

~~THE ANALYSIS OF THE SOFTWARE PROCESS MODEL~~

by

YING-CHI CHEN

B.S., Eastern Michigan University, 1982

M.S., Eastern Michigan University, 1983

A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

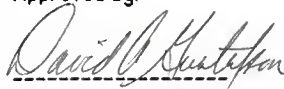
MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1986

Approved by:



Major Professor

ACKNOWLEDGEMENTS

I wish to express my sincere thanks to my committee members, Dr. Hankley and Dr. Melton. My special thanks go to advisor Dr. Gustafson, for his help, guidance, encouragement and support. Thanks also go to my friends, Janice Taylor and Kewing Wong for their assistance.

In addition, I would like to thank my parents, Pong Chen and Wong-Wei Chen for their spiritual support. I would also like to thank my husband, Tin-Yu Ting for his encouragement.

CONTENTS

1. INTRODUCTION-----	1
1.1 Introduction-----	1
1.2 The Goals of This Paper-----	2
1.3 Summary of Contents-----	3
2. THE SOFTWARE PROCESS MODEL AND OTHER MODELS-----	4
2.1 Introduction of Software Process Model-----	4
2.2 Software Document-----	8
2.2.1 Software Life Cycle-----	8
2.2.2 Documents in Software Life Cycle-----	12
2.3 Other Software Models-----	16
2.3.1 Waterfall Model-----	16
2.3.2 Rapid Prototyping-----	18
2.3.3 Software Sculpture-----	19
2.3.4 Incremental Development-----	20
2.3.5 RUDE-----	21
2.4 Summary-----	22
3. MEASURES-----	26
3.1 Basic Concepts-----	26
3.2 Measure Development Paradigm-----	29
3.3 Waste Measure-----	33
3.4 Effort Measure-----	41
3.5 Stability Measure-----	50
3.6 Progress Measure-----	60
3.7 Conclusion-----	68
4. DATA ANALYSIS AND INTERPRETATION-----	71
4.1 Introduction-----	71
4.2 Data Analysis And Interpretation-----	71
4.2.1 Apply The Measures to the Project Groups-----	73
4.2.2 Data Analysis For The Project Groups-----	79

4.2.3 Projection Models For The Project Groups-----	84
4.2.4 Data Analysis And Projection Models For the Whole Project--	87
4.2.5 Evaluations From The Project Groups-----	90
4.2.6 SPM Diagram And Initial Diagram-----	98
4.3 Summary-----	100
5. CONCLUSIONS-----	102
5.1 Introduction-----	102
5.2 The Evaluation of The Measures and The Models-----	103
5.3 Summary-----	109
LIST OF TABLES-----	iii
LIST OF FIGURES-----	iv
APPENDIX A: The SPM Diagrams for Each Group-----	111
APPENDIX B: The Scatter Diagrams Between Completion Time and Waste, Progress, Stability and Effort Measures for Each Group-----	122
APPENDIX C: The Projection Models for the Waste, Progress, Stability and Effort Measures for Each Group-----	137
APPENDIX D: The Values of MRE for Each Group-----	141
APPENDIX E: Definitions-----	150

LIST OF TABLES

Table 4.1. The total value for the waste, progress, stability and effort for a project team-----	75
Table 4.2. The total value for the waste, progress, stability and effort for each group-----	76
Table 4.3. The correlation matrix for the group A-----	77
Table 4.4. The correlation matrix for the group G-----	77
Table 4.5. The correlation coefficients for the measures in each group--	79
Table 4.6. The MRE for the measures in each group-----	85
Table 4.7. The residuals for the measures in each group-----	86
Table 4.8. The correlation matrix for the measures-----	89
Table 4.9. The values of evaluations from the project teams and total values from the measures-----	92
Table 5.1. Mean and standard deviation for the measures-----	105
Table 5.2. The slopes of the measures for each project team-----	106

LIST OF FIGURES

Figure 2.1. SPM diagram for idealized development-----	4
Figure 2.2. The Software Project Model(SPM)-----	6
Figure 2.3. Software Life Cycle-----	10
Figure 2.4. Waterfall model-----	17
Figure 2.5. Incremental program development-----	20
Figure 2.6. Alternatives life cycle models apply to different type of the system-----	23
Figure 2.7. SPM diagram for waterfall model-----	24
Figure 2.8. SPM diagram for rapid prototyping-----	24
Figure 2.9. SPM diagram for incremental development or RUDE-----	24
Figure 2.10. SPM diagram for system sculpture-----	25
Figure 3.1. A simple example for production-----	33
Figure 3.2. An example of the process of software development-----	34
Figure 3.3. The results of the waste, effort, stability and progress measures-----	70
Figure 4.1. The SPM diagram for a project team-----	73
Figure 4.2. The SPM diagram for group G-----	78
Figure 4.3. The SPM diagram for group A-----	78
Figure 4.4. The scatter diagram for the value of waste and completion time with the greatest value of r for group F-----	80
Figure 4.5. The scatter diagram for the value of waste and completion time with the least value of r for group J-----	81
Figure 4.6. The scatter diagram for the value of progress and completion time with the greatest value of r for group F-----	81

Figure 4.7. The scatter diagram for the value of progress and completion time with the least value of r for group D-----	82
Figure 4.8. The scatter diagram for the value of stability and completion time with the greatest value of r for group L-----	82
Figure 4.9. The scatter diagram for the value of stability and completion time with the least value of r for group C-----	83
Figure 4.10. The scatter diagram for the value of effort and completion time with the greatest value of r for group L-----	83
Figure 4.11. The scatter diagram for the value of effort and completion time with the least value of r for group C-----	84
Figure 4.12. The scatter diagram for the waste measure-----	87
Figure 4.13. The scatter diagram for the progress measure-----	87
Figure 4.14. The scatter diagram for the stability measure-----	88
Figure 4.15. The scatter diagram for the effort measure-----	88
Figure 4.16. The scatter diagram for the value of waste from the project teams and the measures-----	92
Figure 4.17. The scatter diagram for the value of progress from the project teams and the measures-----	93
Figure 4.18. The scatter diagram for the value of stability from the project teams and the measures-----	93
Figure 4.19. The scatter diagram for the value of effort from the project teams and the measures-----	94
Figure 4.20. The scatter diagram for the slope of the waste measure and the evaluation from the project teams-----	95
Figure 4.21. The scatter diagram for the slope of the progress measure and the evaluation from the project teams-----	95

Figure 4.22. The scatter diagram for the slope of the stability measure and the evaluation from the project teams-----	96
Figure 4.23. The scatter diagram for the slope of the effort measure and the evaluation from the project teams-----	96
Figure 4.24. Comparing the SPM diagram and initial diagram for a project team-----	100

Chapter One

INTRODUCTION

1.1 Introduction

In the late 1960s, the structured programming revolution changed the nature of programming. The programming community's goals were to produce a product that was not only more reliable but also to produce the product in a more engineering-like fashion. Since then, the development of software engineering is considered as a possible way to achieve these goals. After more than twenty years, software engineering still requires extensive human participation.

The nature of software engineering has been that the development of most methodologies has come from experiences rather than by using mathematical derivations. It is different from the other engineering disciplines which emphasize mathematical formalism. For software engineering, it is impossible to attain the status of a scientific discipline unless it is built upon a sound foundation of measurement. Measurements not only allow us to describe the process of software development by collecting and saving the information about the various activities which occur in the software life cycle, but also help us to understand and predict different situations.

The systematic collection of data from past projects is a good method to

achieve the purposes of better control, management, and prediction. Documents generated in each software life cycle phase can fully reflect the process of software system development.

The Software Process Model (SPM), a newly developed model, views software development as the process of refining a set of documents. For convenience in the analysis of the model, the SPM is defined as a set of document histories. Moreover, the SPM provides a valuable framework for software measures and metrics research and development. These measures and metrics can be used to compare different development efforts. Measures that reflect characteristics of software development can provide valuable feedback to the system analyst to better control the whole development process. The SPM can be developed as a useful software management tool by using a mathematical foundation instead of just experience.

1.2 The Goals of This Paper

As DeMarco [DeM,1982] pointed out, in software engineering "You can not control what you can not measure.". In this paper, I will use the SPM to develop a number of measures and projection models including waste, effort, stability and progress. The purpose of these measures is to allow us to compare the SPMs of different project teams in order to compare and

analyze the differences in the waste, effort, stability, and progress of these teams. The projection models for the waste, stability, progress and effort measures can help us to foresee problems earlier in the life cycle. Using information obtained from these measures and projection models, managers can improve the management of the software development. This research project will apply the SPM to the software projects produced by students in a senior computer science course CS540-541 (Spring, 1988) in order to learn more about the SPM and its application.

1.3 Summary of Content

In this chapter, a brief introduction to the Software Process Model (SPM) and the purpose of this paper is given. Chapter two will present more detail about the SPM. The alternative development models will be given. The documents generated in each software life phase will be introduced. In chapter three, the measure development paradigm will be presented and several measures will be developed according to the SPM. In chapter four, the measures for waste, progress, stability and progress will be applied to data from the CS540-541 projects. The result of the measures will be analyzed. A set of projection models will be generated by using statistical methods. Finally, the measures and models will be evaluated.

Chapter 2

THE SOFTWARE PROCESS MODEL AND OTHER MODELS

2.1 Introduction of Software Process Model

The **Software Process Model (SPM)** is a formalism for viewing the development process [BAK, 87]. The **SPM** has several major characteristics that are listed as follows:

1. The **SPM** is developed from the perspective of the software product.

The **SPM** is an abstraction of the activities and products of actual software development. The model focuses on evolving products of a software project and relationships between products. Each of the products can be described as a document. An idealized **SPM** diagram is shown in Figure 2.1. This diagram shows two passes through the development process.

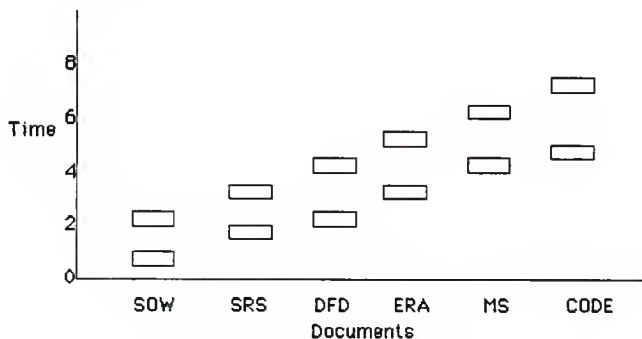


Figure 2.1: **SPM** diagram for idealized development

The vertical axis is time. The horizontal categories are document types. Each point (or node) represents a version of the particular document.

2. It is focused on the process of software development. The software development is viewed as the process of refining a set of documents. The process is time dependent and therefore a real time clock is a critical component of the model.

The following definitions describe the SPM more precisely.

Definition 2.1 The **Software Process Model (SPM)** is a set of document histories, $SPM = \{H_1, H_2, \dots, H_n\}$ where

1. H_i is a history of the versions of document type i .

Definition 2.2 A document history H_i is a tree whose nodes are version of documents and $H_i = (U_i, E_i, r_i)$ where

1. U_i is a set of document versions of type i ,
2. E_i is a set of ordered pairs of the form (a,b) , $a,b \in U_i$ which represent transitions from one version in U_i to another,
3. r_i is the root of the H_i tree or the initial version.

A **tree** is used to represent a document history to allow the modeling of the development of alternative documents in the same document history. Suppose a system is to run on hardware produced by several different vendors. This kind of system may need different specifications, designs, and implementations for each different hardware. This can be modelled by a tree.

The use of **SPM** to describe a possible software project is illustrated in figure 2.2.

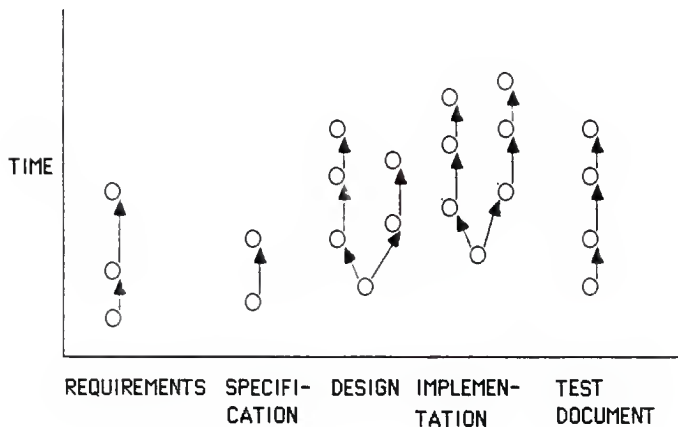


Figure 2.2 : **The Software Project Model (SPM)**

Definition 2.3 A document version $d_i(j) \in U_i$ is an ordered pair

$d_i(j) = \langle d'_i(j), td_i(j) \rangle$, where

1. $d'_i(j)$ consists of the text of the document version,
2. $td_i(j)$ is a realtime stamp which represents the completion date/time for $d'_i(j)$.

The time stamp can be augmented with a wide range of process measures such as the number of persons/hour used to develop the document version, the publication date - the time/date that the document version was released to teams working on other documents, and/or the "extent" of publication for versions released to a limited audience.

3. The **SPM** assumes that development can take place in parallel. This means that several different phases of documents can be developed concurrently. Parallel development is useful for a project with a tight schedule. The **SPM** does not prescribe an order of activities and the products need not be named. Thus, the **SPM** may be applied to a variety of development approaches.
4. The **SPM** provides a framework for quantifying the software process and software products. The model helps clarify the distinction between the

products of software development and the development process. The **SPM** is used to formally characterize research in software measures and metrics. The formal definition for quantitative measures and metrics to assess the development process will be given in chapter 3.

In summary, **SPM** is developed to model the evolution of a set of documents produced in a software project. (The **SPM** provides a framework for measuring, analyzing, and understanding the software development process. This model could be applied to any software development approach.)

2.2 Documents

Documents generated during the development process serve as milestones that can help managers to control and assess progress. Documentation develops as a consequence of each software engineering phase.

Without these documents, the software could not be maintained. Like hardware, computer software should evolve through a series of carefully controlled and systematically executed phases. The idea of the **software life cycle** is developed.

2.2.1 Software Life Cycle

The **Software life cycle** is known as a set of activities which has been defined to describe the software process from system definition through system maintenance. There are many life cycle models [McC, 1981;

BLU,1982],several features are common to all. These features are:

- . Each model begins with an activity which identifies the problem to be solved and describes a solution space for the product. The product should possess the following properties:
 - A product will be produced which will satisfy the requirements.
 - The product will have a post development existence (evolution).
- . Each model recognizes that the process of going from requirements to product can be solved only through decomposition into successively smaller problems each of which can be solved individually.

Different **life cycle models** can be decomposed into three phases : **definition, development and maintenance**. Figure 2.3 represents the overall flow of events during the **software life cycle** [GEN ,1986].

The activities involve in each phase are:

A. Definition Phase:

- . The software project is planned;
- . Budgets and schedules are estimated;
- . Detailed requirements are analyzed and specified;

B. Development Phase:

- . Software requirements are transformed into an operational program using proven methods for designing, coding and testing. The activities in the development phase consist of five steps:

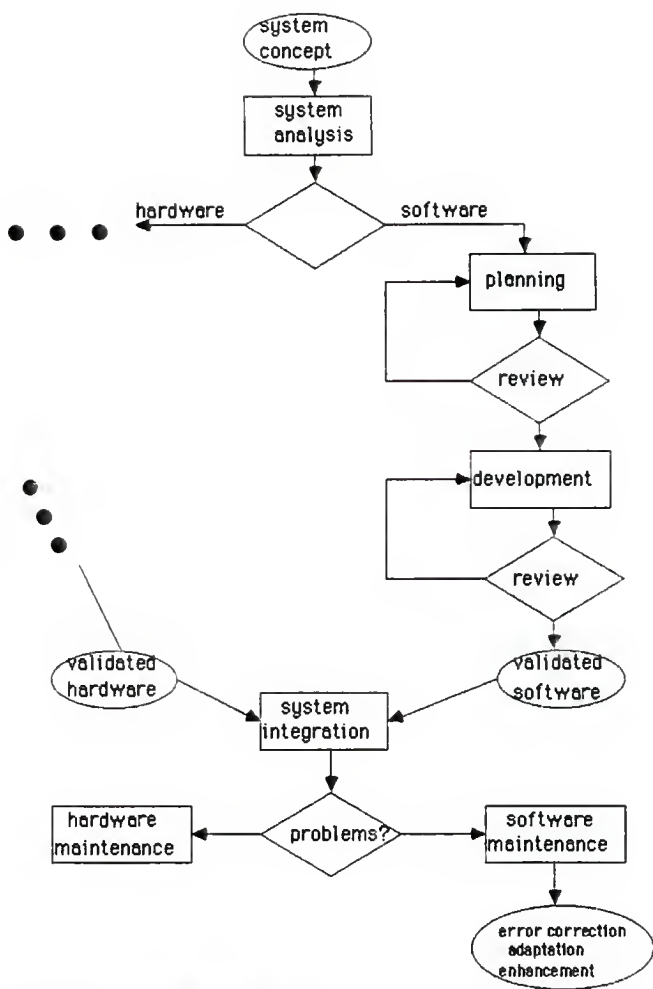


Figure 2.3: Software Life Cycle

1. **Preliminary Design** : It translates a set of well-defined requirements into a workable software structure, the preliminary design.
2. **Detailed Design** : The **preliminary design** is expanded to include the internal design of each module. The resulting expansion of the **preliminary design** is known as **detailed design**.
3. **Code and Unit Test** : **Coding** is the translation from the design representation to the appropriate programming language. **Unit test** is testing an individual module and its interface in order to verify its consistency with the design specification.
4. **Integration Testing** : It combines **unit tested** modules in a manner that allows the entire software package to be assembled and tested in a stepwise fashion. **Integration testing** may be performed using "**bottom-up**" or "**top-down**" techniques.
5. **Formal Validation Testing** : It is evaluating the software with respect to the specified functions defined for the system.

C. Maintenance Phase

- . Problems encountered in the field are corrected;
- . Adaptations of the software are made for different operational environments;
- . Enhancements to functional requirements are implemented.

The **software life cycle** encompasses all activities required to **define, develop, test, deliver, operate, and maintain** a software product.

2.2.2 Documents in Software Life Cycle

Each software life cycle step results in waste of one or more documents. Each type of document provides a unique view of the system. The complete and up-to-date document is developed throughout the life cycle. This is the major reason for **SPM** viewing software development as the process of refining a set of documents.

Documents typical of each **software life cycle** phase are discussed in the following section.

A. Definition Phase :

. **Statement Of Work (SOW)** : This document describes the basic properties, restrictions on inputs and outputs, contents of reports, constraints, and any other pertinent details on the intended software. The **SOW** is usually divided into paragraphs and sections. Each of these is numbered. This allows cross-referencing within the document and indexing of the requirements[GUS, 1987].

. **The Software Requirement Specification (SRS)** : "A requirements document should specify the functions to be performed by the system, from the view point of the user or external environment" [DAV, 1979]. "A requirements document should specify the external behavior of a system, without implying a particular implementation" [HEN, 1980]. SRS is the requirements baseline for the software subsystem. This document is important for management control and technical development.

Documents in the definition phase are typically expressed in English or some other natural language and may incorporate charts, figures, graphs, tables, and equations of various kinds. The **definition phase** begins with an evaluation of the overall system and ends with a software plan review.

6. Development Phase

--Preliminary Design

. **Data Flow Diagrams (DFD)**: The first phase of design after requirements specification is data flow analysis. **DFD** may be edited interactively through a graphics editor, with data item names captured and stored in a data dictionary. The diagrams should be maintained as a tree of drawings according to the DeMarco conventions [DEM, 1978] for leveled **DFD** sets. **DFD** is used as a mechanism for creating a top level structure design for software.

. **Input/Output Specification**: This document specifies the source and type of input and output information associated with a function. This includes a description of the information, its source and destination in quantitative terms, e.g. unit of measure, etc..

. **The Entity-Relationship-Attribute (ERA)**: **ERA** approach was developed in parallel by workers in the fields of data bases [CHE, 1976; CHE, 1980; CHE, 1983; DAV, 1983] and artificial intelligence [BAR, 1982-1983]. Entities are items in the world about which the desired system is intended to process information. An attribute is a value that is connected to an entity. A relation is any connection between entities. **ERA** can be used to specify requirements.

. **Hierarchy Diagram (HD)** : HD is a hierarchical structure of **data flow diagrams** that fit onto one sheet of paper. The more detailed processes are shown on separate **data flow diagrams** on other pages. It is also referred to as a calling diagram. The software structure is a hierarchical representation of a software system.

The **preliminary design** is the first step in the development of the design document. Each module in the software structure is described in detail with interfaces defined, limitations and restrictions identified, and data structure characteristics determined.

--Detailed Design

. **Hierarchy Specification (HS)** : It describes the structure of the software. The hierarchy specification often includes a text description to describe the relationship between the modules.

. **Module Specifications (MS)** : MS specify the behavior of each of the modules. MS includes the specifications of the inputs and outputs.

. **Design Walkthrough**: The design must be reviewed, inspected and evaluated to ensure that the dataflow and specification are correct and consistent.

The goal of **detailed design** is the development of a software representation directly translatable into a programming language.

--CODE

. **Source Code** : Coding is to translate the design description into a

predetermined programming language. Code documentation is part of the source listing or is implied by the code itself. Comments should be included in the code documentation. Source code can be clarified by using enough explanatory information.

Since the early 1970s, structured programming has received much attention in the software engineering field. Because it is better than unstructured programming in three areas: increased reliability, easier verification, and easier modification[DIJ, 1972]. Also, built in and user-defined data types, secure type checking, flexible scope rules, exception handling mechanisms, concurrency constructs and separate compilation are introduced to enhance the quality of the code [FAI, 1985].

--Unit Test, Integration Test, Formal Validation Testing

. **Test** Testing is a series of stimulus-response activities. Test document should describe the inputs and the results of these activities completely, in order to make sure the responses are satisfying its specification and its expected responses.

The goal of **unit and integration testing** is to uncover latent errors by making the software fail. The goal of **validation testing** is demonstrating software traceability to requirements. Eliminating requirement and design errors from an evolving software product is one of the primary goals of the development phase.

C. Maintenance Phase :

Software organizations usually spend over 50 percent of their budget to maintain its old software. The documents created in this phase could include the **Software Problem Report (SPR) and Software Change Report (SCR)**. There is no document generated in maintenance phase by the project teams in this paper because the project discussed in this paper is a two semester project. Time is too limited to produce any documents in the phase of software maintenance.

A set of formal documents is created to describe the analysis and activities of each phase of the software life cycle. The major uses of documentation in software development projects are management reporting and communication among project personnel.

2.3 Other Software Models

To develop a software system in life cycle, there are several models to choose from: the **waterfall model, rapid prototyping, systems sculpture, incremental development and RUDE**.

2.3.1 Waterfall Model

The **Waterfall model** is currently the most widely used model in developing systems. It is a classical life cycle model derived from the

hardware model of **requirement, design, fabrication, test and maintenance** . Software models only substitute fabrication to code/debug from the hardware model. It consists of five phases in the life cycle model : **requirements, design, code/debug, test and maintenance**. These five steps are described as cascading activities, hence it is also called a **waterfall model**. The model is illustrated in Figure 2.4

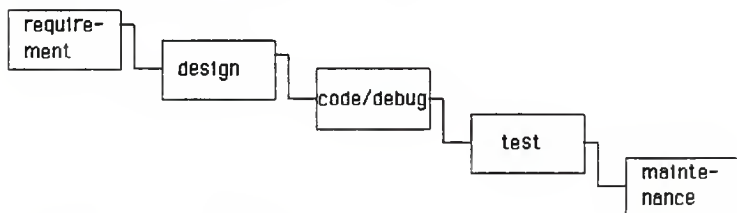


Figure 2.4 **waterfall Model**

The **waterfall model** has been viewed as a succession of independent phases. Each phase required to complete before next phase can start. For instance, a complete set of specifications for a software product is required in the first phase of the life cycle. Sometimes, the users of the system may not really know what they want, and they may be unable to communicate their desires to the project team at the beginning. Moreover, the lack of early end user involvement is another shortage of the **waterfall model**. It may lead to a system that does not meet the user's needs. To overcome these problems, **rapid prototyping** was developed [LUB, 1986].

2.3.2 Rapid Prototyping

A **Prototype** is a model of a software product. It creates an "imitation system" of the system to be developed. It can help end user's early involvement in software development by illustrating input data formats, messages, reports and interactive dialogues for the customer. Once we have developed a **prototype** that illustrates how the system will operate and we have received feedback from the user, the **prototype** should be thrown away and requirement specifications should be written using what was learned from the **prototype**. This eliminates the necessity of writing complete specifications before implementation. Sometimes it is impossible to define the product without some exploratory development. A **Prototype** can give a better understanding of the end user's needs.

For a project that has a **high technical risk**, **rapid prototyping** is recommended to alleviate the risk. **High technical risk** is that the technology that may not support the application [BLU, 1984]. **Prototyping** can be quickly developed and tested certain critical pieces or functions. It provides considerable analysis before the design is complete. On the other hand, if the project has a **high application risk** instead of **high technical risk**, **software sculpture** is recommended [McC, 1982]. **Application risk** is the completed product will not meet the end user's needs. This means that the detailed requirements may not be known until after implementation.

2.3.3 System Sculpture

System sculpture is different from **system architecture**. **System architecture** implies that all requirements for the final product are clear before implementation. It is similar with constructing a building. The blueprint should be generated before construction. In system sculpture, the designer builds the final product through interaction with the user and the model. It starts with a rough functional model, then modifies it step by step. The final system is generated at last. System implementation can improve problem understanding and produce a change in the environment [BLU, 1982].

System sculpture is a dynamic design model that can produce an effective solution corresponding to the real needs. It allows the binding of new requirements into a final product by using application oriented tools during the implementation stage. In contrast the traditional life cycle model (**Waterfall model**), and the life cycle longevity causes the eventual appearance of a final product that can not reflect new requirements and specifications in consequence of the changing of the internal and the external environment.

The final product depends heavily on the skill of the artist. Compared to the classical software model, the experience of the designer is a more important element to the quality of the final system.

2.3.4 Incremental Development

Incremental development establishes the system as a series of versions, each version containing more modules completed and having more capability than the previous version. It is an extension of **rapid prototyping**. The **prototypes** may improve communication between the end user and the developer and enhance the validation process. If the specification does not meet the user's need, then new **prototypes** are produced. As Brooks mentions, it is impossible to "get it right" the first time, and we should always plan to "throw one away" [BRD, 1975]. This cycle is repeated until a validated specification is produced. The process of **incremental development** is presented in the Figure 2.5 [TER, 1986].

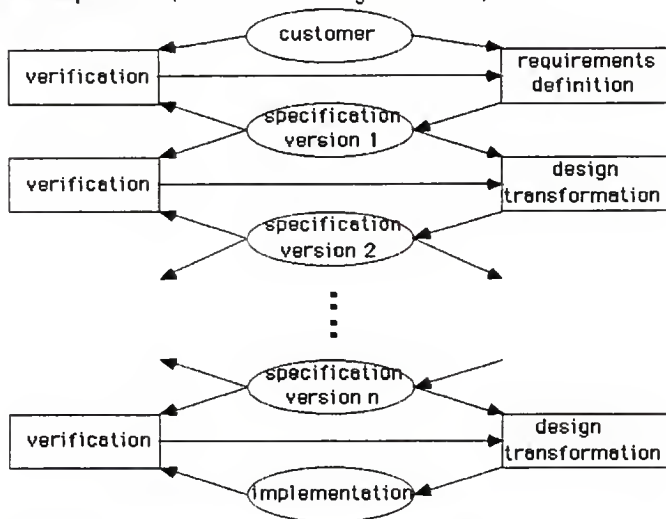


Figure 2.5 : Incremental Program Development

The final components produced by the development satisfies the original specification because each refinement is verified before another is applied.

2.3.5 RUDE

RUDE stands for **Run, Understand, Debug and Edit**. **RUDE** is an incremental, fundamentally exploratory process. It is different from the conventional design **SPIV** [KAT, 1985] :

Specify -> Prove -> Implement -> Verify.

SPIV requires complete specifications and verifications before implementation. This is an all-at-once technique. Contrary to **SPIV**, **RUDE** is adding structured code incrementally.

RUDE paradigm suggests incremental development of an adequate specification. **RUDE** applies incremental development throughout the system life cycle. It is suitable to the nature of Artificial Intelligence (AI) problems. "The development of an AI program can be viewed as an accelerated form of perfective maintenance involving frequent specification changes." [MOS, 1985]. For example, an expert system always deals with incomplete knowledge bases. There is no guarantee that you will actually solve the problem. [HER, 1986]. You must keep searching and learning from a series of trials and errors. To sum up, **RUDE** paradigm is a fundamentally incremental and evolutionary framework for design.

2.4 Summary

The goal of software engineering is to carry out successful software development. To achieve this goal, we must:

- . **Categorize** the different types of systems which are being developed.
- . **Consider** the alternate life cycle models which are applicable to each system. [BLU, 1982]
- . **Identify** and develop the tools and methods which can be applied to a variety of development approaches in order to manage and develop the system successfully.

First at all, we categorize the systems into two classes : **simple system** and **complex system**. The **Simple system** is a system which can write a reasonably complete set of specifications for the software product at the beginning of the life cycle. On the other hand, the **complex system** is a system which can not write an accurate functional specification in the definition phase. The **complex system** can be subdivided into : **technical risk system, application risk system, heuristic system and incremental specification system**.

After categorizing the different types of systems. We can match them with alternative life cycle models in Figure 2.6.

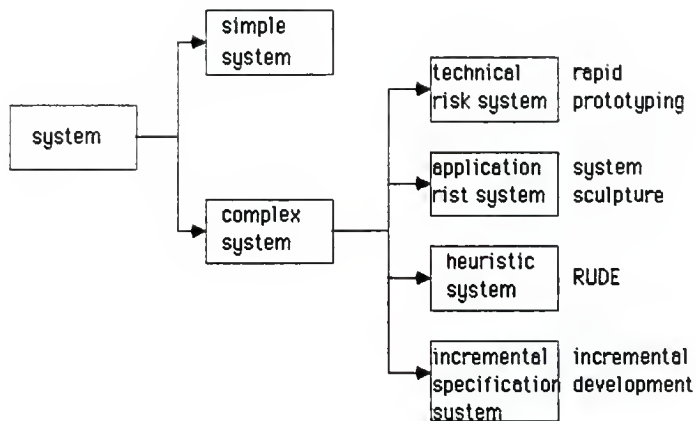


Figure 2.6 : **Alternatives life cycle models** apply to different type of the system.

Actually, we can apply one or more life cycle models in a system depending upon the nature of the system. For example in the development of a complete new system, several dynamic models (**rapid prototyping, incremental development, RUDE, system sculpture**) can be applied simultaneously. New versions of this existing system can be developed by using the **waterfall model**.

Finally, the **SPM** is a tool that can be used to model the different life cycle models by modeling the products of the development process. The **SPM** diagram for alternative life cycle models are presented from Figure 2.7 to Figure 2.9.

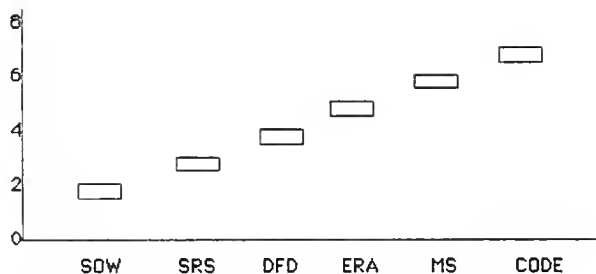


Figure 2.7 : SPM diagram for **waterfall model**

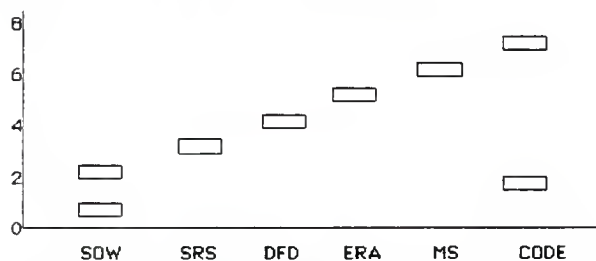


Figure 2.8 : SPM diagram for **rapid prototyping**

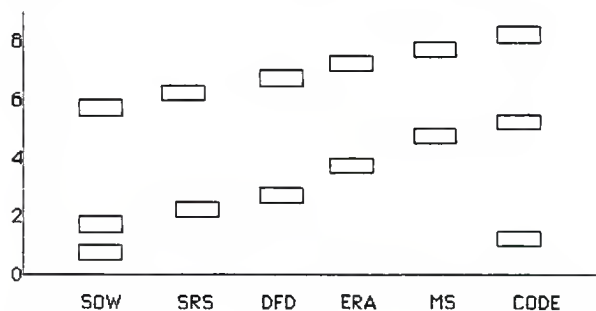


Figure 2.9 : SPM diagram for **incremental development or RUDE**

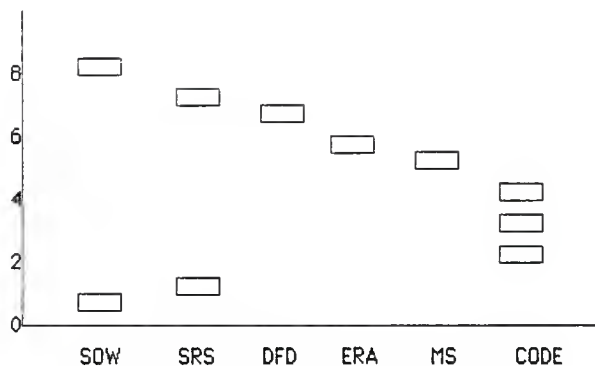


Figure 2.10 : SPM diagram for **system sculpture**

The framework provided by the SPM for quantifying the software process and products is the basis to develop a set of measures that may apply to various approaches in life cycle. This will be expanded in the chapter 3. The SPM, the newly developed model, may be developed into a useful management tool to analyze, control and manage the process of development.

Chapter 3

MEASURES

3.1 Basic Concept

At present, a lot of engineering techniques in software engineering do not correspond to scientific methods. In order to develop better measures, the measurements established in this chapter are based on a framework for quantifying the software process and software products which are provided by the **Software Process Model (SPM)**. This model provides a formal definition of measure and metric which corresponds to the definition used in mathematics. These definitions of **metric** and **pseudo-metric** in mathematics are given as follows:

Definition 3.1: Let H be a set, \mathcal{R} be the set of real numbers, and let f be a function where $f : H \times H \rightarrow \mathcal{R}$. The function f is a metric on H , and H is a metric space if f satisfies the following properties, where $x, y, z \in H$:

- 1) $f(x, x) = 0$
- 2) $f(x, y) = 0$ and if only if $x = y$
- 3) $f(x, y) = f(y, x)$
- 4) $f(x, z) \leq f(x, y) + f(y, z)$

Definition 3.2: The function $f : H \times H \rightarrow \mathcal{R}$ is a pseudo-metric if f satisfies properties 1, 3, and 4.

Hence, a metric $f : H \times H \rightarrow \mathbb{R}$ may be viewed as a function giving the difference or distance between two points. For example, if H is a set of programs, $a \in H$, and $b \in H$, then $f(a,b)$ may represent the difference with respect to some property of the two programs a and b .

From the definition of a metric, we know a metric is a function of the difference between two arguments. In software engineering, the measurements usually do not reveal differences between two variables, for instance software science and McCabe's cyclomatic [McC, 1976]. Therefore, the term "software measure" will be used for quantifiers which take only one argument.

SPM has defined classes of measures that can be applied to **software document version, document histories, and the entire project** according to the above mathematical definitions. These definitions are listed in the following:

1. Software document version

-- Software measure

Definition 3.3: Let V be a set of document versions. A function f where $f : V \rightarrow \mathbb{R}$ is a document measure.

--Software metric

Definition 3.4: A **software metric** is a metric with a metric space consisting of a set of document histories. A **software**

pseudo-metric is a pseudo-metric over a domain consisting of a set of document versions.

For example, a **software pseudo-metric** is the absolute value of the time interval between first version and last version of a **data flow diagram (DFD)**.

2. Software document histories

--Document measure

Definition 3.5: Let H be a specified set of document histories. A document measure is a function $f: H \rightarrow \mathbb{R}$.

An example of a **document measure** is the average value of effort among the different versions of a document.

--Document metric

Definition 3.6: A **document metric** is a metric with a metric space consisting of a set of document histories. A **document pseudo-metric** is a pseudo-metric over a domain consisting of a set of document histories.

For instance, a example of a **document pseudo-metric** is the absolute value of the difference between the number of specification versions and the number of implementation versions.

3. Software project

--Project measure

Definition 3.7: Let P be a set of SPMs for a number of projects. A project measure is a function $f: P \rightarrow \mathbb{R}$.

-- Project metric

Definition 3.8: A **project metric** is a metric with a metric space consisting of a set of SPMs. A **project pseudo-metric** is a pseudo-metric over a domain consisting of a set of document SPMs.

In general, the term "**measure**" is used to describe a function with one argument. And, the term "**metric**" is used to be a function with two arguments.

The measures developed in this thesis are software project measures.

3.2 Measure Development Paradigm.

Lord Kelvin, the great physical scientist, said "If a phenomenon cannot be measured and/or described in a quantitative fashion, the phenomenon is not well understood.". It is often heard that: "I am taller than you.", "He is fatter than his brother." or "You look younger than your age.", etc.. From these conversations, it is still difficult to get a clear picture unless we know exactly how much taller, fatter or younger. This is the same in software engineering; quantitative measurements of the activities in the software life cycle can help us to understand, manage and control the development process.

Several software measures have been proposed. These measures mainly concentrate on the source code in the development phase of life cycle. For example, Halstead's software measures are a source-code oriented complexity measure; a set of primitive measures, such as the number of operators and operands in the program and the total number of operators and operands occurrences are used. These measures are derived after the code is generated [HEL, 1977].

McCabe's complexity measure can only be applied after detailed design is completed. The complexity measure is developed using the number of linearly independent control paths in a program [McC, 1976]. This can not assist us in comprehensive understanding of the whole development process in the early life cycle.

In order to quantify the movements of a software project in the life cycle, a set of measures will be developed in the following sections. The measurements of waste, progress, stability and effort can reflect characteristics of the development process, thus enabling us to describe the development process more precisely.

To verify these software measures, there is a common paradigm in the software engineering area presented. This common paradigm can be described as sequence steps. These steps are given in the following:

1. A small subset of the domain of programs is selected. The programs in

this subset are ranked by experienced computer scientists. The rank can be divided into "equal to", "less than" and "greater than" according to the characteristics of the measures.

2. The measure can be applied to this small subset of programs, then this measure will assign a real number to each program in this subset.
3. The software measure will be considered a worthwhile measure if the values assigned to this subset correspond to the rank given by the computer scientists. In other words the measure is considered worthwhile if it preserves the ordering assigned by the computer scientists.
4. If the measure preserves an ordering on this small subset of programs then it will preserve the ordering on all programs. This is an assumption made by this paradigm.

To generate a useful measure, this common paradigm must be changed. A new paradigm must be developed.

The new paradigm to develop software measures is listed as follows:

1. A domain must be chosen. In our example, the domain is the set of all SPM diagrams.
2. The assumptions for each measure are made which are based on the objects in the domain.

3. The relations are generated by creating a set of transformation functions in the domain. The "equal to", "greater than" and "less than" will be assigned to these transformation functions.
4. The ordering relations for the transformation functions will be categorized into partial and equivalence ordering.
5. The measures will be applied into the domain D , the real number will be assigned to each object in D' . Direct proofs will be used to prove that the ordering relations not only exist in the small subset of the domain but also exists in the whole software domain.

A software measure can be further defined as the follows:

Definition 3.12: Let m be a **software measure** which is a function from D to R , i.e., $m: D \rightarrow R$, iff m preserves the ordering which is implied by the assumptions.

The measurements of **waste, effort, stability, and progress** developed from the **SPM** can be used to express quantitatively these characteristics of the development process. The relations for the software domain are varying in terms of **waste, effort, progress and stability**. These measures, relations and ordering relations will be introduced and discussed in the following sections.

3.3 waste Measure

The idealized SPM diagram for the development process is similar to the **waterfall model**, with respect to the **waste**. The **waterfall model** requires well-defined input information, utilizes well-defined processes, and results in a well-defined product in each phase. No revision, are shown in the waterfall model. Hence, there appears to be no extra time and effort being put in. In other words, no waste will be generated, the waste is generated only when revised documents are produced.

The assumptions for the **waste** are as follows :

1. The **waste** is zero for every initial document.
2. The **waste** is generated as long as a revision is added to documents.
3. The longer the time to revise a document, the more **waste** is generated.
4. The **waste** mainly depends on the time to generate a revision rather than the number of the revised documents. For instance, the waste for (1) and (2) in Figure 3.1 are the same disregarding the fact that the number of revision for (2) is larger than the number of revisions for (1).

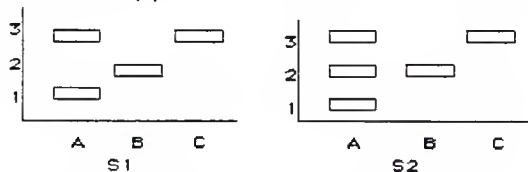


Figure 3.1: A simple example for waste

The **waste** for the software domain D can be denoted by **waste(k)**.

waste(k) is a measure expressed as follows:

$$\text{Waste}(k) = \sum_{f=1}^m \sum_{i=1}^n (\text{tdf}(i) - \text{tdf}(i-1))$$

where 1. **tdf(i)** is the completion time for the latest version of document f,

where i is the version number.

2. **tdf(i-1)** is the completion time for the document generated

before the latest document with the same document type f. If

(i-1)=0 then **tdf(0) := tdf(1)**.

A few examples in Figure 3.2 may help to illustrate how to quantify the waste.

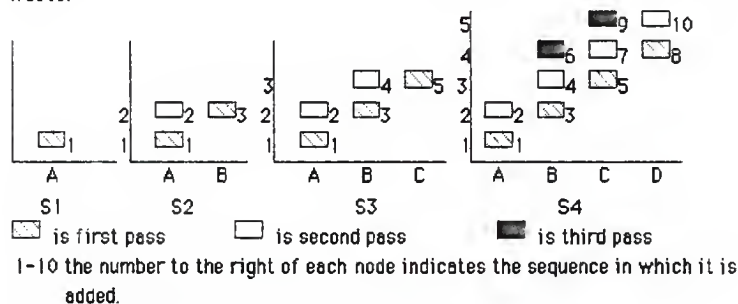


Figure 3.2 An example of the process of software development

For (S1), **waste(S1) = 1-1=0;**

For (S2) **waste(S2) = ((1-1) + (2-1)) + ((2-2)) = 1;**

For (S3) **waste(S3) = ((1-1) + (2-1)) + ((2-2) + (3-2)) + ((3-3)) = 2;**

For (S4) **waste(S4) = ((1-1) + (2-1)) + ((2-2) + (3-2)+(4-3)) + ((3-3) + (4-3) + (5-4)) + ((4-4)+(5-4)) = 6.**

To justify that the waste to be a worthwhile measure, the following steps will be taken:

1. Determine the relations which exist on the documents.
2. Decide the ordering relations.
3. Prove that the measurement waste preserves the ordering.

-- Determine the relations:

The relations among documents can be expressed as transformation functions for the waste. Let $SPM(x)$ be an SPM diagram, then the transformation functions are the following:

T_0 : Add an initial document. $T_0(SPM(x))$ will be the $SPM(x)$ diagram with another initial document added. The first assumption implies that $waste(SPM(x)) = waste(T_0(SPM(x)))$. The relation for T_0 is ' $=_w$ '.

T_1 : Add a revision to a document. $T_1(SPM(x))$ be the $SPM(x)$ diagram with another revised document added. The second Assumption implies that $waste(SPM(x)) < waste(T_1(SPM(x)))$. The relation for T_1 is ' $<_w$ '.

The equivalence function, T_0 maps domain D to D' . D' is a software domain which consists of sets of documents. T_1 establishes a partial ordering on D' .

--Decide the ordering relation

The ordering relation for T_0 and T_1 can be determined by the following proof:

The ordering relations for T_0 :

Let $=_w$ be the relation for T_0 , $x =_w y$ if $waste(SPM(x)) =_w waste(T_0(SPM(y)))$.

Proof : Let x, y and z be documents in D' .

then i) $=_w$ is reflexive on D' , it means $\text{waste}(\text{SPM}(x)) =_w \text{waste}(\text{To}(\text{SPM}(y))) \Rightarrow \text{waste}(\text{SPM}(x)) =_w \text{waste}(\text{To}(\text{SPM}(x)))$.

ii) $=_w$ is symmetric on D' , it means $\text{waste}(\text{SPM}(x)) =_w \text{waste}(\text{To}(\text{SPM}(y))) \Rightarrow \text{waste}(\text{To}(\text{SPM}(y))) =_w \text{waste}(\text{SPM}(x))$.

iii) $=_w$ is transitive on D' , it means $\text{waste}(\text{SPM}(x)) =_w \text{waste}(\text{To}(\text{SPM}(y)))$ and $\text{waste}(\text{SPM}(y)) =_w \text{waste}(\text{To}(\text{SPM}(z))) \Rightarrow \text{waste}(\text{SPM}(x)) =_w \text{waste}(\text{To}(\text{SPM}(z)))$.

The relation $=_w$ is an equivalence relation for To . Since $=_w$ meets the conditions for the equivalence relation : **reflexivity, symmetry and transitivity**. Note, for document x and y both have the same amount of waste, $\text{waste}(\text{SPM}(x)) =_w \text{waste}(\text{To}(\text{SPM}(y)))$, but x and y are not the same document. Let D' be the set of equivalence sets defined by $=_w$. Thus, the order pair $(D', =_w)$ for To is an equivalence set.

The ordering relation for Ti :

Let $<_w$ be a relation for the Ti , $x <_w y$ if $\text{waste}(\text{SPM}(x)) <_w \text{waste}(\text{Ti}(\text{SPM}(y)))$.

Proof : Let x, y and z be documents in D' .

then i) $\text{waste}(\text{SPM}(x)) <_w \text{waste}(\text{Ti}(\text{SPM}(y)))$ and $\text{waste}(\text{Ti}(\text{SPM}(x))) <_w \text{waste}(\text{Ti}(\text{SPM}(z)))$ which implies $\text{waste}(\text{SPM}(x)) <_w \text{waste}(\text{Ti}(\text{SPM}(z)))$. It satisfies the property of transitivity.

ii) In part i), if we had both $waste(SPM(x)) <_w waste(T_1(SPM(y)))$ and $waste(SPM(x)) =_w waste(T_1(SPM(x)))$ then we would have $waste(SPM(x)) <_w waste(T_1(SPM(x)))$ contradicting irreflexivity for partial ordering. And if both $waste(SPM(x)) <_w waste(T_1(SPM(y)))$ and $waste(SPM(y)) <_w waste(T_1(SPM(x)))$, then by transitively $waste(SPM(x)) <_w waste(T_1(SPM(x)))$, again contradicting irreflexivity. For T_1 , $waste(SPM(x)) <_w waste(T_1(SPM(y)))$ is irreflexive, since it is never the case that $waste(SPM(x)) <_w waste(T_1(SPM(x)))$. Therefore T_1 meets the conditions of transitive and irreflexive for partial ordering [SAH, 1981]. Let D' be a partial ordering sets defined by $<_w$. Thus, the order pair $(D', <_w)$ for T_1 is a partial ordering set.

From the above proofs, we know the ordering relations for T_0 and T_1 are the equivalence and partial ordering. For the software domain D , we should combine these two transformation functions together, because the software domain consists of both a set of initial documents and a set of revised documents.

The mapping function: $waste: D \rightarrow \mathcal{R}$ will be proved by direct proof to find out whether it preserves the ordering relation on D . The notation 'LHS' and 'RHS' will be used to represent the equation on the left-hand side equal to the equation on the right-hand side. The notation '(RHS)' is used to define the change after calculation of right-hand side equation.

The ordering relation for To

Prove $waste(SPM(x)) = waste(To(To(To \dots (SPM(x) \dots)))$

1. Basis:

Let be and SPM diagram with $k-1$ document types $SPM(x)$, then

$waste(SPM(x)) = waste(To(SPM(x))) \Rightarrow$

$$\sum_{i=1}^{k-1} \sum_{j=1}^n (tdf(i) - tdf(i-1)) = w \sum_{i=1}^{k-1} \sum_{j=1}^n (tdf(i) - tdf(i-1)) + (tdk(1) - tdk(0))$$

Prove $LHS = w \cdot RHS = LHS + (tdk(1) - tdk(0))$

Since $tdk(1) - tdk(0) = w \cdot tdk(1) - tdk(1) = 0$

Then $LHS = w \cdot (RHS) \quad //$

2. Hypothesis:

Assume we have $m-1$ documents and the number of versions of document f are n_f . Let $waste(SPM(x)) = waste(To(To(To \dots (SPM(x) \dots)))$ is true, then

$$\sum_{i=1}^{m-1} \sum_{j=1}^{n_f} (tdf(i) - tdf(i-1)) = w \sum_{i=1}^{m-1} \sum_{j=1}^{n_f} (tdf(i) - tdf(i-1)) + (tdm(1) - tdm(0))$$

Prove $LHS = w \cdot RHS + (tdm(1) - tdm(0))$

Since $tdm(1) - tdm(0) = w \cdot tdm(1) - tdm(1) = 0$

Then $LHS = w \cdot (RHS) \quad //$

(From basis)

3. Induction:

Assume $\text{waste}(\text{To}(\text{SPM}(x))) = \text{waste}(\text{To}(\text{To}(\text{To}(\dots(\text{SPM}(x))\dots)))$, let $\text{waste}(\text{SPM}(x)) = \text{waste}(\text{To}(\text{SPM}(x)))$, then

$$\sum_{i=1}^{m-1} \sum_{j=1}^n (\text{tdf}(i) - \text{tdf}(i-1)) + (\text{tdm}(1) - \text{tdm}(0)) = \sum_{i=1}^{m-1} \sum_{j=1}^n (\text{tdf}(i) - \text{tdf}(i-1)) + (\text{tdm}(1) - \text{tdm}(0)) + (\text{tdm}+1(1) - \text{tdm}+1(0))$$

Prove LHS = w RHS + (tdm+1(1) - tdm+1(0))

Since tdm+1(0) = w tdm+1(1)

(From assumption)

Then tdm+1(1) - tdm+1(0) = w tdm+1(1) - tdm+1(1) = 0

LHS = w (RHS)

Therefore $\text{waste}(\text{SPM}(D)) = \text{waste}(\text{To}(\text{To}(\text{To}(\dots(\text{SPM}(x))\dots))) //$

The relation for T_0 is = w according to above direct proof.

The relation for T_1

Prove $\text{waste}(\text{SPM}(x)) < \text{waste}(T_1(T_1(T_1(\dots(\text{SPM}(x))\dots)))$

1. Basis:

Let be a SPM diagram $\text{SPM}(x)$, then $\text{waste}(\text{SPM}(x)) < \text{waste}(T_1(\text{SPM}(x)))$

$$\sum_{i=1}^m \sum_{j=1}^n (\text{tdf}(i) - \text{tdf}(i-1)) < \sum_{i=1}^m \sum_{j=1}^n (\text{tdf}(i) - \text{tdf}(i-1)) + (\text{tdf}(i) - \text{tdf}(i-1))$$

Prove LHS < w RHS + (tdf(1) - tdf(1-1))

(tdf(i) - tdf(i-1)) In LHS and RHS will be summed until the first revised document created).

Since $tdf(i) > w \cdot tdf(i-1)$
 (From assumption 2)
 Then $tdr(2) - tdr(1) > 0$
 $LHS < w \cdot (RHS)' \quad //$

2. Hypothesis:

Assume for some documents f , nr is increased by 1, let $waste(SPM(x)) < waste(T_1(T_1(T_1 \dots (SPM(x)) \dots)))$ is true, then

$$\sum_{f=1}^m \sum_{i=1}^{nf} (tdf(i) - tdf(i-1)) < w \sum_{f=1}^m \sum_{i=1}^{nf} (tdf(i) - tdf(i-1)) + (tdf(nf) - tdf(nf-1))$$

Prove $LHS < w \cdot RHS + (tdr(nr) - tdr(nr-1))$
 Since $tdr(nr) > w \cdot tdr(nr-1)$
 (From basis)
 Then $tdr(nr) - (tdr(nr-1)) > 0$
 $LHS < w \cdot (RHS)' \quad //$

3. Induction:

Assume $waste(T_1(SPM(x))) = waste(T_1(T_1 \dots (SPM(x)) \dots))$, let $waste(SPM(x)) < waste(T_1(T_1(SPM(x))))$, let nr is increased by 1 for some documents f , then

$$\sum_{f=1}^m \sum_{i=1}^{nf} (tdf(i) - tdf(i-1)) + (tdf(nf) - tdf(nf-1)) < w \sum_{f=1}^m \sum_{i=1}^{nf} (tdf(i) - tdf(i-1)) + (tdr(nr) - tdr(nr-1)) + (tdr(nr) - tdr(nr-1))$$

Prove $LHS < w \cdot RHS + (tdr(nr) - tdr(nr-1))$
 Since $tdr(nr) > w \cdot (tdr(nr-1))$
 (From hypothesis)
 Then $tdr(nr) - (tdr(nr-1)) > 0$
 $LHS < w \cdot (RHS)'$

Therefore $\text{waste}(\text{SPM}(x)) = \text{waste}(T_1(T_1(T_1 \dots (\text{SPM}(x)) \dots)))$ //

From the above direct proof, we can conclude the relation for T_0 is $= w$ and T_1 is $< w$. The waste measure can be defined as follows:

Definition 3.14: $waste$ is a function from D to R , i.e., $waste: D \rightarrow R$ and it preserves the ordering \leq_w on D . Let $x \leq_w y$ iff $waste(\text{SPM}(x)) = w \cdot waste(T_0(\text{SPM}(y)))$ and $waste(\text{SPM}(x)) < w \cdot waste(T_1(\text{SPM}(y)))$

The waste measure is not suitable for the waterfall model. Waterfall model is composed of a member of initial documents. In the modern computer society, programs have become more and more complex, the early static "waterfall" model was no longer satisfactory to represent this complexity. Making a change during the process of development became very common. Properly updating the document to reflect modifications is required to maintain the system in the life cycle.

3.4 Effort Measure

The measure of effort is to reflect the amount an individual job contributes to the entire job at any particular time point. Total effort is the sum of the ratio of an individual job(a document) to the entire job.

The basis of measurement for effort is the completion and starting time of the document. A document serves as a milestone in the software development. A portion of a job is done when a document is produced. The time to add a new document is related to the effort put into the job. This can be calculated by : $T(d,d') = |t(d') - t(d)|$ where d is a document which is produced before the latest one and d' is the latest document. The time for the entire job can be defined as **development Duration(T)**. The definition for the development duration is:

Definition 3.15: Development Duration is the elapsed time in months during which development effort proceeds.

For those documents which have the same completion time as previous document, there is no additional effort to be added. It means that if several documents are generated at same time, the time for the first document with the same completion time will be used. For the rest of the documents with the same completion time, there is no effort to be added. This assumption is based on a project done by the same team members throughout the life cycle. If a project is divided into several phases and each phase is done by different groups, then this measure is not applicable.

The assumptions for the **EFFORT** are:

1. Adding a document takes effort.
2. Adding a document with the same completion time as the previous document takes no additional effort.
3. The effort is relative to the entire job. The more time to complete a

job, the more effort is taken.

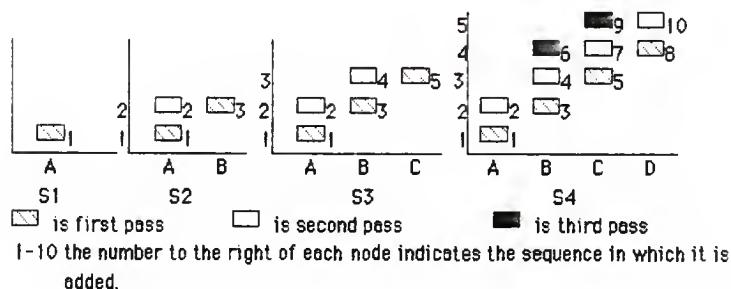
The **effort** for the software domain D can be denoted by **Effort (x)**. **Effort (x)** is used to determine the effort used in the software development x. **Effort(x)** is a measure expressed as follows:

$$\text{Effort}(x) = \sum_{i=1}^m \sum_{j=1}^n \frac{\text{tdf}(i) - \text{tdf}(i-1)}{\text{tdf}(i)}$$

where 1. **tdf(i)** is the completion time for the latest document of type f.

2. **tdk(i-1)** is the completion time for the document before the latest document. The document type for these two sequence documents could be different, i.e., if $\text{df}, \text{dk} \in D$ then $\text{df} = \text{dk}$ or $\text{df} \neq \text{dk}$ and if $(i-1)=0$ then $\text{tdk}(0)=0$.

The measurement for the effort can be illustrated by the same example used previously in Figure 3.2.



For (S1) **Effort (S1)** = $(1-0)/1 = 1$;

For (S2) **Effort (S2)** = $((1-0)/1 + (2-1)/2) + (2-2)/2 = 1.5$;

$$\text{For (S3) Effort (S3)} = ((1-0)/1 + (2-1)/2) + ((2-2)/2 + (3-2)/3) + ((3-3)/3) \\ = 1.83;$$

$$\text{For (S4) Effort (S4)} = ((1-0)/1 + (2-1)/2) + ((2-2)/2 + (3-2)/3 + (4-3)/4) + \\ ((3-3)/3 + (4-4)/4 + (5-4)/5) + ((4-4)/4 + (5-5)/5) \\ = 2.28.$$

To verify that **Effort** is a worthwhile measure, the same steps will be taken as previously mentioned.

--Determine the relations:

The relations among documents can be expressed by the following transformation functions for the **effort**:

To: Add a document with different completion time to previous documents.

$To(SPM(x))$ will be the $SPM(x)$ diagram with another document with different completion time to previous one added. The first assumption implies that $Effort(SPM(x)) < Effort(To(SPM(x)))$. The relation for **To** is ' $<e$ '.

Ti: Add a document with same completion time as previous documents.

$Ti(SPM(x))$ be the $SPM(x)$ diagram with another document with same completion time as previous documents added. The second assumption implies that $Effort(SPM(x)) = Effort(Ti(SPM(x)))$. The relation for **Ti** is ' $=e$ '.

The equivalence function, **Ti** maps domain D to D' . D' is a software domain which consists of sets of documents. **To** establishes a partial ordering on D' .

--Decide the ordering relation

The ordering relation for T_0 and T_1 can be determined by following proof.

The ordering relation for T_0 :

Let $<_E$ be a relation for the T_0 , $x <_E y$ if $\text{Effort}(x) <_E \text{Effort}(T_0(y))$.

Proof : Let x, y and z be documents in D' .

then i) $\text{Effort}(x) <_E \text{Effort}(T_0(y))$ and $\text{Effort}(y) <_E \text{Effort}(T_0(z))$

which implies $\text{Effort}(x) <_E \text{Effort}(T_0(z))$. It satisfies the property of **transitive**.

ii) In part i), if we had both $\text{Effort}(x) <_E \text{Effort}(T_0(y))$ and $\text{Effort}(x) <_E \text{Effort}(T_0(x))$, then we would have $\text{Effort}(x) <_E \text{Effort}(T_0(x))$ contradicting **irreflexivity**. And if both $\text{Effort}(x) <_E \text{Effort}(T_0(y))$ and $\text{Effort}(y) <_E \text{Effort}(T_0(x))$, then by transitively $\text{Effort}(x) <_E \text{Effort}(T_0(x))$, again contradicting **irreflexivity**. $\text{Effort}(x) <_E \text{Effort}(T_0(y))$ is **irreflexive**, it is never the case that $\text{Effort}(x) <_E \text{Effort}(T_0(x))$. It meets the conditions of **transitive** and **irreflexive** for the partial ordering. Thus, the order pair $(D, <_E)$ for T_0 is a partial ordering set.

The ordering relation for T_1 :

Let $=_E$ be the relation for T_1 , $x =_E y$ if $\text{Effort}(x) =_E \text{Effort}(T_1(y))$.

Proof : Let x, y and z be documents in D' .

then i) $=_E$ is reflexive on D' , it means $\text{Effort}(x) =_E \text{Effort}(T_1(y)) \Rightarrow$

$\text{Effort}(x) =_E \text{Effort}(T_1(x))$

ii) $=_E$ is symmetric on D' , it means $\text{Effort}(\text{SPM}(x)) =_E$

$\text{Effort}(T_0(\text{SPM}(y))) \Rightarrow \text{Effort}(T_0(\text{SPM}(y))) =_E \text{Effort}(\text{SPM}(x))$.

iii) \neq is transitive on D' , it means $\text{Effort}(x) \neq \text{Effort}(T_1(y))$ and $\text{Effort}(y) \neq \text{Effort}(T_1(z)) \Rightarrow \text{Effort}(x) \neq \text{Effort}(T_1(z))$.
 Because of T_1 is adding a document with same completion time as previous documents. No effort will be added under this condition.

Thus \neq is an equivalence relation for T_1 . " \neq " satisfies the properties of equivalence relation: **reflexity**, **symmetry** and **transitivity**. Thus, the order pair (D', \neq) for T_1 is an equivalent set.

From the above proofs, we know the ordering relations for T_0 and T_1 are the equivalence and partial ordering. For the software domain D , we should combine these two transformation functions together, because the software domain consists of both a set of documents with the same completion time and a set of documents with different completion time.

The mapping function : $\text{Effort} : D \rightarrow \mathbb{R}$ will be proved by the direct proof to find out whether it preserves the ordering relation on D' . The notation 'LHS' and 'RHS' will be used to represent the equation on the left-hand side equal to the equation on the right-hand side. The notation '(RHS)' is used to define the change after calculation of right-hand side equation.

The procedure for the **direct proof** is expressed as below:

The ordering relation for T_0

Prove $\text{Effort}(\text{SPM}(x)) < \text{Effort}(T_0(T_0(\dots(\text{SPM}(x))\dots)))$

1. Basis:

Let $n=1$ be the initial state of the SPM(x), then

$\text{Effort}(\text{SPM}(x)) < \text{Effort}(\text{To}(\text{SPM}(x))) \Rightarrow$

$$\sum_{i=1}^n \sum_{j=1}^n \frac{\text{tdf}(i) - \text{tdk}(i-1)}{\text{tdf}(i)} < \sum_{i=1}^m \sum_{j=1}^n \frac{\text{tdf}(i) - \text{tdk}(i-1)}{\text{tdf}(i)} + \frac{\text{tdf}(i) - \text{tdk}(i-1)}{\text{tdf}(i)}$$

Prove $\text{LHS} < \text{RHS} + ((\text{tdf}(i) - \text{tdk}(i-1))/\text{tdf}(i))$

(The equation on the LHS and RHS will be summed until the first document with different completion time is created.)

Since $\text{tdf}(i) \geq \text{tdf}(i-1)$

(From assumption 1 and $\text{Effort}(D)$)

Then $(\text{tdf}(i) - \text{tdk}(i-1)) / \text{tdf}(i) > 0$

$\text{LHS} < (\text{RHS})' \quad //$

2. Hypothesis:

Assume we have $m-1$ documents and the number of versions of document f

are n_f . Let $\text{Effort}(\text{SPM}(x)) < \text{Effort}(\text{To}(\text{To}(\dots(\text{SPM}(x)\dots))))$ is true, then

$$\sum_{i=1}^{m-1} \sum_{j=1}^{n_f} \frac{\text{tdf}(i) - \text{tdk}(i-1)}{\text{tdf}(i)} < \sum_{i=1}^{m-1} \sum_{j=1}^{n_f} \frac{\text{tdf}(i) - \text{tdk}(i-1)}{\text{tdf}(i)} + \frac{\text{tdf}(i) - \text{tdk}(i-1)}{\text{tdf}(i)}$$

Prove $\text{LHS} < \text{RHS} + ((\text{tdf}(i) - \text{tdk}(i-1))/\text{tdf}(i))$

Since $\text{tdf}(i) \geq \text{tdk}(i-1)$

(From basis)

Then $((\text{tdf}(i) - \text{tdk}(i-1))/\text{tdf}(i)) > 0$

$\text{LHS} < (\text{RHS})' \quad //$

3. Induction:

Assume $\text{Effort}(\text{To}(\text{SPM}(x))) = \text{Effort}(\text{To}(\text{To}(\dots(\text{SPM}(x)\dots))))$, let $\text{Effort}(\text{SPM}(x))$

$< \text{Effort}(\text{To}(\text{To}(\text{SPM}(x))))$, let n_f be increased by 1 for some documents f , then

Suppose $\text{Effort}(\text{SPM}(x)) = \text{Effort}(\text{To}(\text{SPM}(x)))$, then

$$\sum_{i=1}^{m-1} \sum_{j=1}^n \frac{\text{tdf}(i) - \text{tdk}(i-1)}{\text{tdf}(i)} + \frac{\text{tdf}(i) - \text{tdk}(i-1)}{\text{tdf}(i)} = E \sum_{i=1}^{m-1} \sum_{j=1}^n \frac{\text{tdf}(i) - \text{tdk}(i-1)}{\text{tdf}(i)}$$

$$+ (\text{tdf}(i) - \text{tdk}(i-1)) / \text{tdf}(i) + (\text{tdf}(i) - \text{tdk}(i-1)) / \text{tdf}(i)$$

$$\text{Prove} \quad \text{LHS} < \text{RHS} + (\text{tdf}(i) - \text{tdk}(i-1))$$

$$\text{Since} \quad \text{tdf}(1) < \text{tdk}(i-1)$$

(From assumption 1)

$$\text{Then} \quad (\text{tdf}(i) - \text{tdk}(i-1)) / \text{tdf}(1) > 0$$

$$\text{LHS} < (\text{RHS})'$$

$$\text{Therefore} \quad \text{Effort}(\text{SPM}(D)) < \text{Effort}(\text{To}(\text{To}(\text{To}(\dots(\text{SPM}(x))\dots))) \quad //$$

The relation for To is $<$ according to above direct proof.

The relation for T1

$$\text{Prove} \quad \text{Effort}(\text{SPM}(x)) = \text{Effort}(T_1(T_1(T_1(\dots(\text{SPM}(x))\dots)))$$

1. Basis:

Let $n=1$ be the initial state of the $\text{SPM}(x)$, then

$$\text{Effort}(\text{SPM}(x)) = \text{Effort}(T_1(\text{SPM}(x)))$$

$$\sum_{i=1}^m \sum_{j=1}^n \frac{\text{tdf}(i) - \text{tdk}(i-1)}{\text{tdf}(i)} = E \sum_{i=1}^m \sum_{j=1}^n \frac{\text{tdf}(i) - \text{tdk}(i-1)}{\text{tdf}(i)} + \frac{\text{tdf}(i) - \text{tdk}(i-1)}{\text{tdf}(i)}$$

$$\text{Prove} \quad \text{LHS} = \text{RHS} + (\text{tdf}(i) - \text{tdk}(i-1)) / \text{tdf}(i)$$

(The equation on the LHS and RHS will be summed until first document with same completion time created.)

$$\text{Since} \quad \text{tdf}(i) = \text{tdk}(i-1)$$

(from assumption 2, we know both documents have the same completion time)

$$\text{Then} \quad \text{tdf}(i) - \text{tdk}(i-1) = 0$$

$$\text{LHS} = (\text{RHS})' \quad //$$

2. Hypothesis:

Assume for some documents f , nr is increased by 1,

let $\text{Effort}(\text{SPM}(x)) = \text{Effort}(T_1(T_1(T_1 \dots (\text{SPM}(x)) \dots)))$ is true, then

$$\sum_{f=1}^{n-1} \sum_{i=1}^{nf} \frac{tdf(i) - tdk(i-1)}{tdf(i)} = E \sum_{f=1}^{n-1} \sum_{i=1}^{nf} \frac{tdf(i) - tdk(i-1)}{tdf(i)} + \frac{tdf(i) - tdk(i-1)}{tdf(i)}$$

$$\text{Prove} \quad \text{LHS} = E \text{ RHS} + (tdf(i) - tdk(i-1))/tdf(i)$$

$$\text{Since} \quad tdf(i) = E \quad tdk(i-1)$$

(from basis)

$$\text{Then} \quad (tdf(i) - tdk(i-1))/tdf(i) = 0$$

$$\text{LHS} = E \text{ (RHS)}' \quad //$$

3. Induction:

Let nr be increased by 1 for some documents f , then

$$\text{Effort}(\text{SPM}(x)) < \text{Effort}(T_1(T_1(\text{SPM}(x))))$$

$$\sum_{f=1}^{n-1} \sum_{i=1}^{nf} \frac{tdf(i) - tdk(i-1)}{tdf(i)} + \frac{tdf(i) - tdk(i-1)}{tdf(i)} = E \sum_{f=1}^{n-1} \sum_{i=1}^{nf} \frac{tdf(i) - tdk(i-1)}{tdf(i)}$$

$$+ (tdf(i) - tdk(i-1))/tdf(i) + (tdf(i) - tdk(i-1))/tdf(i)$$

$$\text{Prove} \quad \text{LHS} = E \text{ RHS} + (tdf(i) - tdk(i-1))/tdf(i)$$

$$\text{Since} \quad tdf(i) = E \quad (tdk(i-1))$$

(From hypothesis)

$$\text{Then} \quad (tdf(i) - tdk(i-1))/tdf(i) = 0$$

$$\text{LHS} = E \text{ (RHS)}'$$

$$\text{Therefore} \quad \text{Effort}(\text{SPM}(x)) = \text{Effort}(T_1(T_1(T_1 \dots (\text{SPM}(x)) \dots))) \quad //$$

From the above direct proof, we can conclude the relation for T_0 is $=E$ and T_1 is $<E$. The Effort measure can be defined as follows:

Definition 3.16: **Effort** is a function from D to \mathcal{R} , i.e., **Effort**: $D \rightarrow \mathcal{R}$ and it preserves the ordering \leq_E on D . Let $x \leq_E y$ iff $\text{Effort}(x) \leq_E \text{Effort}(T_0(y))$ and $\text{Effort}(x) =_E \text{Effort}(T_i(y))$

The measurement for effort can be applied to any development model. It reflects the contribution of individual jobs to the entire job.

3.5 Stability Measure

The stability for the software development is constant, meaning it has a value of one for every initial document. The stability decreases whenever the number of revised documents increases. In other words, the value of stability heavily depends upon the number of revised documents. The more revised documents, the less stability it has. Also, the more time to modify a document, the less stability it has. The time to modify a document is another important determinant for the stability. The time to update the document also indicates the requirements for the modification. A minor change for the system only needs a small amount of time. On the other hand, a major change for the system requires a large amount of time. Hence, the stability is declining when the time to revise the document is increasing.

The assumptions for the **STABILITY** are:

1. The **stability** is constant ($=1$) for the initial document in the software life cycle.

2. The more revisions to documents, the less **stability** it has.
3. The more time needed to revise a document, the less **stability** it has.

The **stability** function can be denoted by **Stability(x)** which is used to determine the stability of the software development x. The **Stability(x)** measure is expressed as follows:

$$\text{Stability}(x) = (1 + \sum_{f=1}^m \sum_{i=2}^n \frac{\text{tdf}(1)}{\text{tdf}(i) \cdot j}) / (j' + 1)$$

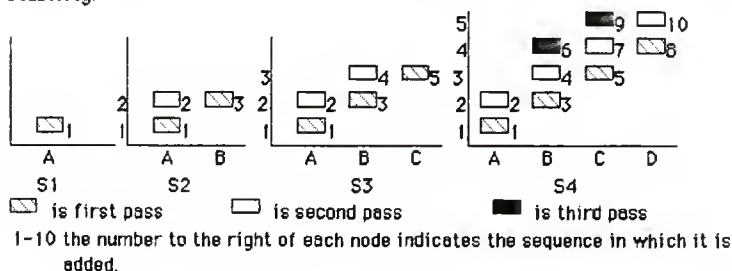
where 1. **tdf(1)** is the completion time for the initial version of document f.

2. **tdf(i)** is the completion time for latest version of document f.

3. **j** is the number of revised documents (not including initial) added up to this document. **j'** is the total number of revisions in the SPM.

4. **i** is the version number. Notice the starting value for **i** is 2 in the summation. When **i=1**, the document is an initial document. Stability for an initial document should be unchanged.

We use the same examples from Figure 3.2 to figure out how to quantify the stability.



For (S1), $\text{Stability}(S1) = (1+0)/(0+1)=1$;

For (S2), $\text{Stability}(S2) = (1+0.5)/(1+1) = 0.75$;

For (S3), $\text{Stability}(S3) = ((1+0.5)+0.33) / (2+1) = 0.61$;

For (S4), $\text{Stability}(S4) = ((1+0.5) + (0.33+0.17) + (0.19+0.12) + 0.13) / (6+1)$
 $= 0.35$.

To verify that **Stability** is a worthwhile measure, the same steps will be taken as previously mentioned.

--Determine the relations:

The relations among documents can be expressed as the following transformation functions for the **stability**:

To: Add an initial document. To(SPM(x)) will be the SPM(x) diagram with another initial document added. The first assumption implies that $\text{Stability}(\text{SPM}(x)) = \text{Stability}(\text{To}(\text{SPM}(x)))$. The relation for To is '='.

T1: Add a revision to a document. T1(SPM(x)) be the SPM(x) diagram with another revised document added. The second assumption implies that $\text{Stability}(\text{SPM}(x)) < \text{Stability}(\text{T1}(\text{SPM}(x)))$. The relation for T1 is '>'.

The equivalence function, To maps domain D to D'. D' is a software domain which consists of set of documents. T1 establishes a partial ordering on D'.

The ordering relation for To:

Let =_s be the relation for To, $x =_s y$ if $\text{Stability}(x) = \text{Stability}(\text{To}(x))$.

Proof : Let x, y and z be documents in D'.

- then i) \approx is reflexive on D' , it means $\text{Stability}(x) \approx \text{Stability}(To(y))$
 $\Rightarrow \text{Stability}(x) \approx \text{Stability}(To(x))$,
- ii) \approx is symmetric on D' , it means $\text{Stability}(SPM(x)) \approx \text{Stability}(To(SPM(y))) \Rightarrow \text{Stability}(To(SPM(y))) \approx \text{Stability}(SPM(x))$.
- iii) \approx is transitive on D' , it means $\text{Stability}(x) \approx \text{Stability}(To(y))$
 and $\text{Stability}(y) \approx \text{Stability}(To(z)) \Rightarrow \text{Stability}(x) \approx \text{Stability}(To(z))$.

The relation \approx is an **equivalence relation** for To . Since " \approx " satisfies the properties of equivalence relation : **reflexivity, symmetry and transitivity**. Thus, the order pair (D', \approx) for To is a equivalence set.

The ordering relation for T_1 :

Let $>$ be a ordering relation for the T_1 , $x > y$ if $\text{Stability}(x) > \text{Stability}(T_1(y))$.

Proof : Let x, y and z be documents in D' .

- then i) $\text{Stability}(x) > \text{Stability}(T_1(y))$ and $\text{Stability}(y) > \text{Stability}(T_1(z))$ which implies $\text{Stability}(x) > \text{Stability}(T_1(z))$.
 It satisfy the property of **transitive**.

- ii) In part i), if we had both $\text{Stability}(x) > \text{Stability}(T_1(y))$ and $\text{Stability}(x) \approx \text{Stability}(T_1(y))$, then we would have $\text{Stability}(x) > \text{Stability}(T_1(x))$ contradicting **irreflexivity**. And if both $\text{Stability}(x) > \text{Stability}(T_1(y))$ and $\text{Stability}(y) > \text{Stability}(T_1(x))$, then by transitively $\text{Stability}(x) > \text{Stability}(T_1(x))$, again contradicting **irreflexivity**. For T_1 ,

Stability (x) >ₛ Stability (T₁(y)) is irreflexive, it is never the case that **Stability (x) >ₛ Stability(T₁(x))**. Therefore, T₁ meets the conditions of **transitive** and **irreflexive** for partial ordering. Thus the order pair (D', >ₛ) for T₁ is a **partial ordering set**.

From the above proofs, we know the ordering relations for T₀ and T₁ are the equivalence and partial ordering. For the software domain D, we should combine these two transformation functions, because the software domain consists of both a set of initial documents and a set of revised documents.

The mapping function : **Stability: D → ℝ** will be prove by the direct proof to find out whether it preserves the ordering relation on D'. The notation 'LHS' and 'RHS' will be used to represent the equation on the left-hand side equal to the equatin on the right-hand side. The notation (RHS)' is used to define the change after calculation of right-hand side equation.

The procedure for the **direct proof** is expressed as below:

The ordering relation for T₀

Prove **Stability(SPM(x))=Stability(T₀ (T₀ (T₀(SPM(x).....))**

1. Basis:

Let n=1 be the initial state of the SPM(x), then

Stability(SPM(x)) = Stability (T₀(SPM(x)))=>

$$(1 + \sum_{i=1}^n \sum_{j=2}^n \frac{\text{tdf}(1)}{\text{tdf}(i)^*j}) \div (j'+1) = s (1 + \sum_{i=1}^n \sum_{j=2}^n \frac{\text{tdf}(1)}{\text{tdf}(i)^*j} \rightarrow \frac{\text{tdf}(1)}{\text{tdf}(1)^*j}) \div (j'+1)$$

$$\text{Prove } \text{LHS}/(j+1) = s \left(\text{RHS} + \frac{\text{tdf}(1)}{\text{tdf}(i)*j} \right) / (j+1)$$

(The equation on the LHS and RHS will be summed until the second initial document is created.)

$$\text{Since } \frac{\text{tdf}(1)}{\text{tdf}(i)*j} = 0$$

$$\text{Then } \text{LHS} = s \text{ (RHS)'$$

(From Stability(D), we know the summation starts from $i=2$. For all initial documents $i=1$, summation in the equation for the stability should be unchanged)

2. Hypothesis:

Assume we have $m-1$ documents, the number of versions of document f are nf and the number of revised documents are pj . Let Stability (SPM(x))

= Stability (To (To (To ... (SPM(x)....))) is true, then

$$\left(1 + \sum_{f=1}^{m-1} \sum_{j=2}^{nf} \frac{\text{tdm}-1(1)}{\text{tdm}-1(nf)*pj} \right) / (pj+1) = s \left(1 + \sum_{f=1}^{m-1} \sum_{j=2}^{nf} \frac{\text{tdm}-1(1)}{\text{tdm}-1(nf)*pj} \right. \\ \left. + \frac{\text{tdm}(1)}{\text{tdm}(i)*pj} \right) / (pj+1)$$

$$\text{Prove } \text{LHS}/(pj+1) = s \left(\text{RHS} + \frac{\text{tdm}(1)}{\text{tdm}(i)*pj} \right) / (pj+1)$$

$$\text{Since } \frac{\text{tdm}(1)}{\text{tdm}(i)*pj} = 0$$

(From basis)

$$\text{Then } \text{LHS} = s \text{ (RHS)'$$

3. Induction:

Suppose Stability (SPM(x)) = Stability (To (SPM(x))), then

$$(1 + \sum_{i=1}^{m-1} \sum_{j=2}^{nf} \frac{tdm-1(1)}{tdm-1(nf)*p_j} + \frac{tdm(1)}{tdm(i)*p_j}) / (p_j' + 1) = s (1 + \sum_{i=1}^{m-1} \sum_{j=2}^{nf} \frac{tdm-1(1)}{tdm-1(nf)*p_j} + \frac{tdm(1)}{tdm(i)*p_j} + \frac{tdm+1(1)}{tdm+1(i)*p_j}) / (p_j' + 1)$$

Prove LHS / (p_j' + 1) = s (RHS + $\frac{tdm+1(1)}{tdm+1(i)*p_j}$) / (p_j' + 1)

Since $\frac{tdm+1(1)}{tdm+1(i)*p_j+1} = 0$
(From hypothesis)

Then LHS = s (RHS)

Therefore Stability (SPM(D)) = Stability (To (To (To (SPM(x).....))) //

The relation for To is =s according to above direct proof.

The relation for T1

Prove Stability (SPM(x)) < Stability (T1 (T1 (T1 (SPM(x).....)))

1. Basis:

Let n=1 be the initial state of the SPM(x), then

Stability (SPM(x)) < Stability (T1 (SPM(x)))

$$(1 + \sum_{i=1}^n \sum_{j=2}^n \frac{tdf(1)}{tdf(i)*j}) / (0+1) > s (1 + \sum_{i=1}^m \sum_{j=2}^n \frac{tdf(1)}{tdf(i)*(j+1)}) / (1+1)$$

Prove LHS < s (RHS + tdf(1) / tdf(2)*1) + 2

(The equation on the LHS and RHS will be summed until the first revised document is created.)

Since tdf(2) > s tdf(1) then (tdf(1) + tdf(2)) < 1

LHS = RHS = 1, RHS + (tdf(1) / tdf(2)) < 2

$$(RHS + (tdf(1) / tdf(2))) / 2 < 1$$

(from assumption 2)

$$\text{Then } LHS \leq (RHS)' \quad //$$

2. Hypothesis:

Assume for some documents f , nr is increased by 1,

let $\text{Stability}(\text{SPM}(x)) < \text{Stability}(T_1(T_1(T_1 \dots (\text{SPM}(x) \dots)))$ is true, then

$$\begin{aligned} & (1 + \sum_{f=1}^m \sum_{j=2}^{nf-1} \frac{tdf(1)}{tdf(nf-1) * (pj-1)}) / (pj'+1) > s (1 + \sum_{f=1}^m \sum_{j=2}^{nf-1} \frac{tdf(1)}{tdf(nf-1) * (pj-1)} \\ & + \frac{tdf(1)}{tdf(nf) * pj}) / (pj'+1) \end{aligned}$$

Since $LHS = RHS$, 'x' is used to substitute LHS and PHS.

$$\text{Prove } \frac{LHS}{pj} > s (RHS + \frac{tdf(1)}{tdf(nf) * pj}) / (pj'+1)$$

$$\frac{x}{pj} > \frac{x}{pj'+1} + \frac{tdf(1)}{tdf(nf) * pj(pj'+1)}$$

$$\frac{x}{pj} - \frac{x}{pj'+1} - \frac{tdf(1)}{tdf(nf) * pj(pj'+1)} > 0$$

$$\text{Since } \frac{x(pj+1) - x(pj)}{pj(pj'+1)} - \frac{tdf(1)}{tdf(nf) * pj(pj'+1)}$$

$$= \frac{x}{pj(pj'+1)} - \frac{tdf(1)}{tdf(nf) * pj(pj'+1)} = x - \frac{tdf(1)}{tdf(nf)}$$

$$tdf(nf) > tdf(1), \frac{tdf(1)}{tdf(nf)} < 1$$

$$x = 1 + \sum_{f=1}^m \sum_{j=2}^{nf-1} \sum_{j=0}^{pj-1} \frac{tdf(1)}{tdf(nf-1) * (pj-1)} > 1$$

$$x - \frac{tdf(1)}{tdf(nf)} > 0$$

$$\text{Then } \frac{LHS}{p_j} > RHS + \left(\frac{tdf(1)}{tdf(nf)*p_j} \right) \div (p_j+1)$$

$$LHS > (RHS) \quad //$$

3. Induction:

Let nf is increased by 1 for some documents f , then

Stability (SPM(x)) < Stability (T₁(T₁(SPM(x)))

$$(1 + \sum_{f=1}^n \sum_{j=2}^{nf-1} \frac{tdf(1)}{tdf(nf-1)*(p_j-1)} + \frac{tdf(1)}{tdf(nf)*p_j}) \div (p_j'+1) > (1 + \sum_{f=1}^m \sum_{j=2}^{nf-1}$$

$$\frac{tdf(1)}{tdf(nf-1)*(p_j-1)} + \frac{tdf(1)}{tdf(nf)*p_j} + \frac{tdf(1)}{tdf(nf+1)*(p_j+1)}) \div (p_j'+2)$$

$$\text{Prove } \frac{LHS}{p_j'+1} > \frac{RHS}{p_j'+2} + \frac{tdf(1)}{tdf(nf+1)*(p_j+1)*(p_j'+2)}$$

Since LHS = RHS, LHS and RHS are substituted by 'x'.

$$\frac{x}{p_j'+1} > \frac{x}{p_j'+2} + \frac{tdf(1)}{tdf(nf+1)*(p_j+1)*(p_j'+2)}$$

$$\frac{x}{p_j'+1} - \frac{x}{p_j'+2} - \frac{tdf(1)}{tdf(nf+1)*(p_j+1)*(p_j'+2)} > 0$$

$$\text{Since } \frac{x(p_j'+2) - x(p_j'+1)}{(p_j'+1)*(p_j'+2)} - \frac{tdf(1)}{tdf(nf+1)*(p_j+1)*(p_j'+2)}$$

$$= x - \frac{tdf(1)}{tdf(nf+1)}$$

$$tdf(nf+1) > tdf(1), \quad \frac{tdf(1)}{tdf(nf+1)} < 1$$

$$x = (1 + \sum_{j=1}^n \sum_{i=2}^{nf-1} \sum_{p=0}^{p_j-1} \frac{tdf(1)}{tdf(nf-1)*(p_j-1)} + \frac{tdf(1)}{tdf(nf)*p_j}) > 1$$

$$\text{Then } \frac{x}{p_j+1} - \frac{x}{p_j+2} - \frac{tdf(1)}{tdf(nf+1)*(p_j+1)*(p_j+2)} > 0$$

LHS > (RHS)

Therefore Stability (SPM(x)) = Stability (T₁(T₁(T₁.....(SPM(x)).....))) //

From the above direct proof, we can conclude the relation for T₀ is =s and T₁ is <s. The Stability measure can be defined as follows:

Definition 3.17: Stability is a function from D to R, i.e., **Stability: D → R** and it preserves the ordering ≥s on D. Let x ≥s y iff stability(x) =s stability(T₀(y)) and stability(x) >s stability(T₁(y)).

The measurement of stability can be used to express the change in the internal and external environment of the development process. In modern computer society, properly updating the document to reflect modifications is required to maintain the system in the life cycle. The early static "waterfall" model is no longer satisfactory to represent this tendency. The stability measure is suitable to any development model to analyze the development situation.

3.6 Progress Measure

Progress value is one for all initial documents, Adding a revised document implies progress. Especially for the **system sculpture, incremental development** and **RUDE**, these development models are developed as a series of versions, each version contains more modules completed and having more capability than the previous version. Therefore with each revision ,there is more progress.

The assumptions for the **PROGRESS** are:

1. For all initial document **progress** value is one.
2. Adding a revised document implies **progress**.
3. The longer the time of between revision the more **progress**.

The **Progress** function can be denoted by **Progress(x)** which is used to determine the progress in the software development x. The **Progress(x)** measure is expressed as follows:

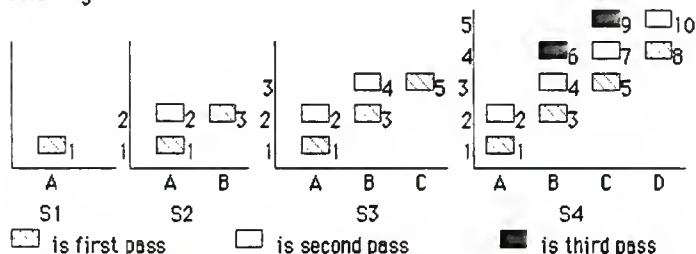
$$\text{Progress}(x) = 1 + \sum_{f=1}^m \sum_{i=2}^n \frac{\text{tdf}(i) - \text{tdf}(i-1)}{\text{tdf}(i)}$$

where 1. **tdf(i)** is the completion time for latest version of document **f**.

2. **tdf(i-1)** is the completion time before the latest document with the same document type **f**.

3. **i** is the version number. Notice the starting value for **i** is 2 in the summation. When **i=1**, the document is an initial document. The progress for an initial document should be unchanged.

We use the same examples as previous one to figure out how to quantify the stability.



1-10 the number to the right of each node indicates the sequence in which it is added.

For (S1), $\text{Progress}(S1) = 1+0=1$;

For (S2), $\text{Progress}(S2) = (1+0.5)= 1.5$;

For (S3), $\text{Progress}(S3) = ((1+0.5)+0.33)=1.83$;

For (S4), $\text{Progress}(S4) = ((1+0.5)+(0.33+0.25)+(0.25+0.2)+0.2)=2.73$.

To verify the **Progress** is a worthwhile measure, the same steps as the previous one will be taken.

--Determine the relations:

The relations among documents can be expressed as the following transformation functions for the **Progress**:

To: Add an initial document. $\text{To}(\text{SPM}(x))$ will be the $\text{SPM}(x)$ diagram with another initial document added. The first assumption implies that $\text{Progress}(\text{SPM}(x)) = \text{Progress}(\text{To}(\text{SPM}(x)))$. The relation for **To** is '='.

T₁: Add a revision to a document. $T_1(SPM(x))$ be the $SPM(x)$ diagram with another revised document added. Assumption 2 implies that $Progress(SPM(x)) < Progress(T_1(SPM(x)))$. The relation for T_1 is ' $<w$ '.

The equivalence function, T_0 maps domain D to D' . D' is a software domain which consists of sets of documents. T_1 establishes a partial ordering on D' .

The ordering relation for T_0 :

Let $=w$ be the relation for T_0 , $x =_p y$ if $Progress(x) =_p Progress(T_0(y))$.

Proof: Let x , y and z be documents in D' .

then i) $=_p$ is reflexive on D' , it means $Progress(x) =_p Progress(T_0(y))$

$\Rightarrow Progress(x) =_p Progress(T_0(x))$,

ii) $=_p$ is symmetric on D' , it means $Progress(SPM(x)) =_p Progress(T_0(SPM(y))) \Rightarrow Progress(T_0(SPM(y))) =_p Progress(SPM(x))$.

iii) $=_p$ is transitive on D' , it means $Progress(x) =_p Progress(T_0(y))$ and $Progress(y) =_p Progress(T_0(z)) \Rightarrow Progress(x) =_p Progress(T_0(z))$.

The relation $=_p$ is an **equivalence relation** for T_0 . Since " $=_p$ " satisfies the properties of equivalence relation: **reflexivity, symmetry and transitivity**. Thus, the order pair $(D', =_p)$ for T_0 is a **equivalence set**.

The ordering relation for T_1 :

Let $<_p$ be a ordering relation for the T_1 , $x <_p y$ if $Progress(x) <_p Progress(T_1(y))$.

Proof : Let x, y and z be documents in D' .

then i) $\text{Progress}(x) \leq_p \text{Progress}(T_1(y))$ and $\text{Progress}(y) \leq_p \text{Progress}(T_1(z))$ which implies $\text{Progress}(x) \leq_p \text{Progress}(T_1(z))$. It satisfy the property of transitive.

ii) In part i), if we had both $\text{Progress}(x) \leq_p \text{Progress}(T_1(y))$ and $\text{Progress}(x) =_p \text{Progress}(T_1(y))$, then we would have $\text{Progress}(x) \leq_p \text{Progress}(T_1(x))$ contradicting irreflexivity. And if both $\text{Progress}(x) \leq_p \text{Progress}(T_1(y))$ and $\text{Progress}(y) \leq_p \text{Progress}(T_1(x))$, then by transitively $\text{Progress}(x) \leq_p \text{Progress}(T_1(x))$, again contradicting irreflexivity. For T_1 , $\text{Progress}(x) \leq_p \text{Progress}(T_1(y))$ is irreflexive, it is never the case that $\text{Progress}(x) \leq_p \text{Progress}(T_1(x))$. Therefore, T_1 meets the conditions of transitive and irreflexive, it is a partial ordering. Thus, the order pair (D', \leq_p) for T_1 is a partial ordering set.

From the above proofs, we know the ordering relations for T_0 and T_1 are the equivalence and partial ordering. For the software domain D , we should combine these two transformation functions together, because the software domain consists of both a set of initial documents and a set of revised documents.

The mapping function : $\text{progress} : D \rightarrow \mathcal{R}$ will be prove by the direct proof to find out whether it preserves the ordering relation on D' . The notation 'LHS' and 'RHS' will be used to represent the equation on the

left-hand side equal to the equation on the right-hand side. The notation (RHS)' is used to define the change after calculation of right-hand side equation.

The procedure for the **direct proof** is expressed as below:

The ordering relation for T_0

Prove $\text{Progress}(\text{SPM}(x)) = \text{Progress}(\text{To}(\text{To}(\text{To}(\dots(\text{SPM}(x))\dots)))$

1. Basis:

Let $n=1$ be the initial state of the $\text{SPM}(x)$, then

$\text{Progress}(\text{SPM}(x)) = \text{Progress}(\text{To}(\text{SPM}(x))) \Rightarrow$

$$1 + \sum_{i=1}^1 \sum_{j=2}^n \frac{td1(i) - td1(i-1)}{td1(i)} = p + \sum_{i=1}^1 \sum_{j=2}^n \frac{td1(i) - td1(i-1)}{td1(i)} + \frac{td2(i) - td2(i-1)}{td2(i)}$$

Prove $\text{LHS} = p \text{ RHS} + \frac{td2(i) - td2(i-1)}{td2(i)}$

(The equation on the LHS and RHS will be summed until the second initial document is created.)

Since $\frac{td2(i) - td2(i-1)}{td2(i)} = 0$

(From $\text{Progress}(D)$, we know the summation starts from $i=2$.)

For all initial documents $i=1$, summation in the equation for the progress should be unchanged)

Then $\text{LHS} = p \text{ (RHS)}' //$

2. Hypothesis:

Assume we have $m-1$ documents and the number of versions of document f

are \mathbf{n} . Let $\text{Progress (SPM}(x)) = \text{Progress (To (To (To ... (SPM}(x) \dots)))}$ is true,

then

$$1 + \sum_{i=1}^{m-1} \sum_{j=2}^{\text{Inf}} \frac{\text{tdm-1}(i) - \text{tdm-1}(i-1)}{\text{tdm-1}(i)} = p + \sum_{i=1}^{m-1} \sum_{j=2}^{\text{Inf}} \frac{\text{tdm-1}(i) - \text{tdm-1}(i-1)}{\text{tdm-1}(i)} \\ + \frac{\text{tdm}(i) - \text{tdm}(i-1)}{\text{tdm}(i)}$$

Prove $\text{LHS} = p \text{ RHS} + \frac{\text{tdm}(i) - \text{tdm}(i-1)}{\text{tdm}(i)}$

Since $\frac{\text{tdm}(i) - \text{tdm}(i-1)}{\text{tdm}(i)} = 0$

(From basis)

Then $\text{LHS} = p \text{ (RHS)}' //$

3. Induction: $\text{Progress (To (SPM}(x)) = \text{Progress (To (To (SPM}(x)))}$

Suppose $\text{Progress (SPM}(x)) = \text{Progress (To (SPM}(x)))$, then

$$1 + \sum_{i=1}^{m-1} \sum_{j=2}^{\text{Inf}} \frac{\text{tdm-1}(i) - \text{tdm-1}(i-1)}{\text{tdm-1}(i)} + \frac{\text{tdm}(i) - \text{tdm}(i-1)}{\text{tdm}(i)} = p + \sum_{i=1}^{m-1} \sum_{j=2}^{\text{Inf}} \frac{\text{tdm-1}(i) - \text{tdm-1}(i-1)}{\text{tdm-1}(i)} \\ + \frac{\text{tdm-1}(i) - \text{tdm-1}(i-1)}{\text{tdm-1}(i)} + \frac{\text{tdm}(i) - \text{tdm}(i-1)}{\text{tdm}(i)} + \frac{\text{tdm+1}(i) - \text{tdm+1}(i-1)}{\text{tdm+1}(i)}$$

Prove $\text{LHS} = p \text{ RHS} + \frac{\text{tdm+1}(i) - \text{tdm+1}(i-1)}{\text{tdm+1}(i)}$

Since $\frac{\text{tdm+1}(i) - \text{tdm+1}(i-1)}{\text{tdm+1}(i)} = 0$

(From Hypothesis)

Then $\text{LHS} = p \text{ (RHS)}' //$

Therefore Progress (SPM (D)) = Progress (To (To (To.....(SPM(x).....))). The relation for To is =p according to above direct proof.

The relation for T1

Prove Progress (SPM(x)) < Progress (T1 (T1 (T1(SPM(x).....)))

1. Basis:

Let n=1 be the initial state of the SPM(x), then

$$\text{Progress (SPM(x))} < \text{Progress(T1(SPM(x)))}$$

$$1 + \sum_{f=1}^m \sum_{i=2}^n \frac{\text{tdf}(i) - \text{tdf}(i-1)}{\text{tdf}(i)} < p \ 1 + \sum_{f=1}^m \sum_{i=2}^n \frac{\text{tdf}(i) - \text{tdf}(i-1)}{\text{tdf}(i)} \rightarrow \frac{\text{tdf}(2) - \text{tdf}(1)}{\text{tdf}(2)}$$

$$\text{Prove} \quad \text{LHS} < \text{RHS} + \frac{\text{tdf}(2) - \text{tdf}(1)}{\text{tdf}(2)}$$

(The equation on the LHS and RHS will be summed until the first revised document is created.)

$$\text{Since} \quad \text{tdf}(2) > \text{tdf}(1), \text{tdf}(2) - \text{tdf}(1) > 0$$

(From assumption 2)

$$\frac{\text{tdf}(2) - \text{tdf}(1)}{\text{tdf}(2)} > 0$$

$$\text{Then} \quad \text{LHS} > (\text{RHS})' //$$

2. Hypothesis:

Assume for some documents f, nr is increased by 1,

let Progress (SPM(x)) < Progress (T1 (T1 (T1(SPM(x).....))) is true, then

$$1 + \sum_{f=1}^m \sum_{i=2}^{nf-1} \frac{\text{tdf}(nf-1) - \text{tdf}(nf-2)}{\text{tdf}(nf-1)} < p \ 1 + \sum_{f=1}^m \sum_{i=2}^{nf-1} \frac{\text{tdf}(nf-1) - \text{tdf}(nf-2)}{\text{tdf}(nf-1)}$$

$$+ \frac{tdf(nf) - tdf(nf-1)}{tdf(nf)}$$

Prove LHS < p RHS + $\frac{tdf(nf) - tdf(nf-1)}{tdf(nf)}$

Since $tdf(nf) > tdf(nf-1)$

$$tdf(nf) - tdf(nf-1) > 0$$

(From basis)

Then LHS < p RHS + $\frac{tdf(nf) - tdf(nf-1)}{tdf(nf)}$

$$\text{LHS} < p \text{ (RHS)} //$$

3. Induction:

Assume Progress (T_1 (SPM(x)) = Progress ($T_1(T_1(T_1(\dots(SPM(x))\dots))$)), let Progress (SPM(x)) < Progress ($T_1(T_1(SPM(x)))$), let nf is increased by 1 for some documents f , then

$$1 + \sum_{f=1}^m \sum_{i=2}^{nf-1} \frac{tdf(nf-1) - tdf(nf-2)}{tdf(nf-1)} + \frac{tdf(nf) - tdf(nf-1)}{tdf(nf)} < p 1 + \sum_{f=1}^m \sum_{i=2}^{nf-1} \frac{tdf(nf-1) - tdf(nf-2)}{tdf(nf-1)} + \frac{tdf(nf) - tdf(nf-1)}{tdf(nf)} + \frac{tdf(nf+1) - tdf(nf)}{tdf(nf+1)}$$

Prove LHS < p RHS + $\frac{tdf(nf+1) - tdf(nf)}{tdf(nf+1)}$

Since $tdf(nf+1) > tdf(nf)$

$$tdf(nf+1) - tdf(nf) > 0$$

(From hypothesis)

$$\text{Then} \quad LHS < p \quad RHS + \frac{tdf(nf+1) - tdf(nf)}{tdf(nf+1)}$$

$$LHS < p \quad (RHS)' //$$

$$\text{Therefore} \quad \text{Progress} (SPM(x)) = \text{Progress} (T_1(T_1(T_1 \dots (SPM(x)) \dots))) //$$

From the above direct proof, we can conclude the relation for T_0 is $=p$ and T_1 is $<p$. The Progress measure can be defined as follows:

Definition 3.19: Progress is a function from D to R , i.e., $\text{progress}: D \rightarrow R$ and it preserves the ordering \leq_p on D . Let $x \leq_p y$ iff $\text{progress}(x) = p$ $\text{progress}(T_0(y))$ and $\text{progress}(x) < p$ $\text{progress}(T_1(y))$.

Progress measurement reflects influence of adding a revision to a document during the progress of the development. It can assist the analyst in assessing the progress of the project in order to better control the schedule.

3.7 Conclusion

Quantitative methods and tools for software engineering management enhance planning, controlling and evaluation of tasks conducted during each

phase of the software life cycle. Over the past decade, work conducted in this area focused primarily on the end product of development--the code representation of software, for example McCabe's cyclomatic number and Halstead's a source code oriented complexity measure. The measurements developed in this chapter, attempted to quantify the activities during the development phase of the software life cycle to help the analyst understand, control and maintenance of the development process.

The measures for **waste, effort stability and progress** were developed from the SPM. The SPM is an abstraction of the activities and products of actual software development and is the foundation on which to build the measures of **waste, effort, stability and progress**.

One of the major purpose of this research is to establish meaningful measures by using methods in mathematics according to the measure development paradigm in the software engineering. In order to achieve this purpose, several steps are taken to develop these measures as follows:

- Make assumptions for each measure based on the evolution of the documents in the software life cycle.
- Determine **relation** or **ordering relations** existing on software domains by using a set of transformation functions.
- Classify these relations as **equivalence or partial orderings**.
- Prove that measurements preserve the ordering.

The results of these calculation for waste, progress, stability and effort in Figure 3.2 can be illustrated in the Figure 3.3

		Added				
No	Time	Document	Waste	Effort	Stability	Progress
1	1	A1	0	1.00	1.00	1.00
2	2	A2	1	1.50	0.75	1.50
3	2	B1	1	1.50	0.75	1.50
4	3	B2	2	1.83	0.61	1.83
5	3	C1	2	1.83	0.61	1.83
6	4	B3	3	2.08	0.50	2.08
7	4	C2	4	2.08	0.44	2.33
8	4	D1	4	2.08	0.44	2.33
9	5	C3	5	2.28	0.39	2.57
10	5	D2	6	2.28	0.35	2.73

Figure 3.3 The results of the waste, effort, stability and progress measures for the example SPM diagram.

Chapter Four

DATA ANALYSIS AND INTERPRETATION

4.1 Introduction

In the area of software development, data processing management often focuses more on coding techniques and system architecture than on how to manage the development. To understand more about the process of software development is one of the important aims of researches in software engineering. Therefore, the progress, effort stability and waste and the projection of these values about the development process is greatly need for better management and control of the development.

The usefulness of software measures in management of software development should be supported by empirical results. The measurement of waste, progress, stability and effort which are developed in chapter three will be applied to the project groups for the CS540-CS541. The results of the measurements will be analyzed by finding the relationship between the completion time and each measure. Statistical methods will be applied to generate projection models as well as to analyze these measures.

4.2 Data Analysis and Interpretation

The usefulness of software measures in management of software development should be supported by empirical results.

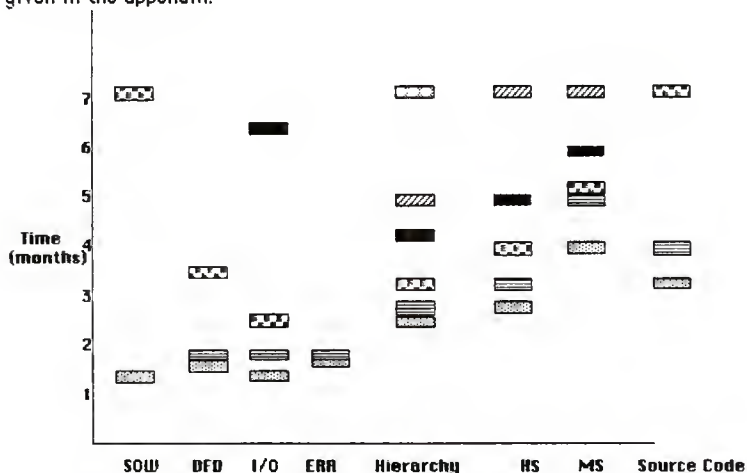
In order to perform analysis and interpretation on the data so as to further understand these measures, the following steps were taken:

1. The measurement of waste, progress, stability and effort which are developed in chapter three were applied to the project groups for the CS540-CS541.
2. The relationship between completion time of each measure were evaluated by the correlation coefficient and the scatter diagrams for each project group.
3. Projection models were generated for each measure by using regression equations. These are aimed at calculating the final values of each measure from the values calculated during the initial stages of the project.
4. The residuals and Mean Magnitude of Relative Error(MRE) for the each measure were used to express the differences between the actual and projected values in order to indicate how stable the measurements are.
5. An evaluation for the waste, progress, stability and effort was made by each project team to check the precision of these measures.
6. The results of these four measures for the thirteen project groups were combined in the analysis of measures overall.

4.2.1 Apply The Measures To The Project Groups

The data used to calculate the measure of waste, progress, stability and effort is based on the information of the thirteen project teams in the computer science course number 540 and 541 (Spring 1988).

To apply the measures into project groups, these project groups are modelled in the **SPM** by tracking the evolution of the set of documents in the software life cycle. The **SPM** diagram for one of the project teams can be illustrated in figure 4.1. The diagram for the rest of the project teams are given in the appendix.



SPM diagram for typical development

Figure 4.1 The SPM diagram for a project team.

In figure 4.1, the horizontal axis represents the documents generated in each phase of software life cycle. The vertical axis is the delivery date for documents. This delivery date is the date on which the documents were turned in for evaluation. The completion time $C(t)$ for the document is calculated by the following equation:

$$C(t) = (c(t) - s(t) + 1) / 30 \text{ where}$$

$c(t)$ is the date to turn in the documents.

$s(t)$ is the date to start the project. The project is starting on September 1, 1987.

The units of the completion time for each document are months. Because there is a one month break between two semesters and the project is interrupted at that period, one month is subtracted from the completion dates in the Spring semester.

The equations for the waste, progress, stability and effort that are developed in the previous chapter are listed in the following.

1. The waste measure $waste(x)$:

$$Waste(x) = \sum_{i=1}^m \sum_{j=1}^n (tdf(i) - tdf(i-1))$$

2. The progress measure $Progress(x)$:

$$Progress(x) = 1 + \sum_{i=1}^m \sum_{j=2}^n \frac{tdf(i) - tdf(i-1)}{tdf(i)}$$

3. The stability measure Stability(x):

$$\text{Stability}(x) = (1 + \sum_{i=1}^m \sum_{j=2}^n \frac{\text{tdf}(1)}{\text{tdf}(i) * j}) / (J+1)$$

4. The effort measure Effort(x):

$$\text{Effort}(x) = \sum_{i=1}^m \sum_{j=1}^n \frac{\text{tdf}(1) - \text{tdf}(i-1)}{\text{tdf}(i)}$$

The calculations for the waste, progress, stability and progress are based on the information from the SPM diagram in figure 4.1. The results of each measure are presented in the table 4.1. The results for the other groups are provided in the appendix.

C(t)	Waste(i)	Progress(i)	Stability(i)	Effort(i)
1.27	0.00	1.00	1.00	1.00
1.63	0.00	1.00	1.00	1.22
1.70	0.00	1.00	1.00	1.26
1.90	0.00	1.00	1.00	1.37
1.93	0.26	1.13	0.94	1.38
2.00	0.45	1.32	0.72	1.42
2.57	1.96	1.91	0.49	1.64
2.80	2.19	1.99	0.40	1.72
3.27	2.66	2.28	0.37	1.87
3.50	4.23	2.28	0.34	1.93
4.10	4.83	2.42	0.31	2.08
4.20	4.83	2.42	0.31	2.10
4.23	6.75	2.88	0.27	2.11
5.00	7.55	3.04	0.26	2.26
5.03	9.15	3.36	0.23	2.27
5.23	9.38	3.40	0.22	2.31
6.03	10.18	3.53	0.21	2.44
6.33	13.94	4.13	0.20	2.99
7.30	28.98	6.19	0.16	2.62

Table 4.1 The total value for the waste, progress, stability and effort for a project team in figure 4.1.

In the table 4.1 indicates the value of the waste, progress and effort increases during the software life cycle. On the other hand, the value of stability decreases over time.

The total values for the waste, progress, stability and effort for all project groups are listed in table 4.2.

Groups	Waste(i)	Progress(i)	Stability(i)	Effort(i)
A	31.54*	7.54*	0.13	2.93
B	28.98	6.19	0.16	2.62
C	24.93	5.44	0.19	2.62
D	30.27	6.00	0.11**	3.57*
E	15.36	5.72	0.15	3.09
F	13.03	4.03	0.24 *	2.56
G	7.00**	2.84**	0.21	2.54**
H	17.14	5.61	0.16	2.96
I	20.36	6.46	0.14	2.64
J	25.88	5.77	0.11**	3.08
K	22.99	7.04	0.14	3.00
L	7.66	3.10	0.24*	2.69
M	21.73	6.12	0.19	2.90

* indicates the greatest value for each measure.

** indicates the least value for each measure.

Table 4.2. The total value for the waste , progress, stability and effort for each group

The table 4.2 shows Group A has both the greatest value of the waste and progress. Group G has the least value for waste and progress, but it has the greatest value of the stability. These results indicate that these measures

maybe highly related to each other. The relationships among the waste, progress and effort for group A and G are provided by the following correlation matrixes in table 4.3 and 4.4.

	W	P	S	E
	+-----+			
W	1.00	.96	-.68	.86
P	.96	1.00	-.82	.95
S	-.68	-.82	1.00	-.93
E	.86	.95	-.93	1.00

Table 4.3 The correlation matrix for the group A.

	W	P	S	E
	+-----+			
W	1.00	.99	-.86	.90
P	.99	1.00	-.91	.94
S	-.86	-.91	1.00	-.97
E	.90	.94	-.97	1.00

Table 4.4 The correlation matrix for the group G.

From the above two tables, we can see the strongest relationship existing between the measurement of waste and progress and the weakest relationship existing between the measurement of waste and stability for both groups.

From the total value for each measure in table 4.2, we can compare group A and group G to see their differences. This can be illustrated from SPM diagram for group A and group G in figure 4.2 and figure 4.3 which are given below.

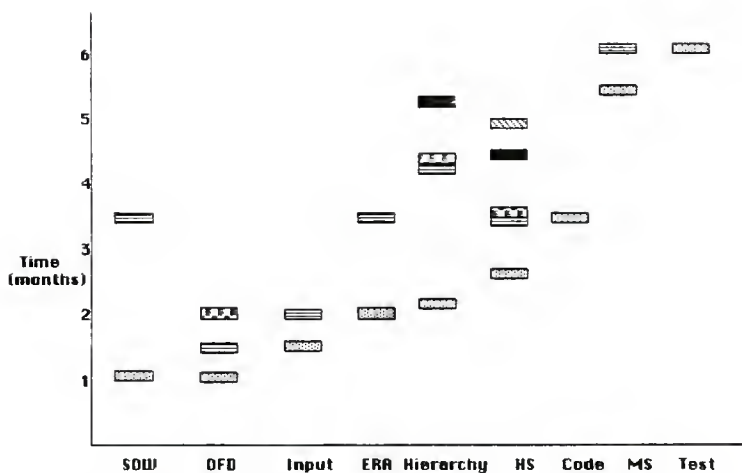


Figure 4.2. The SPM diagram for group 6.

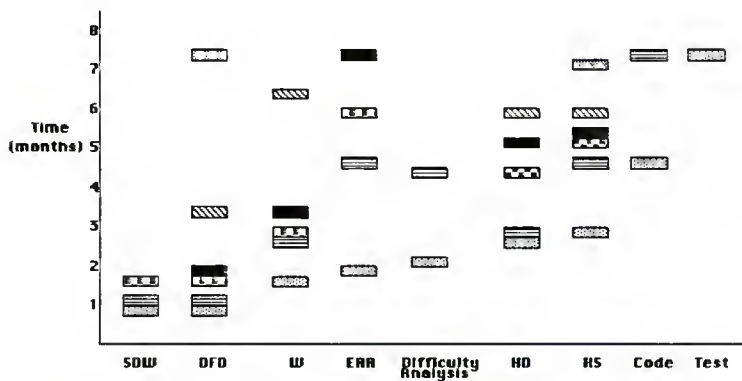


Figure 4.3. The SPM diagram for group A.

From the above two figures, we can see the number of revisions for group G is 22 and the number of revision for group A is 34. Therefore, we can conclude the number of revisions to the document and the time to generate each revision; these are the major elements that influence the value of these measures.

4.2.2. Data Analysis For The Project Groups

To further analyze the empirical results from thirteen project teams, the statistical technique of regression analysis is used. The assumption we have made for regression analysis is that the relationship between completion time and each measure is linear correlated. A relationship between two sets of variables can be evaluated by the correlation coefficient. The correlation coefficient r values for the measures in the thirteen project teams are presented in Table 4.5.

Group	Waste	Progress	Stability	Effort
A	0.9553	0.9762	0.7899	0.9519
B	0.8995	0.9624	0.8724	0.9771
C	0.8907	0.9746	0.5978	0.9376
D	0.8857	0.928	0.8799	0.9495
E	0.9675	0.9591	0.7789	0.9555
F	0.9925	0.9894	0.8893	0.9739
G	0.9679	0.9756	0.9106	0.9686
H	0.9817	0.9667	0.7233	0.9509
I	0.8930	0.9806	0.9060	0.9764
J	0.8130	0.9789	0.8551	0.9599
K	0.9795	0.9842	0.7911	0.9712
L	0.9361	0.9452	0.9358	0.9787
M	0.9241	0.9743	0.8211	0.9676

Table 4.5 The correlation coefficients for the measures in each group.

The greatest and least values of r are (0.9925, 0.8130), (0.9894, 0.9280), (0.9358, 0.5978) and (0.9787, 0.9376) for waste, progress, stability and effort respectively. The mean correlation coefficient r between the completion time and waste, progress, stability and effort measures are 0.93, 0.97, 0.86 and 0.97 respectively. The correlation between the completion time and progress is stronger than the other three measures.

To visualize the relationship between two sets of variables, the scatter diagram is used. The scatter diagrams for the measures of the waste progress, stability and effort with the greatest and the least values of correlation coefficients are shown from figure 4.4 to figure 4.11. Other scatter diagrams will be given in the appendix. These values of r for each measure are also suggested by these scatter diagrams.

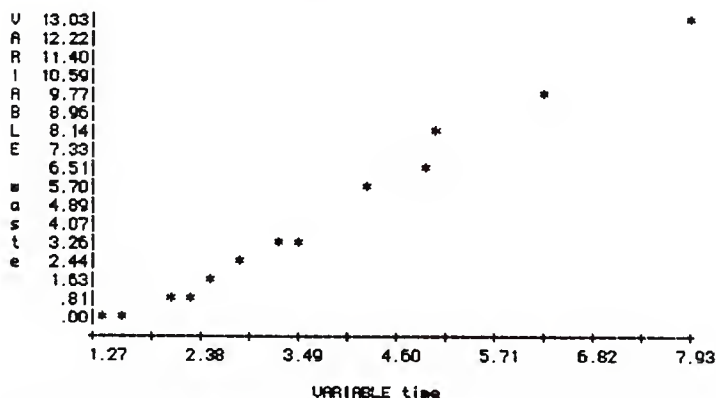


Figure 4.4 The scatter diagram for the value of waste and completion time with the greatest value of r for group F.

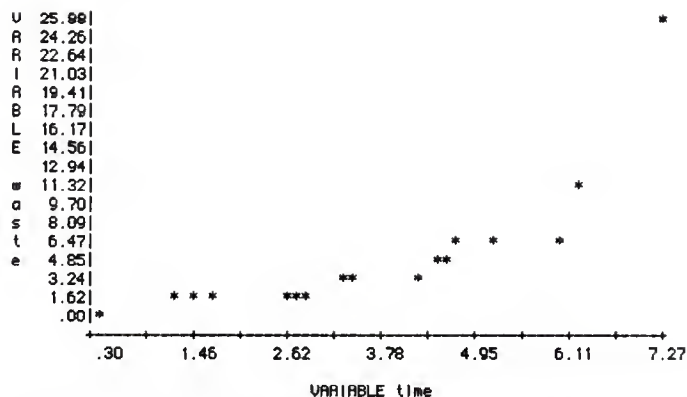


Figure 4.5 The scatter diagram for the value of waste and completion time with the least value of r for group J.

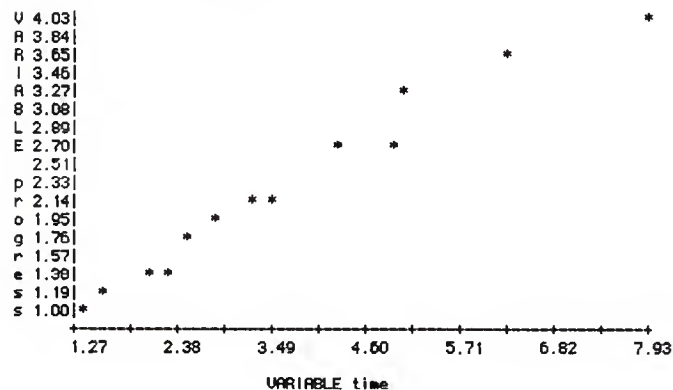


Figure 4.6 The scatter diagram for the value of progress and completion time with the greatest value of r for group F.

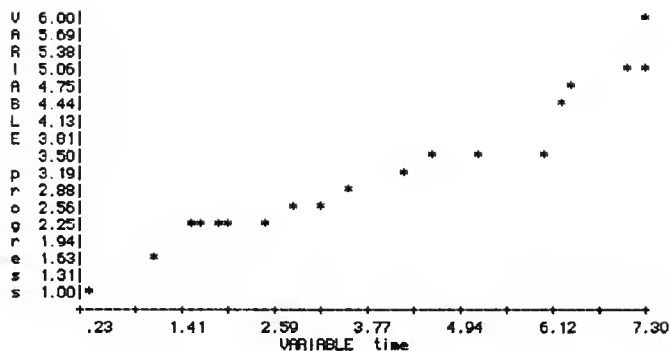


Figure 4.7 The scatter diagram for the value of effort and completion time with the least value of r for group D.

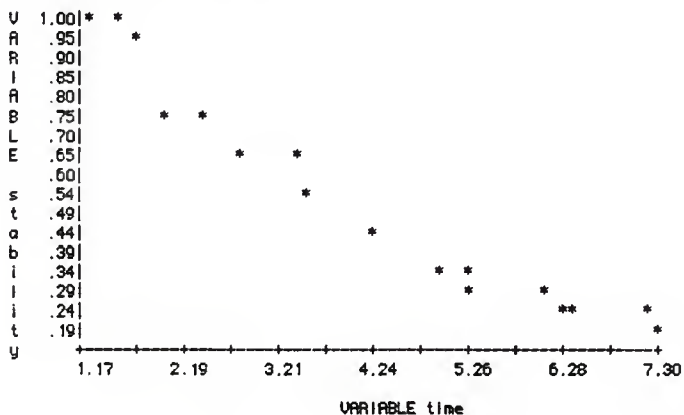


Figure 4.8 The scatter diagram for the value of stability and completion time with the greatest value of r for group L.

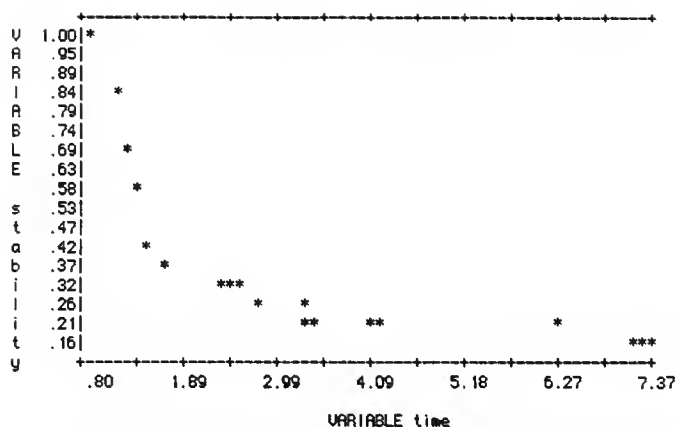


Figure 4.9 The scatter diagram for the value of stability and completion time with the least value of r for group C.

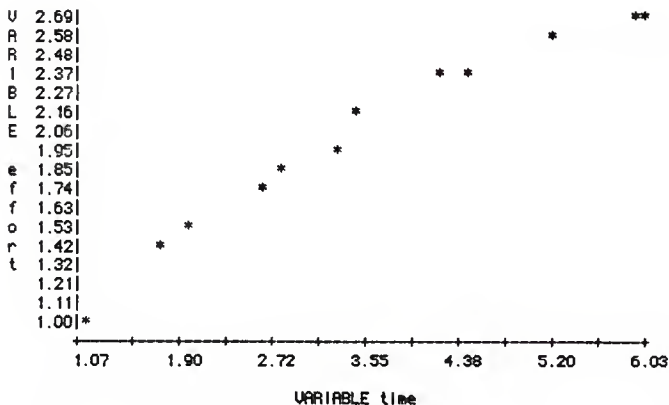


Figure 4.10 The scatter diagram for the value of effort and the completion time with the greatest value of r for group L.

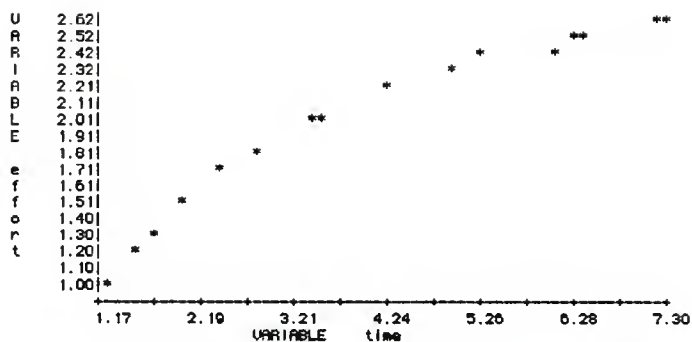


Figure 4.11 The scatter diagram for the value of effort and completion time with the least value of r for group C.

From the above figures, we know that the completion time has a positive relation to the waste, progress and effort and a negative relation to the stability. Generally speaking, the correlation between the completion time and each measure are high. Strong linear relationships implies there exist simple transformation functions among these measures. The transformations can be used to project values for these measures. This will be discussed in section 4.2.3.

4.2.3. Projection Models For The Project Groups

A transformation function exists between two highly related variable. The completion time and each measure are strongly related each other; thus, we can employ the regression to get the coefficient, then use these

coefficients to form a linear equation to project the value of waste, progress, stability and effort. The linear equations for these four measures in each group are listed in the appendix.

To evaluate these functions, we can use the Mean Magnitude Relative Error(MRE). MRE is the absolute value of the difference between the actual values and projected (or predicted) values relative to the actual values. The smaller the value of MRE, the better the projection.

The values of MRE for these measures in thirteen project groups are given in the table 4.6.

Groups	Waste	Progress	Stability	Effort
A	0.481	0.092	0.367	0.081
B	0.562	0.095	0.387	0.050
C	1.751	0.161	0.170	0.080
D	0.634	0.071	0.280	0.078
E	0.170	0.149	0.420	0.093
F	0.228	0.052	0.213	0.052
G	0.220	0.058	0.205	0.050
H	0.255	0.140	0.479	0.095
I	1.590	0.104	0.353	0.065
J	0.690	0.081	0.340	0.068
K	0.482	0.090	0.469	0.060
L	0.352	0.067	0.698	0.049
M	0.610	0.086	0.256	0.075

Table 4.6. The MRE for the measures in each group.

In table 4.6 indicates that the best projection model is effort , the mean MRE for effort is 0.069. On the other hand, the worst project model is

waste , the mean MRE for waste is 0.534.

The performance of the projection models can also be checked by residuals. The residuals are the differences between the actual values and projected values. The value of residuals for measures in each group are shown in the table 4.7.

Group	Waste	Progress	Stability	Effort
A	107.414	2.966	0.320	0.492
B	169.152	2.377	0.461	0.193
C	156.944	1.671	0.891	0.960
D	320.718	2.288	0.208	0.908
E	36.276	4.912	0.659	0.864
F	3.384	0.265	0.170	0.143
G	5.037	0.229	0.130	0.173
H	20.493	2.588	0.677	0.710
I	169.649	2.459	0.316	0.269
J	213.952	1.313	0.261	0.637
K	21.433	1.353	0.402	0.234
L	7.668	0.254	0.078	0.138
M	81.968	1.883	0.260	0.405

Table 4.7. The residuals for the measures in each group.

The table shows that the residuals for the waste are much higher than the other measures. The mean residuals are 101.084, 1.889, 0.372 and 0.471 for waste, progress, stability and effort respectively. In the other words, the projection models for the progress, stability and effort perform better than the waste measure .

4.2.4 Data Analysis And Projection Models For Whole Project

To understand these measures overall, the results of these four measures for the thirteen project groups are combined. Totally, one hundred and thirty data points were obtained. The relationship between the completion time and each measure are analyzed by the scatter diagram. The scatter diagrams for the waste, progress, stability and effort are shown from figure 4.12 to figure 4.15. The horizontal axis shows the development over time. The vertical axis represents the value of each measure.

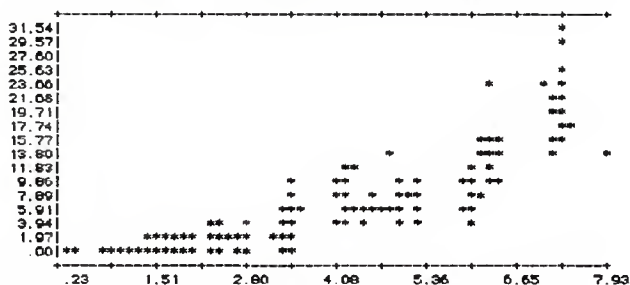


Figure 4.12. The scatter diagram for the waste measure.

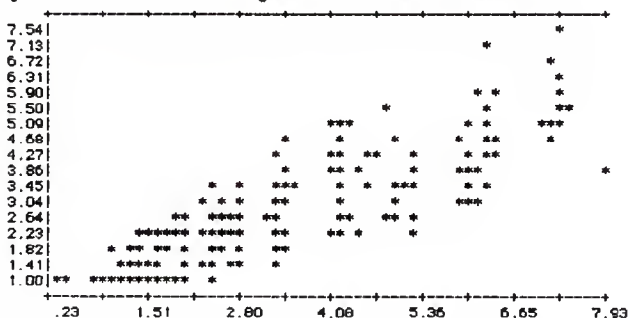


Figure 4.13 The scatter diagram for the progress measure.

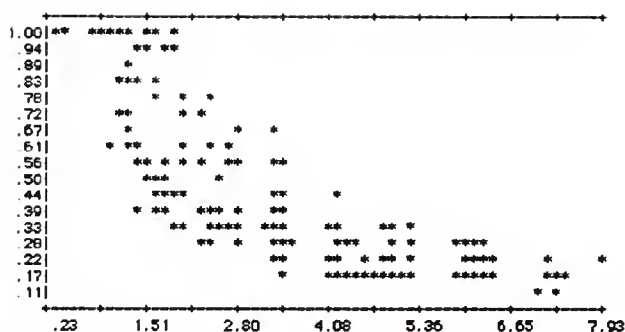


Figure 4.14 The scatter diagram for the stability measure.

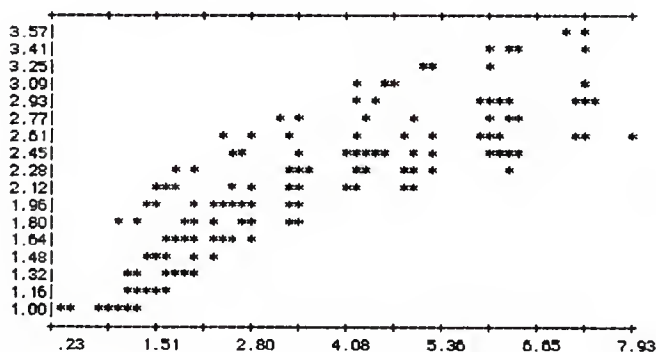


Figure 4.15 The scatter diagram for the effort measure.

The relationships between the completion time and the waste, progress, stability, effort are provided by the following correlation matrix in table 4.8.

	T	W	P	S	E
T	1.00	.86	.85	-.78	.85
W	.86	1.00	.90	-.66	.74
P	.85	.90	1.00	-.82	.83
S	-.78	-.66	-.82	1.00	-.88
E	.85	.74	.83	-.88	1.00

Table 4.8 The correlation matrix for the measures

The correlation between the completion time and each measure are 0.86, 0.85, -0.78 and 0.85 for waste, progress, stability and effort respectively. These results show the completion time and measures are highly related. The projection models for each measure are formed by the regression equations. These equations are listed as follows:

$$\text{Waste}(x) = -4.3651836 + 2.7828031 * C(t).$$

$$\text{Progress}(x) = 0.6034715 + 0.6492754 * C(t).$$

$$\text{Stability}(x) = 0.7894828 - 0.1038421 * C(t).$$

$$\text{Effort}(x) = 1.1440269 + 0.2721306 * C(t).$$

MRE is used to evaluate these functions. The values of MRE for all cases are given in the appendix. The average values of MRE for the waste, progress, stability and effort are 2.546, 0.244, -0.4 and -0.14 respectively. This indicates that the projection models for the waste performs worse than the other three measures. These results correspond to the result from each group. We can use these projection functions (or regression equations) to calculate the final measurements for the waste, progress, stability and effort from values early in the project.

4.2.5. Evaluations From the Project Groups

For checking the precision, an evaluation of the waste, progress, stability and effort was made by each project team. Eleven project groups turned in their evaluation sheets. Besides the values for each measure, some of these groups also gave comments about how they viewed these measures. According to the comments from the team members, we can compare the different assumptions for the measurement of waste, progress, stability and effort between project groups and these measures.

A. For the waste(x):

Groups--The waste is how much wasted effort occurred.

--The waste occurs because of difficulty in motivating the team members.

--The waste is produced when the program trips down the wrong path.

Measure--The waste is generated whenever the revision is created.

B. For the Progress(x):

Groups--The progress depends on how quickly they learned the C language.

--The progress is made when the modules are coded.

--The progress is how much progress was made by the team.

Measure--The progress is made whenever the document is produced.

C. For the Effort(x):

Groups--The effort is how much effort is put into the project.

--More effort was put forth to learn the C language.

Measure--Adding a document takes effort.

D. For the Stability(x):

Groups--The stability is how stable are the requirement, design, development, and maintenance phases in the development process.

--The stability is decreasing when the change is increasing.

--The range for stability from the project groups is from 1 to 7.

Measures--The stability is decreasing whenever a revised document is added.

--The range for stability from the measure is from 0.24 to 0.11.

Although the project groups did not consider the waste, progress, stability and effort for the project in terms of the evolution of documents in each phase of software life cycle, there was a correlation between the evaluations from the project teams and these measures. The value of the evaluations from the project teams and these measures are showed in the tables 4.9.

Groups	W(M)	W(G)	P(M)	P(G)	S(M)	S(G)	E(M)	E(G)
A	31.54	5.50	7.54	7.00	0.13	4.0	2.93	6.0
B	28.98		6.19	7.00	0.16	6.0	2.62	4.0
C	24.93	7.00	5.44	5.00	0.19	5.0	2.62	5.0
D	30.27	1.50	6.00	8.00	0.11	6.0	3.57	7.5
E	15.36	9.00	5.72	5.00	0.15	1.0	3.09	3.0
F	13.03	6.50	4.03	2.50	0.24	3.5	2.56	4.5
H	17.14	5.00	5.61	6.00	0.16	6.0	2.96	6.0
I	20.36	4.50	6.46	5.50	0.14	6.5	2.64	6.0
J	25.88	5.00	5.77	4.00	0.11	6.0	3.48	6.0
K	22.99	4.47	7.04	8.20	0.14	7.4	3.00	7.1
L	7.66	3.67	3.10	5.67	0.24	7.0	2.69	5.0

Table 4.9 The values of evaluations from the project teams and total values from the measures.

**The value of the waste from the evaluations for group B is not available.

M (Measure), G (Group), W(waste), P(Progress), S(Stability), E(Effort).

The following scatter diagram can be used to examine these correlation.

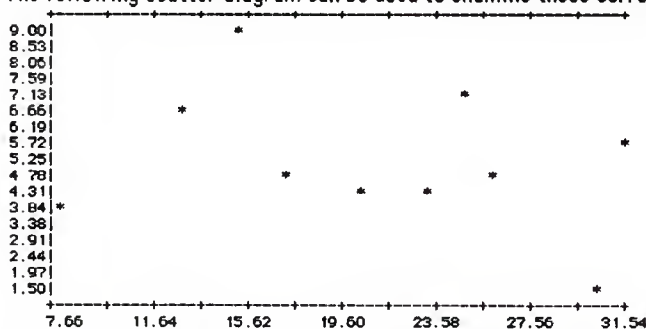


Figure 4.16 The scatter diagram for the value of waste from the project teams and the measures.

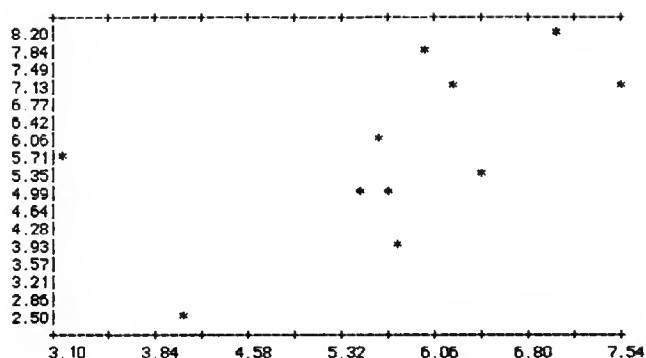


Figure 4.17 The scatter diagram for the value of progress from the project teams and the measures.

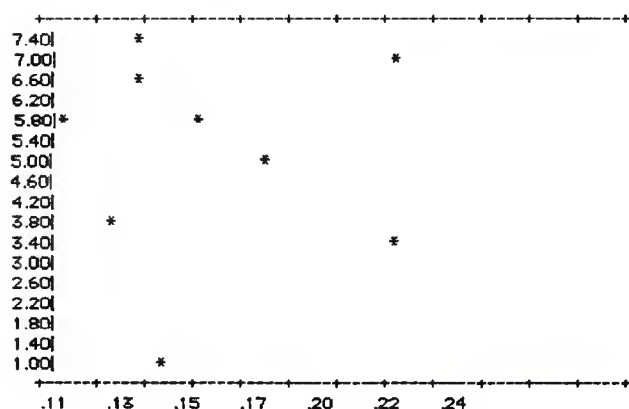


Figure 4.18 The scatter diagram for the value of stability from the project teams and the measures.

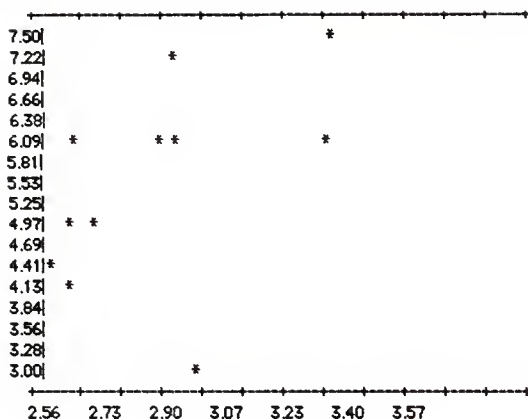


Figure 4.19 The scatter diagram for the value of effort from the project teams and the measures.

With this figures, we can see a correlation existing in Figure 4.16 4.17 and 4.19, even through it is not a perfect one. The diagram for the stability appears to be completely random. This means the correlation between the value from the project teams and stability measure is insignificant. The coefficient correlations are -0.26, 0.57, -0.08 and 0.49 for the waste, progress, stability and effort respectively.

The values from the measures in table 4.9 are total values for these measures. The total value can not reflect the variation that exists among the values for these measure. To justify the problem from the deviation for

each measure, the correlation between the slopes for the measures and the evaluations from the project teams were examined by the scatter diagrams from figure 20 to figure 23.

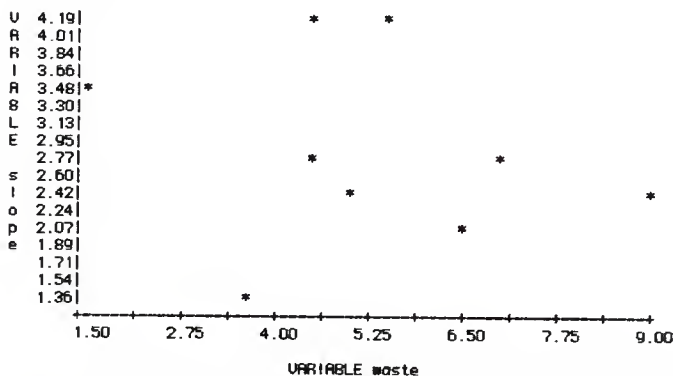


Figure 4.20 The scatter diagram for the slope of the waste measure and the evaluations from the project teams

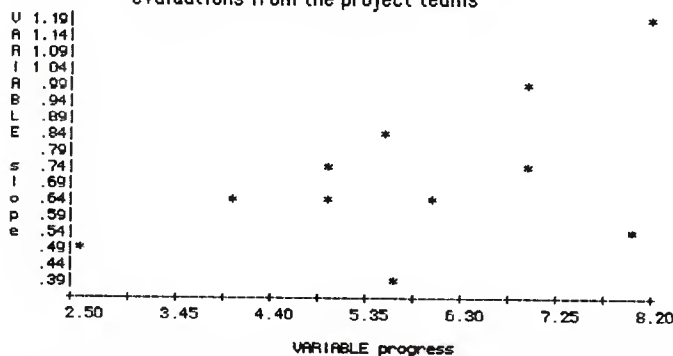


Figure 4.21 The scatter diagram for the slope of the progress measure and the evaluations from the project teams

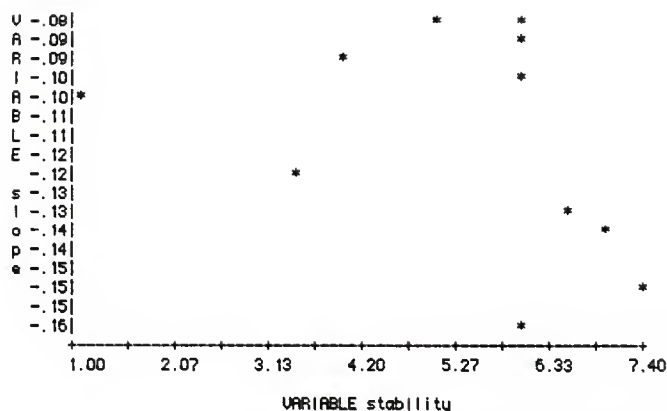


Figure 4.22 The scatter diagram for the slope of the stability measure and the evaluations from the project teams

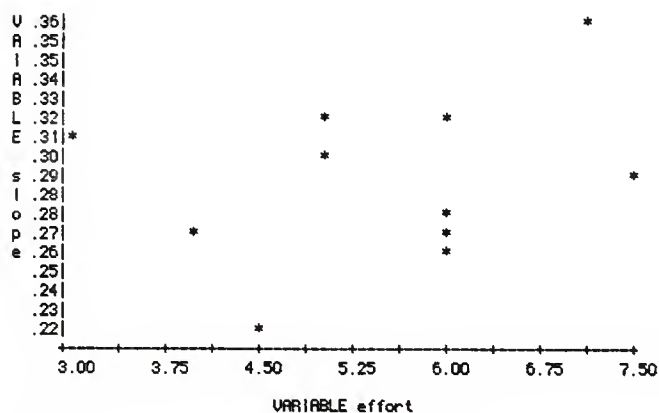


Figure 4.23 The scatter diagram for the slope of the effort measure and the evaluations from the project teams

The correlation coefficients are -0.188, 0.509, -0.376 and 0.237 for waste, progress, stability and effort measures respectively. The value of correlation is increasing from -0.085 to -0.376 for the stability measure.

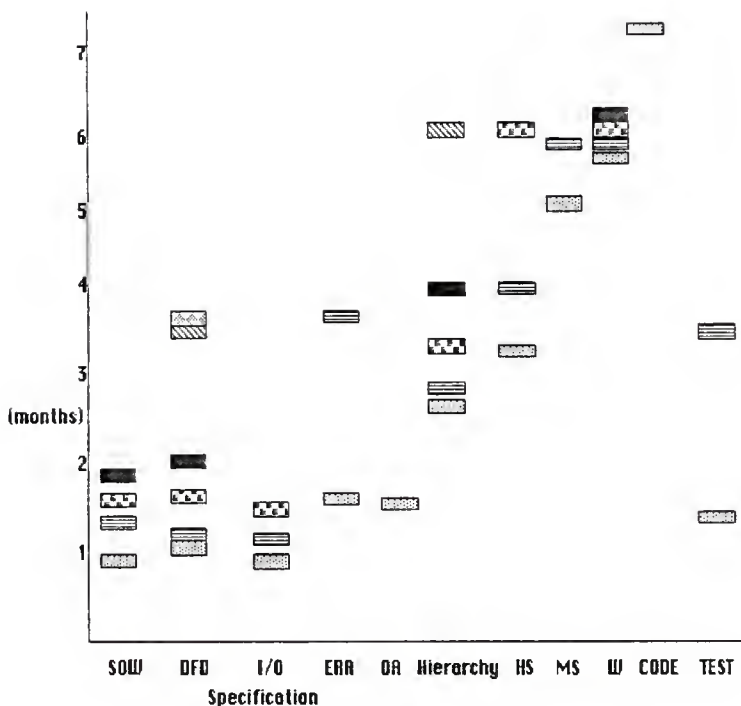
Both correlation from the above two tests shown are relatively low. The reasons for low correlation between the value from the measures and project groups might be caused by the following reasons:

1. The assumptions for these measures from the project teams and measures are quite different. From the comments provided by the project teams, we can see the assumptions made by the team members for the measure are more concentrated on the development phase. On the other hand, the assumptions for the measures in this paper are equal weight throughout the life cycle.
2. The data points used to examine the correlation are few, this might cause the low correlation between two variables. Of course, this will not be the case if the other data are collected with equal or larger variation between the values from the project teams and from the measures.
3. The measurements for the waste, progress, stability and effort are not good enough to completely reflect the activities in the development phases. This might be improved by changing the assumptions for the measures.

4.2.6 SPM Diagram And Initial Diagram

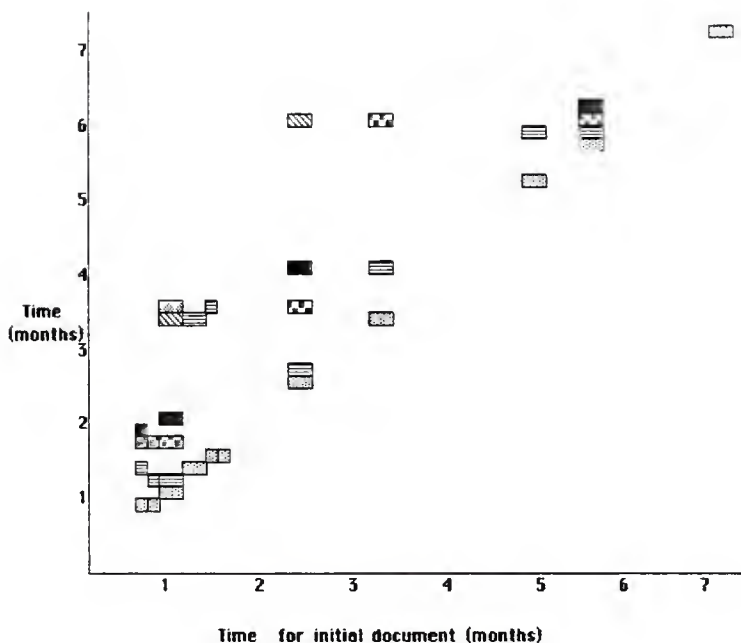
The SPM diagram used in this thesis orders the document in the development process from the analysis phase to maintenance phase. For some project teams, the documents are not produced according to the sequence of the software life cycle. This may increase the difficulty in calculation the measures. For instance, the measurement of effort is partly dependent upon the previous document with different document type; if the document in the previous phase is absent, then there is a question of which document should be used. This problem can be eliminated by replacing the SPM diagram with the initial diagram. The initial diagram orders the documents in the development process, based on the time to produce the initial documents instead of the phase in the software life cycle. This can be illustrated by comparing the SPM and initial diagram for a project team in figure 4.24.

In the actual software development process, the documents might not be generated according to the sequence of the software life cycle especially for the development models other than the waterfall model. The initial diagram can solve this problem. For example, the change rates of the documents which are the ratio of changes for the current document to the previous document with same document type and to the previous document with different document type. The change rate is difficult to calculate if the document in the previous phase has not been produced. The problem can be overcome by substitution of the initial diagram for the SPM diagram.



SPM diagram for typical development

To visualize the software development process, the SPM diagram is recommended. For the purpose of the measurement, the initial diagram might be more appropriate.



Initial Time Diagram

Figure 4.24 Comparing the SPM diagram and initial diagram for a project team.

4.4. Summary

We can not control what we can not measure. A measure is an indication of the phenomenon in the development process. If there is no measurable

indication of the phenomenon in the life cycle, then there is no way to aid in controlling that phenomenon. To successfully control the project, estimation plays an important role.

In this chapter, the projects developed in the computer science course 540 and 541 were used to examine the waste, progress, stability and effort measures. Also, the information from the calculations is helpful to explain the state of the project activities in the development process.

Moreover, the result of the measurement for the project teams is a basis to develop four different models in order to project the waste, progress, stability and effort for the project.

The SPM is useful to help us analyze and compare the differences between projects. Statistical tools were used to analyze data and generate equations for projection. "Statistics can not predict but they can imply." The statistics can help us to explain phenomenons in the software life cycle.[DeM, 1982]

The requirements for a good measure should satisfy several conditions: objectivity, precision, stability, applicability, sensitivity and ease of use. These criteria will be examined and the limitations of the measures will be discussed in the next chapter.

Chapter Five

CONCLUSIONS

5.1 Introduction

Quantitative measures and models are important to understand each phase of the software life cycle. Several metrics have been proposed. They are categorized as the design metrics, the code metrics and the quality metrics. McCabe's complexity metric [McC, 1976], Halstead's software metrics [HAL, 1977] and McCall software quality metric [McC, 1978] are examples of the above three metrics respectively. All of these metrics are focusing on the source code in the development phase of a software life cycle.

Besides measure, quantitative models for software engineering management enhance control and evaluation of tasks conducted in the software life cycle. Quantitative models in software engineering consist of historical experiential models, statistically based models and theoretically based model.

Historical experiential models are the models which rely on experts' experience from past projects, such as, the TRW Wolverton model [WOL, 1974]. Statistically based models use regression as a tool to form a linear or nonlinear equation to produce estimated values for a specific variable; for example, SDC model [NEL, 1966]. Theoretically based models are based on theories about the functions of the human mind during the software

development process, such as, Putnam's Resource Allocation Model [PUT, 1978]. A Composite model is combining the characteristics of the above three models together; for instance, COCOMO model [BOE, 1981].

Measures of waste, progress, stability and effort are developed for quantifying the tasks throughout a life cycle instead of concentrating on one phase in the life cycle. Each task is represented by a set of documents generated in each phase in the development process. The SPM is the model used to record the documents in the software life cycle. The formal framework in software measures and metrics provided by SPM is a foundation to develop the measures in this paper. These measures can help us to understand, analyze and control the development process in terms of waste, progress, stability and effort. Also, the visible characteristic of SPM can assist us to assess the different projects.

The models developed for the progress, stability, effort and schedule should be classified as a theoretically based model. The criteria to evaluate measures and models are common in some aspects, so we can combine these criteria together. These requirements will be judged in the following section.

5.2 The Evaluation of the Measures And Models.

Requirements for the evaluation of measures and models are listed as follows:

1. **Consistency:** A useful measure should preserve the ordering in software domain which is determined by the experts. This is the common paradigm to verify that measure is a worthwhile measure. In other words, if a measure provides a uniform standard to quantify the characteristics of the development process then the measure is a useful measure. The measures developed in this paper preserve the ordering which is defined by the assumptions. The transformation functions are created to express the relation on software domain. Partial and equivalence ordering is the relation existing in these transformation functions. The ordering relation for the measurement of waste, progress, stability and effort proves these measures can consistently quantify the activities in the software life cycle.
2. **Stability:** Since the measures are preserving an ordering in software domain, therefore, the values of the waste, progress and effort increase over time, and the value of stability decreases throughout the life cycle. The linear correlation exists between completion time and these measures. Projections for these measures can be obtained from transformation functions which are generated by regression. The R values and the MRE measure the goodness of these projections.
3. **Objectivity:** The measurements generated in chapter three are based on the SPM. The SPM models the development process by tracking the evolution of the full set of documents. The set of document deliverables is an indicator of the project state. It is also an abstraction of the development process.

The time to generate each document is an objective factor to reflect the movements in each software phase. The problem is how to precisely record the time to produce a document. Especially for the semester project, it becomes extremely difficult to track correctly because the students are taking several courses at the same time. The effort put into the project varies for each student. To reduce the influences of project personnel and the differences among the projects, we apply the same standard to all thirteen project teams. The standard is to calculate the time to generate each document that is based on the time to turn in the document for evaluation. The method for calculating the completion time $C(t)$ was given in the previous chapter. However, to develop a useful model, the completion time should be tracked more carefully.

4. **Sensitivity:** If a small change in one parameter will lead to a relatively greater change in the measure then the measure might not reasonably explain the phenomenons in the development process. The variation for these measures can be checked from the values of mean and standard deviation for the project teams in table 5.1.

VARIABLE	MEAN	STD. DEV.
Waste	5.677	6.353
Progress	2.947	1.498
Stability	0.415	0.263
Effort	2.123	0.631

Table 5.1 Mean and standard deviation for the measures

Table 5.1 shows the variance for the waste measure is much larger than the other measures. It also indicates a small changing in completion time might cause considerably larger changing in the value of the waste.

The variance between the variables can be used to further check the slope of these measures. The slopes for the measures are given in table 5.2.

Group	Waste	Progress	Stability	Effort
A	4.1889	0.9710	-0.0911	0.2727
B	3.5296	0.7202	-0.1597	0.2657
C	2.7929	0.6394	-0.0840	0.3006
D	3.4294	0.5451	-0.0848	0.2895
E	2.4019	0.7541	-0.1051	0.3137
F	2.0575	0.4842	-0.1107	0.2244
G	1.5147	0.3727	-0.1394	0.2840
H	2.4455	0.6363	-0.0903	0.2713
I	2.8308	0.8581	-0.1318	0.2568
J	2.4047	0.6464	-0.0993	0.3218
K	4.1498	1.1920	-0.1510	0.3629
L	1.3594	0.3863	-0.1372	0.3201
M	2.5175	0.6829	-0.0843	0.2802

Table 5.2 The slopes of the measures for each project team

The mean slopes for the stability and effort measures are 2.7402, 0.6837, -0.113 and 0.2895. The slopes for the the stability and effort are more stable than the waste and progress measures. This indicates that the average change in completion time caused by the change in stability and effort are almost the same for the thirteen project groups. The range for the value of these two measures are small. Again, it proves the variance for the values obtain from the stability and effort measures is small.

5. **Precision:** Precision for the measurements can use an objective way to find out whether they really reflect the state of the project. To check the precision, an evaluation of the waste, progress, stability and effort was made by each project groups. The relation between these two values can be checked by coefficient correlations r , r for the values from the project teams and these measures are -0.26, 0.57, -0.08 and 0.49 for the waste, progress, stability and effort measures respectively. And, the coefficient correlations are -0.19, 0.51, -0.38 and 0.24 between the evaluation from the project teams and slope. This shows that the highest correlation existing between the values from the progress measure and the values from the project groups.

For the projection models for whole project groups , the mean MRE for the waste, progress, stability and effort are -2.324, -0.08, -0.15 and 0.28 respectively. These values indicate that the projection values for progress are better than the other models. Whether or not the projection models are acceptable depends on the requirement for the mean MRE. For example, most researchers consider a mean MRE ≤ 0.20 as acceptable for effort prediction models [BAK, 1987]. If we apply the same standard as the effort prediction, then the projection models for the progress and stability in this paper are acceptable . However, there is no well-known standard to evaluate projection models. Standards might vary among different organizations and projects.

The term "waste" has many connotations and the teams' interpretation did not match the assumptions used in developing the measure. A better term that reflects the assumptions might be "revisions" or "revision effort". In general, it is probably better to avoid terms that have strong connotations.

6. **Applicability:** The domain for these measures in this paper is a set of documents in each software life cycle phase. A document is an independent object which is seldom influenced by the project personnel. The projection models developed from these measures can be used in a different project environment as long as an agreement in assumptions for each measure is reached. Moreover, these measures can be applied to different development models, such as, RUDE, rapid prototype, system sculpture, etc..

7. **Ease of use:** The major factor to determine the value of waste, progress, stability and effort is a set of documents produced in the development process. We can obtain this information very easily. Therefore, these measures are easy to apply. The projection models are also easy to apply because they are composed of one dependent variable and one independent variable. Thus it requires little effort to gather data for the models. The drawback for limited data is simplifying assumptions for the measures and models, which cannot always reflect the real project environment. This is also the limitation of the SPM. However, for the purpose of abstraction in the development process, the SPM has already provided a good framework to model the activities in the project environment by tracking the evolution of the documents.

5.3 Conclusions

Since the late sixties and early seventies, the cost to develop software has exceeded the cost to develop hardware. This trend appears more and more obvious. To successfully carry out a software development project, controlling cost has become extremely important. Software engineering was born to achieve this goal.

Over the past decade, software engineering has still depended more on the experience of the software engineer than scientific quantity techniques. This is different from the other areas in engineering. To develop a measurement to quantify the activities and states in the development process is necessary to put software engineering in the same position as the scientifically based areas.

In this paper, several measures and projection models have been generated according to the mathematics methods and the statistics techniques. The framework is based on the SPM. The SPM diagram is useful for evaluating and comparing the different projects by making the evolution of the documents in the software life cycle visible.

To develop a worthwhile measure, a new paradigm is following in this paper. The ordering relations should be preserved in the software domain according to the decision by experts, this is required by measure development paradigm. The measurements for the waste, progress, stability

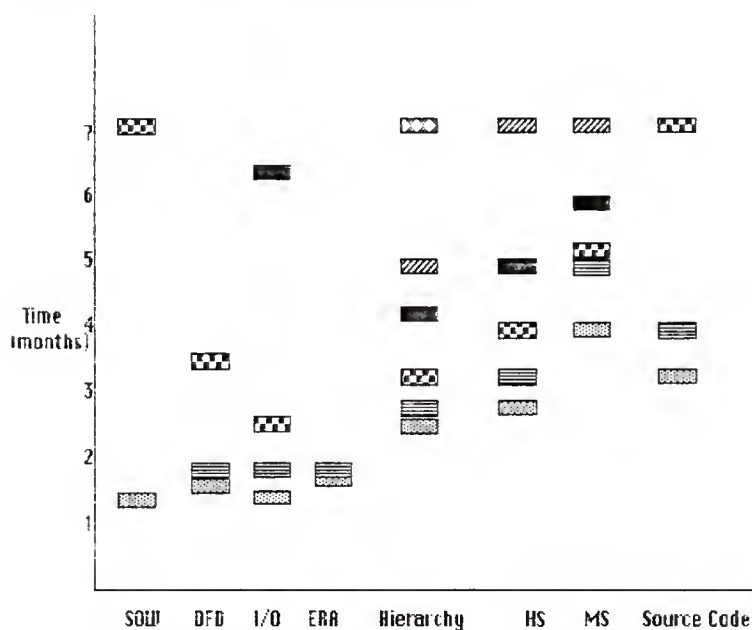
and effort to preserve a relation by generating a number of transformation functions. The partial and equivalence ordering are existing in these relations. The mapping function from each measure to real number R will be prove the ordering relation not only existing in the small subset of the domain but also existing in the whole software domain by direct proof.

The measurements cannot only explain the phenomena but can also foretell the events in the process of the development. The measurements of waste, progress, stability, effort and schedule are an indication of the movements in the whole software life cycle. We can use both the measures and projection models to manage the software development process more effectively and more efficiently.

These measures and models satisfy some criteria for a useful measurement, such as consistency, stability, sensitivity, precision, objectivity, applicability and ease of use. The shortcoming of these measurements is a limited number of arguments which can not completely reflect the software development process. This shortage may be overcome by adding more useful arguments in the SPM.

Quantitative measures and models are in an infant stage. In this paper, I present a way to develop measures and projection models by using scientific methods. There still needs to be more effort put forth into this area.

APPENDIX A: The SPM diagrams for each group.



SPM diagram for group B

DFD : Data Flow Diagrams

ERA : The Entity-Relationship-Attribute Model

HS : Hierarchy Specification



: First pass



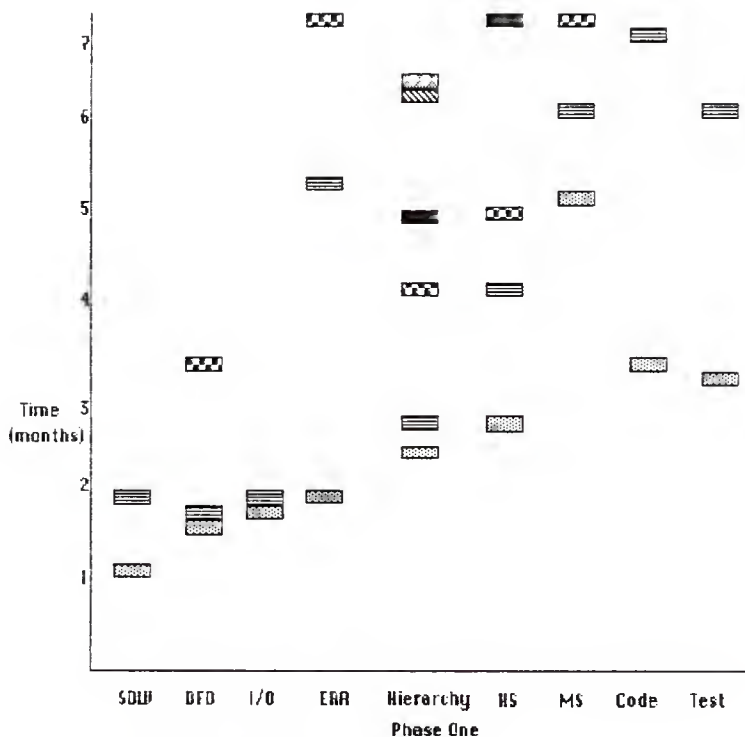
: Fourth pass



: Second pass

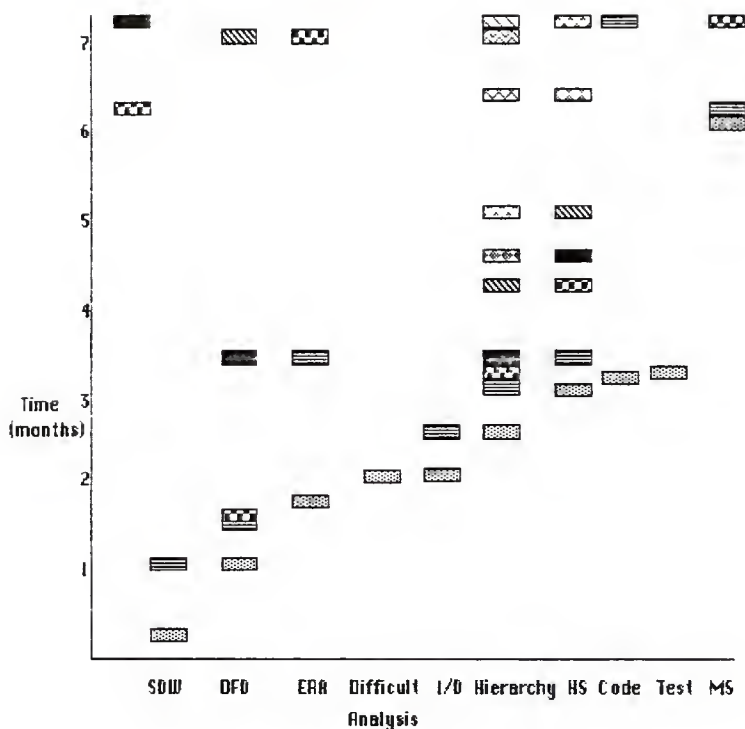


: Third pass



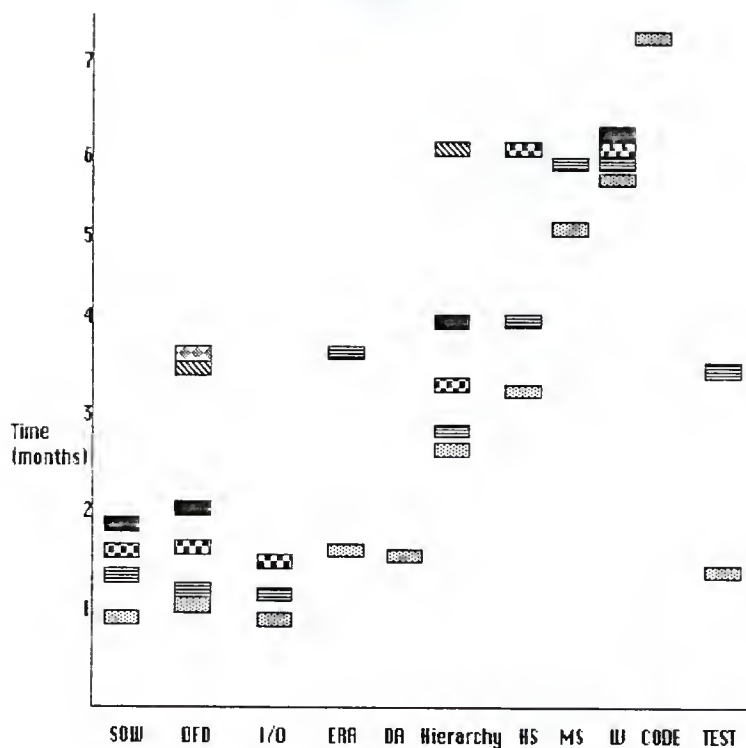
SPM diagram for group C

- DFD : Data Flow Diagrams
 ERR : The Entity-Relationship-Attribute Model
 HS : Hierarchy Specification
 MS : Module Specification
- | | | | |
|--|---------------|--|---------------|
| | : First pass | | : Fourth pass |
| | : Second pass | | : Fifth pass |
| | : Third pass | | : Sixth pass |



SPM diagram for group D

- DFD : Data Flow Diagrams
 ERR : The Entity-Relationship-Attribute Model
 MS : Module Specification
 HS : Hierarchy Specification
- | | | | |
|--|---------------|--|----------------|
| | : First pass | | : Sixth pass |
| | : Second pass | | : Seventh pass |
| | : Third pass | | : Eighth pass |
| | : Fourth pass | | : Ninth pass |
| | : Fifth pass | | : Tenth pass |



SPM diagram for group E

DFD : Data Flow Diagrams

ERA : The Entity-Relationship-Attribute Model

HS : Hierarchy Specification

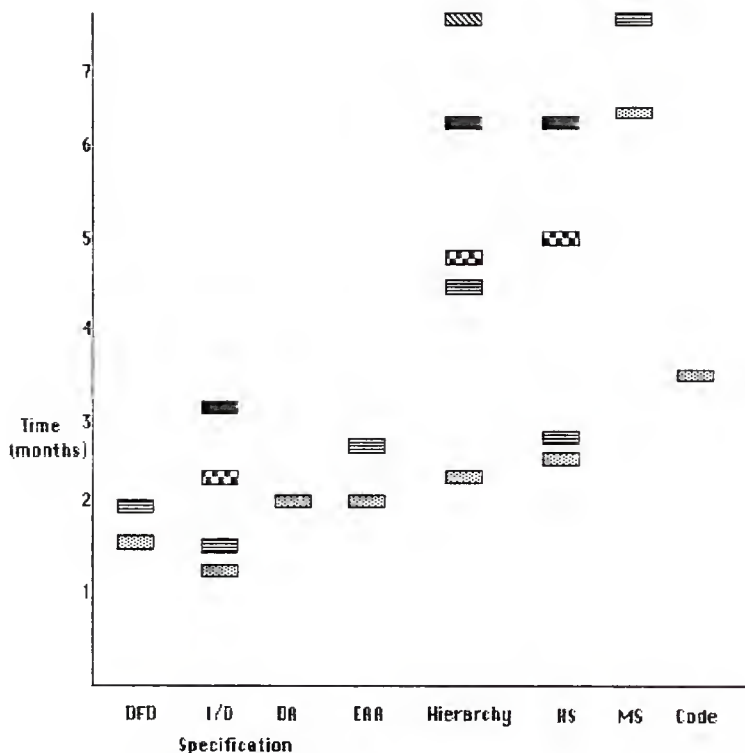
 : First pass

 : Fourth pass

 : Second pass

 : Fifth pass

 : Third pass




SPM diagram for group F


DFD : Data Flow Diagrams

ERA : The Entity-Relationship-Attribute Model


HS : Hierarchy Specification

 : First pass

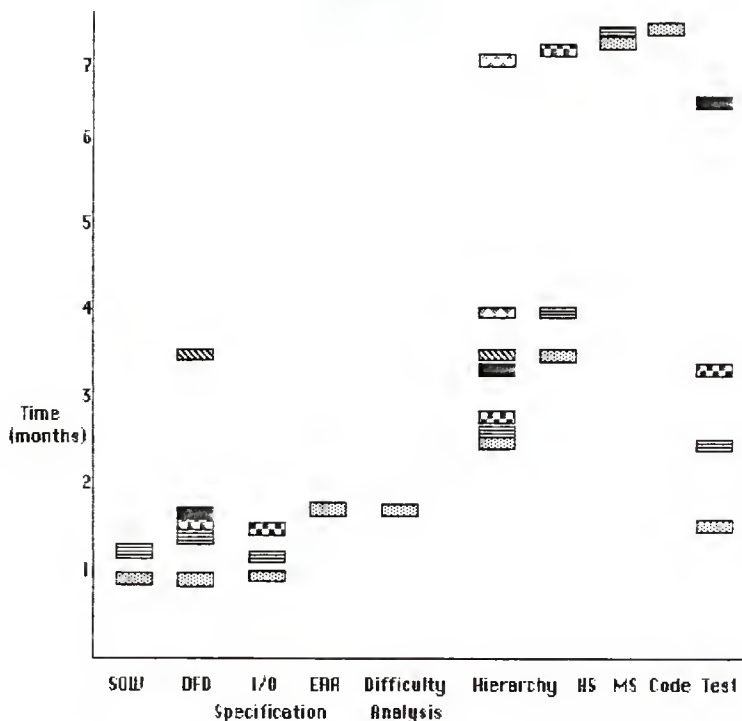
 : Second pass

 : Third pass

 : Fourth pass

 : Fifth pass

(Note: I/O Specification includes Phase One Input Data, Input and Output Variable Trace.)



SPM diagram for group H

DFD : Data Flow Diagrams

ERA : The Entity-Relationship-Attribute Model

HS : Hierarchy Specification



: First pass



: Fifth pass



: Second pass



: Sixth pass



: Third pass

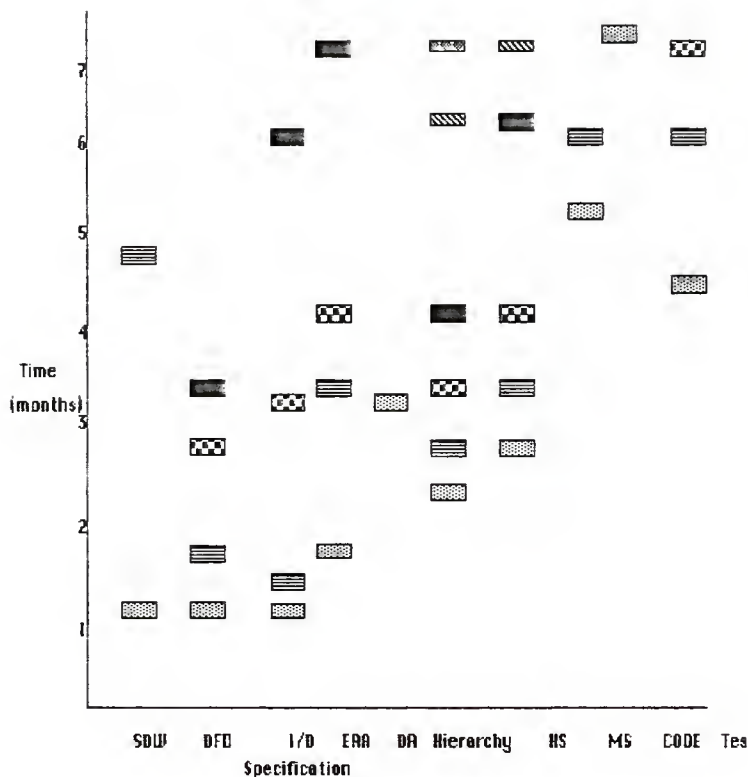


: Seventh pass









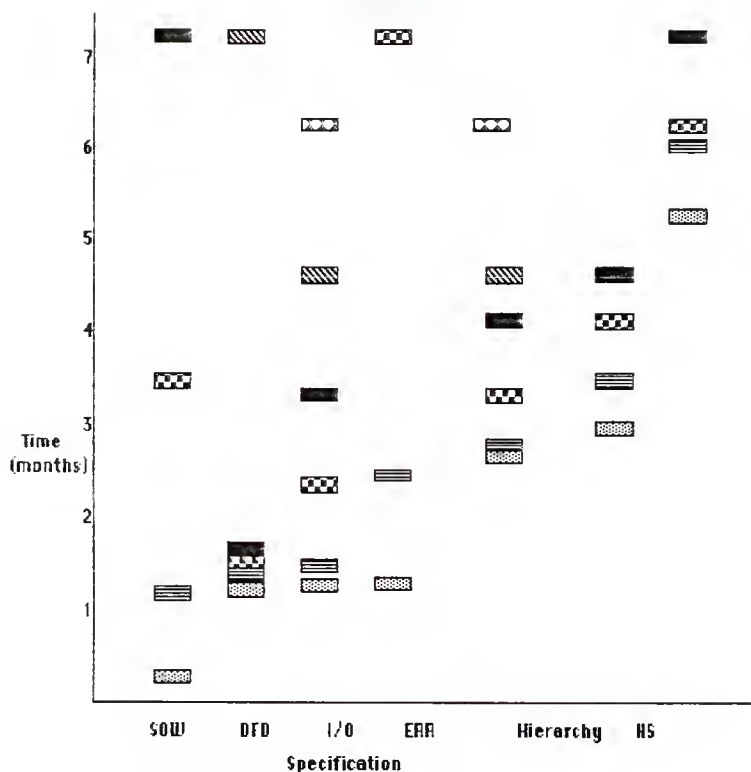
: Fourth pass

(Note: I/O Description includes Input description and Input/Output)



SPM diagram for group I

- DFD : Data Flow Diagrams
 ERA : The Entity-Relationship-Attribute Model
 HS : Hierarchy Specification
- | | |
|---|---|
|  : First pass |  : Fourth pass |
|  : Second pass |  : Fifth pass |
|  : Third pass |  : Sixth pass |



SPM diagram for group J

DFD : Data Flow Diagrams

ERR : The Entity-Relationship-Attribute Model

HS : Hierarchy Specification



: First pass



:Fourth pass



: Second pass



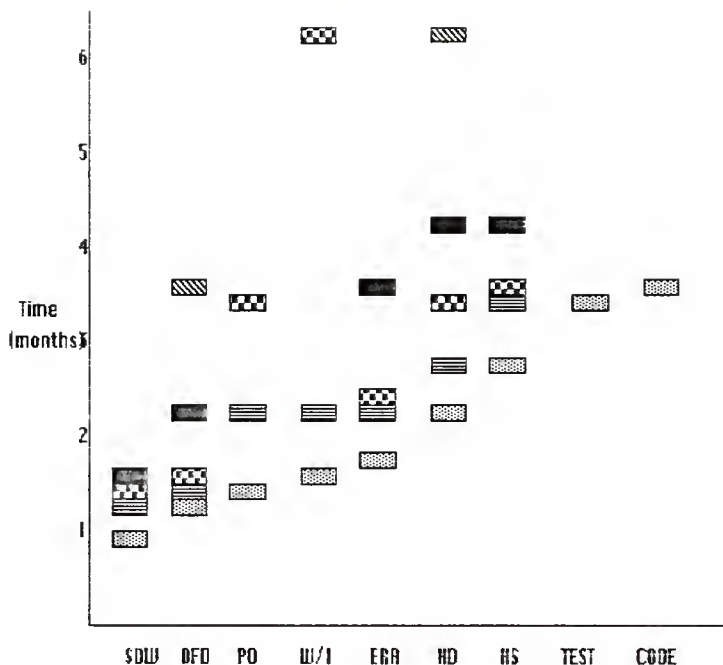
:Fifth pass



:Third pass



:Sixth pass



SPM diagram for group K

SDW : Statement Of Work

DFD/ERA : Data Flow Diagrams/The Entity-Relationship-Attribute Model

PO : Phase One (Specification Trace Tool)

W/I : Walkthrough/ Input

HD : Hierarchy Diagram

HS : Hierarchy Specification

: First pass

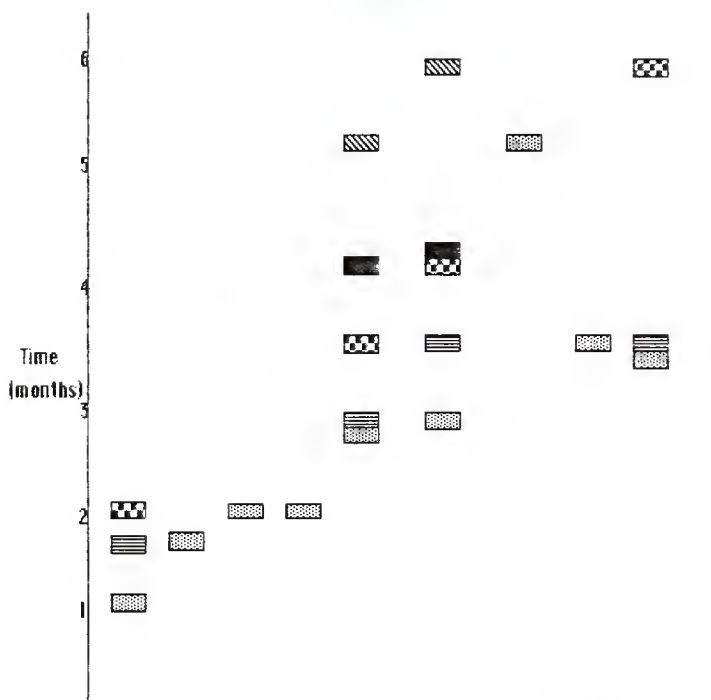
: Second pass

: Third pass

: Fourth pass

: Fifth pass

: Sixth pass



DFD Input ERA DA Hierarchy HS MS Source Test

SPM diagram for group L


DFD : Data Flow Diagrams

ERA : The Entity-Relationship-Attribute Model

HS : Hierarchy Specification

MS : Module specification

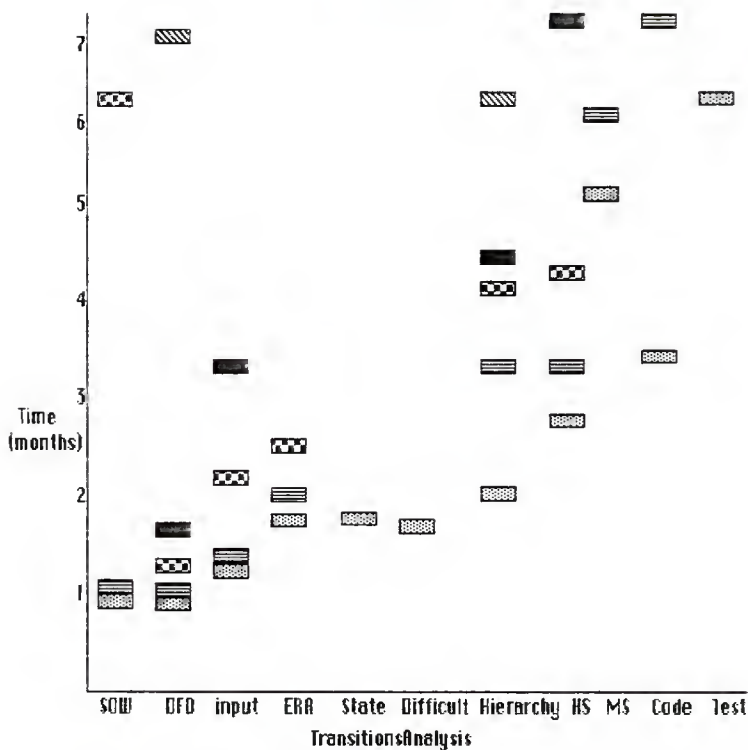
 : First pass

 : Second pass






 : Third pass

 : Fourth pass

 : Fifth pass

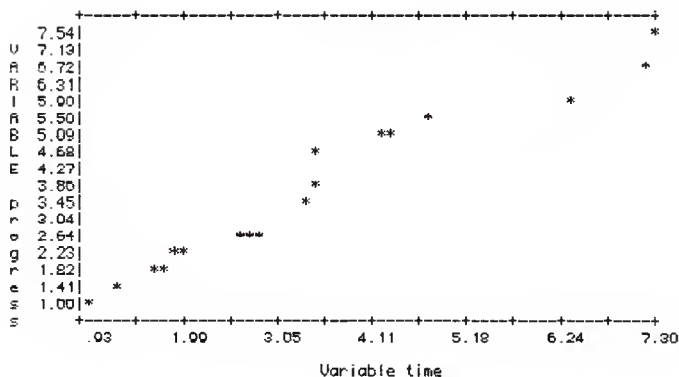
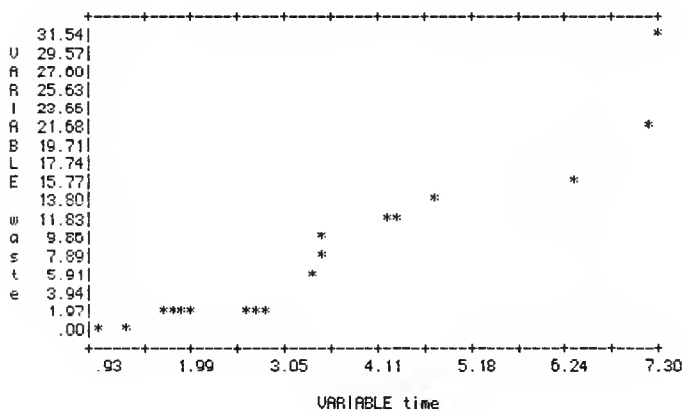


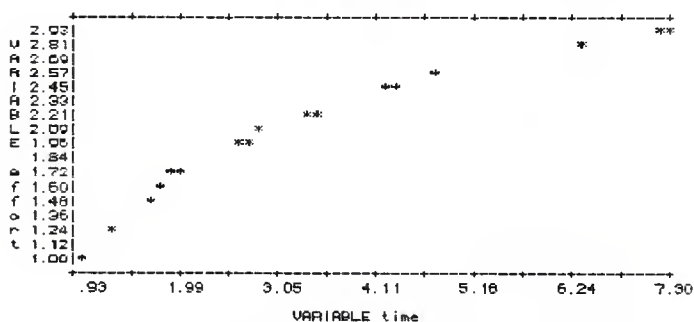
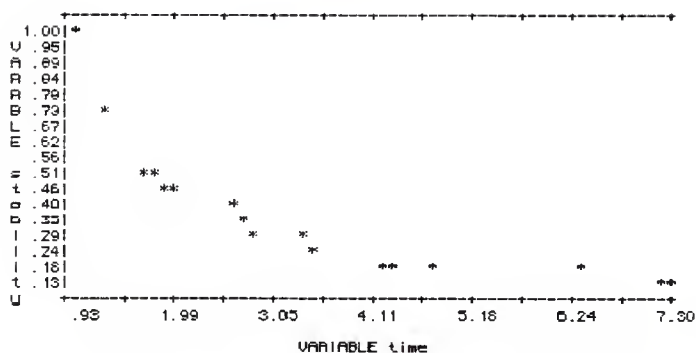
SPM diagram for group M

- DFD : Data Flow Diagrams
 ERA : The Entity-Relationship-Attribute Model
 HS : Hierarchy Specification
-  : First pass
  : Fourth pass
 : Second pass
  : Fifth pass
 : Third pass

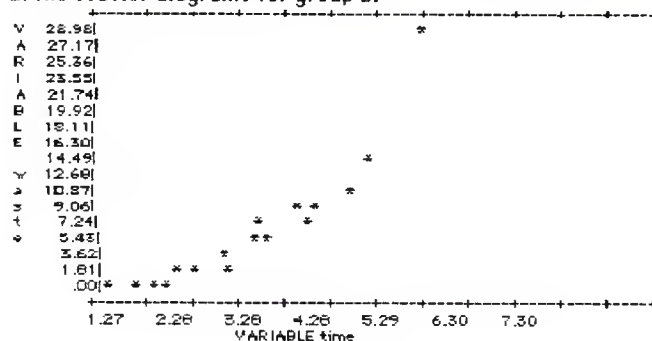
APPENDIX B: The scatter diagrams between completion time and waste, progress, stability and effort measures for each group.

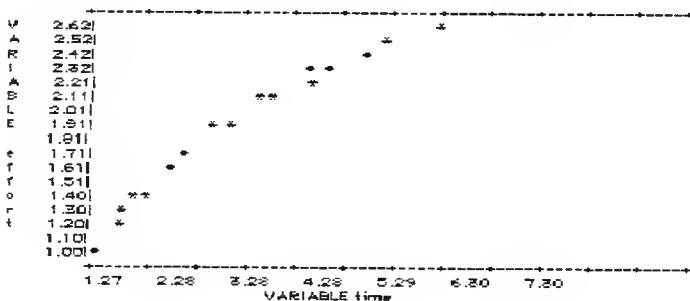
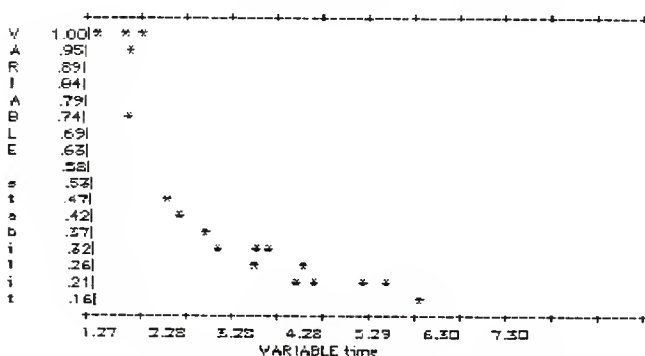
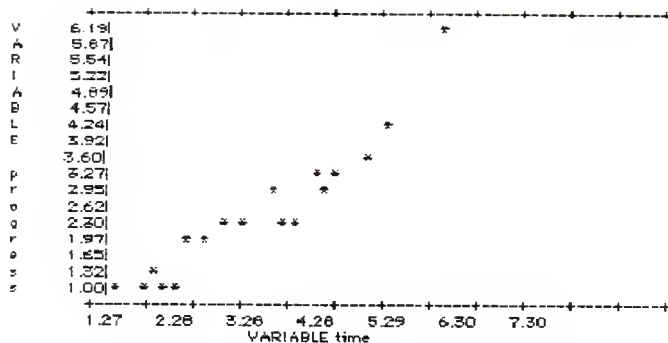
1. The scatter diagrams for group A:



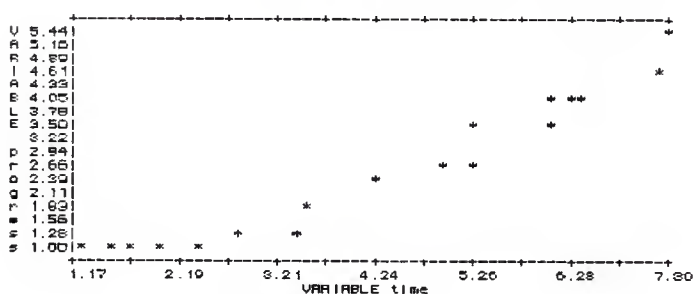
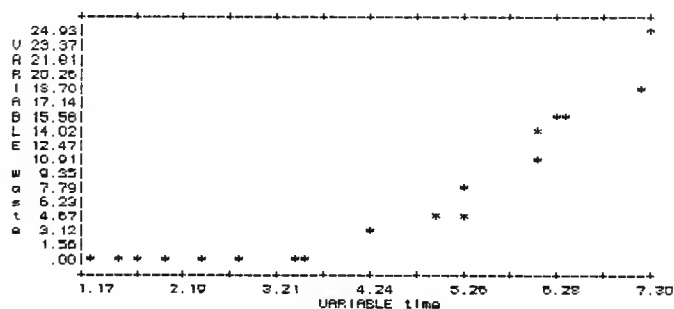


2. The scatter diagrams for group B:

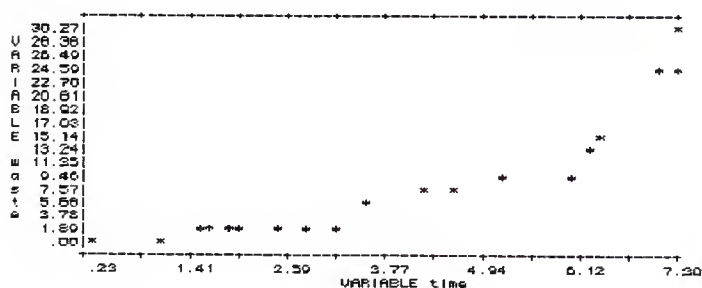


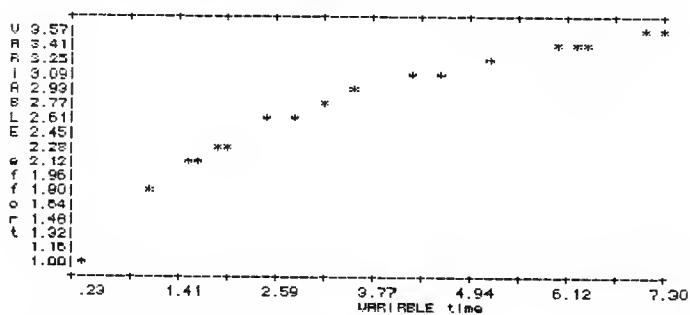
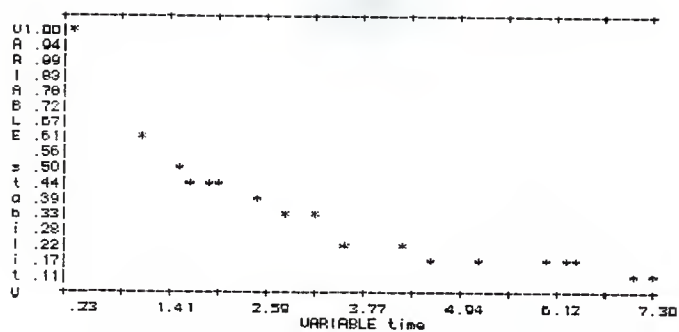


3. The scatter diagrams for group C:

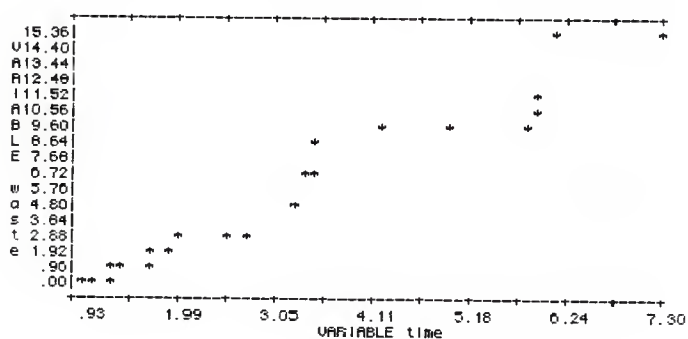


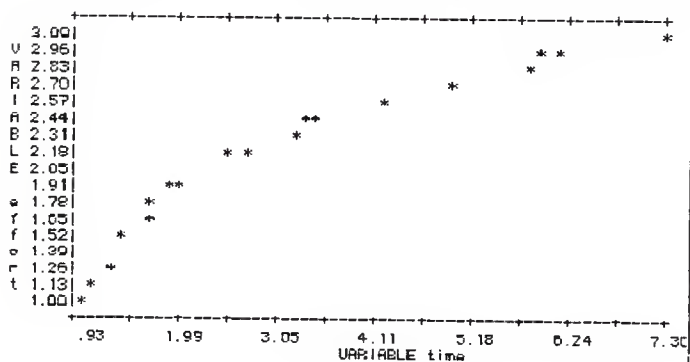
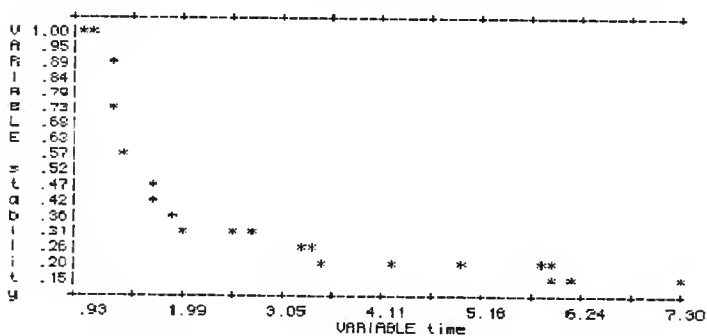
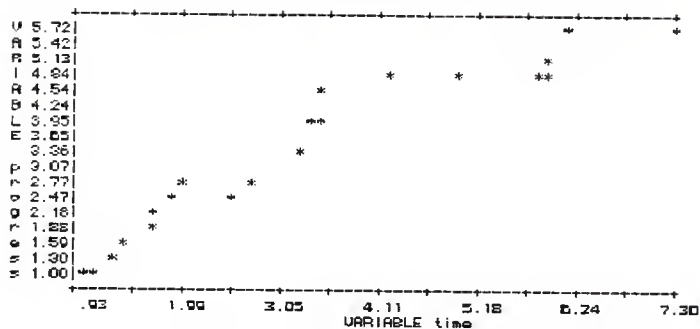
4. The scatter diagrams for group D:



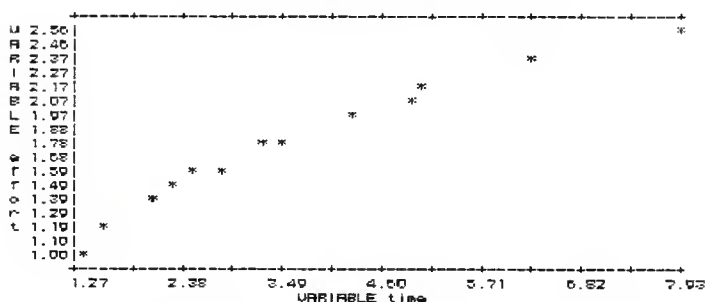
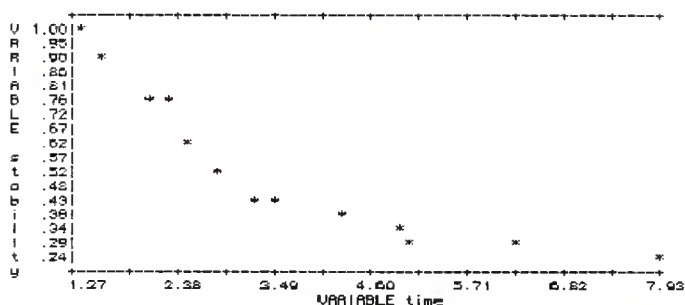


5. The scatter diagrams for group E:

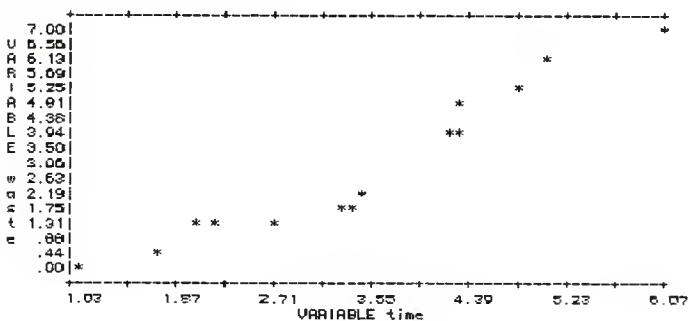


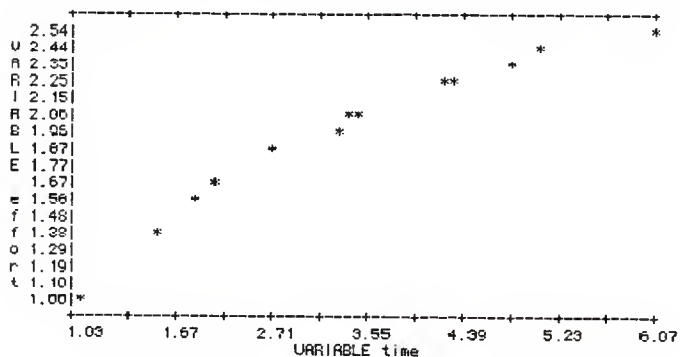
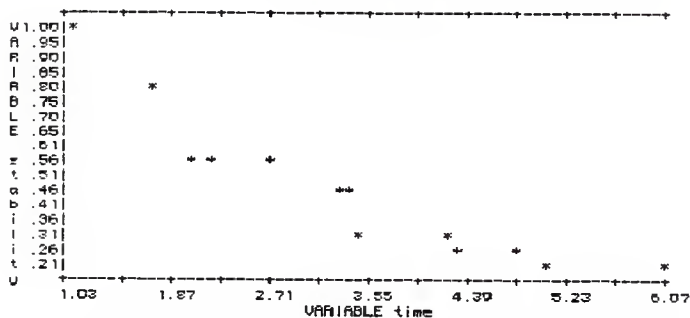
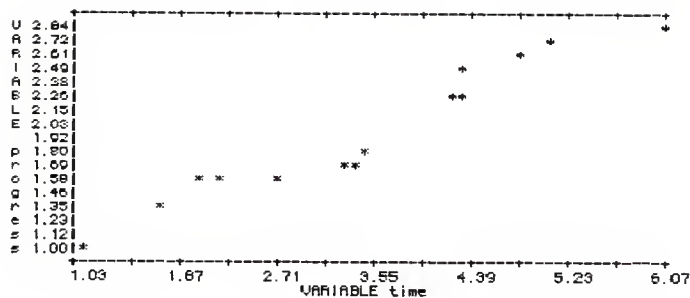


6. The scatter diagrams for group F:

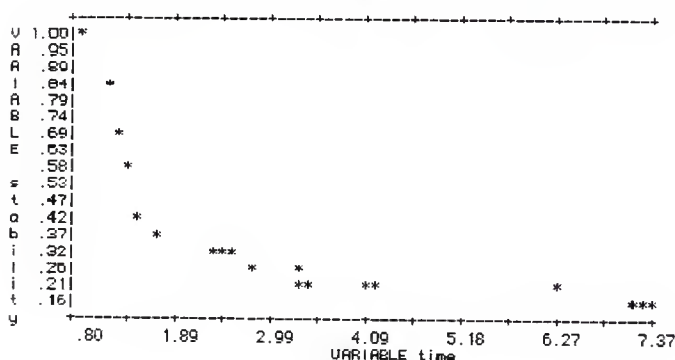
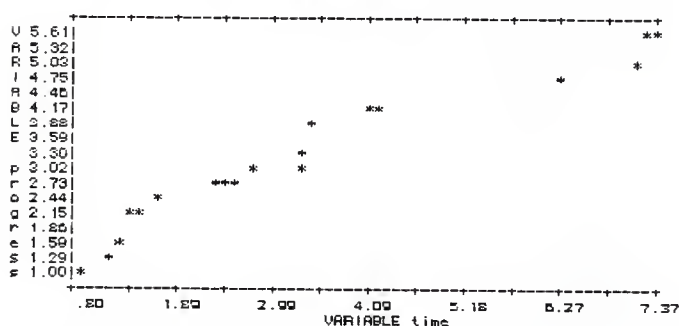
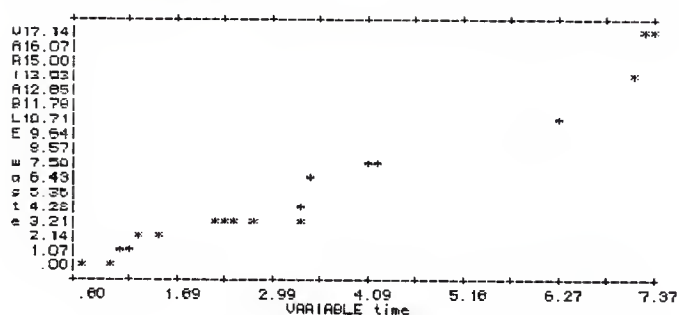


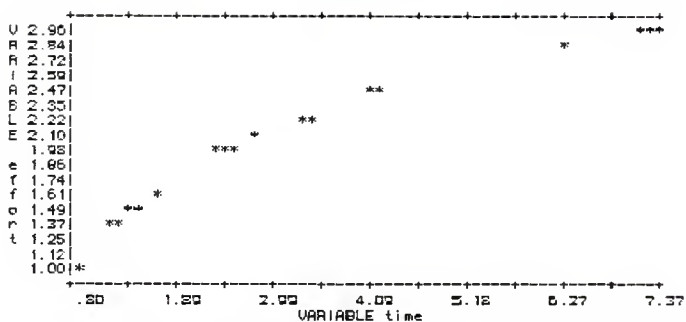
7. The scatter diagrams for group G:



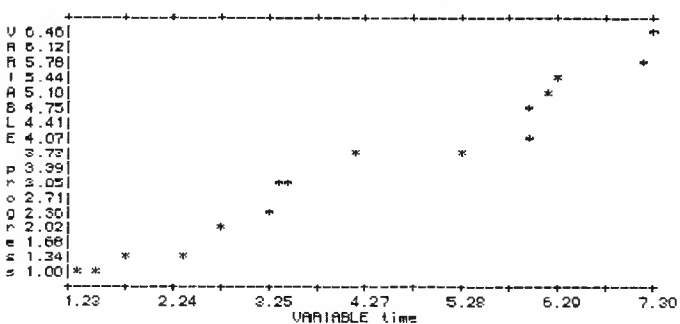
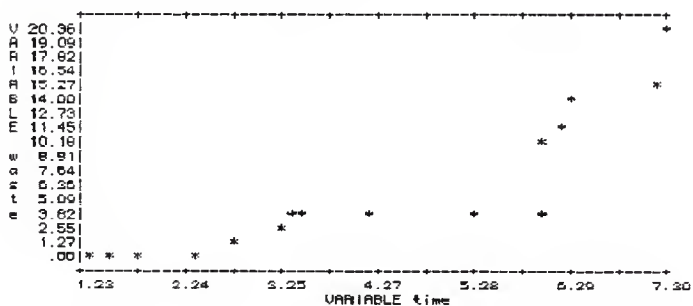


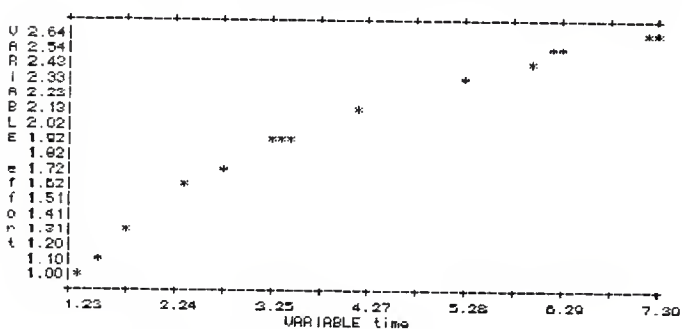
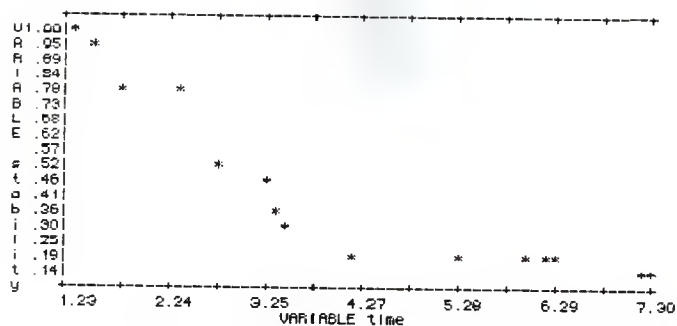
8. The scatter diagrams for group H:



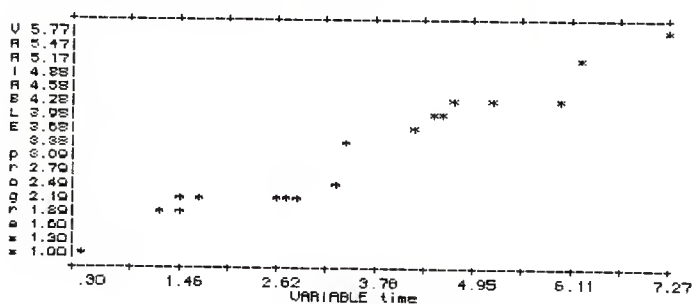


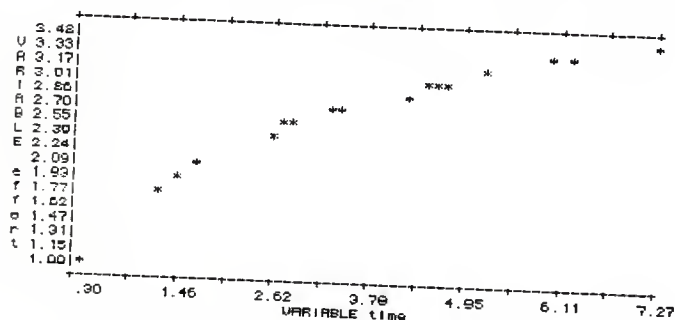
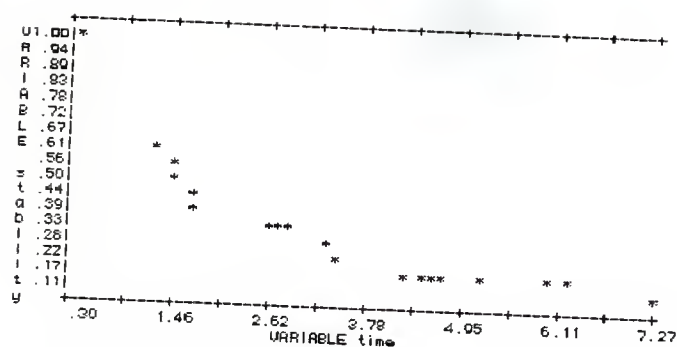
9. The scatter diagrams for group I:



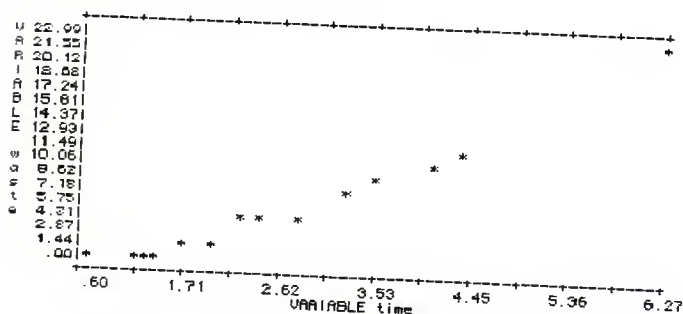


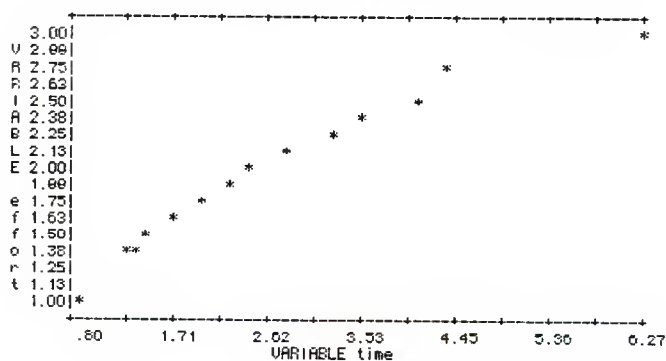
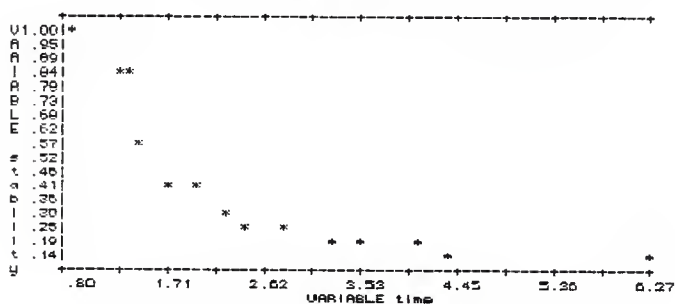
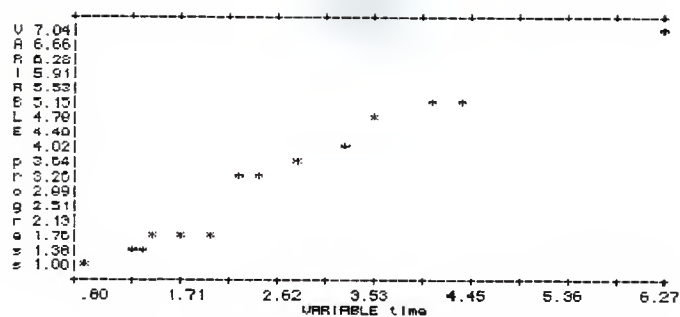
10. The scatter diagrams for group J:



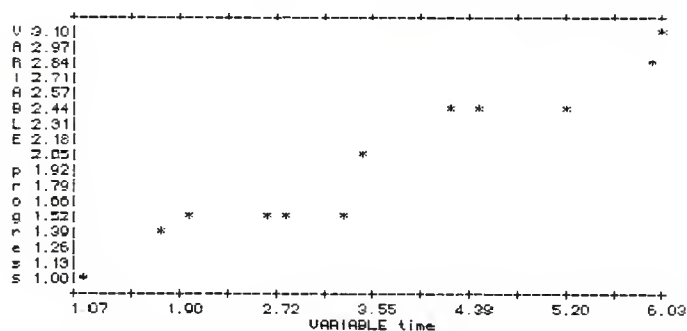
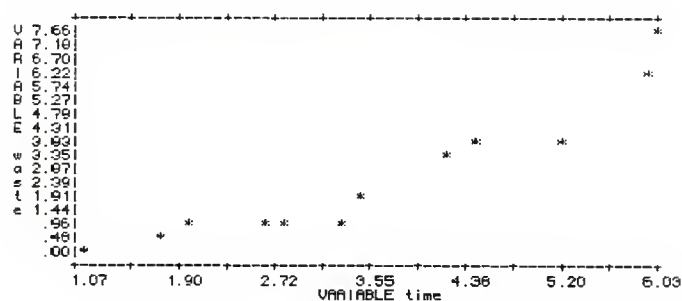


11. The scatter diagrams for group K:

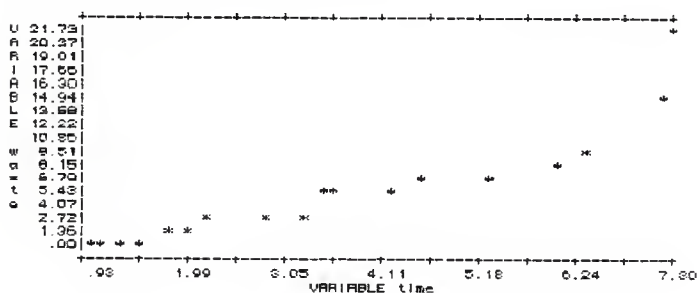


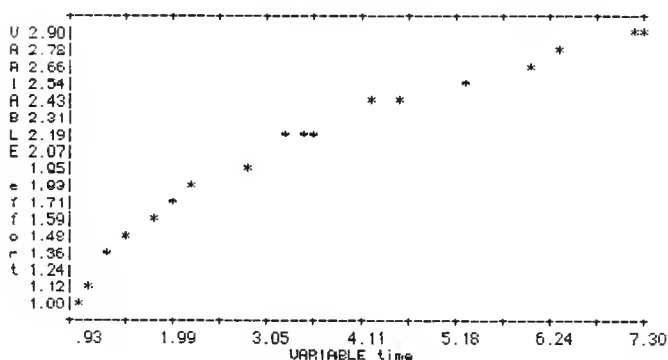
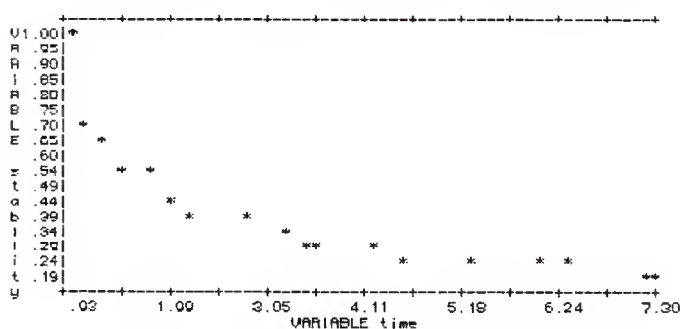
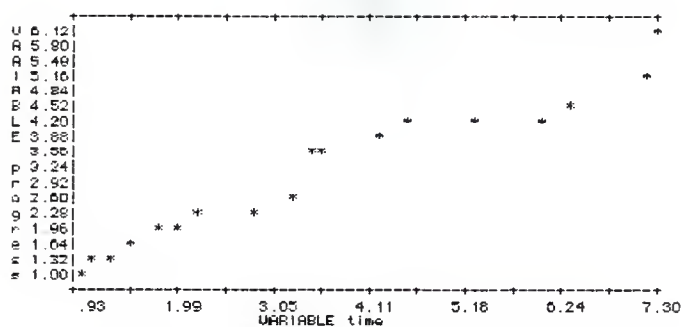


12. The scatter diagrams for group L:



13. The scatter diagrams for group M:





APPENDIX C: The projection models for the waste, progress, stability and effort measures for each group.

1. The projection models for group A:

$$\text{Waste}(x) = -6.581 + 4.189 * C(t).$$

$$\text{Progress}(x) = 0.328 + 0.971 * C(t).$$

$$\text{Stability}(x) = 0.665 + -0.091 * C(t).$$

$$\text{Effort}(x) = 1.150 + 0.2727 * C(t).$$

2. The projection models for group B:

$$\text{Waste}(x) = -7.358 + 0.899 * C(t).$$

$$\text{Progress}(x) = -0.218 + 0.720 * C(t).$$

$$\text{Stability}(x) = 1.083 + (-0.160) * C(t).$$

$$\text{Effort}(x) = 0.889 + 0.266 * C(t).$$

3. The projection models for group C:

$$\text{Waste}(x) = -5.326 + 0.891 * C(t).$$

$$\text{Progress}(x) = -0.190 + 0.639 * C(t).$$

$$\text{Stability}(x) = 0.826 + (-0.084) * C(t).$$

$$\text{Effort}(x) = 0.730 + 0.301 * C(t).$$

4. The projection models for group D:

$$\text{Waste}(x) = -5.476 + 3.429 * C(t).$$

$$\text{Progress}(x) = 0.928 + 0.545 * C(t).$$

$$\text{Stability}(\kappa) = 0.658 + (-0.848) * C(t).$$

$$\text{Effort}(\kappa) = 1.649 + 0.290 * C(t).$$

5. The projection models for group E:

$$\text{Waste}(\kappa) = -2.359 + 2.402 * C(t).$$

$$\text{Progress}(\kappa) = 0.816 + 0.754 * C(t).$$

$$\text{Stability}(\kappa) = 0.738 + (-0.105) * C(t).$$

$$\text{Effort}(\kappa) = 1.110 + 0.314 * C(t).$$

6. The projection models for group F:

$$\text{Waste}(\kappa) = -3.163 + 2.057 * C(t).$$

$$\text{Progress}(\kappa) = 0.485 + 0.484 * C(t).$$

$$\text{Stability}(\kappa) = 0.938 + (-0.111) * C(t).$$

$$\text{Effort}(\kappa) = 0.952 + 0.224 * C(t).$$

7. The projection models for group G:

$$\text{Waste}(\kappa) = -2.226 + 1.515 * C(t).$$

$$\text{Progress}(\kappa) = 0.658 + 0.373 * C(t).$$

$$\text{Stability}(\kappa) = 0.935 + (-0.139) * C(t).$$

$$\text{Effort}(\kappa) = 0.974 + 0.284 * C(t).$$

8. The projection models for group H:

$$\text{Waste}(\kappa) = -2.672 + 2.446 * C(t).$$

$$\text{Progress}(\kappa) = 1.086 + 0.636 * C(t).$$

$$\text{Stability}(\kappa) = 0.685 + (-0.090) * C(t).$$

$$\text{Effort}(\kappa) = 1.161 + 0.271 * C(t).$$

9. The projection models for group I:

$$\text{Waste}(x) = -5.698 + 2.831 * C(t),$$

$$\text{Progress}(x) = -0.186 + 0.858 * C(t),$$

$$\text{Stability}(x) = 0.986 + (-0.132) * C(t),$$

$$\text{Effort}(x) = 0.896 + 0.257 * C(t),$$

10. The projection models for group J:

$$\text{Waste}(x) = -3.504 + 2.405 * C(t),$$

$$\text{Progress}(x) = 0.917 + 0.646 * C(t),$$

$$\text{Stability}(x) = 0.670 + (-0.093) * C(t),$$

$$\text{Effort}(x) = 1.469 + 0.322 * C(t),$$

11. The projection models for group K:

$$\text{Waste}(x) = -5.547 + 4.150 * C(t),$$

$$\text{Progress}(x) = 0.531 + 1.191 * C(t),$$

$$\text{Stability}(x) = 0.811 + (-0.151) * C(t),$$

$$\text{Effort}(x) = 0.993 + 0.363 * C(t),$$

12. The projection models for group L:

$$\text{Waste}(x) = -2.263 + 1.539 * C(t),$$

$$\text{Progress}(x) = 0.600 + 0.386 * C(t),$$

$$\text{Stability}(x) = 0.990 + (-0.137) * C(t),$$

$$\text{Effort}(x) = 0.859 + 0.320 * C(t),$$

10. The projection models for group J:

$$\text{Waste}(x) = -3.504 + 2.405 * C(t),$$

$$\text{Progress}(x) = 0.917 + 0.646 * C(t).$$

$$\text{Stability}(x) = 0.670 + (-0.093) * C(t).$$

$$\text{Effort}(x) = 1.469 + 0.322 * C(t).$$

13. The projection models for group M:

$$\text{Waste}(x) = -3.660 + 2.517 * C(t).$$

$$\text{Progress}(x) = 0.613 + 0.683 * C(t).$$

$$\text{Stability}(x) = 0.702 + (-0.084) * C(t).$$

$$\text{Effort}(x) = 1.057 + 0.280 * C(t).$$

APPENDIX D: The values of MRE for each group.

NOTE: 1. Waste, progress, stability and effort indicate the total values of each measure.

2. W, P, S and E represent the projection values of each measure.

3. MRE for the waste measure can not be calculated whenever the total waste is zero since the denominator can not be zero.

1. The MRE for group A:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	time	waste	w	MRE	prog	p	MRE	stab	s	MRE	effort	e	MRE
2	0.93	0.00	-2.69		1.00	1.23	0.23	1.00	0.58	0.42	1.00	1.40	0.40
3	1.23	0.60	-1.43	3.38	1.49	1.52	0.02	0.71	0.55	0.22	1.24	1.49	0.20
4	1.67	1.48	0.41	0.72	2.01	1.95	0.03	0.49	0.51	0.05	1.51	1.61	0.06
5	1.80	1.48	0.96	0.35	2.01	2.08	0.03	0.49	0.50	0.02	1.58	1.64	0.04
6	1.90	1.71	1.38	0.19	2.14	2.17	0.02	0.43	0.49	0.14	1.70	1.67	0.02
7	2.00	1.71	1.80	0.05	2.14	2.27	0.06	0.43	0.48	0.12	1.75	1.70	0.03
8	2.60	2.64	4.31	0.63	2.49	2.85	0.15	0.38	0.43	0.13	1.98	1.86	0.06
9	2.70	2.74	4.73	0.73	2.53	2.95	0.17	0.34	0.42	0.23	2.02	1.89	0.07
10	2.80	2.94	5.15	0.75	2.73	3.05	0.12	0.32	0.41	0.28	2.05	1.91	0.07
11	3.33	5.00	7.37	0.47	3.35	3.56	0.06	0.27	0.36	0.34	2.21	2.06	0.07
12	3.47	7.14	7.95	0.11	3.97	3.70	0.07	0.24	0.35	0.45	2.25	2.10	0.07
13	3.50	9.54	8.08	0.15	4.65	3.73	0.20	0.22	0.35	0.57	2.26	2.10	0.07
14	4.23	11.03	11.14	0.01	5.00	4.44	0.11	0.20	0.28	0.40	2.43	2.30	0.05
15	4.37	11.17	11.72	0.05	5.04	4.57	0.09	0.19	0.27	0.41	2.47	2.34	0.05
16	4.80	13.47	13.53	0.00	5.52	4.99	0.10	0.16	0.23	0.42	2.56	2.46	0.04
17	6.33	16.47	19.94	0.21	5.99	6.47	0.08	0.16	0.09	0.45	2.80	2.88	0.03
18	7.20	21.27	23.58	0.11	6.66	7.32	0.10	0.15	0.01	0.94	2.92	3.11	0.07
19	7.30	31.54	24.00	0.24	7.54	7.42	0.02	0.13	0.00	1.00	2.93	3.14	0.07
20				MRE=			MRE=			MRE=			MRE=
21				0.48			0.09			0.37			0.06

2. The MRE for group B:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	time	waste	w	MRE	prog	p	MRE	stab	s	MRE	effort	e	MRE
2	1.27	0.00	-2.88		1.00	0.70	0.30	1.00	0.88	0.12	1.00	1.23	0.23
3	1.63	0.00	-1.60		1.00	0.96	0.04	1.00	0.82	0.18	1.22	1.32	0.08
4	1.70	0.00	-1.36		1.00	1.01	0.01	1.00	0.81	0.19	1.26	1.34	0.06
5	1.90	0.00	-0.65		1.00	1.15	0.15	1.00	0.78	0.22	1.37	1.39	0.02
6	1.93	0.26	-0.55	3.10	1.13	1.17	0.04	0.94	0.78	0.18	1.38	1.40	0.02
7	2.00	0.45	-0.30	1.66	1.32	1.22	0.07	0.72	0.76	0.06	1.42	1.42	0.00
8	2.57	1.96	1.71	0.13	1.91	1.63	0.15	0.49	0.67	0.37	1.64	1.57	0.04
9	2.80	2.19	2.52	0.15	1.99	1.80	0.10	0.40	0.64	0.59	1.72	1.63	0.05
10	3.27	2.66	4.18	0.57	2.28	2.14	0.06	0.37	0.56	0.52	1.87	1.76	0.06
11	3.50	4.23	5.00	0.18	2.28	2.30	0.01	0.31	0.52	0.69	1.93	1.82	0.06
12	4.10	4.83	7.11	0.47	2.42	2.73	0.13	0.31	0.43	0.38	2.08	1.98	0.05
13	4.20	4.83	7.47	0.55	2.42	2.81	0.16	0.31	0.41	0.33	2.10	2.00	0.05
14	4.23	6.75	7.57	0.12	2.88	2.83	0.02	0.27	0.41	0.51	2.11	2.01	0.05
15	5.00	7.55	10.29	0.36	3.04	3.38	0.11	0.26	0.28	0.10	2.26	2.22	0.02
16	5.03	9.15	10.40	0.14	3.36	3.40	0.01	0.23	0.28	0.22	2.27	2.23	0.02
17	5.23	9.38	11.10	0.18	3.40	3.55	0.04	0.22	0.25	0.13	2.31	2.28	0.01
18	6.03	10.18	13.93	0.37	3.53	4.12	0.17	0.21	0.12	0.43	2.44	2.49	0.02
19	6.33	13.94	14.98	0.07	4.13	4.34	0.05	0.20	0.07	0.64	2.49	2.57	0.03
20	7.30	28.98	18.41	0.36	6.19	5.04	0.19	0.16	-0.08	1.52	2.62	2.63	0.08
21				MRE=			MRE=			MRE=			MRE=
22				0.56			0.10			0.39			0.05

3. The MRE for group C:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	time	waste	w	MRE	prog	p	MRE	stab	s	MRE	effort	e	MRE
2	1.17	0.00	-2.06		1.00	0.56	0.44	1.00	0.73	0.27	1.00	1.08	0.08
3	1.50	0.00	-1.14		1.00	0.77	0.23	1.00	0.70	0.30	1.22	1.18	0.03
4	1.73	0.23	-0.49	3.15	1.13	0.92	0.19	0.93	0.68	0.27	1.35	1.25	0.07
5	1.97	0.23	0.18	0.24	1.13	1.07	0.05	0.77	0.66	0.14	1.47	1.32	0.10
6	2.43	0.23	1.46	5.35	1.13	1.36	0.21	0.77	0.62	0.19	1.66	1.46	0.12
7	2.80	0.60	2.49	3.16	1.27	1.60	0.26	0.65	0.59	0.09	1.80	1.57	0.13
8	3.37	0.60	4.09	5.81	1.27	1.96	0.55	0.65	0.54	0.16	1.97	1.74	0.12
9	3.50	0.60	4.45	6.42	1.77	2.05	0.16	0.54	0.53	0.01	2.00	1.78	0.11
10	4.23	3.46	6.49	0.88	2.45	2.51	0.03	0.42	0.47	0.12	2.18	2.00	0.08
11	4.93	4.86	8.44	0.74	2.73	2.96	0.08	0.34	0.41	0.21	2.32	2.21	0.05
12	5.23	4.86	9.28	0.91	2.73	3.15	0.16	0.34	0.39	0.14	2.37	2.30	0.03
13	5.27	8.16	9.39	0.15	3.36	3.18	0.05	0.31	0.38	0.24	2.38	2.31	0.03
14	6.03	11.46	11.52	0.00	3.49	3.67	0.05	0.29	0.32	0.10	2.44	2.54	0.04
15	6.07	14.16	11.63	0.18	3.93	3.69	0.06	0.27	0.32	0.17	2.45	2.55	0.04
16	6.30	15.53	12.27	0.21	4.15	3.84	0.08	0.25	0.30	0.19	2.48	2.62	0.06
17	6.33	15.56	12.35	0.21	4.16	3.86	0.07	0.24	0.29	0.23	2.49	2.63	0.06
18	7.20	19.26	14.78	0.23	4.67	4.41	0.05	0.22	0.22	0.01	2.61	2.89	0.11
19	7.30	24.93	15.06	0.40	5.44	4.48	0.18	0.19	0.21	0.12	2.62	2.92	0.12
20				MRE=			MRE=			MRE=			MRE=
21				1.75			0.16			0.17			0.08

4. The MRE for group D:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	time	waste	w	MRE	prog	p	MRE	stab	s	MRE	effort	e	MRE
2	0.23	0.00	-4.69		1.00	1.17	0.17	1.00	0.64	0.36	1.00	1.72	0.72
3	1.03	0.80	-1.94	3.43	1.78	1.61	0.10	0.61	0.57	0.06	1.78	1.95	0.09
4	1.53	1.30	-0.23	1.18	2.10	1.88	0.10	0.52	0.53	0.02	2.10	2.09	0.00
5	1.70	1.47	0.35	0.76	2.20	1.97	0.10	0.44	0.51	0.17	2.20	2.14	0.03
6	1.83	1.47	0.80	0.46	2.20	2.05	0.07	0.44	0.50	0.14	2.27	2.18	0.04
7	2.00	1.47	1.38	0.06	2.20	2.14	0.03	0.44	0.49	0.11	2.36	2.23	0.06
8	2.47	1.94	2.99	0.54	2.39	2.39	0.00	0.39	0.45	0.15	2.55	2.36	0.07
9	2.80	2.27	4.13	0.82	2.51	2.57	0.03	0.36	0.42	0.17	2.67	2.46	0.08
10	3.23	2.70	5.60	1.07	2.64	2.81	0.06	0.34	0.36	0.13	2.80	2.58	0.08
11	3.50	5.34	6.53	0.22	2.92	2.96	0.01	0.22	0.36	0.64	2.88	2.66	0.08
12	4.23	6.80	9.03	0.33	3.27	3.35	0.03	0.21	0.30	0.43	3.05	2.87	0.06
13	4.63	7.60	10.40	0.37	3.44	3.57	0.04	0.19	0.27	0.40	3.14	2.99	0.05
14	5.17	8.68	12.25	0.41	3.65	3.87	0.06	0.17	0.22	0.29	3.24	3.15	0.03
15	6.00	8.68	15.10	0.74	3.65	4.32	0.18	0.17	0.15	0.12	3.38	3.39	0.00
16	6.03	8.71	15.20	0.75	3.65	4.33	0.19	0.17	0.15	0.13	3.38	3.39	0.00
17	6.27	13.95	16.03	0.15	4.49	4.47	0.01	0.16	0.13	0.21	3.42	3.46	0.01
18	6.33	15.85	16.23	0.02	4.86	4.50	0.07	0.14	0.12	0.13	3.43	3.48	0.02
19	7.07	23.73	18.77	0.21	4.96	4.90	0.01	0.13	0.06	0.55	3.54	3.70	0.04
20	7.27	23.93	19.46	0.19	4.99	5.01	0.00	0.12	0.04	0.65	3.56	3.75	0.05
21	7.30	30.27	19.56	0.35	6.00	5.03	0.16	0.11	0.04	0.64	3.57	3.76	0.05
22				MRE=			MRE=			MRE=			MRE=
23				0.63			0.07			0.28			0.08

5. The MRE for group E:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	time	waste	w	MRE	prog	p	MRE	stab	s	MRE	effort	e	MRE
2	0.93	0.00	-0.13		1.00	1.52	0.52	1.00	0.64	0.36	1.00	1.40	0.40
3	1.00	0.00	0.04		1.00	1.57	0.57	1.00	0.63	0.37	1.07	1.42	0.33
4	1.20	0.27	0.52	0.94	1.23	1.72	0.40	0.89	0.61	0.31	1.24	1.48	0.19
5	1.23	0.50	0.60	0.19	1.41	1.74	0.24	0.73	0.61	0.17	1.26	1.49	0.18
6	1.33	0.90	0.84	0.07	1.71	1.82	0.06	0.60	0.60	0.00	1.51	1.52	0.01
7	1.40	0.90	1.00	0.12	1.71	1.87	0.09	0.60	0.59	0.02	1.56	1.54	0.01
8	1.63	1.20	1.56	0.30	1.98	2.05	0.03	0.45	0.57	0.26	1.70	1.62	0.05
9	1.67	1.64	1.65	0.01	2.24	2.08	0.07	0.40	0.56	0.41	1.72	1.63	0.05
10	1.93	2.24	2.28	0.02	2.55	2.27	0.11	0.36	0.53	0.49	1.86	1.71	0.08
11	2.00	2.57	2.44	0.05	2.72	2.32	0.15	0.32	0.53	0.65	1.89	1.73	0.08
12	2.57	2.57	3.81	0.48	2.62	2.75	0.05	0.32	0.47	0.46	2.12	1.91	0.10
13	2.77	2.77	4.29	0.55	2.79	2.91	0.04	0.30	0.45	0.49	2.19	1.97	0.10
14	3.27	4.64	5.50	0.18	3.36	3.28	0.02	0.28	0.39	0.41	2.34	2.13	0.09
15	3.40	6.67	5.81	0.13	3.96	3.38	0.15	0.24	0.38	0.58	2.38	2.17	0.09
16	3.43	6.67	5.88	0.12	3.96	3.40	0.14	0.22	0.38	0.71	2.39	2.18	0.09
17	3.47	8.54	5.98	0.30	4.50	3.43	0.24	0.21	0.37	0.78	2.40	2.19	0.09
18	4.17	10.05	7.66	0.24	4.86	3.96	0.19	0.21	0.30	0.43	2.58	2.41	0.06
19	5.00	10.05	9.65	0.04	4.86	4.59	0.06	0.19	0.21	0.12	2.74	2.67	0.02
20	5.83	10.05	11.64	0.16	4.86	5.21	0.07	0.19	0.12	0.34	2.89	2.93	0.01
21	5.90	10.12	11.81	0.17	4.87	5.27	0.08	0.19	0.12	0.38	2.90	2.96	0.02
22	5.93	10.15	11.88	0.17	4.87	5.29	0.09	0.18	0.11	0.36	2.90	2.96	0.02
23	5.97	11.12	11.98	0.08	5.04	5.32	0.06	0.17	0.11	0.35	2.91	2.98	0.02
24	6.17	15.36	12.46	0.19	5.72	5.47	0.04	0.15	0.09	0.41	2.95	3.04	0.03
25	7.30	15.36	15.18	0.01	5.72	6.32	0.11	0.15	-0.03	1.20	3.10	3.39	0.09
26				MRE=			MRE=			MRE=			MRE=
27				0.17			0.15			0.42			0.09

6. The MRE for group F:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	time	waste	w	MRE	prog	p	MRE	stab	s	MRE	effort	e	MRE
2	1.27	0.00	-0.55		1.00	1.10	0.10	1.00	0.80	0.20	1.00	1.24	0.24
3	1.53	0.26	-0.01	1.06	1.17	1.23	0.05	0.92	0.77	0.16	1.17	1.29	0.11
4	2.00	0.73	0.95	0.30	1.40	1.45	0.04	0.74	0.72	0.03	1.40	1.40	0.00
5	2.23	0.73	1.43	0.95	1.40	1.56	0.12	0.74	0.69	0.07	1.51	1.45	0.04
6	2.47	1.67	1.92	0.15	1.79	1.68	0.06	0.60	0.66	0.11	1.61	1.51	0.06
7	2.80	2.80	2.60	0.07	1.90	1.84	0.03	0.54	0.63	0.16	1.61	1.58	0.02
8	3.27	3.60	3.57	0.01	2.15	2.07	0.04	0.42	0.58	0.37	1.75	1.69	0.04
9	3.50	3.60	4.04	0.12	2.15	2.18	0.01	0.42	0.55	0.31	1.81	1.74	0.04
10	4.23	5.60	5.54	0.01	2.62	2.53	0.03	0.38	0.47	0.24	1.99	1.90	0.04
11	4.90	6.27	6.92	0.10	2.76	2.86	0.04	0.34	0.40	0.16	2.12	2.05	0.03
12	5.03	8.50	7.19	0.15	3.20	2.92	0.09	0.31	0.38	0.23	2.15	2.08	0.03
13	6.23	9.70	9.66	0.00	3.61	3.50	0.03	0.27	0.25	0.08	2.34	2.35	0.00
14	6.30	9.70	9.80	0.01	3.61	3.54	0.02	0.27	0.24	0.11	2.35	2.37	0.01
15	7.93	13.03	13.15	0.01	4.03	4.33	0.07	0.24	0.06	0.75	2.56	2.73	0.07
16				MRE=			MRE=			MRE=			MRE=
17				0.23			0.05			0.21			0.05

7. The MRE for group G:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	time	waste	w	MRE	prog	p	MRE	stab	s	MRE	effort	e	MRE
2	1.03	0.00	-0.67		1.00	1.04	0.04	1.00	0.79	0.21	1.00	1.27	0.27
3	1.67	0.64	0.30	0.53	1.38	1.28	0.07	0.81	0.70	0.13	1.38	1.45	0.05
4	2.00	1.30	0.80	0.38	1.55	1.40	0.09	0.54	0.66	0.21	1.55	1.54	0.01
5	2.23	1.30	1.15	0.11	1.55	1.49	0.04	0.54	0.62	0.16	1.65	1.61	0.03
6	2.70	1.30	1.86	0.43	1.55	1.66	0.07	0.54	0.56	0.03	1.83	1.74	0.05
7	3.30	1.90	2.77	0.46	1.73	1.89	0.09	0.47	0.47	0.01	2.01	1.91	0.05
8	3.37	1.90	2.88	0.51	1.73	1.91	0.11	0.47	0.46	0.01	2.03	1.93	0.05
9	3.47	2.07	3.03	0.46	1.78	1.95	0.10	0.33	0.45	0.37	2.06	1.96	0.05
10	4.23	4.07	4.18	0.03	2.25	2.23	0.01	0.30	0.35	0.15	2.24	2.18	0.03
11	4.27	4.11	4.24	0.03	2.26	2.25	0.00	0.28	0.34	0.21	2.25	2.19	0.03
12	4.33	4.97	4.33	0.13	2.46	2.27	0.08	0.26	0.33	0.27	2.26	2.20	0.02
13	4.80	5.44	5.04	0.07	2.56	2.45	0.04	0.24	0.27	0.11	2.36	2.34	0.01
14	5.03	6.20	5.39	0.13	2.71	2.53	0.07	0.22	0.23	0.06	2.40	2.40	0.00
15	6.03	7.00	6.91	0.01	2.84	2.91	0.02	0.21	0.09	0.55	2.54	2.69	0.06
16	6.07	7.00	6.97	0.00	2.84	2.92	0.03	0.21	0.09	0.58	2.54	2.70	0.06
17				MRE=			MRE=			MRE=			MRE=
18				0.22			0.06			0.20			0.05

8. The MRE for group H:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	time	waste	v	MRE	prog	p	MRE	stab	s	MRE	effort	e	MRE
2	0.80	0.00	-0.72		1.00	1.60	0.60	1.00	0.61	0.39	1.00	1.38	0.38
3	0.83	0.00	-0.64		1.00	1.61	0.61	1.00	0.61	0.39	1.04	1.39	0.33
4	1.17	0.37	0.19	0.49	1.32	1.83	0.39	0.84	0.58	0.31	1.33	1.48	0.11
5	1.20	0.74	0.26	0.64	1.62	1.85	0.14	0.68	0.58	0.15	1.35	1.49	0.10
6	1.33	1.27	0.58	0.54	2.02	1.93	0.04	0.56	0.56	0.01	1.45	1.52	0.05
7	1.43	1.63	0.83	0.49	2.27	2.00	0.12	0.41	0.56	0.35	1.52	1.55	0.02
8	1.63	1.83	1.31	0.28	2.40	2.12	0.12	0.37	0.54	0.45	1.64	1.60	0.02
9	2.37	2.77	3.12	0.13	2.79	2.59	0.07	0.33	0.47	0.43	1.95	1.80	0.08
10	2.43	2.77	3.27	0.18	2.79	2.63	0.06	0.33	0.47	0.41	1.95	1.82	0.07
11	2.60	2.94	3.69	0.25	2.86	2.74	0.04	0.31	0.45	0.45	2.02	1.87	0.08
12	2.77	3.11	4.10	0.32	2.92	2.85	0.02	0.29	0.43	0.50	2.08	1.91	0.08
13	3.27	3.61	5.32	0.46	3.07	3.17	0.03	0.27	0.39	0.44	2.23	2.05	0.08
14	3.30	4.54	5.40	0.19	3.36	3.19	0.05	0.25	0.39	0.55	2.24	2.06	0.08
15	3.37	4.64	5.57	0.20	3.39	3.23	0.05	0.23	0.38	0.65	2.26	2.07	0.08
16	3.40	6.41	5.64	0.12	3.91	3.25	0.17	0.22	0.38	0.72	2.27	2.08	0.08
17	4.10	7.14	7.35	0.03	4.08	3.69	0.09	0.21	0.31	0.50	2.45	2.27	0.07
18	4.17	7.94	7.53	0.05	4.28	3.74	0.13	0.21	0.31	0.47	2.47	2.29	0.07
19	6.23	10.87	12.56	0.16	4.75	5.05	0.06	0.20	0.12	0.39	2.80	2.85	0.02
20	7.17	13.94	14.86	0.07	5.17	5.65	0.09	0.18	0.04	0.79	2.93	3.11	0.06
21	7.27	17.04	15.11	0.11	5.60	5.71	0.02	0.17	0.03	0.83	2.94	3.13	0.07
22	7.37	17.14	15.35	0.10	5.61	5.78	0.03	0.16	0.02	0.88	2.96	3.16	0.07
23				MRE			MRE=			MRE=			MRE=
24				0.25			0.14			0.48			0.10

9. The MRE for group I:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	time	waste	w	MRE	prog	p	MRE	stab	s	MRE	effort	e	MRE
2	1.23	0.00	-2.20		1.00	0.87	0.13	1.00	0.82	0.18	1.00	1.21	0.21
3	1.27	0.00	-2.08		1.00	0.90	0.10	1.00	0.82	0.20	1.03	1.22	0.19
4	1.40	0.13	-1.72	14.20	1.09	1.02	0.07	0.95	0.80	0.20	1.12	1.26	0.12
5	1.70	0.41	-0.87	3.11	1.37	1.27	0.07	0.76	0.76	0.11	1.30	1.33	0.03
6	2.37	0.41	1.03	1.51	1.37	1.85	0.35	0.76	0.67	0.18	1.58	1.51	0.05
7	2.77	0.94	2.16	1.30	1.90	2.19	0.15	0.53	0.62	0.05	1.73	1.61	0.07
8	3.27	2.81	3.58	0.27	2.47	2.62	0.06	0.45	0.55	0.22	1.88	1.74	0.08
9	3.33	3.31	3.75	0.13	2.98	2.67	0.10	0.36	0.55	0.48	1.90	1.75	0.08
10	3.43	3.31	4.03	0.22	2.98	2.76	0.07	0.28	0.53	0.58	1.93	1.78	0.08
11	4.13	3.51	6.01	0.71	3.87	3.36	0.13	0.21	0.44	0.41	2.12	1.96	0.08
12	5.23	3.51	9.13	1.60	3.87	4.30	0.11	0.21	0.30	0.07	2.33	2.24	0.04
13	6.00	4.28	11.31	1.64	4.00	4.96	0.24	0.19	0.20	0.01	2.46	2.44	0.01
14	6.03	9.74	11.39	0.17	4.91	4.99	0.02	0.18	0.19	0.08	2.47	2.44	0.01
15	6.23	11.84	11.96	0.01	5.24	5.16	0.02	0.18	0.16	0.11	2.50	2.50	0.00
16	6.27	13.98	12.07	0.14	5.58	5.19	0.07	0.17	0.16	0.81	2.50	2.51	0.00
17	7.23	14.98	14.79	0.01	5.72	6.02	0.05	0.16	0.03	0.83	2.64	2.75	0.04
18	7.27	20.36	14.90	0.27	6.46	6.05	0.06	0.14	0.03	0.83	2.64	2.76	0.05
19	7.30	20.36	14.99	0.26	6.46	6.08	0.06	0.14	0.02	1.00	2.64	2.77	0.05
20				MRE=			MRE=			MRE=			MRE=
21				1.60			0.10			0.35			0.07

10. The MRE for group J:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	time	waste	w	MRE	prog	p	MRE	stab	s	MRE	effort	e	MRE
2	0.30	0.00	-2.78		1.00	1.11	0.11	1.00	0.64	0.36	1.00	1.57	0.57
3	1.20	0.90	-0.62	1.69	1.75	1.69	0.03	0.63	0.55	0.13	1.75	1.86	0.06
4	1.23	0.90	-0.55	1.61	1.75	1.71	0.02	0.63	0.55	0.13	1.77	1.86	0.05
5	1.43	1.13	-0.06	1.06	1.91	1.84	0.04	0.56	0.53	0.06	1.94	1.93	0.01
6	1.50	1.43	0.10	0.93	2.11	1.89	0.11	0.48	0.52	0.09	1.98	1.95	0.01
7	1.67	1.67	0.51	0.69	2.25	2.00	0.11	0.42	0.50	0.20	2.08	2.01	0.04
8	1.73	1.73	0.66	0.62	2.29	2.04	0.11	0.38	0.50	0.31	2.12	2.03	0.04
9	2.57	1.73	2.68	0.55	2.29	2.58	0.13	0.33	0.42	0.26	2.45	2.30	0.06
10	2.70	1.86	2.99	0.61	2.34	2.66	0.14	0.31	0.40	0.30	2.49	2.34	0.06
11	2.80	1.86	3.23	0.74	2.34	2.73	0.17	0.31	0.39	0.27	2.53	2.37	0.06
12	3.30	2.46	4.43	0.80	2.52	3.05	0.21	0.26	0.34	0.32	2.68	2.53	0.06
13	3.43	3.09	4.74	0.54	3.35	3.13	0.06	0.21	0.33	0.57	2.72	2.57	0.05
14	4.20	3.99	6.60	0.65	3.75	3.63	0.03	0.19	0.25	0.33	2.90	2.82	0.03
15	4.50	5.19	7.32	0.41	4.02	3.83	0.05	0.18	0.22	0.24	2.97	2.92	0.02
16	4.63	5.62	7.63	0.36	4.11	3.91	0.05	0.17	0.21	0.24	3.06	2.96	0.03
17	4.67	6.09	7.73	0.27	4.21	3.94	0.07	0.16	0.21	0.29	3.07	2.97	0.03
18	5.23	6.09	9.07	0.49	4.21	4.30	0.02	0.16	0.15	0.06	3.18	3.15	0.01
19	6.03	6.89	11.00	0.60	4.34	4.82	0.11	0.16	0.07	0.55	3.31	3.41	0.03
20	6.23	11.62	11.48	0.01	5.10	4.94	0.03	0.14	0.05	0.63	3.34	3.47	0.04
21	7.27	25.88	13.98	0.46	5.77	5.62	0.03	0.11	-0.05	1.47	3.48	3.81	0.09
22				MRE=			MRE=			MRE=			MRE=
23				0.69			0.08			0.34			0.07

11. The MRE for group K:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	time	waste	w	MRE	prog	p	MRE	stab	s	MRE	effort	e	MRE
2	0.80	0.00	-2.23		1.00	1.01	0.01	1.00	0.69	0.31	1.00	1.28	0.28
3	1.23	0.43	-0.44	2.03	1.35	1.52	0.12	0.63	0.63	0.25	1.35	1.44	0.07
4	1.33	0.43	-0.03	1.06	1.35	1.64	0.21	0.83	0.61	0.26	1.42	1.48	0.04
5	1.40	0.60	0.26	0.56	1.59	1.72	0.08	0.56	0.60	0.07	1.47	1.50	0.02
6	1.67	1.14	1.38	0.21	1.92	2.04	0.06	0.42	0.56	0.33	1.64	1.60	0.02
7	2.00	1.14	2.75	1.41	1.92	2.44	0.27	0.42	0.51	0.21	1.80	1.72	0.04
8	2.23	3.99	3.71	0.07	3.19	2.71	0.15	0.28	0.47	0.69	1.90	1.80	0.05
9	2.47	4.23	4.70	0.11	3.29	3.00	0.09	0.26	0.44	0.69	2.00	1.89	0.06
10	2.80	4.80	6.07	0.27	3.49	3.39	0.03	0.25	0.39	0.55	2.12	2.01	0.05
11	3.27	6.78	6.02	0.18	4.10	3.95	0.04	0.21	0.32	0.51	2.26	2.18	0.04
12	3.50	9.31	8.98	0.04	4.82	4.22	0.12	0.18	0.28	0.57	2.33	2.26	0.03
13	4.10	9.91	11.47	0.16	4.97	4.94	0.01	0.17	0.19	0.13	2.50	2.48	0.01
14	4.33	11.80	12.42	0.05	5.26	5.21	0.01	0.16	0.16	0.02	2.69	2.56	0.05
15	6.27	22.99	20.47	0.11	7.04	7.52	0.07	0.14	-0.14	1.97	3.00	3.27	0.09
16				MRE=			MRE=			MRE=			MRE=
17				0.48			0.09			0.47			0.06

12. The MRE for group L:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	time	waste	w	MRE	prog	p	MRE	stab	s	MRE	effort	e	MRE
2	1.07	0.00	-0.81		1.00	1.01	0.01	1.00	0.84	0.16	1.00	1.21	0.21
3	1.70	0.63	0.05	0.92	1.37	1.26	0.08	1.37	0.76	0.45	1.37	1.41	0.03
4	2.00	0.93	0.46	0.51	1.52	1.37	0.10	1.52	0.72	0.53	1.52	1.51	0.01
5	2.63	0.93	1.31	0.41	1.52	1.62	0.06	1.76	0.63	0.64	1.76	1.72	0.02
6	2.80	1.10	1.54	0.40	1.58	1.68	0.06	1.82	0.61	0.67	1.82	1.77	0.03
7	3.33	1.10	2.26	1.06	1.58	1.89	0.19	1.98	0.53	0.73	1.98	1.94	0.02
8	3.50	1.97	2.49	0.27	2.03	1.95	0.04	2.18	0.51	0.77	2.18	2.00	0.08
9	4.23	3.43	3.49	0.02	2.38	2.23	0.06	2.35	0.41	0.83	2.35	2.24	0.05
10	4.47	3.67	3.81	0.04	2.43	2.33	0.04	2.41	0.38	0.84	2.41	2.32	0.04
11	5.23	3.67	4.85	0.32	2.43	2.62	0.08	2.55	0.27	0.89	2.55	2.56	0.01
12	5.93	6.10	5.80	0.05	2.84	2.89	0.02	2.67	0.18	0.93	2.67	2.79	0.05
13	6.03	7.66	5.93	0.23	3.10	2.93	0.06	2.69	0.16	0.94	2.69	2.82	0.05
14				MRE=			MRE=			MRE=			MRE=
15				0.35			0.07			0.70			0.05

13. The MRE for group M:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	time	waste	w	MRE	prog	p	MRE	stab	s	MRE	effort	e	MRE
2	0.93	0.00	-1.32		1.00	1.25	0.25	1.00	0.62	0.38	1.00	1.32	0.32
3	1.07	0.27	-0.97	4.58	1.26	1.34	0.07	0.72	0.61	0.15	1.13	1.36	0.20
4	1.30	0.50	-0.39	1.77	1.44	1.50	0.04	0.64	0.59	0.07	1.31	1.42	0.09
5	1.47	0.67	0.04	0.94	1.55	1.62	0.04	0.55	0.58	0.05	1.42	1.47	0.03
6	1.77	1.14	0.80	0.30	1.82	1.82	0.00	0.53	0.55	0.04	1.57	1.55	0.01
7	2.00	1.37	1.37	0.00	1.93	1.98	0.03	0.43	0.53	0.24	1.69	1.62	0.04
8	2.23	2.13	1.95	0.08	2.28	2.14	0.06	0.39	0.51	0.32	1.79	1.68	0.06
9	2.80	2.13	3.39	0.59	2.28	2.53	0.11	0.39	0.47	0.20	2.00	1.84	0.08
10	3.30	3.20	4.65	0.45	2.60	2.87	0.10	0.35	0.42	0.21	2.15	1.98	0.08
11	3.50	5.17	5.15	0.00	3.59	3.00	0.16	0.29	0.41	0.40	2.20	2.04	0.07
12	3.57	5.17	5.33	0.03	3.59	3.05	0.15	0.29	0.40	0.38	2.21	2.06	0.07
13	4.23	5.90	6.99	0.18	3.76	3.50	0.07	0.27	0.35	0.28	2.39	2.24	0.06
14	4.57	7.31	7.84	0.07	4.07	3.73	0.08	0.24	0.32	0.32	2.46	2.34	0.05
15	5.23	7.31	9.51	0.30	4.07	4.18	0.03	0.24	0.26	0.09	2.59	2.52	0.03
16	6.03	8.11	11.52	0.42	4.21	4.73	0.12	0.23	0.19	0.16	2.72	2.75	0.01
17	6.33	9.87	12.28	0.24	4.48	4.94	0.10	0.22	0.17	0.23	2.77	2.83	0.02
18	7.17	15.27	14.39	0.06	5.24	5.51	0.05	0.21	0.10	0.53	2.89	3.07	0.06
19	7.30	21.73	14.72	0.32	6.12	5.60	0.09	0.19	0.09	0.54	2.90	3.10	0.07
20				MRE=			MRE=			MRE=			MRE=
21				0.61			0.09			0.26			0.08

APPENDIX E: Definitions

Definition 1: R is an **equivalence relation** on A iff R is a binary relation on A that is **reflexive** on A , **symmetric**, and **transitive**.

1. R is **reflexive** on A , by which we mean that xRx for all $x \in A$.
2. R is **symmetric** by which we mean that whenever xRy , then also yRx .
3. R is **transitive** by which we mean that whenever xRy and yRz , then also xRz .

Definition 2: A **partial ordering** is a relation R meeting the following two conditions:

1. R is a **transitive** relation: $xRy \ \& \ yRz \rightarrow xRz$.
2. R is **irreflexive**: It is never the case that xRx .

Definition 3: A **preordering** is a relation R meeting the following two conditions:

1. R is **reflexive** on A , by which we mean that xRx for all $x \in A$.
2. R is **transitive** by which we mean that whenever xRy and yRz , then also xRz .

Definition 4: Let A be any set. A **linear ordering** on A (also called a **total ordering** on A) is **binary relation** R on A meeting the following two conditions:

1. R is a **transitive** relation ; i.e., whenever xRy and yRz , then xRz .
2. R satisfies **trichotomy** on A , by which we mean that for any x and y in A exactly one of the three alternatives xRy , $x = y$, yRx holds.

Definition 5: A **partition** π of a set A is a set of nonempty subsets of A that is **disjoint** and **exhaustive**, i.e.,

1. no two different sets in π have any common elements, and
2. each element of A is in some set in π .

REFERENCES

- [BAK,1987]Baker, A., Bieman, J., Gustafson, D. and Melton, A. "Modeling and Measuring The Software Development Process", Proc. HICCS,1987.
- [BAR,1982]Barr, A. Cohen, P. and Feigenbaum, E. A., The Handbook of Artificial Intelligence, William Kaufman, Los Altos, California, 1982-1983.
- [BLU,1982]Blum, B. I., "The Life Cycle-A Debate Over Alternate Models", ACM SIGSOFT, Software Engineering Notes, Vol. 7, No. 4, 18-20, October 1982.
- [BLU,1984]Blum, B. I., "Three Paradigm for Developing Information Systems", IEEE, 534-543, 1984.
- [BOE,1981]Boehm, Barry W., Software Engineering Economics, Englewood Cliffs, N. J.: Prentice-Hall, 1981.
- [BRO,1975]Brooks, F., The Mythical Man-Month, Addison-Wesley, Reading, Mass., 1975.
- [CHE,1976]Chen, P., "The Entity-Relationship Model: Towards a Unified View of Data", Transactions on Data Base System, 1(1):9-36, 1976.
- [CHE,1980]Chen, P., Entity-Relationship Approach to System Analysis and Design, Elsevier/North-Holland, New York, 1980.
- [CHE,1983]Chen, P., Entity-Relationship Approach to Informatin Modeling and Analysis, Elsevier/North-Holland, New York, 1983.
- [DAV,1979]Davis, A. M. and Rauscher, T. G., "Formal Techniques and Automatic Processing to Ensure Correctness in Requirements Specification, In Specification of Reliable Software, 15-35, IEEE, 1979.
- [DAV,1983]Davis, C. Jajodia, S. Ng, P. Yeh, R., Entity-Relationship Approach to Software Engineering North-Holland, 1983.

- [DeM,1978]DeMarco, T. Structure Analysis and System Specification. New York: Yourdon Press, 1978.
- [DeM,1982]DeMarco, T., "Controlling Software Projects : Management, Measurement and Estimation, Yourdon Press, New York, 1982.
- [DIJ,1968]Dijkstra, E. W. "GOTO Statement Considered Harmful.", Communications of the ACM 11, No. 3, 147-148, 1968.
- [END,1977]Enderton, H. B., Elements of Set Theory, Academic Press, 1977.
- [FAI,1985]Fairley, R. E., "Software Engineering Concepts, McGraw-Hill Book Company, 1985.
- [GEN,1986]General Electric Company Corporate Information System, Software Engineering Handbook, McGRAW-Hill Book Company, New York, 1986.
- [GOL,1986]Goldberg, R., "Software Engineering: An Emerging Discipline," IBM System Journal, Vol. 25, Nos. 3,4, 334-353, 1986.
- [HAL,1977]Halstead, M. H. Elements of Software Science, New York: Elsevier North Holland, 1977.
- [HEN,1980]Heninger, K. L., Kallander, J. W. and Parnas D. L., Software Requirements for the A-7E Aircraft, Technical Report, Naval Research Laboratories.
- [HER,1986]Herbert, A. S., "Whether Software Engineering Needs to Be Artificially Intelligent", IEEE Transactions on Software Engineering, Vol. Se-12, No. 7, 726-732, July 1986.
- [KAT,1985]Katke, W., "On "Learning Language"", The AI Magazine, 24-51, Fall 1985.
- [LUB,1986]Lubers, M. D. and Horandi, M. T., "Intelligent Support for Software Specification and Design", IEEE Expert, 33-41, Winter 1986.

- [McC,1976]McCabe, T. J. "A Complexity Measure." IEEE Transaction on Software Engineering, Vol. Se-2, No. 12,308-320, December 1976.
- [McC,1982]McCracken, D. D. and Jackson M. A., "Life Cycle Concept Considered Harmful". SEN(7,2), 35-39, April 1982.
- [MOS,1985]Mostow, J., "What is AI? And What Does It Have to Do with Software Engineering",IEEE Transactions on Software Engineering, Vol. SE-11, No. 11, 1253-1256, November 1985.
- [NEL,1966]Nelsen, E. A., "Management Handbook for the Estimation of Computer Programming Costs", System Development Corporation, NTIS, No. AD-A648750, 1966.
- [PUT,1978]Putnam, L. H., "A General Empirical Solution to the Macro Software Sizing and Estimating Problem.", IEEE Transactions on Software Engineering SE-4,4, 345-361, July 1978.
- [PUT,1980]Putnam, L. H.,et. Software Cost Estimating and Life Cycle Control. New York: IEEE Computer Society Press, 1980.
- [SAH,1982]Sahni, S., Concepts in Discrete Mathematics, The Camelot Publishing Company, Fridley, Minnesota.
- [TER,1986]Terwilliger, R. B. and Campbell, R. H., "PLEASE: Predicate Logic based Executable Specifications", ACM, 349-358, 1986.
- [YOV,1985]Yovits, Marshall C., Advances in Computers. Academic Press, INC, 1985.

THE ANALYSIS OF THE SOFTWARE PROCESS MODEL

by

YING-CHI CHEN

B.S., Eastern Michigan University, 1982

M.S., Eastern Michigan University, 1983

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1988

The nature of software engineering has been that the development of most methodologies has come from experiences rather than by using mathematical derivations. It is impossible to attain the status of scientific discipline unless software engineering is built upon a sound foundation of measurement like other engineering areas. This paper attempts to find a way to develop measures by using scientific methods. These measures are developed using the Software Process Model (SPM).

In this paper, the SPM is described. The measure development paradigm is presented. Measures of waste, effort, progress and stability are developed using the paradigm. The measures are applied to data from CS540-CS541 projects (Spring 1988). The measures for waste, progress, stability and progress are evaluated.