, requiring lower design effort and less time for learning curve improvements of 10, 20, and 30% better than in Table 8.5.

Table 8.6 A lower estimate for an experienced team of engineers and developers that have reduced effort and time through learning curve improvement.



				Learning Curve Improve		
Project	**Activity**	**Previous**	**Units**	**10%**	**20%**	**30%**
Neurostimulator Programmer	effort	439	person-months	395	351	307
	time	57	months	51	46	40

Background Information for the Arc-Fault Detector (AFD)

I worked with Bruce Land at the Applied Physics Laboratory of the Johns Hopkins University from late 1982 to 1991 on the Arc-Fault Detector (AFD) program. To develop a historically accurate record that supports my bottom-up estimate, I contacted Bruce and asked for his recollections on the program. The program went through major iterations, the first from 1979 to 1995, and the second from 1995 to early 2000s. The AFD system originally installed on US Navy submarines. The second phase of the system (from 1995) installed on US Navy carriers.

Personal Communication from Bruce Land

Here is what Bruce Land wrote me on February 14, 2018 about the AFD history:

"The sub and CVN AFD systems were each big efforts and I am hard pressed to rank them. With the original sub [submarine] system we were in the raw sensor design effort which [took] a long time and involved fewer people. We had to invent a new data system to obtain the high speed data we needed from arc testing. The BIT [built-in-test] board was designed from scratch, but it only had the task of BIT. The BIT flow chart was proven first in HP Basic and transported onto the Z80 by a bright young engineer. I remember him complaining about the detail and rigor of the testing of the BIT board software [no I didn't]. All of the system operational logic was done by discrete CMOS chips. Perhaps the most difficult task was laying out of multilayer PCBs using tape and mylar. We spent a lot of time understanding the submarine environment in general and the switchboard environments. We had several levels of shipboard tests of subsystems and systems.

The COTs CVN system did use 5 off the shelf PC104 board. It required a massive mother board to handle all of the I/O. We had a LCD operator panel rather than the LEDs of the sub system. By the time the software was almost complete 4 of the PC104 boards were obsolete, but the replacements had more functionality and we could replace the 4 with 3. [WS] certainly did all of the control unit software, in Forth. The operator interface design was much more complex. I think he did most of the control unit electrical design, but I am not sure at this point. [WS] developed the high-speed interface between the control unit and the sensor interface module (SIM). [DW], a young engineer, did the SIM hardware and software design. At this point the photo sensor design was being improved. The sensors were redesigned to have analog outputs into the SIMs. Both sub and COTs went through arc testing at the system level and the MIL-SPEC qualification testing. A single COTS system underwent deployment testing.

I remember that our first R&D funding for sensors began in 1979 and the first sub production contract was awarded in 1990. We moved into the improved version 2 of the sub system with the development of the TID, and improved photo. This added a new PCB to the control unit which required an entirely new mechanical design, build, & test effort. At some point in the mid-90s we began the COTS effort and that went into the early part of the next decade. I am fuzzy on the dates for the COTS system, but will think about it.

After all of this time I don't remember the number of people working on the programs. I sometimes had 4-5 simultaneous NAVSEA AFD contracts. The COTS efforts and two sub efforts overlapped. Many of the people were part time or on/off efforts and there were subcontracts to outside companies. (PCB layout, test cell mechanics, PCB assemblers, draftsmen, machinists, technicians, etc.). I remember calculating that in today's dollars the funding reached ~$10M in one year. AFD at one time had more funding than the total of all other direct funded programs in the Research Department."

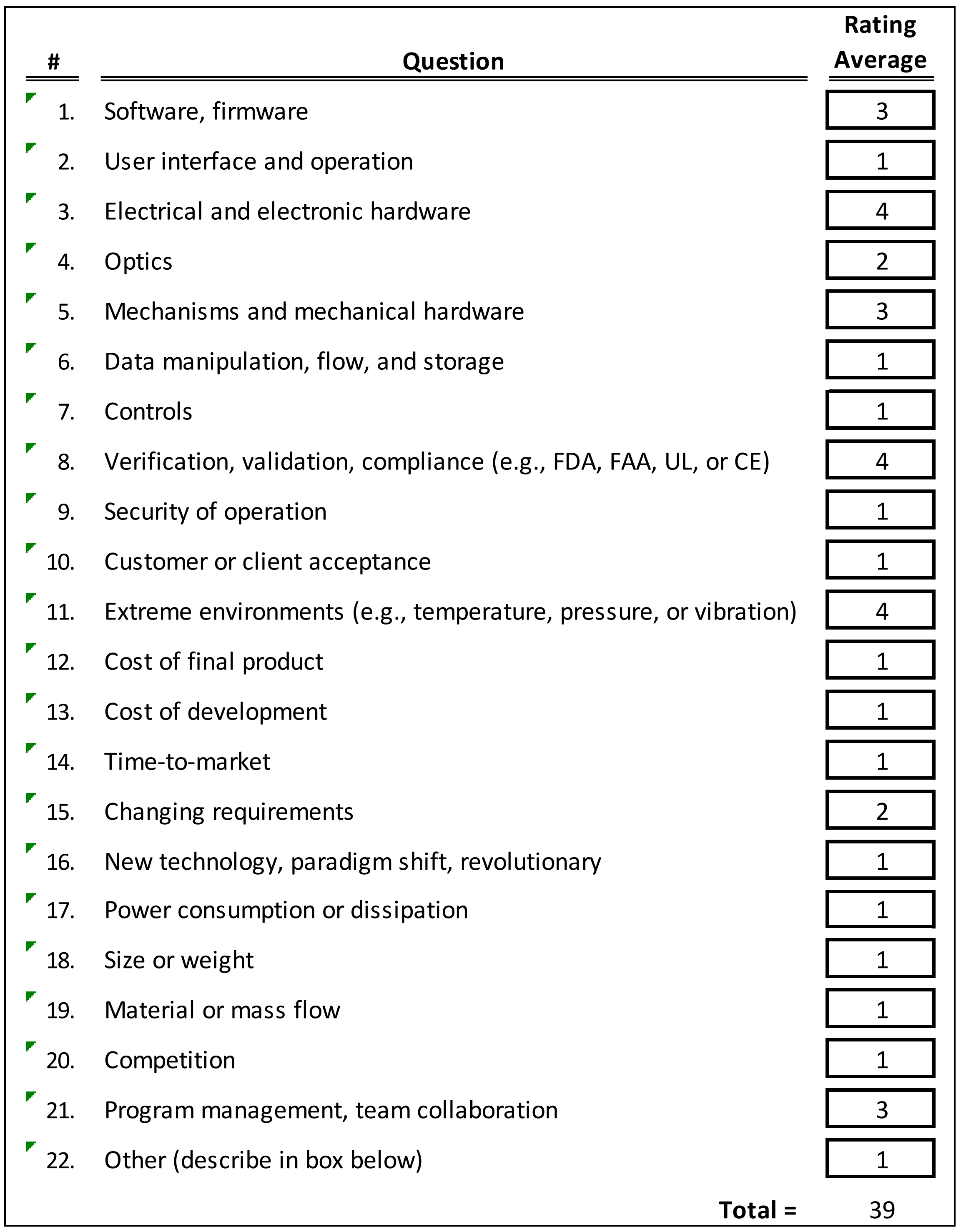
Estimate of Effort for AFD Phase 1 from 1979 to 1990

This part of the effort required very basic and necessary research to determine the best methods for detecting an arcing fault in switchboard cabinets of submarines. Several engineers worked on the basic design; the software for the built-in-test (BIT) was a 2000 byte-long Z80 assembly-language program. This was much more of a cabling and logic circuit design. Table 8.7 shows the results of completing the project before production and taking nearly 1000 person-months; assuming a loaded hourly cost of $125/person, then this effort alone cost about $20.6 MM over 12 years; the cost does not include materials, components, travel, scientific support equipment, or subcontracts. Table 8.8 lists the challenges that I know about the project.

Table 8.7 The total effort and time to develop the Arc-Fault Detector in Phase 1, deployment on submarines.

		Effort per year												
		1979	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	1990	
Engineers, developers	Research Engineer	1	1	1	1	1	1	1	1	1	1	1	1	
	Hardware Engineer 1	0	0	0.5	1	0	0	0	0	0	0	0	0	
	Intern Engineer	0	0	0	0.25	0	0	0	0	0	0	0	0	
	Hardware/Software Engineer	0	0	0	0.17	1	1	1	0.25	0	0	0	0	
	Hardware Engineer 2	0	0	0	0	0	0	0	0.5	1	1	0.5	0.5	
	Technician 1	0	0	1	1	1	1	1	1	1	1	0	0	
	Technician 2	0	0	0.5	0.5	0.5	1	1	1	1	1	0.5	0	
	Technician 3	0	0	0	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0	0	
	Technician 4	0	0	0	0	0.5	0.25	0.25	0	0	0	0	0	
Management, administrative	Program manager	0.5	0.5	0.5	1	1	1	1	1	1	1	1	1	
	Project lead	0.5	0.5	0.5	1	1	1	1	1	1	1	1	1	
	Receptionist/admin	0.5	0.5	0.5	1	1	1	1	1	1	1	1	1	
Drafting, machine shop, and inventory control	Drafting 1	0	0	0.25	0.25	0.25	0.25	0.25	0	0	0.25	0.25	0	
	Drafting 2	0	0	0	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0	0	
	Drafting 3	0	0	0	0.25	0.25	0.25	0.25	0	0	0	0	0	
	Drafting 4	0	0	0	0	0.25	0.25	0.25	0	0	0	0	0	
	Machinist 1	0	0	0	0.25	0.25	0.25	0.25	0.25	0.25	0	0	0	
	Machinist 2	0	0	0	0.25	0.25	0.25	0.25	0.25	0	0	0	0	
	Machinist 3	0	0	0	0	0.25	0.25	0.25	0	0	0	0	0	
	Inventory control 1	0	0	0	0	0.1	0.1	0.1	0	0	0	0	0	
	Inventory control 2	0	0	0	0	0.1	0.2	0.2	0.2	0.2	0.1	0.1	0	
	Total FTEs per year	2.5	2.5	4.75	8.667	9.45	9.8	9.8	8.2	8.2	8.1	5.35	4.5	
		time spent (months) per year												**Total time**
		12	12	12	12	12	12	12	12	12	12	12	12	144
														Total Effort
	Total effort in person-months /year	30	30	57	104	113.4	117.6	117.6	98.4	98.4	97.2	64.2	54	981.8
		Concept and Preliminary Design				**Critical Design**			**Test, Integration, Field Test and Evaluation**					

Table 8.8 Challenges encountered by the Arc-Fault Detector in Phase 1.



#	Question	Rating Average
1.	Software, firmware	3
2.	User interface and operation	1
3.	Electrical and electronic hardware	4
4.	Optics	2
5.	Mechanisms and mechanical hardware	3
6.	Data manipulation, flow, and storage	1
7.	Controls	1
8.	Verification, validation, compliance (e.g., FDA, FAA, UL, or CE)	4
9.	Security of operation	1
10.	Customer or client acceptance	1
11.	Extreme environments (e.g., temperature, pressure, or vibration)	4
12.	Cost of final product	1
13.	Cost of development	1
14.	Time-to-market	1
15.	Changing requirements	2
16.	New technology, paradigm shift, revolutionary	1
17.	Power consumption or dissipation	1
18.	Size or weight	1
19.	Material or mass flow	1
20.	Competition	1
21.	Program management, team collaboration	3
22.	Other (describe in box below)	1
	Total =	39

The rating scale was:

1 = **Low.** Effort, cost, and technology are understood and controlled; e.g., system upgrade to a servomechanism that does not require software changes and only minor differences in the motor and drive.

2 = Effort, cost, and technology are between Low and Medium.

3 = **Medium.** Effort, cost, and technology are understood; e.g., clean-sheet design of a servomechanism that uses available components.

4 = Effort, cost, and technology are between Medium and High.

5 = **High.** Effort, cost, or technology are new to the team; e.g., clean-sheet design of a spacecraft instrument with new software and autonomy.

Estimate of Effort for AFD Phase 2 from 1995 to 2000

This effort did not require the basic research to determine the best methods for detecting an arcing fault in switchboard cabinets of submarines. This was not a clean-sheet design; the operation remained much the same in terms of timing; software replaced the discrete logic and the sensors were redesigned. shows the results of completing the project before production and taking nearly 450 person-months over seven years; the cost does not include materials, components, travel, scientific support equipment, or subcontracts.

Table 8.9 The total effort and time to develop the Arc-Fault Detector in Phase 2, deployment on aircraft carriers.

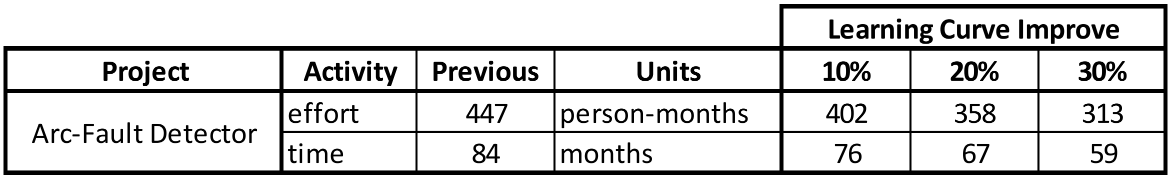
		Effort per year							
		1995	**1996**	**1997**	**1998**	**1999**	**2000**	**2001**	
Engineers, developers	Research Engineer	1	1	1	1	1	1	1	
	Hardware/Software Engineer	1	1	0.25	0.25	0	0	0	
	Software Engineer 2	1	1	1	0.5	0	0	0	
	Technician 1	1	1	1	1	1	0.5	0	
	Technician 2	0.5	1	1	1	0.5	0.5	0	
	Technician 3	0.5	0.25	0.25	0.25	0.25	0	0	
Management, administrative	Program manager	0.5	0.25	0.25	0.25	0.25	0.25	0.25	
	Project lead	0.5	0.5	0.5	0.5	0.5	0.5	0.5	
	Receptionist/admin	1	1	1	0.5	0.5	0.5	0.5	
Drafting, machine shop	Drafting 1	0.25	0.25	0.25	0	0	0	0	
	Drafting 2	0.25	0.25	0.25	0	0	0	0	
	Machinist 1	0.25	0.25	0.25	0	0	0	0	
	FTEs per year	7.75	7.75	7	5.25	4	3.25	2.25	
		time spent (months) per year							**Total time**
		12	12	12	12	12	12	12	84
									Total Effort
	Effort in person-months /year	93	93	84	63	48	39	27	447
		Critical Design			**Test, Integration, Field Test and Evaluation**				

Estimate of effort for AFD Phase 3

This effort also would not require the basic research to determine the best methods for detecting an arcing fault in switchboard cabinets of submarines. This was not a clean-sheet design, it was a follow-on from Phase 2; the operation remained much the same in terms of timing; software would be upgraded, and the sensors may be redesigned. Table 8.10 gives several revised views for completing the project before production. I would expect the team to have moved down the learning curve somewhat, requiring lower design effort and less time for learning curve improvements of 10, 20, and 30% better than in Table 8.9.

Table 8.8 lists the challenges that I perceived would still be with the project.

Table 8.10 A lower estimate for an experienced team of engineers and developers that have reduced effort and time, through learning curve improvement, to develop the Arc-Fault Detector in Phase 3.

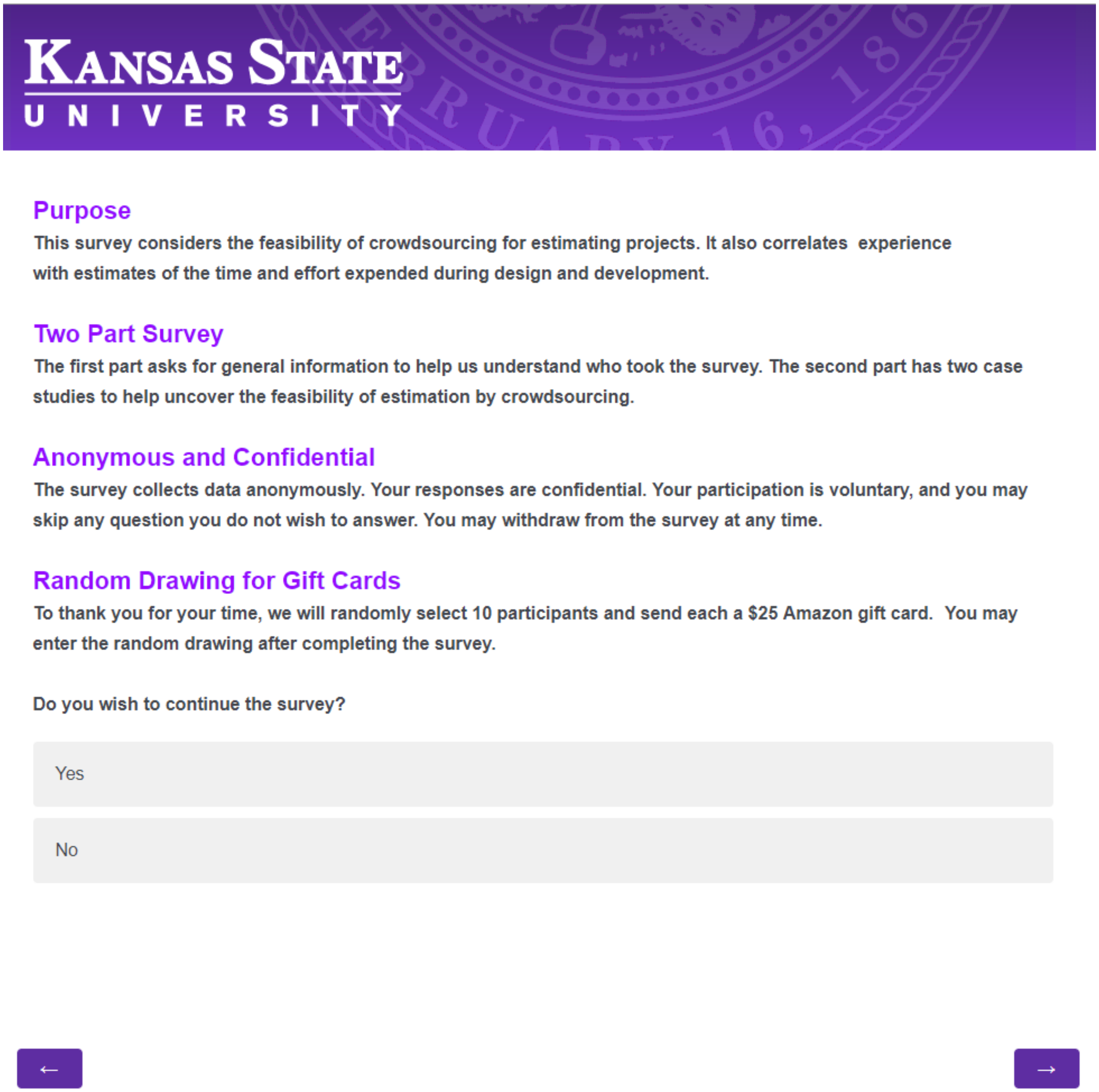


				Learning Curve Improve		
Project	**Activity**	**Previous**	**Units**	**10%**	**20%**	**30%**
Arc-Fault Detector	effort	447	person-months	402	358	313
	time	84	months	76	67	59

Appendix J - Crowdsourcing Survey

Survey Format

The screen shots on this page and the next 14 pages show the survey as seen by participants. The screen shots do not show the progress bar at the top of the screen page. In several screen shots the next or previous buttons do not always show. The raw results of the survey are in the section following the screen shots.



Purpose

This survey considers the feasibility of crowdsourcing for estimating projects. It also correlates experience with estimates of the time and effort expended during design and development.

Two Part Survey

The first part asks for general information to help us understand who took the survey. The second part has two case studies to help uncover the feasibility of estimation by crowdsourcing.

Anonymous and Confidential

The survey collects data anonymously. Your responses are confidential. Your participation is voluntary, and you may skip any question you do not wish to answer. You may withdraw from the survey at any time.

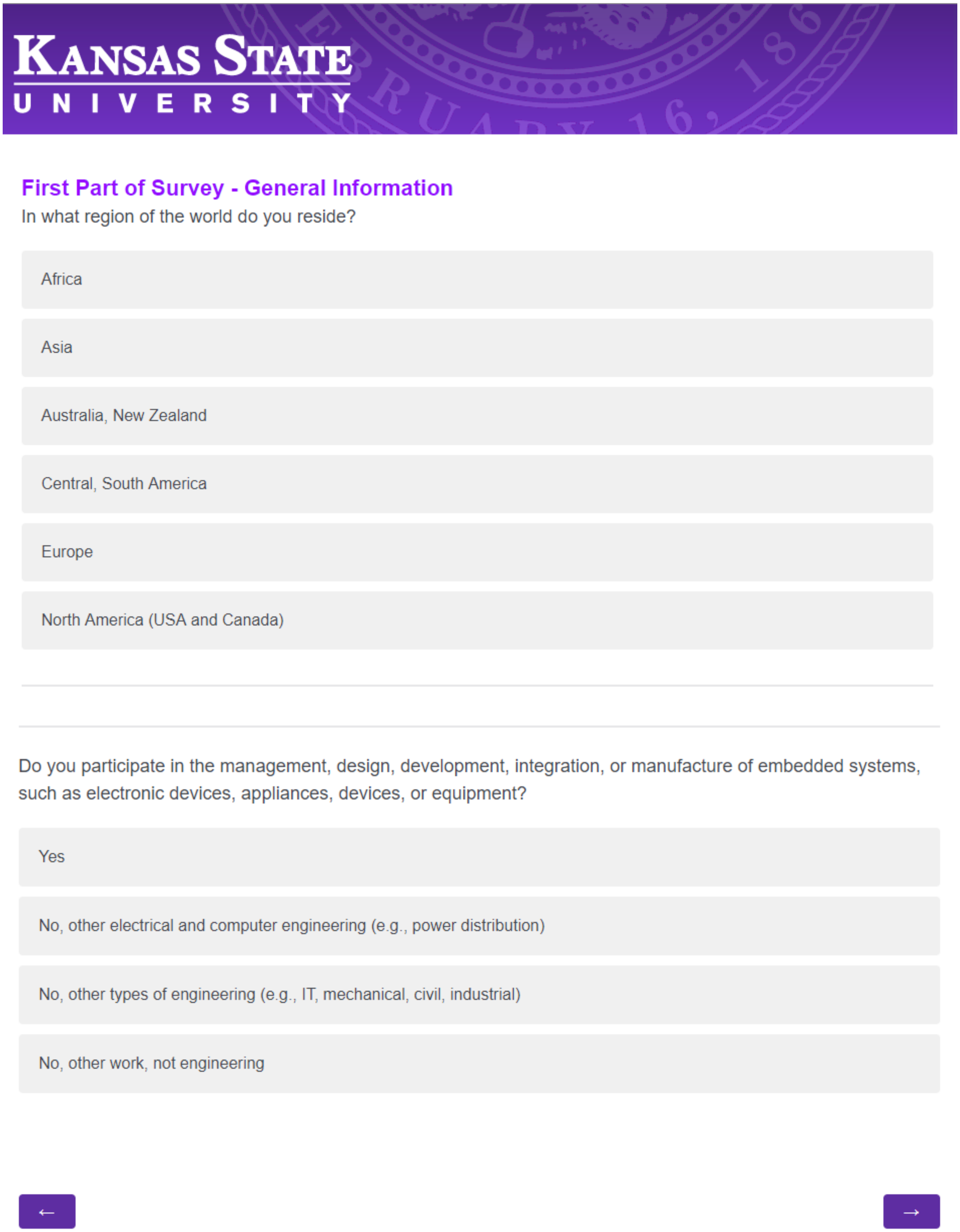
Random Drawing for Gift Cards

To thank you for your time, we will randomly select 10 participants and send each a $25 Amazon gift card. You may enter the random drawing after completing the survey.

Do you wish to continue the survey?

Yes

No



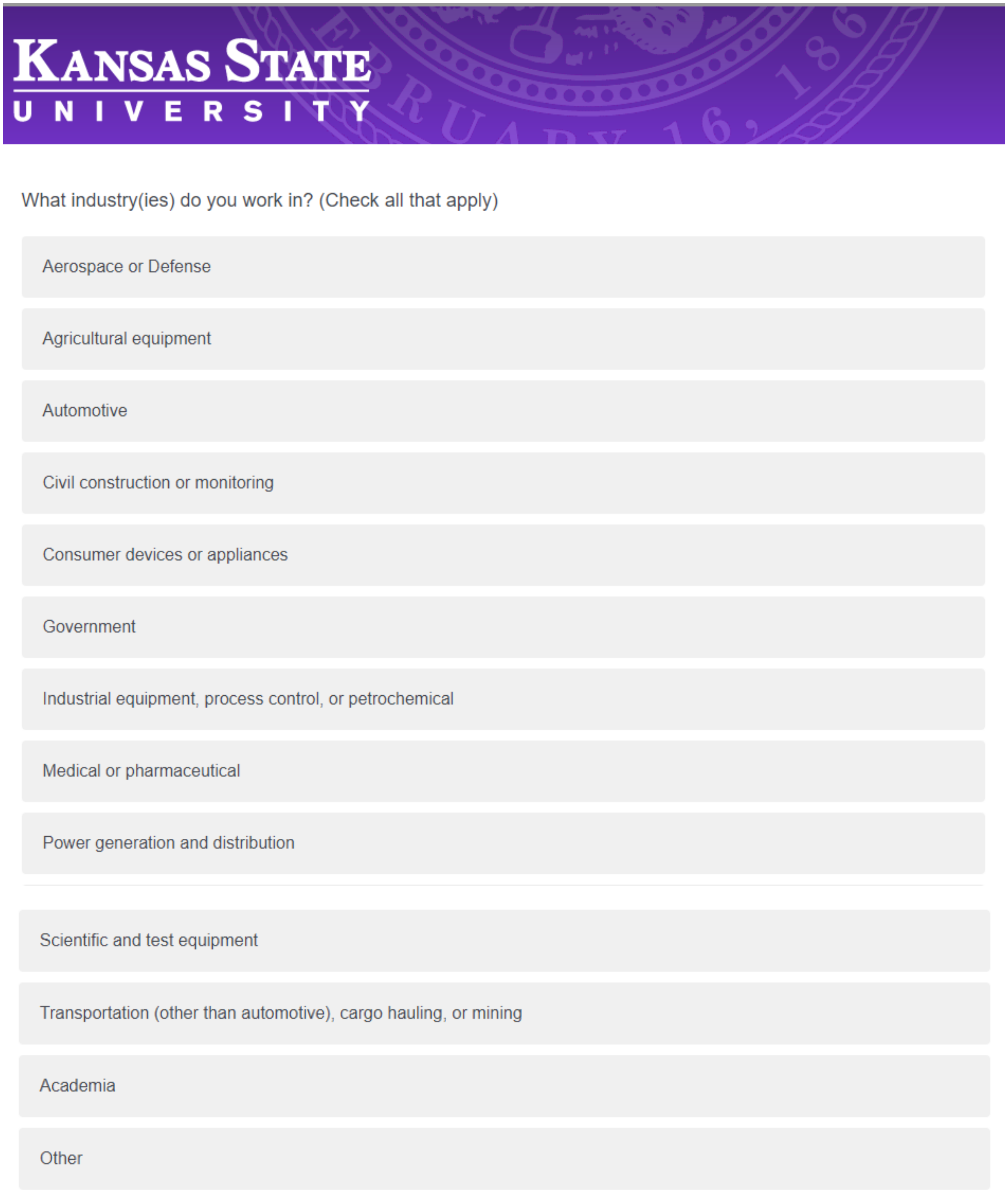
First Part of Survey - General Information

In what region of the world do you reside?

- Africa
- Asia
- Australia, New Zealand
- Central, South America
- Europe
- North America (USA and Canada)

Do you participate in the management, design, development, integration, or manufacture of embedded systems, such as electronic devices, appliances, devices, or equipment?

- Yes
- No, other electrical and computer engineering (e.g., power distribution)
- No, other types of engineering (e.g., IT, mechanical, civil, industrial)
- No, other work, not engineering



KANSAS STATE UNIVERSITY

What industry(ies) do you work in? (Check all that apply)

Aerospace or Defense

Agricultural equipment

Automotive

Civil construction or monitoring

Consumer devices or appliances

Government

Industrial equipment, process control, or petrochemical

Medical or pharmaceutical

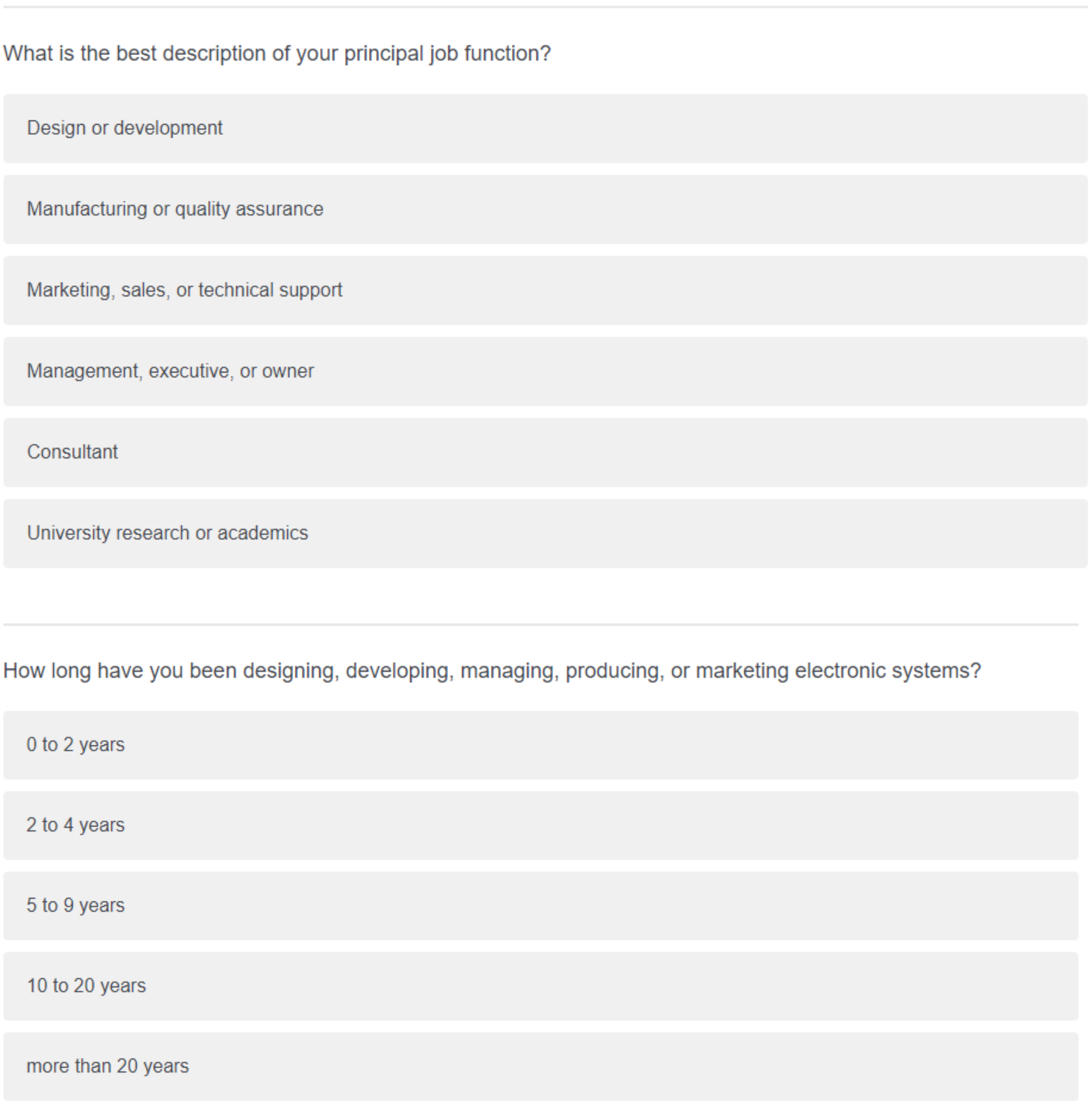
Power generation and distribution

Scientific and test equipment

Transportation (other than automotive), cargo hauling, or mining

Academia

Other



What is the best description of your principal job function?

Design or development

Manufacturing or quality assurance

Marketing, sales, or technical support

Management, executive, or owner

Consultant

University research or academics

How long have you been designing, developing, managing, producing, or marketing electronic systems?

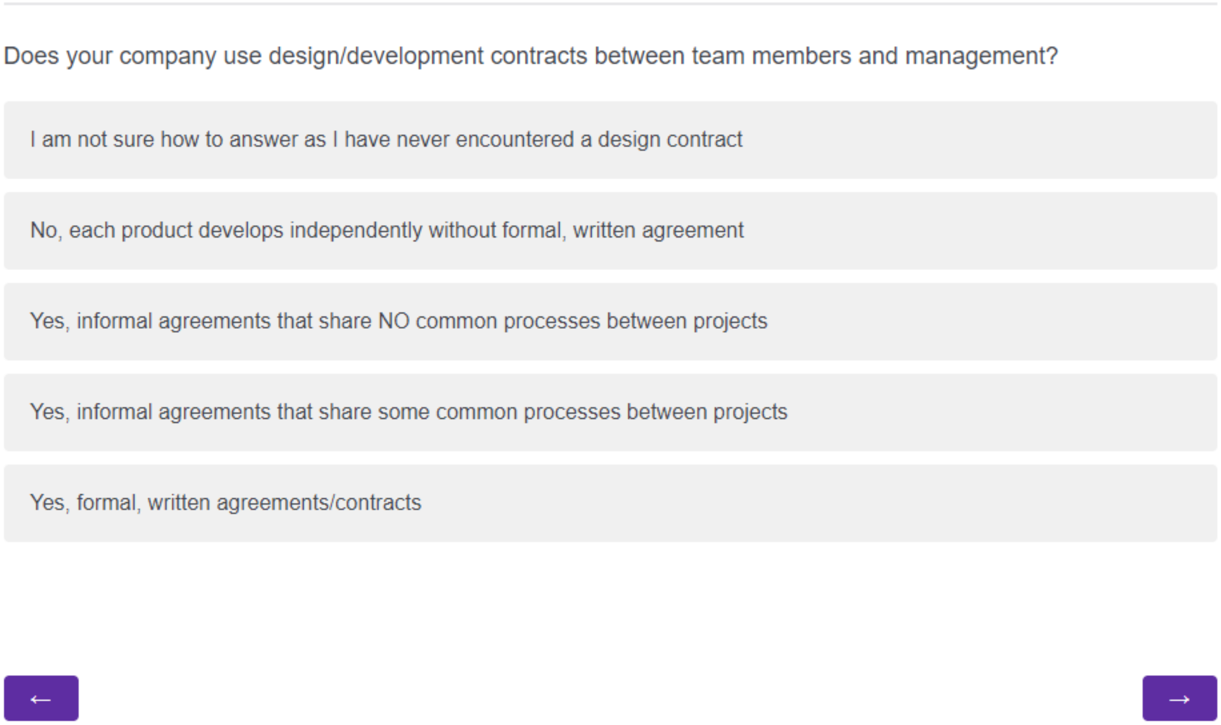
0 to 2 years

2 to 4 years

5 to 9 years

10 to 20 years

more than 20 years



Does your company use design/development contracts between team members and management?

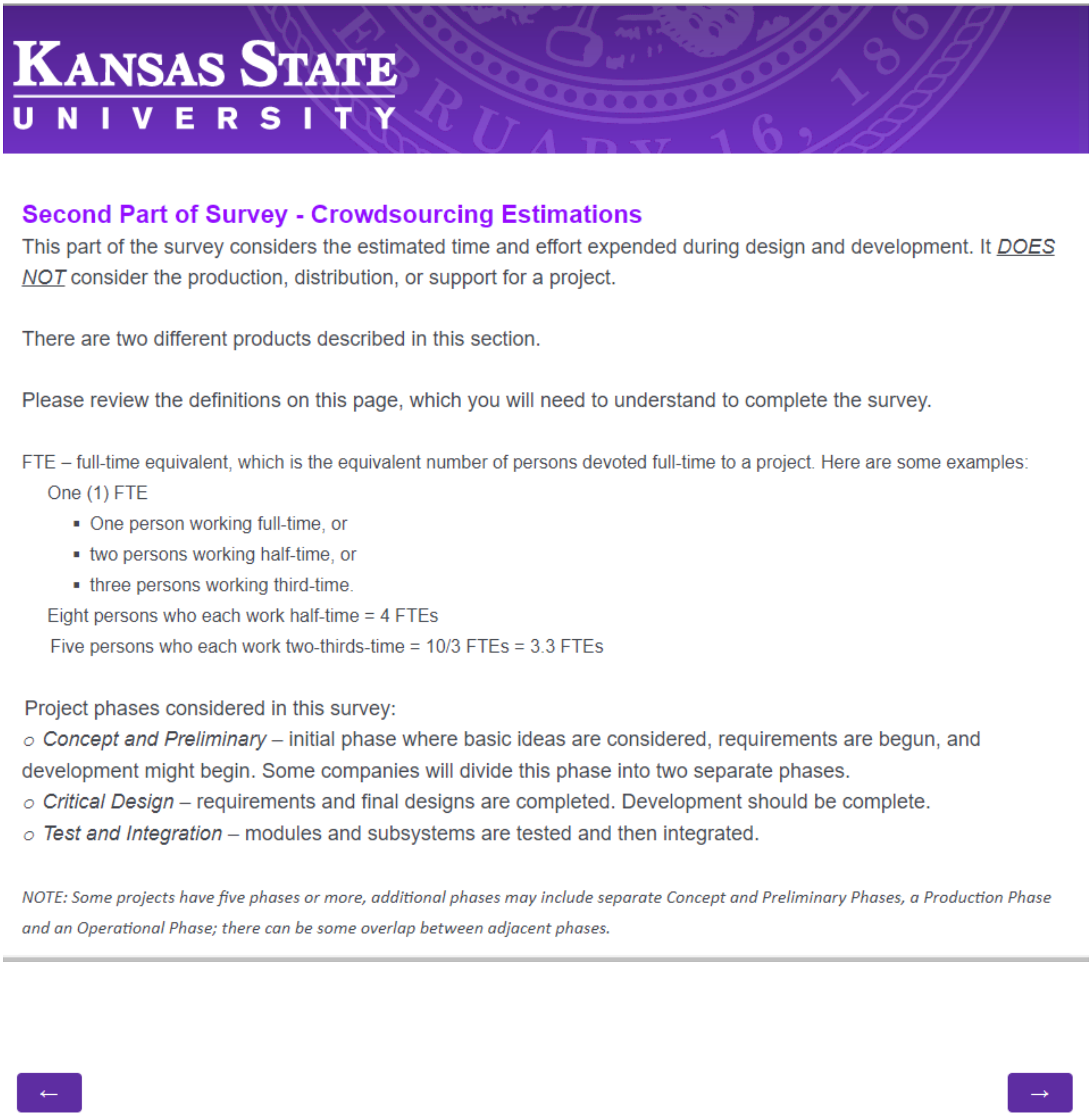
I am not sure how to answer as I have never encountered a design contract

No, each product develops independently without formal, written agreement

Yes, informal agreements that share NO common processes between projects

Yes, informal agreements that share some common processes between projects

Yes, formal, written agreements/contracts



Second Part of Survey - Crowdsourcing Estimations

This part of the survey considers the estimated time and effort expended during design and development. It ***DOES NOT*** consider the production, distribution, or support for a project.

There are two different products described in this section.

Please review the definitions on this page, which you will need to understand to complete the survey.

FTE – full-time equivalent, which is the equivalent number of persons devoted full-time to a project. Here are some examples:

One (1) FTE

- One person working full-time, or
- two persons working half-time, or
- three persons working third-time.

Eight persons who each work half-time = 4 FTEs

Five persons who each work two-thirds-time = 10/3 FTEs = 3.3 FTEs

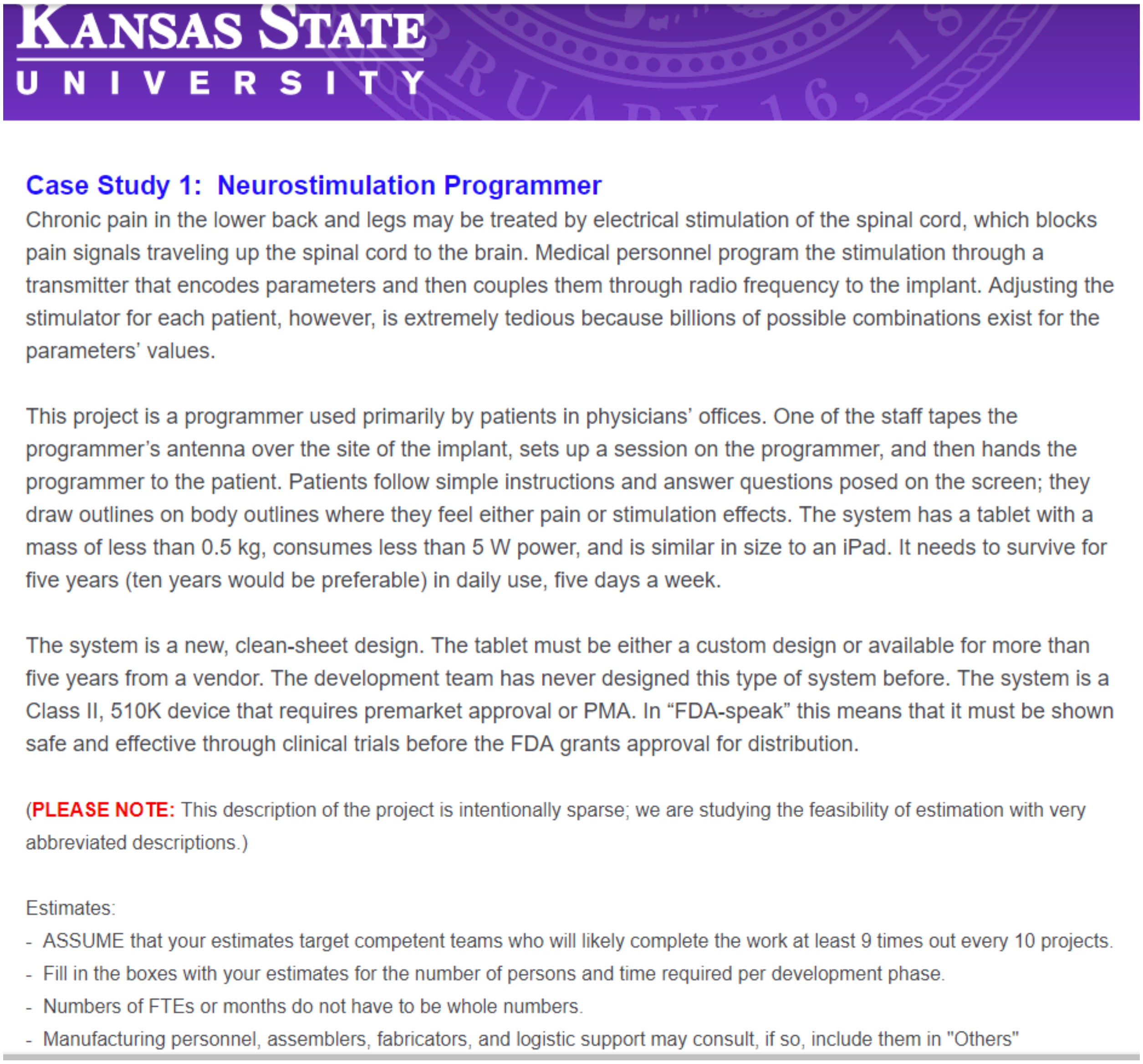
Project phases considered in this survey:

○ *Concept and Preliminary* – initial phase where basic ideas are considered, requirements are begun, and development might begin. Some companies will divide this phase into two separate phases.

○ *Critical Design* – requirements and final designs are completed. Development should be complete.

○ *Test and Integration* – modules and subsystems are tested and then integrated.

NOTE: Some projects have five phases or more, additional phases may include separate Concept and Preliminary Phases, a Production Phase and an Operational Phase; there can be some overlap between adjacent phases.



KANSAS STATE UNIVERSITY

Case Study 1: Neurostimulation Programmer

Chronic pain in the lower back and legs may be treated by electrical stimulation of the spinal cord, which blocks pain signals traveling up the spinal cord to the brain. Medical personnel program the stimulation through a transmitter that encodes parameters and then couples them through radio frequency to the implant. Adjusting the stimulator for each patient, however, is extremely tedious because billions of possible combinations exist for the parameters' values.

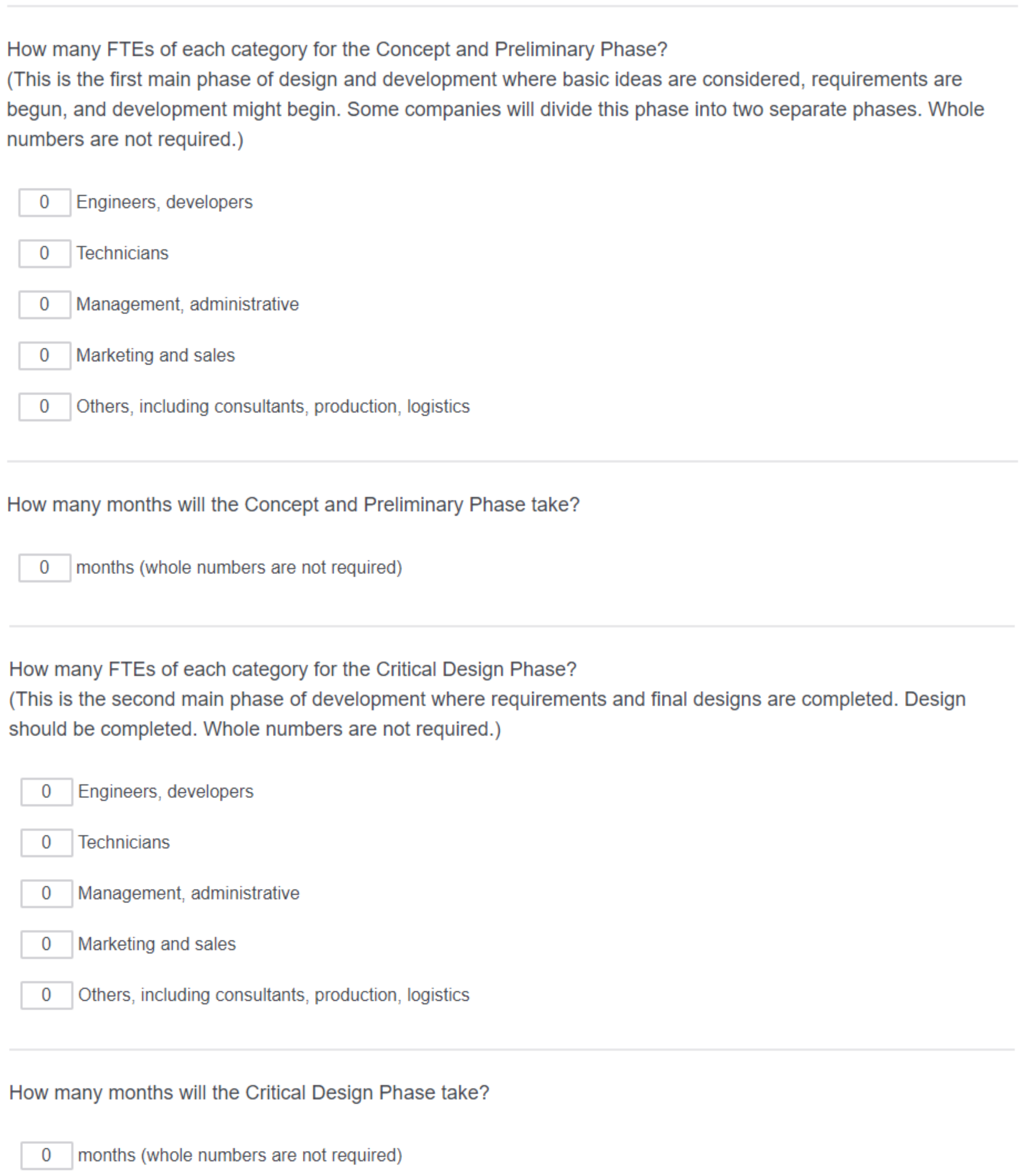
This project is a programmer used primarily by patients in physicians' offices. One of the staff tapes the programmer's antenna over the site of the implant, sets up a session on the programmer, and then hands the programmer to the patient. Patients follow simple instructions and answer questions posed on the screen; they draw outlines on body outlines where they feel either pain or stimulation effects. The system has a tablet with a mass of less than 0.5 kg, consumes less than 5 W power, and is similar in size to an iPad. It needs to survive for five years (ten years would be preferable) in daily use, five days a week.

The system is a new, clean-sheet design. The tablet must be either a custom design or available for more than five years from a vendor. The development team has never designed this type of system before. The system is a Class II, 510K device that requires premarket approval or PMA. In "FDA-speak" this means that it must be shown safe and effective through clinical trials before the FDA grants approval for distribution.

(**PLEASE NOTE:** This description of the project is intentionally sparse; we are studying the feasibility of estimation with very abbreviated descriptions.)

Estimates:

- ASSUME that your estimates target competent teams who will likely complete the work at least 9 times out every 10 projects.
- Fill in the boxes with your estimates for the number of persons and time required per development phase.
- Numbers of FTEs or months do not have to be whole numbers.
- Manufacturing personnel, assemblers, fabricators, and logistic support may consult, if so, include them in "Others"



How many FTEs of each category for the Concept and Preliminary Phase?
(This is the first main phase of design and development where basic ideas are considered, requirements are begun, and development might begin. Some companies will divide this phase into two separate phases. Whole numbers are not required.)

0 Engineers, developers

0 Technicians

0 Management, administrative

0 Marketing and sales

0 Others, including consultants, production, logistics

How many months will the Concept and Preliminary Phase take?

0 months (whole numbers are not required)

How many FTEs of each category for the Critical Design Phase?
(This is the second main phase of development where requirements and final designs are completed. Design should be completed. Whole numbers are not required.)

0 Engineers, developers

0 Technicians

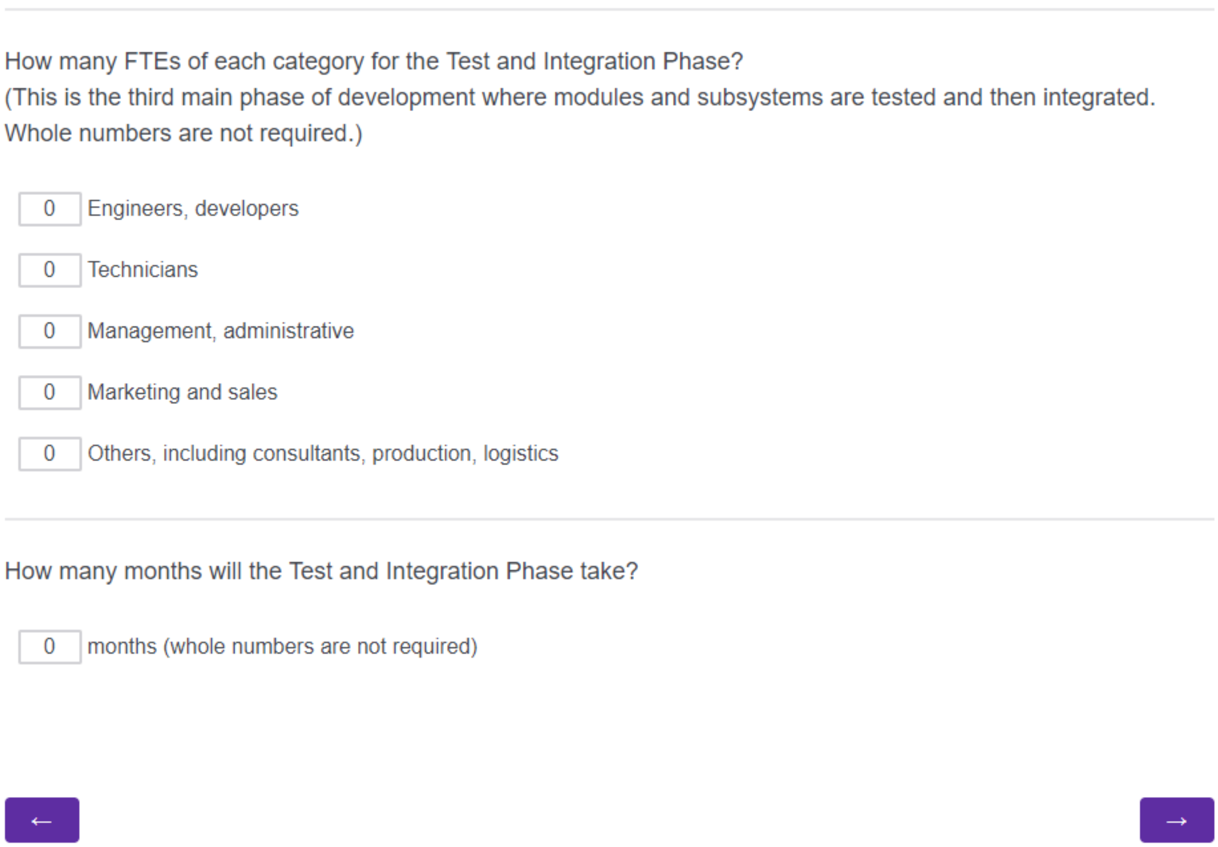
0 Management, administrative

0 Marketing and sales

0 Others, including consultants, production, logistics

How many months will the Critical Design Phase take?

0 months (whole numbers are not required)



How many FTEs of each category for the Test and Integration Phase?
(This is the third main phase of development where modules and subsystems are tested and then integrated. Whole numbers are not required.)

0 Engineers, developers

0 Technicians

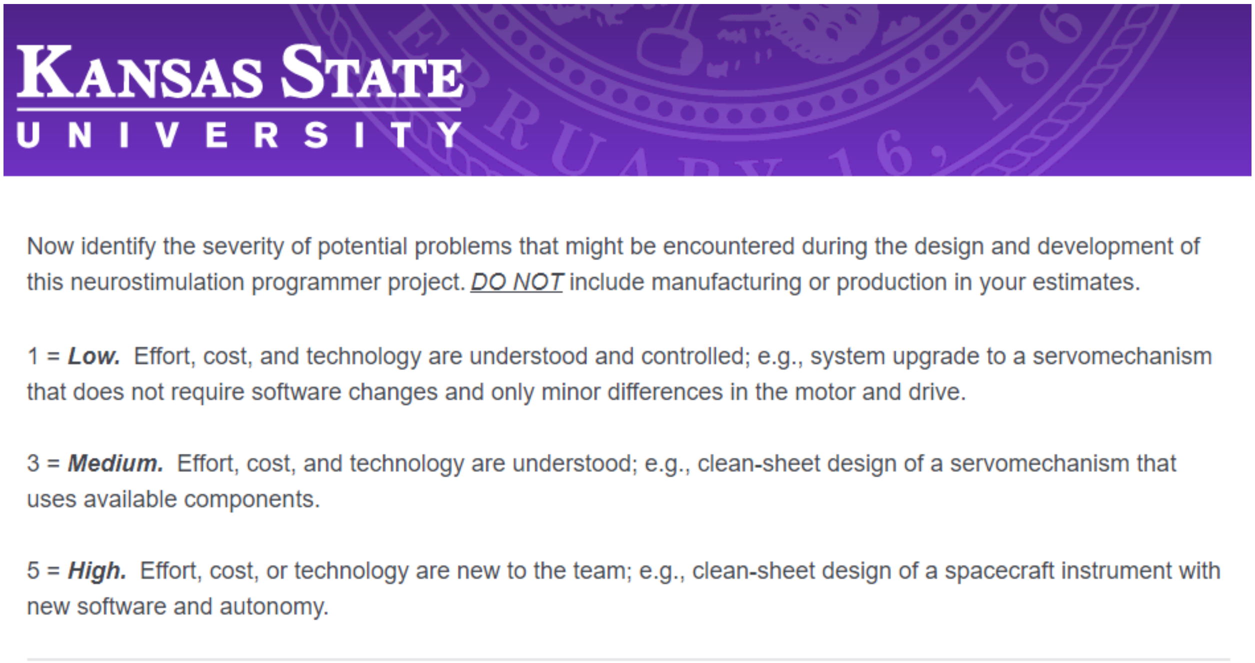
0 Management, administrative

0 Marketing and sales

0 Others, including consultants, production, logistics

How many months will the Test and Integration Phase take?

0 months (whole numbers are not required)

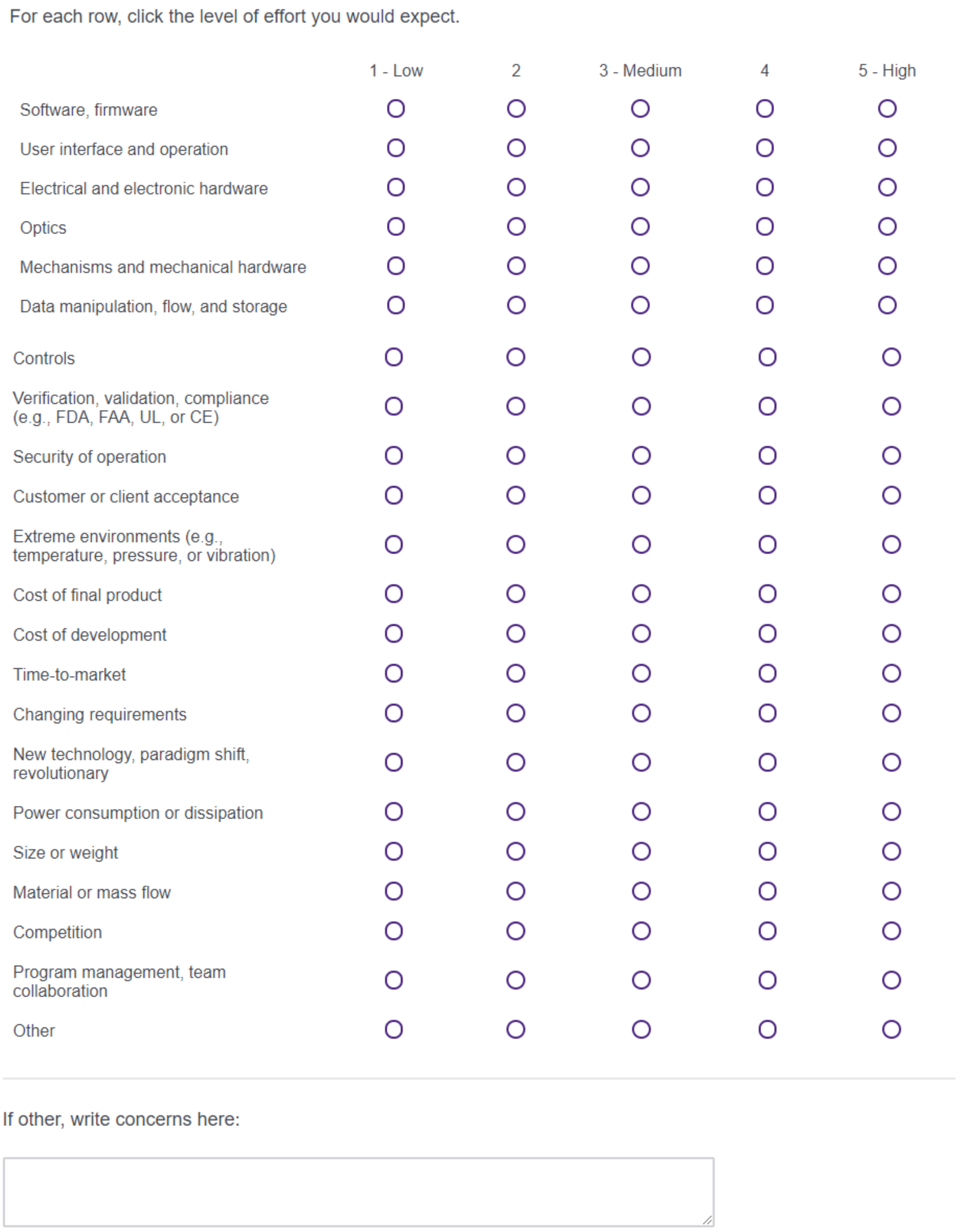


Now identify the severity of potential problems that might be encountered during the design and development of this neurostimulation programmer project. *DO NOT* include manufacturing or production in your estimates.

1 = ***Low.*** Effort, cost, and technology are understood and controlled; e.g., system upgrade to a servomechanism that does not require software changes and only minor differences in the motor and drive.

3 = ***Medium.*** Effort, cost, and technology are understood; e.g., clean-sheet design of a servomechanism that uses available components.

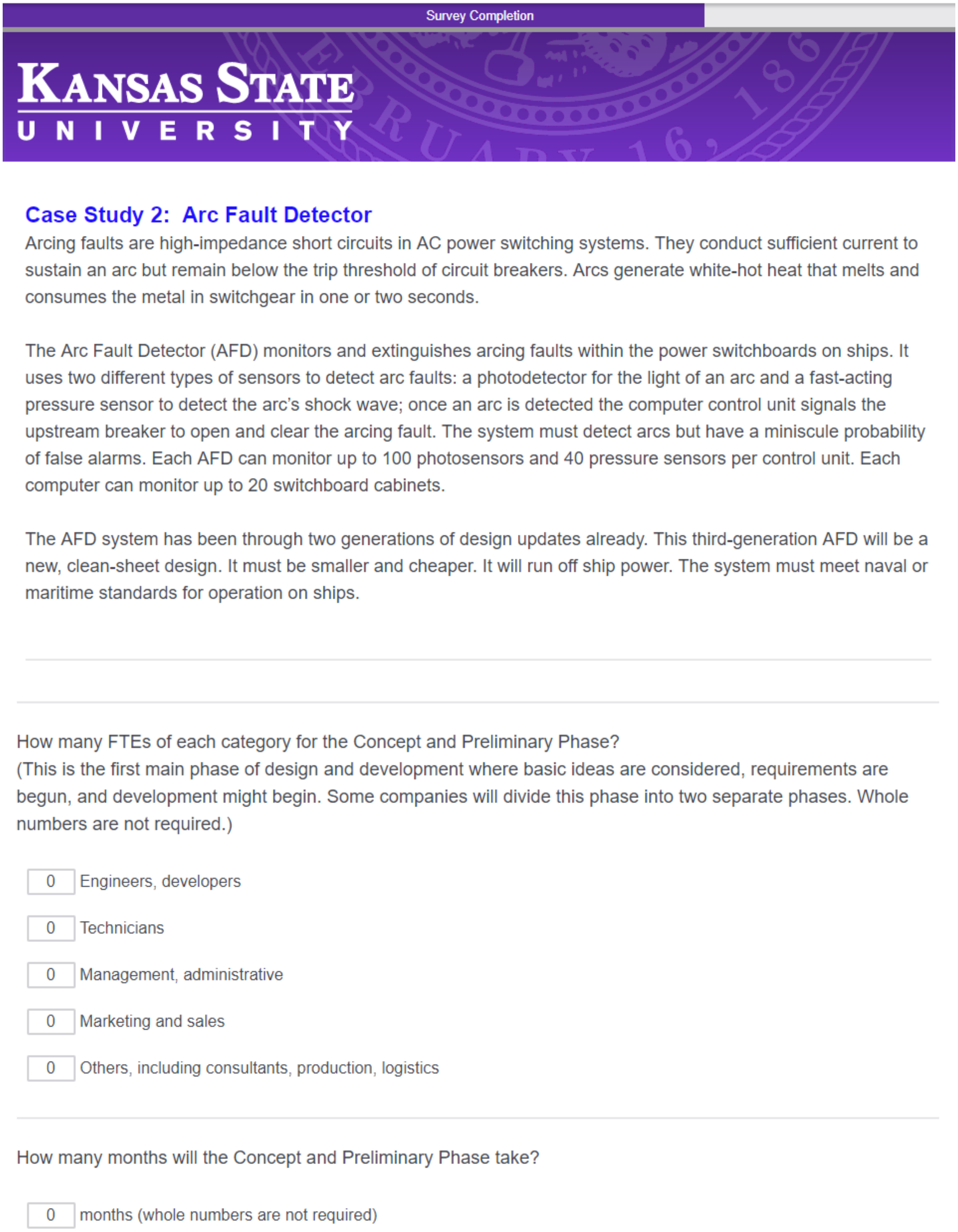
5 = ***High.*** Effort, cost, or technology are new to the team; e.g., clean-sheet design of a spacecraft instrument with new software and autonomy.



For each row, click the level of effort you would expect.

	1 - Low	2	3 - Medium	4	5 - High
Software, firmware	○	○	○	○	○
User interface and operation	○	○	○	○	○
Electrical and electronic hardware	○	○	○	○	○
Optics	○	○	○	○	○
Mechanisms and mechanical hardware	○	○	○	○	○
Data manipulation, flow, and storage	○	○	○	○	○
Controls	○	○	○	○	○
Verification, validation, compliance (e.g., FDA, FAA, UL, or CE)	○	○	○	○	○
Security of operation	○	○	○	○	○
Customer or client acceptance	○	○	○	○	○
Extreme environments (e.g., temperature, pressure, or vibration)	○	○	○	○	○
Cost of final product	○	○	○	○	○
Cost of development	○	○	○	○	○
Time-to-market	○	○	○	○	○
Changing requirements	○	○	○	○	○
New technology, paradigm shift, revolutionary	○	○	○	○	○
Power consumption or dissipation	○	○	○	○	○
Size or weight	○	○	○	○	○
Material or mass flow	○	○	○	○	○
Competition	○	○	○	○	○
Program management, team collaboration	○	○	○	○	○
Other	○	○	○	○	○

If other, write concerns here:



Case Study 2: Arc Fault Detector

Arcing faults are high-impedance short circuits in AC power switching systems. They conduct sufficient current to sustain an arc but remain below the trip threshold of circuit breakers. Arcs generate white-hot heat that melts and consumes the metal in switchgear in one or two seconds.

The Arc Fault Detector (AFD) monitors and extinguishes arcing faults within the power switchboards on ships. It uses two different types of sensors to detect arc faults: a photodetector for the light of an arc and a fast-acting pressure sensor to detect the arc's shock wave; once an arc is detected the computer control unit signals the upstream breaker to open and clear the arcing fault. The system must detect arcs but have a miniscule probability of false alarms. Each AFD can monitor up to 100 photosensors and 40 pressure sensors per control unit. Each computer can monitor up to 20 switchboard cabinets.

The AFD system has been through two generations of design updates already. This third-generation AFD will be a new, clean-sheet design. It must be smaller and cheaper. It will run off ship power. The system must meet naval or maritime standards for operation on ships.

How many FTEs of each category for the Concept and Preliminary Phase?
(This is the first main phase of design and development where basic ideas are considered, requirements are begun, and development might begin. Some companies will divide this phase into two separate phases. Whole numbers are not required.)

0 Engineers, developers

0 Technicians

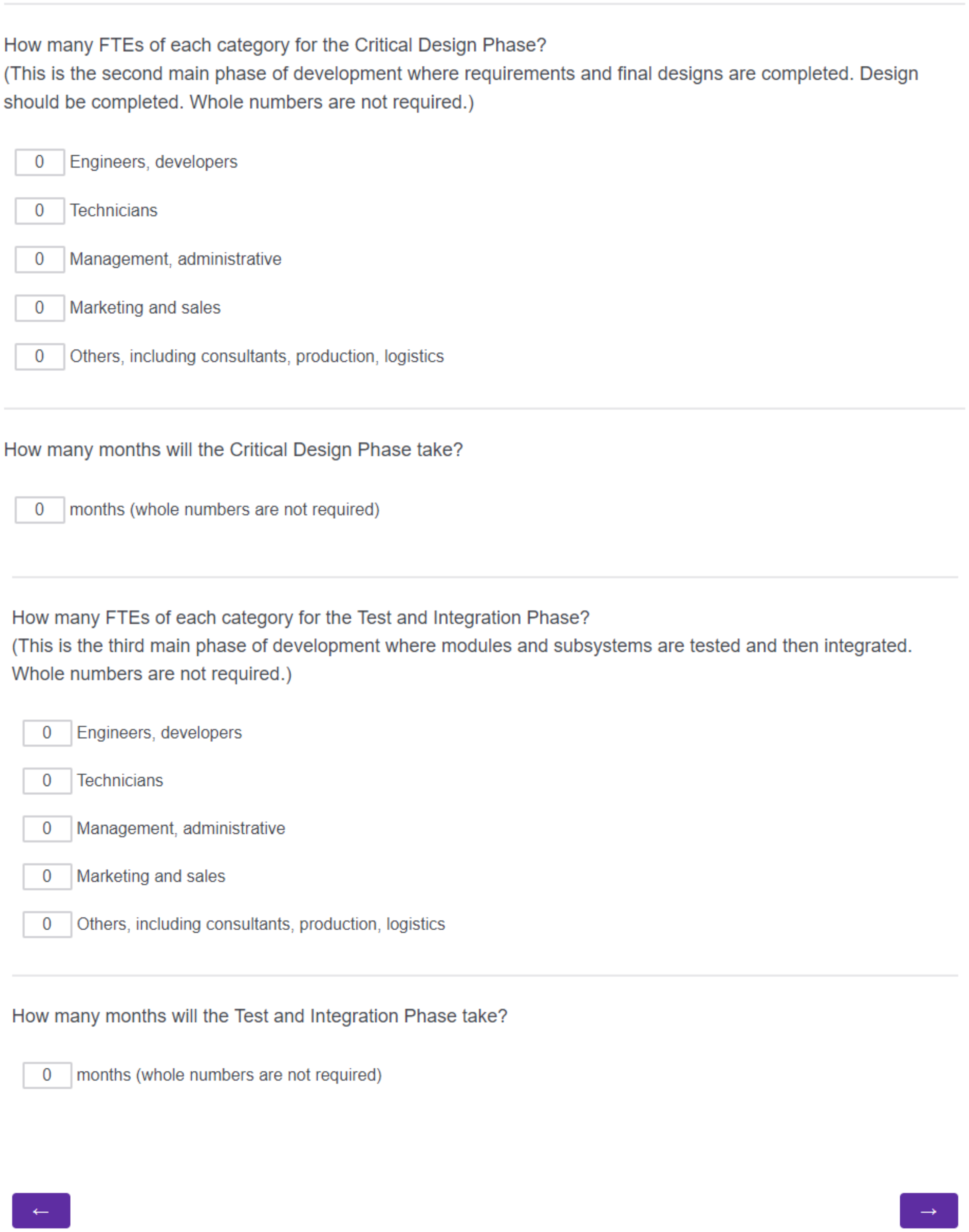
0 Management, administrative

0 Marketing and sales

0 Others, including consultants, production, logistics

How many months will the Concept and Preliminary Phase take?

0 months (whole numbers are not required)



How many FTEs of each category for the Critical Design Phase?
(This is the second main phase of development where requirements and final designs are completed. Design should be completed. Whole numbers are not required.)

0 Engineers, developers

0 Technicians

0 Management, administrative

0 Marketing and sales

0 Others, including consultants, production, logistics

How many months will the Critical Design Phase take?

0 months (whole numbers are not required)

How many FTEs of each category for the Test and Integration Phase?
(This is the third main phase of development where modules and subsystems are tested and then integrated. Whole numbers are not required.)

0 Engineers, developers

0 Technicians

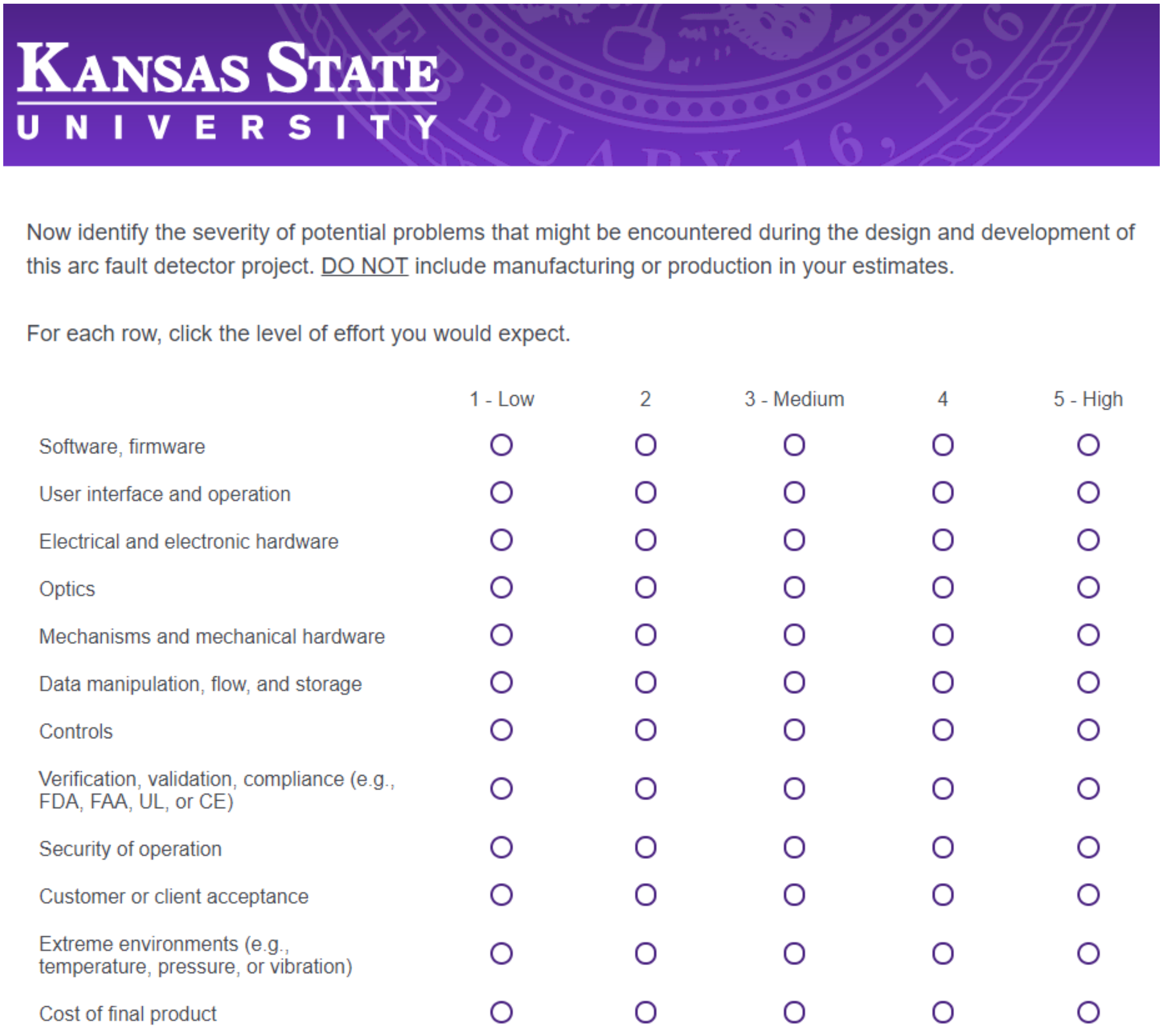
0 Management, administrative

0 Marketing and sales

0 Others, including consultants, production, logistics

How many months will the Test and Integration Phase take?

0 months (whole numbers are not required)

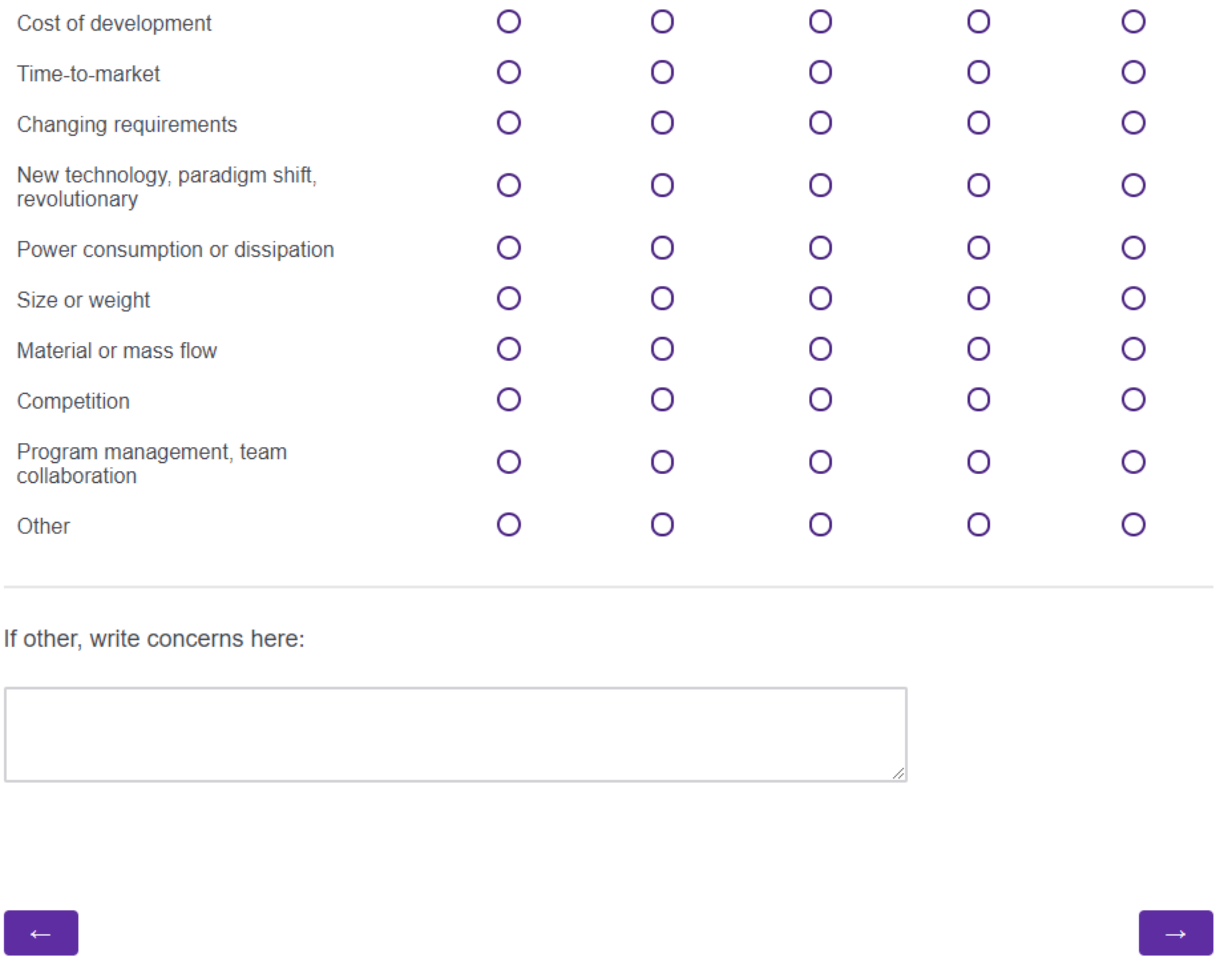


Now identify the severity of potential problems that might be encountered during the design and development of this arc fault detector project. DO NOT include manufacturing or production in your estimates.

For each row, click the level of effort you would expect.

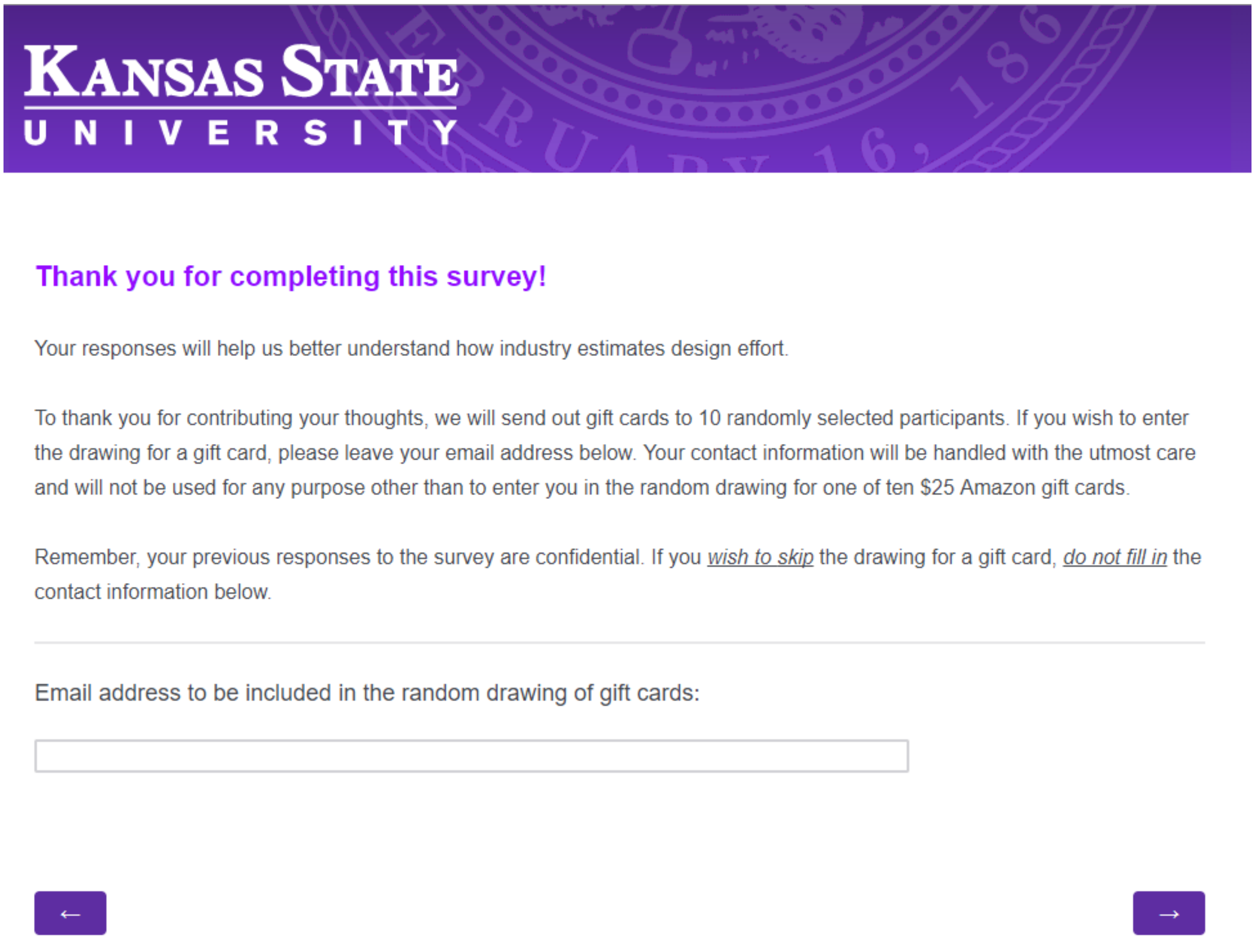
	1 - Low	2	3 - Medium	4	5 - High
Software, firmware	○	○	○	○	○
User interface and operation	○	○	○	○	○
Electrical and electronic hardware	○	○	○	○	○
Optics	○	○	○	○	○
Mechanisms and mechanical hardware	○	○	○	○	○
Data manipulation, flow, and storage	○	○	○	○	○
Controls	○	○	○	○	○
Verification, validation, compliance (e.g., FDA, FAA, UL, or CE)	○	○	○	○	○
Security of operation	○	○	○	○	○
Customer or client acceptance	○	○	○	○	○
Extreme environments (e.g., temperature, pressure, or vibration)	○	○	○	○	○
Cost of final product	○	○	○	○	○

(continued on next page – single page in survey)



Cost of development	○	○	○	○	○
Time-to-market	○	○	○	○	○
Changing requirements	○	○	○	○	○
New technology, paradigm shift, revolutionary	○	○	○	○	○
Power consumption or dissipation	○	○	○	○	○
Size or weight	○	○	○	○	○
Material or mass flow	○	○	○	○	○
Competition	○	○	○	○	○
Program management, team collaboration	○	○	○	○	○
Other	○	○	○	○	○

If other, write concerns here:



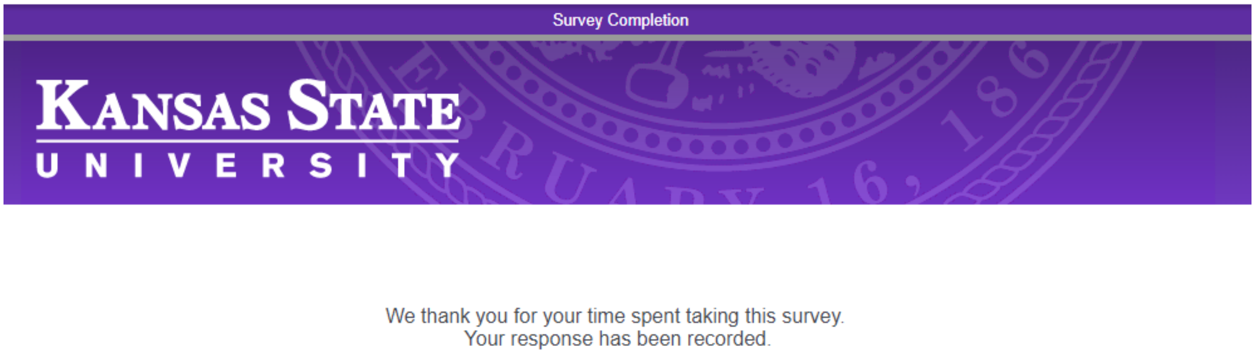
Thank you for completing this survey!

Your responses will help us better understand how industry estimates design effort.

To thank you for contributing your thoughts, we will send out gift cards to 10 randomly selected participants. If you wish to enter the drawing for a gift card, please leave your email address below. Your contact information will be handled with the utmost care and will not be used for any purpose other than to enter you in the random drawing for one of ten $25 Amazon gift cards.

Remember, your previous responses to the survey are confidential. If you *wish to skip* the drawing for a gift card, *do not fill in* the contact information below.

Email address to be included in the random drawing of gift cards:

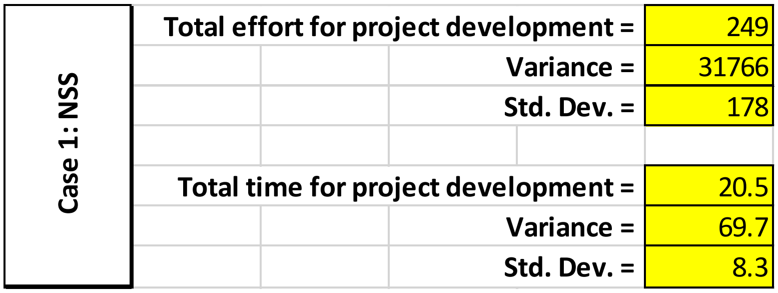


We thank you for your time spent taking this survey.
Your response has been recorded.

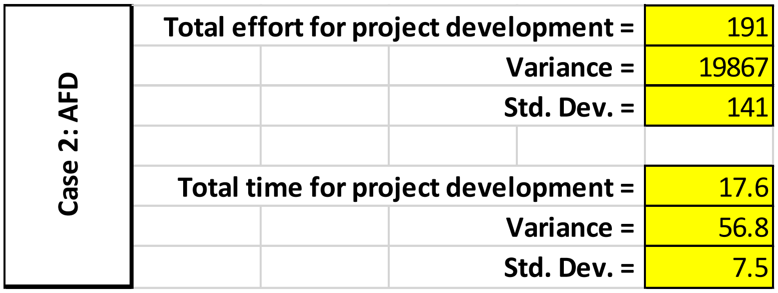
Raw Survey Results

Originally, I thought that the general population might provide a baseline to compare against more technically-oriented people. The variance was so large to prove unusable. Consequently, I am only showing results from 146 technically-oriented respondents.

The raw results were as follows:



Case 1: NSS	Total effort for project development =	249
	Variance =	31766
	Std. Dev. =	178
	Total time for project development =	20.5
	Variance =	69.7
	Std. Dev. =	8.3

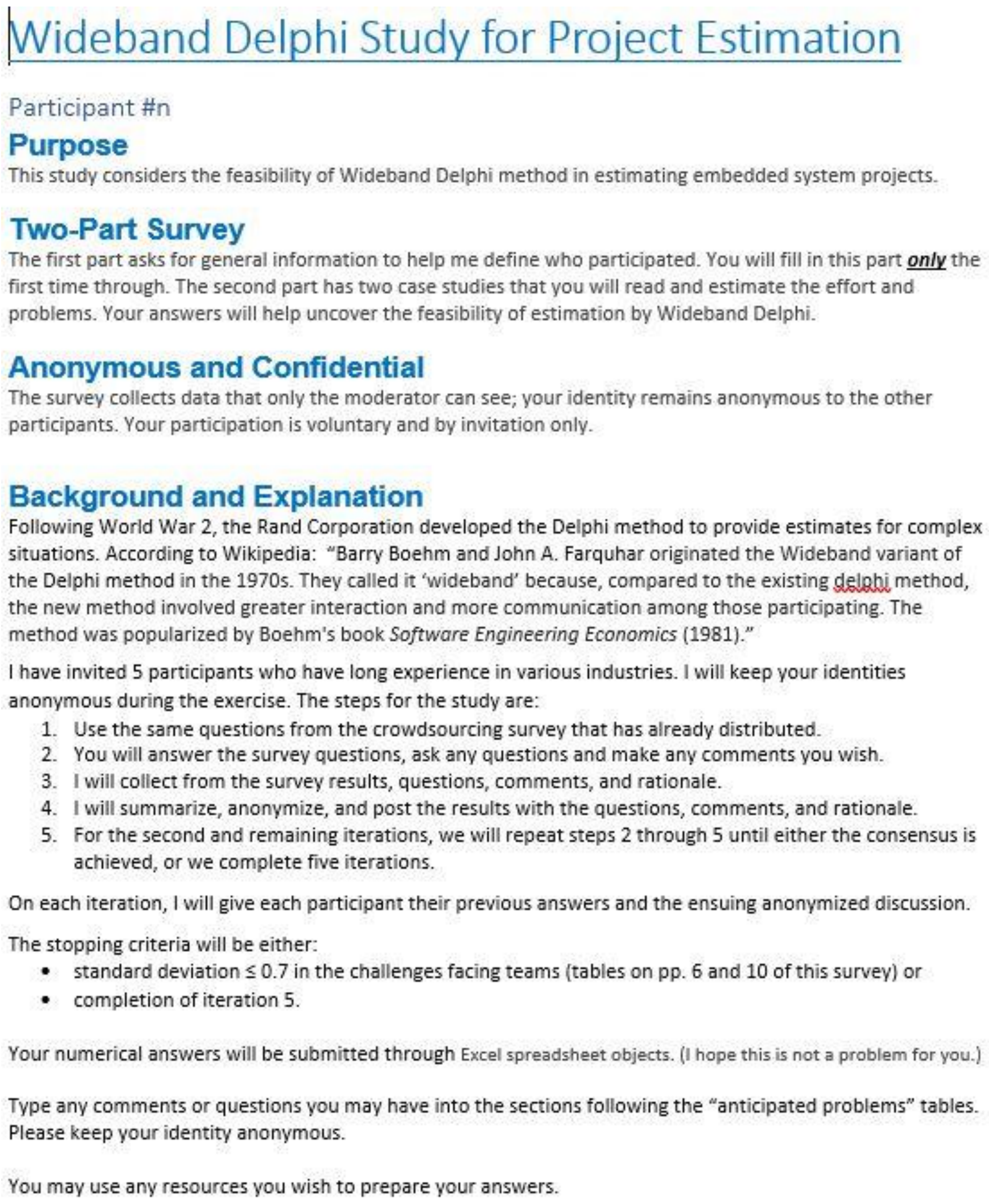


Case 2: AFD	Total effort for project development =	191
	Variance =	19867
	Std. Dev. =	141
	Total time for project development =	17.6
	Variance =	56.8
	Std. Dev. =	7.5

Appendix K - The Delphi Method Study

Study Format

I ran the Delphi method study through 4 iterations. The stopping criteria was that the standard deviations fell within the acceptance criteria. The first iteration used the survey displayed on the next 11 pages. The following iterations dispensed with the instructions and the request for biographical information on the next two pages.



Wideband Delphi Study for Project Estimation

Participant #n

Purpose

This study considers the feasibility of Wideband Delphi method in estimating embedded system projects.

Two-Part Survey

The first part asks for general information to help me define who participated. You will fill in this part ***only*** the first time through. The second part has two case studies that you will read and estimate the effort and problems. Your answers will help uncover the feasibility of estimation by Wideband Delphi.

Anonymous and Confidential

The survey collects data that only the moderator can see; your identity remains anonymous to the other participants. Your participation is voluntary and by invitation only.

Background and Explanation

Following World War 2, the Rand Corporation developed the Delphi method to provide estimates for complex situations. According to Wikipedia: "Barry Boehm and John A. Farquhar originated the Wideband variant of the Delphi method in the 1970s. They called it 'wideband' because, compared to the existing delphi method, the new method involved greater interaction and more communication among those participating. The method was popularized by Boehm's book *Software Engineering Economics* (1981)."

I have invited 5 participants who have long experience in various industries. I will keep your identities anonymous during the exercise. The steps for the study are:

1. Use the same questions from the crowdsourcing survey that has already distributed.
2. You will answer the survey questions, ask any questions and make any comments you wish.
3. I will collect from the survey results, questions, comments, and rationale.
4. I will summarize, anonymize, and post the results with the questions, comments, and rationale.
5. For the second and remaining iterations, we will repeat steps 2 through 5 until either the consensus is achieved, or we complete five iterations.

On each iteration, I will give each participant their previous answers and the ensuing anonymized discussion.

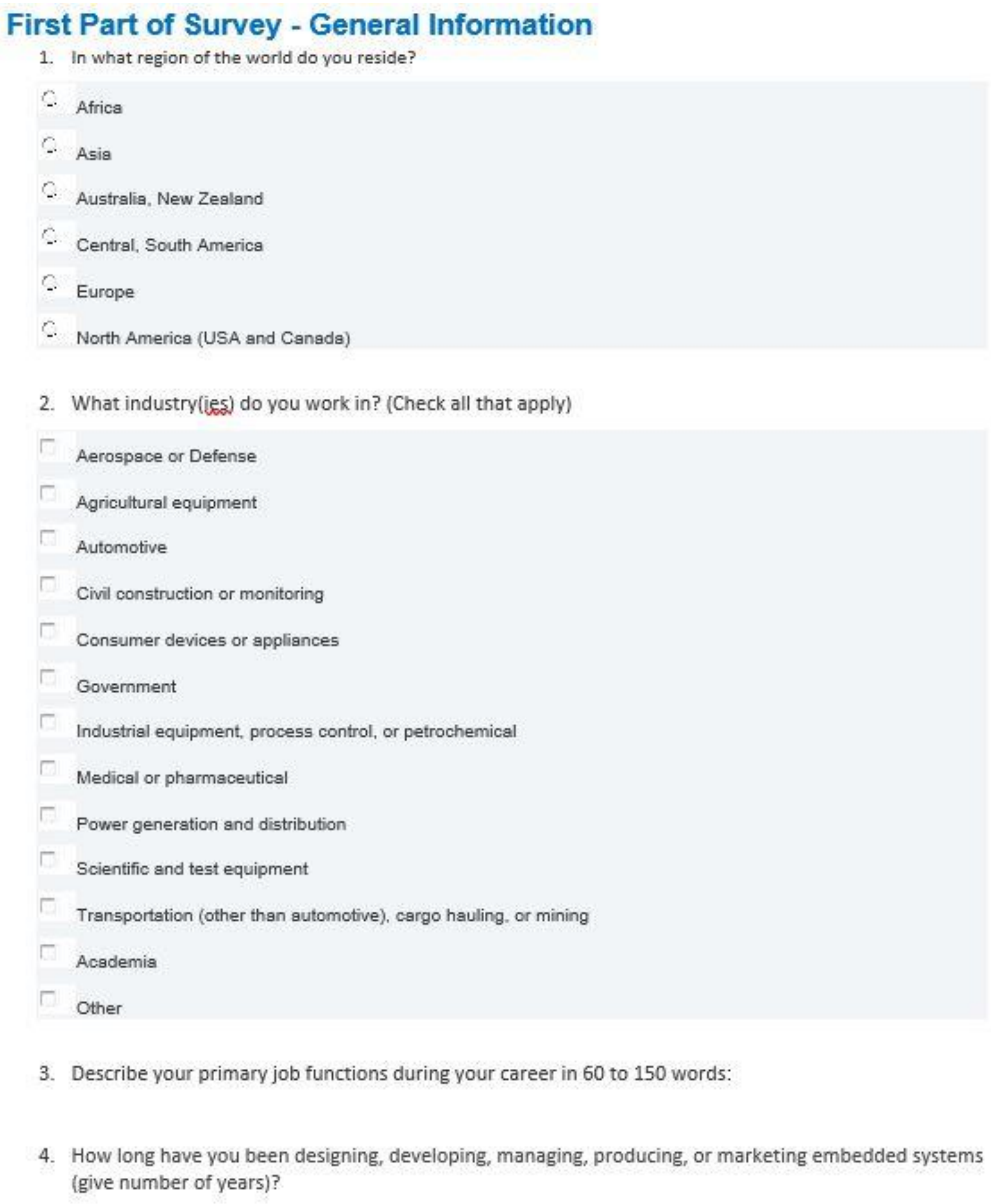
The stopping criteria will be either:

- standard deviation ≤ 0.7 in the challenges facing teams (tables on pp. 6 and 10 of this survey) or
- completion of iteration 5.

Your numerical answers will be submitted through Excel spreadsheet objects. (I hope this is not a problem for you.)

Type any comments or questions you may have into the sections following the "anticipated problems" tables. Please keep your identity anonymous.

You may use any resources you wish to prepare your answers.



First Part of Survey - General Information

1. In what region of the world do you reside?

○ Africa

○ Asia

○ Australia, New Zealand

○ Central, South America

○ Europe

○ North America (USA and Canada)

2. What industry(ies) do you work in? (Check all that apply)

□ Aerospace or Defense

□ Agricultural equipment

□ Automotive

□ Civil construction or monitoring

□ Consumer devices or appliances

□ Government

□ Industrial equipment, process control, or petrochemical

□ Medical or pharmaceutical

□ Power generation and distribution

□ Scientific and test equipment

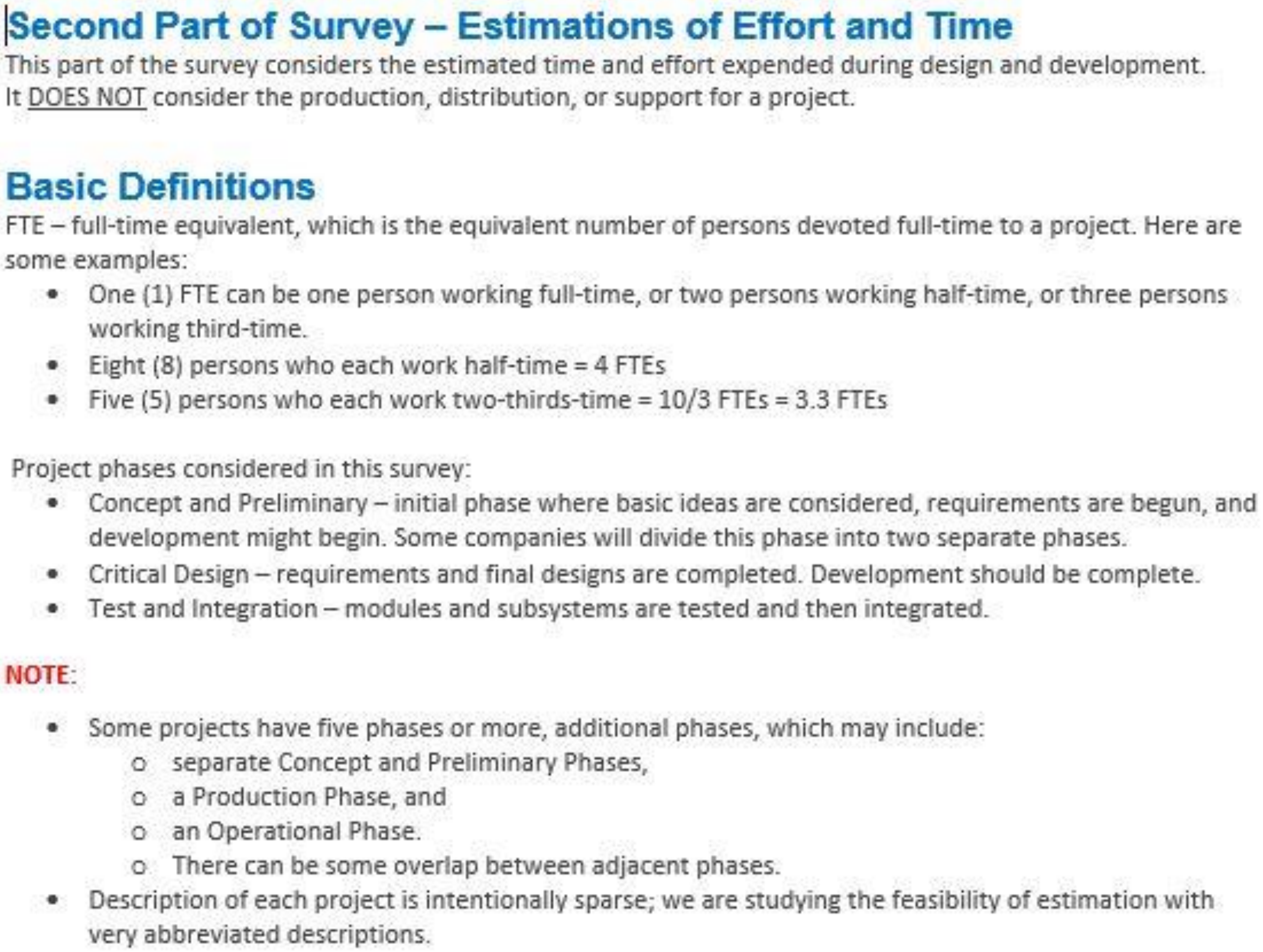
□ Transportation (other than automotive), cargo hauling, or mining

□ Academia

□ Other

3. Describe your primary job functions during your career in 60 to 150 words:

4. How long have you been designing, developing, managing, producing, or marketing embedded systems (give number of years)?



Second Part of Survey – Estimations of Effort and Time

This part of the survey considers the estimated time and effort expended during design and development.
It DOES NOT consider the production, distribution, or support for a project.

Basic Definitions

FTE – full-time equivalent, which is the equivalent number of persons devoted full-time to a project. Here are some examples:

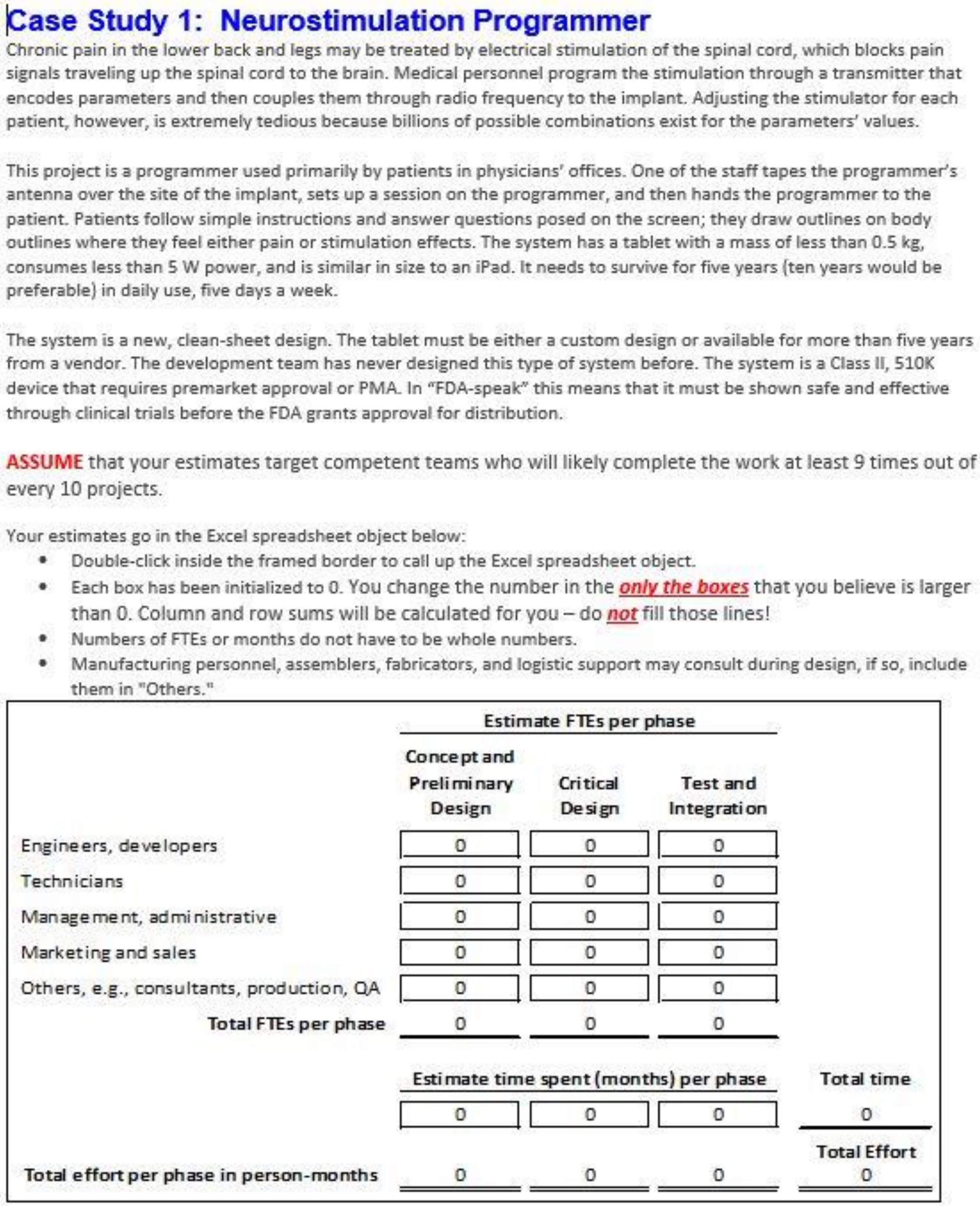
- One (1) FTE can be one person working full-time, or two persons working half-time, or three persons working third-time.
- Eight (8) persons who each work half-time = 4 FTEs
- Five (5) persons who each work two-thirds-time = 10/3 FTEs = 3.3 FTEs

Project phases considered in this survey:

- Concept and Preliminary – initial phase where basic ideas are considered, requirements are begun, and development might begin. Some companies will divide this phase into two separate phases.
- Critical Design – requirements and final designs are completed. Development should be complete.
- Test and Integration – modules and subsystems are tested and then integrated.

NOTE:

- Some projects have five phases or more, additional phases, which may include:
  - separate Concept and Preliminary Phases,
  - a Production Phase, and
  - an Operational Phase.
  - There can be some overlap between adjacent phases.
- Description of each project is intentionally sparse; we are studying the feasibility of estimation with very abbreviated descriptions.



Case Study 1: Neurostimulation Programmer

Chronic pain in the lower back and legs may be treated by electrical stimulation of the spinal cord, which blocks pain signals traveling up the spinal cord to the brain. Medical personnel program the stimulation through a transmitter that encodes parameters and then couples them through radio frequency to the implant. Adjusting the stimulator for each patient, however, is extremely tedious because billions of possible combinations exist for the parameters' values.

This project is a programmer used primarily by patients in physicians' offices. One of the staff tapes the programmer's antenna over the site of the implant, sets up a session on the programmer, and then hands the programmer to the patient. Patients follow simple instructions and answer questions posed on the screen; they draw outlines on body outlines where they feel either pain or stimulation effects. The system has a tablet with a mass of less than 0.5 kg, consumes less than 5 W power, and is similar in size to an iPad. It needs to survive for five years (ten years would be preferable) in daily use, five days a week.

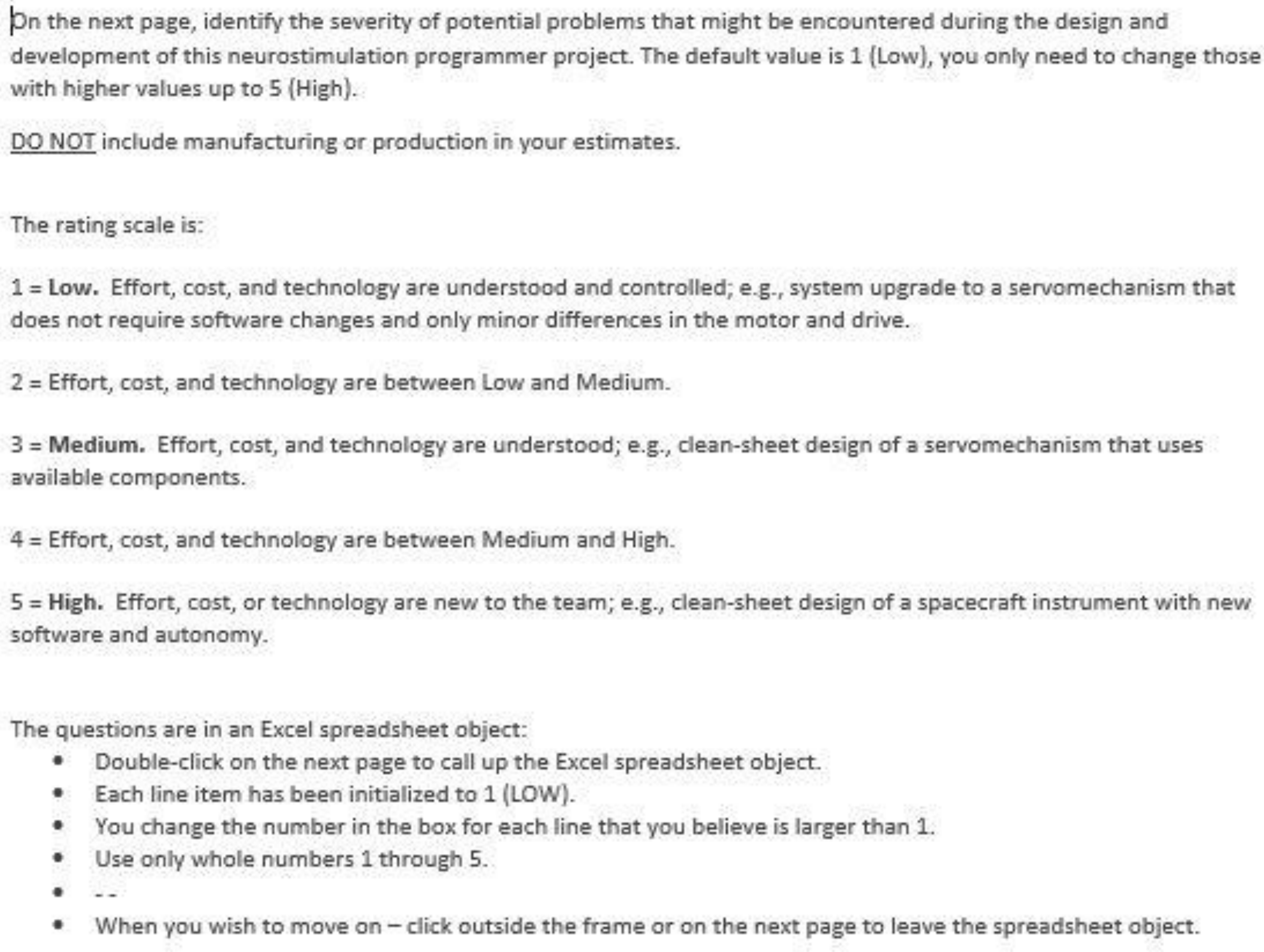
The system is a new, clean-sheet design. The tablet must be either a custom design or available for more than five years from a vendor. The development team has never designed this type of system before. The system is a Class II, 510K device that requires premarket approval or PMA. In "FDA-speak" this means that it must be shown safe and effective through clinical trials before the FDA grants approval for distribution.

ASSUME that your estimates target competent teams who will likely complete the work at least 9 times out of every 10 projects.

Your estimates go in the Excel spreadsheet object below:

- Double-click inside the framed border to call up the Excel spreadsheet object.
- Each box has been initialized to 0. You change the number in the ***only the boxes*** that you believe is larger than 0. Column and row sums will be calculated for you – do ***not*** fill those lines!
- Numbers of FTEs or months do not have to be whole numbers.
- Manufacturing personnel, assemblers, fabricators, and logistic support may consult during design, if so, include them in "Others."

	Estimate FTEs per phase			
	Concept and Preliminary Design	**Critical Design**	**Test and Integration**	
Engineers, developers	0	0	0	
Technicians	0	0	0	
Management, administrative	0	0	0	
Marketing and sales	0	0	0	
Others, e.g., consultants, production, QA	0	0	0	
Total FTEs per phase	0	0	0	
	Estimate time spent (months) per phase			**Total time**
	0	0	0	0
				Total Effort
Total effort per phase in person-months	0	0	0	0



On the next page, identify the severity of potential problems that might be encountered during the design and development of this neurostimulation programmer project. The default value is 1 (Low), you only need to change those with higher values up to 5 (High).

DO NOT include manufacturing or production in your estimates.

The rating scale is:

1 = **Low.** Effort, cost, and technology are understood and controlled; e.g., system upgrade to a servomechanism that does not require software changes and only minor differences in the motor and drive.

2 = Effort, cost, and technology are between Low and Medium.

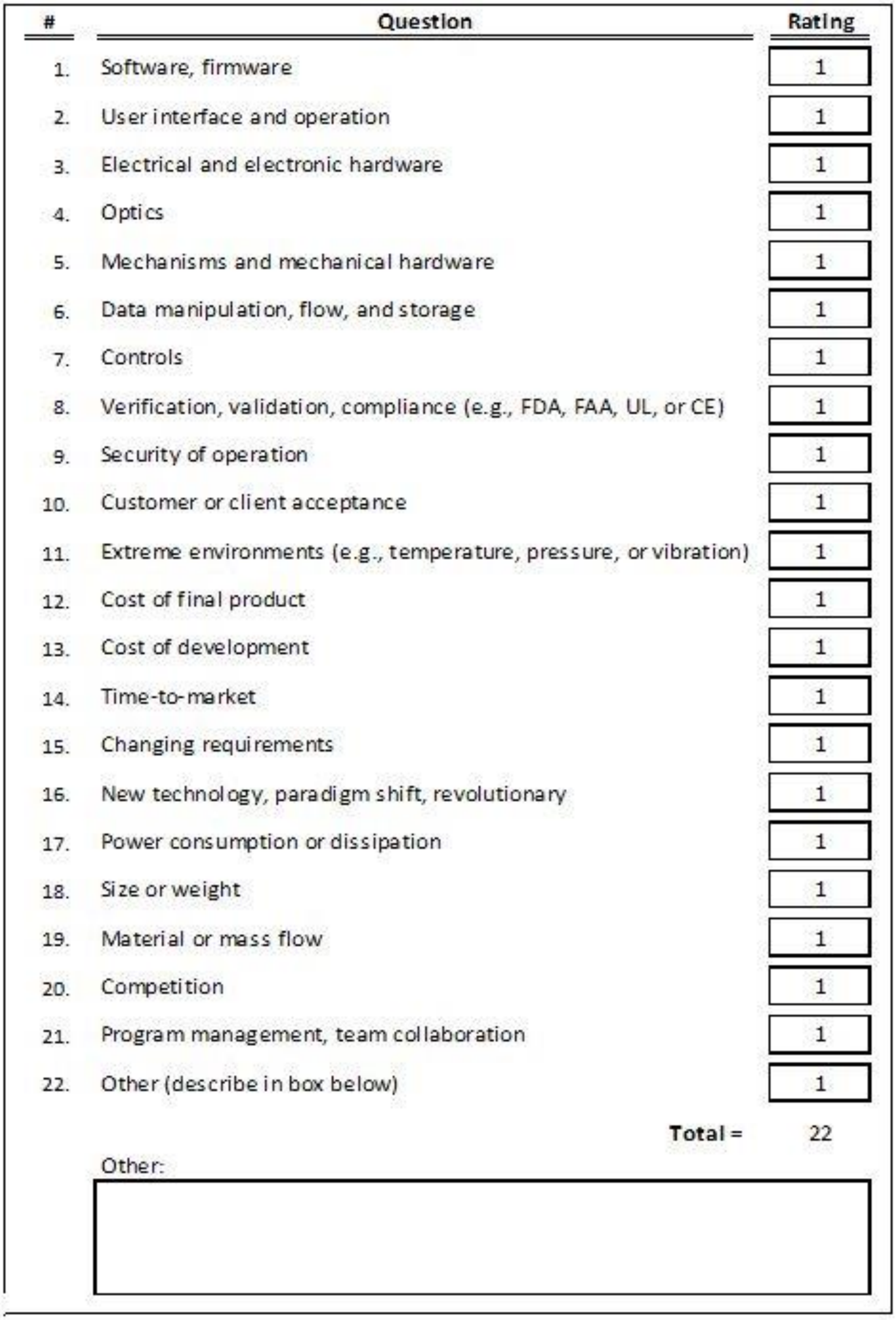
3 = **Medium.** Effort, cost, and technology are understood; e.g., clean-sheet design of a servomechanism that uses available components.

4 = Effort, cost, and technology are between Medium and High.

5 = **High.** Effort, cost, or technology are new to the team; e.g., clean-sheet design of a spacecraft instrument with new software and autonomy.

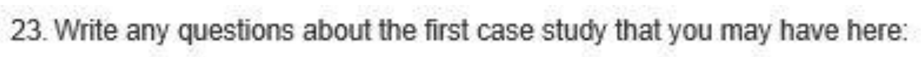
The questions are in an Excel spreadsheet object:

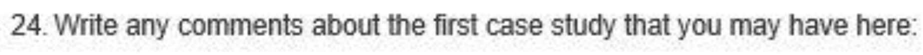
- Double-click on the next page to call up the Excel spreadsheet object.
- Each line item has been initialized to 1 (LOW).
- You change the number in the box for each line that you believe is larger than 1.
- Use only whole numbers 1 through 5.
- - -
- When you wish to move on – click outside the frame or on the next page to leave the spreadsheet object.

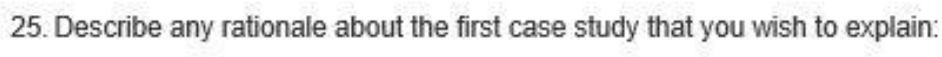


#	Question	Rating
1.	Software, firmware	1
2.	User interface and operation	1
3.	Electrical and electronic hardware	1
4.	Optics	1
5.	Mechanisms and mechanical hardware	1
6.	Data manipulation, flow, and storage	1
7.	Controls	1
8.	Verification, validation, compliance (e.g., FDA, FAA, UL, or CE)	1
9.	Security of operation	1
10.	Customer or client acceptance	1
11.	Extreme environments (e.g., temperature, pressure, or vibration)	1
12.	Cost of final product	1
13.	Cost of development	1
14.	Time-to-market	1
15.	Changing requirements	1
16.	New technology, paradigm shift, revolutionary	1
17.	Power consumption or dissipation	1
18.	Size or weight	1
19.	Material or mass flow	1
20.	Competition	1
21.	Program management, team collaboration	1
22.	Other (describe in box below)	1
	Total =	22

Other:


23. Write any questions about the first case study that you may have here:


24. Write any comments about the first case study that you may have here:


25. Describe any rationale about the first case study that you wish to explain:

Case Study 2: Arc Fault Detector

Arcing faults are high-impedance short circuits in AC power switching systems. They conduct sufficient current to sustain an arc but remain below the trip threshold of circuit breakers. Arcs generate white-hot heat that melts and consumes the metal in switchgear in one or two seconds.

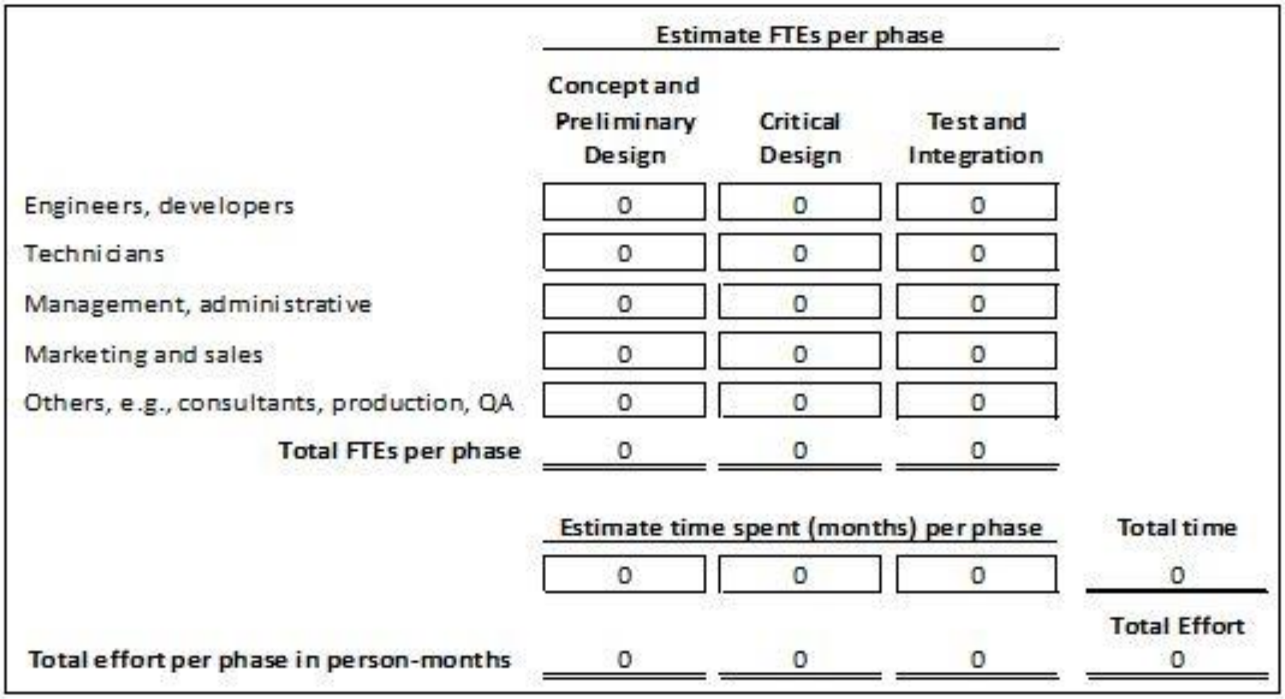
The Arc Fault Detector (AFD) monitors and extinguishes arcing faults within the power switchboards on ships. It uses two different types of sensors to detect arc faults: a photodetector for the light of an arc and a fast-acting pressure sensor to detect the arc's shock wave; once an arc is detected the computer control unit signals the upstream breaker to open and clear the arcing fault. The system must detect arcs but have a miniscule probability of false alarms. Each AFD can monitor up to 100 photosensors and 40 pressure sensors per control unit. Each computer can monitor up to 20 switchboard cabinets.

The AFD system has been through two generations of design updates already. This third-generation AFD will be a new, clean-sheet design. It must be smaller and cheaper. It will run off ship power. The system must meet naval or maritime standards for operation on ships.

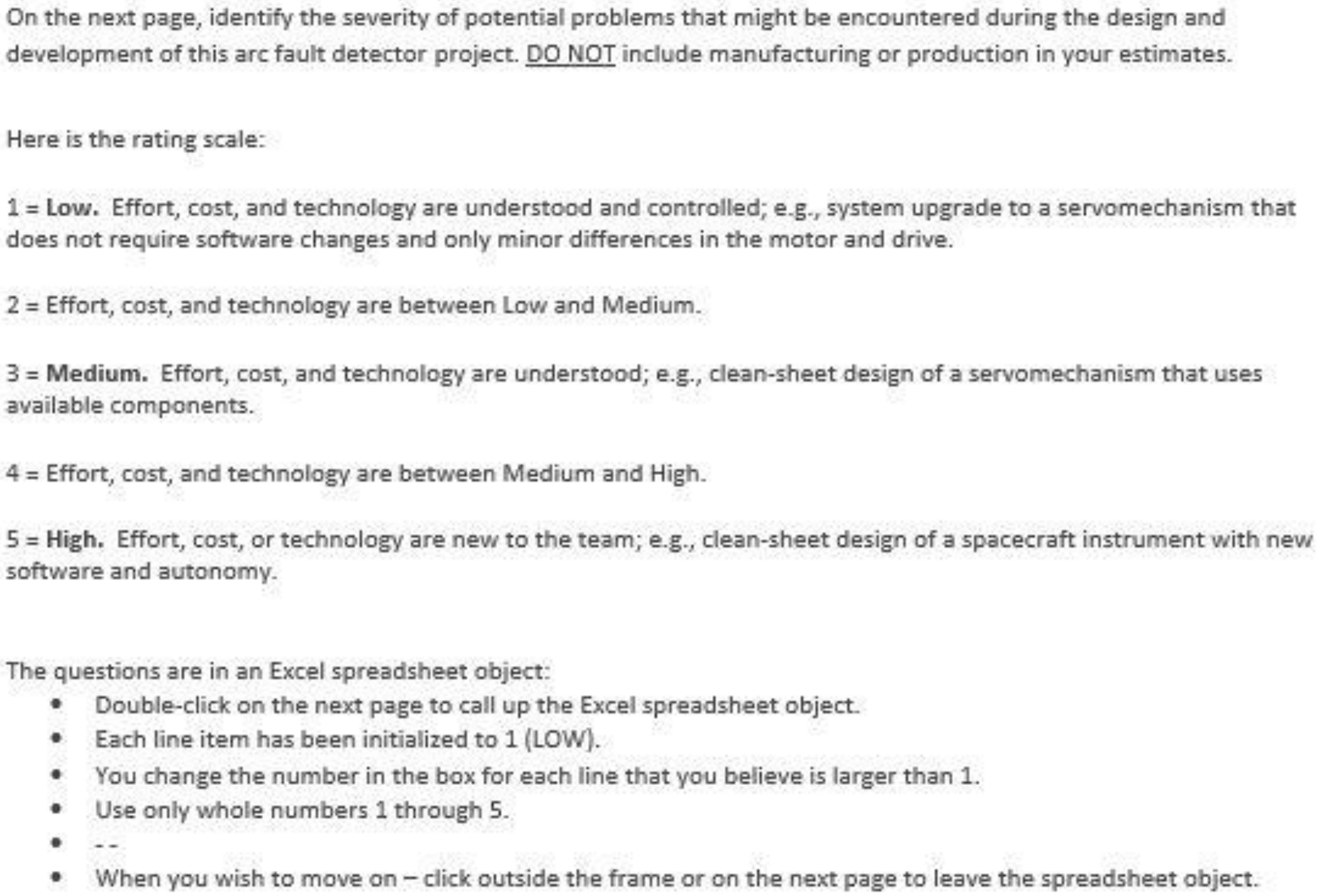
ASSUME that your estimates target competent teams who will likely complete the work at least 9 times out of every 10 projects.

Your estimates go in the Excel spreadsheet object below:

- Double-click inside the framed border to call up the Excel spreadsheet object.
- Each box has been initialized to 0. You change the number in the ***only the boxes*** that you believe is larger than 0. Column and row sums will be calculated for you – do ***not*** fill those lines!
- Numbers of FTEs or months do not have to be whole numbers.
- Manufacturing personnel, assemblers, fabricators, and logistic support may consult during design, if so, include them in "Others."



	Estimate FTEs per phase			
	Concept and Preliminary Design	Critical Design	Test and Integration	
Engineers, developers	0	0	0	
Technicians	0	0	0	
Management, administrative	0	0	0	
Marketing and sales	0	0	0	
Others, e.g., consultants, production, QA	0	0	0	
Total FTEs per phase	0	0	0	
	Estimate time spent (months) per phase			Total time
	0	0	0	0
				Total Effort
Total effort per phase in person-months	0	0	0	0



On the next page, identify the severity of potential problems that might be encountered during the design and development of this arc fault detector project. DO NOT include manufacturing or production in your estimates.

Here is the rating scale:

1 = **Low.** Effort, cost, and technology are understood and controlled; e.g., system upgrade to a servomechanism that does not require software changes and only minor differences in the motor and drive.

2 = Effort, cost, and technology are between Low and Medium.

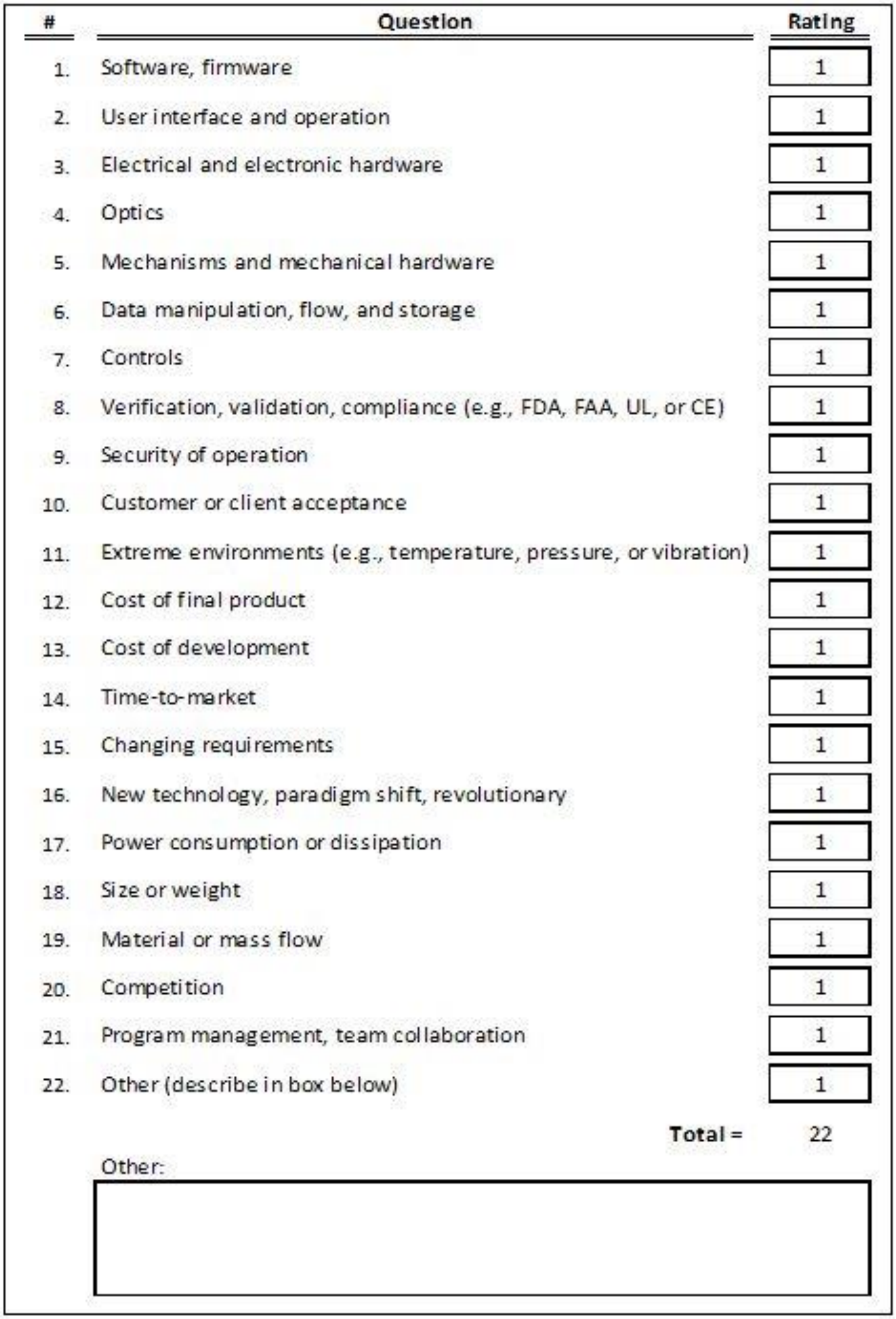
3 = **Medium.** Effort, cost, and technology are understood; e.g., clean-sheet design of a servomechanism that uses available components.

4 = Effort, cost, and technology are between Medium and High.

5 = **High.** Effort, cost, or technology are new to the team; e.g., clean-sheet design of a spacecraft instrument with new software and autonomy.

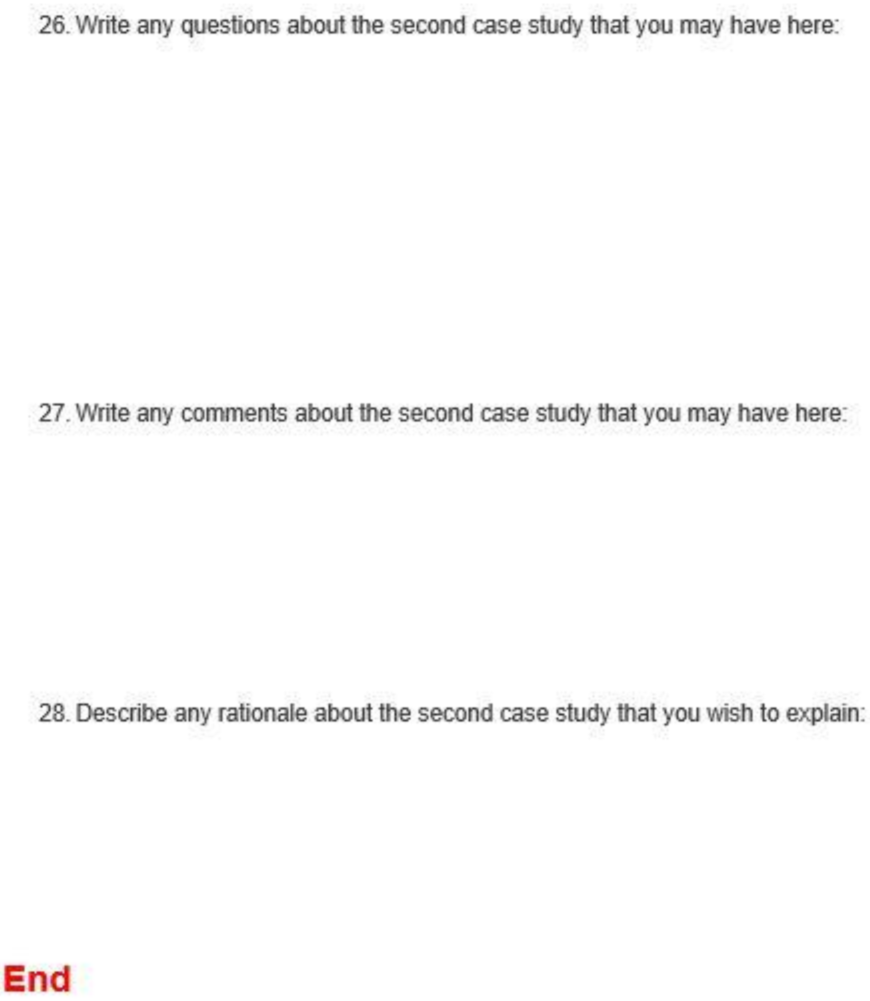
The questions are in an Excel spreadsheet object:

- Double-click on the next page to call up the Excel spreadsheet object.
- Each line item has been initialized to 1 (LOW).
- You change the number in the box for each line that you believe is larger than 1.
- Use only whole numbers 1 through 5.
- --
- When you wish to move on – click outside the frame or on the next page to leave the spreadsheet object.



#	Question	Rating
1.	Software, firmware	1
2.	User interface and operation	1
3.	Electrical and electronic hardware	1
4.	Optics	1
5.	Mechanisms and mechanical hardware	1
6.	Data manipulation, flow, and storage	1
7.	Controls	1
8.	Verification, validation, compliance (e.g., FDA, FAA, UL, or CE)	1
9.	Security of operation	1
10.	Customer or client acceptance	1
11.	Extreme environments (e.g., temperature, pressure, or vibration)	1
12.	Cost of final product	1
13.	Cost of development	1
14.	Time-to-market	1
15.	Changing requirements	1
16.	New technology, paradigm shift, revolutionary	1
17.	Power consumption or dissipation	1
18.	Size or weight	1
19.	Material or mass flow	1
20.	Competition	1
21.	Program management, team collaboration	1
22.	Other (describe in box below)	1
	Total =	22

Other:



26. Write any questions about the second case study that you may have here:

27. Write any comments about the second case study that you may have here:

28. Describe any rationale about the second case study that you wish to explain:

End

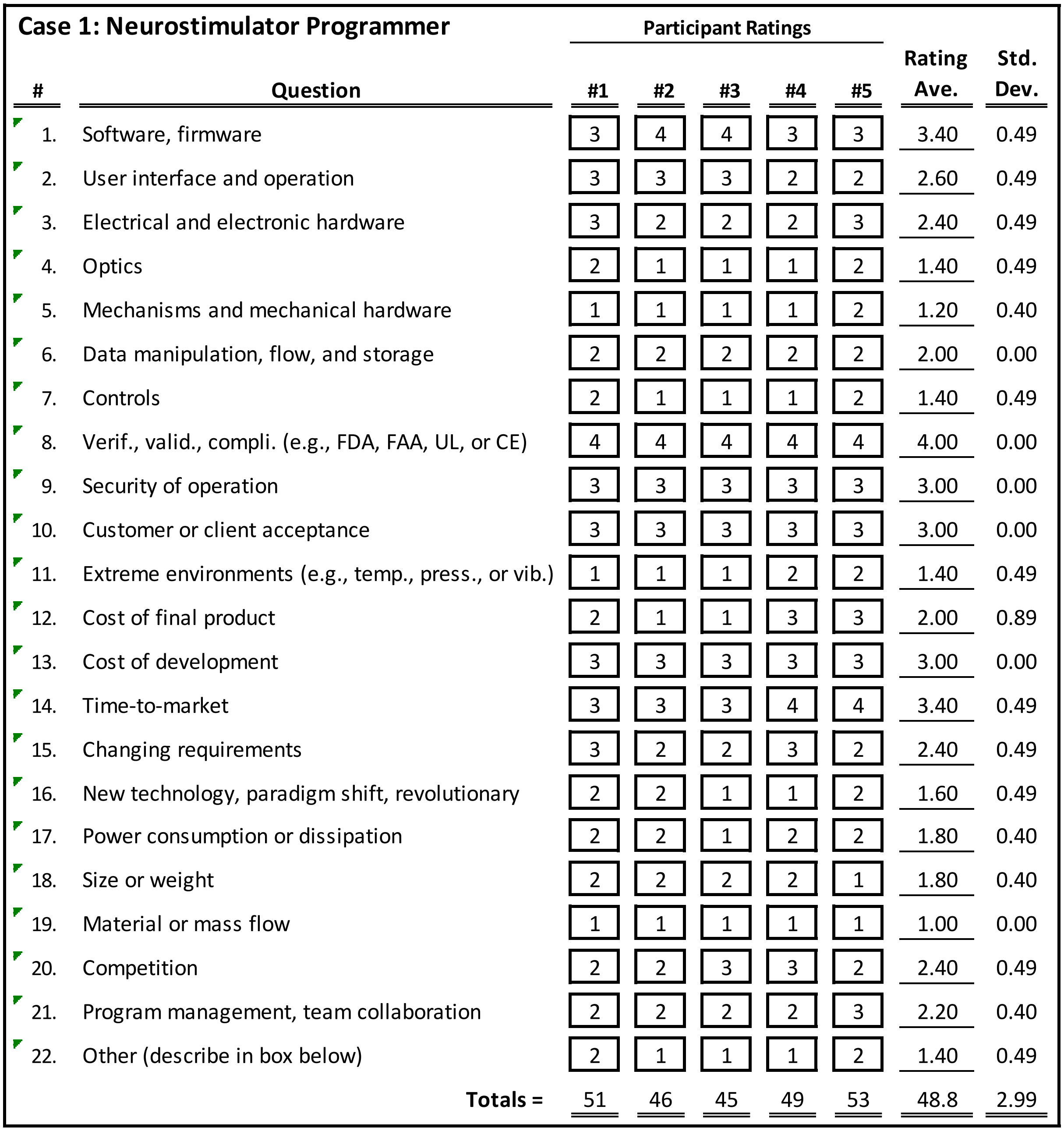
Participants

I sought out very experienced engineers with wide-ranging experience over their respective careers. The table that follows has short biographies of the Delphi Study participants. NOTE: Aus. = Australia, U.S. = United States.

#	Name	Region	years	Industries involved	Career summary
1	Geoff Patch	Aus.	28	Aerospace or Defense	Software Engineer. Worked as team member, project team leader, then Software Engineering Department Manager. Specialised in maritime radar signal processing, primarily signal extraction from noise, target detection and tracking for civilian and military applications including missile fire control. Currently report to CEO, working on enhancing corporate cyber-security measures and information assurance.
2	Dwight Clayton	U.S.	34	Government	Working with various sponsors (mainly federal agencies and commercial companies), I identify, design, develop, and deploy advanced unique instrumentation and controls systems that addresses issues important to the United States and to the world. Over my career, I have been responsible for concept development, hardware design, system design, software design (including embedded systems), project management, and testing of deployed systems.
3	Max Cortner	U.S.	40	Aerospace or Defense, Medical or pharmaceutical, Scientific and test equipment	Design, implement and validate automated test systems for integrated circuits and printed circuit board assemblies. Manage projects involving engineers who developed, implemented and validated automated test systems used to verify the performance of manufactured medical devices to critical requirements. Designed, documented, and executed experiments to verify compliance of a medical device design to both internal and external requirements.
4	Mike Gard	U.S.	35	Aerospace or Defense, Industrial equipment, process control, or petrochemical, Medical or pharmaceutical, Scientific and test equipment	test engineer, design engineer, biomedical (clinical) engineer, engineering manager, senior R&D engineer, senior systems engineer, R&D project manager, adjunct associate professor
5	Tim Cooper	U.S.	30	Aerospace or Defense, Consumer devices or appliances, Industrial equipment, process control, or petrochemical, Transportation (other than automotive), cargo hauling, or mining	Design, development, and test of real-time embedded control and signal processing systems; integration and test of software on hardware platforms; support development of hardware platforms through construction of embedded test software.

Raw Study Results – Technical Challenges

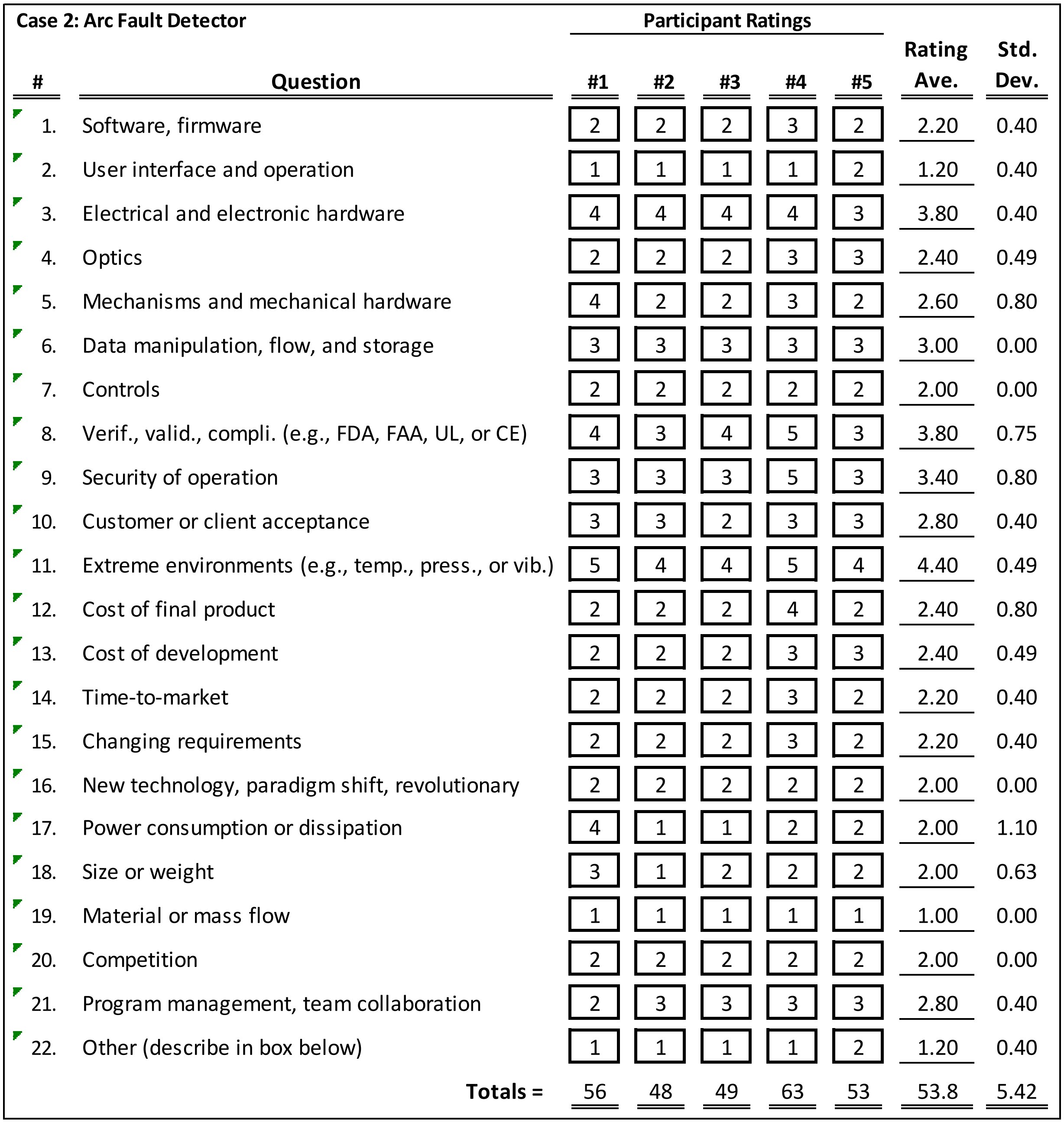
Technical challenges perceived by the Delphi participants for Case Study 1.



Case 1: Neurostimulator Programmer

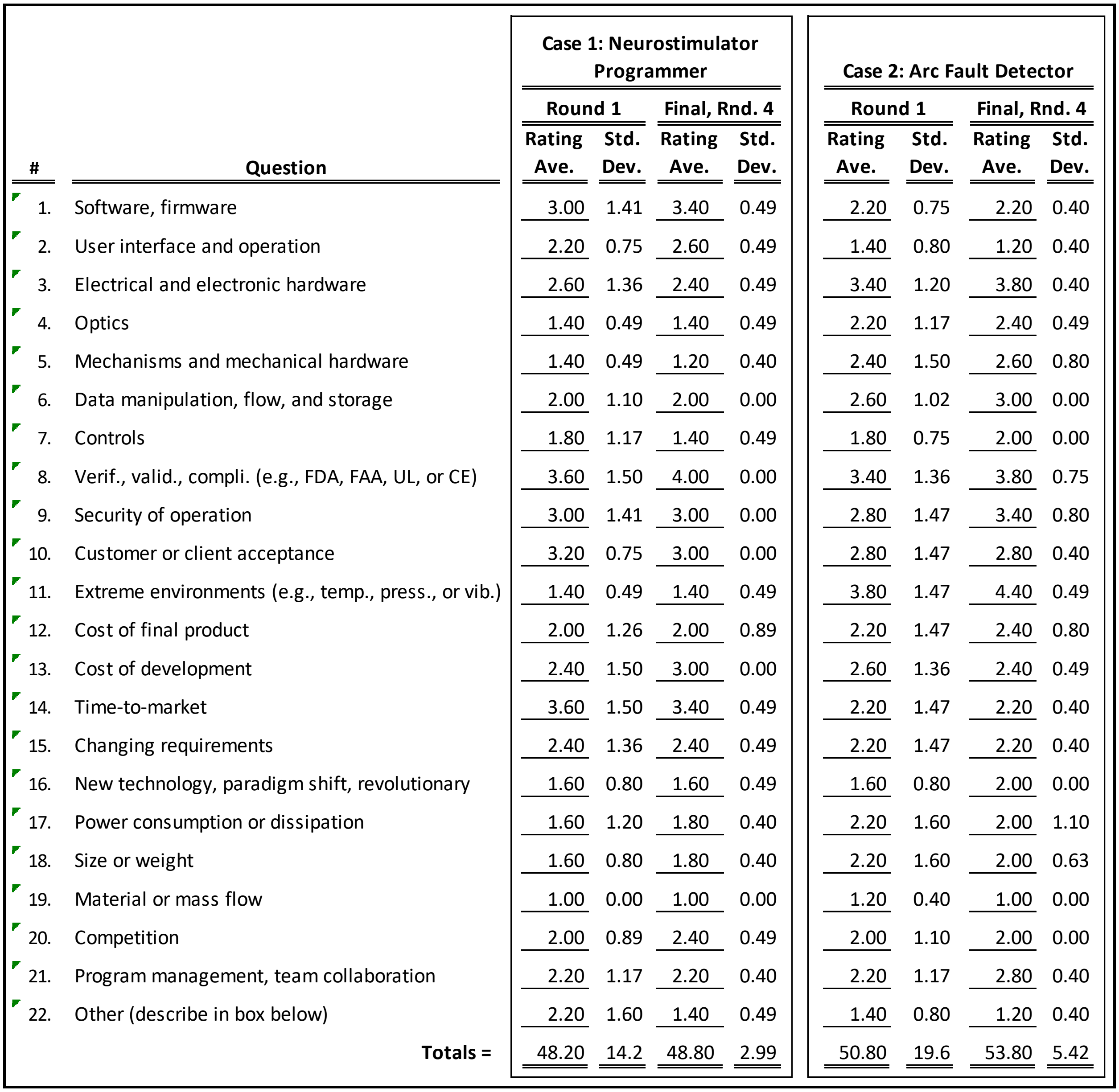
		Participant Ratings						
#	**Question**	**#1**	**#2**	**#3**	**#4**	**#5**	**Rating Ave.**	**Std. Dev.**
1.	Software, firmware	3	4	4	3	3	3.40	0.49
2.	User interface and operation	3	3	3	2	2	2.60	0.49
3.	Electrical and electronic hardware	3	2	2	2	3	2.40	0.49
4.	Optics	2	1	1	1	2	1.40	0.49
5.	Mechanisms and mechanical hardware	1	1	1	1	2	1.20	0.40
6.	Data manipulation, flow, and storage	2	2	2	2	2	2.00	0.00
7.	Controls	2	1	1	1	2	1.40	0.49
8.	Verif., valid., compli. (e.g., FDA, FAA, UL, or CE)	4	4	4	4	4	4.00	0.00
9.	Security of operation	3	3	3	3	3	3.00	0.00
10.	Customer or client acceptance	3	3	3	3	3	3.00	0.00
11.	Extreme environments (e.g., temp., press., or vib.)	1	1	1	2	2	1.40	0.49
12.	Cost of final product	2	1	1	3	3	2.00	0.89
13.	Cost of development	3	3	3	3	3	3.00	0.00
14.	Time-to-market	3	3	3	4	4	3.40	0.49
15.	Changing requirements	3	2	2	3	2	2.40	0.49
16.	New technology, paradigm shift, revolutionary	2	2	1	1	2	1.60	0.49
17.	Power consumption or dissipation	2	2	1	2	2	1.80	0.40
18.	Size or weight	2	2	2	2	1	1.80	0.40
19.	Material or mass flow	1	1	1	1	1	1.00	0.00
20.	Competition	2	2	3	3	2	2.40	0.49
21.	Program management, team collaboration	2	2	2	2	3	2.20	0.40
22.	Other (describe in box below)	2	1	1	1	2	1.40	0.49
	Totals =	51	46	45	49	53	48.8	2.99

Technical challenges perceived by the Delphi participants for Case Study 2.



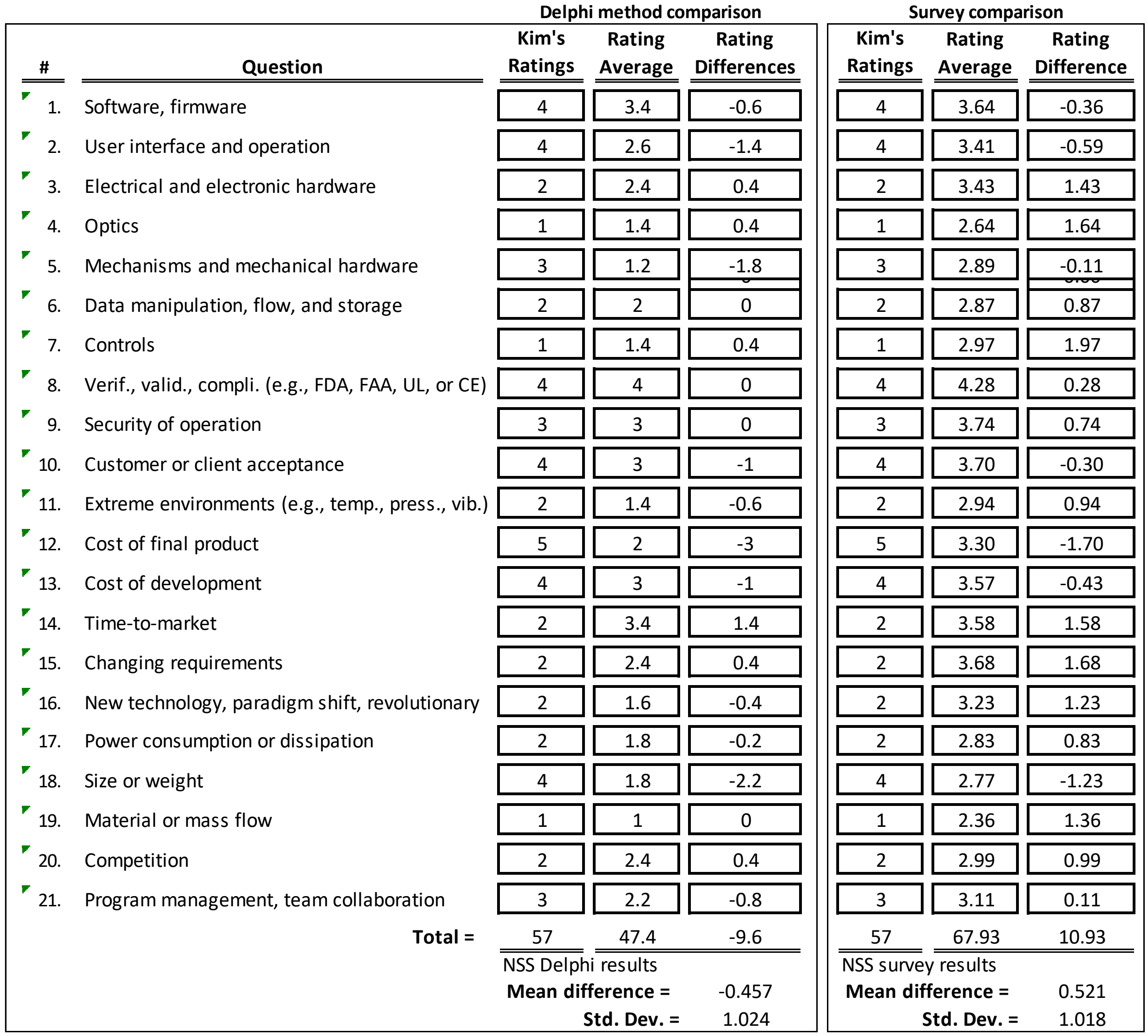
Case 2: Arc Fault Detector		Participant Ratings						
							Rating	Std.
#	Question	#1	#2	#3	#4	#5	Ave.	Dev.
1.	Software, firmware	2	2	2	3	2	2.20	0.40
2.	User interface and operation	1	1	1	1	2	1.20	0.40
3.	Electrical and electronic hardware	4	4	4	4	3	3.80	0.40
4.	Optics	2	2	2	3	3	2.40	0.49
5.	Mechanisms and mechanical hardware	4	2	2	3	2	2.60	0.80
6.	Data manipulation, flow, and storage	3	3	3	3	3	3.00	0.00
7.	Controls	2	2	2	2	2	2.00	0.00
8.	Verif., valid., compli. (e.g., FDA, FAA, UL, or CE)	4	3	4	5	3	3.80	0.75
9.	Security of operation	3	3	3	5	3	3.40	0.80
10.	Customer or client acceptance	3	3	2	3	3	2.80	0.40
11.	Extreme environments (e.g., temp., press., or vib.)	5	4	4	5	4	4.40	0.49
12.	Cost of final product	2	2	2	4	2	2.40	0.80
13.	Cost of development	2	2	2	3	3	2.40	0.49
14.	Time-to-market	2	2	2	3	2	2.20	0.40
15.	Changing requirements	2	2	2	3	2	2.20	0.40
16.	New technology, paradigm shift, revolutionary	2	2	2	2	2	2.00	0.00
17.	Power consumption or dissipation	4	1	1	2	2	2.00	1.10
18.	Size or weight	3	1	2	2	2	2.00	0.63
19.	Material or mass flow	1	1	1	1	1	1.00	0.00
20.	Competition	2	2	2	2	2	2.00	0.00
21.	Program management, team collaboration	2	3	3	3	3	2.80	0.40
22.	Other (describe in box below)	1	1	1	1	2	1.20	0.40
	Totals =	56	48	49	63	53	53.8	5.42

Summary of how the Delphi participants changed their perceptions of technical challenges for both case studies, from Round 1 to Round 4.



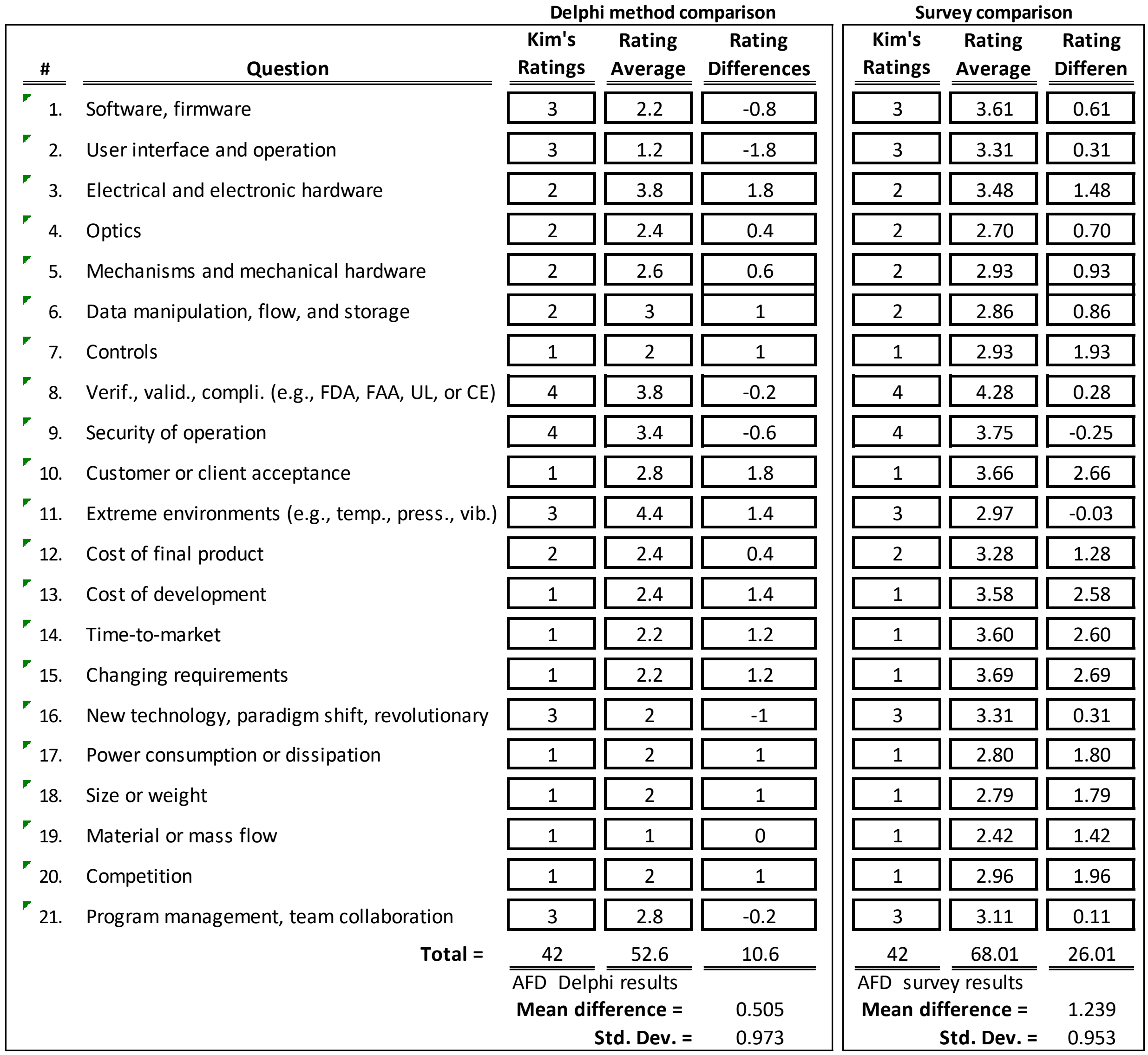
		Case 1: Neurostimulator Programmer				Case 2: Arc Fault Detector			
		Round 1		Final, Rnd. 4		Round 1		Final, Rnd. 4	
#	Question	Rating Ave.	Std. Dev.	Rating Ave.	Std. Dev.	Rating Ave.	Std. Dev.	Rating Ave.	Std. Dev.
1.	Software, firmware	3.00	1.41	3.40	0.49	2.20	0.75	2.20	0.40
2.	User interface and operation	2.20	0.75	2.60	0.49	1.40	0.80	1.20	0.40
3.	Electrical and electronic hardware	2.60	1.36	2.40	0.49	3.40	1.20	3.80	0.40
4.	Optics	1.40	0.49	1.40	0.49	2.20	1.17	2.40	0.49
5.	Mechanisms and mechanical hardware	1.40	0.49	1.20	0.40	2.40	1.50	2.60	0.80
6.	Data manipulation, flow, and storage	2.00	1.10	2.00	0.00	2.60	1.02	3.00	0.00
7.	Controls	1.80	1.17	1.40	0.49	1.80	0.75	2.00	0.00
8.	Verif., valid., compli. (e.g., FDA, FAA, UL, or CE)	3.60	1.50	4.00	0.00	3.40	1.36	3.80	0.75
9.	Security of operation	3.00	1.41	3.00	0.00	2.80	1.47	3.40	0.80
10.	Customer or client acceptance	3.20	0.75	3.00	0.00	2.80	1.47	2.80	0.40
11.	Extreme environments (e.g., temp., press., or vib.)	1.40	0.49	1.40	0.49	3.80	1.47	4.40	0.49
12.	Cost of final product	2.00	1.26	2.00	0.89	2.20	1.47	2.40	0.80
13.	Cost of development	2.40	1.50	3.00	0.00	2.60	1.36	2.40	0.49
14.	Time-to-market	3.60	1.50	3.40	0.49	2.20	1.47	2.20	0.40
15.	Changing requirements	2.40	1.36	2.40	0.49	2.20	1.47	2.20	0.40
16.	New technology, paradigm shift, revolutionary	1.60	0.80	1.60	0.49	1.60	0.80	2.00	0.00
17.	Power consumption or dissipation	1.60	1.20	1.80	0.40	2.20	1.60	2.00	1.10
18.	Size or weight	1.60	0.80	1.80	0.40	2.20	1.60	2.00	0.63
19.	Material or mass flow	1.00	0.00	1.00	0.00	1.20	0.40	1.00	0.00
20.	Competition	2.00	0.89	2.40	0.49	2.00	1.10	2.00	0.00
21.	Program management, team collaboration	2.20	1.17	2.20	0.40	2.20	1.17	2.80	0.40
22.	Other (describe in box below)	2.20	1.60	1.40	0.49	1.40	0.80	1.20	0.40
	Totals =	48.20	14.2	48.80	2.99	50.80	19.6	53.80	5.42

Technical challenges for Case Study 1, Neurostimulator Programmer, in the Delphi method and in the survey; each is compared to my estimates for ratings of technical challenges.



#	Question	Delphi method comparison			Survey comparison		
		Kim's Ratings	Rating Average	Rating Differences	Kim's Ratings	Rating Average	Rating Difference
1.	Software, firmware	4	3.4	-0.6	4	3.64	-0.36
2.	User interface and operation	4	2.6	-1.4	4	3.41	-0.59
3.	Electrical and electronic hardware	2	2.4	0.4	2	3.43	1.43
4.	Optics	1	1.4	0.4	1	2.64	1.64
5.	Mechanisms and mechanical hardware	3	1.2	-1.8	3	2.89	-0.11
6.	Data manipulation, flow, and storage	2	2	0	2	2.87	0.87
7.	Controls	1	1.4	0.4	1	2.97	1.97
8.	Verif., valid., compli. (e.g., FDA, FAA, UL, or CE)	4	4	0	4	4.28	0.28
9.	Security of operation	3	3	0	3	3.74	0.74
10.	Customer or client acceptance	4	3	-1	4	3.70	-0.30
11.	Extreme environments (e.g., temp., press., vib.)	2	1.4	-0.6	2	2.94	0.94
12.	Cost of final product	5	2	-3	5	3.30	-1.70
13.	Cost of development	4	3	-1	4	3.57	-0.43
14.	Time-to-market	2	3.4	1.4	2	3.58	1.58
15.	Changing requirements	2	2.4	0.4	2	3.68	1.68
16.	New technology, paradigm shift, revolutionary	2	1.6	-0.4	2	3.23	1.23
17.	Power consumption or dissipation	2	1.8	-0.2	2	2.83	0.83
18.	Size or weight	4	1.8	-2.2	4	2.77	-1.23
19.	Material or mass flow	1	1	0	1	2.36	1.36
20.	Competition	2	2.4	0.4	2	2.99	0.99
21.	Program management, team collaboration	3	2.2	-0.8	3	3.11	0.11
	Total =	57	47.4	-9.6	57	67.93	10.93
		NSS Delphi results			NSS survey results		
		Mean difference =		-0.457	**Mean difference =**		0.521
			Std. Dev. =	1.024		**Std. Dev. =**	1.018

Technical challenges for Case Study 2, Arc-Fault Detector, in the Delphi method and in the survey; each is compared to my estimates for ratings of technical challenges.



		Delphi method comparison			Survey comparison		
#	Question	Kim's Ratings	Rating Average	Rating Differences	Kim's Ratings	Rating Average	Rating Differen
1.	Software, firmware	3	2.2	-0.8	3	3.61	0.61
2.	User interface and operation	3	1.2	-1.8	3	3.31	0.31
3.	Electrical and electronic hardware	2	3.8	1.8	2	3.48	1.48
4.	Optics	2	2.4	0.4	2	2.70	0.70
5.	Mechanisms and mechanical hardware	2	2.6	0.6	2	2.93	0.93
6.	Data manipulation, flow, and storage	2	3	1	2	2.86	0.86
7.	Controls	1	2	1	1	2.93	1.93
8.	Verif., valid., compli. (e.g., FDA, FAA, UL, or CE)	4	3.8	-0.2	4	4.28	0.28
9.	Security of operation	4	3.4	-0.6	4	3.75	-0.25
10.	Customer or client acceptance	1	2.8	1.8	1	3.66	2.66
11.	Extreme environments (e.g., temp., press., vib.)	3	4.4	1.4	3	2.97	-0.03
12.	Cost of final product	2	2.4	0.4	2	3.28	1.28
13.	Cost of development	1	2.4	1.4	1	3.58	2.58
14.	Time-to-market	1	2.2	1.2	1	3.60	2.60
15.	Changing requirements	1	2.2	1.2	1	3.69	2.69
16.	New technology, paradigm shift, revolutionary	3	2	-1	3	3.31	0.31
17.	Power consumption or dissipation	1	2	1	1	2.80	1.80
18.	Size or weight	1	2	1	1	2.79	1.79
19.	Material or mass flow	1	1	0	1	2.42	1.42
20.	Competition	1	2	1	1	2.96	1.96
21.	Program management, team collaboration	3	2.8	-0.2	3	3.11	0.11
	Total =	42	52.6	10.6	42	68.01	26.01
		AFD Delphi results			AFD survey results		
		Mean difference =		0.505	**Mean difference =**		1.239
			Std. Dev. =	0.973		**Std. Dev. =**	0.953

Appendix L - Web Tools to Estimate Effort and Challenges

Overview of Proposed Tools

The purpose for these tools is to provide engineers and developers with estimations for the level-of-effort, duration of development, and potentially the COGS (cost of goods sold) of the final product. The focus of this tool will be embedded systems and products; it will not be for large vehicles or plants or system-of-systems. Examples of products and systems may include medical devices, military equipment, mechatronics, smaller, restricted networks of sensors, smaller, non-distributed software systems, and control systems for appliances, instruments, engines, industrial processes.

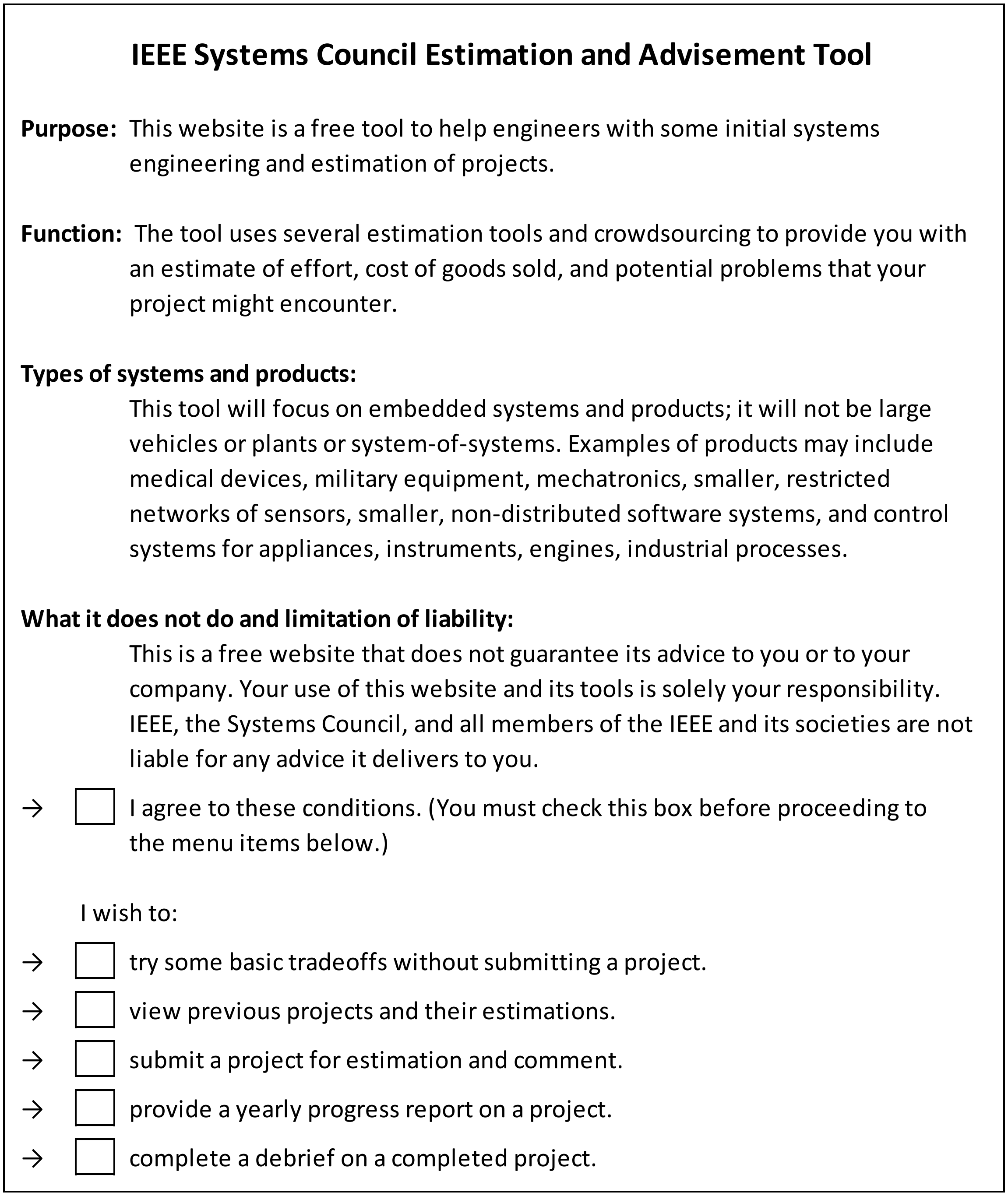
It could use several different methods to generate estimates. One set of methods might be heuristic calculations based on the expected product complexity; we will consider the various model estimators from Barry Boehm and the USC CSSE, such as COCOMO, Agile COCOMO II, COCOTS, COQUALMO, CORADMO, and possibly COSYSMO [204]. Another will be a crowdsource estimate. These will be estimates only, they will not claim accuracy nor liability; clients are not advised to rely on them.

The tool should be free and hosted on a website maintained by a technical society or organization. The website will be visited by clients, prospective clients, contributors, and staff of the hosting organization. A database will support the website and archive the submissions and actions on individual project estimations.

A client will submit a project description through a webpage; it will then generate a short report that client approves before it is submitted for consideration or for archiving. With calculations like heuristic estimations, the tool will generate immediate estimates of level-of-effort and duration of development. The tool will then randomly select and send requests to 400 to 1200 potential crowdsource contributors to provide individual estimates of the level-of-effort, duration of development, and COGS of the final product. The tool will collect the replies from the crowdsource contributors and generate aggregated estimates for a report sent to the client. It will then generate a short report for the client. The following sections of this appendix contain initial plans for webpages and interactions.

Proposed Client Title Webpage

This webpage is what greets clients and prospective clients. This page should be the last one developed. Some clients may only want to view potential tradeoffs or past histories of projects and estimation efforts.



IEEE Systems Council Estimation and Advisement Tool

Purpose: This website is a free tool to help engineers with some initial systems engineering and estimation of projects.

Function: The tool uses several estimation tools and crowdsourcing to provide you with an estimate of effort, cost of goods sold, and potential problems that your project might encounter.

Types of systems and products:

This tool will focus on embedded systems and products; it will not be large vehicles or plants or system-of-systems. Examples of products may include medical devices, military equipment, mechatronics, smaller, restricted networks of sensors, smaller, non-distributed software systems, and control systems for appliances, instruments, engines, industrial processes.

What it does not do and limitation of liability:

This is a free website that does not guarantee its advice to you or to your company. Your use of this website and its tools is solely your responsibility. IEEE, the Systems Council, and all members of the IEEE and its societies are not liable for any advice it delivers to you.

→ □ I agree to these conditions. (You must check this box before proceeding to the menu items below.)

I wish to:

→ □ try some basic tradeoffs without submitting a project.

→ □ view previous projects and their estimations.

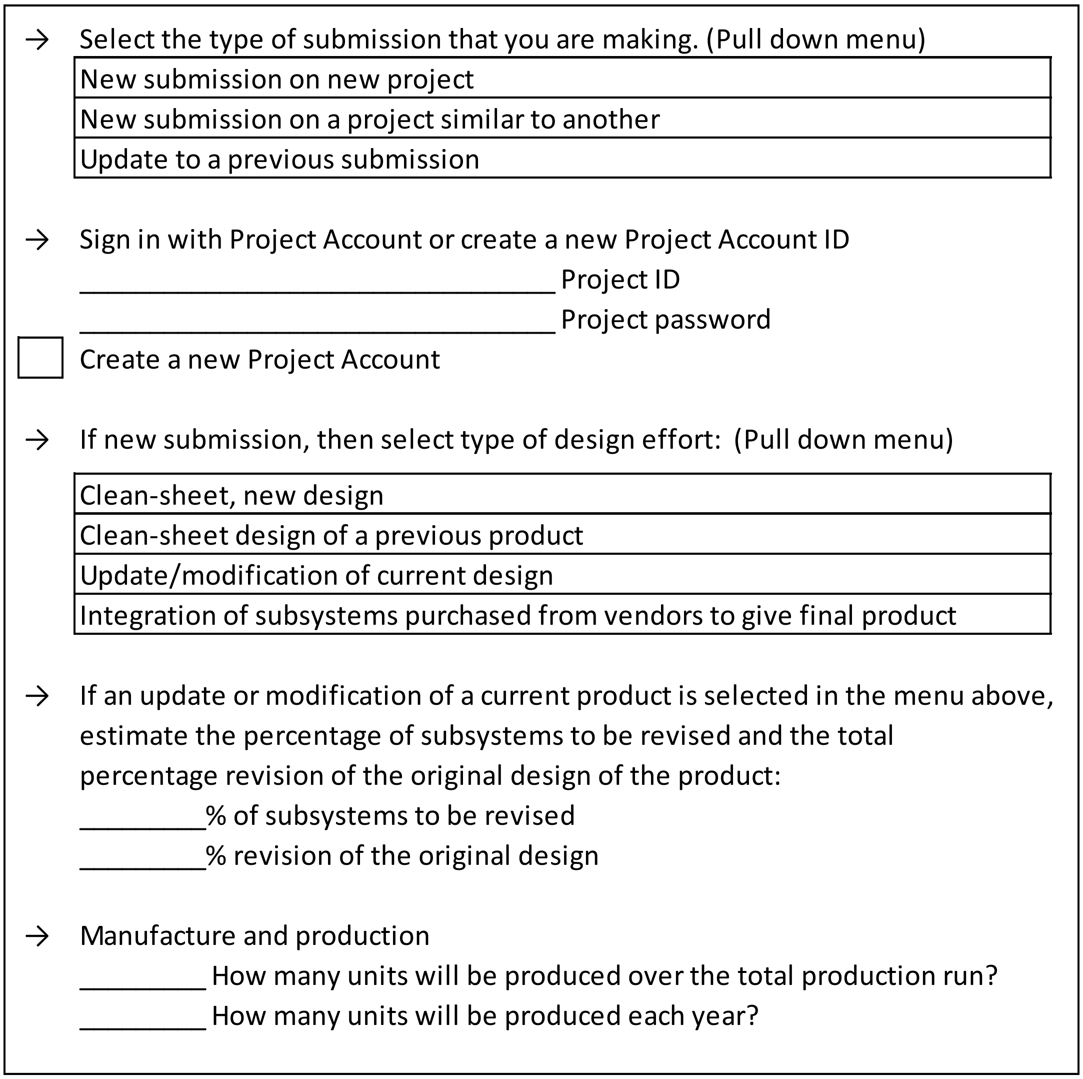
→ □ submit a project for estimation and comment.

→ □ provide a yearly progress report on a project.

→ □ complete a debrief on a completed project.

Proposed Survey for Client to Submit a Project for Estimation

The proposed survey format follows that a client submits a project for estimation. It provides consistent reports to crowdsource participants and minimizes extraneous or biased information. It should minimize time required by crowdsource participants to read and then respond with estimates.



→ Select the type of submission that you are making. (Pull down menu)

New submission on new project
New submission on a project similar to another
Update to a previous submission

→ Sign in with Project Account or create a new Project Account ID

________________________________ Project ID
________________________________ Project password

□ Create a new Project Account

→ If new submission, then select type of design effort: (Pull down menu)

Clean-sheet, new design
Clean-sheet design of a previous product
Update/modification of current design
Integration of subsystems purchased from vendors to give final product

→ If an update or modification of a current product is selected in the menu above, estimate the percentage of subsystems to be revised and the total percentage revision of the original design of the product:

_________% of subsystems to be revised
_________% revision of the original design

→ Manufacture and production

_________ How many units will be produced over the total production run?
_________ How many units will be produced each year?

1. In what industry(ies) will the project operate? (Check all that apply)

□ Aerospace or Defense

□ Agricultural equipment

□ Automotive

□ Civil construction or monitoring, infrastructure

□ Consumer devices or appliances

□ Government

□ Industrial or manufacturing equipment or process control

□ Medical or pharmaceutical

□ Petrochemical or mining

□ Power generation and distribution

□ Scientific and test equipment

□ Transportation (other than automotive), cargo hauling, materials extraction

□ Other - describe:

2. Are there regulatory environments or standards that your product must meet? Check all that apply:

□ Transportation - FAA, DOT, NHTSA

□ Military - DoD,

□ Medical - FDA

□ Consumer - UL, CE Mark, VDE

□ Space - NASA guidelines

□ National Electrical Code (NEC), fire codes

□ Describe any certifications or approvals you must obtain and any regulations or guidelines you must follow:

3. What is the operational environment in which your product will operate? Check all that apply:

□ Indoors, controlled temperature and humidity

□ Outdoors, uncontrolled

□ Marine

□ Corrosive

□ High pressure

□ High radiation

□ Space - vacuum and radiation

□ Vibration (if checked, then select type from pulldown menu)

Low levels for short-term, e.g., sliding appliance across a counter
Low levels over long-term, e.g., distant manufacturing mildly buzzes table top
High levels for short-term, e.g., missile launch
High levels over long-term, e.g., shaker screen for industrial process

□ Mechanical shock (if checked, then select type from pulldown menu)

Low magnitude, short-term, e.g., bumping an appliance with your elbow
Low magnitude, long-term, e.g., repeated drops of a few millimeters
High magnitude, short-term, e.g., 2 meter high drop to concrete surface
High magnitude, long-term, e.g., large gun repeatedly firing

□ Other - describe:

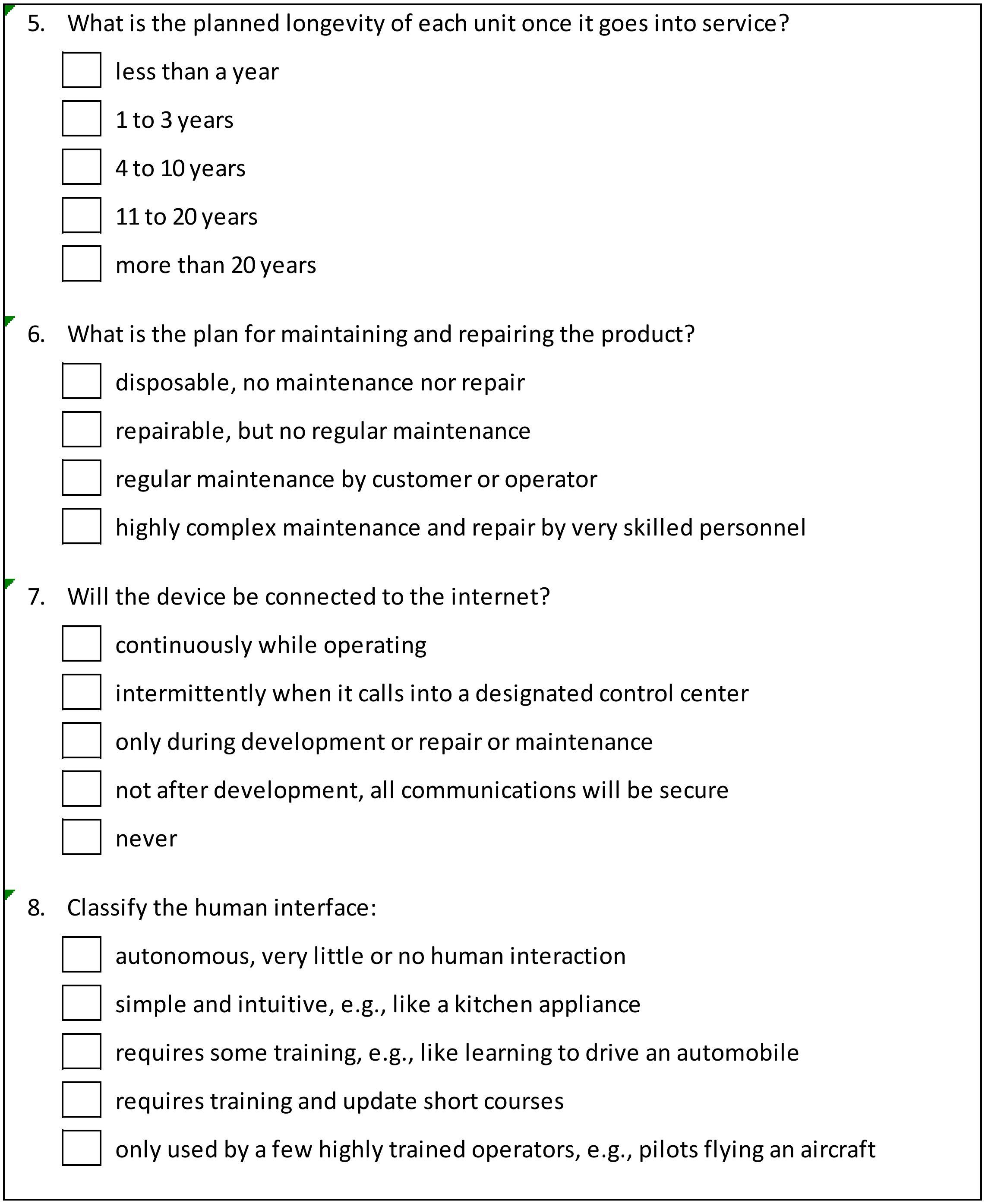
4. Estimate aspects of your project in each of the following three lines:

Number of subsystems (e.g., circuit boards, power supplies, mechanisms): _______

Number of lines of code (LOC): _____________

Number of logical operations or function points: ______________

□ Check only if you have no confidence in your numbers.

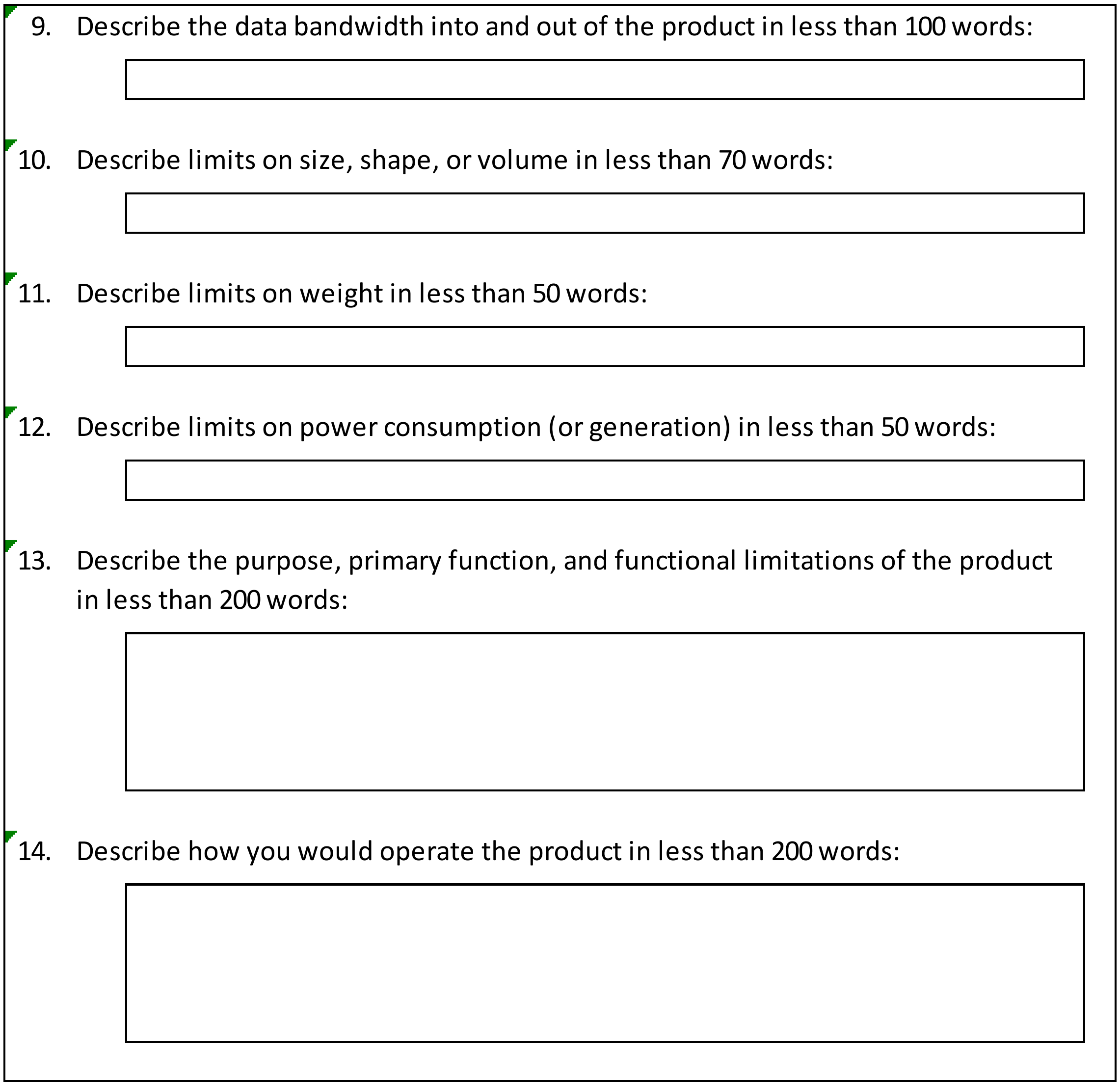


5. What is the planned longevity of each unit once it goes into service?
   - □ less than a year
   - □ 1 to 3 years
   - □ 4 to 10 years
   - □ 11 to 20 years
   - □ more than 20 years

6. What is the plan for maintaining and repairing the product?
   - □ disposable, no maintenance nor repair
   - □ repairable, but no regular maintenance
   - □ regular maintenance by customer or operator
   - □ highly complex maintenance and repair by very skilled personnel

7. Will the device be connected to the internet?
   - □ continuously while operating
   - □ intermittently when it calls into a designated control center
   - □ only during development or repair or maintenance
   - □ not after development, all communications will be secure
   - □ never

8. Classify the human interface:
   - □ autonomous, very little or no human interaction
   - □ simple and intuitive, e.g., like a kitchen appliance
   - □ requires some training, e.g., like learning to drive an automobile
   - □ requires training and update short courses
   - □ only used by a few highly trained operators, e.g., pilots flying an aircraft



9. Describe the data bandwidth into and out of the product in less than 100 words:

10. Describe limits on size, shape, or volume in less than 70 words:

11. Describe limits on weight in less than 50 words:

12. Describe limits on power consumption (or generation) in less than 50 words:

13. Describe the purpose, primary function, and functional limitations of the product in less than 200 words:

14. Describe how you would operate the product in less than 200 words:

The report, approved by the client before completing the submission, will have the following look-and-feel to it. This is what crowdsource participants will see:

Purpose, primary function, and functional limitations of the product:

How to operate the product:

Type of project:
(e.g., clean-sheet or update or integration)

If project is an update:
_______% of subsystems to be revised
_______% revision of total product

Manufacturing or production:
_________ total number of units to be produced over market lifetime.
_________ number of units to be produced each year.

Industries in which the product will operate:

Regulations, standards, certifications, or approvals that the product must meet:

Operational environment in which the product will operate:

Estimates of project complexity:
Number of subsystems within product: _____________
Number of lines of code within product: _____________
Number of logical operations or function points: _____________

Planned longevity of each unit once it enters service: ________ years

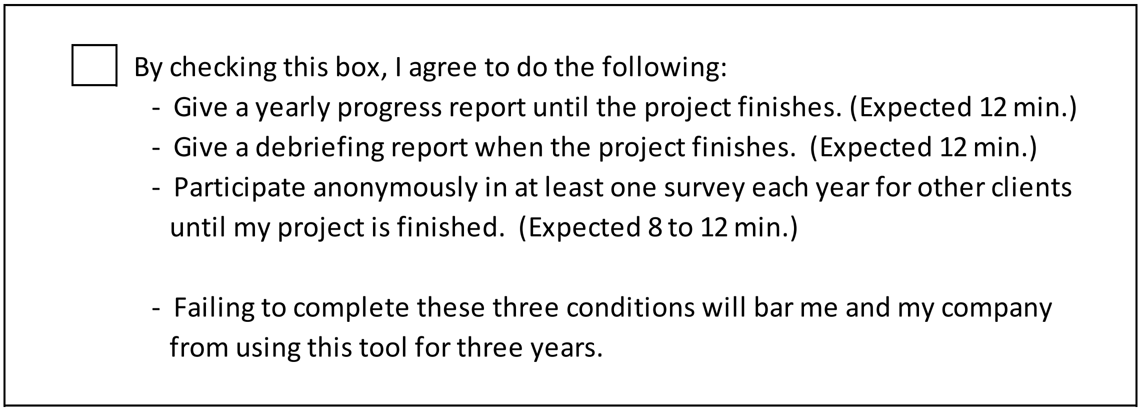
Maintenance and repair philosophy:

Internet connectivity:

Human interface:

Bandwidth:
Size, shape, volume:
Weight:
Power consumption (or generation):

When the client wishes to submit the project for consideration and estimation, the client must check this box to finalize the submission.



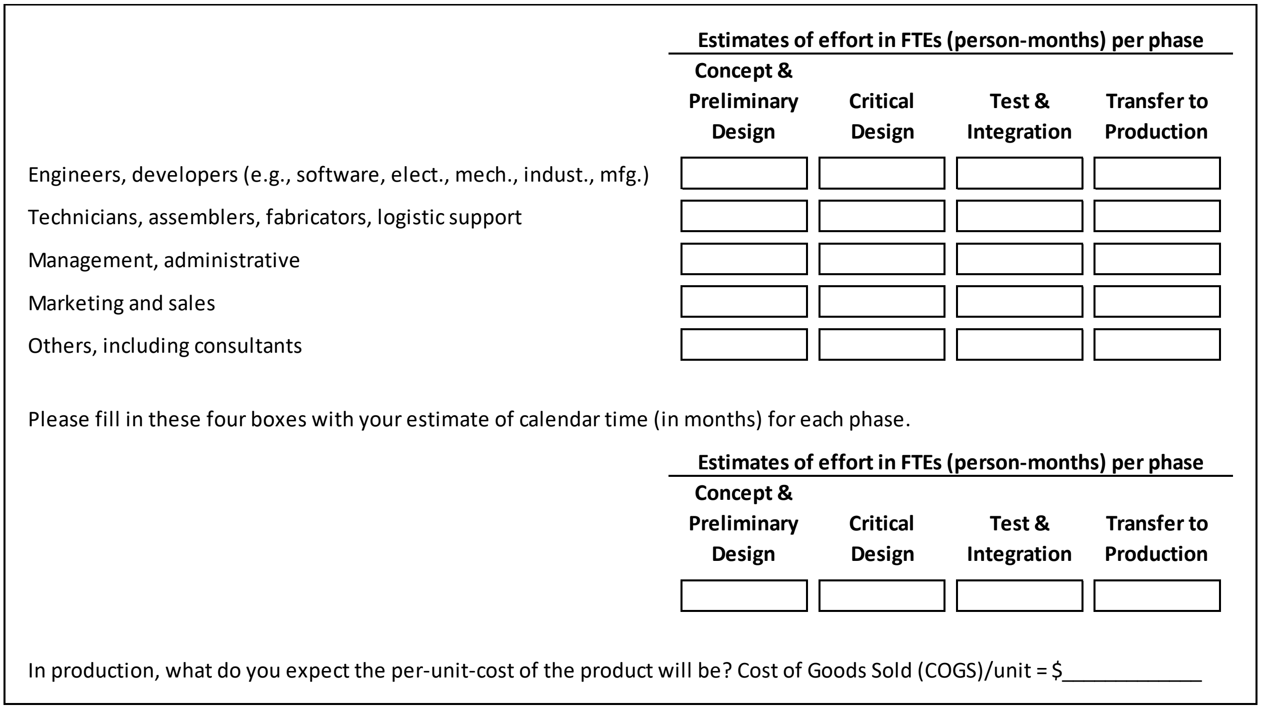
□ By checking this box, I agree to do the following:

- Give a yearly progress report until the project finishes. (Expected 12 min.)
- Give a debriefing report when the project finishes. (Expected 12 min.)
- Participate anonymously in at least one survey each year for other clients until my project is finished. (Expected 8 to 12 min.)

- Failing to complete these three conditions will bar me and my company from using this tool for three years.

Proposed Survey for Participants to Provide Project Estimation

Crowdsource and Delphi method participants will see and read the report given in the section of the appendix above. They then will respond to the following survey:



	Estimates of effort in FTEs (person-months) per phase			
	Concept & Preliminary Design	**Critical Design**	**Test & Integration**	**Transfer to Production**
Engineers, developers (e.g., software, elect., mech., indust., mfg.)				
Technicians, assemblers, fabricators, logistic support				
Management, administrative				
Marketing and sales				
Others, including consultants				

Please fill in these four boxes with your estimate of calendar time (in months) for each phase.

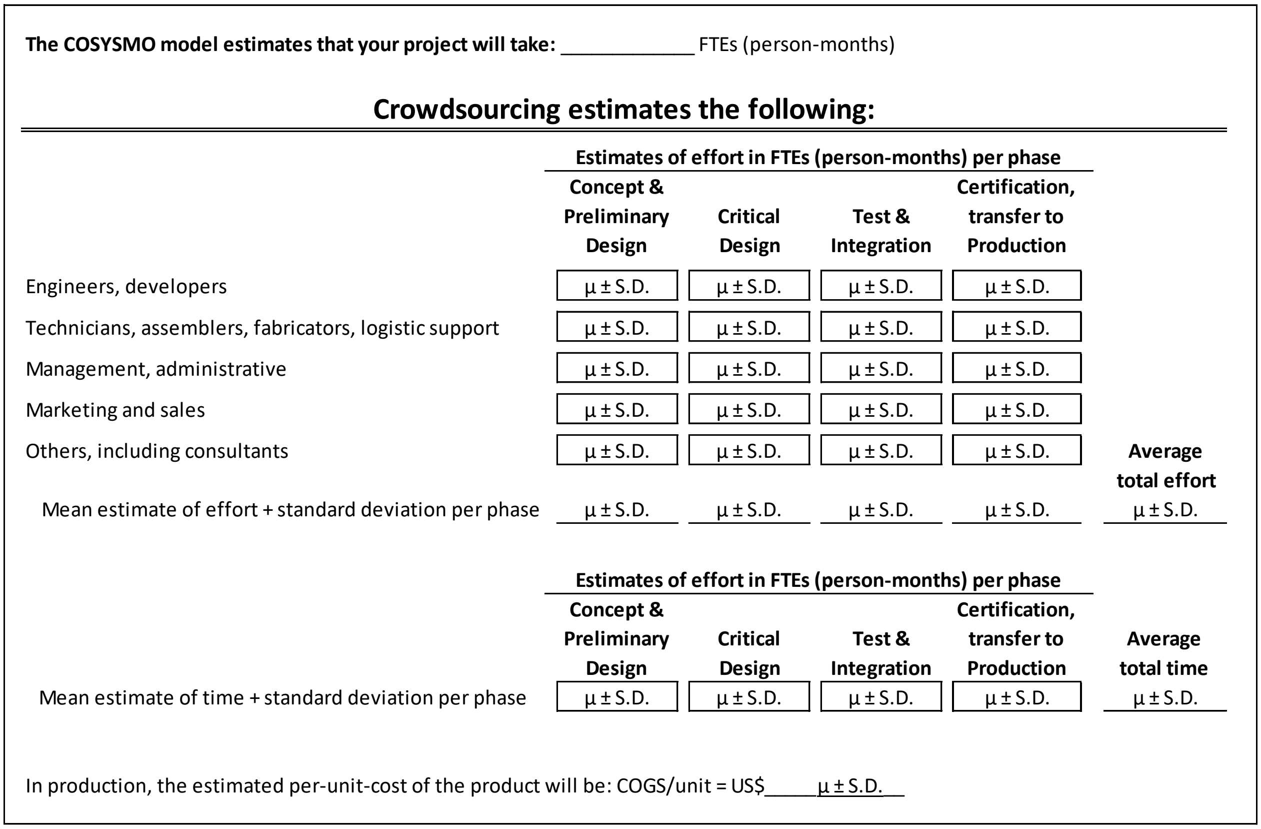
Estimates of effort in FTEs (person-months) per phase			
Concept & Preliminary Design	**Critical Design**	**Test & Integration**	**Transfer to Production**

In production, what do you expect the per-unit-cost of the product will be? Cost of Goods Sold (COGS)/unit = $____________

	Low - effort, cost, and technology are understood and controlled; e.g., system upgrade to a servomechanism that does not require software changes and only minor differences in the motor and drive.		**Medium** - effort, cost, and technology are understood; e.g., clean-sheet design of a servomechanism that uses available components.	**High** - effort, cost, or technology are new to the team; e.g., clean-sheet design of a spacecraft instrument with new software and autonomy.	
	1	2	3	4	5
Software, firmware	□	□	□	□	□
User interface and operation	□	□	□	□	□
Electrical and electronic hardware	□	□	□	□	□
Optics	□	□	□	□	□
Mechanisms and mechanical hardware	□	□	□	□	□
Data manipulation, flow, and storage	□	□	□	□	□
Controls, mechanisms	□	□	□	□	□
Verification, validation, compliance (e.g., FDA or FAA or UL or CE)	□	□	□	□	□
Security of operation	□	□	□	□	□
Customer or client acceptance	□	□	□	□	□
Extreme environments (e.g., temperature or pressure or vibration)	□	□	□	□	□
Cost of final product	□	□	□	□	□
Cost of development	□	□	□	□	□
Time-to-market	□	□	□	□	□
Changing requirements	□	□	□	□	□
New technology, paradigm shift, revolutionary	□	□	□	□	□
Power consumption or dissipation or handling	□	□	□	□	□
Size or weight	□	□	□	□	□
Material or mass flow	□	□	□	□	□
Competition	□	□	□	□	□
Program management, team collaboration	□	□	□	□	□
Other: ______________________________	□	□	□	□	□

Proposed Report Format of the Project Estimation Provided to the Client

The proposed format of the report, supplied to the client, that contains the crowdsource estimates will have the following formats.



The COSYMO model estimates that your project will take: ____________ FTEs (person-months)

Crowdsourcing estimates the following:

	Estimates of effort in FTEs (person-months) per phase				
	Concept & Preliminary Design	**Critical Design**	**Test & Integration**	**Certification, transfer to Production**	**Average total effort**
Engineers, developers	µ ± S.D.	µ ± S.D.	µ ± S.D.	µ ± S.D.	
Technicians, assemblers, fabricators, logistic support	µ ± S.D.	µ ± S.D.	µ ± S.D.	µ ± S.D.	
Management, administrative	µ ± S.D.	µ ± S.D.	µ ± S.D.	µ ± S.D.	
Marketing and sales	µ ± S.D.	µ ± S.D.	µ ± S.D.	µ ± S.D.	
Others, including consultants	µ ± S.D.	µ ± S.D.	µ ± S.D.	µ ± S.D.	
Mean estimate of effort + standard deviation per phase	µ ± S.D.	µ ± S.D.	µ ± S.D.	µ ± S.D.	µ ± S.D.

	Estimates of effort in FTEs (person-months) per phase				
	Concept & Preliminary Design	**Critical Design**	**Test & Integration**	**Certification, transfer to Production**	**Average total time**
Mean estimate of time + standard deviation per phase	µ ± S.D.	µ ± S.D.	µ ± S.D.	µ ± S.D.	µ ± S.D.

In production, the estimated per-unit-cost of the product will be: COGS/unit = US$_____µ ± S.D.__

Classification of Problem Areas

Low (1 to 2) - effort, cost, and technology are understood and controlled; e.g., system upgrade to a servomechanism that does not require software changes and only minor differences in the motor and drive.

Medium (2 to 4) - effort, cost, and technology are understood; e.g., clean-sheet design of a servomechanism that uses available components.

High (4 to 5) - effort, cost, or technology are new to the team; e.g., clean-sheet design of a spacecraft instrument with new software and autonomy.

Estimate

Software, firmware ____________

User interface and operation ____________

Electrical and electronic hardware ____________

Optics ____________

Mechanisms and mechanical hardware ____________

Data manipulation, flow, and storage ____________

Controls, mechanisms ____________

Verification, validation, compliance (e.g., FDA or FAA or UL or CE) ____________

Security of operation ____________

Customer or client acceptance ____________

Extreme environments (e.g., temperature or pressure or vibration) ____________

Cost of final product ____________

Cost of development ____________

Time-to-market ____________

Changing requirements ____________

New technology, paradigm shift, revolutionary ____________

Power consumption or dissipation or handling ____________

Size or weight ____________

Material or mass flow ____________

Competition ____________

Program management, team collaboration ____________

Other: ____ ____________

Challenges Facing the Tool

Both crowdsourcing and the Delphi method require a level of motivation for adequate participation in the survey. Potentially the client would provide a few gift cards (e.g., 20 gift cards, each worth $25, which totals to a $500 commitment) to randomly selected participants. Or the client might give a gift card to each of the first 1000 participants (e.g., 1000 gift cards, each worth $5, which totals to a $5000 commitment). For Delphi participants, a flat consulting fee of perhaps $1,400 may help; $200 to each of the five participants; $200 to the moderator; $200 to the hosting organization.