MODELING AND ANALYSIS OF TELEMENTAL HEALTH SYSTEMS WITH PETRI NETS

by

RYAN AESCHLIMAN

B.S. Industrial & Manufacturing Systems Engineering, Kansas State University, 2015

A THESIS

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2015

Approved by:

Major Professor
Dr. David Ben-Arieh

Copyright

Abstract

Telemental health systems, a form of telemedicine, use electronic communication media to provide patients in remote locations access to psychological and psychiatric specialists. The structure of telemental health systems has a major impact on their performance. Discrete-event simulations offer useful results concerning capacities and utilization of specific resources. Simulation, however, cannot provide theoretical properties of analyzed systems. Petri net representations of systems can overcome this shortfall, offering a wide range of easily-analyzed and useful properties. Their ability to model resource conflict, parallel activities, and failure modes fits nicely with the reality of telemental health systems. Analysis of behavioral properties of Petri nets can provide meaningful information for system analysts. The most useful properties include net boundedness, liveness, and non-reachability of certain undesirable states. The thesis discusses methods to find all these properties. Specifically, it provides property-preserving net reductions to facilitate analysis of boundedness and liveness and describes an integer programming model to solve reachability and coverability problems.

Moreover, this thesis outlines a simulation analysis of synchronous and asynchronous telemental health systems. The paper then describes a Petri net model of a generic telemental health delivery system. The paper subjects the model to an integer programming model and net reduction. The integer programming model indicated that the number of resources in the system remains static, full utilization of resources at a given time is possible, conflict over resources is possible, and improper work prioritization is possible within the model. Net reduction and analysis with open-source software showed that the model is bounded and live. These results can aid telemedicine system architects in diagnosing potential process issues. Additionally, the methods described in the paper provide an excellent tool for further, more granular analysis of telemedicine systems.

Table of Contents

List of Figures

List of Tables

List of Symbols and Abbreviations

Symbol	Subject	Interpretation
GP	Telemedicine	General Practitioner
BIP	Mathematical Programming	Binary Integer Programming
Expo(10)	Simulation	An Exponential Distribution with a Mean of 10 Minutes
PN	Petri Net Definition	Petri Net
P	Petri Net Definition	Set of Places in a Petri Net
T	Petri Net Definition	Set of Transitions in a Petri Net
m	Petri Net Definition	Number of Places in a Petri Net
n	Petri Net Definition	Number of Transitions in a Petri Net
F	Petri Net Definition	Set of All Arcs in a Petri Net
M_0	Petri Net Definition	Initial Marking
A	PN Incidence Matrix	Petri Net Incidence Matrix
a_{ij}	PN Incidence Matrix	Entries in A
A^-	PN Incidence Matrix	Input Incidence Matrix
a_{ij}^-	PN Incidence Matrix	Entries in A^-
r	PN Incidence Matrix	Rank of A
A_{11}, A_{12}, A_{21}, A_{22}	PN Incidence Matrix	Subsets of the A Matrix Broken Down by Rank
u_k	PN State Equation	Step k Control Vector. Index j May Also Be Used
d	PN State Equation	Total Number of Steps in an Analysis
M_d	PN State Equation	Marking After d Steps
ω	PN Coverability Graph	Infinite Marking in Reachability Graph
M_f	Reachability BIP Model	Target Marking
X	Reachability BIP Model	Sum of Some Number of Control Vectors
K	Reachability BIP Model	Reachability Search Depth in Steps
B_f	PN Reachability Analysis	A Set of Linearly Independent Solutions y to the Equation $Ay = 0$
z	PN Reachability Analysis	Set of Integers Used to Form a Linear Combination of B_f Such That $B_f z$ Equals the Change in Tokens From the Original to the Target Marking

Chapter 1 - Introduction

Telemedicine, "the use of electronic information and communications technologies to provide and support clinical health care when distance separates the participants" (Institute of Medicine, 1996), offers powerful tools to deliver care to underserved populations and allow doctors to make the best use of their time. Generally, telemedicine systems involve a specialized provider at a "hub" site using some form of electronic media to deliver care to a patient at a remote "spoke" site.

While telemedicine has found applications in dermatology, optometry, and even surgery, mental health services such as psychology and psychiatry currently stand as a field with some of the greatest potential for effective telemedicine (Locatis & Ackerman, 2013). Applying telemedicine to psychiatry or psychology results in a practice known as telemental health. One standard defines telemental health simply as "the practice of mental health specialties at a distance," aiming to capture as wide a variety of media as possible (Grady et al., 2011).

Despite its strengths, telemedicine has several areas of weakness. Clinical effectiveness concerns and cries for larger sample sizes lie firmly within the realm of healthcare, but system-related concerns can be at least partially resolved with the application of effective models. Chapter two provides a detailed discussion of telemedicine.

Discrete event simulation can provide a valuable tool for examining specific systems. Good simulation models can provide valuable insight regarding system capacity and employee utilization. Simulations excel at showing how systems *should* behave on average, but do not do so well at diagnosing potential failure modes and examining how systems *can* behave given enough time.

Petri nets, a modeling tool with roots in computer science and graph theory, can provide a solution to these shortcomings. Chapter three discusses Petri nets in depth. A diverse set of Petri net properties can model a host of failure modes, ranging from deadlocks to resource scarcity to unbounded queue growth to any sort of generally undesirable system state. Integer programming models can indicate if specific failure modes are possible given an initial system configuration (Bourdeaud'huy, Hanafi, & Yim, 2007), while a combination of Petri net reduction methods and analysis with software can root out deadlocks and unbounded growth.

Telemental health systems lend themselves well to representation by Petri nets. Chapter four presents a Petri net model for a generic telemedicine system. Chapter five applies the analytic techniques from chapter three to this net and reveals the results of the analysis. Generally, telemental health systems that follow the general format laid out in chapter four will not suffer from deadlocks or unbounded growth. Moreover, resources levels will remain static and the system will not clear itself out prematurely. Telemedicine systems can potentially struggle with conflict over employee time, inequitable treatment prioritization, and full employee utilization.

Used together, the methods mentioned in this thesis offer a powerful suite of analytic tools to gauge the effectiveness and stability of telemental health systems. To appreciate their usefulness, however, one must begin with the motivation behind this mode of analysis. The next chapter presents this rationale and lays out the groundwork for modeling telemental health systems with Petri nets.

Chapter 2 - The Challenges of Telemedicine

Telemedicine can manifest itself as both a powerful tool and an administrative quagmire. Consequently, it has been the subject of much research in recent years. Numerous papers probe the pros, cons, and general challenges of the technique, while others lay out promising system configurations. This chapter discusses this body of work and presents a standard set of components for telemedicine systems.

2.1 – Telemedicine Basics

Telemedicine in the broadest sense has a surprisingly long history and can be traced all the way back to the advent of telephone lines in the 1870's, although its current manifestation did not appear until the 1960's (Sang Goo, Mun, Jha, Levine, & Ro, 2000).

Distance stands as a major barrier between mental health care providers and potential patients (Rabinowitz, Brennan, Chumbler, Kobb, & Yellowlees, 2008). Potential patients living in rural or traditionally underserved communities where care providers feel they cannot cost-effectively distribute their time stand at the greatest loss. Telemedicine can shrink this distance and provide treatment to those in far-flung locations. For example, a system for proving expert consultations via radio for Health Aids in remote parts of Alaska has had great success (Sang Goo et al., 2000). Similarly, a project in rural Kansas to directly provide child psychological services via telemedicine helped doctors meet a significant need (Spaulding, 2010).

Moreover, telemental health may be superior to face-to-face care in cases where the patient suffers from paranoia, PTSD, or another condition where "distance" between the patient and doctor helps the patient open up and cope. In-home telemental health can let providers examine patient's living environment firsthand and potentially lead to better diagnoses. Finally,

telemedicine allows providers to examine facial expressions in more detail than face-to-face conversations allow (Rabinowitz et al., 2008).

Proponents of telemedicine frequently cite its cost-effectiveness. One analysis used a queuing network model to conclude that, generally, telemedicine makes financial sense when it is effective and care providers are relatively expensive to utilize. It makes less sense when administrative staff is relatively expensive, conventional systems work well, or the financial risk of mistreating patients is very large. In all cases, telemedicine should be a supplement to face-to-face healthcare services – no organization should serve all of its patients with telemedicine (Tarakci, Sharafali, & Ozdemir, 2007).

2.1.1 – Areas of Weakness for Telemedicine

Telemedicine research lacks in some areas. Some authors contend larger samples sizes and more diverse populations will yield more telling feasibility studies (Rabinowitz et al., 2008). Others argue that telemedicine needs to focus more research on the clinical outcomes of telemedicine as well as its economic justification (Krupinski et al., 2006).

Rabinowitz (Rabinowitz et al., 2008) argues that more research must be done on the impact of telemedicine in rural areas, particularly in terms of patient satisfaction. In particular, patient satisfaction measurement requires better methodologies before it can be confidently used in decisions. While studies have increasingly attempted to measure patient satisfaction, results may have an upward bias due to methods commonly used (Zhang, McClean, Jackson, Nugent, & Cleland, 2013). Notably, many studies include an "after" survey without a corresponding "before" survey to obtain baseline satisfaction.

The organizational structure of telemedicine systems acts as a deterrent to providers. Lasierra et. al. offers a particularly telling case study of an asynchronous teledermatology system

that ultimately failed. They note that successful programs need support from organizational leadership and buy-in from providers. Additionally, the program used actually took more person-hours to resolve each case. Lack of integration with the hospital's healthcare information system and a need for some providers to work outside of normal hours to perform consultations made the program unpopular (Lasierra, Alesanco, Gilaberte, Magallón, & García, 2012).

Accurate diagnosis of patients can be difficult with telemedicine. While the ability to review video data and potentially gather more patient information can help in some cases, the lack of direct eye contact inherent in many video messaging systems and other problems require an unfamiliar change of tactics in diagnosis (Grady et al., 2011). Grady goes on to discuss how video quality can positively impact this. While beneficial, better-quality equipment also adversely affects the already high startup cost of telemedicine systems.

Telemedicine hinges on patient involvement to find success, but often that involvement gets stymied by lack of patient knowledge regarding telemedicine. Literature and the author's experience indicates that, no matter how well-engineered a telemedicine system appears to be, it cannot be successful without an engaged and informed patient base. This "alignment with learning processes" extends to both the patients and providers; organizations must be dynamic and quickly adopt potential improvements in the rapidly developing world of telemedicine (Cegarra-Navarro, Sanchez, & Cegarra, 2012).

One concern telemedicine prompts involves the care provider's ability to generate rapport with patients. Relationship building allows for much more open and effective treatment and patient/provider interaction (Grady et al., 2011); consequently, designers of telemedicine systems should build the system with effective interactions in mind. The authors of Grady's work note

some challenges with this as well – in particular, the expanded geographic of providers' range of care may result in providers interacting with patients beyond their current "cultural competency."

2.1.2 – Structure of Telemedicine

The most intuitive way to deliver telemedicine involves simply setting up a video camera or phone call with the provider on one end and the patient on the other. The provider resides at a centralized "hub site" – often a large regional hospital, while the patient connects at a "spoke site." Spokes can be smaller hospitals, local clinics, or even the patient's home. A telemedicine system will usually have numerous spoke sites.

This simple configuration requires the patient and provider to be available at the same time, earning it the designation of "synchronous" telemedicine. While traditional, synchronous telemedicine aims to remotely provide services that fundamentally look like traditional healthcare, asynchronous, or "store-and-forward," telemedicine offers an entirely new paradigm for healthcare delivery. In asynchronous telemedicine, patients and providers do not communicate in real time. Rather, the patient gathers or generates health data, often with the assistance of a nurse or hospital administrator, and sends it electronically to the specialist. The specialist then reviews the data at their convenience, prepares recommendations, and sends them back to the patient (Yellowlees et al., 2011).

Store-and-forward telemedicine intuitively makes sense for specialties that function like this anyway, such as radiology, or specialties where the data needed for a diagnosis can be effectively summarized and conveyed digitally, such as dermatology (Yellowlees et al., 2011).

Researchers at the University of California-Davis developed an asynchronous procedure for telemental health services in 2011. The process involves patients recording a half-hour interview with a nurse asking questions prepared by a specialist. The specialist then reviews the

interview and makes treatment recommendations. While the process may take longer overall to complete, it makes better use of specialists' time and provides them with flexibility. This technique also allows for better record-keeping since entire interviews already have electronic recordings ready (Yellowlees et al., 2011).

Asynchronous telemedicine represents a significant departure from the stakeholder responsibilities found in more "traditional" telemedicine. Administrators have different tasks, patients take on more responsibility, and specialists spend more of their time doing things only specialists can do (Yellowlees et al., 2011).

Store-and forward telemental health can offer a cost savings over face-to-face healthcare as well. Psychiatric visits have three major components: data collection, data analysis, and treatment planning (Butler & Yellowlees, 2012). Traditionally, all three phases are the burden of the psychiatrist. However, data collection and much of the accompanying clerical work can be executed adequately well by a nurse or staff member, allowing better utilization of providers' valuable time.

A massive literature review from Laurant et al. indicates that, given the very limited amount of available data, nurses can provide similar quality care to what doctors can perform, with similar health outcomes (Laurant et al., 2005). While the authors note that cost advantages in salary are offset by the greater number of tests and services nurses utilize, the limited number of tests inherent in mental healthcare mitigates this drawback. Ultimate cost savings depend highly on context, however. Moreover, some patients actually experience higher patient satisfaction when served by nurses if they consider their cases to be fairly minor and the nurses are able to spend more time than doctors performing consultations.

Interestingly, simply having nurses fill some roles traditionally filled by doctors in healthcare systems does not guarantee a decrease in the workload of doctors. Laurant et al.'s research indicates that doctors may hold on to the delegated responsibilities and spend time doing tasks more efficiently done by nurses. The authors recommend systems include "active steps" to ensure doctors spend their time doing things only doctors can do (Laurant et al., 2005).

A 2012 study by Butler and Yellowlees in California found that, while store-and-forward systems have a higher variable cost, synchronous systems have a higher fixed cost. This results in a breakeven point in larger systems serving approximately 250 patients (Butler & Yellowlees, 2012). Consequently, larger operations should consider synchronous telemedicine while smaller endeavors should turn an eye to asynchronous systems. This fixed cost savings comes partly from hardware requirements. One study found that good-quality store-and-forward care can be delivered with "pro-sumer," that is, expensive-hobbyist-quality, audiovisual equipment available for much less than the professional-grade hardware needed for a good synchronous system (Odor et al., 2011).

Asynchronous telemedicine works. A 2010 feasibility study in California provided clinical evidence that psychiatrists can comfortably diagnose axis I and II disorders with this model. In fact, the providers actually recommended mediation adjustments and long-term treatment plans for 95% of participants (Yellowlees et al., 2010). Despite the success of the model, Yellowlees recommends asynchronous telemedicine as a low-cost supplement to face-to-face meetings, rather than a replacement.

2.1.3 – Gauging the Effectiveness of Telemedicine

With telemedicine subject to such a high degree of scrutiny, accurate and simple metrics to determine the effectiveness of programs becomes absolutely essential. Locatis and Ackerman

suggest three “principles” to model the effectiveness of telemedicine: congruency, fidelity, and reliability. Specifically, they argue that telemedicine procedures should be fundamentally similar to face-to-face procedures, should gather the same types and quality of data as what could be gathered face-to-face, and should transfer the data to care providers as well as if it were face-to-face (Locatis & Ackerman, 2013).

Additionally, patient satisfaction can be measured with a few key metrics (Zhang et al., 2013). Specifically, a good patient-satisfaction survey analyzes how changes in telemedicine methodology impacts patient-provider relationships, pinpoints communication issues, and captures patient opinions regarding the failures and limitations of telemedicine service.

Software effectiveness is also a popular topic of research. One proposed metric for evaluating the effectiveness of software and system quality for store-and-forward telemedicine systems places “freedom from risk”, reliability, and security as the three absolutely essential qualities in any system (von Wangenheim, von Wengenheim, Hauck, McCaffery, & Buglione, 2012). Reliability in the context of telemedicine systems refers to the system’s ability to act as intended in a variety of circumstances and over a variety of time periods.

Literature appears lacking in studies that examine telemedicine from a traditional process engineering perspective. Doctor utilization and capacity in telemedicine systems, while occasionally mentioned, lack a large body of research behind it. Consequently, this area is ripe for research and modeling.

2.1.4 – Key Roles in Telemedicine Systems

Telemental health systems require people to function. Examinations of telemental health systems from both experience at the New Mexico Veteran’s Administration Hospital and literature (Laurant et al., 2005; Tarakci et al., 2007) reveal that telemedicine systems typically

require work from seven different roles to function. The list below denotes these roles and provides a brief description of each.

- **The Patient –** With the patient being the reason the entire system exists, most telemental health systems seek to serve as many patients as possible without compromising quality of care. Patients interact with the system at spoke sites.
- **The Specialist –** The specialist is a health provider with specialized knowledge. They work at the hub site. Telemedicine systems aim to help patients access specialists by eliminating geographical constraints between patient and provider and, in the case of asynchronous telemedicine, minimizing the administrative work specialists need to do and freeing up more of their time to serve patients. In the case of a telemental health system, the specialist is typically a psychologist or psychiatrist.
- **The Nurse –** This role can be performed by any spoke site staff member capable of responding to patient emergencies and generally interacting with and answering questions for the patient. While this role will most often be filled by a nurse, small spoke sites may use a general practitioner.
- **Hub Site Administrator –** This administrative assistant works at the hub site and is in charge of managing the entire telemedicine program, keeping track of patient data, enrolling patients, and preparing the specialist for upcoming appointments.
- **Spoke Site Administrator –** The spoke site administrator gets patients checked in and checked out of appointments. Generally this will be one of many duties (most unrelated to the telemedicine program) the person filling this role performs. That said, successful telemedicine implementation requires coordination at both the hub and spoke sites, making this role critical (Grady et al., 2011).

- **Hub Site Technology Specialist –** Larger telemedicine operations may have a person in charge of setting up audiovisual equipment for appointments. Smaller operations or operations with uncomplicated equipment may have the specialist fulfill this role.
- **Spoke Site Technology Specialist –** Spoke sites need someone on-hand to deal with equipment issues as they arise. This person may be a dedicated information technology coordinator with many duties around the site. Alternatively, the role could be filled by the nurse or a spoke site administrator.

These roles are not mutually exclusive – one person may handle administrative work, technology needs, and care provision. Similarly, these roles are scalable. Large systems may have multiple providers, administrators, and technology specialists working together.

2.1.5 – The Telemedicine Process

While most telemedicine systems feature fairly straightforward processes from a patient's perspective, the background processes and administrative overhead take a significant amount of effort and can be some of the key reasons why telemedicine systems fail. Consequently, good analysis hinges on a solid representation of the entire treatment cycle. Telemental health systems can be decomposed into four major phases: patient enrollment, session preparation, the session itself, and session follow-up. While the character of each of these phases can vary from system to system, the basic building blocks and tasks that must be completed are the same.

This telemedicine model arose from a generalization of the synchronous telemental health process the New Mexico VA Hospital uses. In their system, several psychiatrists, psychologists, and social workers do their jobs in the central VA Hospital in Albuquerque. Patients travel to Community-Based Outpatient Centers (CBOCs) in sites all across New

Mexico, as shown in the figure below. There they establish a connection with providers in Albuquerque, receive their care, and leave. The next few paragraphs summarize this process in more detail.

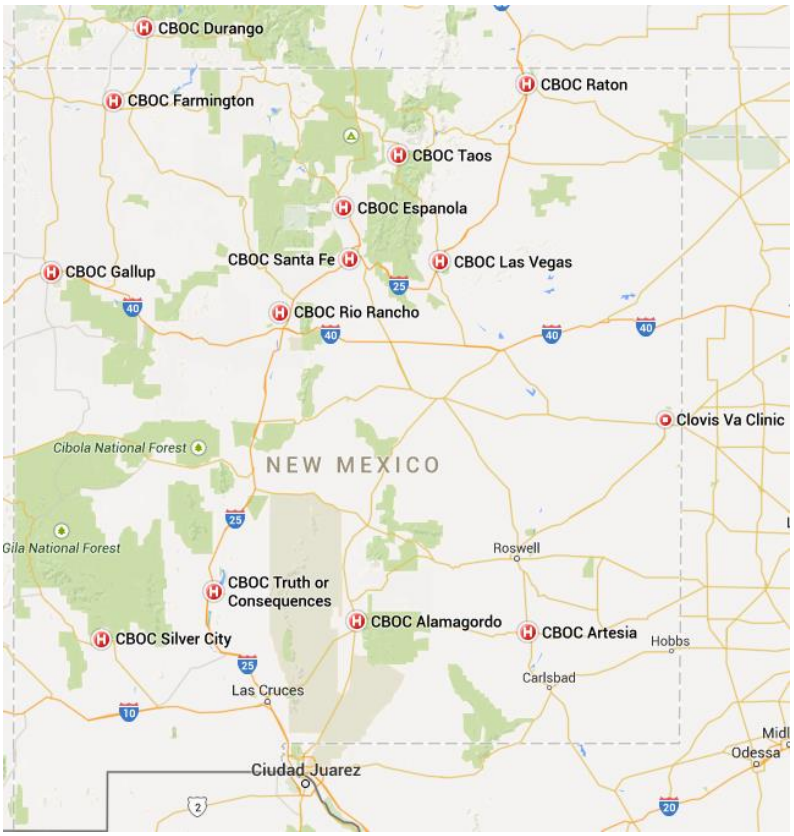

Figure 2.1- CBOC Locations in New Mexico and Southern Colorado

Image Source: (Google, 2015)

Systems begin with an enrollment phase. During this time, the specialist must process and review incoming referrals. Upon approval, administrators (typically at the hub site) will generate a record and schedule a series of appointments. Any preparatory work prior to the day of the appointment occurs during this phase. Enrollment is typically a transient phase; while other phases may be repeated, enrollment usually needs to occur just once.

Session preparation includes all routine tasks that need to be completed on the day of an appointment before it begins. The patient's arrival at the spoke site often triggers this series of processes. Initial paperwork at the spoke site and check-in procedures take up administrative time. Some of this information may be transmitted to the hub site. Rooms at the spoke and hub site need to be prepared and complex telemedicine systems may require technology specialists at the spoke and hub to establish a connection or set up audiovisual equipment. The specialist and hub administration also work to gather and review patient data prior to the appointment in order to provide effective care.

The session itself stands as the third stage in the telemedicine process. With everyone set up and prepared, the specialist at the hub site uses video equipment to deliver psychiatric or psychological care to a patient at the spoke site. A nurse or general practitioner will remain on-hand or sit in on the session at the spoke site to assist the patient as needed.

When the session completes, the process enters the fourth phase: session follow-up. The specialist begins by recommending treatment options, writing prescriptions, and doing paperwork as required. During this time, the nurse/general practitioner can answer any questions the patient may have and forward feedback to the specialist. The specialist sends all recommendations to the spoke site, often via the hub administrator. The patient then works with spoke administration to pick up their treatment, fill any prescriptions, and checkout. The process then returns to just before phase two, where the patient will check in for their next appointment and the cycle begins again. A visualization of this process can be found on figure 2.2 on page 16.

The synchronous model described above is not the only way to deliver telemental health. Some authors have proposed a store-and-forward methodology that can provide psychiatrists and

psychologists with more flexibility in performing their duties (Butler & Yellowlees, 2012)**.** Even so, this system still goes through the four main phases, as described below.

Enrollment in sessions remains exactly the same as in synchronous telemedicine. The specialist must approve any referrals and then administrators at the hub site take care of clerical work and scheduling.

Some time later, the session preparation phase begins. The hub administration gathers patient data and prompts the specialist to prepare for an upcoming therapy session. The specialist then uses this information to generate a set of questions for the patient to answer on camera. They pass these questions on to hub administrators who send this to their counterparts at the spoke site the patient will visit. This ends the first half of session preparation, which occurs exclusively at the hub site. The spoke picks up the workflow for the second half of preparation. The second half occurs the day of the session and may be a few days before the first half of preparation. The day begins by the patient checking in with spoke administration. Meanwhile, the spoke site technology specialist prepares the room for therapy. They do not need to establish a live connection; any input from the specialist comes prerecorded.

The session begins when ready. A nurse or general practitioner sits down with the patient and asks them the questions prepared by the specialist. A video camera records both parties. The specialist does not need to be available during this time – hence the asynchronous label. When the program prepared by the specialist comes to an end, nurse and patient leave the room, signaling the end of the session. The patient may check out at this time and go home.

The follow-up phase begins immediately after when spoke administrators send the recorded patient responses back to the hub site for examination by the specialist. At their

convenience, the specialist then looks over the recorded patient responses and develops a treatment plan. The specialist gives this plan to hub administrators who send it to the spoke site.

Depending on the nature of the treatment, the patient may need to return to the spoke site and check-in briefly with spoke administration. In some cases, they may meet with the nurse or general practitioner to receive more detailed instructions. With treatment in-hand, the patient checks out with the spoke specialist and exits the system. This marks the end of the treatment process until the specialist prepares the next round of questions for the patient's next appointment. Figure 2.3 on page 17 shows a flowchart of this asynchronous process.

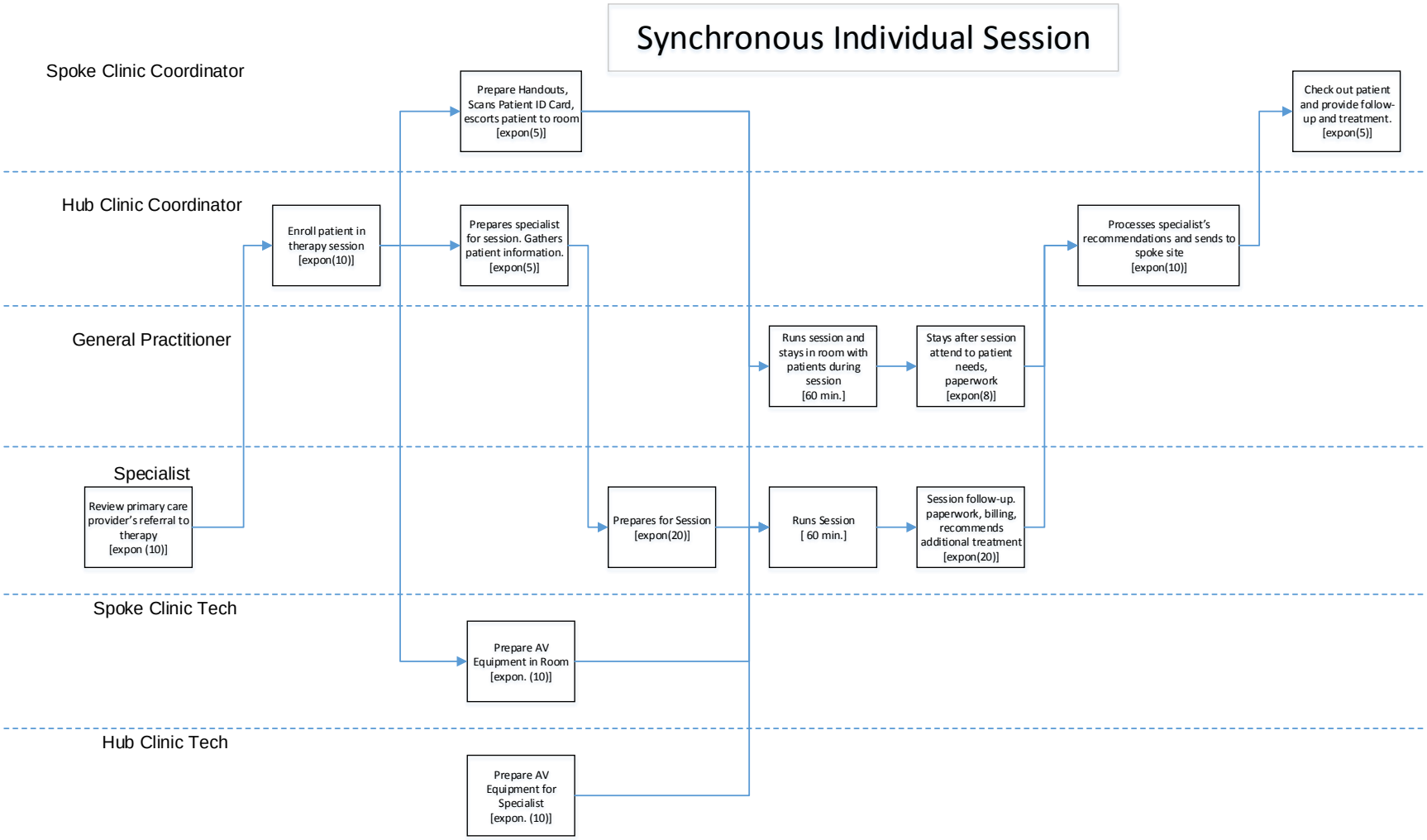

Figure 2.2-Structure of an Individual and Synchronous Telemental Health Session

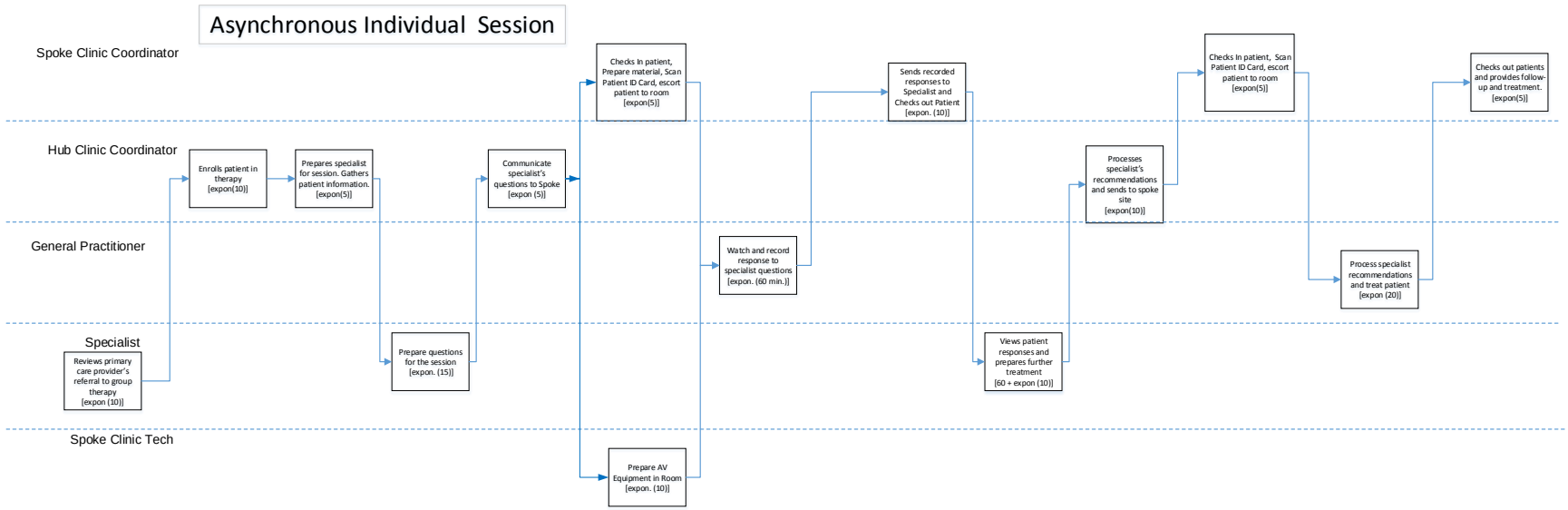

Figure 2.3– Structure of an Individual Asynchronous Telemental Health System

2.2 – Simulating Telemedicine Systems

Telemental health systems lend themselves well to discrete event simulation (Lach & Vazquez, 2004). Medical settings tend to offer the intrinsic structure and clearly defined responsibilities that make modeling them this way both simple and accurate. Moreover, variability can be easily included in the model to capture the unpredictability of treatment times and patient behavior.

Early work with the New Mexico VA called for baseline data and theoretical results easily obtainable with a simulation model. The simulation aims to measure utilization of the hardware and employees that make telemedicine systems work, as well as predict the maximum capacity of such a system given a set number of employees. The model works by tracking the flow of a theoretical object through the system. This object merely triggers events and is a "token" with no physical analog. If anything, the token signifies the flow of paperwork through the system. As the paperwork progresses, it summons employees, patients, and resources to aid in its forward march. The simulation tracks how long these resources receive use as well as how long the paperwork takes to get from referral to a treated patient.

The simulation requires service time distributions for each task. Due to geographic constraints, measured service times proved impractical to obtain. Instead, the average times presented in the following sections are estimates determined reasonable by the New Mexico VA. The simulation assumes exponential distributions as a safe choice for most service times due to their convenient properties and moderate levels of variability.

The next few sections look at simulations of this nature in depth and lay out exactly how to create a discrete-event simulation model for telemedicine systems.

2.2.1 – Modeling Synchronous Systems

Arena, the simulation software from Rockwell Software used to model this system, bases its logic off of four key "blocks" or actions: queue, seize, delay, and release. Paperwork enters a queue to wait for attention from a resource, seizes that resource when it becomes available, uses or delays that resource for a period of time, and releases that resource when finished. For example, the spoke site administrator could have a pile (queue) of clinical recommendations from a therapist to distribute one day. The administrator would deal with the top paper in the stack, work on it for a time, then move on to the next paper in the stack when complete. The paperwork then proceeds through many more queues until finally leaving the system at a dispose block. The simulation basically "sees" this process from the paperwork's point of view.

The model assumes that only one specialist, hub administrator, spoke administrator, spoke nurse or general practitioner, pair of technology specialists, and set of spoke telemedicine equipment can be occupied at any given time. These numbers can be tailored to each spoke site in application. For flexibility, the simulation tracks the room and equipment as a separate resource.

The most basic and intuitive telemedicine systems fall into the category of synchronous individual, where a single patient at a time meets with a provider in a remote location in real-time. The simulation creates one patient at a time at a specified rate and sees how the system performs over a day (480 minutes). This "day by day" model paints an accurate picture of steady state operations.

The model begins with a queue representing the referral review process, as shown in figures 2.4 and 2.5 on the next page. Since this process occurs independently of the actual care delivery, the simulation has a separate series of events modeling them. The model assumes that

for every four patients that have appointments on a given day, one patient needs to have their referral reviewed and enrollment performed. A separate "create" block generates these cases one at a time, according to an interarrival time that varies with the number of patients scheduled that day. In fact, the interarrival time on this block is precisely equal to the interarrival rate of regular patients times four. After arrival, the two queues proceed to their own dispose block for the "paperwork" to leave the system. The dispose block ends the process. The referral entities do not interact with the appointment execution side of the simulation except to use the specialist and hub administrator's time.

Figure 2.4- The first part of the independent referral process

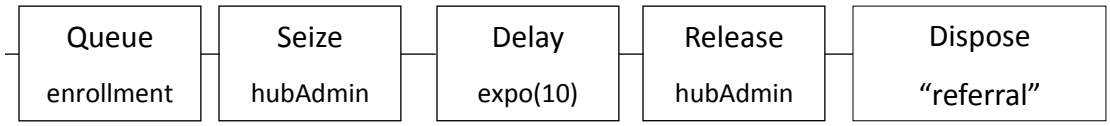

Figure 2.5– The second part of the independent referral process

While not connected to the system directly, the enrollment process still requires the time of the specialist and spoke administrator to complete. Neither operation happens instantly, so both delay for time following an exponential distribution with a mean of 10 minutes (henceforth abbreviated as expo(10) minutes) each.

For the primary treatment phase of the telemental health process, the simulation begins with a "create" block that generates patients one at a time. The first patient arrives at time zero (8:00 AM), and the next arrives at a precise time determined by the desired number of patients scheduled. For example, if the clinic is supposed to serve two patients, the interarrival time is

240 minutes, while a clinic serving eight patients daily would have an interarrival time of exactly 60 minutes.

The output of the create block uses an "alter" block to add one to the number of patients available to seize for appointments. The patient cases proceed to an "alter" block which adds one to the number of "appointment" resources the simulation will attempt to complete. A "duplicate" block then creates four additional copies of the patient's case, allowing the simulation to execute the next four tasks simultaneously.

As patients congregate to the spoke site, they must check in. Each one seizes a patient resource, spoke site staff, and an appointment resource. It delays them all for expo(5), then releases the patient and spoke site staff for other tasks. It keeps the appointment resource seized; it will not release it until the entire appointment process is complete, effectively keeping the same appointment from happening twice.

The second copy of the original group goes to the spoke site technology specialist. The copy seizes the employee as well as the AV equipment at the spoke site. The simulation delays for expo(10) to model the time the spoke tech specialist needs to prepare the room at the spoke site and set up the connection to the hub site. At that point, the employee is free to go, but the AV equipment remains unavailable until the completion of the group therapy session. The third copy triggers the hub site tech specialist to spend expo(10) establishing the connection on the other side.

The fourth and final copy travels immediately to a hub site staff person, who takes expo(5) to gather up-to-date information on the patients and transfer that information to the specialist. The specialist then takes expo(20) researching and preparing for the upcoming session.

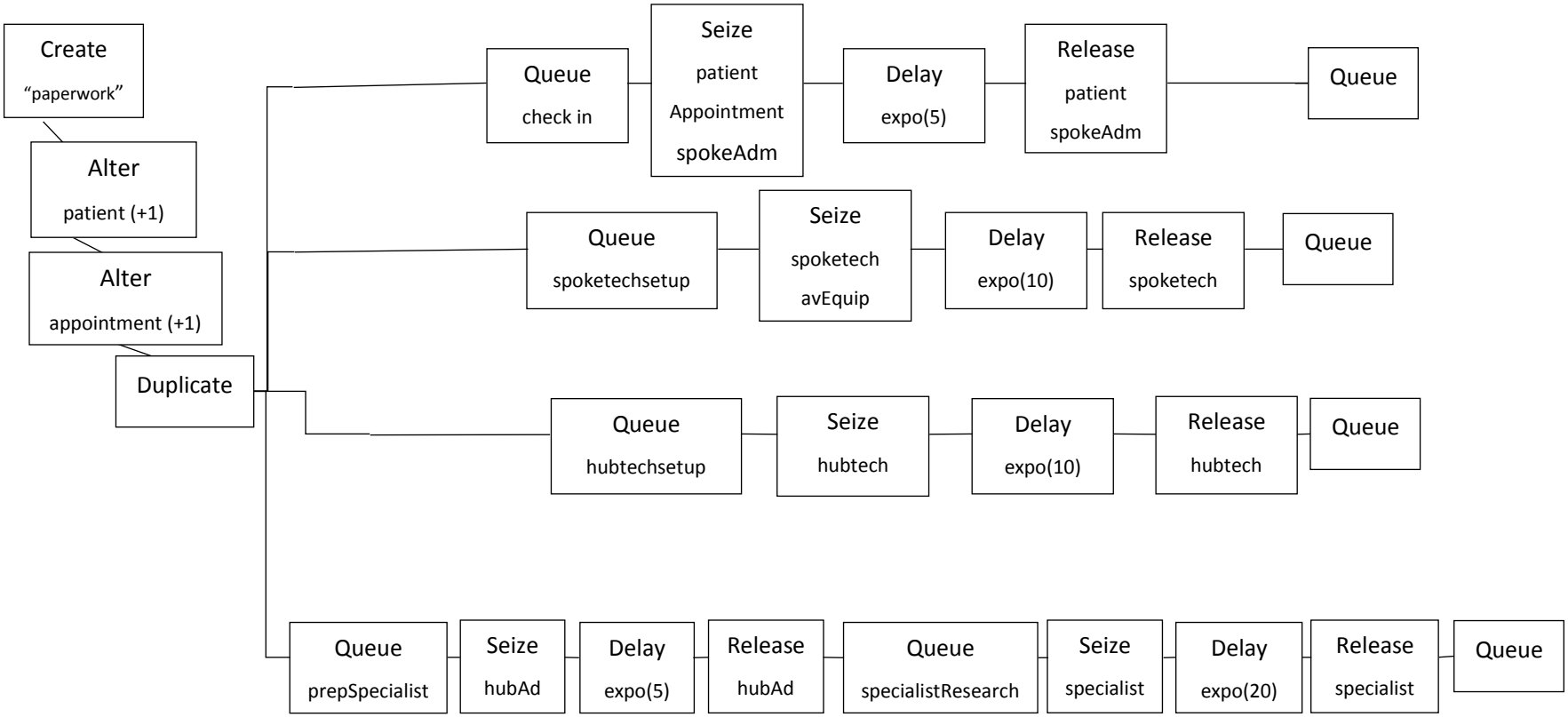

Figure 2.6– Simultaneous preparatory events for the synchronous individual session

The four copies empty into queues which feed into a "match" block which bars progress until all four tasks are complete. When that occurs, all four copies proceed at once. Three simply get disposed, but the first, the original, goes on to trigger the start of the group session. The main event requires exactly 60 minutes of simultaneous time from the patient, a general practitioner (GP)/nurse (to sit in on the session and serve as an in-person point of contact for patients), and the specialist (virtually present through a teleconferencing or telemedicine service). At the end of the session, the simulation releases only the AV equipment; the people involved in the session have follow-up work to do. Figure 2.7, below, shows this simple part of the simulation.

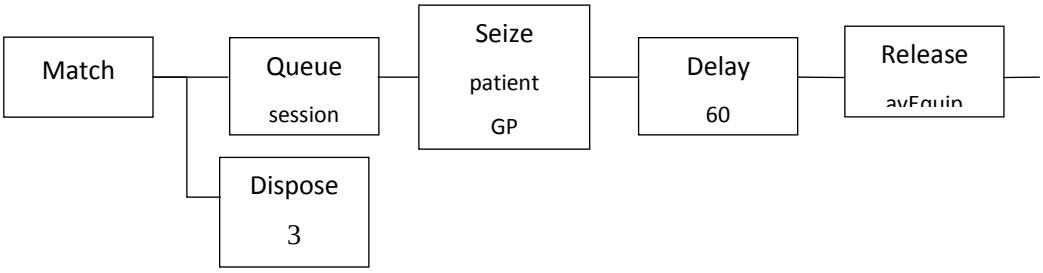

Figure 2.7 – The completion of the preparatory activities and session itself for the individual synchronous model

Upon the completion of the session, general practitioner/nurse takes expo(8) minutes after the session to talk to patients, answer questions, and collect feedback. They send their notes to the specialist when complete. At the same time, the specialist then takes expo(20) minutes to prepare follow-up from the session and recommend further treatment, if necessary. The simulation releases them as they give their follow-up to hub site staff. Figure 2.8 illustrates this.

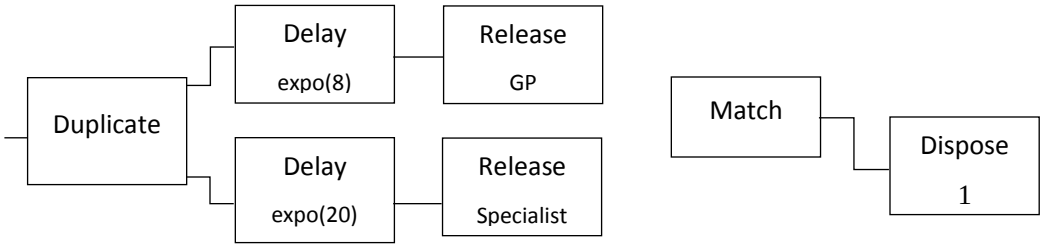

Figure 2.8– Immediate follow-up from the session in the individual synchronous model

When the match block allows the simulation to progress (that is, when the specialist and general practitioner complete their follow-up work), the hub site staff takes expo(10) to send that follow-up to the spoke site. Finally, the patient takes expo(5) minutes of a spoke site staff member's time to receive follow-up instructions. Figure 2.9 shows this logic.

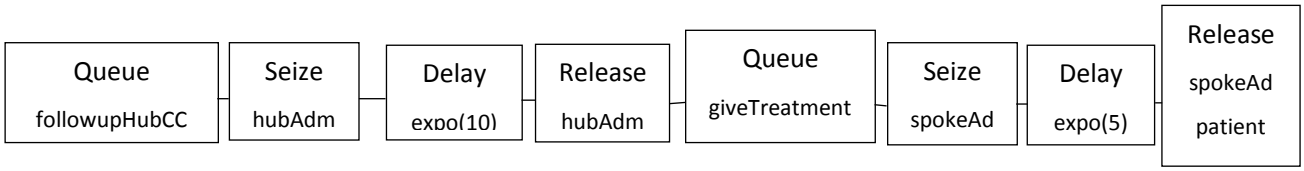

Figure 2.9– Wrap-up activities for the individual synchronous session

After this, the patients are free to go. A "duplicate" block makes a copy of each patient and sends it to the "duplicate" block at the start of figure 3 via a "delay" block that holds each patient for 2280 minutes. In more intuitive terms, after the appointments, patients wait a week (minus the time of the appointment itself) and go on to their next one. Since the simulation only lasts a day, this means the patients are effectively gone. That said, this construction also grants flexibility for later variations of the model that look at larger timeframes. On the other side of the duplicate block, the simulation releases the appointment and permanently reduces the number of available appointments by one. Figure 2.10 shows this part of the simulation.

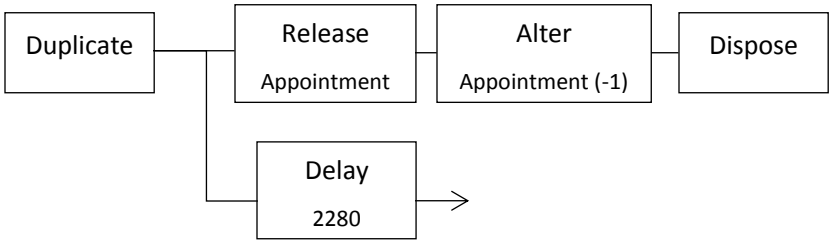

Figure 2.10 – Final simulation closure logic for the individual synchronous session

The model runs for 200 replications, each lasting one eight-hour day (480 minutes). At the end of each day, the queues and statistics get reset; each day is independent of every other day. The table below summarizes the general structure of the model.

Table 2.1– Tasks and their times in the individual synchronous model

Action	Time	Actor	Assumptions
Referral Review	10 min	Specialist	Occurs 25% as often as appointment execution. No patient is turned away.
Enrollment in Program/Session	10 min	Hub Admin	
Check-In	5 min	Spoke Admin, Patient	
Room Technology Preparation	10 min	Spoke Technology Specialist, Room	
Hub-Side Video Preparation	10 min	Hub Technology Specialist	
Gather Materials for Specialist	5 min	Hub Admin	
Prepare for Session	20 min	Specialist	
Run Session	60 min	Patient, Nurse/GP, Specialist, Room	
Answer Patient Questions/Follow-Up	8 min	Nurse/GP	
Prescribe Treatment and other Follow-Up	20 min	Specialist	
Send Treatment to Spoke	10 min	Hub Admin	
Check-Out	5 min	Spoke Admin	

2.2.2 – Simulation Results

Arena performed 200 replications individual synchronous telemental health model for each of several varying patient scheduling levels. Essentially the model scheduled some number of potential appointments and tracked how many the available staff could actually fulfill in the

allotted time. The simulation tracked the utilization of staff members and physical resources while also gauging the number of appointments that could be completed in that time. Figure 2.11, below, shows how employee and resource utilization changes as the number of enrolled patients increases and table 2.2 provides those numbers directly.

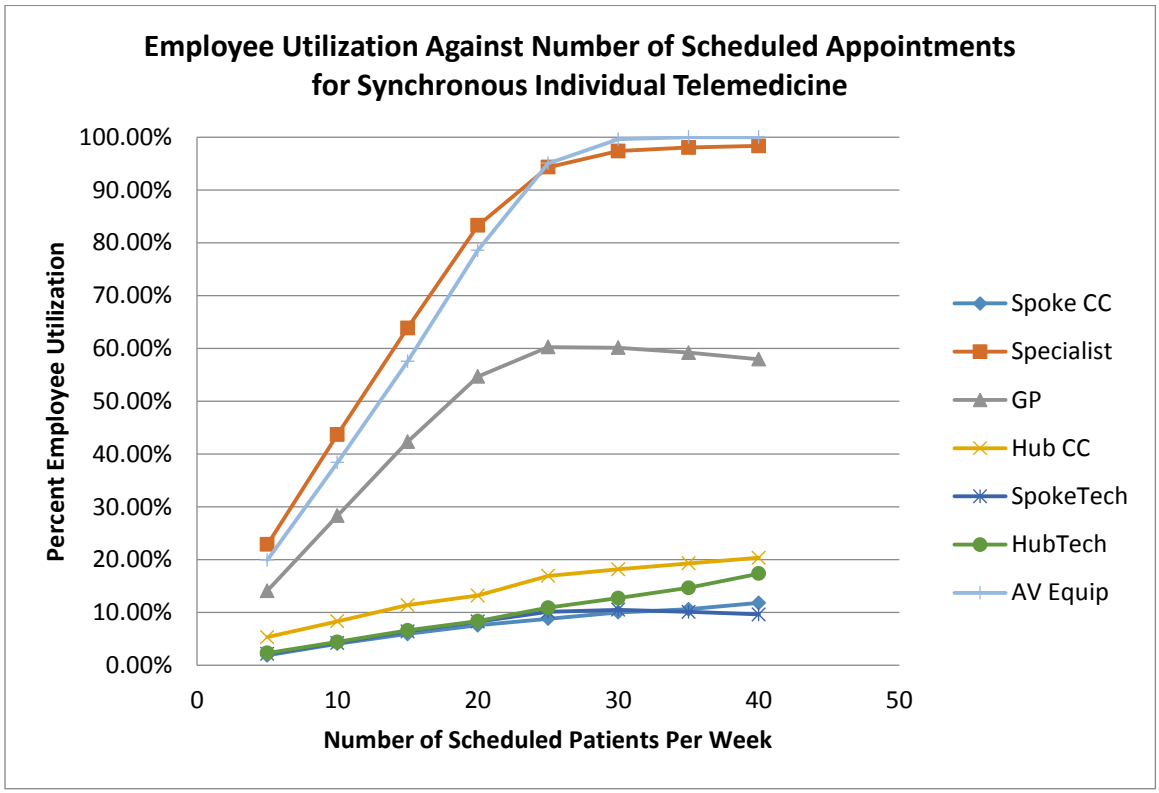

Figure 2.11– Employee utilization for the synchronous individual model.

Table 2.2– Resource utilization for the synchronous individual model

# patients	Spoke CC	Specialist	GP	Hub CC	SpokeTech	HubTech	AV Equip
5	1.89%	22.87%	14.11%	5.34%	2.15%	2.36%	19.95%
10	4.09%	43.69%	28.34%	8.34%	4.25%	4.45%	38.40%
15	5.94%	63.87%	42.32%	11.39%	6.44%	6.61%	57.52%
20	7.59%	83.28%	54.69%	13.21%	8.27%	8.37%	78.57%
25	8.84%	94.29%	60.25%	16.95%	10.14%	10.92%	95.03%
30	10.04%	97.35%	60.14%	18.20%	10.53%	12.74%	99.60%
35	10.63%	98.02%	59.20%	19.30%	10.13%	14.67%	99.98%
40	11.81%	98.34%	57.94%	20.38%	9.63%	17.36%	100.00%

The AV Equipment and Specialist approach 100% utilization at 30 scheduled appointments per week. The general practitioner approaches 60% utilization after 25 appointments per week. Synchronous individual telemedicine systems with one specialist have an effective limit of about 25 appointments per week, or five per day. This leaves roughly three hours per day for non-session activities the specialist needs to complete, such as reviewing referrals, preparing for sessions, and performing follow-up. Figure 2.12, on the next page, details the waiting times for certain processes in this model and shows how it varies as the number of scheduled patients increases.

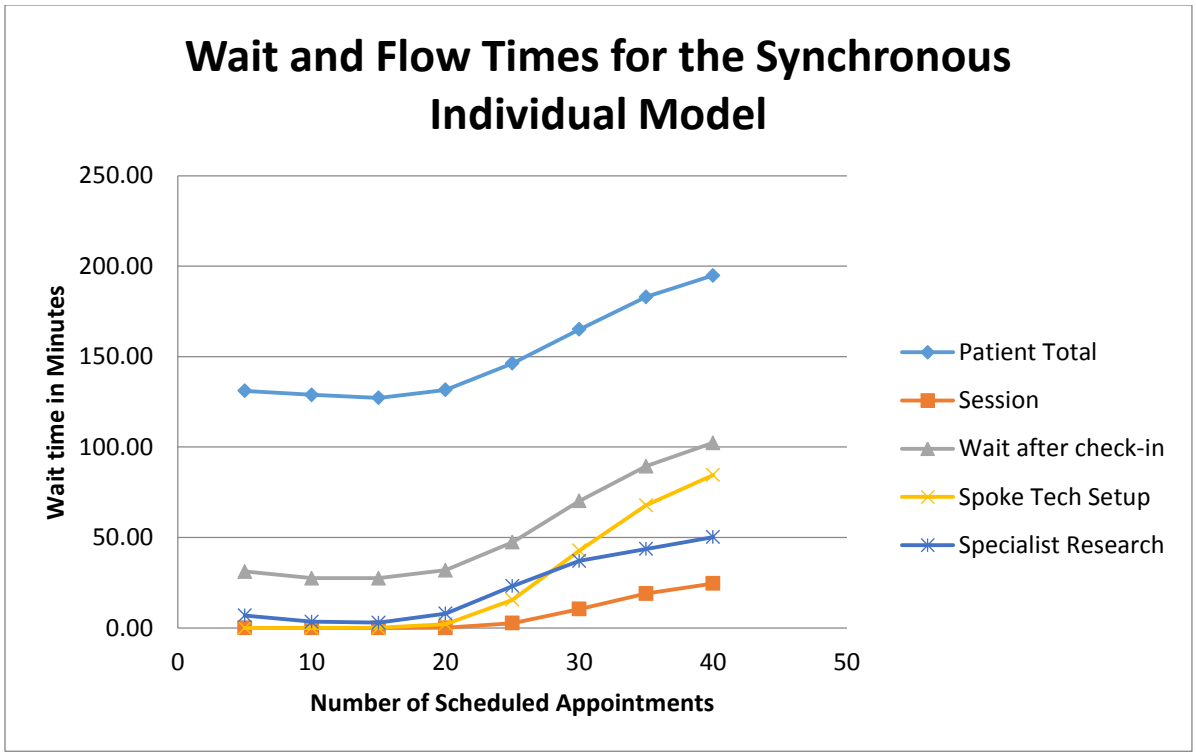

Figure 2.12 – Times for the synchronous individual model by number of scheduled appointments

In the figure above, "Patient Total" refers to the total time for the patient from their arrival to when they leave the building. "Session", "Spoke Tech Setup", and "Specialist Research" refer to the waiting time to execute those steps of the process. "Wait after check-in" plus "session" shows to how long patients have to wait after checking in before their appointment starts. Note how the wait times remain about constant until the four appointments per day mark, after which they start rising. The average patient waits for thirty to fifty minutes after checking in before their session begins, but beyond that the simulation processes patients at a fairly reasonable rate. These realistic times help validate the model and indicate it provides a reasonable approximation of similarly configured telemedicine systems.

Chapter 3 - An Introduction to Petri Nets

Petri nets model systems of discrete events, such as telemedicine processes. They can act as powerful tools for modeling telemedicine system reliability and effectively predict if workflows can get "derailed" during a typical day's operations. While valuable, their use has largely remained limited to the computer science and manufacturing fields. This section discusses the basic principles and properties of Petri nets. Readers familiar with their use may skip to Chapter 4: Modeling for a thorough description of how Petri nets can model telemedicine systems.

3.1 – Petri Net Fundamentals

Petri nets model systems of events and specialize in capturing concurrency, conflict, and preconditions for actions. They first appeared in Dr. Carl Adam Petri's 1962 Ph.D. thesis, "Communication with Automata," and have been the subject of an entire body of research since then (Girault & Valk, 2003).

3.1.1 – Petri Net Mechanics

Rooted in graph theory, Petri nets take a simple assembly of nodes and arcs and apply a unique set of rules and mechanics to model a wide array of systems. Tadao Murata's 1989 paper "Petri Nets: Properties, Analysis and Applications" offers an excellent introduction to these mechanics and still serves as a reference in many contemporary papers (Murata, 1989). Most of the discussion in the following section paraphrases Murata's work.

Petri nets have two basic components: places and transitions. Transitions represent actions. They usually require inputs to "fire", consume those inputs, and usually generate outputs upon firing, which in turn may become inputs for other transitions. Rectangles denote transitions

in visual representations, with arrows pointing into the rectangle symbolizing inputs and arrows pointing out of the rectangle symbolizing outputs.

Places represent conditions. Places may be marked with some number of "tokens", which denote the state of that condition. Firing transitions consumes tokens if they receive input from a place, and add tokens if they output into a place. Graphically, circles denote places, and a dot (or number of dots) inside the circle indicates the number of tokens. Outgoing arrows are inputs for transitions, and incoming arrows are outputs for transitions. Note that these arcs (arrows) always link transitions to places and vice versa. Places are never linked to other places, and transitions are never linked to other transitions. A place with an arc *from* the place *to* a transition is called an "input place" for that transition. A place with an arc *to* the place *from* a transition is called an "output place" for the transition. For Petri nets without weighted arcs, transitions cannot fire unless all their input places contain at least one token. Figure 3.1, below, shows places, transitions, input arcs, and output arcs in a very basic net. The leftmost place has one token in it. Firing the leftmost transition would remove that token from the leftmost place and add a token to the center place.

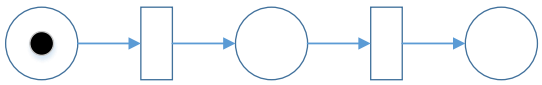

Figure 3.1- A very basic Petri net with three places (circles) and two transitions (rectangles)

All Petri nets have an initial marking, that is, a list of the number of tokens on each place at the beginning of analysis. Initial markings may have a strong impact on the eventual properties of the system and most net properties depend heavily on them.

Putting this all together results in a model capable of portraying any combination of any number of states of readiness, called a "marking", and examining how performing actions within the system impacts the system's state and capabilities. Each transition that fires changes the

marking. A firing sequence refers to a series of transitions that fire in a specified order. Places have no artificial constraint on the number of tokens they can contain, so a single initial marking could generate an infinite number of different firing sequences in some nets.

The example in figure 3.2, on the next page, illustrates firing sequences and conflict. Box number one shows a Petri net. Transition one (*t1*) has on input arc. Its input place has one token in it, so *t1* it can fire. In Petri net terminology, *t1* is "enabled". Firing *t1* results in the marking in box number two. Transition one removes one token from its input place and puts one token on each of its two output places. With the marking in number two, both *t2* and *t3* are enabled and either one may be fired. Firing *t2* results in the marking in number 3a. Transition 2 removes one token from both of its input places and puts one token in its output place. Firing *t3* yields the marking in number 3b, where the transition takes one token from its input place and puts one token on its output place. Firing either *t2* or *t3* while in marking number 2 disables the other transition; the two transitions are in conflict.

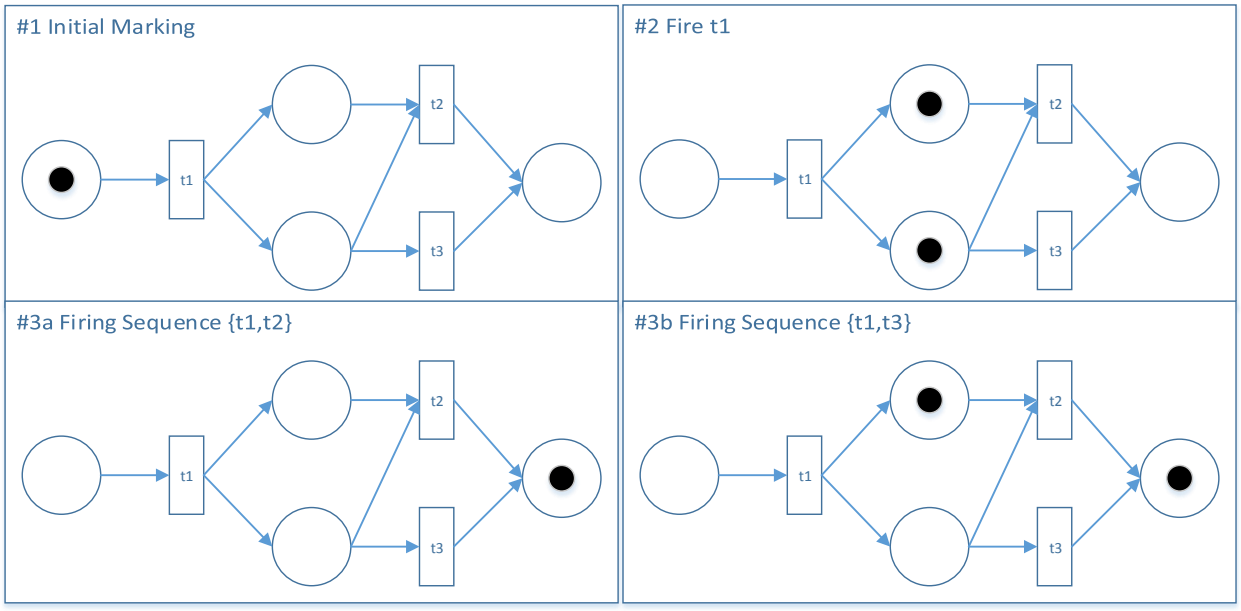

Figure 3.2 – Four possible markings arising from an initial marking given in the top-left box

Petri nets model discrete-event systems and are vaguely similar in nature to discrete-event simulations. That said, they have a few *absolutely* critical differences. While discrete-event simulations rely on time as an integral part of their simulation logic, Petri nets have no conception of time whatsoever. Discrete-event simulations are, by nature, stochastic, while Petri nets are deterministic. Analysis with discrete-event simulation tends to focus on numerical output, while Petri-net analysis yields qualitative properties. Finally, while events in simulations *will* occur after some period of time, Petri Nets are *permissive*, meaning that enabled transitions do not have to fire (Peterson, 1981). For these reasons, the two tools have very different uses. While discrete-event simulation offers an excellent way to model a specific telemedicine system, find system bottlenecks, predict output and capacity, and measure cycle-times, Petri nets generate more generalized results, revealing system properties applicable to any system with a similar configuration.

3.1.2 – A Mathematical Definition of Petri Nets

Formal definitions of Petri nets have varied somewhat over the past 30 years. This paper will use a simplified definition heavily based on Murata's work (Murata, 1989):

Table 3.1 – Formal Definition of Petri Nets

A Petri net is a 4-tuple $PN = (P, T, F, M_0)$, where:
$P = \{p1, p2, p3, \ldots, pm\}$ is a finite set of places,
$T = \{t1, t2, t3, \ldots, tn\}$ is a finite set of transitions,
$F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs,
$M_0 : P \rightarrow \{0, 1, 2, 3, \ldots\}$ is an initial marking.
$P \cap T = \emptyset$ and $P \cup T \neq \emptyset$

Mathematically, Petri Nets are bipartite directed graphs, where places and transitions make up the two partitions of the set of vertices (Girault & Valk, 2003).

Murata includes another term, *W*, representing arc weights. The nets in this paper do not use any weights; all arcs have an effective weight of one. Consequently, the formal definition omits this term for simplicity.

3.2 – State Transition Equations

While a graphical representation may be the most intuitive way to express a Petri net, other forms are more amiable to computer analysis. Petri nets may be modeled as a series of equations most compactly shown in an incidence matrix. This matrix, often denoted in literature as *A*, has the dimensions *n* x *m*, where *n* is the number of transitions and *m* is the number of places in the net. An entry a_{ij} within *A* denotes the *net* change in token number in place *j* after firing transition *i* (Murata, 1989). For example, a value of +1 at a_{23} means that transition two

places one token on place three, while a -1 would mean that the transition consumes one token from place three as an input.

Note that this matrix can be decomposed into the difference between two simpler matrices. Murata defines a_{ij}^{+} as the weight of the output arc from transition *i* to place *j* and a_{ij}^{-} as the weight of the input arc from place *j* to transition *i*. Taking $a_{ij}^{+} - a_{ij}^{-}$ yields a_{ij}, the net change in tokens after firing the transition.

The change in tokens in all places from firing one transition is simply the row of the *A* matrix corresponding to that transition. It can be mathematically isolated by multiplying the transpose of *A* by the "control vector" u_k, which is a column vector with a number of rows equal to the number of transitions, in the same order as in *A*. A single control vector contains only ones or zeroes. A one in the vector represents firing the corresponding transition one time. For example, a control vector where the first entry is a one and the rest are zeroes represents firing the first transition once. A number of firing vectors can be added to one another and multiplied by the transpose of A to capture the effect of several transitions firing in some sequence. A three in the control vector denotes that the corresponding transition fires thrice in the firing sequence, for example. Note that this technique cannot detect if transitions are enabled and captures no information regarding the order of the firing – this poses a fundamental constraint on the usefulness of this analytic technique.

The marginal change in tokens found by multiplying the transpose of *A* by either u_k or the sum of *d* control vectors can be added to the initial marking M_0 (or some other marking *M*) to find the marking M_d resulting from the firing sequence in u_k. This relationship goes by several terms in literature, the simplest of which is the state equation. Equation 1, on the next page, shows the general form of the state equation:

$$M_d = M_0 + A^T \sum_{k=1}^{d} u_k \quad (1)$$

While this opens the door to some powerful forms of analysis, doing so inflicts the model with a few restrictions. Consolidating all input and output information into one number results in a loss of information that may allow some transitions to fire inappropriately. Most notably is the inability of the model to handle self-loops, where one transition has a given place as both an input and an output. However, a simple change to the Petri net can handle most self-loop problems. If a transition has a self-loop, the event it represents may be possible to model with two transitions instead: one representing the start of the event, and one representing the end. These two transitions interact with the same place, but one only uses the place as an input while the other uses it as an output. While useful, these dummy transitions also make the net larger than it needs to be for other types of analysis.

3.3 – Petri Net Properties

Compared to other modeling tools, Petri Nets stand out due to their simplicity and power. Unfortunately, basic Petri Net theory has no concept of time. Events modeled by transitions are assumed to happen instantaneously. The only constraint on the firing order of transitions is if they are "enabled", that is, if all input places to the transition contain tokens. Petri nets do not force enabled transitions to fire, however; this property is known as "permissiveness". Since Petri nets are permissive and do not account for time, they lack the specific predictive power of something like a discrete-event simulation.

This weakness becomes a strength, however, when one uses Petri nets to examine the long-term behavior of various system configurations. A number of Petri net properties model desirable or undesirable system behaviors. The following sections contain a sampling of those

most relevant to telemental health, adapted from Tadao Murata's 1989 paper on basic Petri net theory (Murata, 1989).

3.3.1 – Reachability

Given an initial marking, does some firing sequence exist that can produce some other, given marking? This property is useful for both troubleshooting the accuracy of the model as well as analyzing ways the system could potentially behave. Reachability analysis proves very powerful for diagnosing potential failure modes (Henry, Layer, & Zaret, 2010).

For example, consider the Petri net on the next page in figure 3.3. Suppose the net represents a chemical storage facility and transition three represents some catastrophic chemical reaction caused by storing two chemicals next to each other. Naturally, designing the system or choosing an initial marking that prevents transition three from firing is desirable. This can be accomplished by performing reachability analysis for any state that has a token in both place D and place E. For this initial marking, a state where D and E are both marked is not reachable. Transition 1 can generate an infinite number of tokens in place A and B, but firing transition 2 to put a token on place E also removes a token from place E. Since D has only one token and no way of regaining it upon its loss, a marking where D and E are marked is not reachable. Returning to the hypothetical example, this net could symbolize a procedure to remove a set amount of chemical D from a dangerous spot before moving in a new chemical E nearby. Note that having more tokens in D in the initial marking makes undesirable markings reachable.

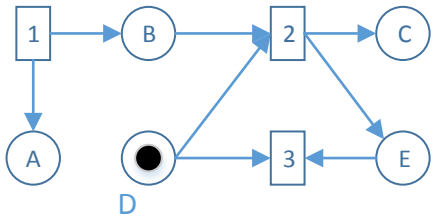

Figure 3.3– Sample Petri net for reachability analysis

Strictly speaking, reachability tests all places in the entire marking for strict equality. In the example above, this would mean asking the question "does there exist a marking where places A, B, C, D, and E all contain exactly one token?", or something similar to it.

3.3.2 – Coverability

Given an initial marking and a target marking, does some firing sequence exist that can produce a marking that, in each place in the net, contain a number of tokens greater than or equal to the number of tokens in that place in the target marking? This property is like a looser version of reachability, where strict equality to the target marking get replaced by a "greater than or equals to" sign. From an analysis perspective, this property can test for undesirable behaviors with less work than reachability. For example, a coverability analysis could ask the question "does there exist in the above Petri net a marking where D and E have at least one token and all other places have at least zero tokens?" This statement captures all possible markings where transition three could be enabled and is a much more powerful result than one particular marking not being reachable.

3.3.3 – Boundedness/Safeness

Given an initial marking, what is the maximum number of tokens that can appear on any place at any time? For unbounded nets, this number is infinity. For bounded nets, this number is

some integer. If this number is one, the net can be called “safe”. This property conveniently models upper bounds on necessary system capacity. The net figure 3.3 is unbounded, since transition one can be fired an infinite number of times. The net below, in figure 3.4, is safe. Transition 1 can fire one time, after which transition 2 can fire one time. While the total number of tokens in the net increases, no place ever ends up with more than one token, making the upper bound for number of tokens on each place equal to one.

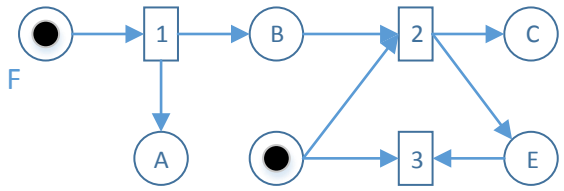

Figure 3.4– An example of a safe (1-bounded) Petri net

3.3.4 – Liveness

Given an initial marking, does some firing sequence exist that fires any number of transitions any number of times? If a net is live, then it is free of deadlocks and any transition can eventually be fired after reaching any marking reachable from the initial marking. Liveness can be examined on a transition-by-transition basis as well. A live transition can be eventually fired in some firing sequence from any given marking, while a dead transition can never be fired in any firing sequence.

Limited forms of liveness exist for transitions. L1-liveness means the transition can be fired at least once given an initial marking. L2-liveness means it can be fired some integer number of times. L3-liveness means that it can be fired an infinite number of times, and L4-liveness means that it can be fired at least once given any arbitrary initial marking.

The examples in figures 3.3 and 3.4 are both reach partial or full deadlocks after which transition two can no longer fire. The net below in figure 3.5, however, is live for any initial marking. Transition one is L4-live and transitions two and three are L3-live.

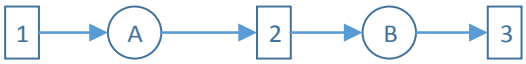

Figure 3.5– A live Petri net

Liveness is a very strong property and can be difficult to prove for complicated nets. That said, it also should appear in nets that represent systems that process an essentially infinite resource or cycle around to perform the same actions over and over again.

3.3.5 – Fairness

Two transitions have a bounded-fair relationship if neither can fire an infinite number of times without the other firing. A firing sequence is unconditionally fair if it is either finite or if every transition in the net appears an infinite number of times. The net in figure 3.6, below, illustrates both types of fairness. Transitions one and two have a bounded-fair relationship since transition one cannot fire until transition two fires and vice versa. The firing sequence 1-2-1-2… is unconditionally fair and infinite.

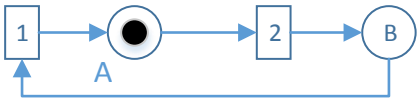

Figure 3.6– A Petri net exhibiting several types of fairness

3.3.6 – Controllability

Is “any marking reachable from any other marking”? Controllability is another strong property but can be very useful. The Petri net in figure 3.5 is completely controllable since transitions can be fired to produce any arbitrary marking on places A and B.

3.3.7 – Conservativeness

Does the number of tokens in the net remain constant? If so, a net is conservative. Alternatively, places may be assigned weights (including a weight of zero) to account for concurrency or other modeling features. A Petri net with weighted places is conservative if, regardless of the firing sequence, the weighted sum of tokens remains constant. The net in figure 3.6, above, shows conservativeness since it will always contain exactly one token. Adding a second place (call it C) in parallel to place B and then assigning each a weight of ½ would also maintain conservativeness.

While modeling systems with Petri nets proves moderately useful for clarifying system logic in and of itself, analytic techniques to determine the properties of Petri nets really make the tool shine. In particular, variants on reachability, coverability, and boundedness can be used to model undesirable situations in business processes that can be rooted out with the help of Petri nets. The rest of this section outlines mathematical techniques to prove or disprove these properties for Petri nets.

3.4 – Solving the Reachability Problem

The reachability property of Petri nets can prove useful for determining the long-term health and stability of the system. One may identify a few undesirable states, develop a marking to model each one, and determine if the markings are reachable.

Using reachability to diagnose potential failure modes works well in numerous fields. One paper applies a reachability and coverability analysis to cybersecurity, modeling potential attacks as transitions and security vulnerabilities and exploits as places (Henry et al., 2010). Given an initial marking that represents an existing set of vulnerabilities, a search for reachable

states eventually lists all the vulnerabilities that could be reached using existing footholds to open other vulnerabilities.

3.4.1 – The Reachability Tree

The reachability tree is one of the most traditional methods of determining the reachable states of Petri nets (Peterson, 1981). It can be used to determine reachability, coverability, and boundedness. This makes the reachability tree an excellent tool for analyzing workflows.

The reachability tree is a graphical representation of all states reachable from some initial marking. The tree takes the initial marking as a root node and creates an arc for each enabled transition. The markings resulting from the firing of each enabled transition then make up the next layer of the tree and the process repeats. If some places have strictly more tokens in the new marking than in the old and the new marking covers the old, the tree is unbounded and can get a potentially infinite number of tokens. Consequently, this tree can grow infinitely if the net is live and new markings can always be reached.

A simple workaround for this problem exists and is known as a coverability graph (Murata, 1989). Short of simply enumerating each reachable state, the coverability graph summarizes a potentially infinite number of markings by terminating a branch when it reaches a marking that covers a previous marking. Transitions enabled in the covered marking are enabled in the marking that covers it, meaning that the net can potentially loop around again and again, covering that marking an infinite number of times. This loop gets marked in the tree and the branch terminates. Places that have an infinitely growing number of tokens in this loop get denoted with "ω" number of tokens.

A net is bounded if no ω appears in the graph. A transition in a net is at least L1-live if it appears as an arc in the coverability graph (Murata, 1989).

As an example, consider the Petri net in figure 3.7, below. The coverability graph of this net can be found in figure 3.8 directly below. In this simplified notation, the letters in the circles represent one token in the given place, such that "A, C, E, E" signals one token on place A, one on place C, and two on place E. In its initial marking, both transitions one and three can fire, leading to two branches on the graph. Firing one or three from the initial marking does not prevent later firing the other. However, firing transition one first enables transition two to fire. Firing transition two prevents later firing of three and vice versa. Transitions two and three can be said to be "in conflict" at this point. Every transition has an arc in the coverability graph, making them all at least L1-live. No ω appears in the graph, indicating the net is bounded. Specifically, the net is 2-bounded since the firing sequence (t1,t2) puts two tokens on place E.

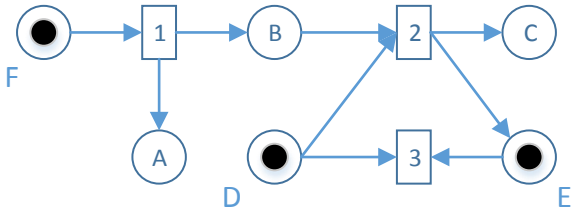

Figure 3.7– Example Petri net for coverability graph construction

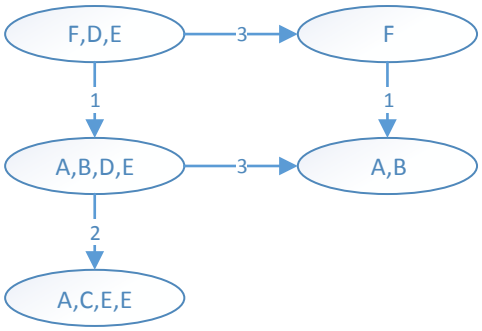

Figure 3.8– Coverability graph of the net in figure 3.7

The figure on the next page shows the coverability graph for figure 3.3, the Petri net used to discuss reachability. The ω following some letters indicates that those places could potentially have an infinite number of tokens and hence any arbitrary marking can be covered by another

with more tokens on those places. For example, firing transition one from the initial marking will result with one token on places A, B, and D. Transition one can then be fired infinitely more times to put infinitely more tokens on places A and B. D will only ever hold one token. In this net, transition three is not live and the net is unbounded since the ω symbol appears in the graph.

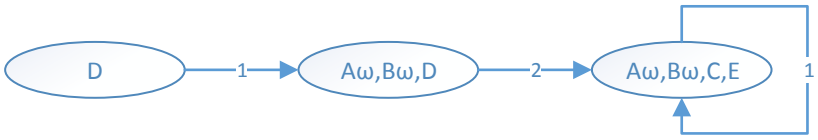

Figure 3.9– Coverability graph of the net in figure 3.3

The coverability graph offers a convenient way to summarize even infinitely-sized reachability trees for small Petri nets. Even with this space-saving assumption, however, formulating coverability graphs for moderately-sized Petri nets can take a large amount of memory and time.

Numerous software packages offer a convenient set of tools to model Petri net reachability trees. While a substantial number of software options exist, many date back to the 1990's, lack vibrant user bases, have poorly-designed user interfaces, require uncommon operating systems, or simply are not easy for those unfamiliar with Petri nets and computer science to use (Bonet, Llado, Puigjaner, & Knottenbelt, 2007; Thong, 2015).

For this analysis, the Platform Independent Petri net Editor tool (PIPE) offered the most appealing set of features. PIPE is an open source, free-of-charge analysis tool developed at Imperial College in London, England and available online (Dingle, Knottenbelt, & Suto, 2009). The user community actively supports the software and it features a wide range of tools. PIPE can quickly classify nets, includes a simulation tool, can determine if nets are bounded and deadlock-free, and even constructs graphical depictions of the coverability graph. Additionally,

PIPE's intuitive user interface makes it appropriate for those with limited familiarity with Petri nets.

Unfortunately, the sheer size of many Petri nets, including the one used later in this analysis (see Chapter Four: Modeling), results in a "state space explosion" (Murata, 1989). Adding a few more places to a large net can frequently cause massive additions to the reachability tree. Large coverability graphs cannot be easily generated with software like PIPE on a typical personal computer. Computational complexity poses no small problem for Petri net analysis. In fact, the problems of determining reachability, coverability, boundedness, and liveness in a general Petri net all require exponential space (Bourdeaud'huy et al., 2007; Jones, Landweber, & Lien, 1977; Yen, 2008). Consequently, this thesis turns its attention to techniques capable of traversing the massive solution space of these problems within a reasonable amount of time.

3.4.2 – A Binary Integer Programming Approach

The incidence matrix representation of a Petri net provides a seemingly convenient way to model the reachability problem. Intuitively, one could simply take an incidence matrix and target marking and solve for a control vector to reach that marking. Unfortunately, the power of the state equation has some severe limitations. A target marking M_f is reachable from the initial marking if some feasible firing sequence exists that can generate that marking. Therefore, a feasible solution for the state equation for the sum of u_k would appear to quickly determine reachability. While this condition is *necessary* for reachability, a phenomenon called *spurious solutions* prevents the condition from being sufficient (Miyazawa, Tanaka, & Sekiguchi, 1996). Solving for the sum of u_k yields a vector of nonnegative integers indicating the number of times each transition fires. Unfortunately, this vector cannot be decomposed simply into an ordered

firing sequence, nor does it check if each transition in the sequence is enabled. While a solution generates a sequence of transitions, a firing sequence of *enabled* transitions may not exist.

Circumventing this problem requires a more sophisticated approach. Literature is rich with highly efficient solutions to the reachability problem for specific classes of Petri nets (Khomenko & Koutny, 2007; Li, Sun, Gao, Gu, & Zheng, 2011). Nets with conflicting transitions and strictly L2-live transitions, such as the nets used later in this analysis, are too complex to use most of these techniques. Instead, this paper uses a binary integer programming (BIP) model proposed by Bourdeaud'huy, Hanafi, and Yim (Bourdeaud'huy et al., 2007). While not the only algorithm capable of performing general reachability analysis (Mayr, 1984), BIP models can be easy to implement and easy to change, providing a great deal of flexibility.

This model provides elegant answers to several of the challenges of the general reachability problem. The formulation used for analysis of this model is below.

$$Minimize\ 0 \tag{2}$$

$$subject\ to:$$

$$\sum_{j=1}^{k-1} A \cdot u_j - A^{-} \cdot u_k \geq -M_0\ ,\ \ \forall\ k \in [1, K] \tag{3}$$

$$\sum_{k=1}^{K} A \cdot u_k = M_f - M_0 \tag{4}$$

$$n \cdot \sum_{t=1}^{n} u_{kt} \geq \sum_{t=1}^{n} u_{(k+1)t}\ , \qquad \forall\ k \in [1, K] \tag{5}$$

$$u_{kt}\ is\ binary\ for\ all\ k, t$$

Symbols in this model are the same as those discussed previously. The BIP formulation includes two new symbols, however. The constant *K* is the search depth, discussed in the next paragraph. The matrix A^- is the input incidence matrix. In this binary matrix, a one as element *(i,j)* represents that the transition in the row *i* takes the place in column *j* as input. These symbols can be found on the next page, in table 3.2.

Table 3.2 – List of terms used in BIP formulation

Symbol	Meaning
K	Number of steps in analysis. This analysis uses 500.
u_k	The control vector for a single step. This formulation broadens it to include any number of transitions in one step as long as all of them are enabled. The index j may also be used.
A	The incidence matrix
A^-	The input incidence matrix. Instead of $a_{ij}^+ - a_{ij}^-$, each element is just a_{ij}^-. This matrix represents the places that must have tokens for a transition to fire.
M_0	The initial marking
M_f	The target marking; the marking being tested for reachability
n	The number of transitions. This model has 18

The model uses a step-based approach to save computation time. A step consists of a number of transition firings that may happen simultaneously and is represented by a control vector consisting of all transitions enabled at a given marking. This leverages the natural parallelism of the Petri net to pare down the model and improve run times. The downside to this approach is the finite number of steps the model can consider at a time. The depth of the search in steps, denoted as K, must be set by the analyst.

The feasible solution for the model is a set of single-step control vectors in the order in which they fire. This model solves the problem of spurious solutions; all transitions that fire are enabled. If the model has no feasible solution, then the target marking M_f is not reachable within the tested number of steps. Note that this means the model cannot *prove* a state is not reachable in general; rather, it can only disprove reachability within a given number of steps.

The model uses an intuitive approach to solve the reachability problem and each constraint serves a straightforward purpose. Constraint (3) wards off spurious solutions by ensuring that, for every step, the transitions in the control vector are all enabled in the marking

generated by all the previous control vectors. Specifically, the marking created from the sum of all prior control vectors times the incidence matrix and added to the initial marking must have more tokens in each place than the vector representing the needed inputs for the next control vector. It does this with a new matrix called the input incidence matrix: a matrix composed of a_{ij}^- that denotes only if place *j* has an input arc to transition *i*. Two transitions in conflict (that is, one firing would disable the other) cannot fire in the same step under this constraint. All transitions selected in a step must all be able to take the needed number of tokens from their input places before any other transition places tokens on an output place.

The state equation is found in constraint (4), where the incidence matrix times the sum of all the control vectors must equal the change between the target marking and the initial marking. For reachability, this constraint has a hard equality requirement. For the coverability problem, however, a greater-than-or-equal-to sign can be substituted. This gives the model the power to do two things. First, the model can consider solutions that cover the target marking rather than meet it exactly. Second, the model can practically disregard places if the target marking has no upper bound for them. This allows the model to test for certain subsets of a partial reachability or coverability problem; one may wish to test a few places for markings above a certain threshold but ignore the others.

Constraint (4) can also prove partial reachability in a more general sense by simply iterating the constraint fewer times. Instead of evaluating every place, the model can include this constraint for just a few, relevant places. This allows the model to check for precise equality (rather than coverability) for a subset of places at the cost of making non-reachability more difficult to prove.

Constraint (5) and the binary decision variable constraint take care of housekeeping. Specifically, constraint (5) ensures that any empty steps (where the control vector is entirely zero) get moved to the end of the set of control vectors. Most target markings will be reachable in less than 500 steps, and once reached, the model fills the remaining control vectors with "do nothings". This constraint aids in readability and also allows certain types of objective functions to be used.

The objective function of the model has little use and does not highly affect the outcome of the modeling. Feasibility or infeasibility is the main goal of analysis for reachability, so the objective function only adds extra information that may or may not be interesting. The authors propose several alternative objective functions, ranging from a meaningless function to merely ensure feasibility, to more complex functions that seek to find reachable states in as few steps as possible. While the objective function does not aid in reachability analysis, it *can* point to the fastest way to reach the target marking. The precise nuances of this firing sequence depend on the objective function in question. That said, these objective functions also increase run time and may prove unwieldy for large nets.

Notably absent from the formulation is a constraint mandating that only one transition fire per step. The binary restriction on the decision variables prevents *reentry*, where a transition fires multiple times in a step, but concurrence is allowed. The absence of this constraint allows for the step-based approach discussed earlier.

3.4.3 – Disproving Reachability with the State Transition Equations

While the finite step limit makes the unreachability of states impossible to prove with the binary integer programming model, Murata offers a convenient formulation to mathematically

disprove reachability, allowing confirmation for what Bourdeaud'huy's model suggests (Murata, 1989).

The technique stems from the state equation. The full proof can be found in Murata's work, but in practice, the existence of a nonzero solution for *z* in the equation below is "a sufficient condition for nonreachability". In other words, the difference in tokens between the initial and target marking must be a "linear combination of the row vectors of B_f":

$$M_f - M_0 = B_f^T z \tag{6}$$

In the above equation, $M_f - M_0$ represents the marginal change in tokens between the initial marking and target marking. The variable *z* is an (*m*-*r*) x 1 column vector, where *m* is the number of places in the net and *r* is the rank of the incidence matrix *A*.

The matrix B_f can be derived from a concatenation of the identity matrix and the product of two specific subsets of the incidence matrix. Specifically, *A* can be decomposed as shown in equation 7, where the variables above and to the left of the matrix indicate the dimensions of each submatrix.

$$A = \begin{matrix} m-r & r \\ \end{matrix} \begin{matrix} \\ \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \end{matrix} \begin{matrix} r \\ n-r \end{matrix} \tag{7}$$

From this, the matrix B_f can be given as follows:

$$B_f = [\, I_{m-r} : -A_{11}^T (A_{12}^T)^{-1}] \tag{8}$$

The term I_{m-r} represents an (*m*-*r*) x (*m*-*r*) identity matrix. In essence, B_f represents a set of linearly independent solutions *y* to the equation $Ay = 0$. If the change in the marking is orthogonal to everything in that set of solutions, then the target marking is reachable. Equation 9, below, describes this relationship.

$$B_f(M_f - M_0) = 0 \tag{9}$$

Unfortunately, the technique in equation 9 still suffers from spurious solutions. Its contrapositive in equation 6, however, can prove nonreachability definitively.

This analysis can be performed in the software R. Appendix 2 contains sample code from an implementation of this technique.

This analysis only works for strict reachability, since it just considers the whole of one marking at a time. Since partial reachability and coverability both concern only a portion of all possible markings, this process would need to be repeated possibly infinite times to enumerate every possible marking that satisfies a given partial reachability or coverability condition.

Inability to theoretically disprove reachability poses little problem. The Bourdeaud'huy model can disprove reachability within an arbitrarily large number of steps, and many real-world systems will likely have some sort of manual "reset" periodically where the starting state returns to the initial state. As such, these systems can be engineered to avoid undesirable states for long enough to render them a nonissue.

3.5 – Determining Petri Net Boundedness

To keep workloads to a reasonable level for employees, system boundedness is a useful property. Boundedness implies that for a given initial marking, no place can ever reach an infinite number of tokens. Keeping tokens to a finite level ensures that work will not multiply out of control. This both troubleshoots potential system issues and validates the construction of models that should be theoretically bounded.

3.5.1 – Using the Coverability Graph to Determine Boundedness

Evaluation of the coverability graph can quickly determine system boundedness. Reachability trees enumerate every possible marking, so an unbounded Petri net yields an

unbounded reachability tree. Coverability graphs provide a more practical tool, as one merely needs to search the graph for markings containing an ω. This method suffers the same pitfalls as discussed above, however. This technique cannot be practically implemented for very large graphs due to computational limitations. Thankfully, the next section describes methods to circumvent this problem.

3.5.1 – Model Simplification with Boundedness-Preserving Net Reductions

While the reachability tree offers the simplest method to determine system boundedness and liveness, state explosion makes direct analysis in this vein impossible for large nets. Consequently, researchers have developed several methods of Petri net reduction that generate smaller nets while preserving interesting properties of the original, such as boundedness (Chuanliang, 2010; Hyung, Favrel, & Baptiste, 1987; Murata & Koh, 1980; Suzuki & Murata, 1983).

Fundamentally, a net transformation involves changing a "reducible subnet" (Hyung et al., 1987). This transformation can take two forms. The first is consolidation of all the elements in the subnet into a single "macroplace" or "macrotransition". This place or transition represents the entire subnet. All arcs into any place or transition in the subnet become arcs into the macroplace or macrotransition. The same holds true for arcs leading out of the subnet. The second type of transformation involves simply removing the subnet entirely.

While the transformations described later in this section maintain the properties of boundedness and liveness, they do not necessarily maintain proper interpretation of the Petri net. Consolidating several places together may result in a macroplace that has no intuitive meaning in the real-world interpretation of the net. Consequently, net transformations of this type work best

to simply find properties of more complex nets, rather than condense nets down for easier interpretation.

The list below describes some of these boundedness-preserving net reductions as they apply to Petri nets without weighted arcs, as well as examples loosely or directly adapted from a number of papers (Hyung et al., 1987; Murata & Koh, 1980; Suzuki & Murata, 1983):

- Two transitions with an unmarked place between them that only interacts with those two transitions and where the only path from the first to the second transition goes through that place can be reduced by combining the two transitions into one. The reducible subnet consists of two transitions and the place sandwiched between them. This consolidated transition can be called a "macrotransition" (Hyung et al., 1987). Murata calls this "Fusion of Series Transitions", or "FST" (Murata & Koh, 1980). Figure 3.10, below, illustrates this change. The unnamed place in the middle interacts only with transitions one and two, so it gets removed and the two places get consolidated. Input places to either transition in the original net become inputs for the new transition, and the same goes for output places.

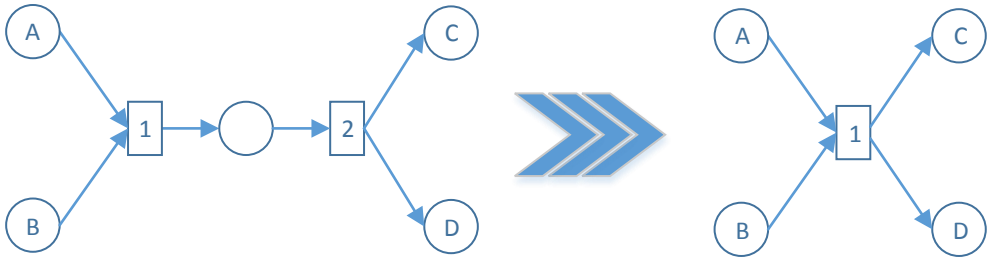

Figure 3.10 – Combining transitions in series

- Similarly, two places with a transition between them where the first is unmarked and outputs only to that transition can be merged into one place. Again, the only path from the first transition to the second must run directly through the transition. The reducible

subnet consists of the two places and the transition between them. Some literature calls the result of this union a “macroplace” (Hyung et al., 1987). Murata describes this as a “Fusion of Series Places” or “FSP” (Murata & Koh, 1980). Figure 3.11 shows this process. The unnamed place in the middle of the net gets removed and places A and B get consolidated. All input transitions to A or B (from anywhere other than the removed transition) become inputs to the new place. The same holds true for output transitions.

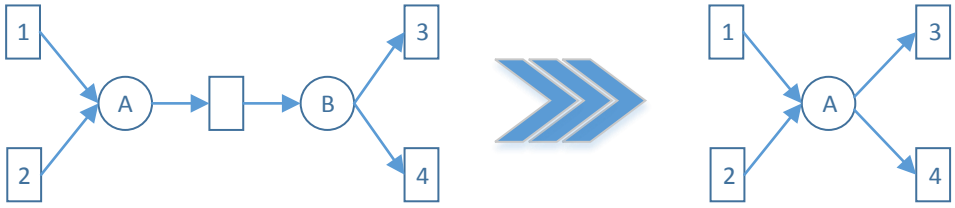

Figure 3.11– Combining places in series

- A marked place with a self-loop to one or more transitions can be removed if it has a marking greater than the number of transitions that it inputs into. This only works if the self-loop is not the only input to any of the transitions. This can be called an “Elimination of Self-loop Places” or “ESP” (Murata & Koh, 1980). Figure 3.12 illustrates this alteration, as the place between transitions one and two receives removal.

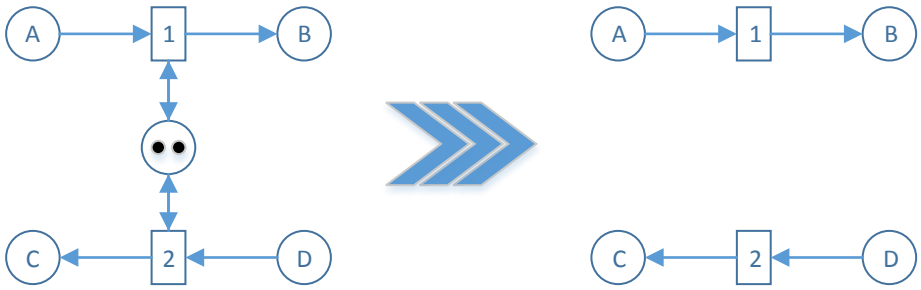

Figure 3.12 – Removing self-loop places

- A circuit where some number of unmarked places are each connected by a series of transitions in a circuit can be reduced to a single macroplace, as shown in figure 25. The

circuit including places A and B gets consolidated into a single place A'. All input transitions to any place in the circuit become input places for A', and any output transitions for any place in the circuit become output places for A'.

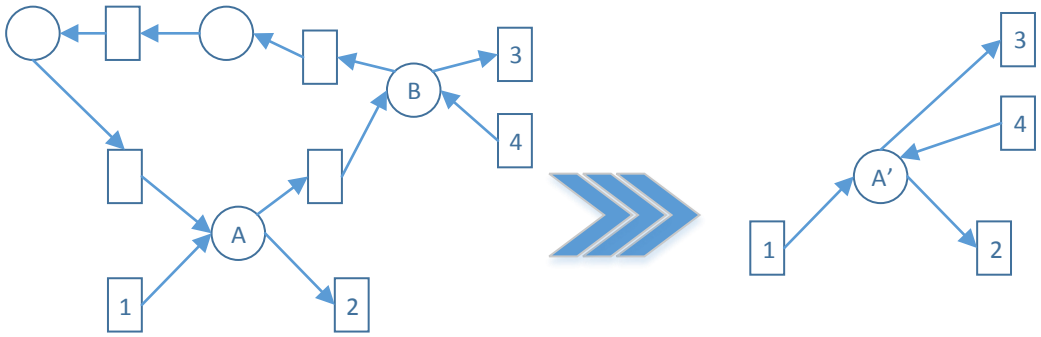

Figure 3.13 – Consolidating a circuit

- Places in parallel can be consolidated, as per the illustration in figure 3.14. In this case, places A and B get merged into place A'. Murata calls this "Fusion of Parallel Places", or "FPP" (Murata & Koh, 1980).

Figure 3.14 – Consolidating parallel places

- Transitions in parallel can be consolidated like transitions one and two in figure 3.15. This can be called a "Fusion of Parallel Transitions", or "FPT" (Murata & Koh, 1980).

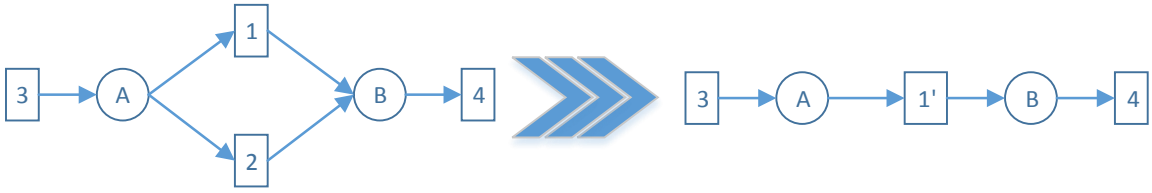

Figure 3.15 – Consolidating parallel transitions

While most work focuses on reduction of the graphical representation of Petri nets, it is possible to reduce nets based on their incidence matrix (Medina-Marin, Seck-Tuoh-Mora, Hernandez-Romero, & Quezada-Quezada, 2013). This means that large nets can be easily reduced by computer programs without tedious effort on the part of modelers. That said, current research in reduction via the incident matrix has yet to rival the body of research on graphical reduction. This area offers significant opportunities for further work.

Chapter 4 - Modeling

Telemental health systems can be easily decomposed into a handful of fundamental components which can be modeled as a Petri net. The approach belies several advantages. First of all, telemedicine systems feature significant amounts of parallelism that can be easily captured as Petri net. Next, the finite resources of telemedicine systems mean that Petri nets' ability to capture conflict works well. Telemedicine systems can be very complex, with information and signals moving all over the network and many actors waiting for dependent events. Finally, the fundamental diversity of system organization and treatment time among different providers makes general results quite useful. Petri nets do not consider time and can offer more generalized results than discrete-event simulations. Petri nets reachability analysis offers an excellent way to diagnose potential failure modes or generally undesirable states for telemedicine systems. Boundedness analysis can also determine if workloads stay at a manageable level.

Petri nets prove to be excellent modeling tools for telemedicine systems. Some research has already attempted to model aspects of telemedicine with Petri nets. One study develops a predictive maintenance tool using Petri nets for pacemaker hardware that can be implemented with telemedicine (Yang, 2004). Another study applies Petri nets to the software applications of telemedicine, using their analytic potential to engineer a solution to collaborative synchronous software deadlocks (Bouillon, 2003). Despite these applications, the business practices and care procedures of telemedicine remain a ripe field for the application of Petri net modeling and analysis.

This section discusses the specific nuances of using Petri nets to model telemedicine systems and provides an example of a generalized system. While individual systems may have

variations on the ensuing model, the basic elements and their dependencies on each other should remain fairly consistent across implementations.

The model uses transitions to symbolize the actions involved in a telemedicine delivery process. To avoid self-loops and allow for the construction of a useful incidence matrix, some processes split into two transitions. The first marks the start of the process, while the second, called a dummy transition in this paper, signals the end.

The model has three types of places that represent three different things. Regular places signal completion of a step in the process and readiness for the next step. Dummy places lie right before dummy transitions and signal that a step is in-progress. Finally, resource places represent the availability of patients and staff. Whenever an action or series of sequential actions directly involves them, the corresponding transition uses their place as an input. Whenever they finish that action or series of actions, the final transition in that series uses their place as an output. If a patient or staff member is needed for just one task at a time, this setup forms a self-loop. Self-loops cause analysis problems but can be mitigated by using a dummy transition to model the task as two transitions representing the start and end of the job.

The number of tokens in each patient or staff place in the initial marking represents the maximum number of patients or staff available to execute steps in the telemedicine process. When a transition requires the time of a patient or staff member, their corresponding place loses a token until the completion of that event. In this way, the Petri net can model the availability of staff and capture deadlocks or conflict caused by their absence. That said, deadlocks captured this way may not be absolute in practice since the system is fundamentally controlled by people that have the common sense that Petri nets lack.

In practice, one person may fill multiple roles, or one role may be filled by several people. In the former case, consolidating the appropriate places can accurately model the system (although the addition of dummy transitions may be necessary to avoid self-loops). In the latter, simply adding tokens to the initial marking to symbolize the number of people filling the role can capture having several people fill one role.

As discussed in chapter 2, telemental health systems can be decomposed into four major phases: patient enrollment, session preparation, the session itself, and session follow-up. Most systems will execute the patient enrollment phase once per patient, but will repeat the other three phases some number of times. While enrollment may occur several times in some systems, further analysis will assume that enrollment is a transient phase that must only occur once per patient.

4.1 – Modeling Synchronous Telemedicine Systems

Synchronous telemental health systems exhibit a high degree of parallelism for individual patients; consequently, Petri net models describe their behavior quite well. The remainder of this section goes over each of the four phases in treatment as they pertain to synchronous telemental health systems. A full version of the net developed can be found in Appendix 1.

4.1.1 – Patient Enrollment

Telemental health processes begin with a referral for the patient to begin therapy. As such, the Petri net representation of a telemental health system begins with *place 1: referrals received*. The number of tokens on this place represents the number of referrals for service. This place has no input arcs, so the number of patients in the system should remain static for the purposes of the model. *Transition 1: referral review* takes one token from *place 1* as an input and

one token from *resource place 3: specialist*. It outputs one token to *dummy place 1: referral review in progress*. *Dummy transition 1: finish referral review* takes this single place as input, returns a token to *resource place 3* to indicate the readiness of the specialist, and puts one token on *place 2: referral approved*. This model assumes that all referrals result in approval of further treatment.

When the referral receives approval and a token lies in *resource place 4: hub site administrator, transition 2: patient enrollment* becomes enabled. Firing it puts one token in *dummy place 2: enrollment in progress*. Firing *dummy transition 2: finish enrollment* takes a token from *dummy place 2*, returns a token to *resource place 4* (the hub administrator), and puts tokens on *places three through six* in the next phase of the net.

In short, enrollment requires only the specialist and hub site administrator to execute. Both are available by the end of the process. The number of times this process can be done corresponds to the number of referrals awaiting review, that is, the number of tokens in *place 1*. Figure 4.1 and tables 4.1 and 4.2, below, show how these places come together.

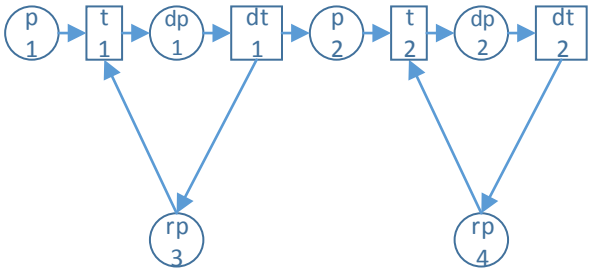

Figure 4.1– Telemedicine Phase 1: Patient Enrollment (Outgoing Arcs Omitted)

Table 4.1– Transitions in Telemedicine Phase 1: Patient Enrollment

Transition Number	Process Name	Input Places	Output Places
Transition 1	Referral Review	p1, rp3	dp1
Dummy Transition 1	Finish Referral Review	dp1	p2, rp3
Transition 2	Patient Enrollment	p2, rp4	dp2
Dummy Transition 2	Finish Enrollment	dp2	p3, p4, p5, p6, rp4

Table 4.2– Places Introduced in Telemedicine Phase 1: Patient Enrollment

Place Number	Place Name
Place 1	Referrals Received
Dummy Place 1	Referral Review in Progress
Place 2	Referral Approved
Dummy Place 2	Enrollment in Progress
Resource Place 3	Specialist
Resource Place 4	Hub Site Administrator

4.1.2 – Session Preparation

The completion of enrollment readies the model to split into four different paths that may be completed concurrently. *Dummy transition 2* places one token each on *places three through six*, discussed in turn below.

Place 3: patient may arrive represents the time about an hour or so prior to an appointment where the patient may come check in. *Transition 3: check-in* takes one token from this place as input in addition to one token from *resource place 1: patient* and one token from *resource place 5: spoke site administrator. Transition 3* outputs one token to *dummy place 3: check-in in progress. Dummy transition 3: finish check-in* takes *dummy place 3* as input, and returns a token to *resource place 5*, freeing the spoke site administrator. The patient should

proceed directly to the appointment before doing any other task, so the corresponding token is not returned. Finishing check-in also puts a token on *place 8: patient ready for session*. *Places eight through eleven* indicate completed preparation that must be marked to enable the session.

Place 4: session scheduled is an output node from the completion of enrollment. This place begins the series of preparatory events on the specialist's side. One token on *place 4* and another token on resource *place 4* (the hub site administrator) enables *transition 4: specialist priming*. In this transition, the hub site administrator gathers patient data and records to hand off to the specialist. Firing *transition 4* places a token on *dummy place 4: priming in progress*, which is an input place for *dummy transition 4: finish priming*. *Dummy transition 4* returns a token to the hub site administrator's place and puts a token on *place 7: patient data acquired*. If the specialist's place has a token, this enables *transition 7: specialist preparation*, where the specialist reviews patient data, writes questions if necessary, and generally prepares for the interview. Firing *transition 7* puts a token on *place 9: specialist ready for session*. The specialist remains occupied waiting for the session to start.

Place 5: hub space reserved is an output node from enrollment completion. *Transition 5: AV preparation (hub)* takes *place 5* as input, as well as *resource place 6: hub site technology***.** Rather than the person who manages the technology, this place represents the actual equipment itself and the number of tokens in its initial marking indicates the number of rooms or computers available for scheduling. When looking at resource utilization purely from the scope of telemedicine, the room always has slightly utilization than the technology specialists and receives use at about the same time, so modeling them as one in the same proves useful for condensing the model. Firing *transition 5* puts a token on *place 10: hub technology ready* and

symbolizes the work needed to set up the AV equipment and associated physical assets for the appointment.

Place 6: spoke space reserved starts a series of places and transitions nearly identical in function to that started by *place 5*. *Place 6* enables *transition 6: AV preparation (spoke)*, which also takes a token from *resource place 7: spoke site technology* and puts a token on *place 11: spoke technology ready*. For both this and *transition 5*, the AV equipment does not become available until after the appointment.

With all the preparatory work completed, phase two of the telemedicine system reaches its conclusion and phase three begins. Figure 4.2 and tables 4.3 and 4.4, below, show how these pieces come together.

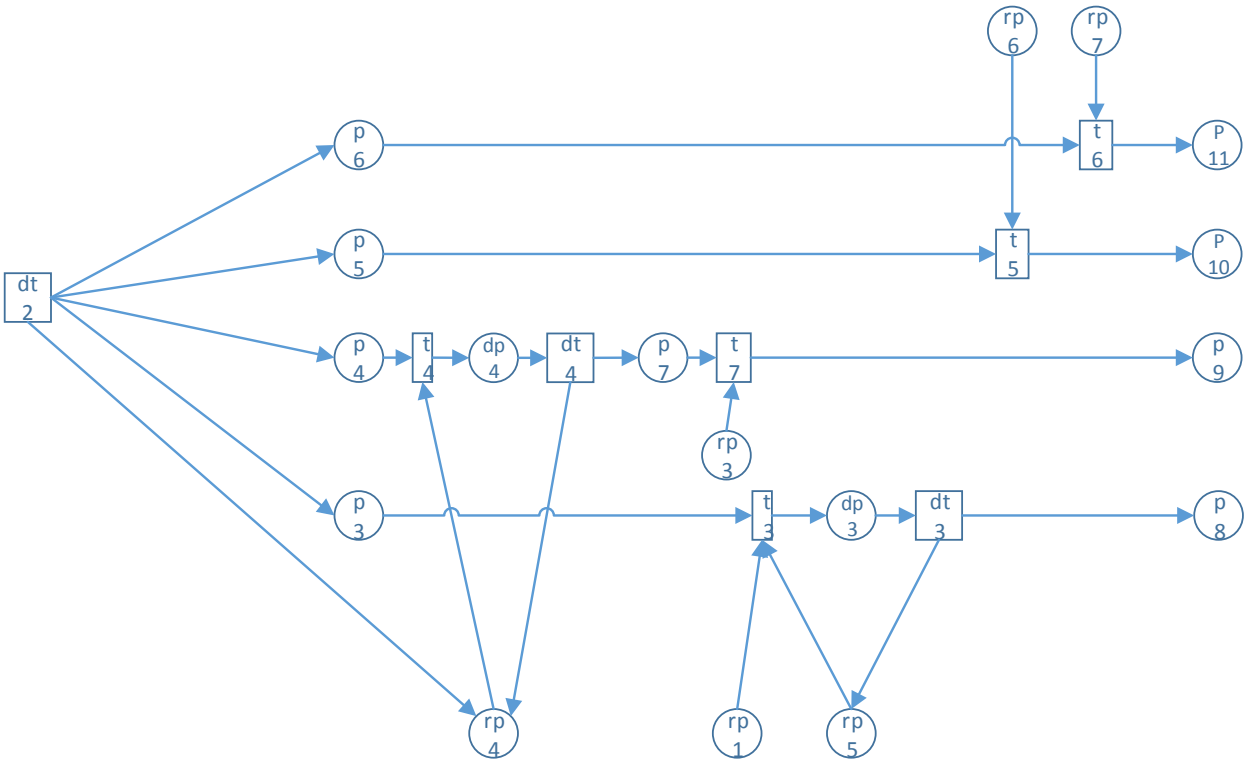

Figure 4.2 – Phase 2: Session Preparation

Table 4.3 – Transitions in Phase 2: Session Preparation

Transition Number	Process Name	Input Places	Output Places
Dummy Transition 3	Finish Check-In	dp3	p8, rp5
Transition 4	Specialist Priming	p4, rp4	dp4
Dummy Transition 4	Finish Priming	dp4	p7, rp4
Transition 5	AV Preparation (Hub)	p5, rp6	p10
Transition 6	AV Preparation (Spoke)	p6, rp7	p11
Transition 7	Specialist Preparation	p7, rp3	p9

Table 4.4 – Places Introduced in Phase 2: Session Preparation

Place Number	Place Name
Place 3	Patient May Arrive
Dummy Place 3	Check-In in Progress
Place 4	Session Scheduled
Dummy Place 4	Priming in Progress
Place 5	Hub Space Reserved
Place 6	Spoke Space Reserved
Place 7	Patient Data Acquired
Place 8	Patient Ready for Session
Place 9	Specialist Ready
Place 10	Hub Technology Ready
Place 11	Spoke Technology Ready
Resource Place 1	Patient
Resource Place 2	Nurse
Resource Place 5	Spoke Site Administrator
Resource Place 6	Hub Site Technology
Resource Place 7	Spoke Site Technology

4.1.3 – The Therapy Session

While fairly simple to model, the session itself is the most important piece of any telemedicine system. *Transition 8: therapy session* represents the appointment where the patient and doctor connect via some form of audiovisual equipment. The patient must be checked in and available (ensured by a token existing on *place 8*), the specialist must be prepared and available (*place 9*), and rooms and equipment in both the hub site and spoke site must be prepared and available (*places 10 and 11*) for the session to begin. In addition *to places 8 through 11*, *transition 8* needs input from *resource place 2: nurse***.** This employee works with the patient at the spoke site to handle any difficulties that arise.

When *transition 8* fires and the session completes, the rooms and AV equipment become available again, indicated by output arcs to resource *places 6 and 7*. Additionally, the session outputs to two places that represent the beginning of the fourth phase of the system. Figure 4.3, below, as well as tables 4.5 and 4.6 show the components of this part of the process.

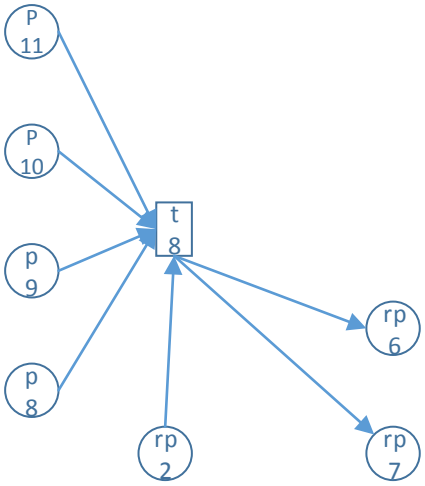

Figure 4.3 – Phase 3: Therapy Session

Table 4.5 – Transitions in Phase 3: Therapy Session

Transition Number	Process Name	Input Places	Output Places
Transition 8	Therapy Session	p8, p9, p10, p11, rp2	p12, p13, rp6, rp7

Table 4.6 – Places Introduced in Phase 3: Therapy Session

Place Number	Place Name
Resource Place 2	Nurse

4.1.4 – Session Follow-Up

The fourth and final stage of the telemedicine process is follow-up from the therapy session. *Place 12: session done (hub)*, an output place from the session's transition, enables *transition 9: specialist follow-up* when it has a token on it. *Transition 9* encompasses all post-session activity by the specialist, including consolidating notes, developing treatment recommendations, and making drug prescriptions if necessary. Firing *transition 9* frees the specialist, putting a token back on *resource place 3*, and also puts a token on *place 14: specialist follow-up complete*.

Concurrent to all this, *place 13: session done (spoke)* also acts as an output place for the session. When it has a token, it enables *transition 10: patient follow-up*, which represents all action between the patient and nurse immediately after the session. During this time, the patient may ask questions of the nurse if necessary, the nurse may elaborate on instructions from the specialist, and any last-minute notes can be sent to the specialist. Firing *transition 10* puts a token back on the nurse's place to indicate the end of their role and also puts a token on *place 15: patient follow-up complete*.

When both the hub and spoke site complete their immediate follow-up (that is, both *place 14 and 15* have tokens), *transition 11: follow-up processing* can fire. This transition also requires input from the hub site administrator's place. This short step simply involves consolidating all data from the hub and spoke sites, preparing treatment for the patient, and sending that information back to the spoke site. Firing *transition 11* puts a token on *dummy place 11: follow-up processing in progress*, which enables *dummy transition 11: finish follow-up processing***.** This dummy transition has two output places. The first frees the hub administrator by returning a token to resource *place 4*. The second puts a token on *place 16: specialist follow-up sent***.**

A token in *place 16* enables the final set of actions. If the spoke administrator is available, *transition 12: check-out* takes one token from *resource place 5* and one token from *place 16*. Firing this transition puts a token on *dummy place 12: check-out in progress*, which enables *dummy transition 12: finish check-out*. Firing this final dummy transition frees the patient and spoke administrator by putting tokens on resource *places 1 and 5*. The system is now ready to begin the next appointment of the patient, so *dummy transition 12* also outputs to *places 3-6* to enable the telemental health system to start again in phase two, the session preparation. The follow up stage can be seen in figure 4.4, with tables 4.7 and 4.8 describing the components of this segment.

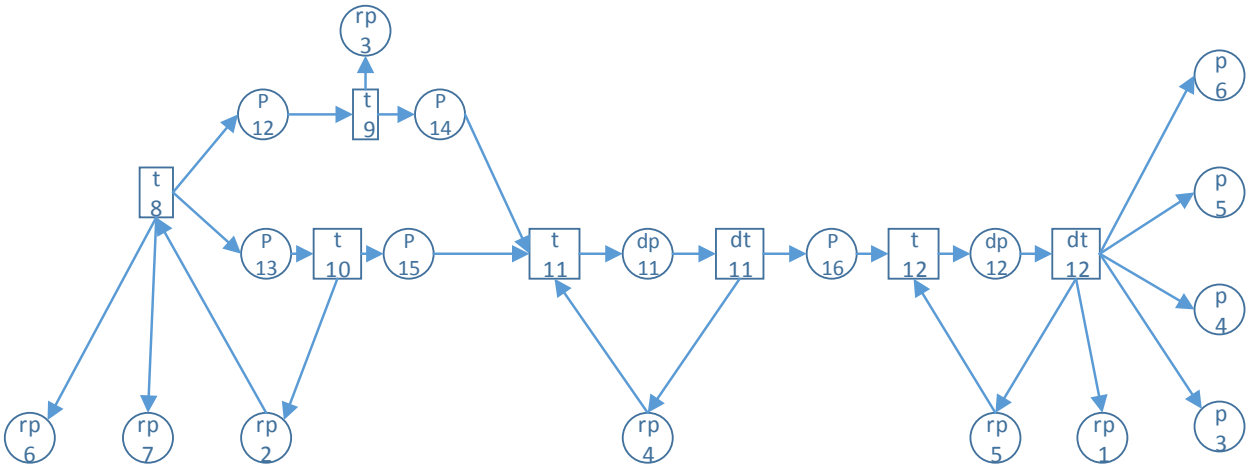

Figure 4.4 – Phase 4: Session Follow-Up

Table 4.7 - Transitions in Phase 4: Session Follow-Up

Transition Number	Process Name	Input Places	Output Places
Transition 9	Specialist Follow-Up	p12	p14, rp3
Transition 10	Patient Follow-Up	p13	p15, rp2
Transition 11	Follow-Up Processing	p14, p15, rp4	dp11
Dummy Transition 11	Finish Follow-Up Processing	dp11	p16, rp4
Transition 12	Check-Out	p16, rp5	dp12
Dummy Transition 12	Finish Check-Out	dp12	p3, p4, p5, p6, rp1, rp5

Table 4.8 - Places Introduced in Phase 4: Session Follow-Up

Place Number	Place Name
Place 12	Session Done (Hub)
Place 13	Session Done (Spoke)
Place 14	Specialist Follow-Up Complete
Place 15	Patient Follow-Up Complete
Dummy Place 11	Follow-Up Processing in Progress
Place 16	Specialist Follow-Up Sent
Dummy Place 12	Check-Out in Progress

In all, the model has 29 places and 18 transitions, including dummies and resources.

4.1.5 – Initial Marking

The initial marking used in later analysis represents a fairly typical situation for a telemental health system just starting up. The discrete-event simulation in Chapter 2 revealed that a telemedicine system with one provider working at full utilization can handle approximately 25 patients a week. Consequently, the initial marking for the Petri net model includes 25 tokens in the patient resource place, and 25 tokens in the "referrals received" place. Additionally, it places one token in each of the other six resource places, indicating one available nurse, specialist, and so on. No other places are marked. That said, this marking contains the tokens needed to eventually fire every transition at least 25 times.

4.2 – Modeling Asynchronous Telemedicine Systems

Asynchronous telemental health systems feature all of the same basic components that synchronous systems have but arrange them slightly differently. In contrast to the high level of parallelism *within* individual patient cases found in synchronous systems, asynchronous systems have lots of parallelism *between* cases. In other words, synchronous systems have multiple actors working on one patient at the same time, while asynchronous systems have those same actors working on different cases at the same time. As previously discussed, this provides a high degree of flexibility and allows specialists to best use their time.

The consequence of this setup is a much more linear processing scheme for asynchronous systems, as well as more subdivisions of the process. Activity on a patient's case passes back and forth between hub and spoke sites and never occurs in both at the same time. Session preparation

and session follow-up both have subcomponents that occur exclusively at the hub and spoke. Moreover, boundaries between these subdivisions can imply some delay time. Unlike in the synchronous system, where all activities after enrollment happen as quickly in succession as possible, there may be a delay of a day or two between phases. As long as response times are reasonably timely, this freedom to do actions when convenient can improve the utilization of employees.

The following sections describe the phases of the asynchronous system in-depth. Since much of the net is similar to the synchronous model, much of the explanation will be shortened for the sake of brevity.

4.2.1 – Patient Enrollment

The asynchronous system begins with patient enrollment. This section of the net is actually identical to its counterpart in the synchronous system, with the exception of the output of dummy transition 2. A full description of that system can be found on page 59. Regardless, figure 4.5 below details this part of the process.

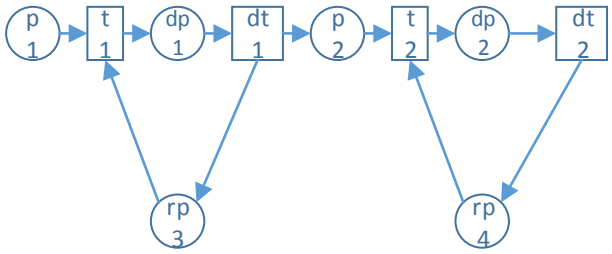

Figure 4.5 – Phase 1: Patient Enrollment – Asynchronous

4.2.2 – Hub-Side Session Preparation

Since the model is asynchronous, the process requires action by the hub administrator in *resource place 4* to gather information. They pass this off to the specialist in *resource place 3*. They generate a set of questions and hand them back to the hub administrator to send to the spoke site. Figure 4.6 shows these processes.

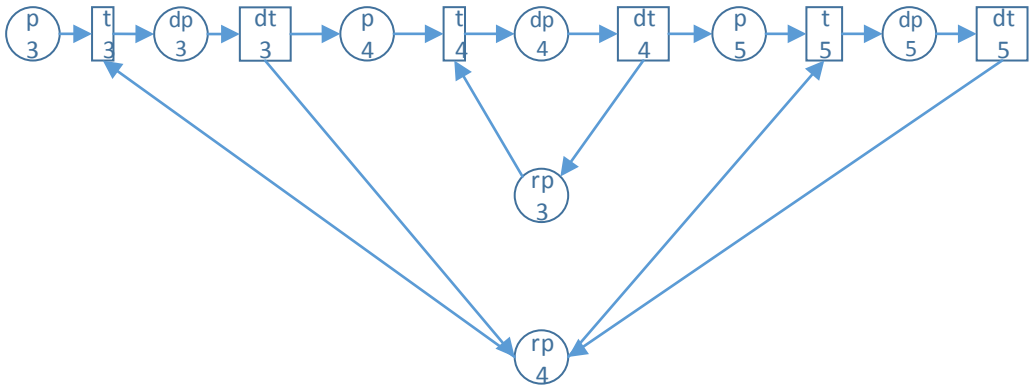

Figure 4.6 – Phase 2a: Hub-Side Session Preparation

4.2.3 – The Day of the Therapy Session

When the patient (*resource place 1*) checks in, the spoke administrator in *resource place 5* and the spoke technology specialist in *resource place 6* both have work to do checking in the patient and setting up the room. They can complete these in parallel. When both are ready and the nurse in *resource place 2* is available, the session (*transition 8*) can begin. The nurse reads the patient the questions prepared by the specialist and the patient answers on camera. Afterward, the nurse must complete follow-up work. The spoke administrator then checks out the patient and sends the recorded responses back to the hub site. Figure 4.7 on the next page shows this.

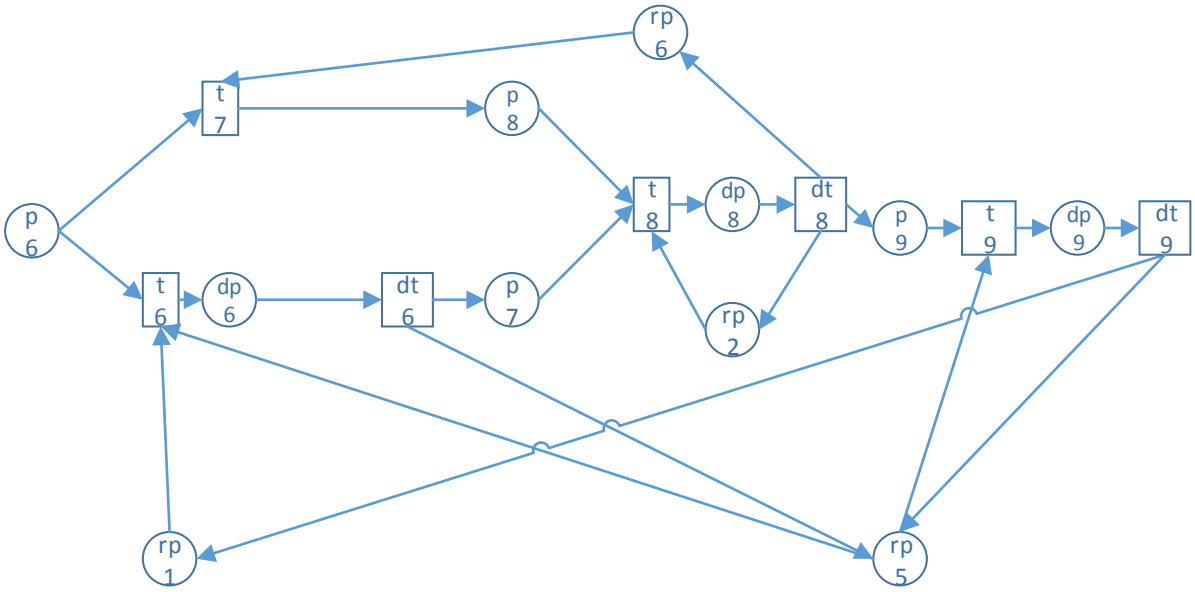

Figure 4.7 – Phase 2b: Spoke-Side Preparation and Phase 3: Therapy Session

4.2.4 – Hub-Side Session Follow-Up

After receiving patient responses, the specialist in *resource place 3* must examine those responses and recommend treatment. The specialist passes this recommendation off to the hub administrator in *resource place 4*, and they send those responses back to the spoke site. Figure 4.8 captures this part of the process.

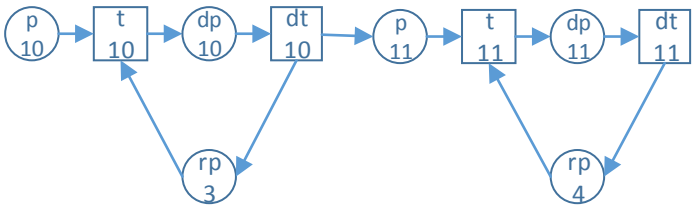

Figure 4.8 – Phase 4a: Hub-Side Session Follow-Up

4.2.5 – Spoke-Side Session Follow-Up

The process ends with the patient (*resource place 1*) checking in once again with the spoke administrator (*resource place 5*). The patient then meets with the nurse briefly to receive instructions or medication and checks out again with the spoke administrator. Figure 4.9 depicts the final leg of the process.

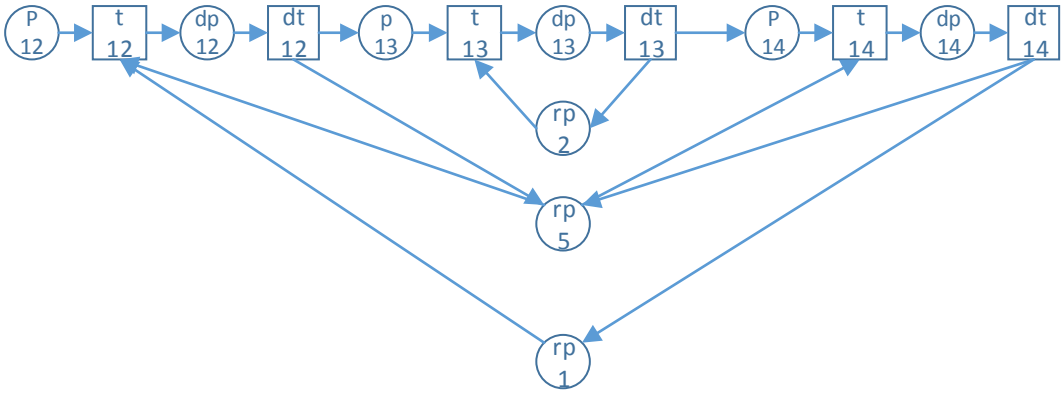

Figure 4.9 – Phase 4b: Spoke-Side Session Follow-Up

In all, the model has 33 places and 27 transitions, including dummies and resources.

4.2.6 – Initial Marking

Just like the synchronous model, the asynchronous Petri net representation of a telemedicine system has an initial marking that represents the start of the system. *Place 1* and *resource place 1* have 25 tokens each that represent patients, while each other resource place has one token to symbolize a single staff member.

4.3 – Modeling the Incidence Matrix

The state transition equation provides an extremely useful tool for analyzing the behavior of Petri nets, as discussed in the previous chapter. However, using this tool requires an incidence matrix. This matrix happens to have a rank of 17 for the synchronous model. The reachability analysis in the next chapter will use this fact. The incidence matrices for both synchronous and asynchronous telemedicine systems can be found on the next two pages.

	p1	dp1	p2	dp2	p3	dp3	p4	dp4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14	p15	dp11	p16	dp12	rp1	rp2	rp3	rp4	rp5	rp6	rp7
t1	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0
dt1	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
t2	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0
dt2	0	0	0	-1	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
t3	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	-1	0	0
dt3	0	0	0	0	0	-1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
t4	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0
dt4	0	0	0	0	0	0	0	-1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
t5	0	0	0	0	0	0	0	0	-1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0
t6	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	-1
t7	0	0	0	0	0	0	0	0	0	0	-1	0	1	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0
t8	0	0	0	0	0	0	0	0	0	0	0	-1	-1	-1	-1	1	1	0	0	0	0	0	0	-1	0	0	0	1	1
t9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	1	0	0	0	0	0	0	1	0	0	0	0
t10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	1	0	0	0	0	1	0	0	0	0	0
t11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	-1	1	0	0	0	0	0	-1	0	0	0
dt11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	1	0	0	0
t12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	-1	0	0
dt12	0	0	0	0	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	1	0	0

Figure 4.10 – Incidence Matrix for the Synchronous Model

	p1	dp1	p2	dp2	p3	dp3	p4	dp4	p5	dp5	p6	dp6	p7	p8	dp8	p9	dp9	p10	dp10	p11	dp11	p12	dp12	p13	dp13	p 14	dp14	rp1	rp2	rp3	rp4	rp5	rp6
t1	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0
dt1	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
t2	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0
dt2	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
t3	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0
dt3	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
t4	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0
dt4	0	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
t5	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0
dt5	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
t6	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	-1	0
dt6	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
t7	0	0	0	0	0	0	0	0	0	0	-1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1
t8	0	0	0	0	0	0	0	0	0	0	0	0	-1	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0
dt8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
t9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0
dt9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0
t10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	-1	0	0	0
dt10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	1	0	0	0
t11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	-1	0	0
dt11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	1	0	0
t12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	-1	0	0	0	-1	0
dt12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	1	0
t13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	-1	0	0	0
dt13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	1	0	0	0	0	0
t14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	-1	0
dt14	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	1	0

Figure 4.11- Incidence Matrix for the Asynchronous Model

Chapter 5 - Analyzing Telemedicine Systems

While simply modeling telemental health systems with Petri nets has some value in and of itself, the true power of the tool comes from its analysis. This chapter applies the analytic techniques discussed in Chapter 3 to the model built in Chapter 4.

5.1 – Reachability

Reachability in practice tests whether undesirable system states can be reached from a given starting state. These states could range from failure modes, to potential deadlocks, to states that simply cause personal or political problems. The following section captures some of the most glaring possible issues in telemedicine systems and their corresponding Petri net marking representations.

5.1.1 – Analysis Goals and Undesirable States

Reachability analysis for Petri nets can have one of two major roles. The first is model verification. Most telemedicine systems will not spontaneously make their doctors disappear into thin air, so a Petri net modeling telemedicine systems should not let that happen. Confirming this helps confirm the validity of the model and usefulness of other results.

The second and far more interesting goal of reachability analysis is to predict possible system behavior. States where employees have massive piles of work to accomplish may be very real but undesirable. Testing systems for this can highlight problems and potentially reveal systematic changes to stop them.

Telemedicine systems feature a number of these undesirable states. The list below describes their characteristics and what they look like when modeled in a Petri net.

1. An empty net, where all tokens somehow get consumed. More generically, any situation where the net is dead.
2. Growing resources. The number of patients and staff available should be bounded by the initial marking. Markings where more than these numbers become available imply a problem with the model.
3. Situations where all patients and providers are occupied. This state actually is not terribly undesirable, as it signals a high degree of utilization and parallelism.
4. Partial deadlock situations where one of the seven resource places loses all its tokens and cannot regain them. Alternatively, markings where zero tokens exist in the resource places and dummy places.
5. Conflict situations where an employee could be needed at several transitions at once.
6. Situations where many tokens reside in phase 1 of the net after a large number of steps. This is essentially the fairness property discussed earlier.

Variations on reachability analysis or coverability analysis can test the model for the existence of all these states. The next section explains how to execute this analysis.

5.1.2 – Implementing the Binary Integer Programming Model

The BIP model discussed in the previous chapter proves very useful for determining the reachability of the undesirable states considered above. The application of the BIP model in the paper uses a 500-step search depth: enough steps to allow transitions to fire modeling 25 patients enrolling, attending sessions, and checking out. The model opts for a very simple objective function that accepts the first feasible solution found.

Implemented in optimization programming language (OPL) using IBM's CPLEX solver and a base search depth of 500 steps, the model can detect infeasible solutions in as little as 30 seconds, although it can take much longer for more complex markings. The search depth of 500 steps provides ample time for the model to find a feasible solution. Thanks to the model's definition of "step" that allows multiple enabled transitions to fire in one step, this depth provides more than enough time for each transition in the model to fire 25 times, effectively providing all 25 patients the opportunity to complete a session from enrollment to check-out.

5.1.3 – BIP Model Results

This tool provides a convenient way to test several of the undesirable markings for reachability and coverability. The results can be found below:

1. The empty net, where no tokens exist in any place, is unreachable within 500 steps. This is a pure reachability problem where the model seeks a value of 0 tokens in each place. Application of the equations disproving reachability in Murata's work, discussed in chapter three, confirms that the empty net is truly unreachable from the initial marking (Murata, 1977).
2. Growing resource situations, where more tokens appear in a place in some marking than in the initial marking, are all unreachable within 500 steps. This is a coverability problem, where the resource under consideration in the target marking has a value of its starting value plus one (this equals two in most cases) and all other places have a value of zero. The zero allows any value of the variable to occur in the target marking, since any feasible value should be greater than or equal to zero.
3. Situations where all employees and patients become occupied are reachable. This problem is a partial reachability problem where the model only checks if the resource

places have exactly zero tokens. It accomplishes this by selectively iterating the second constraint representing the state transition equation.

4. Partial deadlock situations where a given resource disappears, that is, where no tokens exist in their resource place or any of the places indicating that they are busy, are not reachable within 500 steps. This is a partial reachability problem where the model checks if the resource place and all the places where that person is occupied have exactly zero tokens. Unfortunately, the models used cannot definitively disprove reachability sometime after 500 steps.
5. Situations where conflicts over resources exist are reachable. Specifically, with the initial marking it is possible to reach situations where the specialist, hub site administrator, and spoke site administrator must choose between doing one of several tasks and delaying the others. Patients, nurses, and technology specialists do not have any conflict situations in the model. These are partial coverability problems, where the model checks if all transitions that have the resource place for specialists and administrators as inputs are enabled.
6. Situations where a large number of tokens could remain at the start of the net after quite some time are reachable within 500 steps. The model could spend all 500 steps moving one token into the repeating portion of the net and simply running that one patient over and over again. The permissive nature of Petri nets makes this a reality, although this problem is not likely to occur in practice.

The time needed to evaluate certain markings for reachability can take over two hours in this model. Thankfully, Bourdeaud'huy et. al. (2006) noted a useful change to the model that does not significantly degrade performance. Specifically, the authors of the model propose

relaxing the binary constraint on the decision variables. This relaxation to a linear-programming model opens the door to comparatively rapid analysis with the simplex method. As with all linear relaxations of integer problems, however, the nearest integer solution to a linear program result may not be feasible. Regardless, an LP-relaxation still may find a result in the ballpark of a feasible solution.

5.2 – Boundedness

Ensuring Petri net boundedness serves to both validate the model and diagnose potential failure modes. For places that represent some finite quantity (such as the number of nurses), unbounded token growth would indicate a problem with the model. For places that represent a work queue, token proliferation represents an undesirable workload and boundedness ensures that this workload remains manageable in practice.

Determining the boundedness of a net typically involves generating the reachability tree of the net (Murata, 1989). As discussed already, this cannot be quickly accomplished for large nets, even with software. The net generated in the previous chapter is far too large for software like PIPE to analyze (Bonet et al., 2007; Dingle et al., 2009). However, with the smaller instance of the net found via the net transformations described below, PIPE can easily solve for boundedness.

5.2.1 – Net Transformations

The synchronous individual can be reduced according to the model described in the following steps. These steps correspond to transformations described in literature (Hyung et al., 1987) and explained in chapter three.

1. Remove place 2, referral approved, and consolidate dummy transition 1 and transition 2 into one transition. This new “macrotransition” takes dummy place 1 and the hub administrator as input and has the specialist and dummy place 1 as output. Remove place 7, patient data acquired, and consolidate dummy transition 4 and transition 7 as above. Also remove place 16, specialist follow-up sent, and consolidate dummy transition 11 and transition 12 as before. Figure 5.1, below, shows the net before these changes. Consolidated transitions and deleted places appear in circles.

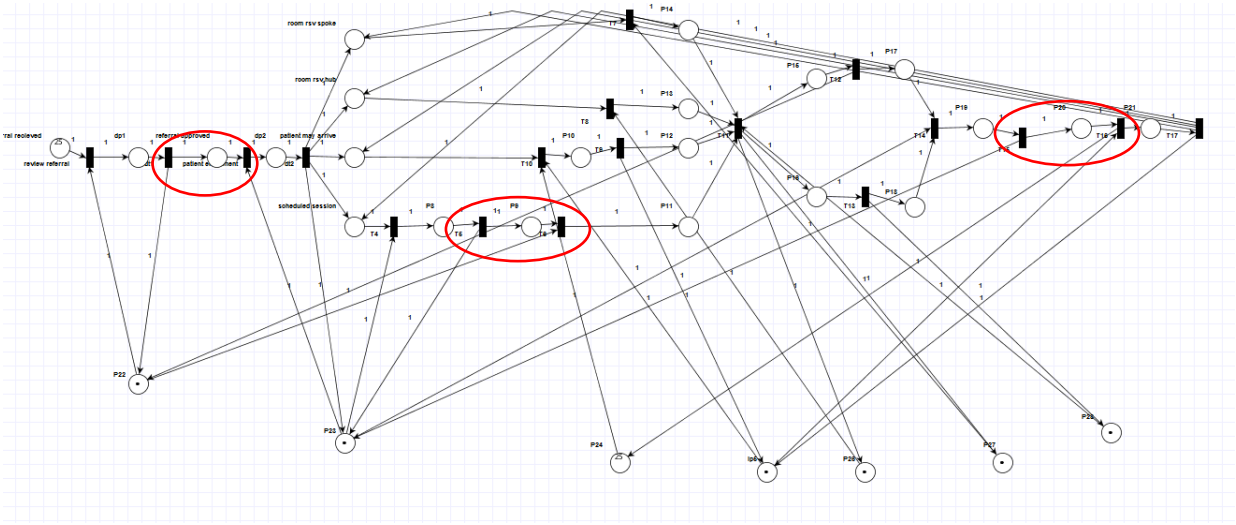

Figure 5.1 – Synchronous Petri net with several potential reductions denoted in circles

2. Remove places between two transitions where the first seizes a resource and the next immediately releases it. In this instance, there exists a path from the first transition to the second and a path from the second transition to the first. Defining the reducible subnet to include just the non-resource place and the two transitions it lies between allows for reductions similar to those above. These changes result in the removal of the place and the consolidation of the two transitions on either side. Additionally, the resource place remains connected and marked to the new “macrotransition” via a self-loop. This

transformation works for dummy place 1, dummy place 2, dummy place 3, dummy place 4, dummy place 11, and dummy place 12. Figure 5.2, below, shows these changes.

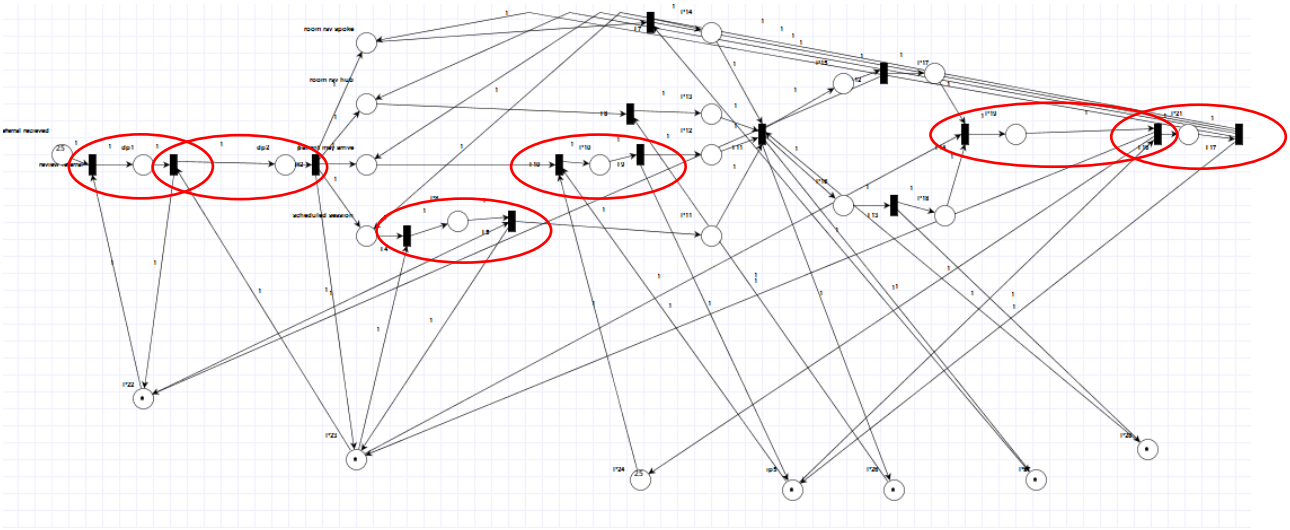

Figure 5.2 - Synchronous Petri net with three subnets consolidated

3. The net generated from the previous reductions invites several more promising changes. With the exception of resource places, the final transition in the net before it loops back to the beginning of the session preparation phase has two input places that meet the requirements for removal stated above. This change can be seen in figure 5.3, on the next page. At this point, the net becomes abstract enough to make interpretation of markings difficult.

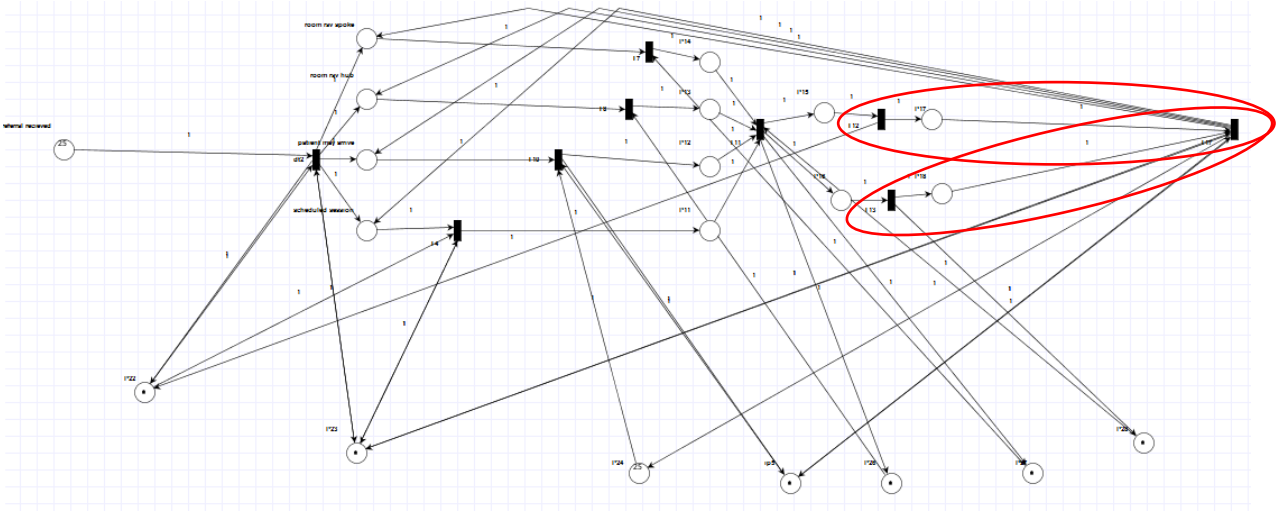

Figure 5.3- Synchronous Petri net with all dummy places eliminated

4. Step five leaves two places that each have one input from the transition loosely representing the session and one output to the final transition. These places can be merged together, representing a “session complete” state in a very general sense. Figure 5.4 shows this change and figure 5.5 shows the Petri net that results from these changes.

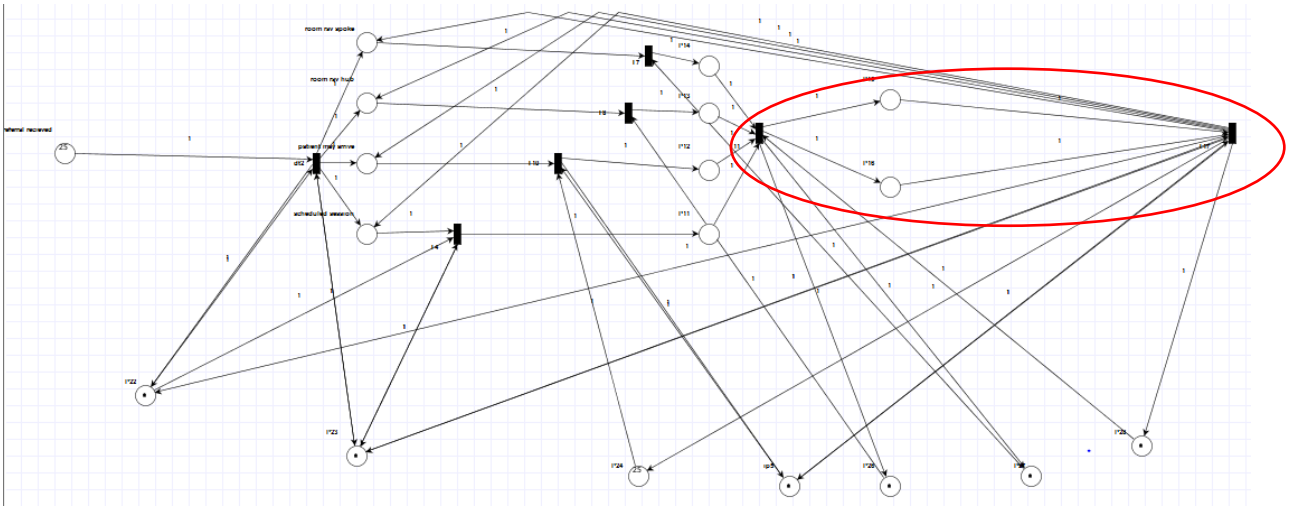

Figure 5.4 - Synchronous Petri net with additional places removed

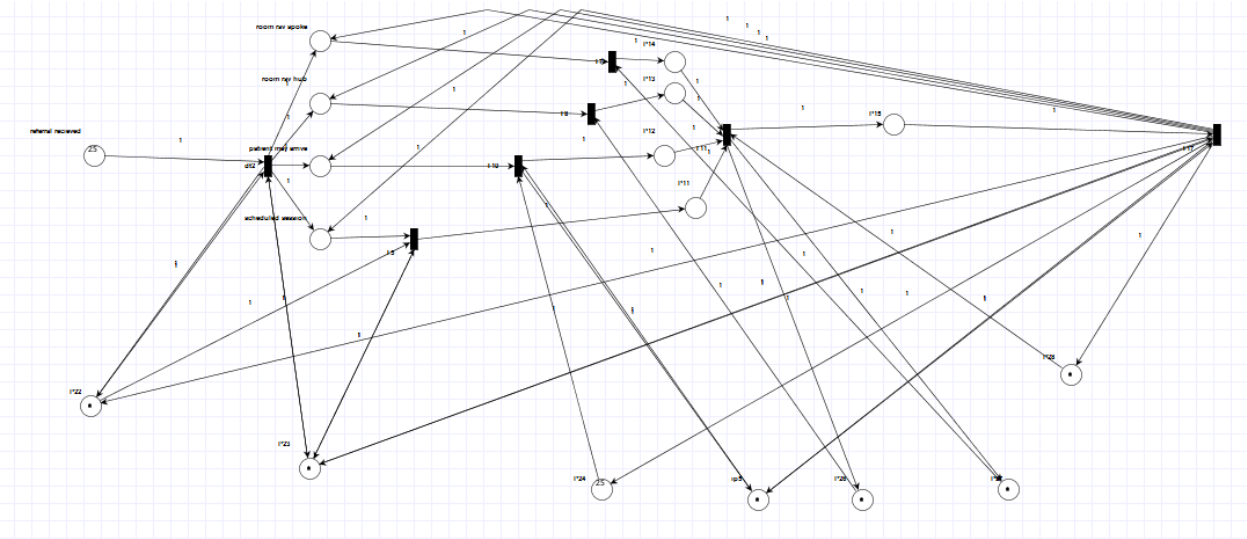

Figure 5.5 – Synchronous Petri Net with parallel places consolidated

While more reductions can be made, those listed above are all that prove necessary for PIPE to perform a boundedness analysis.

5.2.2 – Software Analysis Results

PIPE's analysis determined that the reduced net is bounded. Since the transformations used preserve boundedness, the full net is also bounded. This means that the process is limited and, given a fixed demand for telemedicine services, cannot generate an infinite number of patient cases, paperwork trails, or appointments.

5.2.3 – Petri Net Liveness

The set of transformations used to reduce the model for analysis of boundedness also preserve the property of deadlock-freeness. While a fairly low level of liveness (Murata, 1989), deadlock-freeness nevertheless guarantees that the telemedicine system modeled by the net never comes to a complete and unresolvable standstill.

Chapter 6 - Conclusion

Telemedicine, particularly telemental health, can provide solutions to some of the most pressing needs of today's world. By eliminating geographical barriers that would otherwise prevent patients from accessing specialists, doctors with valuable insight can extend their effective range to use their education in the best way possible. Moreover, telemental health can potentially provide care that is better than available face-to-face treatment in certain situations.

Telemental health faces some challenges, however. Further studies of the clinical ramifications of telemental health, as well as the development of effective measurements to gauge system quality, must occur before widespread adoption. Other hurdles relate to system implementation involve ensuring actors become familiar with the system's complexities. Since telemental health can come in both synchronous and asynchronous varieties, clearly communicating the interactions and dependencies of the process becomes essential for effective operation. Both types of system configuration have their advantages and disadvantages, with asynchronous telemedicine costing a little less to setup but injecting more complexity and unconventional wisdom into the system.

Despite the differences between synchronous and asynchronous modes, most telemedicine systems have the same basic components. The major actors include patients, specialists, nurses/general practitioners, administrators at the spoke and hub site, and technology specialists at the hub and spoke. Systems go through four distinct phases: a transient enrollment phase, session preparation, therapy, and session follow-up.

Telemedicine systems lend themselves well to discrete-event simulation. A simulation implemented in chapter one indicated that a basic system of similar structure to that of the New Mexico VA Hospital's telemental health system, with one of each major employee (that is, one

specialist, one nurse, one set of administrators, and one set of technology specialists) can serve about 25 patients per week before the specialist has no time remaining for other tasks. While useful for analyzing specific system configurations with given demands and service rates, discrete-event simulation provides little insight into theoretic system properties or possible long-term behavior of telemedicine systems.

Petri nets, on the other hand, can provide useful insight into the more abstract behavior of systems. This tool flaunts its computer science pedigree with its ability to model concurrency, conflict, and preconditions in complex systems with ease. Petri nets boast a wide range of theoretical properties and excel at modeling long-term behavior and potential failure modes. In particular, reachability, coverability, boundedness, and liveness stand out for their potential use in the analysis of telemedicine systems. Analyzing Petri nets for these properties proves straightforward as well. Binary integer programming provides a powerful tool for modeling and solving the reachability and coverability problem in nets, while boundedness and liveness can be determined using a combination of property-preserving net transformations and open-source software.

Telemental health systems can be easily modeled as Petri nets. Using transitions to model actions in the system, places to track the progress of a patient's appointment, and other places to model the availability of the different system actors results in a useful model that captures many of the conflicts inherent in telemedicine systems. The model is adaptable; minor changes in dependencies, extra steps, or varying numbers of employees can all be accommodated with only minor changes to the model.

Reachability analysis can provide very useful results for the system. Certain markings can be identified that represent system failure modes. The binary integer programming model

accepts these markings as input, as well as the incidence matrix of the Petri net representing the system. A feasible model indicates that a state is reachable within a certain number of firings. Analysis indicated that an empty net, the appearance of extra employees, and partial deadlocks where employees disappear are all unreachable within 500 steps. Application of more traditional state equations confirmed this for any arbitrary number of steps for the first two assertions. Situations of full utilization of all employees, conflict over employee time, and preferential treatment of one case are all reachable.

Testing the model for boundedness both confirms that the Petri net matches predicted system behavior and can diagnose potential exploding workloads. While raw Petri net interpretations of telemedicine systems prove too large for software such as PIPE to process, boundedness-and-liveness-preserving net reductions can alleviate this problem. Software analysis of a reduced version of a Petri net representing a synchronous telemedicine system revealed the model offered both boundedness and liveness. Consequently, telemedicine systems similar in design to the one described in chapter 4 should pose no problems relating to deadlocks or massively expanding workloads.

Continued applications of Petri nets in telemedicine stands as a rich area of further research. The simulation models discussed in chapter 1 of this paper can, with minor changes, offer perspective into the potential of group telemedicine for certain types of therapy. In group telemedicine, several patients attend a single group therapy session facilitated by a specialist at a remote location. While not appropriate for treating all conditions, this method can offer unique benefits to patients while also further extending the effective capacity of care providers. While promising, having multiple patients attend one session can further complicate the Petri net representation of the model and merits further examination with the tools used in this paper.

The Petri net tools used for the analysis in this work serve as the foundation for numerous variations on basic Petri nets. Time Petri nets, stochastic Petri nets, colored Petri nets, and other contemporary models increase the complexity of the model but may also help iron out assumptions made by the models in this paper (Girault & Valk, 2003)**.** Further analysis of telemedicine systems using these more advanced tools may provide meaningful results tailored for individual systems.

Moreover, S-invariants, T-invariants, and their application in finding structural properties of Petri nets (Murata, 1989) represent a substantial body of work in Petri net theory that may find use in telemedicine. While they can lack the accessibility and immediately intuitive appeal of other forms of analysis, these powerful properties may yield faster or stronger results than the analytic techniques presented in chapter 3.

In short, this thesis presents a collection of tools for system analysis using Petri nets and applies these tools to generalized telemental health systems of similar construction to a real-world organization. Reachability, boundedness, and liveness analysis all offer meaningful results for telemedicine system architects and can guide the formation of stable, efficient systems. Finally, these tools can serve as a starting point for further research into the use of Petri nets in telemedicine.

References

Bonet, P., Llado, C. M., Puigjaner, R., & Knottenbelt, W. J. (2007). PIPE v2.5: A Petri net tool for performance modeling. *23rd Latin American Conference on Informatics,* San Jose, Costa Rica.

Bouillon, Y. (2003). Model-based approach to control over concurrency in interactive CSCW applications: Application to telemedicine. *Annals of Telecommunications - Annales Des Télécommunications, 58*(5), 766-81.

Bourdeaud'huy, T., Hanafi, S., & Yim, P. (2007). Mathematical programming approach to the Petri nets reachability problem. *European Journal of Operational Research, 177*(1), 176-197. doi:10.1016/j.ejor.2005.10.060

Butler, T., & Yellowlees, P. (2012). Cost analysis of store-and-forward telepsychiatry as a consultation model for primary care. *Telemedicine and E-Health, 18*(1), 74-7. doi:10.1089/tmj.2011.0086

Cegarra-Navarro, J. -., Sanchez, A. L. G., & Cegarra, J. L. M. (2012). Creating patient e-knowledge for patients through telemedicine technologies. *Knowledge Management Research & Practice, 10*(2), 153-63. doi:10.1057/kmrp.2011.47

Chuanliang, X. (2010). Property analysis of Petri net reduction. *Second International Symposium on Networking and Network Security (ISNNS 2010), ,* 250-3.

Dingle, N. J., Knottenbelt, W. J., & Suto, T. (2009). PIPE2: A tool for the performance evaluation of generalised stochastic Petri nets. *Performance Evaluation Review, 36*(4), 34-9. doi:10.1145/1530873.1530881

Girault, C., & Valk, R. (2003). *Petri nets for systems engineering* Springer-Verlag Berlin Heidelberg.

Google. (2015). Map of CBOCs in New Mexico. Retrieved from https://www.google.com/maps/search/CBOC+in+New+Mexico/@34.1662325,-106.0260685,7z/data=!3m1!4b1

Grady, B., Myers, K. M., Nelson, E., Belz, N., Bennett, L., Carnahan, L., . . . Voyles, D. (2011). Evidence-based practice for telemental health. *Telemedicine and E-Health, 17*(2), 131-148. doi:10.1089/tmj.2010.0158

Henry, M., Layer, R., & Zaret, D. (2010). Coupled Petri nets for computer network risk analysis. *International Journal of Critical Infrastructure Protection, 3*(2), 67-75. doi:10.1016/j.ijcip.2010.05.002

Hyung, L. K., Favrel, J., & Baptiste, P. (1987). Generalized Petri net reduction method. *IEEE Transactions on Systems, Man and Cybernetics, 17*(2), 297-303. doi:10.1109/TSMC.1987.4309041

Institute of Medicine. (1996). In Field M. J. (Ed.), *Telemedicine: A guide to assessing telecommunications in health care*. Washington, DC: National Academy Press.

Jones, N. D., Landweber, L. H., & Lien, Y. E. (1977). Complexity of some problems in Petri nets. *Theoretical Computer Science, 4*(3), 277-99. doi:10.1016/0304-3975(77)90014-7

Khomenko, V., & Koutny, M. (2007). Verification of bounded Petri nets using integer programming. *Formal Methods in System Design, 30*(2), 143-76. doi:10.1007/s10703-006-0022-1

Krupinski, E., Dimmick, S., Grigsby, J., Mogel, G., Puskin, D., Speedie, S., . . . Yellowlees, P. (2006). Research recommendations for the American telemedicine association. *Telemedicine and E-Health, 12*(5), 579-589. doi:10.1089/tmj.2006.12.579

Lach, J. M., & Vazquez, R. M. (2004). Simulation model of the telemedicine program. *Proceedings of the 2004 Winter Simulation Conference,* , 2012-17.

Lasierra, N., Alesanco, A., Gilaberte, Y., Magallón, R., & García, J. (2012). Lessons learned after a three-year store and forward teledermatology experience using internet: Strengths and limitations. *International Journal of Medical Informatics, 81*(5), 332-43. doi:10.1016/j.ijmedinf.2012.02.008

Laurant, M., Reeves, D., Hermens, R., Braspenning, J., Grol, R., & Sibbald, B. (2005). Substitution of doctors by nurses in primary care. *Cochrane Database of Systematic Reviews,* (2), CD001271. doi:10.1002/14651858.CD001271.pub2

Li, D., Sun, X., Gao, J., Gu, S., & Zheng, X. (2011). Reachability determination in acyclic Petri nets by cell enumeration approach. *Automatica, 47*(9), 2094-8. doi:10.1016/j.automatica.2011.06.017

Locatis, C., & Ackerman, M. (2013). Three principles for determining the relevancy of store-and-forward and live interactive telemedicine: Reinterpreting two telemedicine research reviews and other research. *Telemedicine and E-Health, 19*(1), 19-23. doi:10.1089/tmj.2012.0063

Mayr, E. W. (1984). An algorithm for the general Petri net reachability problem. *SIAM Journal on Computing, 13*(3), 441-60. doi:10.1137/0213029

Medina-Marin, J., Seck-Tuoh-Mora, J. C., Hernandez-Romero, N., & Quezada-Quezada, J. C. (2013). Petri net reduction rules through incidence matrix operations. *25th European Modeling and Simulation Symposium, EMSS 2013, ,* 496-503.

Miyazawa, I., Tanaka, H., & Sekiguchi, T. (1996). Classification of solutions of matrix equation related to parallel structure of a Petri net. *Proceedings 1996 IEEE Conference on Emerging Technologies and Factory Automation.ETFA '96, ,* 446-52. doi:10.1109/ETFA.1996.573746

Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE, 77*(4), 541.

Murata, T. (1977). State equation, controllability, and maximal matchings of Petri nets. *IEEE Transactions on Automatic Control, 22*(3), 410-14. doi:10.1109/TAC.1977.1101509

Murata, T., & Koh, J. Y. (1980). Reduction and expansion of live and safe marked graphs. *IEEE Transactions on Circuits and Systems, 27*(1), 68-7. doi:10.1109/TCS.1980.1084711

Odor, A., Yellowlees, P., Hilty, D., Parish, M. B., Nafiz, N., & Iosif, A. (2011). PsychVACS: A system for asynchronous telepsychiatry. *Telemedicine and E-Health, 17*(4), 299-303. doi:10.1089/tmj.2010.0159

Peterson, J. L. (1981). *Petri net theory and the modeling of systems*. Englewood Cliffs, NJ: Prentice-Hall, Inc.

Rabinowitz, T., Brennan, D., Chumbler, N., Kobb, R., & Yellowlees, P. (2008). New directions for telemental health research. *Telemedicine and E-Health, 14*(9), 972-976. doi:10.1089/tmj.2008.0119

Sang Goo, L., Mun, S. K., Jha, P., Levine, B. A., & Ro, D. (2000). Telemedicine: Challenges and opportunities. *Journal of High Speed Networks, 9*(1), 15-30.

Spaulding, R. (2010). Cost savings of telemedicine utilization for child psychiatry in a rural Kansas community. *Telemedicine and E-Health, 16*(8), 867-871. doi:10.1089/tmj.2010.0054

Suzuki, I., & Murata, T. (1983). A method for stepwise refinement and abstraction of Petri nets. *Journal of Computer and System Sciences, 27*(1), 51-76. doi:10.1016/0022-0000(83)90029-6

Tarakci, H., Sharafali, M., & Ozdemir, Z. (2007). Optimal staffing policy and telemedicine. *Association for Information Systems - 13th Americas Conference on Information Systems, AMCIS 2007: Reaching New Heights, 1*, 226-232.

Thong, W. (2015). A survey of Petri net tools. *Lecture Notes in Electrical Engineering, 315*, 537-551. doi:10.1007/978-3-319-07674-4_51

von Wangenheim, C. G., von Wengenheim, A., Hauck, J. C., McCaffery, F., & Buglione, L. (2012). Tailoring software process capability/maturity models for telemedicine systems. *18th Americas Conference on Information Systems 2012, AMCIS 2012, 3*, 2472-2480.

Yang, S. K. (2004). A Petri net approach to remote diagnosis for failures of cardiac pacemakers. *Quality and Reliability Engineering International, 20*(8), 761-76. doi:10.1002/qre.599

Yellowlees, P., Odor, A., Parish, M., Iosif, A., Haught, K., & Hilty, D. (2010). A feasibility study of the use of asynchronous telepsychiatry for psychiatric consultations. *Psychiatric Services, 61*(8), 838-40. doi:10.1176/appi.ps.61.8.838

Yellowlees, P., Odor, A., Patrice, K., Parish, M. B., Nafiz, N., Iosif, A., & Hilty, D. (2011). Disruptive innovation: The future of healthcare? *Telemedicine and E-Health, 17*(3), 231-234. doi:10.1089/tmj.2010.0130

Yen, H. (2008). Concurrency, synchronization, and conflicts in Petri nets. *Lecture Notes in Computer Science, 5148*, 33-35. doi:10.1007/978-3-540-70844-5_4

Zhang, S., McClean, S. I., Jackson, D. E., Nugent, C., & Cleland, I. (2013). Patient satisfaction evaluation of telemedicine applications is not satisfactory. *XIII Mediterranean Conference Onf Medical and Biological Engineering and Computing, 41*, 1140. doi:10.1007/978-3-319-00846-2_282

Appendix A - Complete Representation of Petri Net Models

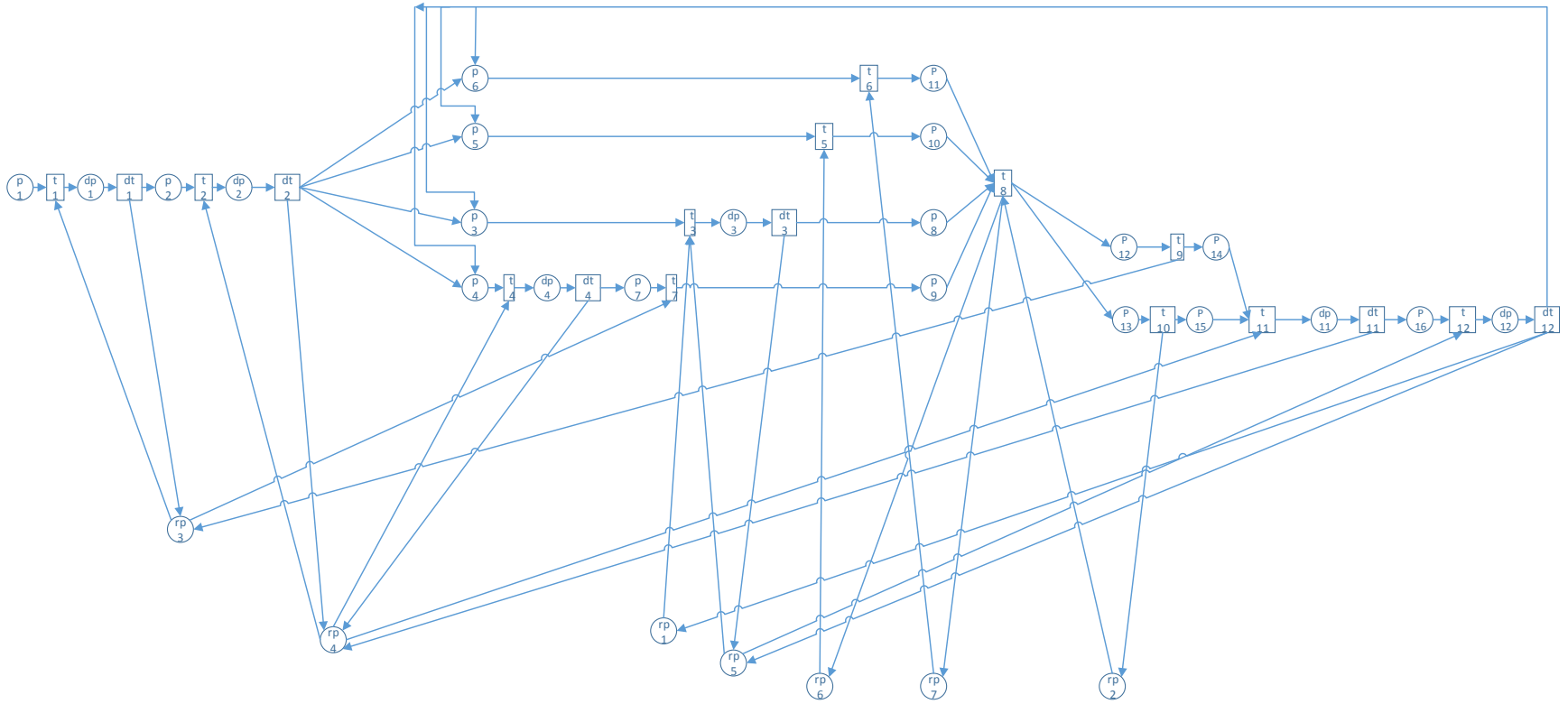

Figure 6.1- Complete Petri Net Representation of Synchronous Individual Telemental Health System

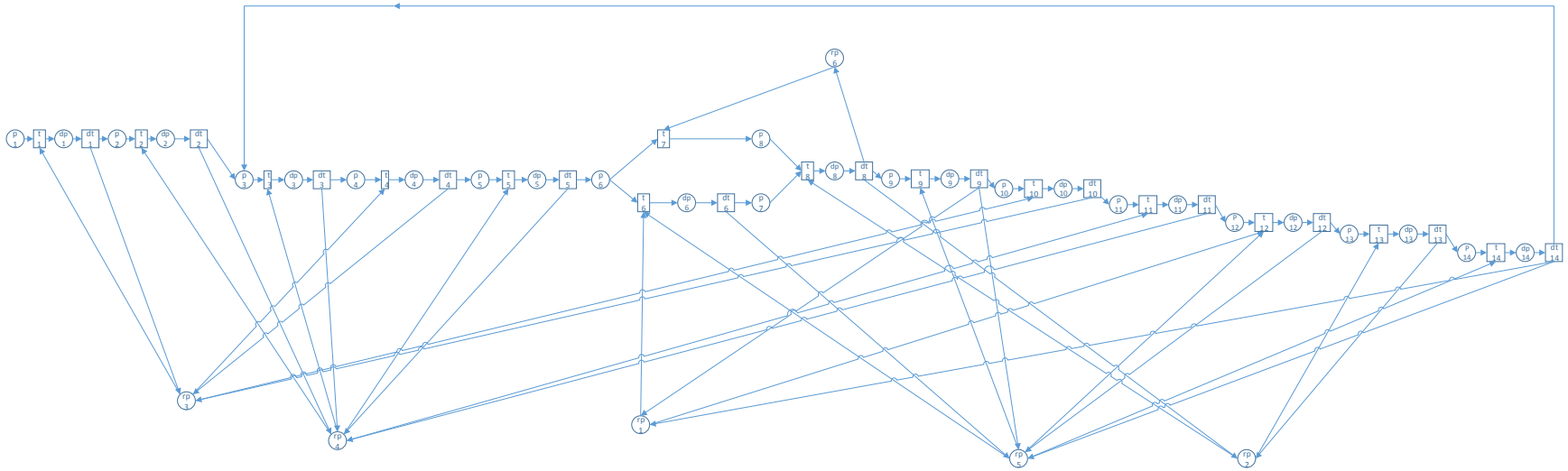

Figure 6.2 – Complete Petri Net Representation of Asynchronous Individual Telemental Health System

Table A.1 – List of All Transitions in Synchronous Telemental Health Petri Net Formulation

Transition Number	Process Name	Input Places	Output Places
Transition 1	Referral Review	p1, rp3	dp1
Dummy Transition 1	Finish Referral Review	dp1	p2, rp3
Transition 2	Patient Enrollment	p2, rp4	dp2
Dummy Transition 2	Finish Enrollment	dp2	p3, p4, p5, p6, rp4
Transition 3	Check-In	p3, rp1, rp5	dp3
Dummy Transition 3	Finish Check-In	dp3	p8, rp5
Transition 4	Specialist Priming	p4, rp4	dp4
Dummy Transition 4	Finish Priming	dp4	p7, rp4
Transition 5	AV Preparation (Hub)	p5, rp6	p10
Transition 6	AV Preparation (Spoke)	p6, rp7	p11
Transition 7	Specialist Preparation	p7, rp3	p9
Transition 8	Therapy Session	p8, p9, p10, p11, rp2	p12, p13, rp6, rp7
Transition 9	Specialist Follow-Up	p12	p14, rp3
Transition 10	Patient Follow-Up	p13	p15, rp2
Transition 11	Follow-Up Processing	p14, p15, rp4	dp11
Dummy Transition 11	Finish Follow-Up Processing	dp11	p16, rp4
Transition 12	Check-Out	p16, rp5	dp12
Dummy Transition 12	Finish Check-Out	dp12	p3, p4, p5, p6, rp1, rp5

Table A.2 – List of All Places in Synchronous Telemental Health Petri Net Formulation

Place Number	Place Name
Place 1	Referrals Received
Dummy Place 1	Referral Review in Progress
Place 2	Referral Approved
Dummy Place 2	Enrollment in Progress
Place 3	Patient May Arrive
Dummy Place 3	Check-In in Progress
Place 4	Session Scheduled
Dummy Place 4	Priming in Progress
Place 5	Hub Space Reserved
Place 6	Spoke Space Reserved
Place 7	Patient Data Acquired
Place 8	Patient Ready for Session
Place 9	Specialist Ready
Place 10	Hub Technology Ready
Place 11	Spoke Technology Ready
Place 12	Session Done (Hub)
Place 13	Session Done (Spoke)
Place 14	Specialist Follow-Up Complete
Place 15	Patient Follow-Up Complete
Dummy Place 11	Follow-Up Processing in Progress
Place 16	Specialist Follow-Up Sent
Dummy Place 12	Check-Out in Progress
Resource Place 1	Patient
Resource Place 2	Nurse
Resource Place 3	Specialist
Resource Place 4	Hub Site Administrator
Resource Place 5	Spoke Site Administrator
Resource Place 6	Hub Site Technology
Resource Place 7	Spoke Site Technology

Table A.3 – List of All Transitions in Asynchronous Telemental Health Petri Net Formulation

Transition Number	Process Name	Input Places	Output Places
Transition 1	Referral Review	p1, rp3	dp1
Dummy Transition 1	Finish Referral Review	dp1	p2, rp3
Transition 2	Patient Enrollment	p2, rp4	dp2
Dummy Transition 2	Finish Enrollment	dp2	p3, rp4
Transition 3	Specialist Priming	p3, rp4	dp3
Dummy Transition 3	Finish Priming	dp3	p4, rp4
Transition 4	Prepare Session	p4, rp3	dp4
Dummy Transition 4	Finish Preparation	dp4	p5, rp3
Transition 5	Send Recorded Session	p5, rp4	dp5
Dummy Transition 5	Recorded Session Sent	dp5	p6, rp4
Transition 6	Check-In	p6, rp1, rp5	dp6
Dummy Transition 6	Finish Check-In	dp6	p7, rp5
Transition 7	AV Preparation (Spoke)	p6, rp6	p8
Transition 8	Patient Views Session	p7, p8, rp2	dp8
Dummy Transition 8	Finish Session	dp8	p9, rp2, rp6
Transition 9	Send Patient Responses	p9, rp5	dp9
Dummy Transition 9	Finish Sending Responses	dp9	p10, rp1, rp5
Transition 10	Review Responses	p10, rp3	dp10
Dummy Transition 10	Finish Reviewing Responses	dp10	p11, rp3
Transition 11	Send Recommendations	p11, rp4	dp11
Dummy Transition 11	Finish Sending Recommendations	dp11	p12, rp4
Transition 12	Check-In (treatment)	p12, rp1, rp5	dp12
Dummy Transition 12	Finish Check-In (treatment)	dp12	p13, rp5
Transition 13	Administer Treatment	p13, rp2	dp13
Dummy Transition 13	Finish Administering Treatment	dp13	p14, rp2
Transition 14	Check-Out Patient	p14, rp5	dp14
Dummy Transition 14	Finish Check-Out	dp14	p3, rp1, rp5

Table A.4 – List of All Places in Asynchronous Telemental Health Petri Net Formulation

Place Number	Place Name
Place 1	Referrals Received
Dummy Place 1	Referral Review in Progress
Place 2	Referral Approved
Dummy Place 2	Enrollment in Progress
Place 3	Session Scheduled
Dummy Place 3	Patient Data Gathering in Progress
Place 4	Patient Data Gathered
Dummy Place 4	Session In Preparation
Place 5	Session Prepared
Dummy Place 5	Sending Session Recording
Place 6	Patient May Arrive
Dummy Place 6	Check-In in Progress
Place 7	Patient Ready
Place 8	AV Equipment Ready
Dummy Place 8	Session in Progress
Place 9	Session Complete
Dummy Place 9	Sending Patient Responses
Place 10	Patient Responses Sent
Dummy Place 10	Response Review in Progress
Place 11	Recommendations Made
Dummy Place 11	Sending Recommendations
Place 12	Recommendations Sent
Dummy Place 12	Check-In (treatment) in Progress
Place 13	Patient Ready for Treatment
Dummy Place 13	Treatment in Progress
Place 14	Treatment Complete
Dummy Place 14	Check-Out in Progress
Resource Place 1	Patient
Resource Place 2	Nurse
Resource Place 3	Specialist
Resource Place 4	Hub Site Administrator
Resource Place 5	Spoke Site Administrator
Resource Place 6	Spoke Site Technology

Appendix B - R Code to Disprove Reachability

```
incidence <- array(c(-
1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-1,0,0,0,0,0,-
1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,-
1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-1,0,0,0,0,0,0,-
1,1,0,1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,-
1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-1,0,0,0,-1,0,0,0,0,0,0,0,-
1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,-
1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-1,0,0,0,0,0,0,0,0,0,0,-
1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,-
1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-1,0,0,0,0,0,0,0,0,0,0,0,-
1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-1,0,0,0,0,0,0,0,0,0,0,0,-
1,0,1,0,0,0,0,0,0,0,0,0,0,0,-1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-
1,-1,-1,-1,1,1,0,0,0,0,0,0,-
1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-
1,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-
1,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-1,-
1,1,0,0,0,0,0,0,-1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-
1,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-
1,1,0,0,0,0,-1,0,0,0,0,0,0,1,0,1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-
1,1,0,0,0,1,0,0),dim = c(29,18))
incidence = t(incidence)
testmark1 =
c(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
initialmark =
c(25,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,25,1,1,1,1,1,1)
u <- ginv(incidence)%*%t(testmark1-initialmark)
y <- qr(incidence)
y$rank
#(make sure incidence matrix not transposed for these)
A11 <- incidence[1:17,1:(29-17)]
A12 <- incidence[1:17,13:29]
B <- cbind(diag(12),-t(A11)%*%ginv(t(A12)))
z <- t(ginv(B))%*%t(t(testmark1-initialmark))
```