PERFORMANCE MODELLING OF APPLICATIONS IN A SMART ENVIRONMENT

Submitted by

Xiao Chen

In Partial Fulfilment of the Requirements

For the Degree of Doctor of Philosophy

School of Computing Science

Newcastle University

Newcastle upon Tyne, United Kingdom

Autumn 2012

To My Parents

Acknowledgements

I would like to express the deepest appreciation to my supervisor, Dr Nigel Thomas, he continually and convincingly conveyed a spirit of adventure in regard to research and scholarship, and an excitement in regard to teaching. Without his guidance and persistent help this thesis would not have been possible.

I would like to sincerely thank my committee members, Professor Michael Harrison and Professor Aad van Moorsel, whose help and concern supported me through four years. I am deeply touched by their enthusiasm for work and will carry the same spirit in my future career.

In addition, I owe my gratitude to my parents who always give me understanding and encouragement whenever I need them.

Finally, special thanks must be given to Yishi Zhao, Huqiu Zhang, Wen Zeng, and some special friends for me.

Abstract

In today’s world, advanced computing technology has been widely used to improve our living conditions and facilitate people’s daily activities. *Smart environment* technology, including kinds of smart devices and intelligent systems, is now being researched to provide an advanced intelligent life, easy, comfortable environment. This thesis is aimed to investigate several related technologies corresponding to the design of a smart environment. Meanwhile, this thesis also explores different modelling approaches including formal methods and discrete event simulation.

The core contents of the thesis include performance evaluation of scheduling policies and capacity planning strategies. The main contribution is in developing a modelling approach for smart hospital environments. This thesis also provides valuable experience in the formal modelling and the simulation of large scale systems.

The chief findings are that the dynamic scheduling policy is proved to be the most efficient approach in the scheduling process; and a capacity scheme is also verified as the optimal scheme to obtain the high work efficiency under the condition of limited human resource.

The main methods used for the performance modelling are Performance Evaluation Process Algebra (PEPA) and discrete event simulation. A great deal of modelling tasks was completed with these methods. For the analysis, we adopt both numerical analysis based on PEPA models and statistical measurements in the simulation.

Contents

List of figures

List of tables

Chapter 1 Introduction

1.1 Background

How will the human world be in the future? How will be the living space in that future world? Without doubt, the world will be smarter. As described in some science fiction, smart technology will be widely used to improve human life; such as, smart navigation systems covering all zones in the city, intelligent home electronics undertaking some housework, even high intelligent robots living with us, and so on. Due to such advanced computing technology, citizens will be able to enjoy more comfort in their life. Smart systems and equipments could undertake, or assist in, more physical work, such as, cleaning, cooking, driving, or delivering, and so on. People only need to give commands via talking. All these aspects combine to construct a smart environment around people's living space, which marks a start of the smart age.

Regarding the above imagination, people do not just picture that as far in the future. Nowadays, urban performance gradually depends on the social infrastructure and quality and availability of knowledge communication ('intellectual and social capital'), rather than only referring to the endowment of hard infrastructure in a city ('physical capital'). According to this background, the concept of the "smart city" is introduced as "a strategic device to encompass modern urban production factors in a common framework and to highlight the growing importance of Information and Communication Technologies (ICTs), social and environmental capital in profiling the competitiveness of cities" [1]. Smart cities can be identified depending on the following six aspects: "a smart economy; smart mobility; a smart environment; smart people; smart living; and, finally, smart governance" [2]. These six aspects link with the neoclassical and traditional regional theories of urban growth and development [2]. During these six elements, smart environment is what is of most concern in this thesis as it highly depends on the modern computing technology. As depicted in the first paragraph, a smart environment can be simply described as a space filling up with all kinds of smart devices and being controlled via a smart system; nevertheless, it has a scientific definition.

A smart environment is, according to Mark Weiser, "a physical world that is richly and invisibly interwoven with sensors, actuators, displays, and computational elements, embedded seamlessly in the everyday objects of our lives, and connected through a continuous network" [3]. Cook and Das, define Smart environment as "a small world where different kinds of smart device are continuously working to make inhabitants' lives more comfortable" [4]. Three different types of smart environments have been defined by Poslad for systems services and devices: “virtual (or distributed) computing environments, physical environments and human environments, or a hybrid combination of these” [5]:

- “Virtual computing environments enable smart devices to access pertinent services anywhere and anytime.”
- “Physical environments may be embedded with a variety of smart devices of different types including tags, sensors and controllers and have different form factors ranging from nano to micro to macro sized.”
- “Human environments: humans, either individually or collectively, inherently form a smart environment for devices. However, humans may themselves be accompanied by smart devices such as mobile phones, use surface-mounted devices (wearable computing) and contain embedded devices (e.g., pacemakers to maintain a healthy heart operation).”

Generally, the core aim of smart environments lies in facilitating the experience of individuals in various environments by means of replacing the physical labour, hazardous work and repetitive tasks with automated agents [5]. As presented in [82], in a smart environment, a set of ubiquitous systems is embedded to provide relevant information to individuals in a physical environment to guide and support their experience of environment. In the real world, such smart environment can be applied in various physical environments: In hospitals, patients are directed to have a test, or a surgery, or to see a doctor, or attend a wellness check according on their appointments or decisions made by doctors or nurse. In an airport, passengers are notified about which queue to check-in, baggage screening and, passport control via the guiding information from the system. On a university or a college campus, students are directed to register in the school, collect their student card, pay fees, or meet their tutors at the beginning of academic year. In a sports centre or buildings, visitors are guided to a specific seat or office with guidance of the smart system. In addition, many other public areas can build smart environments to direct people to their destination or in a task. These environments will utilize public and personal displays and simple inexpensive sensor technologies to support activity, comfort and pleasure of its inhabitants by providing them with information they need, when and how they need it.

Smart environments are broadly classified to have the following features:

- Remote control of devices, like power line communication systems to control devices.
- Device Communication via middleware and Wireless network.
- Information Acquisition/Dissemination.
- Improved Services by Smart Devices.
- Predictive and Decision-Making ability.

According to these features, a large number of computing technologies must be used to promote a smart environment, such as, algorithm design, network modelling, wireless communication, multilayered software architecture, middleware, imaging recognition & processing, sensor design, parallel processing, operation system, and so on.

As part of smart environment, applicable scheduling algorithms must be used to guide individuals and allocate service resources to achieve a high efficiency. This thesis is inspired by a queuing model of patient flow in the hospital by Au-Yeung [6]. In a complex queuing network with limited serving ability, short queue length and fast response time are required most, but usually are seldom achieved. This is especially important in the healthcare systems providing critical patient care, such as NHS. Obviously, a healthcare system should treat its patients, especially those need critical care as soon as possible. In fact, long waiting times for treatment becomes a serious problem due to high patient demand and limited resource. Patient waiting time is now a hot public concern in the United Kingdom. According to a King’s Fund report, improving waiting times of the patients in Accident and Emergency department and the patients having cancer and cardiac are considered as two of the public’s top four priorities for public healthcare in the UK [7]. To solve these problems, performance targets for the NHS are introduced by the UK government, and some of these targets are based on patient waiting time and treatment time. From the Health Commission [8], 98% of patients should wait 4 hours or less in an Accident and Emergency department from arrival to admission, transfer or discharge. The report also indicates that some departments in the hospital require further efficiency improvement [9]. Thus, it is essential to promote more research in performance modelling based on the healthcare models in order to solve the potential scheduling and capacity bottleneck.

Smart environments could be an efficient way to improve the efficiency of healthcare system. Smart environments can improve the experience and collective behaviour of individuals within a physical space. For example in the hospital, smart environments can be a set of embedded systems, public displays, cameras, sensors, even individuals’ smart device, and so on. With this environment, each individual can be recognized by the system so that the smart system is able to provide useful information, such as navigation information or queuing time, to the patients via public displays or patients’ smart devices. The smart environment is able to schedule patients to different doctors or nurses so as to deploy limited resources efficiently. On the other hand, the smart system should have ability to carry out efficient capacity planning in terms of the available patient information. According to the amount of patients for different demands, the smart system can decide how many staffs are required for each position and can redeploy staffs based on the change of the patient information. Hence, to build a smart environment, one critical point lies in the design of the crucial algorithms and policies used in the core system, such as scheduling algorithms or capacity planning policies.

1.2 Research approach

The core aim of the thesis is to research the scheduling policies and capacity planning strategies based on the patient flow and work flow in the healthcare system, using some creative modelling techniques with different model languages and tools.

This thesis just follows the concept of the patient flow used in Au-yeung's work and produces a completely different patient flow scenario which is used to build specific scheduling models. Such patient flow defines a series of patient activities and behaviours of the doctors and other staff in the hospital. For capacity planning strategies, this thesis uses a scenario from a Rheumatology department of a major UK hospital trust, which includes a work flow and estimated statistical figures of that department.

The main modelling technique used in the thesis is *Performance Evaluation Process Algebra* (PEPA) with fluid flow analysis. According to the previous work of Harrison [10], PEPA based modelling techniques can be employed in analysing usability aspects of smart environments where many people are present before fielding the system. Moreover, the *fluid flow*, an approach based on *ordinary differential equations* (ODEs), derived from process algebraic specifications modelling both large groups of users and the environment, may provide useful analysis of the collective behaviour and the more quantitative aspects of usability of smart environments [10]. As Harrison's work proves that PEPA and its fluid flow approximation can be used for the models with complex structure and a large number of behaviours, this thesis will also apply PEPA as the main tool to create patient flow based scheduling and capacity models.

In addition, to augment the PEPA based analysis, *Chemical Model Definition Language* (CMDL) and *Discrete Event Simulation* are also used for model enhancement and validation. CMDL is usually used for chemistry modelling, but it is has an equivalent text semantics for PEPA, and is able to carry out the same fluid flow analysis in the same tool environment. The reason for using CMDL is that the analysis based on CMDL supports a functional rate. Direct analysis with PEPA cannot handle the analysis with function rates.

Discrete Event Simulation, a simulation based technique, is applied to create an equivalent model in terms of the PEPA model so as to validate the approximate fluid results. In the thesis, two kinds of simulation approaches are used to create models, which are Java based simulation with the tool Simjava and C++ based simulation with the tool OMNet++.

1.3 Research objective

The main objectives of this thesis in modelling technique exploration can be summarized by the following questions:

- Can PEPA model a system with complex structure and functionality, such as the structure of a complex action flow and the functionality in scheduling?
- Is the fluid flow approximation an accurate and reliable method to solve PEPA model and generate analysis?
- Is function rate successfully used to achieve the scheduling process in the model?
- Can the discrete event simulation validate the PEPA based analysis?

In addition to these objectives of modelling techniques, this thesis also has the objectives to compare the performance of different scheduling policies and investigate the optimal capacity plans and the efficient workflow based on the above mentioned modelling techniques.

1.4 Contributions of the thesis

The contribution of thesis is, on one hand, an earlier study of modelling a smart hospital environment using stochastic process algebra. This thesis investigates the performance of scheduling policies and capacity planning strategies under specified model scenario. The findings are valuable reference for the improvement of the smart environment. Such theoretical research provides useful guidance for the industry application. On the other hand, formal methods are successfully used for the complex and large scale models, which is a very challenging and creative job as just few researches has been done about this. Details of the contribution are specified as:

- **Performance evaluation of scheduling policies:** One key contribution of the thesis is to investigate the performance of several scheduling policies under a hospital scenario. These are the dynamic scheduling policy and the static scheduling policy. The findings prove that the dynamic scheduling policy has better performance than the static policy under certain conditions.

- **Performance evaluation of capacity plans:** Another contribution is to evaluate the performance of capacity plans. This part models the real world workflow based on a hospital department and explore different capacity plan via comparing the queue in the models of each plan. This part demonstrates the most efficient capacity plan suiting the scenario. These findings are useful to refine the workflow of the hospital department in order to reduce the waiting time of patients, and save on hospital resources.
- **Modelling with formal methods:** To complete the scheduling and capacity modelling, formal methods (PEPA and CMDL) and related numerical analysis (Fluid Flow Analysis) are adopted due to its formal and strict definition for models, and efficient and low cost analysis from models. The novel points are that this thesis uses the formal modelling techniques (mainly PEPA) for the large scale models. PEPA is used to model very complex system behaviours and carry out the analysis of system with huge amount states.
- **Functional Rates:** Another novel point is to extend the scope of formal modelling by using a functional rate. To complete the analysis of models with function rates, CMDL is used to undertake this mission, because it supports the specification of functional rates in the model and it also has a convenient and equivalent transformation from PEPA via the PEPA workbench.
- **Discrete event simulation:** Simulation is applied to validate the outcomes from formal methods. Moreover, simulation is able to accomplish all kinds of model scenario, though the time cost is quite high. The contributions of using simulation focus on two aspects: validation for the formal analysis; and the exploration of techniques for the equivalent model creation in terms of the formal models.

In this thesis, Chapter 3 has been published as two conference papers for UKPEW 2010 [77] and UKPEW 2011 [15] respectively. Chapter 4 has been published in the 1st Int'l Workshop on Healthcare Systems Engineering in the year 2011 [72].

1.5 Structure of the thesis

Chapter 1 introduces the background of the research topic and brief objectives of this project.

Chapter 2 is the literature review that simply introduces the related work, research scenario and specification of all used modelling techniques and terminologies.

Chapter 3 models the scheduling policies founded on a patient flow model and investigates its performance with PEPA based fluid flow analysis.

Chapter4 validates the previous scheduling models via Java based discrete event simulation.

Chapter 5 accomplishes a model based on the work flow of a hospital department with PEPA, and carries out capacity analysis.

Chapter 6 validates the previous capacity model via C++ based discrete event simulation.

Chapter 7 concludes the findings of the project, and critically assesses the contributions and weakness, and finally look far ahead into the future work.

Chapter 2 Literature review

Chapter 1 briefly introduces the background of this thesis. This chapter will provide more details of literature review related to the main topics of this thesis. This chapter includes two sections: the previous work referring to the topics discussed in the thesis and the introduction of main researching approaches applied for these topics.

2.1 Related work

This section aims to describe the related work of other researchers about the research scenario of this thesis. These related references can be grouped into three sub-sections by the contents: patient flow, scheduling and capacity.

2.1.1 Patient flow

Many previous researches have been done for the patient flow. Why patient flow is important in the healthcare system? As in the recent years, healthcare systems have been challenged to deliver high quality care with limited resource. One important reason is the delay in the care process. This delay mainly includes the queuing time in the waiting list outside the hospital and waiting time for the service in the hospital. The reasons causing such delay refer to two aspects: one is the patient scheduling process and the other is the resource allocation. Thus, the patient flow must be refined to improve the work efficiency and service quality by optimizing the scheduling process of the patients and the capacity planning for the limited healthcare resource.

As mentioned in the introduction chapter, this project is aimed to investigate the performance of the scheduling policies and capacity planning strategies based on the hospital patient flow model. The concept “patient flow” must be introduced briefly. The term flow describes the progressive movement of products, information and people through a sequence of processes. In healthcare, flow is the movement of patients, information or equipment between departments, staff groups or organisations as part of a patient's care pathway. Moreover, researches referring to the patient flow have been done in larger range. These researches usually focus on the factors which could affect the patient flow in the hospital, as well as the control of the waiting time via managing the patient flow.

An-Yeung’s work is aimed to investigate the efficient priority scheme that mainly refers to the priority of different patient groups [6]. In An-Yeung’s work, patients are grouped in terms of their physical status at first, and then the basic model is built based on the experience of each group of patients in the hospital, including patients’ behaviours and healthcare activities. This process is specified as “*patient flow*”, and its related model is the patient flow model. Figure 2.1 and Figure 2.2 show the top level patient flow model in An-Yeung’s work [6].

In Figure 2.1, each circle represents a service component with a queue. The circles with arrows, such as AR, AEU and Resusc, means these components are aggregated server components and include detailed sub-components. However, the circles without arrows, such as Reception and Nurse, are just simple service components without aggregated sub-components. The description of these components is specified in Figure 2.2.

Based on Figure 2.1, the patient flow can be explained as a series of activities in the hospital. For example, the walk-in patients must go to the reception when they arrive in the hospital; according to the decision of the receptionist, some of them go to the assessment room (AR), and others go to the AEU. Patients arrive in a department of the hospital and leave there after finishing related examination, and finally leave the hospital when they finish all activities. Such a series of activities of patients can be called the patient flow.

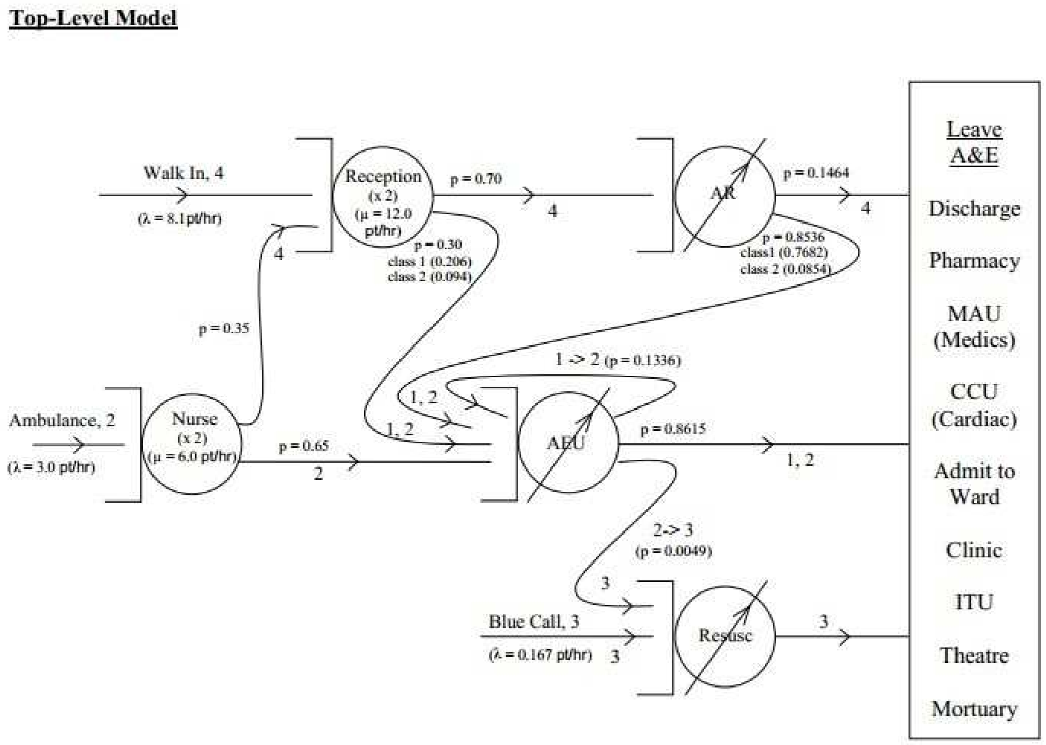


Figure 2. 1 Top-level of queuing network model of patient flow [6]

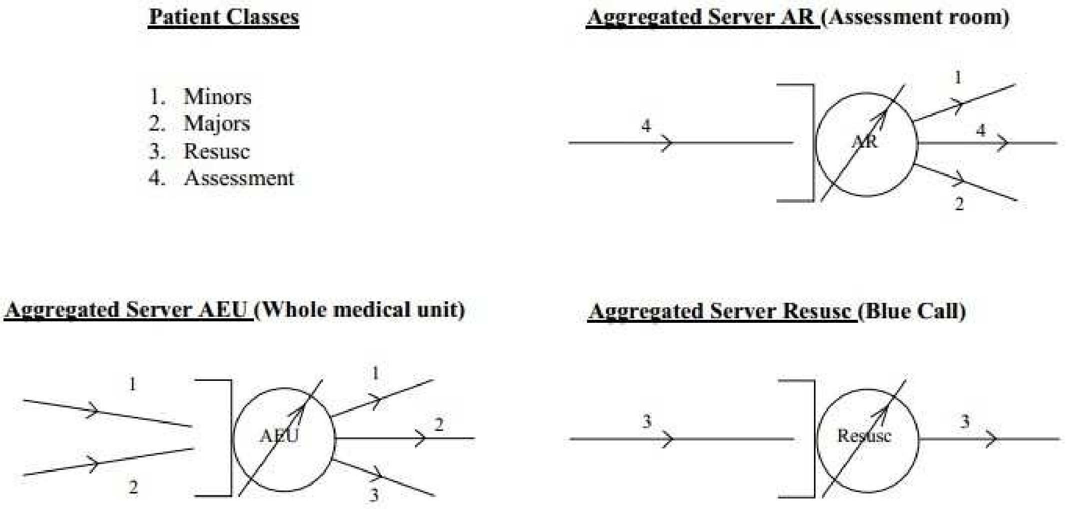

Patient Classes
1. Minors
2. Majors
3. Resusc
4. Assessment
Aggregated Server AR (Assessment room)
4
AR
1
4
2
Aggregated Server AEU (Whole medical unit)
1
2
AEU
1
2
3
Aggregated Server Resusc (Blue Call)
3
Resusc
3

Figure 2. 2 Components of top-level patient flow model [6]

This thesis uses the similar patient flow model as the start. As this thesis aims to explore the some modelling work with human factors, the patient flow model is a good choice which meets this requirement. Based on the patient flow model, different scheduling policies are embedded in the patient flow model in order to evaluate their performance by measuring the queue length of the patient flow model. Moreover, various capacity plans are also explored with the patient flow model, as well as the investigation of the hospital workflow. Patient flow model is the base of all modelling work in this thesis. This thesis uses the patient flow as a base to achieve the scheduling process and capacity planning. An-yeung's work just provides a prototype for the issues explored in this thesis, which uses this patient flow model structure and generates new patient flow models for investigating related scheduling policies and capacity plans.

Devaraj [88] aims to devote his research to examining the use of information technology among hospitals and explore how this information technology influences the patient care and hospital performance. The Theory of Swift Even Flow (TSEF) is used to present an operations management based perspective which is about the effect of information technology in hospital operations. In this research, the role of information technology has been examined based on the patient flow model in order to explore the consequences of information technology for the improvement of hospital efficiency and performance. This research uses the statistical data from over 500 U.S. hospitals which shows how the information technology is associated with the patient flow as well as the improved revenues of the hospital.

Wiler's research [89] compares a number of existed modelling methodologies for the emergency department crowding problem, such as formula based equations, regression models, time-series analyses, queuing theory-based models, and discrete event simulation, and then outlines the potential applications and limitations for each methodology about the usability in emergency department operations and crowding research. This research aims to explore each operational modelling technique and their current use in the emergency department setting with predicting patient load and crowding, and explore the potential future applications to emergency department operations research.

Klein's study [90] is to develop a simulation platform that runs in spreadsheet in order to measure the performance and utilization of the operational interventions of the patient flow in an emergency department. The spreadsheet based simulation has lower cost and powerful functionality and performance in contrast to the traditional discrete event simulation. Furthermore, spreadsheet simulation is able to carry out efficient development and management of simulations which generates results as reliable as those from traditional software.

Miró's research [22] investigated the potential factors affecting the patient flow, work efficiency and waiting status in the emergency department; and also explore the effects of reorganisation in the emergency department in terms of the previous indicators. This research uses a patient flow model to achieve all basic activities of the emergency department, and detect these factors in terms of the measurement of the patient flow model. They obtain the findings that internal factors also determine the effectiveness and overcrowding of the emergency department.

Coats [23] creates a mathematical model of the patient flow in an Accident and Emergency department based on the existing information. This research is aimed to assess the information required for a detailed model and then determine whether further data collection for a more detailed model is necessary. In this research, the scenario of Accident and Emergency department is represented through a patient flow model involving a series of activities in this department. The discrete event simulation based on the SIMUL8 software package is adopted to complete the patient flow model.

Côté [24] researches the patient flow through creating an Erlang-based stochastic model. Then the analytic results for the stochastic process are obtained from an effective tool. According to the research, the stochastic model can be developed easily for the patient flow.

Asplin [25] investigates an interesting issue of the US emergency departments (ED). This investigation focuses on the daily surge capacity and hospital wide patient flow research. The investigation uses a series of equations to model the dynamic nature of emergency department census. As a result, the short term forecasts of ED census are enabled; meanwhile, this research also illustrates the influence from the unexpected surges in patient demand.

Marshall's [26] research aims to explore the length of stay and flow of patients in the healthcare system. In this research, the length of stay is modelled with different probabilistic solutions, such as, Markov models, phase-type distributions, and conditional phase-type distributions. Furthermore, a mixed-exponential model and queuing simulation are also used for a compartmental model of the patient flow. According to Marshall's work, modelling patient flow is able to assist the understanding of activities in the healthcare system. Moreover, it is useful to use to a patient flow model to represent the functionality of the healthcare system.

Hung's [27] work is aimed at investigating the aspects of an emergency department that can be refined to improve patient flow, reduce waiting time, increase staff work efficiency, and then generate accurate predictions of patient flow and resource utilization. In this research, discrete event simulation is applied to model the patient flow. The simulation tool is the Arena discrete event simulation modelling software. The simulation results are validated by comparing with the actual patient flow data. From Hung's work, simulation based patient flow model can accurately represent the actual patient flow in hospital departments, and can generate reliable simulation results on various scenarios. Furthermore, changes and effects on the actual patient flow can be easily and effectively simulated via the patient flow model.

Medeiros [28] researches the patient flow of hospital emergency department in order to find solutions of problems about the growing demand for healthcare service. In the research, a patient flow model is built to represent a series of human activities and the whole process in the hospital emergency department. This research has two difficult key points in modelling of the patient flow: the transient system status and the dynamic arrival rates during the day. Simulation is used to model the patient flow to analyze the resource requirements, patient census and patient waiting time. According to this research, the major benefit of using simulation is that they are able to evaluate the performance of alternative model schemes.

Kolker [29] uses a discrete event simulation methodology to establish a model for the emergency department patient flow, and investigates the performance characteristics of the emergency department, such as percent of time on ambulance diversion, the number of queuing patients and the upper limits of patient length of stay. According to Kolker's specification, the main approaches currently used for the patient flow modelling are the queuing theory and the process model simulation. However, queuing theory has some limitations for the emergency department process. Thus, process model simulation approach should be more flexible and versatile in the modelling process [26]. This work obviously demonstrates the benefits of using a predictive simulation methodology.

Gunal [30] aims explore the performance of Accident and Emergency Department (A&E) of National Health Service in UK. This research applies simulation and utilizes real data from A&E to generate analysis of the healthcare service. It focuses on the aspects influencing the performance which are the multitasking behaviour and experience level of medical staff. The key problems discussed in the research are the total time in A&E, the hours waiting for emergency admission via A&E post decision to admit, and the availability of ward beds. These issues are the focus of all hospital performance related research. For the simulation, Micro Saint Sharp is used to develop a discrete event simulation model of the A&E department. The validation of the simulation is carried out via comparing with the real data from the A&E department. The novel aspect of the model is multitasking behaviour of medical staff.

Overall, in this thesis, patient flow can be used as carrier of different scheduling policies or capacity plans in order to carry out related analysis for their performance. Thus, the first important step is to choose the appropriate patient flow scheme for the following task.

2.1.2 Scheduling

Since we know the importance of refining the patient flow, the key problem is how to improve the efficiency of the patient flow. To improve the patient flow efficiency, scheduling processes must be considered by employing some efficient scheduling policies for the patient flow. Generally, scheduling is the process of deciding how to commit resources between a variety of possible tasks. In computing science, scheduling is the action allocating processes, data flows or threads to access system resources (e.g. communications bandwidth, processor time). Usually, scheduling can be divided into two types: static scheduling and dynamic scheduling. Ouelhadj [13] demonstrates that “dynamic scheduling is of great importance for the successful implementation of real world scheduling systems” in contrast to static scheduling. Static scheduling is usually a kind of scheduling list containing each possible order of process. Dynamic scheduling is a kind of scheduling algorithm. This dynamic scheduling algorithm is able to calculate the priorities; meanwhile, the system is executing. The goal of the dynamic scheduling is to alter conditions and then generate an optimal configuration in a self-sustained manner. In fact, defining an efficient dynamic policy is very difficult because of the difficulty and complicity of the problems. Dynamic scheduling techniques have been widely used in computing area especially for the networking. However, few studies have concerned promoting the dynamic scheduling in the flow control. Thereby, the half of the thesis explores the performance of different scheduling policies, including both dynamic scheduling and static scheduling policies, in the patient flow scheduling process.

Shah [91] is interested in design of low-complexity, myopic, distributed and stable scheduling algorithms that are used in the queuing network models. In his study, two kinds of models are considered to adopt the scheduling algorithm: a queuing network model which has randomly changing number of packets in the queues of wireless notes communicating through a shared medium; and a buffered circuit switched network model that is used for an optical core of future internet to capture the randomness in calls or flows in the network. Shah's work developed a stable scheduling algorithm for both models, which is myopic, distributed and performs few logical operations at each node per time unit.

Cheng's [92] work has the purpose to evaluate a dynamic scheduling based nursing support system in order to support the work of nurse in acute centre. This nursing support system is designed to predict the complex practical environment which has unpredictable patient incoming and the variability of processing times in nursing care. The system is implemented in a series of laboratory experiments in the simulated conditions. Cheng's research finds the conclusion that the dynamic scheduling method generates an average improvement 71% based on earliness or tardiness of care in contrast to the performance of nurse with their previous procedures. Thus, the dynamic scheduling based nursing support system is highly applicable to improve nursing work in the practical environment.

Nayak [93] explores the dynamic task scheduling problem based on a grid environment of multi-processors. In this paper, task scheduling is formulated as an optimization problem and then optimized with a novel hybrid optimization algorithm. This novel algorithm is achieved by combining Genetic Algorithm and Bacteria Foraging optimization. The performance of this algorithm is proved to be superior through the simulation.

Guo [31] researches the clinic scheduling based on a simulation approach. Its scenario is about scheduling outpatients for a doctor's visit so as to guarantee the adequate care of patients and improve the utilization of the resource. This research uses simulation to develop the models of scheduling policies based on the patient flow logic. The whole research is achieved by three steps: firstly, the key operational parameters are obtained by creating a modelling strategy; secondly, a computational test bed is provided to refine the scheduling strategy used in the call centre; finally, this research aims to provide a decision support tool for the clinic management. Patient scheduling is a crucial determinant in any healthcare system; thus, this research provides a meaningful approach to explore related scheduling strategy in a hospital clinic system.

Pham's [32] research aims to explore surgical case scheduling for the hospital. Surgical case scheduling is to allocate healthcare resource to patients need surgical treatment in time. This research is very important in efficiently utilizing healthcare resource and improving the quality of healthcare. A novel surgical case scheduling approach, named multi-mode blocking job shop (MMBJS) is proposed in this research. Using the model, computational experiments are carried out with the DPLEX solver. This research provides useful information of scheduling for healthcare systems and related modelling processes.

Hashimoto [33] studies the performance of patient flow in the clinic in order to enhance patient convenience including minimized waiting time due to the seriously competitive healthcare resource. Hashimoto uses simulation to model a healthcare system and study the medical situations. Simulation is used to identify the problems and discover the limitations so that the current scheduling process can be refined. The simulation program is completed with Turbo Pascal 6.0. From Hashimoto's work, it is helpful to use the simulation of the patient flow to study the scheduling process in the clinic system and find out the problems to operate the system well.

Lowery's [34] study has the objective of developing an admissions scheduling system in order to manage hospital inpatient bed occupancy. Technically, this research develops a simulation model for the process from the arrivals to the discharges in the hospital, which is used to perform the scheduling system. The whole research is completed with three steps: firstly, create a simulation model of the hospital, which is used to design a scheduling system; secondly, validate the simulation model; finally, use the model to create a refined scheduling system to solve the occupancy problem. The simulation model is written in the GPSS/H simulation language. This research demonstrates efficient solutions and many benefits of using simulation again.

Kopach [35] promotes the research to improve the patient access and reduce the patient no-show rate based on the open access scheduling. Kopach uses discrete event simulation to model and study the effects referring to the clinic performance in terms of the clinic throughput and patient continuity of care, and obtains the conclusions that open access is able to perform efficient scheduling process in clinic throughput and little sacrifice in continuity of care if configuration is correct. This research successful accomplishes a simulation framework involving the models of the call-in process, patient no-show and clinic activities, as well as related analysis of the effect of healthcare providers on clinic performance and continuity of care. Meanwhile, an overbooking approach is explored to improve the patient access; the effect of the fraction of patients using open access is also analyzed in this research. Finally, a tentative model is created to explore the relationship between appointment lead time and patient no-show, as well as the negative effect of patient no-show based on the clinic throughput.

Cayirli [36] provides a comprehensive research survey about appointment scheduling in healthcare service. According to Cayirli's survey, literature reviews about outpatient scheduling can be organized as follow:

- **Number of services**

  Most researches in the literature use a single queue model representing the patient queuing process. However, there are still some studies exploring more complex clinic environments through some hospital activities, such as, registration, pre-examination, post-examination, x-ray, laboratory, checkout, and so on [37, 38, 39]. For these multi-stage models, Markov process modelling is used to achieve the patient flow mode with a series of activities.

- **Number of doctors**

  In previous work, most researchers focus on the single server system for the appointment scheduling. However, in a real hospital, patients are generally arranged a one-to-one doctor-patient appointment; thus, in the model, each doctor is designed with an independent queue [37, 38]. For another case, in some clinics, no appointment system is available for the patients whom are sent to the first available doctor [40, 41, and 42]. Furthermore, Swisher et al. [39] also model a more complex system which has different types of medical staff; and each patient category has a probability of requiring a particular type of staff.

- **Number of appointments per clinic session**

  The number of appointments per clinic session is a key factor in an appointment system. This factor has a close relationship with the patient waiting time, which is studied by Vissers [43], Heaney et al [44] and Meza [45]. Furthermore, this factor is also considered when exploring the performance of scheduling strategies by Welch and Bailey [46], Vissers and Wijngaard [47], and Ho and Lau [48].

- **Arrival process**

  In arrival process, there are two major concerns which are the unpunctuality of patients and the presence of no-shows. The unpunctuality of patients is the difference between appointment time and actual arrive time. Mercer [49, 50] uses an independent variable with a certain maximum lateness limit to model the unpunctuality of patients in the queuing model. The presence of no-shows is depicted by the no-show probability. According to Ho and Lau's [48] assessment, the no-show probability is the major factor which can affect the performance of an appointment system.

- **Service time**

  Service time can be defined as the total time of claiming the doctor's attention, preventing the doctor from serving others [51]. In fact, the service rate can be altered by doctors in terms of the varying amount of waiting patients. This case has been examined by Bailey [51], Rockart and Hofmann [52], Rising et al [37], and Babes and Sarma [40]. According to the studies of Bailey [51], Blanco White and Pike [53], Vissers and Wijngaard [47], Ho and Lau [48], Klassen and Rohleder [54], Denton and Gupta [55], both patient waiting times and doctor idle times can be seriously disturbed by the high variability of service times. Investigation about the effect of service time duration shows that shorter mean service times can cause shorter patient waiting times [51, 53, 56].

- **Lateness and interruption level of doctors**

  In the healthcare system, doctors' unpunctuality is a sensitive factor affecting patient waiting times, which is researched by Blanco White and Pike [53], Fetter and Thompson [57], Vissers [43], Mahachek and Knabe [58], Babes and Sarma [40], Liu and Liu [41, 42]. Another important factor related to the patient waiting times is the interruption level, also named "gap times". This includes several behaviours of doctors, such as, interaction with other staff, phone calls, writing up notes, or comfort breaks, and so on. These interrupted activities can directly influence the consultation. Gap times between consultations have been researched by Rising et al [37] and Lehaney et al [59] with the simulation model.

- **Queue discipline**

  Most previous studies use the first-come-first-served (FCFS) policy for the arriving patients. This policy is usually based on the order of the appointments. However, according to the studies of Rising et al. [37] and Cox et al [38], for the walk-ins in the clinic, the first priority is given to the emergencies, and then the scheduled patients are served. The walk-ins have the lowest priority and are served on a FCFS basis.

Overall, the scheduling topic has been studied by a lot of researchers. This thesis finds a novel research point. This is to study different dynamic scheduling policies under various conditions and to find out the optimal scheduling policy in each kind of condition. In addition, formal method is used in the modelling work; however, most other researchers use simulation as the main tool. These are two different points in contrast to the previous related work by other researchers.

2.1.3 Capacity

To solve the efficiency problem in the patient flow, efficient capacity planning for the healthcare resource is also a significant issue. Wolstenholme [14] illustrates that U.K. health services in all localities currently have to face two major problems: those of growing waiting lists for elective surgery in acute hospitals and the growing rate of non-elective hospital admissions, particularly of older people in winter. The latter is sometimes referred to the "winter bed crisis". Both are resource allocation problems. Hence, efficient capacity planning is essential to work out the resource allocation problems. Capacity planning is the process of determining the production capacity needed by an organization to meet changing demands for its products. The other half of the thesis researches the capacity planning for the hospital resource which is based on the scenario in the Rheumatology department of Royal Hallam Hospital in Sheffield, UK.

Vansteenkiste's [94] research aims to define a method to allocate operating room block times among surgical disciplines in order to treat patients within an acceptable time. As the demand for surgical treatment is rising but the operating room resource is limited, this kind of method for capacity management becomes an urgent requirement for the hospital. This study introduces a concept of the individual patient deviation from the optimal due time, and explores the potential ability of this concept as a driver for operating room allocation. The analysis is carried based on the abdominal and gynaecologic surgery with the retrospective data. According to the results of the analysis, the due-time concept is proved to be a valid measure to quantify operating room resource use. The due-time based model generates a transparent and acceptable system for the reallocation of operating room times.

Rau [95] studies the strategic capacity planning for an outpatient physical therapy clinic in Taipei through a simulation model. This study has a main objective which is to find out the capacity limits in the number of patients for each session of the outpatient physical therapy service. The discrete event simulation is used to model the dynamics of patient mixes with realistic treatment plans. Thereafter, estimation for the practical capacity of the physical therapy rooms will be made based on the simulation model.

Schutz's [96] research considers the capacity allocation in the industry service to solve the problem caused by the cancellations for the reserved service and no-shows. This research relates this problem to capacity allocation in radiology services. In this study, Continuous-time Markov process is used to model this problem. The Continuous-time Markov model is solved by the simulation based approximate dynamic programming (ADP) together with a discrete event simulation of the service period. To solve this problem, an adapted ADP algorithm is found and investigated to certify the benefits of using this algorithm to solve the problem. The model results prove that this ADP algorithm performs very well.

Kanter [60] studies the capacity of hospital emergency surge based on the scenario of the New York state. This research investigates the hospital occupancy and the patterns of variation in occupancy in the state wide scope. Furthermore, it also explores the ability of hospital in new patient accommodation. Analysis is based on the statistical data which are collected from the New York Statewide Planning and Research Cooperative System from 1996 to 2002. This analysis is carried out with the STATA tool version 9.1. This research uses statistical methods to analyze the capacity of hospital in terms of the real data collected from the system.

Lovejoy [61] has the objective to decide how to extend the capacity of the hospital. According to the research scenario, there are two options for capacity extension, build new operating rooms or extend working hours. The selection of these options must be considered seriously, as it refers to the benefits of patients, surgical staff and the hospital administrators. This research investigates the trade-offs in terms of three performance aspects, wait to get the schedule, scheduled procedure start-time reliability, and hospital profits. Lovejoy uses mathematical modelling to create the models and promote the analysis, and finally provides some process improvements and suggestions for the selection of extending operation room hours.

Nguyen [62] focuses on develop a new method of determining the number of acute hospital beds to minimize the roundabout approaches of the previous methods. The previous methods deciding the amount of acute hospital beds are based on the target bed occupancy rates. Nguyen designs a new method and applies it to two different departments in order to compare with the ratio method. The results of this research show that the new method is efficient and robust, which can be used as a real alternative to the ratio methods. An application using this new method has been developed for the hospital.

According to Gallivan [63], many UK hospitals operate close to the capacity. The overload operation of the hospital may lead to the degraded performance of the booking system. Gallivan argues that there are three key factors affecting the hospital operation: reserve capacity, unpredictable variability, and blocking. This research explores the interaction of these three factors in order to improve the efficiency of hospital operation. Mathematical modelling is used to carry out the analysis and the data is collected from the hospital records. The main measurements of this study focus on the capacity requirements of an intensive care unit for a hypothetical booked admission system and the length of stay for the intensive care. This research has a conclusion that the hypothetical booked admission system may cause low efficiency in the operation when the length of stay has high variability and the reserve capacity is limited.

Vissers [64] explores the allocation of inpatient resources in the hospital, such as beds, operation rooms and nursing staff, and so on. This research uses patient flow to describe the allocation process in the hospital and carry out the evaluation of impacts on the utilization of hospital resources. Vissers has developed an approach to allocate resources in terms of the demand so as to get the balance of resource utilization [65]. This approach uses computer models to support the decision making of the resource allocation in the hospital. This research demonstrates a case study that is about using this approach for the shortage of beds and frequent admission stops in the hospital. From this case study, more details have been obtained in the area of patient flow based capacity planning.

Kim [66] researches the efficient utilization of a hospital intensive care unit (ICU). This subject is important due two to aspects, the limited and critical resource of the ICU and the utilization of the resources which referring to the welfare of patients and the cost of the hospital. Kim focuses on investigating the criteria on the decision making in the ICU, which may affect the operations of other departments of the hospital. This objective is achieved by analyzing the process of admission-and-discharge in the ICU. Queuing based computer simulation is used to simulate this process. The data is collected from the ICU. The final results aims to improve the unit capacity utilization in the ICU and the quality of service provided to patients.

Elkhuizen [67] studies the capacity of the nursing staff required in the wards by creating a comprehensive capacity model, and also explores the approaches to improve the capacity utilization. In the research, Elkhuizen uses a capacity to calculate the required number of nurses. This capacity model is also used to address the three operational, tactical, and strategic capacity management problems: workflow optimization on paediatric wards, consequence prediction of length of stay reduce on nursing staff for a cardiology ward, and nursing staff calculation with the consequence of merging two medicine wards. This research successfully uses the capacity model to support the capacity decisions on the operational, tactical, and strategic levels. Furthermore, the capacity model also supports the efficient requirement management of staff and beds.

McGowan [68] carries out research to evaluate the efficiency of operating room at a time of maximum capacity and occupancy. Based on McGowan's investigation, when the academic medical centres experience the maximum capacity, patient throughput and care quality may be seriously affected. The situation of the operating room throughput is the most concern in the overall academic medical centre. To solve these problems, McGowan applies a multidisciplinary approach for the management of operating room. This approach involves seven strategies: "daily communication, improved bed planning, discharge by noon program, internal staffing pool, special assignments for a patient transition unit, incentives, and stepped up environmental services". With the approach, the capacity and occupancy problem on the operating room has an effective solution.

Delia [69] explores how the variation of quantity of patients and beds affects the hospital surge capacity. According to Delia's description, due to the fact that emergency department in the United States requires adequate surge capacity at a time of mass casualty events, research for hospital surge capacity is essential. This research is aimed to improve the analysis of hospital surge capacity in terms of administrative records. The surge capacity is measured based on occupancy rates and the number of idle beds depending on the disaster planning benchmark. A statistical method is used to promote the refined calculation for the number of idle beds, and then make a comparison with the annual bed statistics and federal disaster planning benchmarks. Finally, Delia identifies misleading information from the annual bed statistics in the availability of hospital surge capacity.

Overall, capacity planning is a popular research topic which aims to improve the utilization of the limited resource. A good capacity plan can improve the work efficiency as well as the resource utilization. In this thesis, the research of capacity focuses on the allocation of hospital staff for each section of a work flow. It aims to provide a optimal capacity plan for a fixed workflow in the hospital department.

2.2 Modelling and analysis techniques

This section illustrates the modelling approaches used in this thesis including formal method and discrete event simulation. The formal method is used to simply the modelling process of large scale models. The discrete event simulation is applied to validate the formal method. The use of formal method in such kind of model scenario is a novel point in this thesis.

Harrison [10] uses a stochastic modelling technique to indicate how the effect of a particular smart environment design can impact the collective behaviour of users of the environment. According to Harrison [10], Performance Evaluation Process Algebra [16] (PEPA) and the analysis techniques associated with PEPA can provide a scalable approach. In Harrison's research, a fluid flow analysis [17] based on stochastic process algebra (PEPA) has been used for the analysis of a large scale model for the first time. This model has complex structure and a large number states and behaviours in order to model the individual movement in an area. According to the results of Michael's work, PEPA model generates accurate analysis and reliable outcomes based on fluid flow approximation. As inspired by Harrison's work, the patient flow in this project can be modelled with PEPA and analyzed with the fluid flow analysis. Michael's research proves that PEPA model and fluid flow approximation can be used to model complex human behaviour based large scale model. In this thesis, PEPA is used to undertake the modelling work with complex human factors. This thesis has different research points in modelling with Michael's work, which is to use function rates to model some complex process rather than just creating a very large scale model to complete all model behaviours.

Karnon [80] aims to apply the mathematical modelling techniques to the study of health care case study. Karnon's research uses a cohort Markov model to indentify a cost-effective screening programme for cervical cancer to improve the delivery of health care at a more aggregated level. According to Karnon's study, queueing theory has been successfully used to evaluate a large number of screening options and estimate the number of places required for a new transition care facility in a hospital. Karnon demonstrates the potential for using the mathematical modelling (e.g. queueing theory) in the health care.

Gorunescu [81] uses queueing theory to model behaviours of patients in a hospital and presents a way of optimizing the hospital resource management to improve the healthcare quality. In Gorunescu's research, the queueing theory is applied to model the main characteristics of the access of patients to hospital. Thereafter, mean bed occupancy is studied in terms of the model. Based on the queueing model, Gorunescu generates a way of optimizing the bed occupancy management by balancing costs of empty beds against costs of delayed patients. Gorunescu's study illustrates the queueing theory is able to applied for the health care applications in order to research the hospital resource management and planning.

2.2.1 PEPA: Performance Evaluation Process Algebra

This sub-section introduces a formal method used in the thesis which is called Performance Evaluation Process Algebra (PEPA). Why PEPA is used as the main modelling technique in this work? PEPA has some attractive features that are not achieved by other modelling techniques, such as, compositionality, formality, abstraction. These features mean that PEPA is able to model the interaction of subsystems which belongs to the same top-level system (namely compositionality); PEPA is able to define all items with a precise meaning in the language (formality); and PEPA has the ability to create the complex model from detailed components, and some model details can be disregarded if this is required (abstraction).

2.2.1.1 Definition and features

Performance Evaluation Process Algebra (PEPA), introduced by Jane Hillston in the 1990s, is a stochastic process algebra designed for modelling computer and communication systems. Classical process algebras, such as Hoare's CSP and Milner's CCS, are extended by PEPA through introducing of transitions probabilistic and branching timing.

Communicating Sequential Processes (CSP) is a formal language for describing patterns of interaction in concurrent systems [82]. This CSP is first created by C.A.R. Hoare in 1978, and the CSP is widely used in industry to specify and verify the concurrent aspects of a variety of different systems [83].

Calculus of Communicating Systems (CCS) is a process calculus created by Robin Milner around 1980. The actions of CCS can model indivisible communications between exactly two participants. CCS is a formal language which includes choice between actions and scope restriction, and primitives for describing parallel composition [84, 85].

According to Hillston’s specification, PEPA has several active features of process algebras [16]:

- ❖ **Parsimony**: Process algebras are simple languages with only a few elements. This parsimony means that it is easy to reason about the language and provides a great deal of flexibility to the modeller.
- ❖ **Formal definition**: The language is given a structured operational semantics (SOS) [86, 87], providing a formal interpretation of all expressions. The notions of equivalence which are subsequently

developed are based on these semantic rules. This gives a formal basis for the comparison and manipulation of models and components, and introduces the possibility of developing tools to automate, or semi-automate, these tasks.

- **Compositionality**: This means the ability to model a system as the interaction of subsystems. In PEPA, the *cooperation* combinatory forms the basis of composition. The model simplification and aggregation techniques can be developed which are complementary to this combinator. This means that part of a model can be simplified in isolation, if its interaction with the rest of the system is modelled by such a combinatory, and replaced by the simplified component without jeopardising the integrity of the whole model.

2.2.1.2 Description and terminology

PEPA defines a system to be interactions between components which engaged either singly or multiply in *activities* [16]. These components are either atomic or composed of components, and correspond to identifiable parts in the system, or roles in the behaviour of the system [16].

In PEPA, the term *activity* is used to distinguish from the usual process algebra notion of an instantaneous *action*. Every activity in PEPA is associated with an exponential distributed duration variable; however, the term *action* relates to the behaviour of the system [16]. Each activity has an *action type* (or simply *type*), and each discrete action within a system has a unique type, and all possible types are included in a countable set, A [16]. However, in the situation when a system is executing some action (or sequence of actions) with unknown identity, there is a distinguished action type, τ, which can be recognized as *unknown* type [16]. As PEPA only supports the exponential distribution for its parameters, the duration of an activity; thus, an exponential distributed random variable can be represented by a single real number parameter under the exponential distribution [16]. This parameter is called the *activity rate* (or simply rate).

Thus, each activity in PEPA, a, is defined as a pair (α, r) where $\alpha \in$ A is the action type and r is the activity rate. As a result, there is a set of activities, Act $\subseteq$ A × R^+, where R^+ is the set of positive real numbers together with the symbol τ [16].

2.2.1.3 Syntax

PEPA has only four combinators, which are, prefix, choice, cooperation and hiding.

- **Prefix:** It is the basic building block of a sequential component: the process $(a,r).P$ performs action a at rate r and then evolves to component P.
- **Choice:** It generates a competition between two or more potential processes: the process $(a,r).P + (b,s).Q$ represents that either action a or b wins the race at the rate r or s and then behaves as P or Q respectively.
- **Cooperation:** Its operator joins two processes together, in which these two processes may share some actions: the process$P \bowtie_L Q$, $L = \{a,b\}$ denotes that processes P and Q must cooperate with the shared actions a and b. Any other action is performed independently. Additionally, $P||Q$ in PEPA syntax means $P \bowtie_L Q,\ L = \phi$.
- **Hiding:** the process $P|\{a\}$ hides the action a from view and prevents other processes from joining with it.
- **Constant:** As assumed, a countable set of *constants* is defined. The constants are components with the meaning of defining an equation such as $A \overset{\text{def}}{=} P$ which gives the constant A the behaviour of the component P. This is the way of assigning names for behaviour components.

The syntax of PEPA in describing the above processes is given as:

$$P ::= (a,\lambda).P \mid P + Q \mid P < L > Q \mid P|L \mid A$$

This PEPA statement involves all four combinators mentioned in the previous paragraph. The last part of this statement P ::= A is to identify component P with A. When the rate of the action is passive, we use the symbol ⊤.

Here, a simple example is used to demonstrate PEPA model in terms of the description in Hillston's thesis [16]. This simple PEPA model describes a simple system in which a process repetitively carries out some task. The task must be completed with two steps: firstly, the process need access a resource; and secondly, for the resource, it is continuously ready for use except it has been used which need be reset for reuse. According to above specification, this simple PEPA model can be defined as:

$$Process \overset{def}{=} (use, r_1).\, Process'$$

$$Process' \overset{def}{=} (task, r_2).\, Process$$

$$Resource \overset{def}{=} (use, r_3).\, Resource'$$

$$Resource' \overset{def}{=} (update, r_4).\, Resource$$

$$System \overset{def}{=} Process \bowtie_L Resource$$

$$L = \{use\}$$

This model has two components: Process and Resource. The Process will carry out two actions: use with the rate r1, means access the resource which is in cooperation with the Resource, and task with the rate r2, means completing the remainder task. Similarly, the Resource also has two actions: use with the rate r3, and update with the rate r4. The System represents the cooperation of Process and Resource for the system actions described in previous paragraph.

However, as the analyser for PEPA cannot undertake the analysis with function rates in the model, another modelling technique Chemical Model Definition Language (CMDL) is introduced to solve this problem. Next sub-section will give more detailed information about CMDL and why use it.

2.2.2 CMDL: Chemical Model Definition Language

Consistent with Ramsey's definition in CMDL user manual, "the Chemical Model Definition Language (CMDL) is a simplified model definition language designed to minimize the amount of repetitive typing required to define a model”. A fundamental concept in CMDL is the symbol value, which consists of a symbol name and a value. A value may be either a constant or a mathematical expression. Sometimes, the mathematical expression is enclosed in square brackets, which is defined as a bracketed mathematical expression. Here is a simple symbol value definition:

$$S = 1.0;$$

In the above example, "S" is the symbol name, and its value is 1.0.

The CMDL language is centred on the reaction statement. A reaction statement defines a one-way chemical reaction involving zero or more chemical species participate as reactants and zero or more chemical species participate as products. The following example is a simple CMDL reaction statement:

$$A = 100.0;\ B = 100.0;\ C = 0.0;\ D = 0.0;$$

$$creation\_of\_c\_d, \qquad A + B \rightarrow C + D, \qquad 1.0;$$

In this reaction, *A* and *B* are reactants, and *C* and *D* are products. "*creation_of_c_d*" is the name of reaction, and reaction rate is 1.0.

Alternatively, the reaction rate can be a mathematical expression instead of a constant. In the example of above reaction, one might write:

$$creation\_of\_c\_d, \qquad A + B -> C + D, \qquad 2.0 * A * B;$$

Or

$$creation\_of\_c\_d, \qquad A + B -> C + D, \qquad [2.0 * A * B];$$

In the first reaction, the mathematical expression will be evaluated immediately, and to use the resultant value as the rate for the reaction.

However, the second bracketed mathematical expression is defined as a deferred evaluation expression. This expression is used to calculate the rate at each time the rate need be computed. This expression is sometimes used as a "custom rate expression” which a self-defined rate by users distinguishing from the CMDL built-in methods of calculating the reaction rate. The square brackets used in the expression notify the parser that a custom reaction rate is defined here.

The reason why we impose CMDL model lies in the custom reaction rate. PEPA analyser cannot support the analysis with a functional action rate, while the CMDL model analysis has the ability to perform such functional reaction rate, and the PEPA model can be transformed to an equivalent CMDL model. In the CMDL model, the reaction name is equivalent to the action type in PEPA model; the reactants and products in a reaction are actually components in PEPA model; and the reaction rate of CMDL model is the action rate in PEPA model. Moreover, it is convenient to transform a PEPA model to a CMDL model via Eclipse based PEPA analyser.

CMDL can be used as an alternative technique instead of PEPA because the PEPA Plug-in can transform PEPA to an equivalent CMDL model. This equivalence will be verified by solving specified models in the following chapters. As the CMDL model is completely equivalent to the PEPA model, the analysis based on the CMDL model is reliable when it is used to evaluate the performance of scheduling policies and capacity plans.

2.2.3 Fluid flow analysis

Fluid flow analysis, based on a set of coupled ordinary differential equations ODEs, is a novel performance analysis technique for large scale systems in the stochastic process algebra PEPA.

In many cases, process algebra uses an exponentially distributed random variable in mathematical models based on the continuous time Markov chain [17] as in Section 2.2.1. To solve these models, either steady state or transient probability distribution can be used via linear algebra. However, there is a problem existing in the state space explosion. Process algebra is prone to generate extremely large state spaces making numerical solution via linear algebra very costly or intractable even very simple models [17]. Due to this effect, fluid flow analysis is developed to support the analysis of PEPA for systems with large numbers of replicated components.

According to the theory of the fluid flow analysis [17], in a system when the number of components is large and the steps are relatively small, the movements between behaviours can be approximated to be continuous rather than occurring in discrete jumps. Hence, the fluid flow analysis replaces the discrete event system represented by the derivation graph of a PEPA process by a continuous model, and this continuous model is achieved by a set of coupled ordinary differential equations (ODEs) [17]. Thus, by adopting a continuous approximation, the fluid flow approximation, of the behaviour of the model, we are able to analyse systems of arbitrarily large scale without resource to the underlying state pace representation.

2.2.4 Stochastic simulation

Stochastic simulation algorithms and methods are initially developed to analyse chemical reactions involving large numbers of species with complex reaction kinetics [18]. The first algorithm of stochastic simulation is the Gillespie algorithm which was proposed by Daniel Gillespie in 1977. It is an exact procedure used for the numerical simulation of the time evolution of a well-stirred chemically reacting system. In probability theory, the Gillespie algorithm statistically generates the right solution of a stochastic equation. This algorithm was initially created by Joseph L. Doob in 1945, and then Daniel Gillespie generalised it in 1977. In Gillespie's paper, this algorithm is used to simulate chemical or biochemical systems of reactions efficiently and accurately using limited computational power. The core contribution of Gillespie is that the use of his algorithm is extended from statistical thermodynamics to the numerical simulation of the dynamic evolution of a chemically reacting system [19]. With the increased processing speed of modern computers, Gillespie's algorithm has been widely used for the simulation the large and complex computer systems.

In 2006, stochastic simulation technique is introduced to the PEPA model analysis by Bradley et al [19]. As mentioned before, the traditional process algebra model limits in the state space explosion problem due to the practical size and the complexity of the model. One possible way to solve this problem is the fluid flow analysis [17]. Another way is to use a formal transformation from stochastic process algebra PEPA to an equivalent rate equation description, and then use a stochastic simulator to provide trace executions of the underlying rate equations and use multiple traces to provide confidence intervals for being in a given state [19].

2.2.5 Discrete event simulation

Usually, it is more efficient and time saving to solve models by analytical or numerical means. However, numerical techniques cannot solve all model types, and there are still some models outside the scope of the numerical modelling techniques. These models can still be analyzed using simulation. Simulation has no fundamental restrictions towards what kinds of models can be solved. Simulation can be divided into two types: continuous event simulation and discrete event simulation. In this thesis, only discrete event simulation is applied as the function of the discrete event simulation fits the requirements of this model task.

In discrete-event simulation (DES), the operation of a system is represented as a chronological sequence of events. Each event occurs at an instant time point and marks a state change in the system [20]. Generally, a discrete event simulation consists of a set of system state variables and the logic of what happens when system events occur. In addition, discrete event simulations also include a *clock*, an *events* list, *random number generators*, statistics and ending conditions. In discrete event simulations, the state changes occur at discrete time points [21]. The *simulation time* is the value of the parameter "time" used in the simulation program. This parameter usually corresponds to the time value in the real world system. The *run time* is the time used to run a simulation program [21]. In a discrete event system, the state changes against the time, and each cause of a state change is called an *event* [21]. Frequently the state changes themselves are also called *events*. As the events take place one by one in the simulation, we speak of discrete event simulation. In discrete-event simulations, the timing of event movements from one state to another in the system exactly describes the performance of the simulated system [21]. Thus, to evaluate the performance of systems, measurements based on specific events can be derived in terms of the average inter-event time or the average time between events [21]. Such measures form the basis of the computation of performance estimates.

Discrete event simulation is widely used in the computer network area. For example, a classic discrete event simulation case for the network is to model a queue. Here, the hospital patient queue is used as an example to present the use of the simulation for a queuing network. In the hospital, patients arrive at a department to be served by a consultant or a nurse. In this case, the queuing network entities are patient queue and hospital staff. The events in the queuing system are patient-arrival and patient-departure. The event of consulting service is a part of the logic of the arrival and departure events. The system states changed by these events are the number of patients in the queue and the status of consultants that is busy or idle. Patient arrival time and consultant service time are random variables used to model the system stochastically. Furthermore, discrete event simulation is also suitable to model hospital applications. This thesis uses the theory of the discrete event simulation to build hospital patient flow model and capacity model.

Chapter 3 Dynamic scheduling model with fluid flow approximation

3.1 Model Scenario

The aim of this chapter is to investigate the performance of different scheduling policies by employing Performance Evaluation Process Algebra (PEPA) and related fluid flow analysis. A model is presented which is based on a hospital patient flow scenario, with a large number of patients in the system.

To model the patient scheduling process, a simple model of a hospital scenario with a smart environment is created. In the model, new patients need to register at the reception on arrival. Each patient will then be issued an identity card or registered with their own smart device in the system. Thereafter, the system is able to recognize each registered patient in the hospital and provide routing information for each patient on the nearest public display or send text message to their smart device. The system can also choose the optimal route having the least waiting queue for patients.

In this case, it is assumed that all patients have three tests: blood test, x-ray film and electrocardiographic examination (ECG), and there is no pre-defined order required in the three activities. Thus, the patients will be given routing information once arriving in the hospital. After finishing these three activities, the patients leave the hospital. Under these assumptions, the hospital has a series of incoming patients without a stop.

In the following sections, static and dynamic scheduling policies will be introduced and combined with the hospital scenario in order to examine whether such policies are able to improve the efficiency by scheduling patients' activities in the hospital.

3.2 Model specification of static and dynamic scheduling policy

In this section, the definitions of static scheduling policy and dynamic scheduling policy are demonstrated through designing a specific scheme for each policy. PEPA is used to model the two schemes in order to generate measurements and analysis.

3.2.1 Policy specification

According to the scenario, two potential scheduling schemes are designed to promote the modelling and analysis. The first one is a simple static scheme. In this scheme, the route of each patient is fixed at the beginning when they arrive in the hospital and register at the reception. During their whole examining process, the route cannot be changed. As displayed in Figure 3.2.1, all six potential routes are fixed, which cover all possible permutation and combinations of threes tests. Once a route is decided for a patient, it is not allowed to change until the end. Hence, this scheme actually represents a complete static scheduling process. The advantage of this scheme is simple and easy to generate, as the system only need to make one-off decision for each patient.

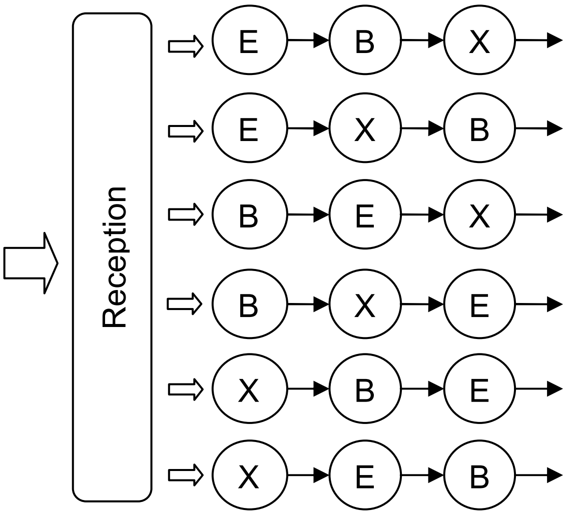


Figure 3.2.1 Static scheduling scheme (E: ECG, B: Blood, X: X-Ray)

However, the potential problem is the applicability and efficiency in the scheduling process. Such static schemes only support the static scheduling policy because of its one-off decision making mechanism. In fact, for the dynamic scheduling policy, multiple routing selections are required in the scheduling process. This is why the second scheme is introduced here.

The second scheme is a dynamic scheme that is more flexible compared with the first one. In this scheme, the route is no longer fixed. The system only advises the route for the first examination after registration. After the first examination is finished, the system then automatically provides the route of the next examination that the patient needs to go. Hence, in this way, the system can dynamically help patients to select the route step by step.

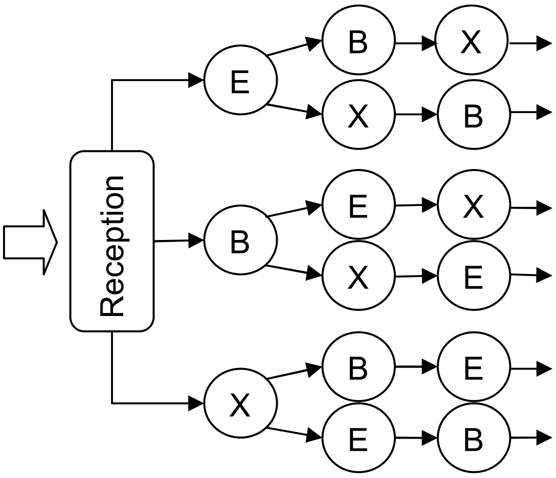


Figure 3.2.2 Dynamic scheduling scheme (E: ECG, B: Blood, X: X-ray)

Figure 3.2.2 depicts the basic structure of this dynamic scheme. The most important feature of the dynamic scheme is that it can carry out a dynamic scheduling in the routing process, which means the system is able to dynamically choose the next destination and modify the route after completing a test rather than making a once-for-all decision at the beginning. Thus, this dynamic scheme is a good way to perform a routing process with a specific dynamic scheduling policy.

In the following sections, these two schemes will be modelled with PEPA and analyzed to explore the efficiency of two schemes.

3.2.2 Policy modelling with PEPA

On the basis of the assumptions introduced above, two different PEPA models are built to achieve the static scheduling scheme and dynamic scheduling.

3.2.2.1 Static model scheme

In keeping with the static scheduling scheme, the whole process can be modelled in PEPA:

$$\begin{aligned} Patient \overset{def}{=} & (decide_1, \top).Patient_1 \\ & + (decide_2, \top).Patient_2 \\ & + (decide_3, \top).Patient_3 \\ & + (decide_4, \top).Patient_4 \\ & + (decide_5, \top).Patient_5 \\ & + (decide_6, \top).Patient_6 \end{aligned}$$

$$\begin{aligned} Patient_1 &\overset{def}{=} (ecg, r_{ecg}).Patient_7 \\ Patient_7 &\overset{def}{=} (xray, r_{xray}).Patient_8 \\ Patient_8 &\overset{def}{=} (blood, r_{blood}).Patient_9 \\ Patient_9 &\overset{def}{=} (wait, r_{wait}).Patient \end{aligned}$$

$$\begin{aligned} Patient_2 &\overset{def}{=} (ecg, r_{ecg}).Patient_{10} \\ Patient_{10} &\overset{def}{=} (blood, r_{blood}).Patient_{11} \\ Patient_{11} &\overset{def}{=} (xray, r_{xray}).Patient_{12} \\ Patient_{12} &\overset{def}{=} (wait, r_{wait}).Patient \end{aligned}$$

$$\begin{aligned} Patient_3 &\overset{def}{=} (blood, r_{blood}).Patient_{13} \\ Patient_{13} &\overset{def}{=} (ecg, r_{ecg}).Patient_{14} \\ Patient_{14} &\overset{def}{=} (xray, r_{xray}).Patient_{15} \\ Patient_{15} &\overset{def}{=} (wait, r_{wait}).Patient \end{aligned}$$

$$Patient_4 \overset{def}{=} (blood, r_{blood}).Patient_{16}$$
$$Patient_{16} \overset{def}{=} (xray, r_{xray}).Patient_{17}$$
$$Patient_{17} \overset{def}{=} (ecg, r_{ecg}).Patient_{18}$$
$$Patient_{18} \overset{def}{=} (wait, r_{wait}).Patient$$

$$Patient_5 \overset{def}{=} (xray, r_{xray}).Patient_{19}$$
$$Patient_{19} \overset{def}{=} (blood, r_{blood}).Patient_{20}$$
$$Patient_{20} \overset{def}{=} (ecg, r_{ecg}).Patient_{21}$$
$$Patient_{21} \overset{def}{=} (wait, r_{wait}).Patient$$

$$Patient_6 \overset{def}{=} (xray, r_{xray}).Patient_{22}$$
$$Patient_{22} \overset{def}{=} (ecg, r_{ecg}).Patient_{23}$$
$$Patient_{23} \overset{def}{=} (blood, r_{blood}).Patient_{24}$$
$$Patient_{24} \overset{def}{=} (wait, r_{wait}).Patient$$

$$ECG \overset{def}{=} (ecg, r_{ecg}).ECG$$
$$Xray \overset{def}{=} (xray, r_{xray}).Xray$$
$$Blood \overset{def}{=} (blood, r_{blood}).Blood$$

$$Wait \overset{def}{=} (wait, r_{wait}).Wait$$

$$Decide_1 \overset{def}{=} (decide_1, r_{d1}).Decide_1$$
$$Decide_2 \overset{def}{=} (decide_2, r_{d2}).Decide_2$$
$$Decide_3 \overset{def}{=} (decide_3, r_{d3}).Decide_3$$
$$Decide_4 \overset{def}{=} (decide_4, r_{d4}).Decide_4$$
$$Decide_5 \overset{def}{=} (decide_5, r_{d5}).Decide_5$$
$$Decide_6 \overset{def}{=} (decide_6, r_{d6}).Decide_6$$

$$Patient[n]$$
$$\bowtie_L$$
$$(ECG||Xray||Blood||Decide_1||Decide_2||Decide_3||Decide_4||Decide_5||Decide_6||Wait)$$
$$L = \{ecg, xray, blood, decide_1, decide_2, decide_3, decide_4, decide_5, decide_6, wait\}$$

In this PEPA model, a series of *Patient* components with numbers represent a set of patient status generated after a set of actions. *ECG*, *X-Ray* and *Blood* components stand for three assumed physical examinations, and each component has an action representing the behaviour at this component. *Decide* components are defined to model the decision making process which is used to determine the route of patients. Six different *Decide* components provide six kinds of *decide* actions to control six different routes respectively. Finally, a *Wait* component provides an action "*wait*", which is used to model the process that patients stay outside the hospital. In addition, the rate of each action is specified in the analysis stage. "⊤" means a passive infinite rate, which means this action is a passive action. According to Hillston's definition of the passive rate, the action with a passive rate must be shared with another component that determines the rate of the shared action. In this model, using passive rate for *decide* actions is to model the process that the routing decisions are made by the system and then the routing information is passed to the patients. Thus, the rate of patients getting route information depends on the decision making process in the system.

$$\begin{aligned} Patient \overset{def}{=} & (decide_1, \top).Patient_1 \\ & + (decide_2, \top).Patient_2 \\ & + (decide_3, \top).Patient_3 \\ & + (decide_4, \top).Patient_4 \\ & + (decide_5, \top).Patient_5 \\ & + (decide_6, \top).Patient_6 \end{aligned}$$

In PEPA model, this block uses the "*choice*" operator in PEPA to model the process that the system generates six different routes for patients and directs the registered patients to the corresponding routes. Thereafter, the following blocks model the process that the patients complete their examinations on each possible route respectively.

$$\begin{aligned} Patient_1 & \overset{def}{=} (ecg, r_{ecg}).Patient_7 \\ Patient_7 & \overset{def}{=} (xray, r_{xray}).Patient_8 \\ Patient_8 & \overset{def}{=} (blood, r_{blood}).Patient_9 \\ Patient_9 & \overset{def}{=} (wait, r_{wait}).Patient \end{aligned}$$

For example, the above block is the process happened in the first possible route. In this route, patients take the ECG test first, and then go to the x-ray scan. Finally they take the blood test. Thereafter, the patients leave the hospital and re-enter after a waiting period. Note that in the physical interpretation of this model is not necessary that exactly the same patient would re-enter the hospital after a delay, but rather that this mechanism ensures that there is a finite maximum population in the hospital.

Similarly, five other blocks model the same routing process but in different orders. The rest model statements are the definitions of the components and their behaviours, such as:

$$ECG \overset{def}{=} (ecg, r_{ecg}).ECG;$$
$$Xray \overset{def}{=} (xray, r_{xray}).Xray;$$
$$Blood \overset{def}{=} (blood, r_{blood}).Blood;$$

$$Wait \overset{def}{=} (wait, r_{wait}).Wait;$$

$$Decide_1 \overset{def}{=} (decide_1, r_{d1}).Decide_1;$$
$$Decide_2 \overset{def}{=} (decide_2, r_{d2}).Decide_2;$$
$$Decide_3 \overset{def}{=} (decide_3, r_{d3}).Decide_3;$$
$$Decide_4 \overset{def}{=} (decide_4, r_{d4}).Decide_4;$$
$$Decide_5 \overset{def}{=} (decide_5, r_{d5}).Decide_5;$$
$$Decide_6 \overset{def}{=} (decide_6, r_{d6}).Decide_6;$$

All the above components are defined in terms of the process in the model scenario.

Each action in model is related with a rate which specifies how fast this action is executed in the model. It is worth to mention that the rates of decide action are all passive, but all others are active. The difference between these two rate types will be introduced later in analysis section with ODEs.

Finally, at the end of the PEPA model, there is a cooperation which joins the components together and specifies the number of instances of each component and the shared actions. The last block in the PEPA model is the cooperation statements.

3.2.2.2 Dynamic model scheme

In contrast to the static scheme, the model of dynamic scheme has some difference in defining *Patient* components and *Decide* components. The model statements are detailed as follows:

$$\begin{aligned}
\mathit{Patient} &\overset{def}{=} (\mathit{decide}_1, \top).\mathit{Patient}_1 \\
&\quad + (\mathit{decide}_2, \top).\mathit{Patient}_2 \\
&\quad + (\mathit{decide}_3, \top).\mathit{Patient}_3
\end{aligned}$$

$$\begin{aligned}
\mathit{Patient}_1 &\overset{def}{=} (\mathit{ecg}, r_{ecg}).\mathit{Patient}_4 \\
\mathit{Patient}_4 &\overset{def}{=} (\mathit{decide}_2, \top).\mathit{Patient}_5 \\
&\quad + (\mathit{decide}_3, \top).\mathit{Patient}_6 \\
\mathit{Patient}_5 &\overset{def}{=} (\mathit{xray}, r_{xray}).\mathit{Patient}_7 \\
\mathit{Patient}_7 &\overset{def}{=} (\mathit{blood}, r_{blood}).\mathit{Patient}_8 \\
\mathit{Patient}_8 &\overset{def}{=} (\mathit{wait}, r_{wait}).\mathit{Patient} \\
\mathit{Patient}_6 &\overset{def}{=} (\mathit{blood}, r_{blood}).\mathit{Patient}_9 \\
\mathit{Patient}_9 &\overset{def}{=} (\mathit{xray}, r_{xray}).\mathit{Patient}_{10} \\
\mathit{Patient}_{10} &\overset{def}{=} (\mathit{wait}, r_{wait}).\mathit{Patient}
\end{aligned}$$

$$\begin{aligned}
\mathit{Patient}_2 &\overset{def}{=} (\mathit{xray}, r_{xray}).\mathit{Patient}_{11} \\
\mathit{Patient}_{11} &\overset{def}{=} (\mathit{decide}_1, \top).\mathit{Patient}_{12} \\
&\quad + (\mathit{decide}_3, \top).\mathit{Patient}_{13} \\
\mathit{Patient}_{12} &\overset{def}{=} (\mathit{ecg}, r_{ecg}).\mathit{Patient}_{14} \\
\mathit{Patient}_{14} &\overset{def}{=} (\mathit{blood}, r_{blood}).\mathit{Patient}_{15} \\
\mathit{Patient}_{15} &\overset{def}{=} (\mathit{wait}, r_{wait}).\mathit{Patient} \\
\mathit{Patient}_{13} &\overset{def}{=} (\mathit{blood}, r_{blood}).\mathit{Patient}_{16} \\
\mathit{Patient}_{16} &\overset{def}{=} (\mathit{ecg}, r_{ecg}).\mathit{Patient}_{17} \\
\mathit{Patient}_{17} &\overset{def}{=} (\mathit{wait}, r_{wait}).\mathit{Patient}
\end{aligned}$$

$$\begin{aligned}
\mathit{Patient}_3 &\overset{def}{=} (\mathit{blood}, r_{blood}).\mathit{Patient}_{18} \\
\mathit{Patient}_{18} &\overset{def}{=} (\mathit{decide1}, \top).\mathit{Patient}_{19} \\
&\quad + (\mathit{decide2}, \top).\mathit{Patient}_{20} \\
\mathit{Patient}_{19} &\overset{def}{=} (\mathit{ecg}, r_{ecg}).\mathit{Patient}_{21}
\end{aligned}$$

$$Patient_{21} \stackrel{def}{=} (xray, r_{xray}).Patient_{22}$$
$$Patient_{22} \stackrel{def}{=} (wait, r_{wait}).Patient$$
$$Patient_{20} \stackrel{def}{=} (xray, r_{xray}).Patient_{23}$$
$$Patient_{23} \stackrel{def}{=} (ecg, r_{ecg}).Patient_{24}$$
$$Patient_{24} \stackrel{def}{=} (wait, r_{wait}).Patient$$

$$XRay \stackrel{def}{=} (xray, r_{xray}).XRay$$
$$Blood \stackrel{def}{=} (blood, r_{blood}).Blood$$
$$ECG \stackrel{def}{=} (ecg, r_{ecg}).ECG$$
$$Wait \stackrel{def}{=} (wait, r_{wait}).Wait;$$

$$Decide_1 \stackrel{def}{=} (decide_1, r_{d1}).Decide_1$$
$$Decide_2 \stackrel{def}{=} (decide_2, r_{d2}).Decide_2$$
$$Decide_3 \stackrel{def}{=} (decide_3, r_{d3}).Decide_3$$

$$Patient[n] \bowtie_L$$
$$(XRay||Blood||ECG||Decide_1||Decide_2||Decide_3||Wait)$$
$$L = \{ xray, blood, ecg, decide_1, decide_2, decide_3, wait \}$$

In terms of the basic components, such as, *ECG*, *Blood*, *X-ray* and *Wait*, the dynamic scheduling model has quite similar components compared with the static scheduling model, and such components perform the same function as those in the static scheduling model except the *Decide* component.

In this model, there are three *Decide* components in which each *decide* action controls a separate access to *ECG*, *Blood* and *X-ray* respectively. For example, $decide_1$ action can direct patients to the ECG test; $decide_2$ action points to the x-ray test, and $decide_3$ action is for blood test. The key point in this dynamic scheduling model is the definition of the *Decide* components which is different from those in the static scheduling model.

$$Decide_1 \stackrel{def}{=} (decide_1, r_{d1}).Decide_1$$
$$Decide_2 \stackrel{def}{=} (decide_2, r_{d2}).Decide_2$$
$$Decide_3 \stackrel{def}{=} (decide_3, r_{d3}).Decide_3$$

In the previous static scheduling model, the *Decide* component is defined to control the access to six different routes. However, here, the above *Decide* components are defined to access three kinds of tests, such as: *Decide*$_1$ for ECG, *Decide*$_2$ for x-ray scan and *Decide*$_3$ for blood test. Once the *decide* action is started, the system will schedule related patients to the corresponding test.

$$\begin{aligned} Patient \overset{def}{=} & (decide_1, \top).Patient_1 \\ & + (decide_2, \top).Patient_2 \\ & + (decide_3, \top).Patient_3 \end{aligned}$$

The first block of the above model represents the process of allocating incoming patients to one of the three tests based on the specified decision rate by using the "*choice*" operator in PEPA. This is the first decision in whole routing process. After the patients complete the first test, the second decision will be made for the next test.

$$\begin{aligned} Patient_1 & \overset{def}{=} (ecg, r_{ecg}).Patient_4 \\ Patient_4 & \overset{def}{=} (decide_2, \top).Patient_5 \\ & \quad + (decide_3, \top).Patient_6 \\ Patient_5 & \overset{def}{=} (xray, r_{xray}).Patient_7 \\ Patient_7 & \overset{def}{=} (blood, r_{blood}).Patient_8 \\ Patient_8 & \overset{def}{=} (wait, r_{wait}).Patient \\ Patient_6 & \overset{def}{=} (blood, r_{blood}).Patient_9 \\ Patient_9 & \overset{def}{=} (xray, r_{xray}).Patient_{10} \\ Patient_{10} & \overset{def}{=} (wait, r_{wait}).Patient \end{aligned}$$

For example, the above block follows the status of *Patient*$_1$. Patients complete ECG test and wait for the routing information. After receiving the decision made by the system, some patients go to the x-ray scan first, and then the blood test; conversely, the others go to the blood tests first, and then the x-ray scan. Once all tests are finished, the patients leave the hospital, and return to the start point after a time period which is specified by the *wait* action and its rate. For the following two blocks after the above one, they model the similar process but for the other two groups of patients which are separated in the first block.

3.2.3 Analysis of two models

Regarding the static and dynamic scheduling schemes, two different PEPA models are created in the previous section. The main difference between two schemes lies in the design of *Decide* components and the scheduling process. This section will compare these two schemes based on the measurements by solving ODEs. As noted before, the static scheme only supports the static scheduling process; however, the dynamic scheme can perform the real dynamic scheduling by using a set of function rates for the *decide* actions.

Factually, when the decision rates are specified with constants rather than functions, two schemes both perform the static scheduling process and their performance possibly is equivalent. If the two schemes do indeed have equivalent performance, then the dynamic scheme can used for the following modelling and analysis because of its ability to perform a dynamic scheduling with function rates.

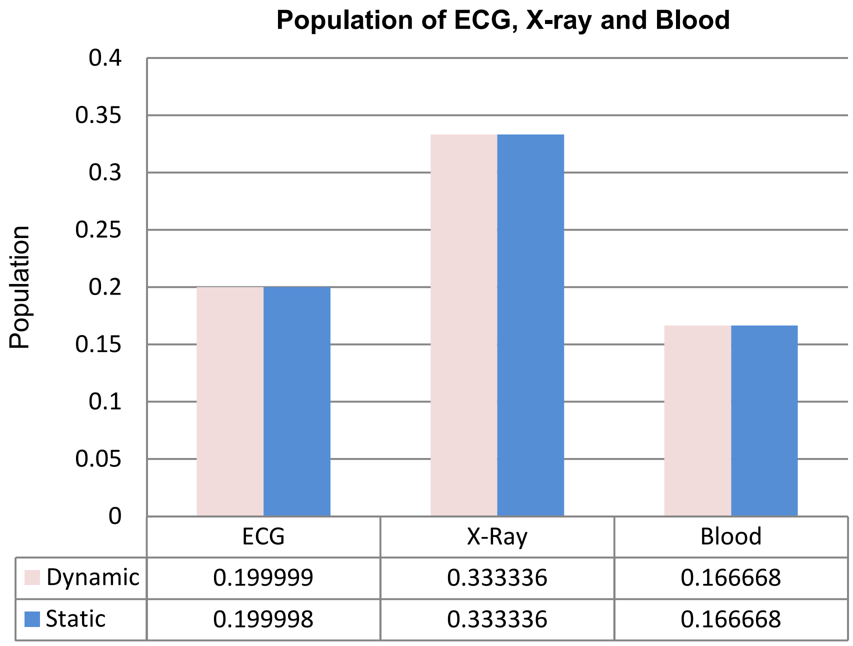
In order to verify the equivalency of the two schemes, it is necessary to compare the populations of two kinds of models at each test component: *ECG*, *X-ray* and *Blood*. Fluid flow analysis is adopted to derive the population by solving a set of ODEs.

Here, all parameters are initialized with the same value in both models, which ensures that models can be run on the same conditions. The parameters are defined as follows:

- The rates of *ecg*, *xray* and *blood* action are uniform: $r_{ecg} = 10.0$, $r_{blood} = 12.0$ and $r_{xray} = 6.0$. These parameter values are choosen in terms of the estimated service time of each test in the hospital. Usually, X-ray test is faster than other tests. The blood test needs the longest time in contrast to others.
- The rates of *decide* action, for both the static model and dynamic model, are unified with the value 120.0.
- The rate of *wait* action is 2.0.
- The number of instance at *Patient* component is 100.
- Any other component has only one instance.

The ODE analysis is executed at the latest Eclipse PEPA Plug-in platform. The type of analysis is set to steady state; the ODE analysis time is set from 0 to 500; number of time points is 1000; absolute tolerance is 1.0E-8; relative tolerance is 1.0E-4; and steady-state convergence norm is 1.0E-6.

The population at the component *ECG*, *X-ray* and *Blood* is displayed as:



	ECG	X-Ray	Blood
▭ Dynamic	0.199999	0.333336	0.166668
■ Static	0.199998	0.333336	0.166668

Figure 3.2.3 Populations of ECG, X-Ray and Blood

As displayed in Figure 3.2.3, the population of two schemes is equivalent at each service component, which means the static scheme and dynamic scheme have equivalent performance when the decision rates are specified as constants. Hence, it can be concluded that the dynamic model are equivalent to the static model when the model uses constant rates for the *decide* actions.

To observe the real dynamic scheduling process, this decision rate must be specified as a function rate. Thus, the dynamic scheme should be used for the future modelling as it supports the dynamic scheduling process by using function rates and also has equivalent performance with the static scheme when it uses constant rates. Hence, this dynamic scheduling scheme will be applied for the further modelling work from the next section.

3.3 Preliminary scheduling model analysis

This section uses the dynamic model introduced above to achieve the static scheduling process and the dynamic scheduling process. As explained in Section 3.2.3, the dynamic model can support both the static scheduling and dynamic scheduling policies. When the rates of decide actions are constant, the dynamic model scheme models a static scheduling process; when the rates of decide actions are defined as functions, the dynamic model scheme represents a dynamic scheduling process. Hence, a constant rate and a functional rate are adopted to achieve the static scheduling and the dynamic scheduling process respectively.

For the static scheduling process, the rate expression is defined as:

$$rate\ of\ decide = \ Constant\ Value$$

For the dynamic scheduling process, the basic expression of function rate is defined as follows:

$$rate\ of\ decide = \frac{Constant\ Value}{Transient\ Queue\ Length}$$

More details of these two types of rate will be introduced later.

To carry out the analysis of models, PEPA Plug-in will be used as the main analysis tool which is an embedded plug-in for the Eclipse platform. As this PEPA Plug-in cannot execute the calculation of function rate in the model run, the PEPA model has to be converted to an equivalent CMDL model by the tool. In the following section, all measurements and analysis are implemented based on the CMDL model.

3.3.1 Model transformation

As explained above, PEPA Plug-in cannot analyze the models using function rates, but the CMDL model can support the analysis for the function rates. Thus, this section will introduce the converted CMDL model from the original PEPA model as well as the use of the function rates.

Model transformation from PEPA to CMDL can be processed automatically by the Eclipse PEPA Plug-in. The transformed CMDL model is equivalent to the original PEPA model. The following block of statements is a part of CMDL model referring to the *decide* actions.

```
decide11, Patient11 + Decide1 → Patient12 + Decide1, [Patient11*r_d1*Decide1];
decide12, Patient18 + Decide1 → Patient19 + Decide1, [Patient18*r_d1*Decide1];
decide13, Patient     + Decide1 → Patient1   + Decide1, [Patient*r_d1*Decide1];

decide21, Patient + Decide2 → Patient2 + Decide2, [Patient*r_d2*Decide2];
decide22, Patient18 + Decide2 → Patient20 + Decide2, [Patient18*r_d2*Decide2];
decide23, Patient4 + Decide2 → Patient5 + Decide2, [Patient4*r_d2*Decide2];

decide31, Patient11 + Decide3 → Patient13 + Decide3, [Patient11*r_d3*Decide3];
decide32, Patient4 + Decide3 → Patient6 + Decide3, [Patient4*r_d3*Decide3];
decide33, Patient + Decide3 → Patient3 + Decide3, [Patient*r_d3*Decide3];
```

In the above CMDL model, each statement has three sections, in which the first section is the name of reaction that actually is the action type in PEPA model; the middle section represents the reaction process that reflects the change of states in PEPA model; the last section in the square brackets is the rate of the reaction.

The key points of the model are the *decide* action and its rate that control the race of deciding action. In the CMDL model, the previous *decide* action is divided into three sub-actions, because each *decide* action in the PEPA model has three cooperations with those *Patient* components. The rate expression in each single reaction represents the actual decision rate in this reaction. As each *decide* action has three cooperations in the PEPA model, for example *decide1*, its three cooperations are stated with three separated reactions in CMDL, see the first block of the above model. In the CMDL model, the rate of *decide1* is no longer *r_d1.* This is because the *decide* action in the PEPA model is a passive action, so its rate is a passive rate. Hence, in its converted CMDL model, the rate of each single reaction becomes the below expression:

$$Current\ Patient\ Population \times Constant\ Decision\ Rate \times Decide\ Component\ Number$$

The value of the above expression is the actual reaction rate in the CMDL model. In the static scheduling model, all decision rates are constants. To enable the dynamic scheduling process, a functional rate must be employed in the CMDL model.

In order to optimize the scheduling process, the system needs to control the patient dispatching process. When the waiting queue of a component, for example ECG, is growing, the system should prevent more patients from entering this component, and then schedule them to other components with less waiting queue, such as X-Ray or Blood. In the model, this process can be achieved by invoking a function rate that is varying against the total population at the current component. For example, when the total population of ECG is increasing, the rate of *decide1* should be reduced in terms of an appropriate function rate; conversely, the rate should be increased while the total population declines. In this case, a hyperbolic function can be used to alter the decision rate.

According to the above design and the CMDL model, a hyperbolic function is applied instead of the previous constant decision rate. The new decision rate expression becomes:

$$Transient\ Patient\ Population \times \frac{Constant\ Decision\ Rate}{Transient\ Queue\ Length} \times Decide\ Component\ Number$$

In terms of the above rate expression, the CMDL model with the dynamic scheduling should be refined as follows:

```
  decide11, Patient11 + Decide1 → Patient12 + Decide1,
[Patient11*(r_d1/(Patient1+Patient12+Patient16+Patient19+Patient23+c))*Decide1];
  decide12, Patient18 + Decide1 → Patient19 + Decide1,
[Patient18*(r_d1/(Patient1+Patient12+Patient16+Patient19+Patient23+c))*Decide1];
  decide13, Patient    + Decide1 → Patient1   + Decide1,
[Patient*(r_d1/(Patient1+Patient12+Patient16+Patient19+Patient23+c))*Decide1];


  decide21, Patient + Decide2 → Patient2 + Decide2,
[Patient*(r_d2/(Patient5+Patient9+Patient2+Patient21+Patient20+c))*Decide2];
  decide22, Patient18 + Decide2 → Patient20 + Decide2,
[Patient18*(r_d2/(Patient5+Patient9+Patient2+Patient21+Patient20+c))*Decide2];
  decide23, Patient4 + Decide2 → Patient5 + Decide2,
[Patient4*(r_d2/(Patient5+Patient9+Patient2+Patient21+Patient20+c))*Decide2];
```

```
   decide31, Patient11 + Decide3 → Patient13 + Decide3,
[Patient11*(r_d3/(Patient7+Patient6+Patient14+Patient13+Patient3+c))*Decide3];
   decide32, Patient4 + Decide3 →Patient6 + Decide3,
[Patient4*(r_d3/(Patient7+Patient6+Patient14+Patient13+Patient3+c))*Decide3];
   decide33, Patient + Decide3 → Patient3 + Decide3,
[Patient*(r_d3/(Patient7+Patient6+Patient14+Patient13+Patient3+c))*Decide3];
```

In the above CMDL model, the previous constant rate is replaced by a hyperbolic based function rate, and the variables of the rate function are the value of the total population at *ECG* component. Moreover, in the function rate expression, *c* is just a constant with a tiny value used to avoid zero (and hence an execution error) in the denominator of hyperbolic function.

3.3.2 Model analysis

The analysis of CMDL model will be carried out to explore the performance of the dynamic scheduling policy with function rate comparing with the static scheduling policy.

In order to facilitate the analysis, initial conditions of two types of policies should be uniformed. The values of these parameters are defined based on the estimated service time of each test in the hospital. Here, related parameters of the model can initialized as follows:

$r_{ecg} = 10.0;$ $r_{d1} = 100.0;$ $r_{wait} = 2.0;$

$r_{xray} = 6.0;$ $r_{d2} = 60.0;$

$r_{blood} = 12.0;$ $r_{d3} = 120.0;$

In the first analysis, we assume the number of components as:

Patient = 100,

ECG = 1, Blood = 1, XRay = 1,

$\text{Decide}_1 = 1$, $\text{Decide}_2 = 1$, $\text{Decide}_3 = 1$.

To obtain the performance of each policy, ODE analysis is adopted to capture the population of each component. In ODE analysis, the adaptive step-size 5th order

Dormand Prince ODE solver is used, which is provide by Eclipse PEPA Plug-in platform. In the ODE analysis, time ranges from 0 to 30, with 1000 data points, a step size of 1.0E-5, and relative and absolute error equal to 1.0E-4. With these conditions, the ODE analysis generates the following results:

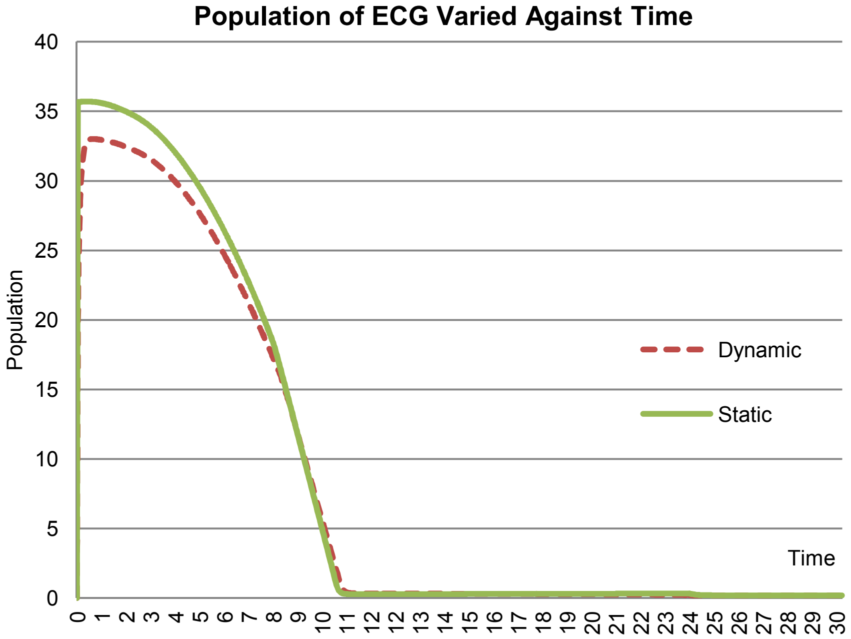


Figure 3.3.1 Population of dynamic and static scheduling policy at ECG component (with 1 X-Ray Components)

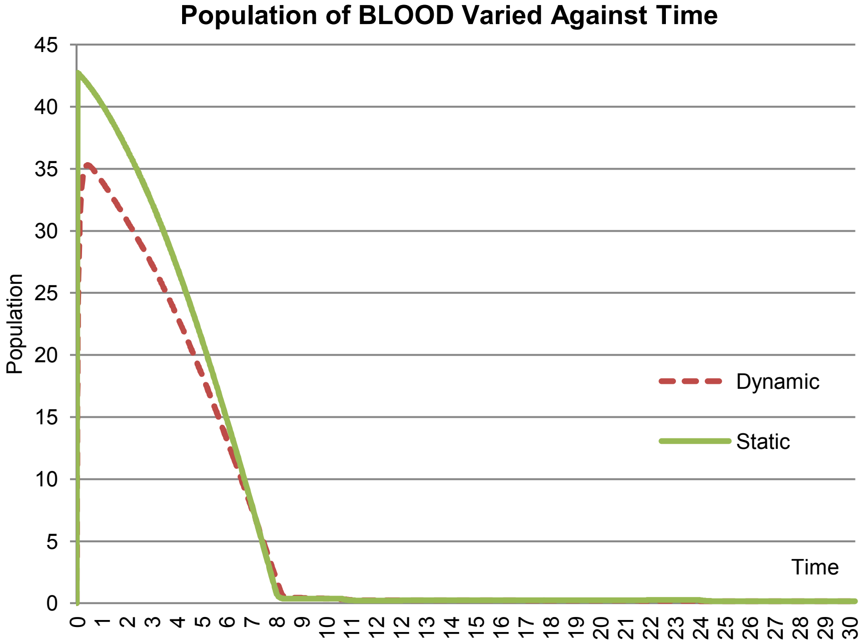


Figure 3.3.2 Population of dynamic and static scheduling policy at Blood component (with 1 X-Ray Components)

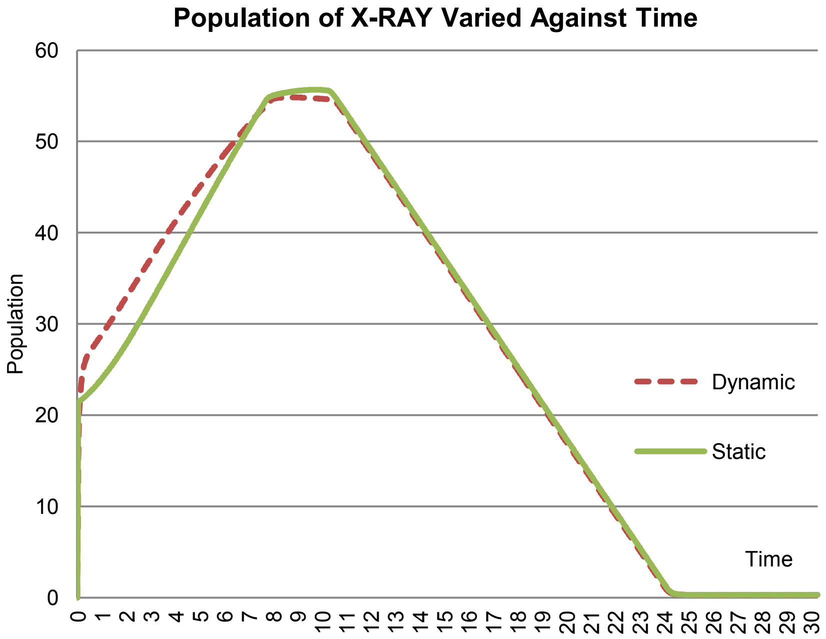


Figure 3.3.3 Population of dynamic and static scheduling policy at X-ray component (with 1 X-Ray Components)

Figure 3.3.1, 3.3.2 and 3.3.3 plot the population of both dynamic and static policies varying against the simulation time. As can be seen from the three charts, the patient population at each component has a dramatic growth at the beginning, and then reaches the peak; finally, all lines come to the convergence when the system approaches the steady state.

In Figure 3.3.1 and 3.3.2, it is clear that lines for the dynamic scheduling policy sits below the static scheduling policy during the whole time interval before convergence. At the beginning, there is tiny difference between two policies because of the rapid growth; however, in fact, the dynamic policy still performs slightly better. In Figure 3.3.3, however, the line of the dynamic policy appears in a higher level at certain times before the peak time. According to the figures displayed, it can be conclude that dynamic scheduling policy generates less population than the static policy at *ECG* and *Blood* component before the convergence; however, at the component *X-Ray*, the population of dynamic policy is higher than that of the static policy before reaching the peak.

As it is assumed that the rate of *xray* action is the smallest compared with the rates of *blood* and *ecg* actions, an initial hypothesis is that the dynamic policy finds the optimal performance before convergence by overloading the action having the slowest rate. In order to verify this hypothesis, further analysis will be generated by altering the initial conditions.

In the subsequent analysis, the number of *X-Ray* and *Decide*$_2$ components is increased from 1 to 3 respectively, which aims to raise the total rate capacity (namely,$rate\ capacity\ = number\ of\ instance\ of\ Xray\ \times rate\ of\ Xray$) of *X-Ray* from 6.0 to 18.0.

Patient = 100,

ECG = 1, Blood = 1, XRay = 3,

Decide$_1$ = 1, Decide$_2$ = 1, Decide$_3$ = 3.

All other action rates remain the same as before. Thus, in current situation, the rate of *ecg* action is the smallest that is 10.0. Here are the figures from ODE analysis:

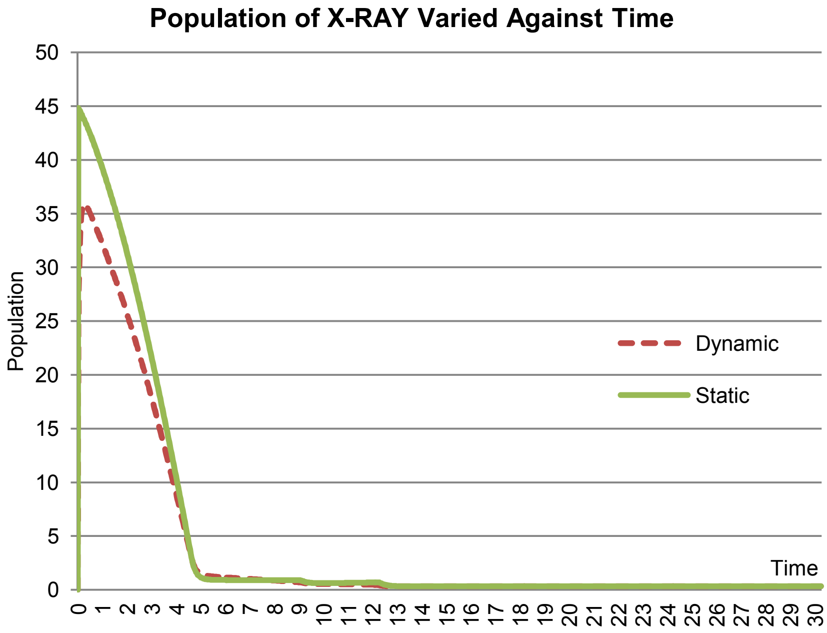


Figure 3.3.4 Population of dynamic and static scheduling policy at X-ray component (with 3 X-Ray Components)

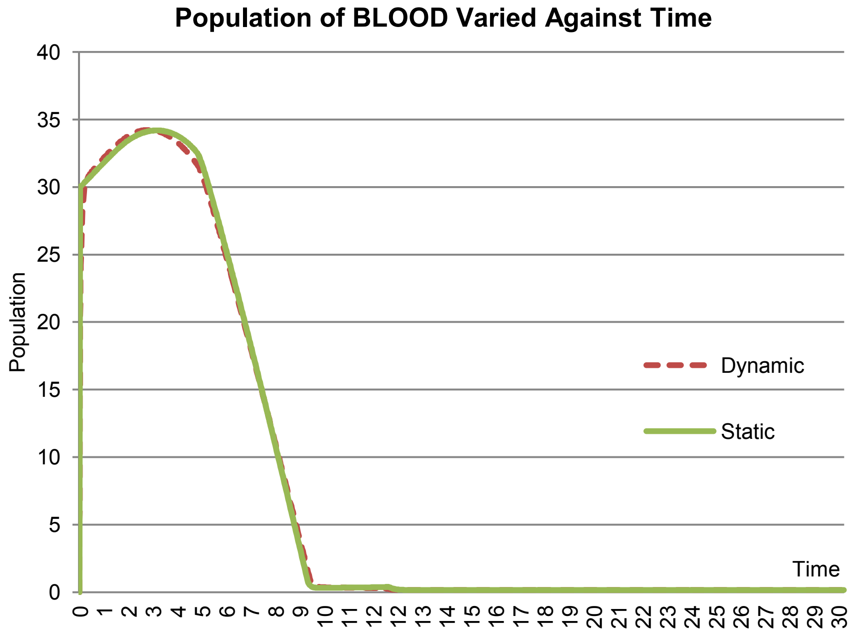


Figure 3.3.5 Population of dynamic and static scheduling policy at Blood component (with 3 X-Ray Components)

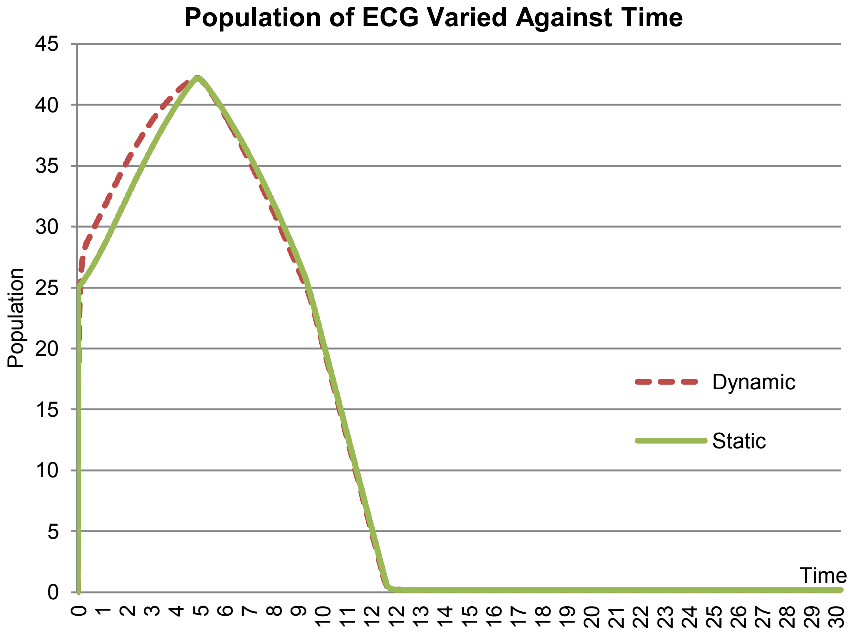


Figure 3.3.6 Population of dynamic and static scheduling policy at ECG component (with 3 X-Ray Components)

From Figure 3.3.4, 3.3.5 and 3.3.6, it is obvious that the status of lines have altered with the change of conditions. Now, the dynamic policy has the best performance at component *X-Ray* in contrast to the static policy, see Figure 3.3.4. This is because the rate capacity of *X-Ray* has the largest value compared with other two tests. However, *ECG* component has the smallest action rate; in Figure 3.3.6, the dynamic policy loses its optimal performance, because the population of the dynamic policy is greater than that of the static policy during a certain time interval before the peak. For the component *Blood* in Figure 3.3.5, two lines almost match together, which denotes two policies have similar performance at this component.

Hence, there is no doubt that the dynamic scheduling policy loses its optimal performance at the components with smaller action rate, especially the component with the smallest action rate. However, the dynamic scheduling policy obtains the optimal performance evidently at the component with the largest action rate.

3.3.3 Further analysis

In the previous section, the related analysis indicates that the dynamic scheduling policy has inferior performance at the component with smaller rate. Since the apparent problem is caused by the rate value of a component in the system, can the problem be solved by eliminating the difference between the action rates of components? This section aims to find the answer of this question.

To eliminate the difference between action rates of three components, the initial conditions can be modified as follows:

$r_{ecg} = 10.0;$ $r_{d1} = 100.0;$ $r_{wait} = 2.0;$

$r_{xray} = 10.0;$ $r_{d2} = 100.0;$

$r_{blood} = 10.0;$ $r_{d3} = 100.0;$

The values used in this sub-section are just assumed in terms of the approximated average rate value of three tests in order to obtain the equivalent rate of each test and to verify the previous question.

In addition, the number of components is set the same as before:

Patient = 100,

ECG = 1, Blood = 1, XRay = 1,

$Decide_1 = 1$, $Decide_2 = 1$, $Decide_3 = 1$.

These new conditions guarantee that no difference exists among the rates of *ECG*, *Blood* and *X-Ray* components. ODE analysis setting is in keeping with the preceding. Figures on current conditions are as follows:

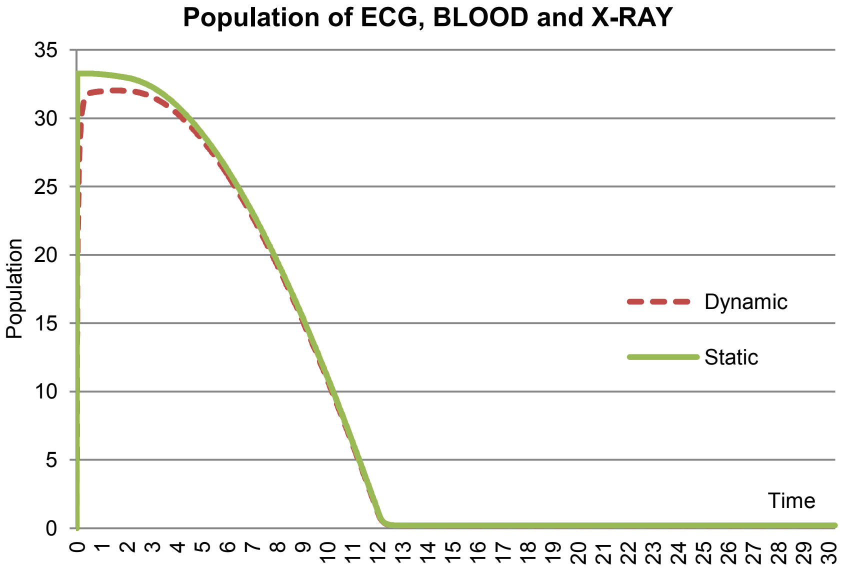


Figure 3.3.7 Population of dynamic and static policy at ECG, Blood and X-Ray (yime: 0~30)

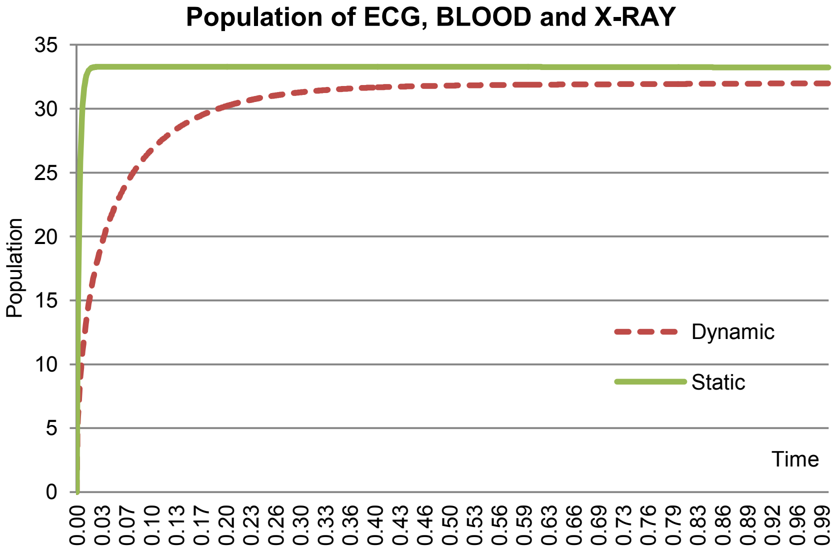


Figure 3.3.8 Population of dynamic and static policy at ECG, Blood and X-Ray (time: 0~1)

As noted before, in this situation, ECG, Blood and X-Ray have been specified with the same action rate, so the plots for ECG, Blood and X-Ray are completely the same, which is displayed in Figure 3.3.7. Furthermore, Figure 3.3.8 shows the details of Figure 3.3.7 within the time interval from 0 to 1.

Figure 3.3.7 and 3.3.8 clearly show that the dynamic scheduling policy has less population than the static scheduling policy during the whole time interval before the convergence, when all the components have identical rates. In other words, the dynamic scheduling policy has the ability to provide the optimal performance in scheduling process at each component of the system, when all these components have equivalent service ability.

Consequently, to gain the best performance of the dynamic scheduling policy, a good way is to adjust the rate of system components to approach almost the same value.

3.3.4 Summary

The goal of this section is to explore the performance of static scheduling policy and dynamic scheduling policy in a smart environment. The study is carried out by modelling the patient flow with both policies, and then the population at each component is analyzed to evaluate the performance of two kinds of scheduling policies. According to the figures and analysis, there are two conclusions which can be obtained as follows:

Firstly, the dynamic scheduling policy is able to perform better than the static policy only when the system is not in the steady state. This means that if the system approaches a steady state that has the mean population close to zero due to the modelling assumptions, the dynamic scheduling policy will lose the optimal performance and show a similar performance compared with the static policy. Hence, the dynamic scheduling policy is suitable for the system without a steady state, or the system with a steady state which has the mean value not close to zero. In other words, the dynamic scheduling policy is best for the system having stable queues or dynamic queues in a period.

Secondly, the dynamic scheduling policy has the optimal performance at all participative components only under the conditions that each participative component has quite close or equivalent action rate comparing with each other. It denotes that when the dynamic scheduling policy is applied in the system, it is necessary to consider adjusting the rate of each component to a same level to obtain the optimal performance of the dynamic policy.

In the preliminary dynamic model, incoming patients arrive at a constant arrival rate and each department serves patients at a specified constant service rate. Meanwhile, all patients is assumed to take the same tests. Factually, regarding the real hospital scenario, patients usually arrive the hospital with a varying rate during each working day, and hospital departments also have variable service ability. Furthermore, patients usually have differet examinations and treatments in the hospital. According to these actual issues, it is necessary to change the model conditions to achieve different scenarios, and the dynamic scheduling policy must be re-evaluated under the new conditions. The next section will explore the performance of the dynamic scheduling policy with varying arrival rate and service rate.

3.4 Refined dynamic scheduling model scenario with novel dynamic scheduling policy

The dynamic scheduling policy gives a better performance when the hospital system has stable arrivals and stable service ability. In order to explore the applicability of dynamic policy in greater depth, the previous original scenario needs to be updated, as well as the model and its parameters.

Regarding the new scenario, firstly, for ease of exposition, it is assumed that only two tests will be performed on patients: blood test and x-ray scan. Clearly, patients may take different tests in the hospital, but these are not included in the model presented here.. Thus, there are three classes of patients: those who only need a blood test, those who only need an x-ray scan, and, finally, those who need both. Secondly, new arrivals and service are no long generated with constant rates. Periodically varied function rate is used to model the dynamic change of the incoming patient flow and service ability of departments. The main target of this new case is to help those patients who need both tests to decide which test comes first.

In addition, a novel dynamic scheduling policy will be introduced, which is based on comparing transient response time with the average response time. Thus, in this subsection, there is an evaluation of the performance of a static scheduling policy and two dynamic scheduling policies.

3.4.1 Updated scheduling model

According to the preceding model specification, the updated model has three patient classes: *PatientB* (*Blood* test only), *PatientX* (*X-ray* test only) and *PatientBX* (both *Blood* and *X-ray*). Each group of patients has its own arrival rate. Group *PatientB* will be sent to a blood test directly and the *PatientX* group will be directed to the x-ray test on arrival. For group *PatientBX*, patients will be sent to either the blood test or the x-ray test first depending on the current queuing situation of both tests. The details of the updated model in PEPA are stated as:

$$\begin{aligned}
PatientB &\overset{def}{=} (arriveB,\ r_{arrive\_b}).PatientB_1 \\
PatientB_1 &\overset{def}{=} (register,\ r_{register}).PatientB_2 \\
PatientB_2 &\overset{def}{=} (blood,\ r_{blood}).PatientB_3 \\
PatientB_3 &\overset{def}{=} (wait,\ r_{wait}).PatientB
\end{aligned}$$

$$\begin{aligned}
PatientX &\overset{def}{=} (arriveX,\ r_{arrive\_x}).PatientX_1 \\
PatientX_1 &\overset{def}{=} (register,\ r_{register}).PatientX_2 \\
PatientX_2 &\overset{def}{=} (xray,\ r_{xray}).PatientX_3 \\
PatientX_3 &\overset{def}{=} (wait,\ r_{wait}).PatientX
\end{aligned}$$

$$\begin{aligned}
PatientBX &\overset{def}{=} (arriveBX,\ r_{arrive\_bx}).PatientBX_1 \\
PatientBX_1 &\overset{def}{=} (register,\ r_{register}).PatientBX_2 \\
PatientBX_2 &\overset{def}{=} (decideB,\ \top).PatientBX_3 \\
&\quad + (decideX,\ \top).PatientBX_4 \\
PatientBX_3 &\overset{def}{=} (blood,\ r_{blood}).PatientBX_5
\end{aligned}$$

$$PatientBX_5 \overset{def}{=} (xray, r_{xray}).PatientBX_6$$

$$PatientBX_6 \overset{def}{=} (wait, r_{wait}).PatientBX$$

$$PatientBX_4 \overset{def}{=} (xray, r_{xray}).PatientBX_7$$

$$PatientBX_7 \overset{def}{=} (blood, r_{blood}).PatientBX_8$$

$$PatientBX_8 \overset{def}{=} (wait, r_{wait}).PatientBX$$

$$Blood \overset{def}{=} (blood, r_{blood}).Blood$$

$$Xray \overset{def}{=} (xray, r_{xray}).Xray$$

$$Reception \overset{def}{=} (register, r_{register}).Reception$$

$$DecideB \overset{def}{=} (decideB, r_{db}).DecideB$$

$$DecideX \overset{def}{=} (decideX, r_{dx}).DecideX$$

$$Wait \overset{def}{=} (wait, r_{wait}).Wait$$

$$PatientB[N_1]||PatientX[N_2]||PatientBX[N_3] \bowtie_L$$

$$Blood||Xray||Reception||DecideB||DecideX||Wait$$

$$L = \{register, blood, xray, decideB, decideX, wait\}$$

The above model clearly presents the updated hospital scenario. In addition, a periodic function is also applied for arrival rate and service rate so as to model varying system arrivals and service ability. Here, two trigonometric functions, sine and cosine, are used to fluctuate the arrival rate and the service rate. To compare the performance in equivalent conditions, the static and dynamic scheduling policies are evaluated in the conditions of the same function arrival rate and service rate.

3.4.2 Scheduling policies

In Section 3.3, the preliminary analysis of two scheduling policies is completed by evaluating the performance of a static policy and a dynamic policy. Here, a new dynamic scheduling policy will be introduced together with the previous two kinds of scheduling policies. The new dynamic scheduling policy is noted as dynamic scheduling policy 2 which will be detailed in Section 3.4.2.3.

3.4.2.1 Static scheduling policy

The static scheduling policy is a method which could schedule incoming clients to the right destination with a stable rate in the scheduling process. The static scheduling policy is a simple policy in contrast to others. According to the scheduling model adopted in Section 3.4.1, the static scheduling policy is achieved by specifying a constant rate for the action *decideB* and *decideX* respectively. For the definitions,

$$DecideB \overset{def}{=} (decideB,\ r_{db}).DecideB$$

$$DecideX \overset{def}{=} (decideX,\ r_{dx}).DecideX$$

when the static scheduling policy is applied, the rates *r_db* and *r_dx* must be constants, which mean the scheduling process runs stably without any changing.

$$Decision\ Rate\ of\ Static\ Policy = Constant\ Value$$

3.4.2.2 Dynamic scheduling policy 1 – based on queue length

The dynamic scheduling policy is a way to dynamically adjust the scheduling process by varying related scheduling rates. In contrast to the static scheduling, the dynamic scheduling policy has the ability to change the scheduling rates in terms of the transient queuing situation.

As introduced in Section 3.3, dynamic scheduling policy 1 uses transient queue length to determine scheduling rate so as to manage the scheduling process dynamically. According to this policy, when the queue length increases, this means that a growing number of clients are waiting for service, the scheduling rate should reduce with the increment of queue length. In policy 1, a hyperbolic function is adopted to create a function rate to control the scheduling process. The variable of the function rate is the value of transient queue length. Then the value of the function is used as the rate of scheduling, which is the rate of "*decide*" action. When the queue length at a component increases, the rate of related "*decide*" action will reduce in order to prevent patients coming to this component. The expression of function rate for policy 1 is:

$$Decision\ Rate\ of\ Dynamic\ Policy\ 1 = \frac{Constant\ Value}{Transient\ Queue\ Length}$$

3.4.2.3 Dynamic scheduling policy 2 – based on response time

The dynamic scheduling policy 2 controls the scheduling process based on the change of ratio of average response time and transient response time. For instance, when a patient who needs both blood and x-ray examinations comes to the hospital, the system immediately estimates transient response time at blood and x-ray tests respectively and then compares the transient response time with the average response time. If the transient response time is longer than the average response time, this means the number of patients at this test is higher than average level; hence, it is necessary to restrain the sending of more to this test. Conversely, if the transient response time does not express the average response time, the system should send more patients for this test. Here, a function P is set which equals the ratio of average response time and transient response time.

$$P = \frac{Average\ Response\ Time}{Transient\ Response\ Time}$$

In the above formula, average response time is a fixed value, which can be calculated once the parameters of the model are initialized. However, the transient response time varies with time. The value of the transient response time can be calculated during the run of the model, too. Details about how to calculate transient response time and average response time will be introduced later. According to the formula, as average response time is fixed, if transient response time increases (P decreases), the rate of incoming patients, namely the rate of "*decide*" action in the model, must be reduced; conversely, the rate should be increased when the transient response time becomes less (P increases). Hence, in order to manage patient flow in the scheduling process, the function rate of "*decide*" action can be set up as:

$$Decision\ Rate\ of\ Dynamic\ Policy\ 2 = P\ \times\ Constant\ Decision\ Rate$$

In order to carry out the above function rate in model analysis, the value of P must be calculated first. For this reason, methods to get transient response time and average response time should be found.

Before finding out transient response time and average response time, it is necessary to introduce a concept which will be used to calculate response time, namely $M|M|1||K$ queue model (or terminal model, see Figure 3.4.1).

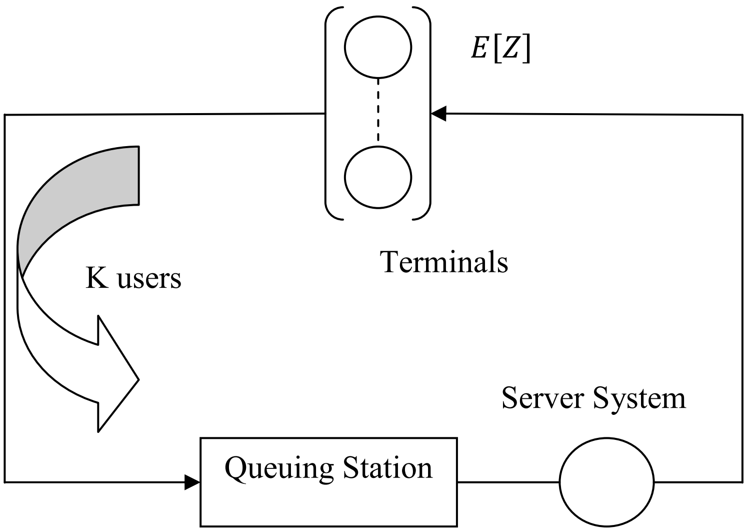


Figure 3.4.1 Simple terminal model

As introduced in Haverkort's book [21], in the $M|M|1||K$ terminal model, there are K system users sitting behind their terminals. These users send requests after exponentially distributed think time Z with mean $E[Z] = \frac{1}{\lambda}$. In the terminal model, the more users there are in the thinking state, the higher the effective completion rate of thinkers is; this means, the effective arrival rate at the system is proportional to the number of thinkers. Thus, in the model, users can be modelled as infinite servers, where each user has his or her own server. Once requests have been submitted by users, they only need wait until the response. The system completes the job with mean time $E[S] = \frac{1}{\mu}$. Such a so-called *terminal model* is an example of a closed queuing network with a population of K customers circling between terminals and server system. In fact, the patient flow scheduling model here is a typical and complex terminal model. Now the transient response time and the average response can be found in terms of the terminal model.

In this terminal model, a single server system is modelled, which means each type of server only serves one client at a time. Thus, the transient response time $T[R]$ at the server system equals the sum of transient waiting $T[W]$ time for all customers in the queuing station and the transient service time $T[S]$ for a single client in the server, namely $T[R] = T[W] + T[S]$. Here, arrival rate is represented by λ, service rate by μ, transient queue length by l. The formula for calculating transient response time is stated as equation (1):

$$T[R] = T[W] + T[S] = \frac{l}{\mu} + \frac{1}{\mu} = \frac{(l+1)}{\mu} = (l+1)\,E[S] \dots\dots\dots (1)$$

The average response time at the server system is measured with a different method. This method, known as *mean-value analysis*, can be applied in many cases to more general queuing models, such as our scheduling model. First of all, we must address the average response time at terminals $E[R_t]$ and at the server system $E[R_s(K)]$. In an infinite-server queuing station, every job has its server; thus, no queuing or waiting occurs. Therefore, $E[R_t] = E[Z]$, independently of the number of K clients actually present. For processing the server system, however, the average response time depends on the number of K clients.

To compute $E[R_s(K)]$, average circle time must be introduced as $E[C(K)] = E[R_t] + E[R_s(K)] = E[Z] + E[R_s(K)]$. Here, $E[C(K)]$ expresses the mean time it takes for a client to go once through the cycle “think-serve”. The throughput $X[K]$ can now be represented as $X[K] = \frac{K}{E[C(K)]}$, which is the product of the frequency with which users cycle ($\frac{1}{E[C(K)]}$) and the number of users (K). Hence, combining the above two expressions, we obtain equation (2):

$$X[K] = \frac{K}{E[C(K)]} = \frac{K}{(E[Z] + E[R_s(K)])} \dots\dots\dots (2)$$

From equation (2), we derive the *response time law*:

$$E[R_s(K)] = \frac{K}{X[K]} - E[Z] \dots\dots\dots (3)$$

As we know, the throughput $X[K]$ equals the product of the server-busy probability and the service rate of the system, as $X[K] = (1 - p_0) \times \mu = \frac{1-p_o}{E[S]}$. However, if we change the value of K, p_0 will change as well; therefore, we need to write p_0 as a function of K. In summary, this produces equation (4):

$$E[R_s(K)] = \frac{K \times E[S]}{1 - p_0(K)} - E[Z] \ldots\ldots\ldots (4)$$

In this case, the aim is to model patient flow scheduling process with a massive number of patients in the system; thus, the value of K is set up on a large scale. For large values of K, the idle fraction is very small so that the denominator $1 - p_0(K)$ will approach 1. Consequently, for large K, the following approximation is obtained as equation (5):

$$E[R_s(K)] \approx K \times E[S] - E[Z] \ldots\ldots\ldots (5)$$

So far, we have already obtained both transient response time $T[R]$ and average response time $E[R_s(K)]$. As noted at the beginning of this sub-section, the new scheduling policy uses a ratio function P to regulate the rate of "decide" action so as to schedule patients in the system efficiently and dynamically. The previous expression of P now can be substituted with equation (1) $T[R]$ and equation (5) $E[R_s(K)]$. Overall, the rate of "*decide*" action is expressed as:

$$Function\ Rate\ 2 = \begin{cases} P \times Constant\ Decision\ Rate, & for\ dynamic\ scheduling \\ \quad\ Constant\ Decision\ Rate, & for\ static\ scheduling \end{cases},$$

if and only if

$$P = \frac{E[R_s(K)]}{T[R]} = \frac{K \times E[S] - E[Z]}{(l+1)\, E[S]} \ldots\ldots\ldots (6)$$

3.5 ODE analysis

In this section the three scheduling policies based on the new model scenario will be analysed. Analysis focuses on the comparison between static and two dynamic policies. To obtain the performance of each policy, fluid flow analysis is applied in the model analysis by solving a set of ordinary differential equations which come from the original PEPA model. The analysis places extra emphasis on a large number of clients. In this situation, the system keeps running to serve patients without a stop; in other words, there are always a large number of patients in the hospital. Thus, the patient flow approximates to a fluid flow rather than a discrete flow. In the following analysis, the model components are initialized as:

PatientB = 1000;   PatientX = 1000;   PatientBX = 2000;
Blood = 1;   Xray = 1;
DecideB = 1;   DecideX = 1;

The total number of patients is 4000. Such a large scale of instances guarantees the accuracy of fluid flow analysis, although it may not be a realistic operating assumption in practice. Except for patient components, all other components are initialized to 1, in which each testing process in the hospital is a single server system.

In order to model the real situation, the following analysis sets up four condition groups based on arrival rate and service rate:

- Stable arrival rate and service rate. This models an ideal situation that the incoming patients arrive in the hospital with a constant rate.
- Varying arrival rate and stable service rate. This means patients arrive in the hospital with a changing rate and models the peak time situation in the hospital.
- Stable arrival rate and varying service rate. This means the service in the hospital is also changing with the time which meets the real situation in the hospital because the hospital staff cannot continuously.
- Varying arrival rate and service rate. This models a complex hospital environment with changing incoming patients and unstable serving ability. This situation is the closest approach to the real situation in the hospital.

These four groups of conditions are used to model different hospital scenarios, especially that patient flow or hospital service changes with time during a day. To vary the arrival rate and service rate, both sine and cosine function are adopted for arrival rate and service rate to alter their values over time.

3.5.1 Analysis with stable arrival rate and service rate

In this sub-section, analysis is carried out of a set of stable conditions: constant arrival rate and service rate. Constant arrival rate means the number of incoming arrivals stays around a mean value which is the defined constant arrival rate; and it the constant service rate is similar with the constant arrival rate, which stands for service ability being kept at a constant mean value.

3.5.1.1 Analysis conditions setup

First of all, we assume the value of arrival rate and service rate as:

$$r_{arriveB} = 400.0 \quad r_{arriveX} = 400.0 \quad r_{arriveBX} = 400.0$$

$$r_{blood} = 120.0 \quad r_{xray} = 80.0$$

Although both arrival rate and service rate are self-defined, they cannot be set at random. To guarantee the accuracy of fluid flow analysis, arrival rate must be greater than service rate. Otherwise, if service rate is greater than arrival rate, there may be no queue waiting at service component, which means the model is not fluid any more. Thus, the measurement from ODEs is not accurate. However, for different types of arrival rate or service rate, their values can be defined randomly. For example, here we assume all three patient groups have the same arrival rate, and of course, they can be set with different arrival rates. For service rate, we generally think blood test is faster than an x-ray scan, so the rate of blood test is defined as greater than that of an x-ray scan.

Secondly, for other rates in the model, e.g. rates of *register* and *wait* action, they are initialized in a uniform way:

$$r_{register} = 2000.0 \quad r_{wait} = 2000.0$$

Both rates of *register* and *wait* are set to a large value. Owing to such a large registration rate, patients can complete their registration very fast without generating a long queue at this action. Large waiting rates are used to ensure that patients re-enter hospital quickly after a short time outside the hospital, which guarantees the circularity of patient flow in the model.

Furthermore, the most important rate in the model is the rate of *decide*, which is used to model the scheduling process. For the static scheduling process, we set the rate of *decide* as a constant; for the dynamic scheduling process, the rate of *decide* must be a function being used to perform different scheduling policies. In order to compare static policy and two dynamic policies, we use the introduced rate expressions in the previous Section 3.4 and initialize them with suitable values.

- Initialization for static scheduling policy

$$Rate\ of\ decide\ for\ Blood: \qquad r_{db} = 1.0$$

$$Rate\ of\ decide\ for\ Xray: \qquad r_{dx} = 1.0$$

Here, the rate of "*decide*" action for both Blood and X-ray are initialized with a small constant value 1.0. Why is the value of decision rate so small? This is because the decide action is not just used to schedule patients to the right test, and also this action is able to prevent patients entering hospital when there is a long queue waiting for service. To avoid unnecessary growth in the number of patients queuing in the hospital, a better way is to block most patients outside the hospital until the service is soon available to them. Hence, a small value must be used for the rate of decide so as to control the number of patients entering hospital.

- Initialization for dynamic scheduling policy 1 based on queue length

$$Rate\ of\ decide\ for\ Blood:\ r_{db} = 1.0 \times \frac{1}{Transient\ Queue\ Length\ at\ Blood}$$

$$Rate\ of\ decide\ for\ Xray:\ \ r_{dx} = 1.0 \times \frac{1}{Transient\ Queue\ Length\ at\ Xray}$$

According to the definition of dynamic Scheduling policy 1, this policy adopts a hyperbolic function to alter the decision rate against the varying of queue length. From the above expression, function rate of "*decide*" action for policy 1 is the constant rate multiplied by a function, in which the constant is the same as the rate for static policy and the function is to change the final value with the varying queue length. The exact value of transient queue length can be directly obtained in the CMDL model.

- Initialization for dynamic scheduling policy 2 based on response time

The rate of "decide" action based on policy 2 is represented as:

$$Rate\ of\ decide = \frac{K \times E[S] - E[Z]}{(l+1)\ E[S]} \times C$$

According to the specific mode in this chapter, K represents the number of patients at each service component, such as Blood or X-ray; $E[S]$ is mean service time, which equals $\frac{1}{\mu}$; $E[Z]$ is the mean think time with the value of $\frac{1}{\lambda} + \frac{1}{r_{wait}} + \frac{1}{r_{register}}$ $(\lambda\ is\ total\ arrival\ rate)$; then, l is transient queue length that can be obtained directly from the CMDL model; finally, C is a constant decision rate being used for comparison with functional rate.

Here, for two types of decide action, rates are calculated with different parameters:

$Rate\ of\ decide\ for\ blood$:

$$r_{db} = \frac{K_{blood} \times E_{blood}[S] - E_{blood}[Z]}{(l_{blood}+1) \times E_{blood}[S]} \times C = \frac{3000 \times \frac{1}{120} - (\frac{1}{800} + \frac{1}{2000} + \frac{1}{2000})}{(l_{blood}+1) \times \frac{1}{120}} \times 1.0$$

$Rate\ of\ decide\ for\ xray$:

$$r_{dx} = \frac{K_{xray} \times E_{xray}[S] - E_{xray}[Z]}{(l_{xray}+1) \times E_{xray}[S]} \times C = \frac{3000 \times \frac{1}{80} - (\frac{1}{800} + \frac{1}{2000} + \frac{1}{2000})}{(l_{xray}+1) \times \frac{1}{80}} \times 1.0$$

Finally, for the ODE analysis, the adaptive step-size 5th order Dormand Prince ODE solver is used as provided in Eclipse PEPA Plug-in platform. In the ODE analysis, time ranged from 0 to 60, with 200 data points, a step size of 1.0E-5, relative error and absolute error equal to 1.0E-4.

3.5.1.2 Analysis results

According to the previous condition setup, the following figures were obtained from a single run under the condition that both arrival rate and service rate are stable:

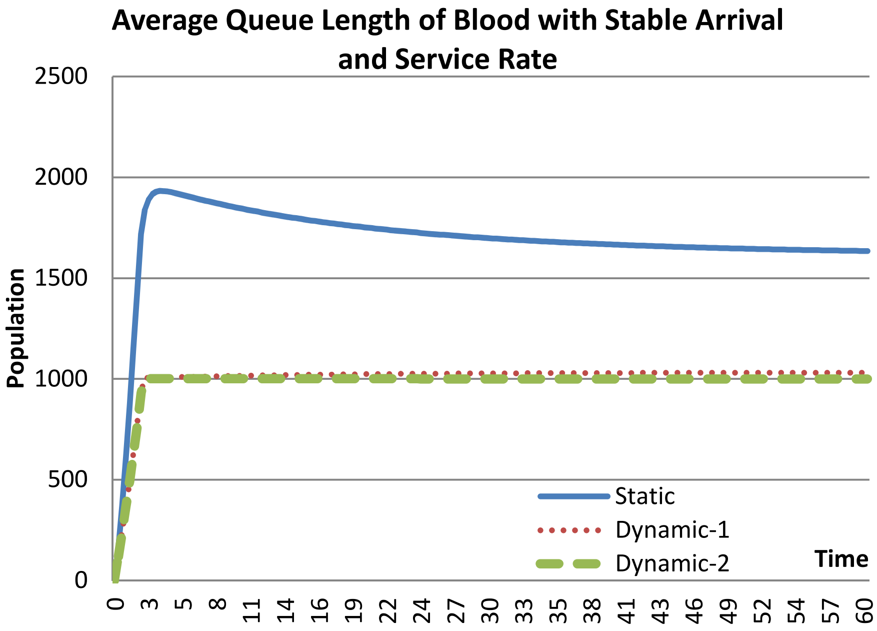


Figure 3.5.1 Average queue length of blood with stable arrival rate and service rate

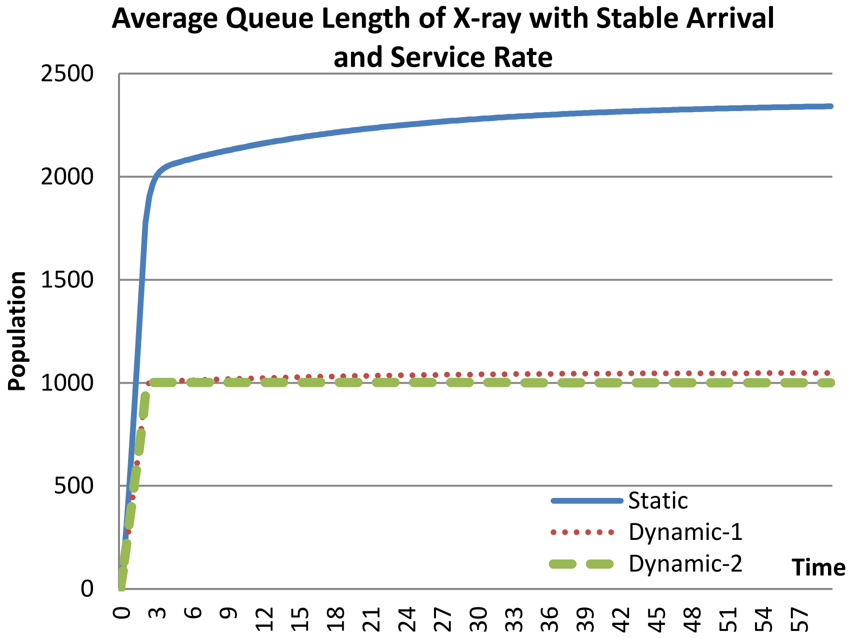


Figure 3.5.2 Average queue length of X-ray with stable arrival rate and service rate

Figure 3.5.1 and Figure 3.5.2 respectively display the average queue length for blood test and x-ray scan with three types of scheduling policies. The blue line represents the change of static policy; the red line and green line standing for dynamic policy 1 and dynamic policy 2 respectively. Both diagrams clearly show that dynamic policies have much less population in queue length compared with the static policy.

From ODE analysis figures at *Blood*, the queue length of dynamic policy 1 is around 1030; and the queue length of policy 2 is about 1000, which is a bit better than policy 1. However, the static policy has a population of roughly 1600 in queue. For *X-ray*, the queue length of dynamic policy 1 is about 1050, while dynamic policy 2 has a queue length of around 1000. For the static policy, however, the queue length at x-ray has a population of around 2340. Such a dramatic difference demonstrates that dynamic scheduling policies produce a much better performance than static scheduling policies. In addition, according to Figure 3.5.1 and Figure 3.5.2, when arrival rate and service are stable, dynamic policy 2 is a little better than policy 1. Thus, the conclusion may be drawn that dynamic scheduling policy 2 is more suitable for stable arrival and service conditions.

Finally, it is worth mentioning that, at the beginning, population for static policy increases faster than dynamic policies. This is because the rate of *decide* action in static policy is constant, so the speed of deciding action does not change with increase of queue length; however, rate in dynamic policies initially equals a constant value, but this rate is altered with the change of queue length. At the very beginning, the queue length at each component increases, and then such increase causes reduction in the function rates used in dynamic policies. Thus, dynamic policies have slower increase in queue length compared with the static policy.

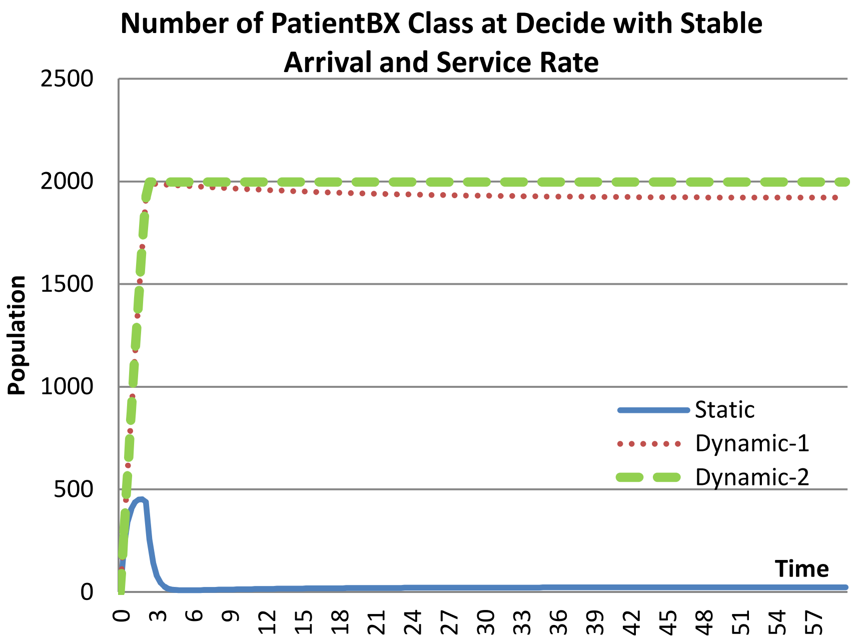


Figure 3.5.3 Population of PatientBX waiting at Decide component with stable arrival and service rate

Figure 3.5.3 represents how many patients who need both blood and x-ray tests are blocked outside the hospital by three scheduling policies. It is clear that dynamic policy 2 prevents about 2000 patients and policy 1 prevents around 1900; however, only about 20 patients are prevented by the static policy. These results mean that the static policy allows almost all patients to enter the hospital and wait there for service, but dynamic policies allow just a small number of patients into hospital so as to avoid a long waiting queue in the hospital.

3.5.2 Analysis with varying arrival rate and stable service rate

Analysis in this sub-section is based on unstable conditions. In contrast to the preceding sub-section, we only replace constant arrival rate with a functional rate, but the service rate remains stable. All the other rates remain the same as those in Section 3.5.1. Here, the arrival rates are modified to:

$$r_{arriveB} = 400 \times (\sin(time) + 1)$$

$$r_{arriveX} = 400 \times (\sin(time) + 1)$$

$$r_{arriveBX} = 400 \times (\sin(time) + 1)$$

On the basis of the changed arrival rates, a set of new figures for ODE analysis is obtained:

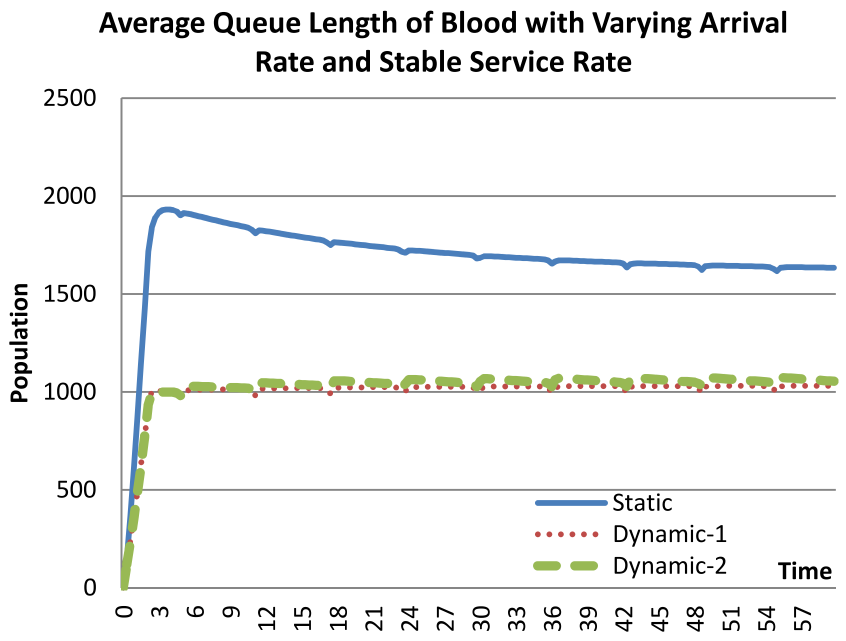


Figure 3.5.4.a Average queue length of Blood with varying arrival rate and stable service Rate

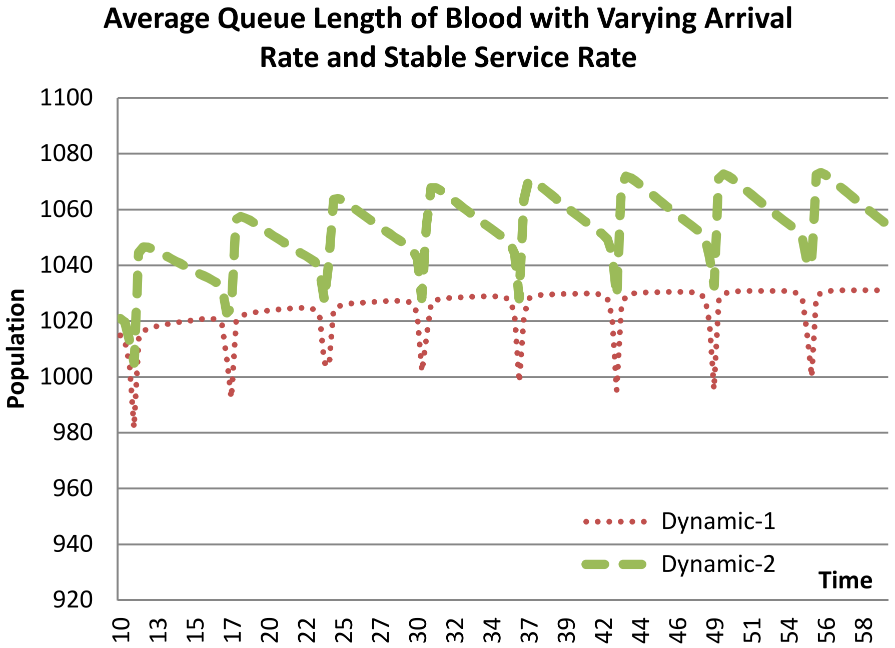


Figure 3.5.4.b Average queue length of blood with varying arrival rate and stable service rate (average queue length for Dynamic 1 and Dynamic 2, time: 10~60)

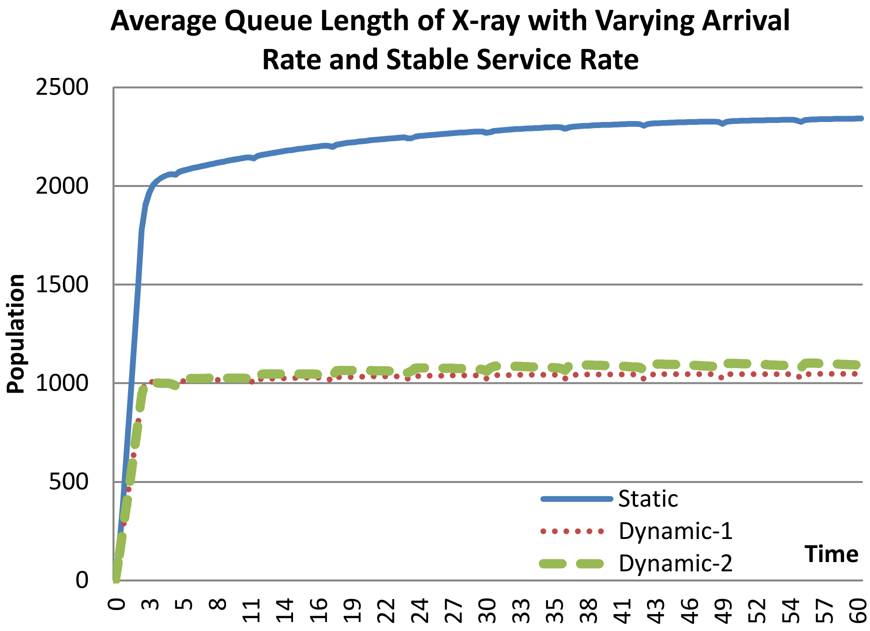


Figure 3.5.5.a Average queue length of X-ray with varying arrival rate and stable service rate

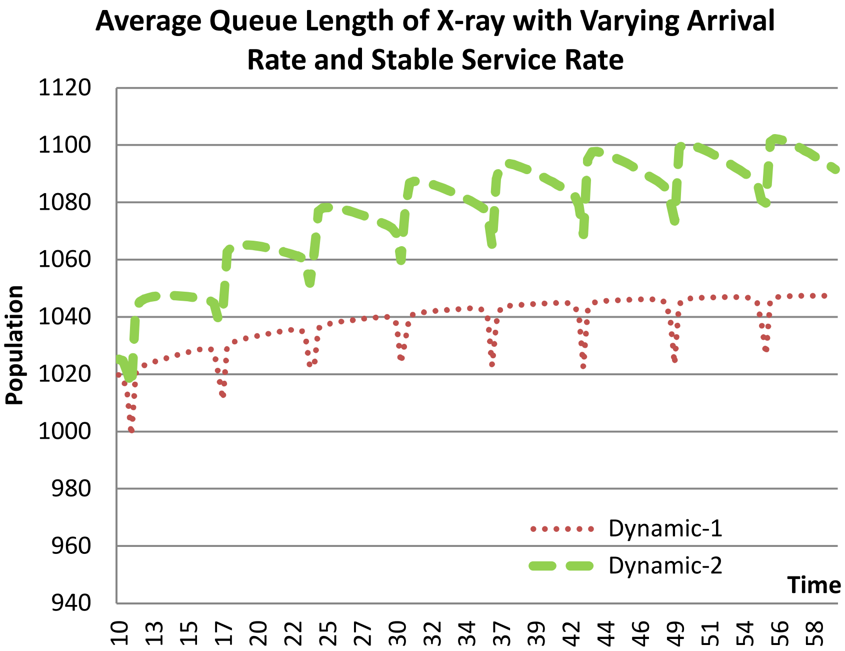


Figure 3.5.5.b Average queue length of X-ray with varying arrival rate and stable service rate (average queue length for Dynamic 1 and Dynamic 2, time: 10~60)

As can be seen from Figure 3.5.4.a and 3.5.5.a, dynamic scheduling policy has far fewer patients in the queue waiting at both blood and x-ray in contrast to the static. In Figure 3.5.4.a, the static policy has about 1600 patients in queue that is similar with the value in Figure 3.5.1. Dynamic policy 1 has the average queue length of about 1030, while dynamic policy 2 is a bit more around 1050. In Figure 3.5.5.a, the queue length for the static policy is about 2340, which is the same as that in Figure 3.5.2. However, the queue length for dynamic policy 1 and dynamic policy 2 are roughly 1050 and 1090 respectively. Figure 3.5.4.b and Figure 3.5.5.b show the difference between the two dynamic policies providing details about average queue length of the two dynamic policies. From these figures, it is clear that dynamic policy 1 has a smaller queuing population. Moreover, the red line for policy 1 appears more stable than the green line.

Consequently, on the condition that arrival rate varies and service rate is stable, static policy has a much longer queue length than dynamic policies; dynamic policy 2 has a few more patients in the queue than does dynamic policy 1. Thus, when arrival is unstable, dynamic policy 1 is more suitable than dynamic policy 2, as in dynamic policy 1, decision rate alters with the length of waiting queue, which is directly influenced by the varying arrivals. In other words, dynamic policy based on queue length has a better adaptability when arrival rate is not stable.

Moreover, compared with lines in Figure 3.5.1 and 3.5.2, all lines in Figure 3.5.4 and Figure 3.5.5 have very small and periodical fluctuations. Such small and periodical waves are caused by the varying arrival rates.

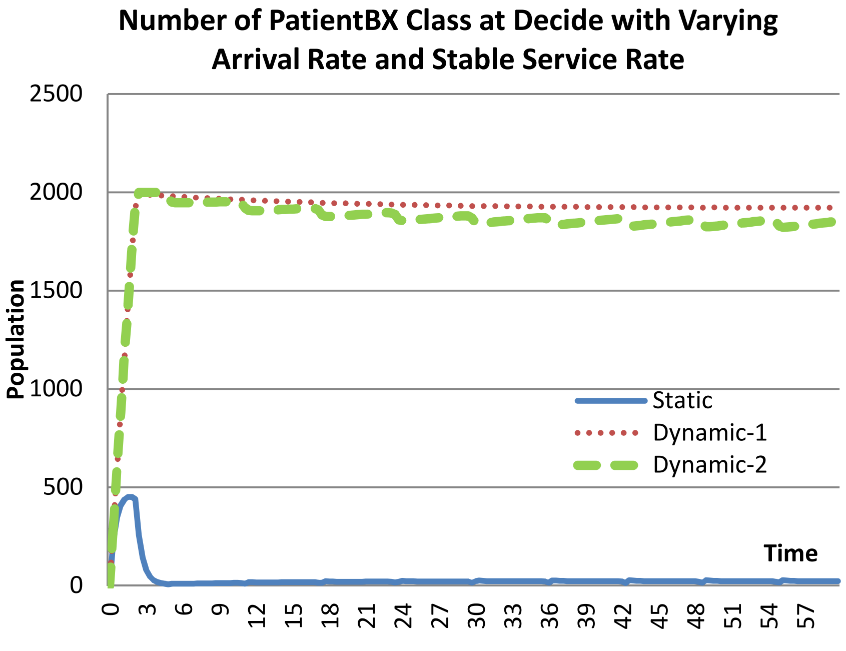


Figure 3.5.6 Population of PatientBX waiting at Decide component with varying arrival and stable service rate

Figure 3.5.6 depicts the number of patients who need both blood and x-ray tests and are prevented from entering hospital by the scheduling policies. From the diagram, dynamic policy 1 prevents about 1920 patients on average, while dynamic policy 2 prevents about 1850 on average. However, the static policy just blocks about 20 patients, most patients are let in without an efficient block. Furthermore, the line for dynamic policy 2 has obvious fluctuations but the line for dynamic policy 1 is quite straight. This testifies that dynamic policy 1 has the better ability in reducing the wave caused by the functional arrival rate. All this evidence demonstrates dynamic policy 1 is the optimal policy in the condition of varying arrivals.

3.5.3 Analysis with stable arrival rate and varying service rate

In order to know how scheduling policies adapt to unstable service conditions, in this sub-section, analysis is conducted with a varying service rate, and arrival rates are kept constant. Except for service rates, all other parameters remain the same as those in Section 3.5.1. The varying service rate is the constant service rate multiplied by a sine function or a cosine function.

$$r_{blood} = 120 \times (2 \times \sin(0.5 \times time) + 2)$$
$$r_{xray} = 80 \times (2 \times \cos(0.5 \times time) + 2)$$

With the above varying service rates, some amazing results emerged from fluid ODE analysis.

Figure 3.5.7.a describes the change of average queue length at *Blood* for three scheduling policies in the condition of varying service rates. An interesting observation is the dramatic fluctuation of static policy between 1400 and 1800. Such waves illustrate that the static policy cannot stably schedule patients when service has large variation. On the contrary, dynamic policies perform much better ability in scheduling. According to Figure 3.5.7.b, dynamic policy 2 has very stable average queue length which is between 1000 and 1005. However, dynamic policy 1 is a bit poorer, as its average queue length fluctuates between 1005 and 1025. Although the fluctuation of dynamic policy 1 is more obvious than dynamic policy 2, it is still much superior to the static policy.

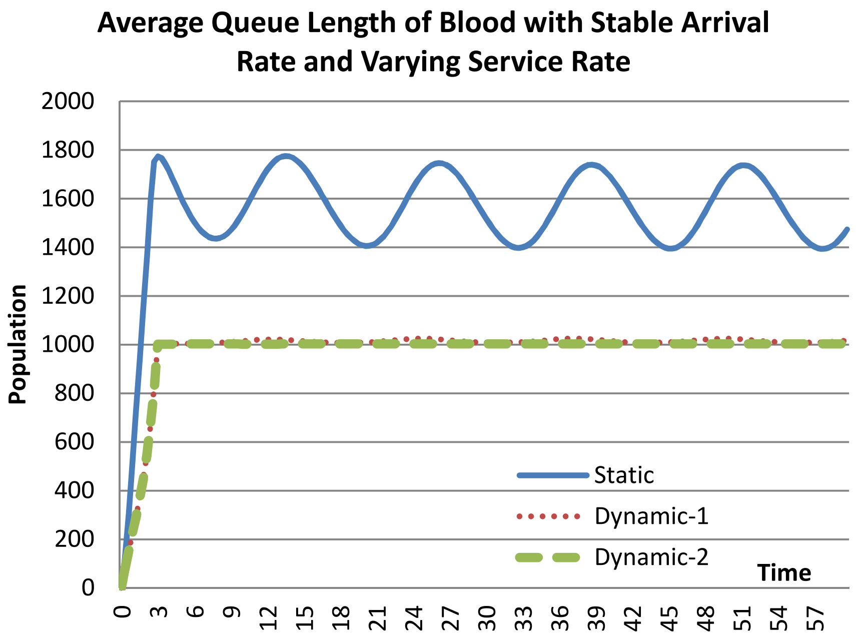


Figure 3.5.7.a Average queue length of Blood with stable arrival rate and varying service rate

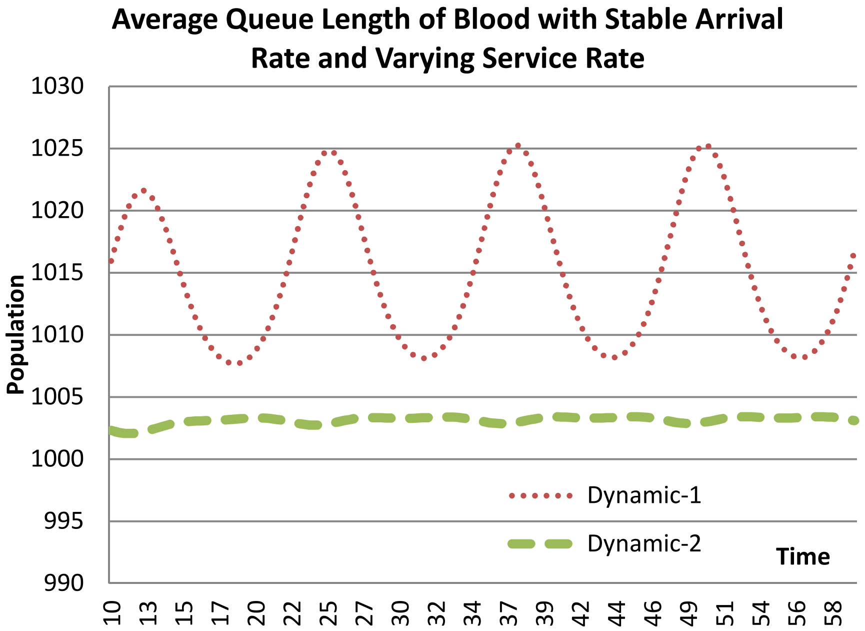


Figure 3.5.7.b Average queue length of blood with stable arrival rate and varying service rate (average queue length for Dynamic 1 and Dynamic 2, time: 10~60)

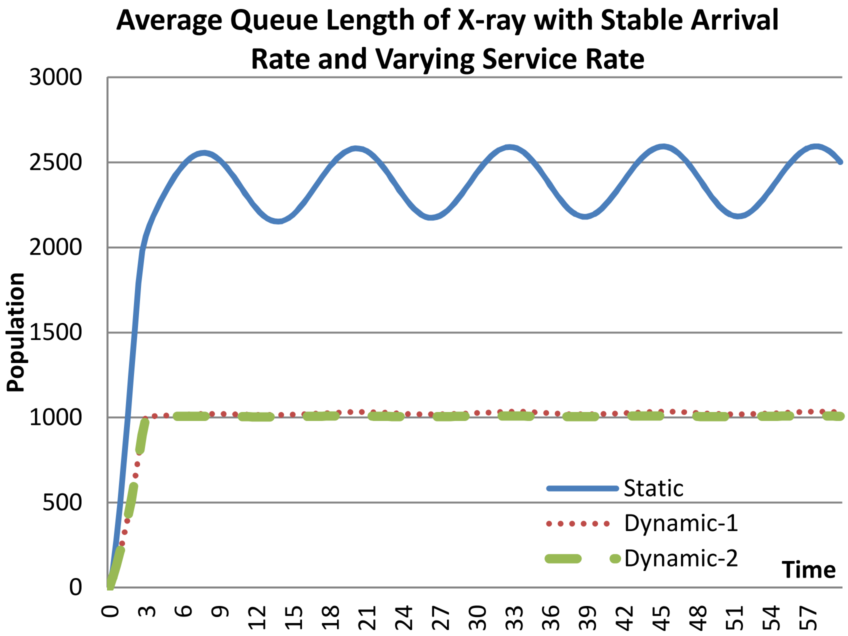


Figure 3.5.8.a Average queue length of X-ray with stable arrival rate and varying service rate

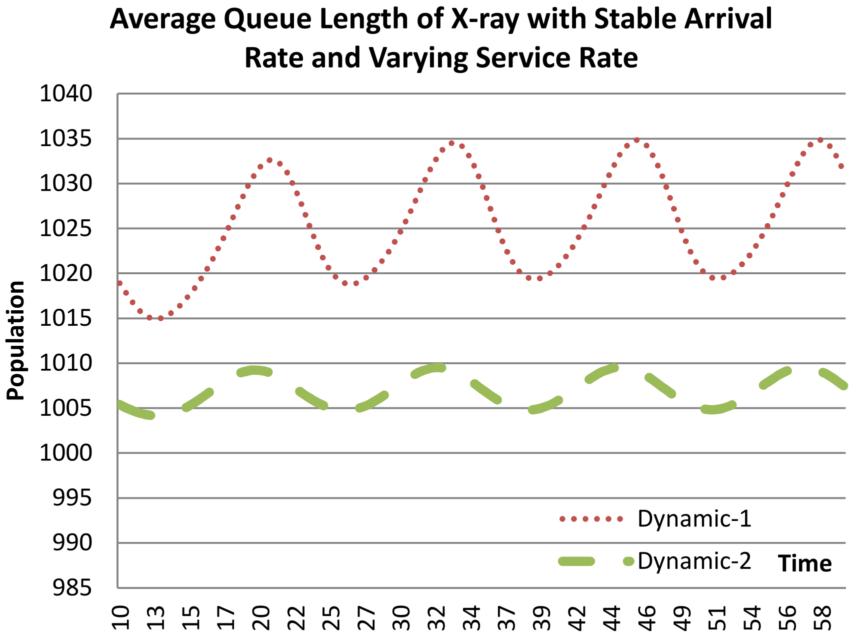


Figure 3.5.8.b Average queue length of X-ray with stable arrival rate and varying service rate (average queue length for Dynamic 1 and Dynamic 2, time: 10~60)

For the *X-ray*, the situation of average queue length is similar with that at *Blood*. As displayed in Figure 3.5.8.a, static policy creates distinct waves between 2100 and 2600. However, dynamic policies appear quite smooth and steady. From Figure 3.5.8.b, we can see that dynamic policy 2 has a weaker undulation between 1005 and 1010, and a smaller population in average queue length. Dynamic policy 1 is inferior to policy 2, as its average queue length undulates on a higher level between 1015 and 1035. However, the average queue length of both dynamic policies is much shorter than the queue length of the static policy.

Consequently, in the condition of varying service rate, dynamic policies show their smooth and steady ability in the scheduling process, especially dynamic policy 2, which is the optimal policy under the current circumstances. Conversely, the static policy completely loses its stability compared with its previous performance

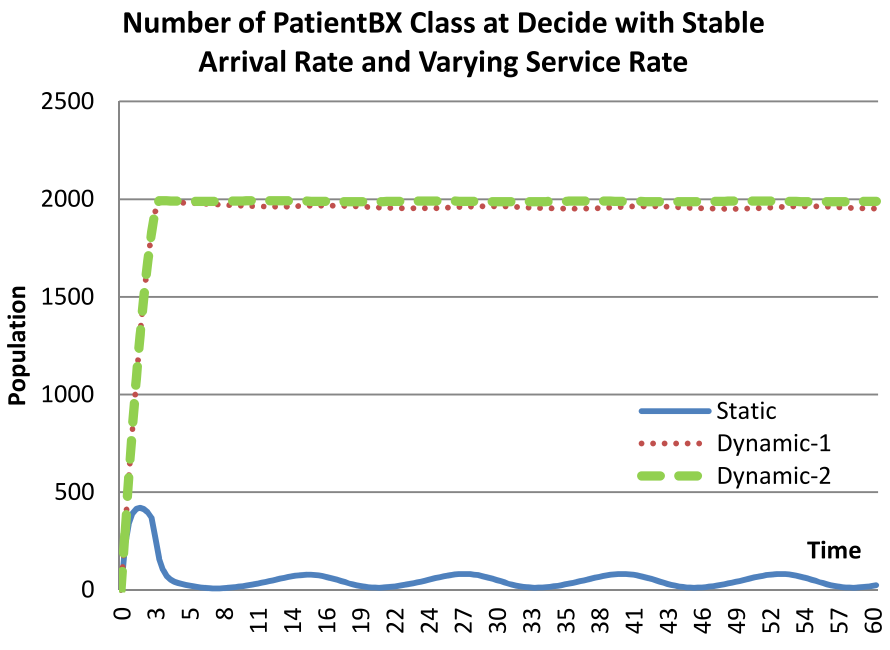


Figure 3.5.9 Population of PatientBX waiting at Decide component with stable arrival and varying service rate

As per Figure 3.5.9, about 1950 patients who need both blood and x-ray tests are prevented from entering hospital by dynamic policy 1, and nearly 1990 patients are blocked by dynamic policy 2. However, the number of patients prevented by the static policy is no more than 100; what is worse, the value of this number always changes with time. All these figures prove the ability of three policies in the scheduling process, and also confirm that the previous conclusion is unquestionable. In the condition of varying service, dynamic policies have an excellent performance in scheduling, and dynamic policy 2 is the optimal policy.

3.5.4 Analysis with varying arrival rate and service rate

In previous sections, we have analyzed the performance of three scheduling policies with constant rates, functional arrival rate or functional service rate respectively. Factually, in the real world hospital environment, both arrivals and service are unstable. The number of patients may vary during day and night, and also the service in the hospital changes at different times. This section aims to model a more realistic hospital environment by adding function rates for both arrival rate and service rate, and then measure the performance of the three scheduling policies with ODE's fluid flow analysis. In the analysis, parameters in Section 3.5.1 remain unchanged, except arrival rate and service rate, which are modified as:

$$r_{arriveB} = 400 \times (\sin(time) + 1)$$

$$r_{arriveX} = 400 \times (\sin(time) + 1)$$

$$r_{arriveBX} = 400 \times (\sin(time) + 1)$$

$$r_{blood} = 120 \times (2 \times \sin(0.5 \times time) + 2)$$

$$r_{xray} = 80 \times (2 \times \cos(0.5 \times time) + 2)$$

Fluid flow analysis proceeded with the above modified parameters by solving a set of ODEs, and new figures were generated as follows:

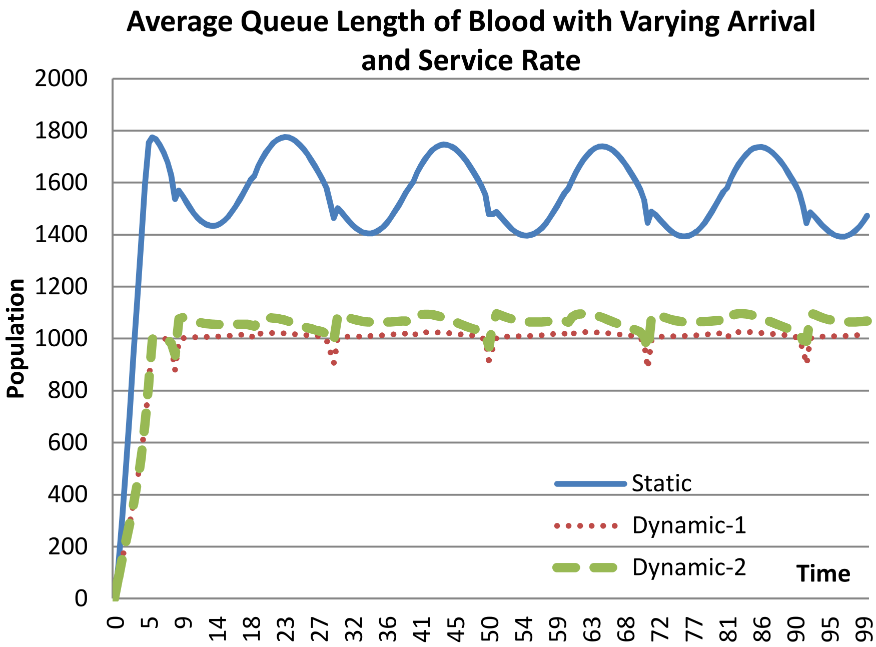


Figure 3.5.10 Average queue length of blood with varying arrival and service rate

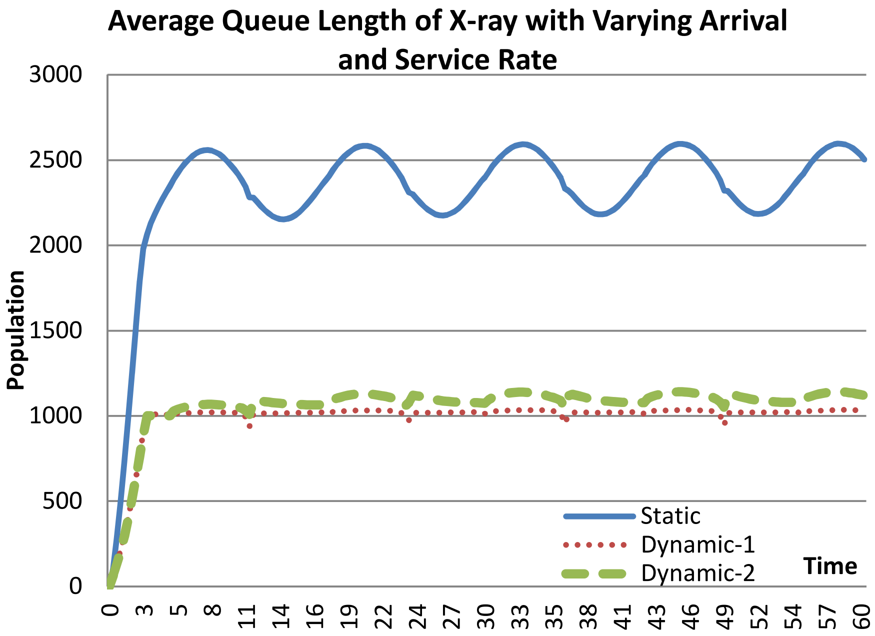


Figure 3.5.11 Average queuel of X-ray with varying arrival and service rate

Figure 3.5.10 and Figure 3.5.11 respectively depict the average queue length of three scheduling policies at component *Blood* and *X-ray*. According to the figures, it is obvious that dynamic policies have much smaller population in the queue compared with the static, and lines standing for dynamic policies appear more stable than for the static policy.

From Figure 3.5.10, the red line for dynamic policy 1 keeps steady at around 1000, and also with some downward sharp waves. The green line for dynamic policy 2 lies at a higher level than the red line, in which it is nearly 1100 on average. Furthermore, the red line for dynamic policy 1 appears more smooth and stable than the green line for dynamic policy 2. However, the static policy is still the worst of all having an average queue length fluctuating between 1400 and 1800. In this figure, all lines have small sharp waves caused by the varying arrival rates, as appeared in Section 3.5.2.

In Figure 3.5.11, the trend of each line is similar with that in Figure 3.5.10. The average queue length of static policy increases to the interval between 2100 and 2600, because of the rate of x-ray is less than the rate of the blood test. Dynamic policy 1 has queue length around 1000, and policy 2 slightly waves around 1100.

To observe the ability of three scheduling policies, Figure 3.5.12 represents how many patients who need both blood and x-ray tests are successfully prevented from entering hospital so as to avoid a long time queuing in the hospital.

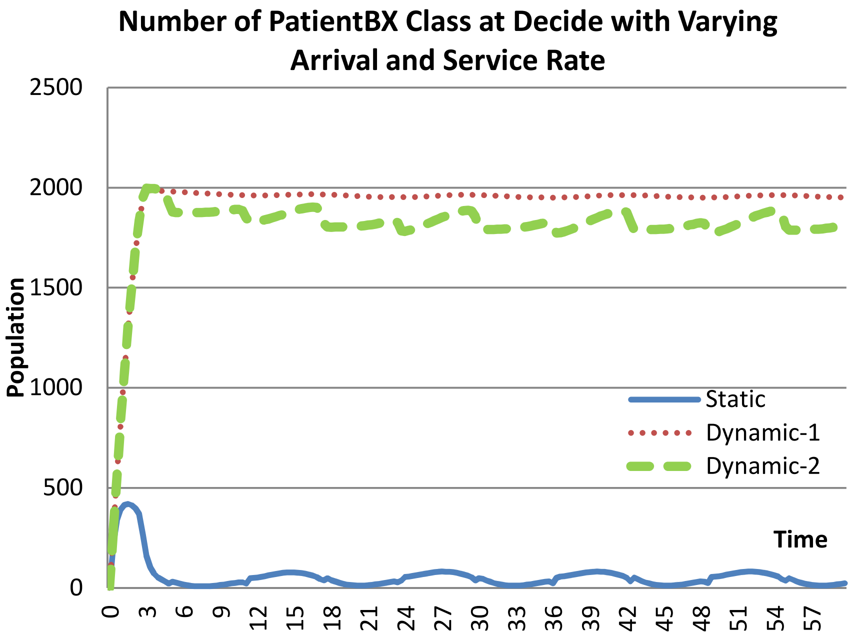


Figure 3.5.12 Population of PatientBX waiting at Decide component with varying arrival and service rate

From the Figure 3.5.12, the number of patients blocked by dynamic policy 1 is about 1950 on average. For dynamic policy 2, its line periodically undulates around 1800. This means dynamic policy 1 is more efficient and stable than the dynamic policy 2 in the scheduling process. However, the static policy prevents no more than 100 patients from entering hospital. Overall, dynamic policy 1 is the optimal policy when arrival rate and service rate are both function rates.

In conclusion, from the above figures, it can be declared that the dynamic policies have a much more efficient and stable ability in the scheduling process than the static policy. Of the two dynamic policies, dynamic policy 1 has stronger adaptability and better performance under such conditions in this section in contrast to the dynamic policy 2. However, when both arrival rate and service rate are dynamically changing, it is hard to say which dynamic policy is better, and the optimal policy depends on the extent of rate variation. If the variation of arrival rate is greater than that of the service rate, the queue length based dynamic policy 1 will be better than the dynamic policy 2. Conversely, if the service rate has a greater variation than the arrival rate, the response time based dynamic policy 2 will yield a better performance than the dynamic policy 1.

3.5.5 Summary

This section explores the performance of three scheduling policies under four different kinds of conditions. According to the measurements and analysis, the performance of the three policies can be concluded:

- Both dynamic scheduling policies give a much better performance than the static policy at all service components.
- Dynamic policy 1, which is based on queue length, is more suitable for the system with varying arrival rate.
- Dynamic policy 2, which is based on response time, is the optimal policy under varying service rate conditions.

- When both arrival rate and service rate are varying, the best dynamic policy depends on which factor, varying arrival rate or varying service rate, plays a greater role in the system.

With the above conclusions, it is more efficient to select a dynamic scheduling policy in terms of the real situation.

3.6 Conclusion

This chapter aimed to explore the performance of dynamic scheduling policies by comparing these with static scheduling policies. The main method used for measurement was fluid flow approximation, by solving a set of ordinary differential equations generated from original PEPA models. The ODE analysis was carried out with an equivalent CMDL model converted from PEPA model.

First of all, two PEPA models were created based on the two types of scheduling schemes as introduced in Section 3.2. After comparing the two schemes, the dynamic scheme was adopted for the following analysis, because it supported function rates used to model the dynamic scheduling process.

Thereafter, a kind of dynamic scheduling policy based on the varying queue length was analyzed by replacing the constant decision rate with a function rate. Through comparing the average queue length between the dynamic policy and static policy, an initial conclusion was obtained. This conclusion is that the dynamic policy gives a better performance than the static policy especially when the system conditions become unstable.

According to the initial conclusion, further analysis were proceeded to explore the performance of dynamic scheduling policy under more complex conditions. To promote further exploration, the previous PEPA model was modified to facilitate the following measurements. Furthermore, a new dynamic scheduling policy was introduced in this section, and such new policy was based on the varying transient response time. Measurements were completed on four steps, and each step had different conditions. The analysis was conducted by altering conditions (arrival rate and service rate) step by step from stable to variable.

The final outcomes from the analysis demonstrate that dynamic scheduling policies give an excellent performance in the scheduling process under all kinds of circumstances. However, the static policy becomes the loser, as shown by its performance measurements. Regarding the two different types of dynamic scheduling policies, each one has its good points. Thus, we can determine which dynamic policy best suits the real conditions.

Overall, this chapter explored the performance of various scheduling policies with the fluid flow approach, and the findings bring us much excitement and new clues for future work.

In this chapter, formal method PEPA is applied to create the scheduling model and fluid flow approximation is used to complete the model analysis. However, according to the features of the fluid flow analysis, the analysis based on the fluid flow approximation can carry out highly accurate analysis when the system has a large number of job events. Thus, when just a few active events exist in the system, does the fluid flow approximation still generate an accurate analysis?

Hence, in Chapter 4, another modelling approach, discrete event simulation, will be firstly used to validate the analysis of the PEPA based scheduling model, and secondly used to investigate the performance of scheduling policies in the system with a small number of events.

Chapter 4 Dynamic scheduling model with discrete event simulation

4.1 Introduction

This chapter aims to verify the performance of the scheduling policies studied in Chapter 3 by the discrete event simulation. The work of the chapter has been published in the 1st Int'l Workshop on Healthcare Systems Engineering in the year 2011 [72].

4.1.1 Modelling techniques

This chapter will continue investigating the performance of scheduling policies, but in this chapter the discrete event simulation is applied for the performance evaluation process. In the previous chapter, models of three scheduling policies were created with the PEPA process algebra, and the analysis was conducted by fluid flow approximation via solving a set of ODEs. Fluid flow approximation is a novel performance analysis technique for large scale systems modelled in the stochastic process algebra PEPA. The fluid flow technique works with a set of ordinary differential equations (ODEs) and is well suited to systems with large number of replicated components. When the number of components is large enough, steps between events are relatively small, and then we can approximate model behaviours by considering the movement between states to be continued, rather than occurring in discontinuous jumps [17]. Hence, in previous chapter, our models analysis is carried out with a large number of replicated customer components in system.

However, in reality, there are generally fewer patients in the system. When just a small number of patient components exist in the system, the event flow in the system can no longer be thought of as continuous. This means, in this situation, the previous well suited fluid flow technique may not be accurate in terms of its basic theory. Thus, how do we know the performance of scheduling policies in the system with a few customer components, and whether are these policies still have the same performance as them in the large scale system? Therefore, in this chapter, discrete event simulation will be used to simulate three scheduling policies with based on the same scheduling scenario, and then measurements will be obtained to evaluate their performance. Here, discrete event simulation is implemented through Java programming and all measurements are generated based on the original simulation output without approximation; thereafter, the simulation results will be validated by comparing with the results of formal models (PEPA).

4.1.2 Scenario description

The model scenario used in this chapter is equivalent to the previous chanter. The same patient flow model is employed to represent the scheduling policies. The three different scheduling policies modelled are the same as those in the Chapter 3, which consist of the static scheduling policy, the queue length based dynamic scheduling policy 1 and the response time based dynamic scheduling policy 2.

As assumed in Section 3.4, here the assumptions of the scenario remain the same for the simulation:

- Two tests: blood test and x-ray test.
- Three classes of patients: patients only need blood test, namely class B; patients only need X-ray test, namely class X; patients need both, namely class BX.

All new incoming patients must register before entering the hospital either register online or go to the reception. Patients who only need a single test (blood or X-ray) will be scheduled to the queue for the related service, but those who need both tests must be determined which test should come first. Such a decision is made by the smart system using three different scheduling policies. The focus in this chapter is to observe related properties of the queue at each service component (blood and Xray), such as the average queue length and the average response time. All patients will be discharged from hospital after completing their tests. Discharged patients will re-enter hospital after a waiting period outside the hospital, which guarantees the fixed total number of potential patients in the hospital. Figure 4.1 displays the scheme of patient flow model.

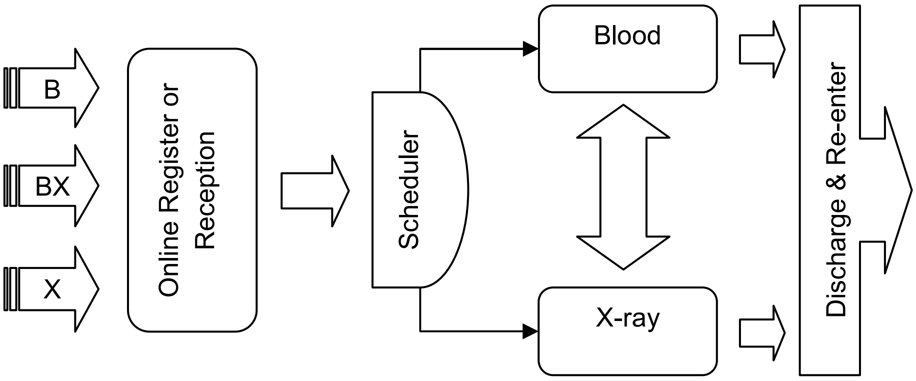


Figure 4.1 Hospital patient flow model

4.1.3 Scheduling policies

Based on the above patient flow model, three scheduling policies are embedded in the model, and these three policies have the similar functionality of scheduling patients but they provide difference scheduling process. Thus, these three scheduling policies are detailed as follows:

- Static scheduling policy: The static scheduling policy is a method which could schedule incoming customers to the right destination with a stable rate in scheduling process. Such static is completely the same as that in the previous chapter seeing Section 3.4.2.1.
- Dynamic scheduling policy 1: The dynamic scheduling policy 1 is a type of dynamic scheduling policy using the transient queue length to determine the scheduling rate so as to manage the scheduling process. Dynamic policy 1 has the same features as that in Section 3.4.2.2.
- Dynamic scheduling policy 2: The dynamic scheduling policy 2 controls the scheduling process based on the change of transient

response time. In this chapter, the dynamic policy 2 has been altered to be a little different from the one employed in Chapter 3. Here the policy uses the varying transient response time rather than the ratio of average response time and transient response time, namely $P = \frac{Average\ Response\ Time}{Transient\ Response\ Time}$. This is because the average response time in the PEPA model can be approximated by calculating mean values for the terminal model, noted in Section 3.4.2.3; However, according to the theory of terminal model, this approximation has high accuracy if and only if the number of clients is quite large. However, this chapter aims to measure the performance of policies with a small scale client population. Therefore, the average response time cannot be approximated in terms of the current condition. Furthermore, regarding the features of discrete event simulation, the average response time cannot be obtained until the simulation runs out. Thus, the previous scheduling scheme, using the ratio of average response time and transient response time, has to be abandoned. Here, it is necessary to directly use the transient response time to dynamically alter the functional decision rate in the scheduling process. The expression of decision rate for the current dynamic policy 2 can be represented as:

$$
Decision\ Rate\ of\ Dynamic\ Policy\ 2 = \frac{Constant\ Value}{Transient\ Response\ Time}
$$

4.1.4 Simulation tool – SimJava

As defined in SimJava tutorial, SimJava is a process oriented discrete event simulation tool. SimJava is an API extending Java through building blocks in order to create and implement simulations. The SimJava was originally based on HASE++, a C++ simulation library. HASE is a Hierarchical computer Architecture design and Simulation Environment developed at the University of Edingburgh.

HASE can support the visualisation of activities occurring inside the computer through simulation when they execute programs.

SimJave is quite similar with the programming of Java packages in the simulation. In SimJava, each real-life or hypothetical system can be considered as a set of defined entities and a series of interactions between the defined entities in the system. The entities communicate with others in the system through creating a series of discrete events which are passed from one entity to another. The simulation time in SimJava elapses with the transition of these events.

SimJava v2.0 is the latest version which provides considerable statistical and reporting support. SimJava 2.0 is a powerful simulation tool because of its provided functions for statistical measurements and analysis, such as, the embedded statistical measurements including default measures and customer measures. These measures are able to collect figures for throughput, utilization, queue length, waiting time, service time and resident time.

Moreover, SimJava 2.0 provides the function for modellers to generate a report and a graph viewer for each run in order to facilitate the analysis.

4.1.5 Summary

This chapter will introduce new models for three scheduling policies with Java based discrete event simulation. According to the simulation measurement results, we will analyze and compare the performance of three policies in order to conclude their performance under discrete customer situation. In the following section, we will orderly introduce the design of simulation model, implementation and validation of model and measurement analysis.

4.2 Simulation model design

4.2.1 Model structure

According to the research aim of this chapter, at first we will rebuild the patient flow model in terms of the principles of discrete event simulation. Before going into implementation details of discrete event simulation, the overall structure design must be accomplished. Regarding the scenario description mentioned in previous section, we can obtain a model scheme as displayed in Figure 4.2.

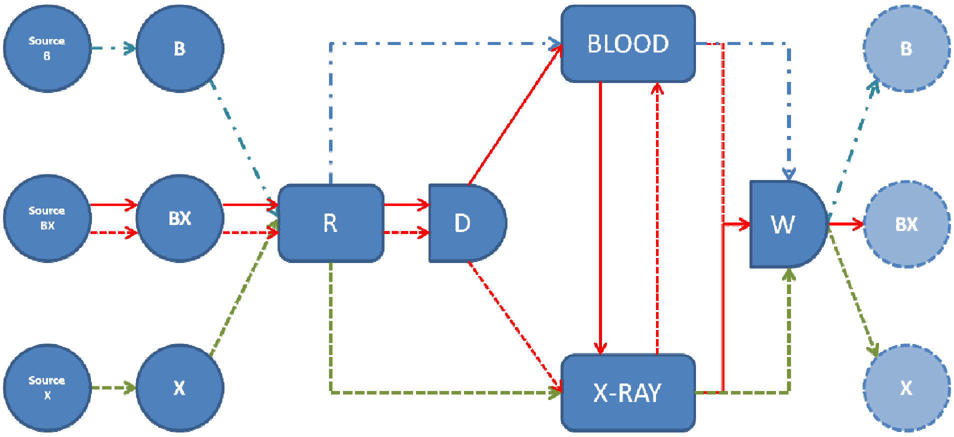


Figure 4. 2 Simulation model design scheme

In Figure 4.2, all circle components represent patients in different states:

- Components Source B, Source X and Source BX respectively generate a certain number of patients in each class: blood only, x-ray only and both. These components guarantee that the total population of patients in system is fixed and constant. It is worth underlining that source components are just used to generate a fixed number of patient instances. Once these instances are created, source components do not work again until the run out of the simulation.
- Components B, X and BX model the processes that each patient class generates new arrivals with a specified arrival rate.
- Component R is register step, which models each arrival is registered in the system.

- Component D is a decider or scheduler, which determines how fast the system sends a patient to queue for service in hospital. Those three scheduling policies will be modelled and embedded in this component. Scheduler component is to control and adjust patient flow with its embedded scheduling mechanism. So far, all introduced components are out-hospital components.
- Blood and X-RAY components represent blood test and x-ray scan in hospital. Patients will gain their service at these points. Either Blood or X-RAY component serves patient one by one. Patient will discharge from hospital once service is done.
- Component W models the process patients discharge from and stay outside the hospital. After a specified period, these patients will re-enter the model loop as they starts at component B, X, BX.
- Dotted circle components are just duplications of component B, X and BX. They are absolutely the same components. Here, they are labelled to represent patient return clearly.
- The blue line in the figure represents the action flow of Patient B who just needs blood test. Similarly, the green line represents the action flow of Patient X who just need X-ray scan. For the Patient BX who needs both tests, the twin red lines represents their action flow; the twin red lines will split for different tests after component D, half to the blood test first, and half to the X-ray scan first, and then each branch will go to the second test after completing the first.

4.2.2 Simulation model vs. PEPA model

A simulation model is created on the basis of the same scenario as the PEPA model in Chapter 3. In other words, the structure of the simulation model should be equivalent to the PEPA model. Here, the two models are compared in detail.

The original PEPA model is specified as below:

$$
\begin{aligned}
PatientB &\overset{def}{=} (arriveB,\ r_{arrive\_b}).PatientB_1\\
PatientB_1 &\overset{def}{=} (register,\ r_{register}).PatientB_2\\
PatientB_2 &\overset{def}{=} (blood,\ r_{blood}).PatientB_3\\
PatientB_3 &\overset{def}{=} (wait,\ r_{wait}).PatientB
\end{aligned}
$$

$$
\begin{aligned}
PatientX &\overset{def}{=} (arriveX,\ r_{arrive\_x}).PatientX_1\\
PatientX_1 &\overset{def}{=} (register,\ r_{register}).PatientX_2\\
PatientX_2 &\overset{def}{=} (xray,\ r_{xray}).PatientX_3\\
PatientX_3 &\overset{def}{=} (wait,\ r_{wait}).PatientX\\
PatientBX &\overset{def}{=} (arriveBX,\ r_{arrive\_bx}).PatientBX_1\\
PatientBX_1 &\overset{def}{=} (register,\ r_{register}).PatientBX_2\\
PatientBX_2 &\overset{def}{=} (decideB,\ \top).PatientBX_3\\
&\quad + (decideX,\ \top).PatientBX_4\\
PatientBX_3 &\overset{def}{=} (blood,\ r_{blood}).PatientBX_5\\
PatientBX_5 &\overset{def}{=} (xray,\ r_{xray}).PatientBX_6\\
PatientBX_6 &\overset{def}{=} (wait,\ r_{wait}).PatientBX\\
PatientBX_4 &\overset{def}{=} (xray,\ r_{xray}).PatientBX_7\\
PatientBX_7 &\overset{def}{=} (blood,\ r_{blood}).PatientBX_8\\
PatientBX_8 &\overset{def}{=} (wait,\ r_{wait}).PatientBX\\
Blood &\overset{def}{=} (blood,\ r_{blood}).Blood\\
Xray &\overset{def}{=} (xray,\ r_{xray}).Xray\\
Reception &\overset{def}{=} (register,\ r_{register}).Reception
\end{aligned}
$$

$$
\begin{aligned}
DecideB &\overset{def}{=} (decideB,\ r_{db}).DecideB\\
DecideX &\overset{def}{=} (decideX,\ r_{dx}).DecideX
\end{aligned}
$$

$$
Wait \overset{def}{=} (wait,\ r_{wait}).Wait
$$

$$
\begin{gathered}
PatientB[N_1] || PatientX[N_2] || PatientBX[N_3]\\
\bowtie_L\\
Blood || Xray || Reception || DecideB || DecideX || Wait\\
L = \{register,\ blood,\ xray,\ decideB,\ decideX,\ wait\}
\end{gathered}
$$

The simulation model defines the same set of components as those in the above PEPA model except for the extra three Source components. Thereafter, the event flow in each model is compared. In Figure 4.2, the blue chain-dotted line represents a closed flow of Patient B (blood test only). They are created by the source, and then start their trip from component B. From B, these patients leave for the next point R (Registration) with a specified arrival rate; after the registration, they are directed to the component BLOOD in which they queue for the service. At the service component BLOOD, patients have blood test one after another at a service rate. Once patients complete their tests, they discharge from the service immediately and move to W (wait) component that is out of hospital. After a period at W component, patients return to the component B and re-start their trip. Such a loop does not terminate until the run out of simulation time. Compared with the process in the simulation model, the PEPA model demonstrates the same process by noting each state of patient B and the related activities in the first block of the PEPA model.

Likewise, the green dotted line, in Figure 4.2, displays a similar process but for Patient X (X-ray test only). In contrast to the simulation model, the second block in the PEPA model describes the equivalent process.

The key point in the simulation model is the flow of Patient BX (patient that need both blood and x-ray) which is represented with the red line in Figure 4.2. At the beginning, they have the same process as Patient B or Patient X: all patients are generated by the Source and then start from component BX and leave for component R. From there on, this class of patients will go to a scheduling component which is D (decision made for scheduling) in the diagram. The D component is to decide how fast these patients can be scheduled to the next component and determine which component the patients should be sent to in the next step. In the simulation model, the patient flow splits into two sub-groups in component D depending on the decision rate. Each sub-group flows to the following server component with a specified decision rate; meanwhile, the population of each sub-group sent to the next component also depends on the ratio of each type of decision. In fact, this decision rate simulates a type of scheduling policy in order to control the patient flow towards the component BLOOD or X-RAY. In the simulation, the decision rates are defined as rd_{blood} and rd_{xray} for the blood and the X-ray respectively; furthermore, the proportion of Patients BX going to the blood first is $\frac{rd_{blood}}{rd_{blood}+ rd_{xray}}$ and the proportion to the x-ray first is$\frac{rd_{xray}}{rd_{blood}+ rd_{xray}}$. Each sub-group of Patient BX will complete one test first and then go to the other one in the simulation model. When the patients have both tests done, they leave the hospital and return to the start component BX after a short stop at W component. This design is to obtain an equivalent process compared with that in the PEPA model.

Here, it is assumed that all components except BLOOD and X-RAY can be considered as out-hospital components. The main function of the D component is to prevent many patients from entering the hospital so as to avoid long queuing in the hospital, and also able to scheduling patients in terms of the proportion calculated on the basis of the decision rates. Thus, the decision rate represents the mechanism of flow control by implementing relative scheduling policies in it.

In addition, some difference occurs between the simulation model and the PEPA model in defining the expression of the decision rate. As noted in the third block of the PEPA model, the rate of the cooperating actions *decideB* and *decideX* is defined as passive, using symbol T. According to Hillston's definition of the passive rate, the passive decision rate can be explained as: the rate of decision is infinite but its value depends on the population of patients at this state and the number of D components. In others words, we can consider this passive rate as the situation that each patient has a specialized D component deciding for himself with the rate defined for D component because of the cooperation in PEPA. Conversely, if this decision rate is specified as an active rate like all other rates in the PEPA model, the meaning of rate is different, which means all patients share this single decision rate rather than that each patient has a separated decision rate.

The aim of using a passive decision rate in PEPA model is to model a fast scheduling process as the decision making process factually is very fast in a real world system. However, in the discrete event simulation, such passive rate mechanism cannot be simulated. Instead, it has to simulate the scheduling and deciding process in terms of the PEPA model with active decision rate in cooperation. This is an important difference between the current simulation and the original PEPA model. All other flows and activities in simulation are completely identical to those in the PEPA model.

4.3 Simulation model implementation

This section will briefly introduce the simulation design details in terms of the simulation tool SimJava 2.0. The simulation program example will be used to demonstrate the implementation process.

4.3.1 SimJava based simulation design

In the previous section, the details of the simulation design have been introduced. Thus, this section will briefly explain the implementation of the simulation model.

As mentioned before, discrete event simulation techniques are adopted to achieve our research requirements. First of all, a suitable simulation tool must be chosen to build the simulate model. SimJava 2.0 is used in this simulation work. The SimJava is a discrete event, process oriented simulation package. It is an API that augments Java with building blocks for defining and running simulations. The approach to simulating systems adopted by SimJava is similar to other process based simulation packages. Each system is considered to be a set of interacting processes or entities as they are referred to in SimJava. These entities communicate with each other by passing events. The simulation time progresses on the basis of these events.

In this simulation, each type of component displayed in Figure 4.2 is simulated as a specified entity. For example, Patient B, X and BX can be defined as the same type of entity, such as Patient entity. Thus, the simulation can create different instances of *Patient* entity which represent B, X and BX respectively. BLOOD and X-RAY components can be defined with the same type of entity *Test*, so the BLOOD and X-RAY can be created as instances of *Test* entity. Similarly, such entity defining approach can be used for other components, e.g. Source components, D components and W component. All the instances of these entities are linked by the ports designed in the entity class.

Another key point in the simulation is the event. Entities communicate with each other by passing events. In the simulation, the Source entity is used to generate a specified number of events. Each event is initialized with a tag which represents the type of event when they are created in the Source. For example, events generated for the Patient B can be tagged with number 1; events for Patient X can be tagged with 2; and finally 3 for Patient BX. All events are passed from one component to another in the system representing the patient flow in the hospital. This simulation defines six types of entities: Source, Patient, Register, Decider, Test and Wait.

In terms of the simulation model design scheme, the Source entity has three instances to generate different kinds of events representing three groups of patients respectively; the Patient entity also has three instances linking with the corresponding Source instances, which are used to forward the incoming events to the next component with an arrival rate. The Register entity has only one instance which models the registration process in the hospital.

Furthermore, Register also has ability to direct patient events. For the events from entity B or X, they are sent to the BLOOD or X-RAY entity directly by the Register, but for the events from BX, they are separated in the Register. Half of the BX events are sent in the BLOOD direction and the other half are sent to X-RAY. The BX events must go through the Decider entity before arriving at the BLOOD or XRAY. There is a single Decider instance in the simulation model which is to achieve the scheduling for the Patients BX. Such Decider restrains the number of sending events to the next component in order to reduce the queue in BLOOD or X-RAY components, and also determines the proportion of BX patients going to different test in the next step. To achieve this, the simulation uses those three scheduling policies in the Decider entity. Hence, the scheduling process in the model is simulated in the Decider entity. This is the crucial part in this simulation.

The Decider instance connects to two different Test instances which simulate the blood test and X-ray scan in the hospital. The rate of events flowing from Decider to Test is controlled by a scheduling policy embedded in the Decider entity. The queue length and the response time at the instances of Test entity reflect the performance of current scheduling policy. Hence, in the Test entity, related design for obtaining measurements are created to monitor each event passing through this entity and record all necessary information in order to generate the final measurement results.

Moreover, the Test entity can also distinguish the type of event by reading its tag. For the events Patient BX who need both blood test and X-ray scan, if an event has not completed any test yet, the event will be re-tagged by a Test instance after completing such first test, and then the event will be sent to the other Test instance directly; in the other Test instance, if the Test instance finds that the current event has a modified tag, it knows this event has already completed a test, so this event will discharge from the hospital after this test rather than being sent to the other test; if the Test instance finds that the current event has the original tag, it knows that this event has no test finished before, which will be sent to the other test after this test.

Finally, for all discharged patient events, they all enter a Wait instance in which events will pause for a period and then return to their original start instances that are those three instances of the Patient entity.

4.3.2 Simulation implementation with SimJava 2.0

In this simulation, the kinds of scheduling policies are modeled respectively under four different rate conditions which will be introduced in Section 4.5. Here, we use the simulation program of the static policy as an example to introduce the simulation process.

For the three kinds of scheduling policies, the simulation program structure is quite similar. The structure includes six functional java classes and a program implementation class. The functional classes are used to define the components in Figure 4.2 and implement the scheduling process, and the implementation class is used to initialize and combine the functional classes, and set up the related simulation parameters such as rates and the number of instances of each component. In the simulation, these classes are Source, Patient, Register, Scheduler, Test, Wait and a Main class for implementing the whole program.

In contrast to the Figure 4.2, Source class is designed to generate instances representing the components Source B, Source X and Source BX in the figure. Patient class is to generate three kinds of patient instances for components B, X and BX respectively in the figure. Scheduler class is the core part in this simulation which implements the scheduling process representing the component D in the figure. Test class is to simulate the BLOOD and X-RAY components which have the same program structure in the simulation. Wait class represents a wait process outside the hospital which is the implementation of component W. In the last Main class, all instances of each component are initialized with specified parameters, and all instances are linked together based on the action flow of each patient group described in the Figure 4.2.

In the appendix, the source code of each scheduling policy is detailed under the stable rate condition. Each simulation includes above demonstrated classes. The main difference between three scheduling policies exists in the Scheduler class.

4.4 Simulation model validation

Once the simulation is finished, the first thing is to validate the simulation with the original formal model. In this section, the validation is generated by comparing the simulation model with the original PEPA model and CMDL model. Without the validation, it cannot determine whether this simulation model is equivalent to the original PEPA model. To validate the simulation model, it is necessary to compare the results from the simulation with those of the PEPA model.

4.4.1 Validation with PEPA and CMDL on stochastic simulation

As mentioned previously, the current simulation model has a small difference in defining the rate of the decision action in the cooperation of the PEPA model. Therefore, in the validation, it is necessary to unify the rate type in the PEPA model, which is to specify all rates to be active rates. This exactly means that the third command line in the third block of the PEPA model in Section 4.2.2 is changed to:

$$PatientBX_2 \overset{def}{=} (decideB, r_{db}).PatientBX_3 + (decideX, r_{dx}).PatientBX_4$$

The detailed difference between passive rate and active rate has been illustrated in the previous section. Here, it is assumed that all actions in the PEPA model are active actions, and their rates are all active rates. With this assumption, the simulation model should be equivalent to the PEPA model.

To validate the simulation, both the PEPA model and its corresponding CMDL model must be employed for comparison, because the CMDL model has been used in the previous work to carry out an ODE analysis with function rates. Because the fluid flow approximation is not accurate under a small scale population, stochastic simulation is used instead to carry out measurements based on the PEPA and the CMDL model.

In order to start measurements, all parameters used for the PEPA, CMDL and simulation are initialized in the following table:

Arrival Rate	$\lambda_B = 10.0 \quad \lambda_X = 10.0 \quad \lambda_{BX} = 10.0$
Service Rate	$\mu_B = 8.0 \quad \mu_X = 4.0$
Decision Rate	$r_{db} = 1.0 \quad r_{dx} = 1.0$
Register Rate	$r_{register} = 2000.0$
Wait Rate	$r_{wait} = 2000.0$

Table 4. 1

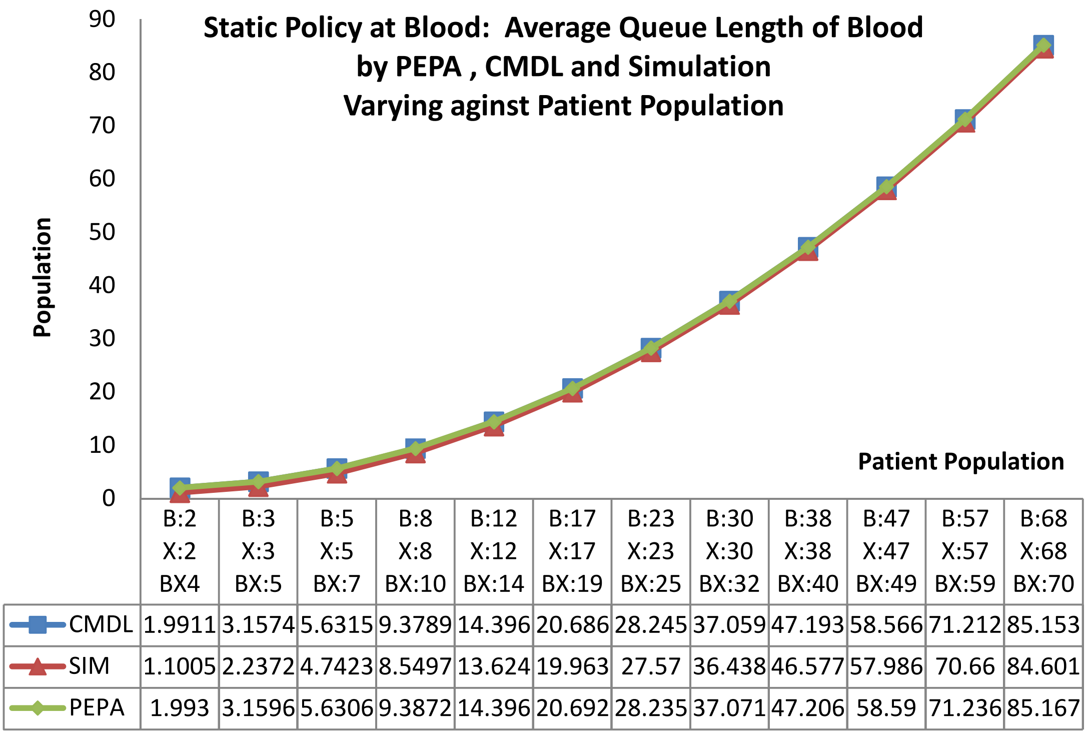
In the measurements of the simulation, the average queue length at BLOOD and X-RAY component will be measured to compare with that in the PEPA and CMDL model. The queue length will be measured against the varying patient population. Before measurements, the environmental conditions of the simulation are set as:

PEPA & CMDL	Discrete Event Simulation
• Stochastic Simulation: Gillespies Stochastic Algorithm • Simulation time: 200 • Number of data points: 200 • Replications: 5000 • Confidence interval: 0.95	• Simulation time: 15000 • Replications: 30 • Confidence interval: 0.95
Stochastic simulation and discrete event simulation are executed on Eclipse platform.	

Table 4. 2

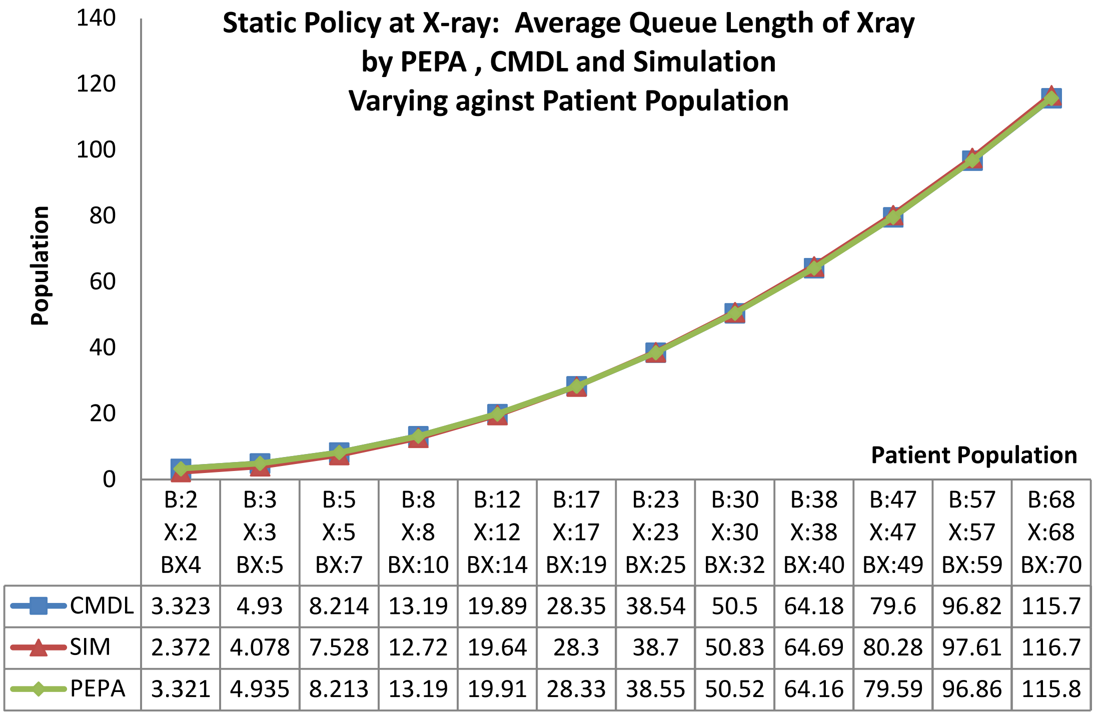
Regarding the above setup, the validation for the discrete event simulation is created by comparing the average queue length of both BLOOD and X-RAY components based on three scheduling policies. For the static policy, all three types of models, PEPA, CMDL and the simulation, can be compared together, because the static policy only use constant rates which are supported by the PEPA analyzer. However, in the model of dynamic policies, functional decision rates have been used to achieve the dynamic scheduling process. Because the current PEPA analyser does not support analysis for the function rates, the comparison can only performed between the CMDL model and the simulation. More information about transformation from PEPA to CMDL model has been introduced in Chapter 3. In this section, the validation will proceed with four steps on account of three scheduling policies.

Firstly, it is necessary to compare PEPA, CMDL models with the simulation model based on the static scheduling policy. Measurements are obtained on average queue length at each service component.



	B:2 X:2 BX4	B:3 X:3 BX:5	B:5 X:5 BX:7	B:8 X:8 BX:10	B:12 X:12 BX:14	B:17 X:17 BX:19	B:23 X:23 BX:25	B:30 X:30 BX:32	B:38 X:38 BX:40	B:47 X:47 BX:49	B:57 X:57 BX:59	B:68 X:68 BX:70
CMDL	1.9911	3.1574	5.6315	9.3789	14.396	20.686	28.245	37.059	47.193	58.566	71.212	85.153
SIM	1.1005	2.2372	4.7423	8.5497	13.624	19.963	27.57	36.438	46.577	57.986	70.66	84.601
PEPA	1.993	3.1596	5.6306	9.3872	14.396	20.692	28.235	37.071	47.206	58.59	71.236	85.167

Figure 4.3 Average queue length at Blood for static scheduling policy

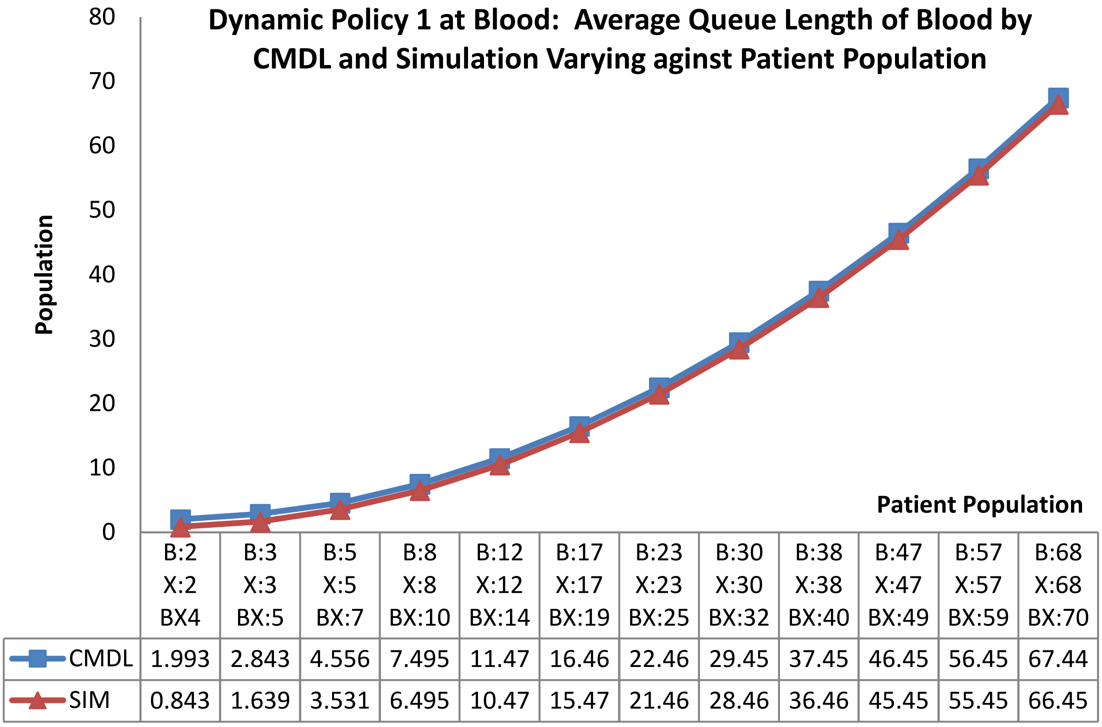


	B:2 X:2 BX4	B:3 X:3 BX:5	B:5 X:5 BX:7	B:8 X:8 BX:10	B:12 X:12 BX:14	B:17 X:17 BX:19	B:23 X:23 BX:25	B:30 X:30 BX:32	B:38 X:38 BX:40	B:47 X:47 BX:49	B:57 X:57 BX:59	B:68 X:68 BX:70
CMDL	3.323	4.93	8.214	13.19	19.89	28.35	38.54	50.5	64.18	79.6	96.82	115.7
SIM	2.372	4.078	7.528	12.72	19.64	28.3	38.7	50.83	64.69	80.28	97.61	116.7
PEPA	3.321	4.935	8.213	13.19	19.91	28.33	38.55	50.52	64.16	79.59	96.86	115.8

Figure 4.4 Average queue length at X-ray for static scheduling policy

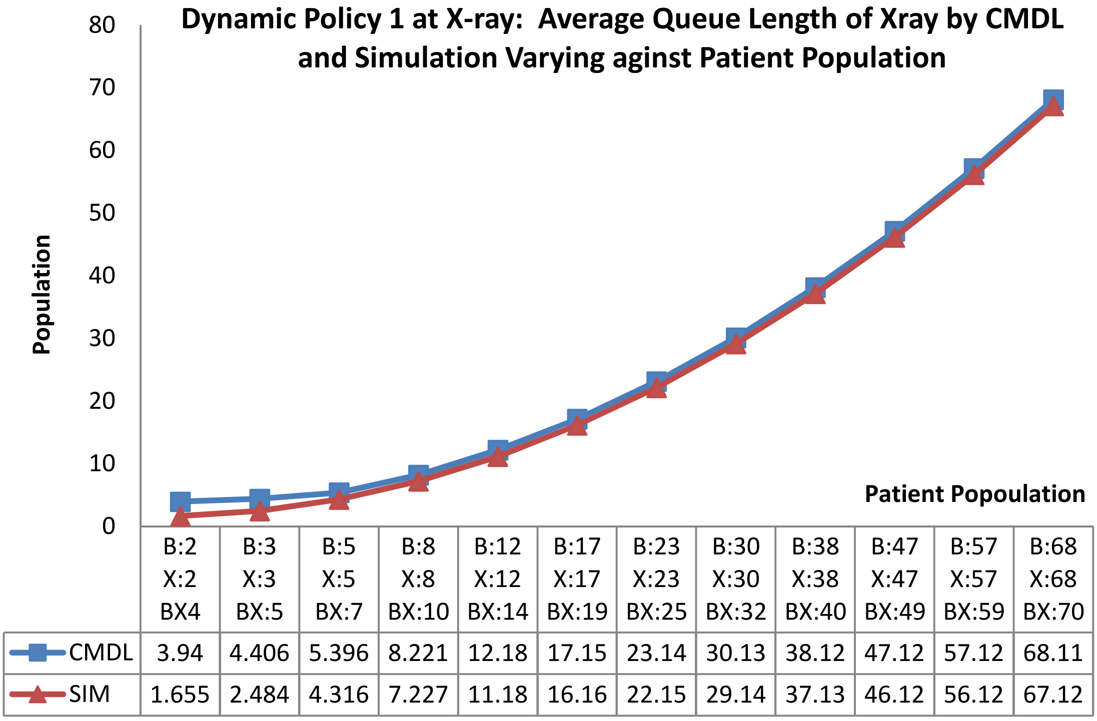
Figure 4.3 and Figure 4.4 show the situation of three model types for the static policy at Blood and X-ray component respectively. They depict the average queue length of Blood and X-ray respectively in three models. As can been seen from two diagrams, three lines almost overlap. Regarding the exact simulation figures in both diagrams, the PEPA model and the CMDL model only have their difference in the level 0.01. For the discrete event simulation, the difference between the simulation itself and others is a bit greater at the beginning of lines which has less patient population. This difference in the start point is around 1.0. With the increase of population, the difference has a bit fluctuation but no more than 1.0. At the end of line, when the population of patients increases to a larger scale, this difference remains in the level 1.0. At the Blood component, the simulation model has a bit less queue length than both PEPA and CMDL models. However, at the X-ray component, the line for the simulation model is slightly above that of other two models. Thus, it is clear that the difference between the simulation and other models becomes unobvious with the growth of the population. Consequently, the simulation model for the static policy can be considered consistent with the original PEPA model.

Secondly, figures from the simulation and the CMDL model are collected based on the dynamic policy 1. It is worth mentioning that the validation is on the basis of the comparison between the simulation and the CMDL mode as the PEPA model cannot support the analysis with function rates.



	B:2 X:2 BX4	B:3 X:3 BX:5	B:5 X:5 BX:7	B:8 X:8 BX:10	B:12 X:12 BX:14	B:17 X:17 BX:19	B:23 X:23 BX:25	B:30 X:30 BX:32	B:38 X:38 BX:40	B:47 X:47 BX:49	B:57 X:57 BX:59	B:68 X:68 BX:70
CMDL	1.993	2.843	4.556	7.495	11.47	16.46	22.46	29.45	37.45	46.45	56.45	67.44
SIM	0.843	1.639	3.531	6.495	10.47	15.47	21.46	28.46	36.46	45.45	55.45	66.45

Figure 4.5 Average queue length at Blood for dynamic scheduling policy 1

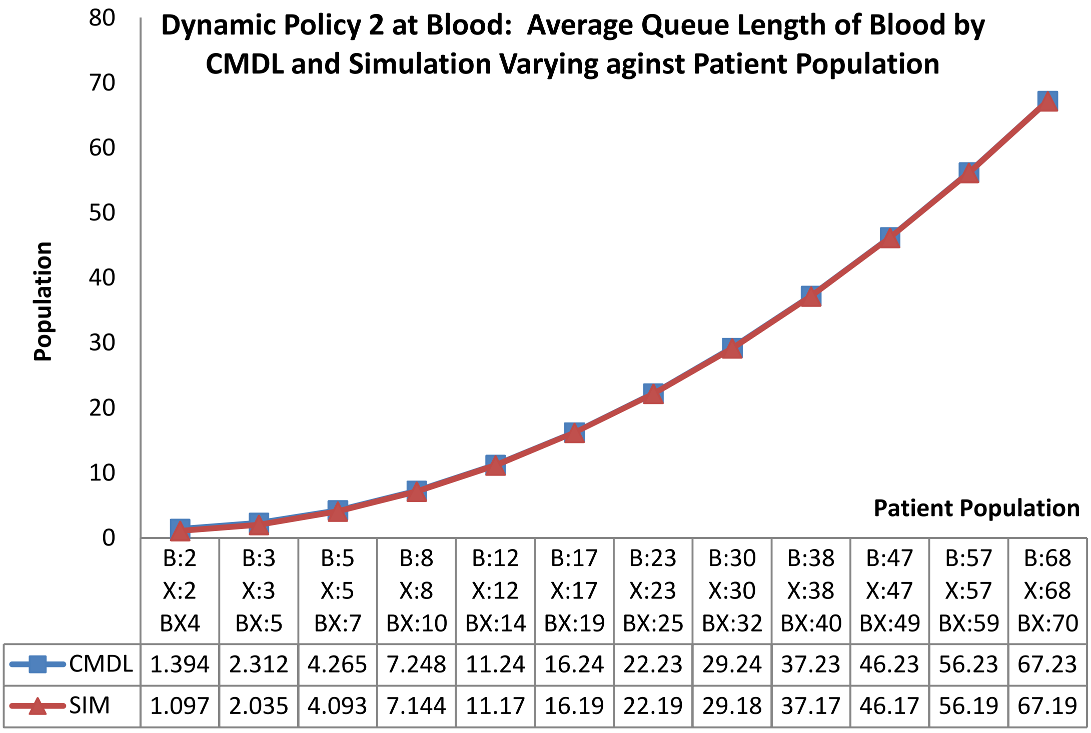


	B:2 X:2 BX4	B:3 X:3 BX:5	B:5 X:5 BX:7	B:8 X:8 BX:10	B:12 X:12 BX:14	B:17 X:17 BX:19	B:23 X:23 BX:25	B:30 X:30 BX:32	B:38 X:38 BX:40	B:47 X:47 BX:49	B:57 X:57 BX:59	B:68 X:68 BX:70
CMDL	3.94	4.406	5.396	8.221	12.18	17.15	23.14	30.13	38.12	47.12	57.12	68.11
SIM	1.655	2.484	4.316	7.227	11.18	16.16	22.15	29.14	37.13	46.12	56.12	67.12

Figure 4.6 Average queue length at X-ray for dynamic scheduling police 1

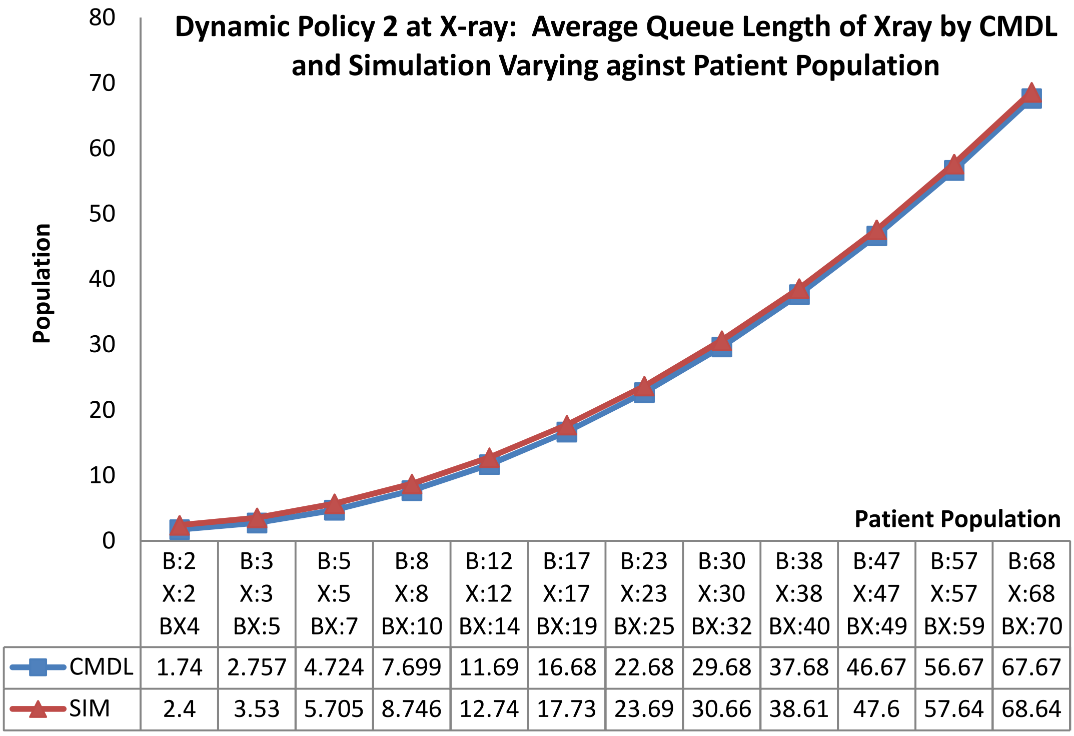
Figure 4.5 and Figure 4.6 illustrate the average queue of the Blood and X-ray in the simulation and CMDL model with dynamic policy 1. From two diagrams, the blue line representing the CMDL model always sits above the red line for the simulation; and two lines branch obviously at the beginning in which the total population in the system is no more than 20. Such a difference is just about 1.0 at Blood component and 2.0 at X-ray component. However, with the increase of the population, the difference between two models for Blood and X-ray component reduces to roughly 1.0. Consequently, for the modelling of the dynamic scheduling policy 1, the simulation model can be considered as consistent with the CMDL model especially when the total population in the system is greater than 20.

Finally, the validation for the simulation model of the dynamic policy 2 is different from the previous policies, because in this simulation, the dynamic policy 2 has different mechanism compared with this policy in PEPA and CMDL models. The reason about this difference has been demonstrated in Section 4.1.3. However, it is still necessary to accomplish a comparison between the simulation model and the CMDL model so as to find out how much difference exists between two types of models.



	B:2 X:2 BX4	B:3 X:3 BX:5	B:5 X:5 BX:7	B:8 X:8 BX:10	B:12 X:12 BX:14	B:17 X:17 BX:19	B:23 X:23 BX:25	B:30 X:30 BX:32	B:38 X:38 BX:40	B:47 X:47 BX:49	B:57 X:57 BX:59	B:68 X:68 BX:70
■ CMDL	1.394	2.312	4.265	7.248	11.24	16.24	22.23	29.24	37.23	46.23	56.23	67.23
▲ SIM	1.097	2.035	4.093	7.144	11.17	16.19	22.19	29.18	37.17	46.17	56.19	67.19

Figure 4.7 Average queue length at Blood for dynamic scheduling policy 2



	B:2 X:2 BX4	B:3 X:3 BX:5	B:5 X:5 BX:7	B:8 X:8 BX:10	B:12 X:12 BX:14	B:17 X:17 BX:19	B:23 X:23 BX:25	B:30 X:30 BX:32	B:38 X:38 BX:40	B:47 X:47 BX:49	B:57 X:57 BX:59	B:68 X:68 BX:70
CMDL	1.74	2.757	4.724	7.699	11.69	16.68	22.68	29.68	37.68	46.67	56.67	67.67
SIM	2.4	3.53	5.705	8.746	12.74	17.73	23.69	30.66	38.61	47.6	57.64	68.64

Figure 4.8 Average queue length at X-ray for dynamic scheduling police 2

Figure 4.7 and Figure 4.8 show the average queue length at Blood and X-ray of two kinds of models with the dynamic policy 2. In Figure 4.7, the two lines, at Blood component, have tiny difference even in the area with smaller population; moreover, the red line for the simulation is a bit lower than the blue line for the CMDL model. This difference decreases from about 0.3 at the first data point to a smaller value around 0.1 after the third data point. From the fourth data point on, the difference between two lines stays approximately at 0.1.

However, at component X-ray, namely Figure 4.8, the situation of lines is a bit different, because the difference between two lines is a bit larger than that in Figure 4.7. In contrast to the location of lines at Blood component in Figure 4.7, the queue length of the simulation model becomes greater than that in the CMDL model. The difference is about 0.7 at the first three data points, and then increases to about 1.0 after the third data point. Hence, although the simulation model has different design scheme for the dynamic policy 2, it still shows approximate performance compared with the original dynamic policy 2 in the PEPA model.

4.4.2 Validation with CMDL on fluid flow approximation

The previous section has completed the validation of the simulation model by comparing it with the stochastic simulation based PEPA model and CMDL model. This section aims to compare the simulation model with CMDL model using the fluid flow approximation. To generate the results from the fluid flow approximation, the previous CMDL model must be modified with the same parameters as well as the design for the dynamic policy 2 in the CMDL model. Figure 4.9, Figure 4.10 and Figure 4.11 represent the comparison between the simulation and the CMDL model with the fluid flow analysis. The comparison is organized in terms of the three scheduling policies respectively and the measurements are based on the average queue length at the X-ray component. The measurements results at the Blood component are almost the same as the results at the X-ray component according to the measurements.

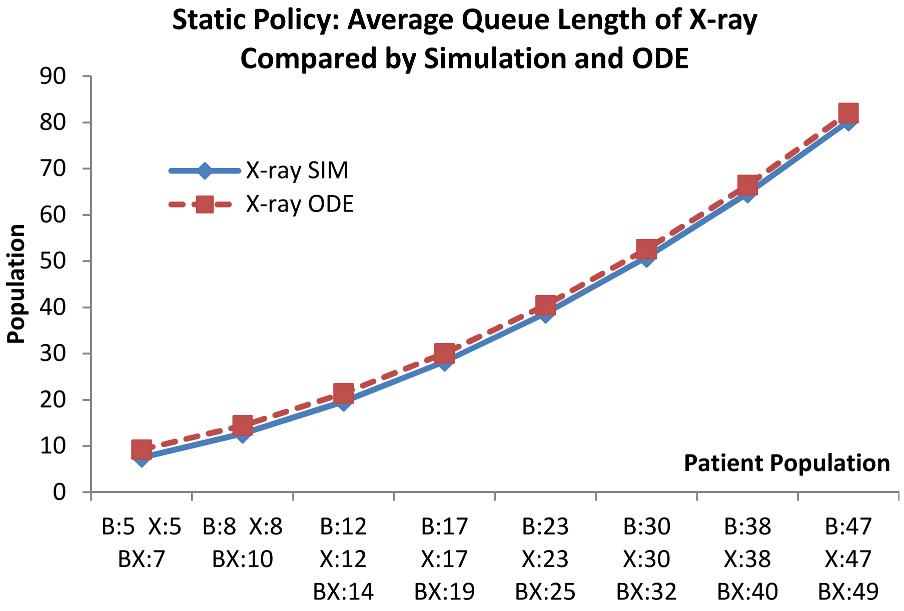


Figure 4.9 Average queue length at X-ray for static policy

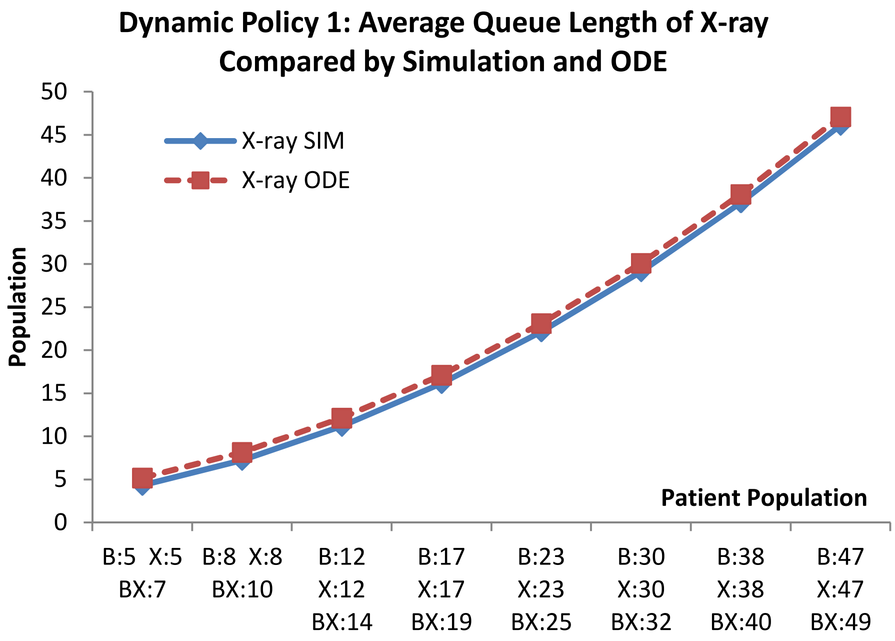


Figure 4.10 Average queue length at X-ray for dynamic policy 1

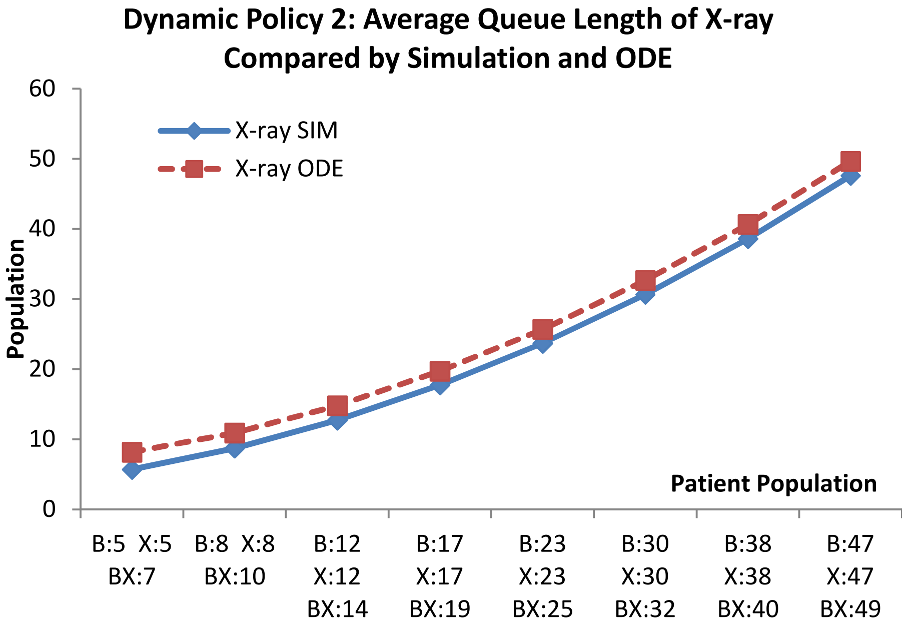


Figure 4. 11 Average queue length at X-ray for dynamic policy 2

As viewed in Figure 4.9, 4.10 and 4.11, the red dotted lines standing for the results of ODE analysis are close to the blue lines representing the simulation measurement results. For the static policy in Figure 4.9, the gap between two lines is roughly between 1.5 and 2.0; in Figure 4.10, for the dynamic policy 1, this difference is just around 1.0; however, for the dynamic policy 2 in Figure 4.11, the difference is greater than that in the preceding two diagrams, which approximates to 2.0, but such difference is still acceptable.

Another interesting point is that the ODE analysis is almost as accurate as the discrete event simulation even though the patient flow is in the condition of discrete population, and sometimes in small scale population size. However, it is worth emphasizing that the difference between the simulation and the ODEs in each diagram remains stable with the increase of the patient population; however, when the population is in a very low level, the difference between the simulation and the ODEs becomes greater. Thereby, the ODE based fluid flow analysis is not a good choice for the situation with a low level patient population.

To sum up, through the investigation and comparison of performance between the simulation model and the PEPA, CMDL models, we find out this simulation has high accuracy and equivalency with the original PEPA model. Consequently, in the following study, this simulation model can be applied to measure the performance of the scheduling policies in a discrete situation.

4.5 Measurement and analysis

This section will generate a set of measurement results based on the previous defined simulation model in order to compare three scheduling policies. To investigate the performance of the scheduling policies, four different types of system environments are assumed: system with stable arrival rate and service rate; system with varying arrival rate and stable service rate; system with stable arrival rate and varying service rate, and system with varying arrival rate and service rate. Then the comparison will be implemented in such four situations respectively, and measurements are obtained in terms of the increase of patient population or arrival rate. In the simulation measurements, the Eclipse platform is utilized to execute the simulation program. Simulation time is set to 50000 time units including 30000 warm up times. The number of replications is 30 and the confidence interval is 0.95.

Beside the simulation environment setup, it is also necessary to unify the model parameters seeing the following tables.

For the measurements in terms of the increase of the patient population, the parameters are initialized as follows:

Arrival Rate	$\lambda_B = 10.0 \quad \lambda_X = 10.0 \quad \lambda_{BX} = 10.0$
Service Rate	$\mu_B = 8.0 \quad \mu_X = 4.0$
Decision Rate	$r_{db} = 1.0 \quad r_{dx} = 1.0$
Register Rate	$r_{register} = 2000.0$
Wait Rate	$r_{wait} = 2000.0$

Table 4. 3

For the measurements in terms of the increase of the arrival rate, the parameters are specified as follows:

Patient Population	$P_B = 50 \quad P_X = 50 \quad P_{BX} = 50$
Service Rate	$\mu_B = 8.0 \quad \mu_X = 4.0$
Decision Rate	$r_{db} = 1.0 \quad r_{dx} = 1.0$
Register Rate	$r_{register} = 2000.0$
Wait Rate	$r_{wait} = 2000.0$

Table 4. 4

4.5.1 Measurement with stable arrival rate and service rate

Regarding the analysis design, this section will compare the average queue length and average response time of each scheduling policy against the patient population and the arrival rate respectively under the model with the stable arrival rate and service rate.

All figures introduced in this section are measurements based on the component X-ray, because the scheduling policies at the Blood component have similar performance with that at the X-ray. Hence, in following sections, analysis is also based on the situation of the component X-ray.

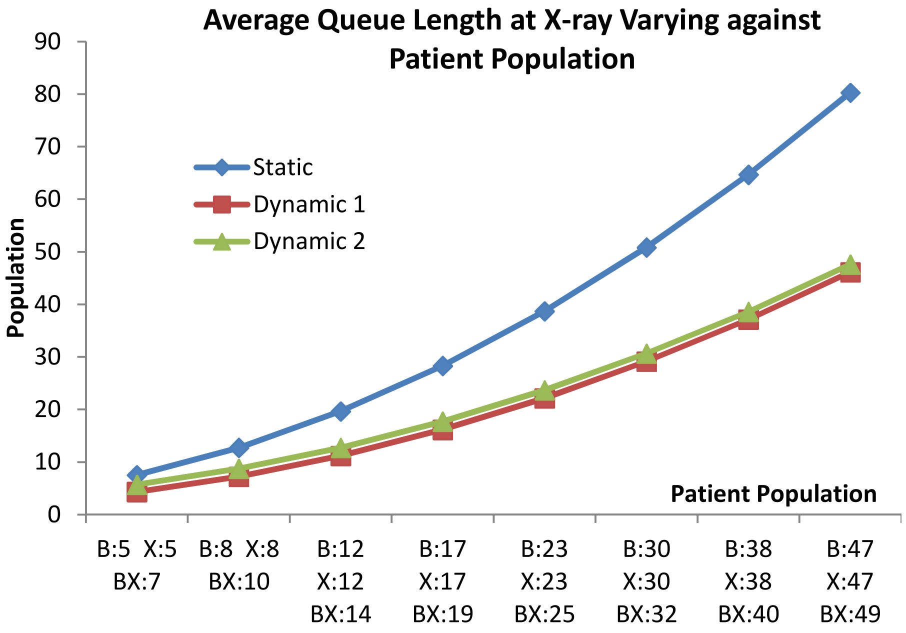


Figure 4. 12 Average queue length at X-ray against population under stable arrival rate and service rate

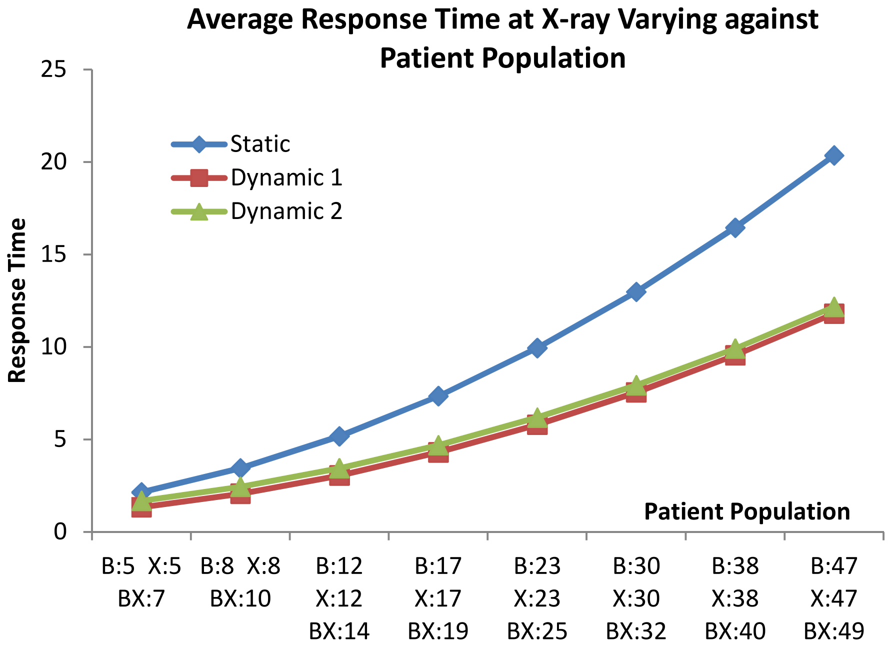


Figure 4. 13 Average response time at X-ray against population under stable arrival rate and service rate

Figure 4.12 and Figure 4.13 displays the change of average queue length and average response time respectively against the varying population. All three policies have increased queue length and average response time with the growth of the patient population. Moreover, both diagrams clearly display that the static policy has more queue length and response time at each data point than the two dynamic policies. This means the dynamic polices have better performance in the scheduling process than the static policy with the increase of the population. Compared with static policy, the more population the system has the better the dynamic policies perform. Especially in the data points of larger population, the difference between the static policy and the dynamic policies is quite obvious. However, when the population is in a lower level which means that the patient flow in the system is in a discrete situation, for example, at the first population data point, it can be recognized that the blue point for the static policy still sits in a higher position than the two dynamic policies. This is the key point of the simulation which is to simulate and measure the performance of the three scheduling policies in a discrete situation.

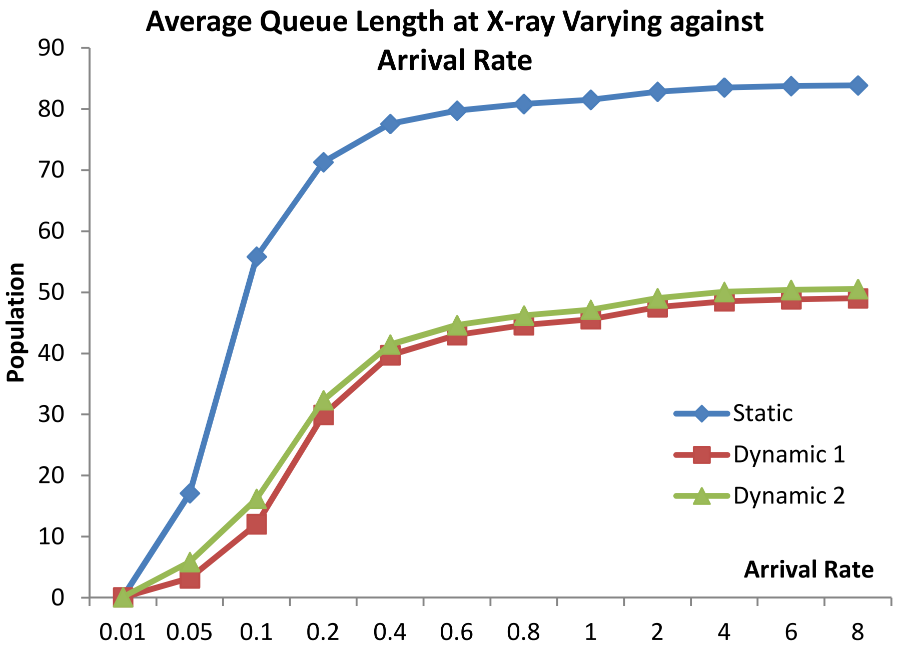


Figure 4. 14 Average queue length at X-ray against arrival rate under stable arrival rate and service rate

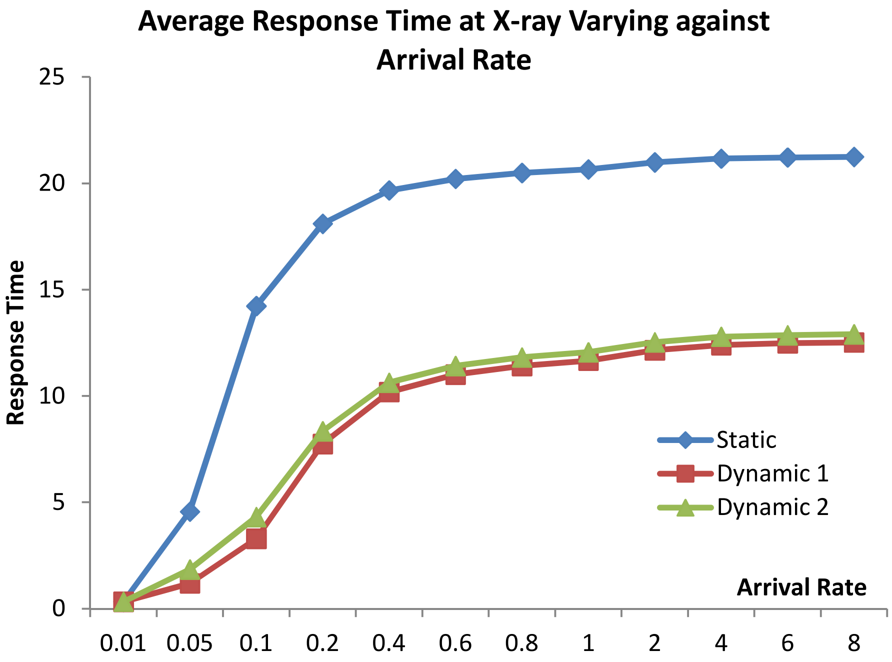


Figure 4. 15 Average response time at X-ray against arrival rate under stable arrival rate and service rate

Figure 4.14 and Figure 4.15 represent the average queue length and average response time of three scheduling policies at X-ray against the varying arrival rate. Both figures have similar trends at each line. The queue length and the response time have sharp increase with the growth of the arrival rate from 0.01 to 0.4. Thereafter, the queue length and the response time tend to stable with the increase of arrival rate. Thus, when the total population is fixed, the number of patients existing in the hospital finally reaches a stable level no matter how large the value of the arrival rate increases to. Furthermore, the most important point is that in these diagrams the dynamic policies has much better performance than the static policy, which means the dynamic policies has much less queue length and response time with the increase of the arrival rate. Even at the first data point 0.01, the static policy still has a bit greater queue length and response time than the dynamic policies though it is not that clear in the diagrams. For two dynamic policies, the situation is similar with that in Figure 4.12 and Figure 4.13, the dynamic policy 1 has smaller value in the average queue length and the response time.

For two dynamic policies in the above four figures, the dynamic policy 1 has shorter queue length and less response time than the dynamic policy 2, which is different from their performance in the Chapter 3. In Chapter 3, when the arrival rate and service rate are stable, the dynamic policy 2 has smaller average queue length than the dynamic policy 1. This is because of the modified design of the dynamic policy 2 in this chapter. The modification has been introduced in Section 4.2. Consequently, it can be argued that the current dynamic policy 2 indeed has different performance compared with the previous dynamic policy 2 in Chapter 3.

Hence, in the situation of stable arrival rate and service rate, measurement results confirms that both dynamic policies much better performance than static policies no matter the patient flow in system is fluid or discrete. With the increase of population, the difference between static policy and dynamic policies continues growing up; but for the varying of arrival rate, the difference in performance comes to a balance.

In summary, according to the measurements in this section, it is clear that the dynamic scheduling policies has superior performance in contrast to the static scheduling policy, and when the system has stable arrival rate and service rate the dynamic policy 1 is the best scheduling policy of all.

4.5.2 Measurement with varying arrival rate and stable service rate

In this section, measurements will be carried out based on the varying arrival rate and stable service rate. This section aims to explore the average queue length and average response time of three scheduling policies at component X-ray against the varying population and arrival rate.

To generate the varying arrival rate, a sine function is applied to the original constant arrival rate, which is to multiply a sine function with the constant arrival rate and alter the constant arrival rate dynamically during the simulation run. The change of sine function is based on a loop from integer 1 to 1000. The variable of the sine function will be added by 1 after each sample of the distribution is generated in the simulation.

The functional arrival rates are specified as follows:

Arrival Rate of Patient B	$\lambda_B = 10.0 \times (6.0 \times \sin(0.01x + 1.0) + 6.0)$
Arrival Rate of Patient X	$\lambda_X = 10.0 \times (6.0 \times \sin(0.01x + 2.0) + 6.0)$
Arrival Rate of Patient BX	$\lambda_{BX} = 10.0 \times (6.0 \times \sin(0.01x + 3.0) + 6.0)$

Table 4. 5

According to the above specification, the dynamic arrival rates, based on original service rate 10.0, alter between 0 and 120.0 due to the value of amplitude namely 6.0; and the period of sine function is set to be very long by initializing a tiny value 0.01 for the angular frequency. This functional arrival rate models the situation that the hospital has a greater number of arrivals in the day time but just a few during the night. To distinguish the situation of each patient group, the sine function is initialized with different values for the phase, such as 1.0 for Patient B, 2.0 for Patient X and 3.0 for Patient BX. These initializations simulate three types of patients arriving in the hospital with different rates at the same time. Thus, the value of the arrival rates circulates with the use of the sine function until the end of the simulation.

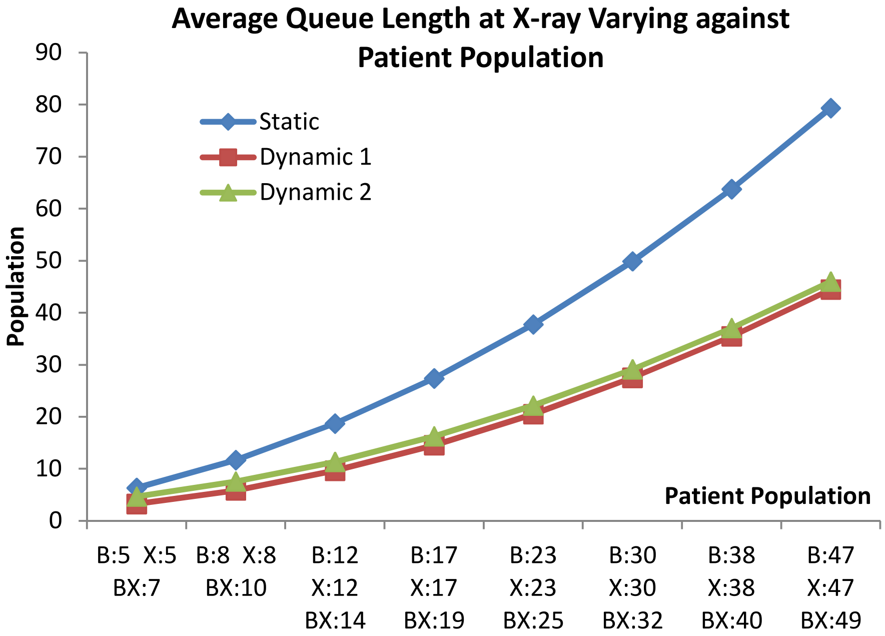


Figure 4. 16 Average queue length at X-ray against population under varying arrival rate and stable service rate

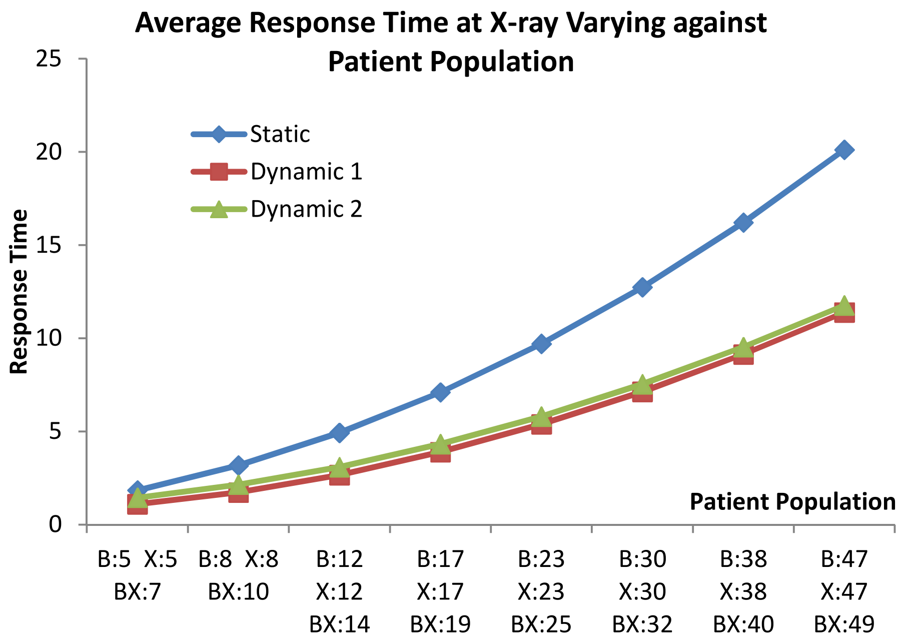


Figure 4. 17 Average response time at X-ray against population under varying arrival rate and stable service rate

Figure 4.16 and Figure 4.17 illustrate the average queue length and average response time at X-ray respectively against the varying patient population. Obviously, in both figures the dynamic policies have much better performance than the static policy especially with the increase of the population, which means that the dynamic policies have shorter average queue length and less average response time in measurements. When small populations exists in the system, for example, at the first data point in the diagram which generates a discrete situation in the system, the dynamic policies still have a bit better performance than the static policy, though the difference is just about 2 to 3 for the average queue length and 0.4 to 0.7 for the average response time. This difference enlarges dramatically with the growth of the population.

In addition, for the two dynamic policies, it is clear that the dynamic policy 1 performs better than the dynamic policy 2 in the scheduling process. Referring to the details in Figure 4.16, for the average queue length, the dynamic policy 1 is roughly 1.5 shorter than the dynamic policy 2 on average; meanwhile, in Figure 4.17, the average response time of the dynamic policy 1 is about 0.4 less than that of the dynamic policy 2. Thus, the dynamic policy 1 is the best policy in the condition of the varying arrival rate and stable service rate.

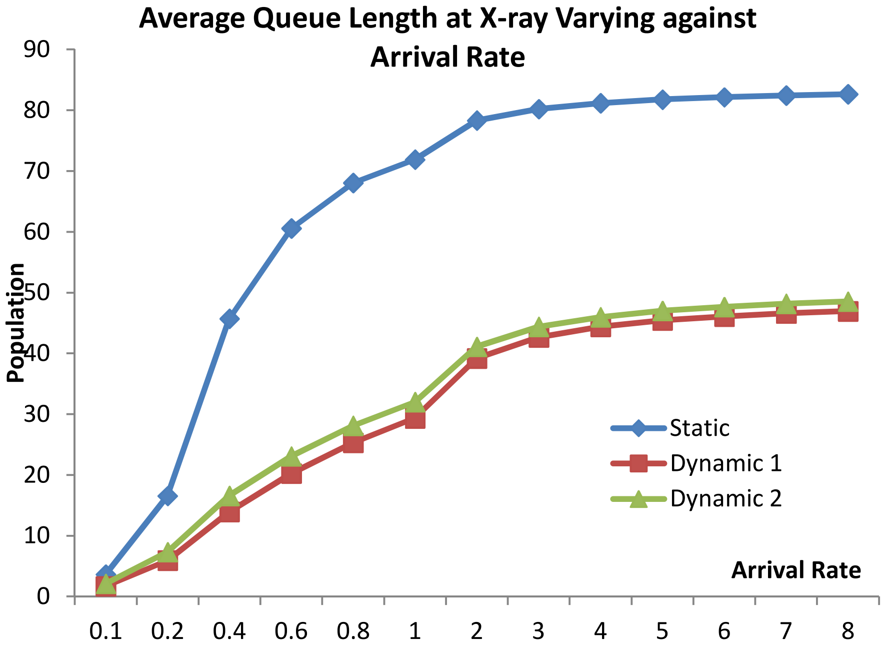


Figure 4. 18 Average queue length at X-ray against arrival rate under varying arrival rate and stable service rate

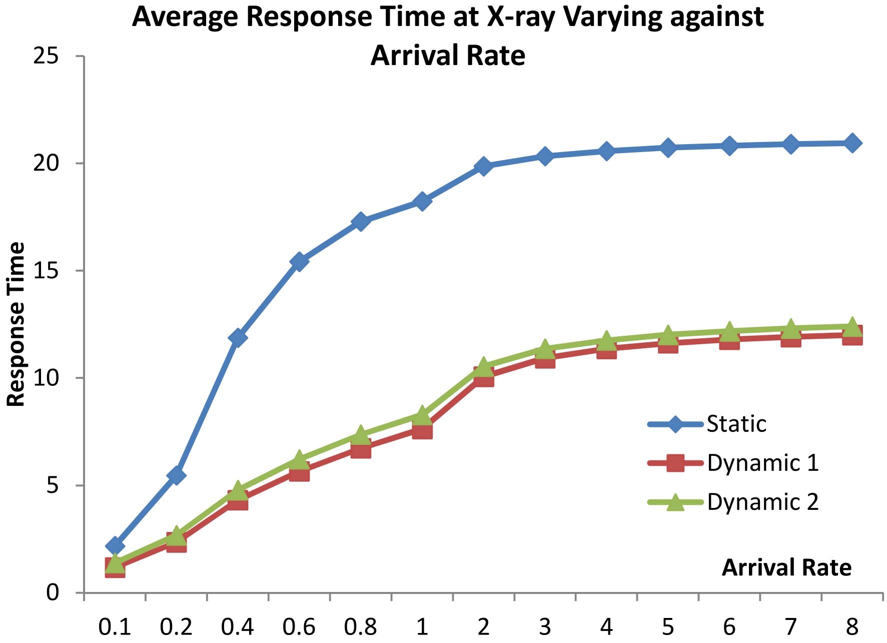


Figure 4. 19 Average response time at X-ray against arrival rate under varying arrival rate and stable service rate

Figure 4.18 and Figure 4.19 show the average queue length and average response time at X-ray of three scheduling policies against the varying arrival rate. Both figures represent similar trends, which means the average queue length and average response time have dramatic increase when the arrival rate increases from 0.1 to 2.0. Thereafter, the queue length and the response time tend to a slight growth with the increased arrival rate. Such slight growth is due to the limited patient population in the system. From the figures, the dynamic policies obviously perform better than the static policy in the scheduling process. With the growth of the arrival rate, the difference between the dynamic policies and the static policy continues to enlarge until reaching a stable level. Compared with the dynamic policy 2, the dynamic policy 1 is still a bit better in the queue length and response time, which has difference roughly between 0.5 and 1.5 for the average queue length and between 0.2 and 0.4 for the average response time.

In summary, in the system with the varying arrival rate and stable service rate, the dynamic policy 1 based on the queue length is the optimal policy which means this policy is more suitable in the condition of the varying arrival rate. This conclusion about the feature of the dynamic policy 1 verifies the features of the policy demonstrated analyzed by the fluid flow approximation in Chapter 3.

4.5.3 Measurement with stable arrival rate and varying service rate

This section aims to explore the performance of three scheduling policies when only the service rate alters during the simulation process but the arrival rate remains stable, which simulates that the hospital usually provides fast response and service during the day time; however, in the night, just a few staff work in the hospital thus the service rate is quite low at this moment.

As introduced in Section 4.4.2, a similar method is used to alter the service rate, but this time a cosine function is utilized to dynamically modify the service rate. Such cosine function still alters with the change of an integer value varying between 1 and 1000 cyclically.

Details of the service rates are specified as follows:

Service Rate of Blood Test	$\mu_{\text{Blood}} = 8.0 \times (6.0 \times \cos(0.01x + 1.0) + 6.0)$
Service Rate of X-ray Scan	$\mu_{\text{X-ray}} = 4.0 \times (6.0 \times \cos(0.01x + 2.0) + 6.0)$

Table 4. 6

Regarding the above tables, the service rates of blood test and X-ray scan are dynamically altered based on their original service rates, namely 8.0 or 4.0. The rate variation is between 0 and 12 times of the original service rate by setting the amplitude to 6.0, which means the value of the service rate changing from 0 to 96.0 for the blood test and from 0 to 48.0 for the X-ray scan. Such a large variation simulates the difference of the serving ability between the day time and the night. The period of the varying service rate is also large because the use of a small number for the angular frequency, namely 0.01. Moreover, different values are used for the phase parameters in case the rates of the blood and the X-ray come to a same value at the same time point.

The following figures illustrate the performance of the scheduling policies on the above conditions. Measurements are generated for the average queue length and average response time at the X-ray against the varying population and arrival rate.

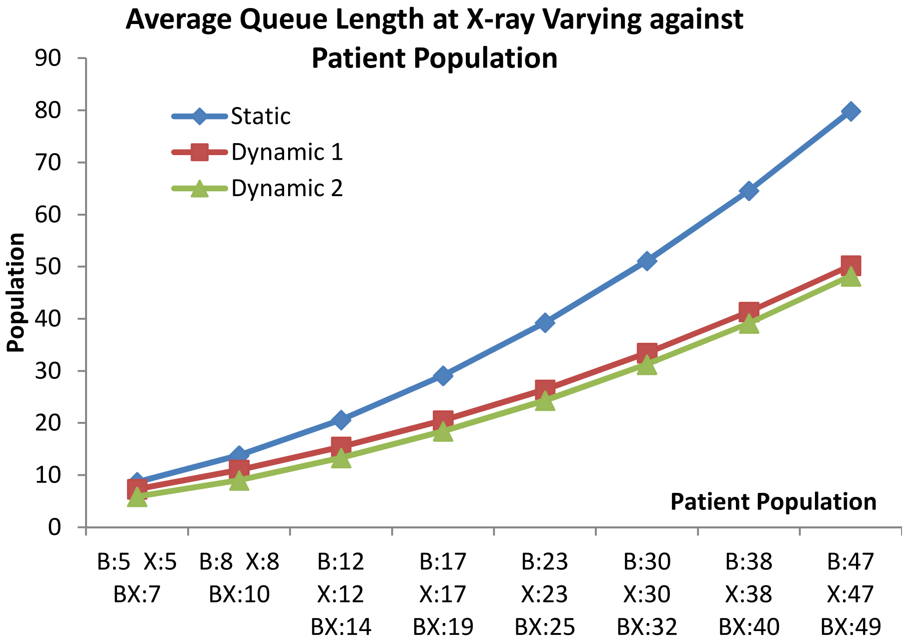


Figure 4. 20 Average queue length at X-ray against population under stable arrival pate and varying service rate

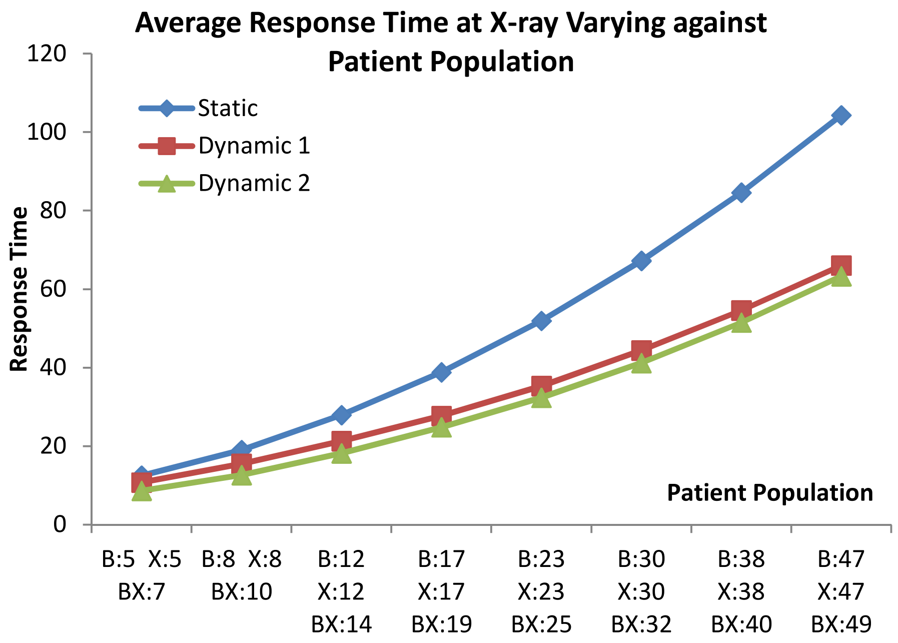


Figure 4. 21 Average response time at X-ray against population under stable arrival rate and varying service rate

Figure 4.20 and Figure 4.21 respectively represent the average queue length and the response time at X-ray component against the population. From the figures, the trend of lines continues going up with the increased population. The two dynamic policies have much shorter queue length and less response time compared with the static policy, and the difference between the static and the dynamics becomes greater with the increased population. These basic features keep similar with those diagrams in the previous section.

However, for lines representing the dynamic policies, their position is changed in current conditions. In contrast to Figure 4.16 and Figure 4.17, when the service rate becomes variable and the arrival rate remains constant, the dynamic policy 2 becomes the best policy. It has shorter average queue length and less response time at each data point than the dynamic policy 1. For the queue length, the dynamic policy 2 has about 2.0 in population less than the dynamic policy 1 except the first data point that has the value about 1.5. For the response time, the difference between them is about 3.0 on average, but at the first data point it is just around 2.0.

Such features are also revealed in the below Figures 4.22 and Figure 4.23, which display the average queue length and average response time at X-ray with the varying arrival rate.

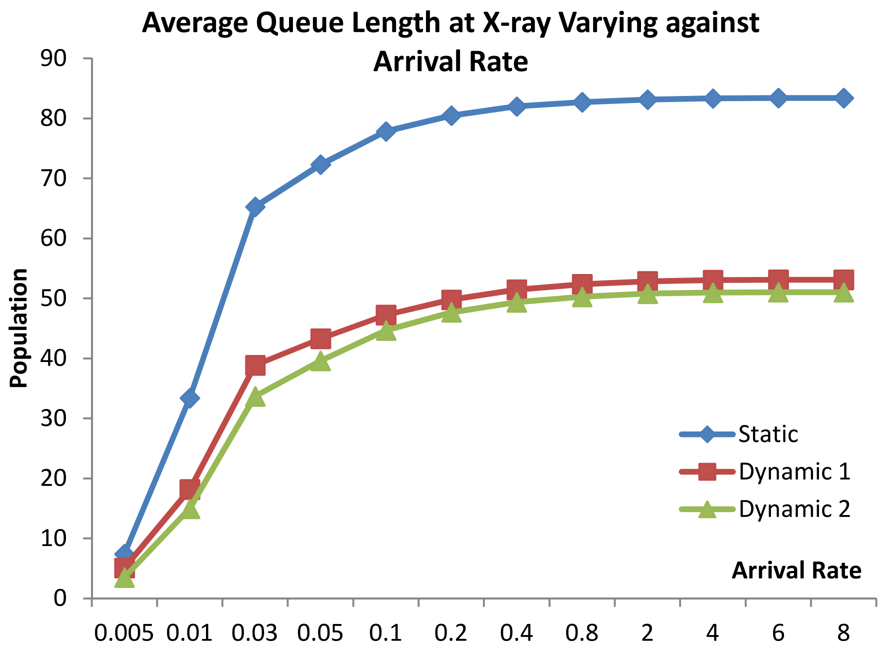


Figure 4. 22 Average queue length at X-ray against arrival rate under stable arrival rate and varying service rate

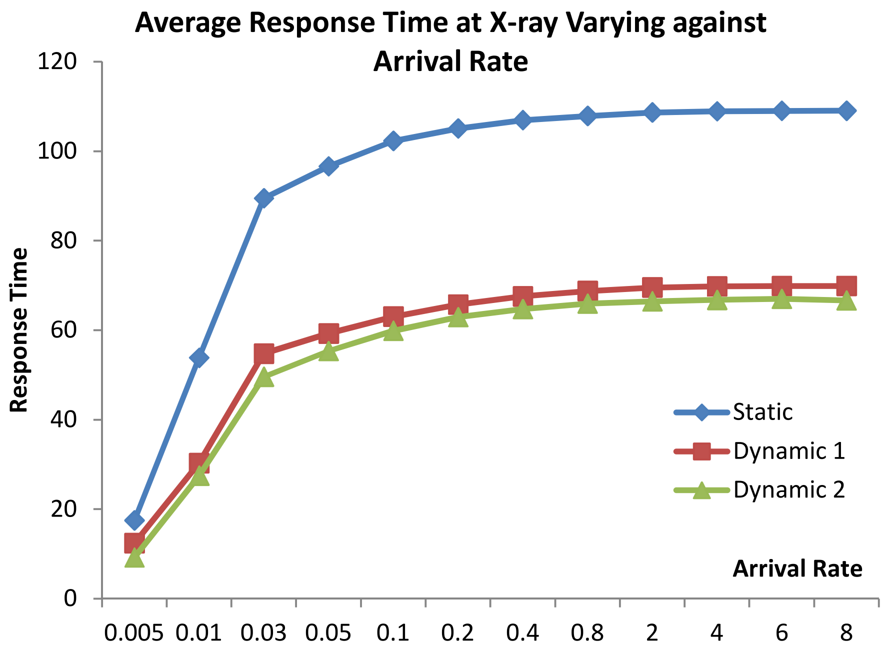


Figure 4. 23 Average response time at X-ray against arrival rate under stable arrival rate and varying service rate

As can be seen in Figures 4.22 and 4.23, all three scheduling policies have very sharp rise in the queue length and the response time with very tiny increase of the arrival rate from 0.005 to 0.1. From the point 0.2 on, three lines increase slowly and reach to be stable at large arrival rates. In addition, these figures also denote the dramatic disparity between the static scheduling policy and the dynamic scheduling policy. There is no doubt that the dynamic policies are much better than the static policy. The dynamic policy 2 is also superior to the dynamic policy 1, which has some miss-distance around 2.0 for the average queue length and roughly 3.0 for the average response. However, when the arrival rate is quite small, e.g. at point 0.03, the disparity of the two dynamic policies is even greater than the average value, which approximates to 5.0 for both average queue length and average response time.

Overall, in the condition with the varying service rate and stable arrival rate, the dynamic policy 2 is the best scheduling policy of all because of its shortest average queue length and average response time. The dynamic policy 1 is a bit inferior to the dynamic policy 2, but it is still much better than the static policy. Hence, it can be concluded that the dynamic scheduling policy 2 is most suitable policy in condition of the varying service rate.

4.5.4 Measurement with varying arrival rate and service rate

This section generates the last measurements of the chapter by initializing both arrival rate and service rate to be variable. In fact, it joins the conditions of the previous two sections. In this part, the values of the arrival rate and service rate are specified as follows:

Arrival Rate of Patient B	$\lambda_{B} = 10.0 \times (6.0 \times \sin(0.01x + 1.0) + 6.0)$
Arrival Rate of Patient X	$\lambda_{X} = 10.0 \times (6.0 \times \sin(0.01x + 2.0) + 6.0)$
Arrival Rate of Patient BX	$\lambda_{BX} = 10.0 \times (6.0 \times \sin(0.01x + 3.0) + 6.0)$
Service Rate of Blood Test	$\mu_{Blood} = 8.0 \times (6.0 \times \cos(0.01x + 1.0) + 6.0)$
Service Rate of X-ray Scan	$\mu_{X-ray} = 4.0 \times (6.0 \times \cos(0.01x + 2.0) + 6.0)$

Table 4. 7

All the above functional arrival rates and service rates remain the same as those used in the two preceding section (see Table 4.5 and Table 4.6) and all figures are generated in the same way as before.

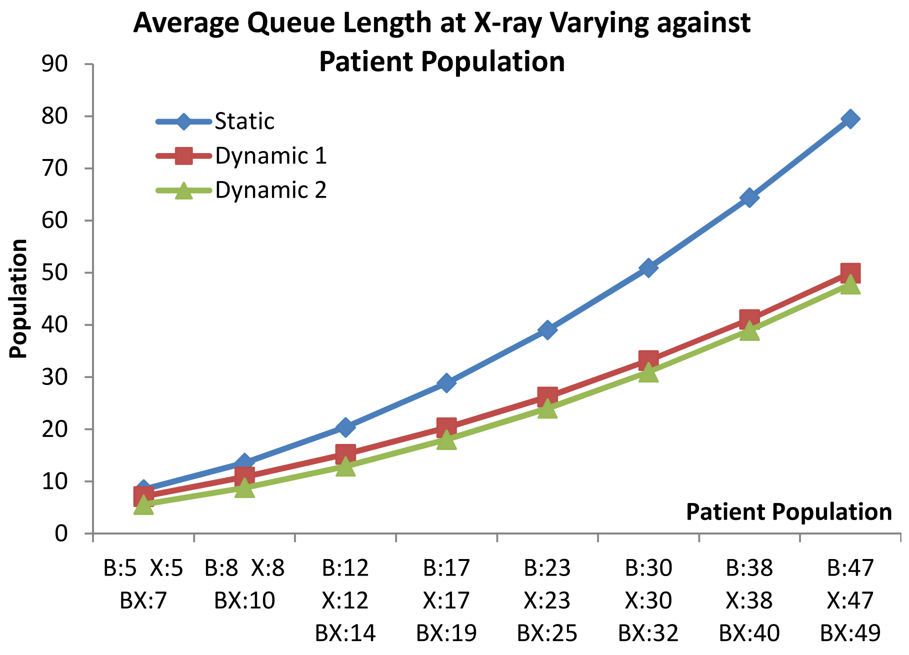


Figure 4. 24 Average queue length at X-ray against population under varying arrival rate and service rate

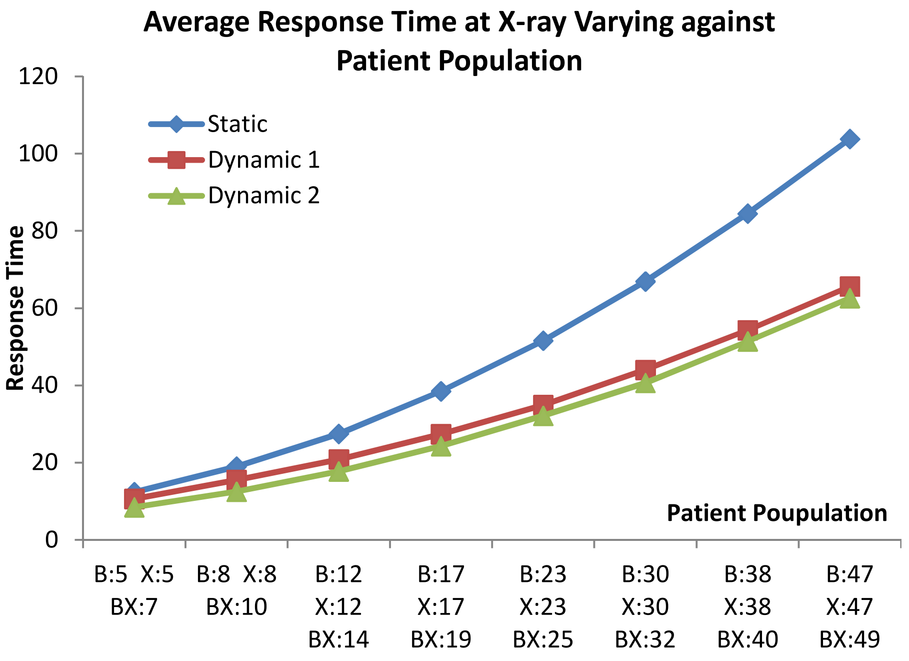


Figure 4. 25 Average response time at X-ray against population under varying arrival rate and service rate

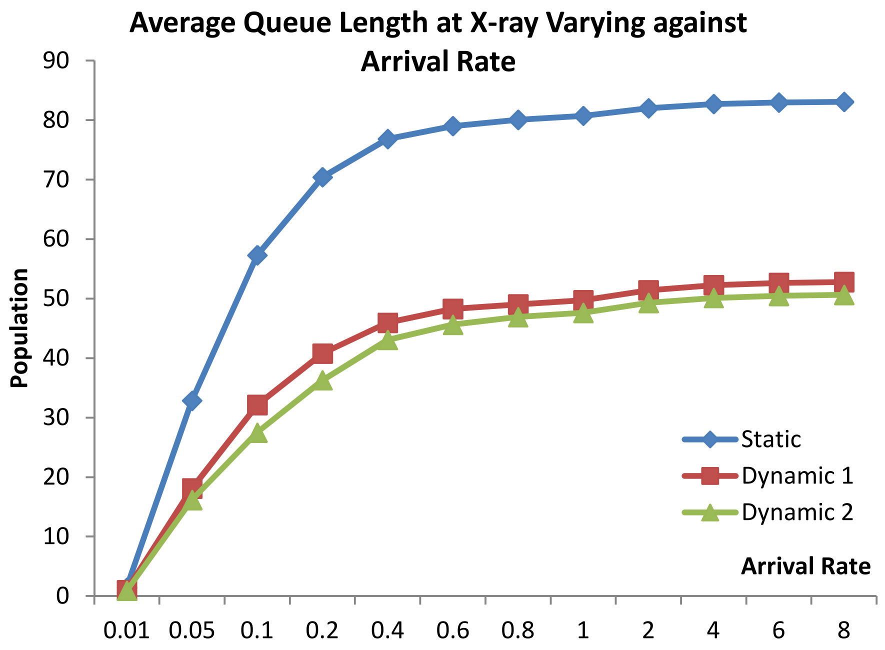


Figure 4. 26 Average queue length at X-ray against arrival rate under varying arrival rate and service rate

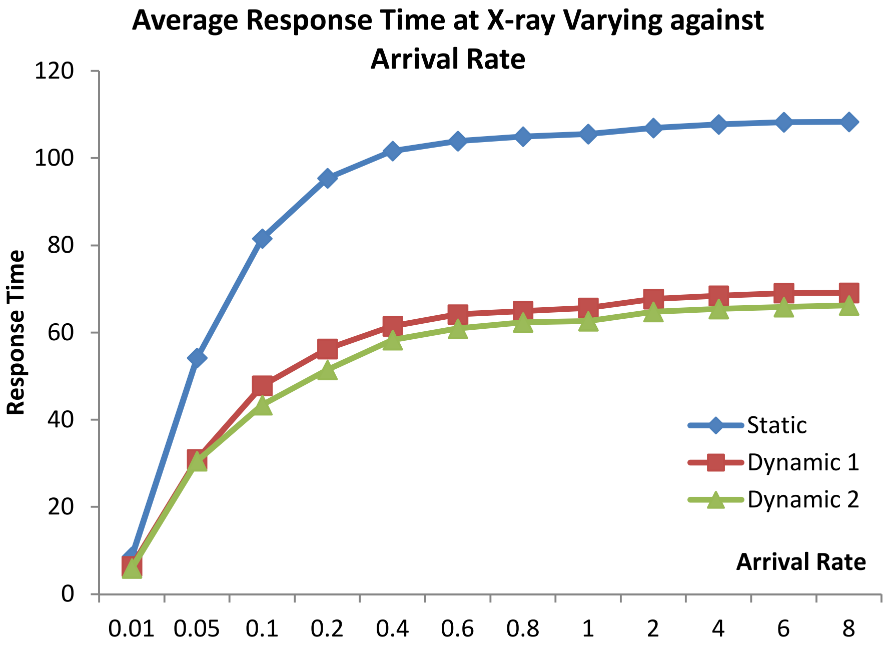


Figure 4. 27 Average response time at X-ray against arrival rate under varying arrival rate and service rate

Figure 4.24 and Figure 4.25 respectively present the average queue length and response time at X-ray with the varying population; meanwhile, Figure 4.26 and Figure 4.27 demonstrate the average queue length and response time varying against the arrival rate. No matter in which diagram of the four, the general trend of all lines is similar with that in figures of the preceding three sections, including the rise and the run of lines, as well as the similar features about the difference between the static policy and the dynamic policy or between the two dynamic policies. The repetitive details are not stated here.

Nevertheless, it is worth emphasizing that in this situation dynamic policy 2 has a bit better performance than dynamic policy 1 in both average queue length and average response time. However, this is not to say the dynamic policy 2 keeps this superior performance always in the situation of varying arrival rate and service rate. Thus, the best dynamic policy of the two candidate dynamic policies depends on the factor, either the arrival rate or the service rate, which generates more effect to the system. Therefore, if the varying arrival rate plays a main role which affects the patient flow more in system, now the queue length based dynamic policy 1 must be superior; conversely, if the varying service rate generates a more effective impact to the hospital in the system, the optimal policy must be the response time based dynamic policy 2.

Consequently, although in the above four figures the dynamic policy 2 is slightly better than the dynamic policy 1, it just testifies that the dynamic policy 2 has superior performance on the basis of the current system conditions. Henceforth, for a comprehensive model system especially with the dynamic arrival rate and service rate, the selection of the dynamic policies must be based on the real conditions of the system.

4.6 Conclusion

The main goal of this chapter is to investigate the performance of three scheduling policies in a discrete environment. To achieve the goals, Java based discrete event simulation is applied to complete the modelling and measurements.

To create the simulation model, a simulator is introduced that is named SimJava with Version 2.0. Thus, the simulation is built in keeping with the model scenario and the original PEPA model. Before starting measurements, the simulation model must be validated with the original PEPA model and the CMDL model. According to the validation, it can be certified that the simulation model can be considered equivalent to the PEPA model.

In the following parts, four groups of measurements are generated in four kinds of conditions based on the simulation. Consistent with the measurement results, the features of three scheduling policies can be summarized as follows:

- Static scheduling policy, which is the poorest scheduling policy of those considered. It always has the worst performance in each case of measurements.
- Dynamic scheduling policy 1, which has the optimal performance in condition that only the arrival rate is varying. As this policy has its scheduling mechanism altering with the transient queue length, it becomes the most efficient policy in the case of the varying arrival rate.
- Dynamic scheduling policy 2, which is the best policy while only the service rate is varying in the system. This is because that the dynamic policy 2 has a dynamic regulation design with the transient response time. Accordingly, this policy releases its excellent performance in the scheduling process under the environment having the varying service rate.
- For a comprehensive environment affected by the varying arrival rate and the varying service rate, the optimal candidate policy should come from the two dynamic policies. The selection of the dynamic policy depends on the real situation of the system conditions. If in such a system environment, when the variation of the system mainly refers to the incoming arrivals, the queue length based dynamic policy 1 must be adopted as the first choice. Conversely, for the system in which the main variation from the

very unstable service ability, the response time based dynamic policy 2 has no doubt to be a smart choice.

In conclusion, in accordance with the simulation measurement results, it is clear that the dynamic scheduling policies have excellent performance in contrast to the static scheduling policy in a discrete condition. Additionally, the two dynamic policies have their own features and advantages in different conditions. Thus, the use of the dynamic policies must be on the basis of the real system environment.

Chapter 5 Modelling of capacity and scheduling for a healthcare department

5.1 Introduction

This chapter aims to study capacity planning for the rheumatology department of Royal Hallamshire Hospital in Sheffield, United Kingdom, and also investigate the performance of the patient scheduling policy based on this department scenario. Capacity research is carried out from building a model in terms of the real workflow in rheumatology department. Most parameters used in the model are statistical data provided by the department. The focus of capacity research is to investigate the minimum number of consultants required and the assignment of consultants for different patient classes. Scheduling investigation is achieved by updating initial capacity model based on a scheduling strategy. As previously, the main modelling techniques used in the work are Performance Evaluation Process Algebra (PEPA) and Chemical Model Definition Language (CMDL).

Section 5.2 introduces the scenario and the main goal of the research. Section 5.3 describes the case study and model scenario. Section 5.4 gives the initial PEPA models according to the scenario. Section 5.5 refines the model based on an alternative workflow. Section 5.6 updates the model with a scheduling strategy. Section 5.7 briefly concludes the analysis.

5.2 Case study

This research is based on a scenario of rheumatology department of Royal Hallamshire Hospital in Sheffield. The scenario is provided by the rheumatology department through an interview with the staff of rheumatology department, which demonstrates the current work flow in this department. Rheumatology department runs the service for a number of patients with a fixed procedure. The situation in the rheumatology department is depicted in Figure 5.1.

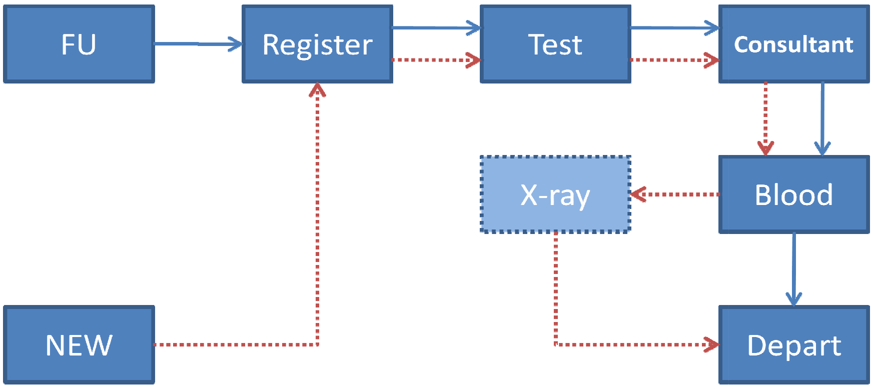


Figure 5.1 Rheumatology department workflow

As displayed in Figure 5.1, patients can be generally grouped in two classes in terms of the research objects, namely follow-up patients (FU) and new patients (NEW). These two patient flows are indicated in Figure 1 with a solid line for the follow-up and a dotted line for the new. Each rectangle represents a step point of the whole process. Both groups of patients must register for an appointment (Register) with the department first, then each patient has a general check up (Test) before meeting the consultants (Consultant). After seeing the consultants, follow-up patients go for a blood test (Blood) and leave the department (Depart). However, new patients must go to another department for an X-ray inspection (X-ray) after the blood test, and then leave the department.

5.3 Initial model construction and analysis

In this section, an initial model is specified based on the workflow of the Rheumatology department of Royal Hallamshire Hospital shown in Figure 5.1. This section aims to investigate the performance of the current workflow including the queuing and the capacity aspects. Analysis is conducted based to a CMDL model from the PEPA model. Main analysis techniques used in the section are the fluid flow approximation and the stochastic simulation for the validation.

5.3.1 Model construction

Model is initially created with PEPA in terms of the model description in the case study, because PEPA defines a formal modelling process and is convenient to model the whole process in the scenario. The use of PEPA is similar with that used in the scheduling models in Chapter 3. The PEPA model can be signified as:

$$\begin{aligned}
New &\overset{def}{=} (arriveNew, r_{arriveNew}).New\_reg \\
New\_reg &\overset{def}{=} (register, r_{register}).New\_test \\
New\_test &\overset{def}{=} (test, r_{test}).New\_con \\
New\_con &\overset{def}{=} (newCon, r_{newCon}).New\_blood \\
New\_blood &\overset{def}{=} (blood, r_{blood}).New\_xray \\
New\_xray &\overset{def}{=} (xray, r_{xray}).New\_depart \\
New\_depart &\overset{def}{=} (depart, r_{depart}).Stop
\end{aligned}$$

$$\begin{aligned}
FU &\overset{def}{=} (arriveFU, r_{arriveFU}).FU\_reg \\
FU\_reg &\overset{def}{=} (register, r_{register}).FU\_test \\
FU\_test &\overset{def}{=} (test, r_{test}).FU\_con \\
FU\_con &\overset{def}{=} (fuCon, r_{fuCon}).FU\_blood \\
FU\_blood &\overset{def}{=} (blood, r_{blood}).FU\_depart \\
FU\_depart &\overset{def}{=} (depart, r_{depart}).Stop
\end{aligned}$$

$$\begin{aligned}
Register &\overset{def}{=} (register, r_{register}).Register \\
Test &\overset{def}{=} (test, r_{test}).Test \\
Consultant\_New &\overset{def}{=} (newCon, r_{newCon}).Consultant\_New \\
Consultant\_FU &\overset{def}{=} (fuCon, r_{fuCon}).Consultant\_FU \\
Blood &\overset{def}{=} (blood, r_{blood}).Blood \\
Xray &\overset{def}{=} (xray, r_{xray}).Xray \\
Depart &\overset{def}{=} (depart, r_{depart}).Depart \\
Stop &\overset{def}{=} (stop, r_{stop}).Stop
\end{aligned}$$

$$\begin{gathered}
(New[m]||FU[n]) \bowtie_L \\
(Register[n_1]||Test[n_2]||Consultant\_New[n_3]||Consultant\_FU[n_4]||Blood[n_5] \\
||Xray[n_6]||Depart[n_7]||Stop[n_8]) \\
L=\{register, test, newCon, fuCon, blood, xray, depart, stop\}
\end{gathered}$$

The above model constructs a system based on the workflow displayed in Figure 5.1. The first block of the model defines the action flow of new patients, which starts from registration, and then goes through test, seeing consultants, blood test, x-ray test and finally departs from hospital. In the first block, New defines the initial status of the new patients who are coming to the hospital. The arriveNew action means that the new patients has arrived in the hospital and move the next state New_reg. New_reg represents the state waiting for registration at the reception. After the action register, the new patients move the next state that is the test. New_test defines this test activity with the action test. Similarly, New_con is the next state after New_test, and it defines the process that the new patients are served by the consultants. Thereafter, the new patients go to have a blood test which is the state New_blood, and then the X-ray scan follows the blood test. Thus, the state New_xray is defined after the New_blood. Finally, when the new patients complete all tests, they will carry out the last action that is the departure. So, New_depart is the last state in this action flow. Thereafter, this flow terminates at Stop state.

Similarly, the second model block defines the action process of follow-up patients, which is the same as that of new patients except for missing an x-ray inspection. The third block is to define all other components in the model system, which are the corresponding rectangles noted in Figure 5.1. Here, the consultants are defined in two types: *Consultant_New* and *Consultant_FU*. The last block specifies the cooperation of all components in the model as well as the instance number of each component. Stop component models a terminal in the system, in which all patients have finished all and stop there.

The PEPA model is created from the original scenario, but the following analysis is generated from a CMDL model which is transformed from this PEPA model via an analyzer Eclipse PEPA plug-in. The reason for using CMDL is owing to the limitation of the PEPA analyzer (Eclipse PEPA plug-in). The main limitation is that the current Eclipse based PEPA Plug-in cannot support the analysis of models using function rates although PEPA itself is able to support function rates. However, CMDL modelling language supports the analysis referring to functions. Thus, PEPA models using function rates must be converted to the equivalent CMDL model in order to carry out the analysis.

The equivalent CMDL model is displayed as:

```
arriveFU, FU -> FU_reg, r_arriveFU;
arriveNew, New -> New_reg, r_arriveNew;

register1, FU_reg + Register -> FU_test + Register,
[FU_reg/((New_reg+FU_reg)+c)*r_register*min((New_reg+FU_reg),Register)];
register2, New_reg + Register -> New_test + Register,
[New_reg/((New_reg+FU_reg)+c)*r_register*min((New_reg+FU_reg),Register)];

test1, New_test + Test -> New_con + Test,
[New_test/((New_test+FU_test)+c)*r_test*min((New_test+FU_test),Test)];
test2, FU_test + Test -> FU_con + Test,
[FU_test/((New_test+FU_test)+c)*r_test*min((New_test+FU_test),Test)];

fuCon, FU_con + Consultant_FU -> FU_blood + Consultant_FU,
[r_fuCon*min(FU_con,Consultant_FU)];
newCon, New_con + Consultant_New -> New_blood + Consultant_New,
[r_newCon*min(New_con,Consultant_New)];

xray, New_xray + Xray -> New_depart + Xray, [r_xray*min(New_xray,Xray)];

blood1, FU_blood + Blood -> FU_depart + Blood,
[FU_blood/((New_blood+FU_blood)+c)*r_blood*min((New_blood+FU_blood),Blood)];
blood2, New_blood + Blood -> New_xray + Blood,
[New_blood/((New_blood+FU_blood)+c)*r_blood*min((New_blood+FU_blood),Blood)];

depart1, FU_depart + Depart -> Stop_2 + Depart,
[FU_depart/((New_depart+FU_depart)+c)*r_depart*min((New_depart+FU_depart),Depart)];
depart2, New_depart + Depart -> Stop_1 + Depart,
[New_depart/(New_depart+FU_depart+c)*r_depart*min((New_depart+FU_depart),Depart)];

stop1, Stop_1 + Stop_3 -> Stop_1 + Stop_3,
[Stop_1/((Stop_1+Stop_2)+c)*r_stop*min((Stop_1+Stop_2),Stop_3)];
stop2, Stop_2 + Stop_3 -> Stop_2 + Stop_3,
[Stop_2/((Stop_1+Stop_2)+c)*r_stop*min((Stop_1+Stop_2),Stop_3)];
```

As introduced about CMDL model in Section 3.3.1, this CMDL has the same model syntax. PEPA model defines a series of states in the action flow. In the related

CMDL model, it defines the interaction and transition of states in terms of related actions. CMDL is usually used for chemistry reaction. Hence, it is unreadable for such kind of patient flow model. The transformation from PEPA to CMDL is carried out by the PEPA analyser automatically.

Each CMDL statement has three parts which are separated by a comma. Take the X-ray component in above model for example, the statements representing the X-ray scan process is:

```
xray,    New_xray + Xray -> New_depart + Xray,    [r_xray*min(New_xray,Xray)];
```

The first part specifies the action name. The middle part represents the state transition, which is from New_xray to New_depart. This means the new patients finish the X-ray scan and then depart from the hospital. The last part is the rate of the action which is equivalent to the actual rate in the PEPA model.

Before promoting an analysis on the CMDL, validation for both PEPA model and its corresponding CMDL model must be performed first.

5.3.2 Model initialization

To complete model validation, parameters used in the model must be initialized before execution. As mentioned before, this research adopts a lot of statistic data provided by the rheumatology department. These data consists of the average cycle time of related activities in the Rheumatology department, such as, registration, test, consultation, blood test, and the discharge. All the activities are measured to obtain the statistical figures. All these statistical figures are the cycle time of each activity in the workflow. The service rate used in the model can be calculated from these cycle time in terms of the formula $\mu_{service\ time} = \frac{1}{t_{cycle\ time}}$.

According to the statistical data, the service rate at each step of the workflow can be addressed. In all situations, the time unit is unified as hour. Thus, the rates in each step can be obtained as follows:

Rate of Registration	$\mu_{register} = 44.44$	Statistic cycle time of registration is 81 seconds; 44.44 patients arrive each hour.
Rate of Test	$\mu_{test} = 33.96$	Statistic cycle time of test 106 seconds; 33.96 patients complete tests each hour.
Rate of Consultant	$\mu_{conFU} = 4.0$ $\mu_{conNew} = 2.0$	Statistic cycle time of meeting consultant is 19 minutes on average, 30 minutes for the new and 15 minutes for the follow-up; each consultant can serve 2 .0 news and 4.0 FUs each hour.
Rate of Blood	$\mu_{blood} = 14.29$	Statistic cycle time of blood test is 252 seconds; 14.29 patients test blood each hour.
Rate of Depart	$\mu_{depart} = 14.81$	Statistic cycle time of departure is 243 seconds; 14.81 patients depart each hour.

Table 5.1 Statistical parameters used in the mode

In Table 5.1, the cycle time is obtained from the statistical measurements. Before starting this part, we have interviewed the staff of the Rheumatology department of Royal Hallamshire Hospital in Sheffield, United Kingdom, and collected above figures, a set of values of cycle time for each activity in the patient flow. These figures are gathered based on the daily statistics of the patient amount at the related sections of the Rheumatology department. Hence, this chapter investigates a real word work flow, and the analysis is carried out based on the genuine and realistic figures. Thus, the study will be valuable for the Rheumatology department to improve the efficiency and refine the current work flow.

The service time μ is calculated with the previous formula$\mu_{service\ time} = \frac{1}{t_{cycle\ time}}$. Take the rate of registration for example, as the cycle time is 81 seconds and the time unit is 1 hour (3600 seconds), the service rate of registration is

$$\mu_{register} = \frac{1}{t_{register}} = \frac{3600s}{81s} = 44.4444\ .$$

In addition to the statistic parameters, a set of hypothetic parameters are used in the model. It is assumed that the department runs at 100% capacity in the model, which means 100 new patients and 400 follow-up patients come to the department each week. If the department works 8 hours each day and from Monday to Friday each week, thus in each single working day, 20 new patients and 80 follow-up patients must be served. Thus, there are, on average, 2.5 new patients and 10 follow-up patients arriving in the department each hour. According to these hypothetical conditions, the following parameters can be assumed:

Arrival rate of New	$\lambda_{\mathrm{NEW}} = 2.5$
Arrival Rate of FU	$\lambda_{\mathrm{FU}} = 10.0$
Takt Time at Register	$\mathrm{Takt}_{\mathrm{register}} = 210$ seconds
Takt Time at Test	$\mathrm{Takt}_{\mathrm{test}} = 210$ seconds
Takt Time at Consultant	$\mathrm{Takt}_{\mathrm{conNEW}} = 24$ minutes $\mathrm{Takt}_{\mathrm{conFU}} = 6$ minutes
Takt Time at Blood	$\mathrm{Takt}_{\mathrm{blood}} = 288$ seconds
Takt Time at Depart	$\mathrm{Takt}_{\mathrm{depart}} = 210$ seconds

Table 5.2 Hypothetical parameters used in the model

In addition, this model also assumes the existence of an x-ray inspection which is an external test conducted in another department. Its rate is assumed to be 12.0 slots/hour. The last component "Stop" is specified with a large value 100.0, as its value does not matter to the analysis results.

In terms of the cycle time and takt time in Table 5.1 and 5.2, the number of each component type can be determined. Takt time is used to set the pace for the industrial manufacturing lines. The concept of takt time has main target to match the pace of production with customer demand. For example, the cycle time for consulting a new patient is 30 minutes; however, the takt time for consulting a new patient need 24 minutes. Thus, at least 2 consultants for new patients are required to insure that all new patients can be served on time. In the same way, the number of consultants serving the follow-up patients should be 3 at least. Hence, the number of each component used in the model is shown in the below Table 5.3.

$\mathbf{N}_{\mathbf{new}}$	$\mathbf{N}_{\mathbf{FU}}$	$\mathbf{N}_{\mathbf{register}}$	$\mathbf{N}_{\mathbf{test}}$	$\mathbf{N}_{\mathbf{conNew}}$
20	80	1	1	2
$\mathbf{N}_{\mathbf{conFU}}$	$\mathbf{N}_{\mathbf{blood}}$	$\mathbf{N}_{\mathbf{xray}}$	$\mathbf{N}_{\mathbf{depart}}$	$\mathbf{N}_{\mathbf{stop}}$
3	2	2	2	1

Table 5.3 Number of components used in the model (100% capacity)

5.3.3 Model validation

This section validates both the PEPA model and CMDL models with two different analyzing techniques: stochastic simulation and fluid flow approximation. Stochastic simulation is used here, because it supports an exact procedure for numerically simulating the time evolution of the system. The fluid flow approximation is the main analyzing method used in this thesis. The reasons why use the fluid flow approximation lies in its efficient solving process and accurate results based on the PEPA model. More details about these two analyzing techniques have been introduced in Section 2.2.

Both PEPA model and CMDL model are run on the Eclipse platform with PEPA plug-in. Stochastic simulation is conducted with Gillespies stochastic algorithm, in which time ranged from 0 to 10, with 1000 data points, a number of replications 10000, confidence interval equal to 0.05. Fluid flow approximation is carried out with adaptive step-size 5th-order Dormand Prince ODE solver, in which time ranged from 0 to 10, with 1000 data points, a step size of 1.0E-3, relative error and absolute error equal to 1.0E-4. All parameters used in validation were described in previous initialization section.

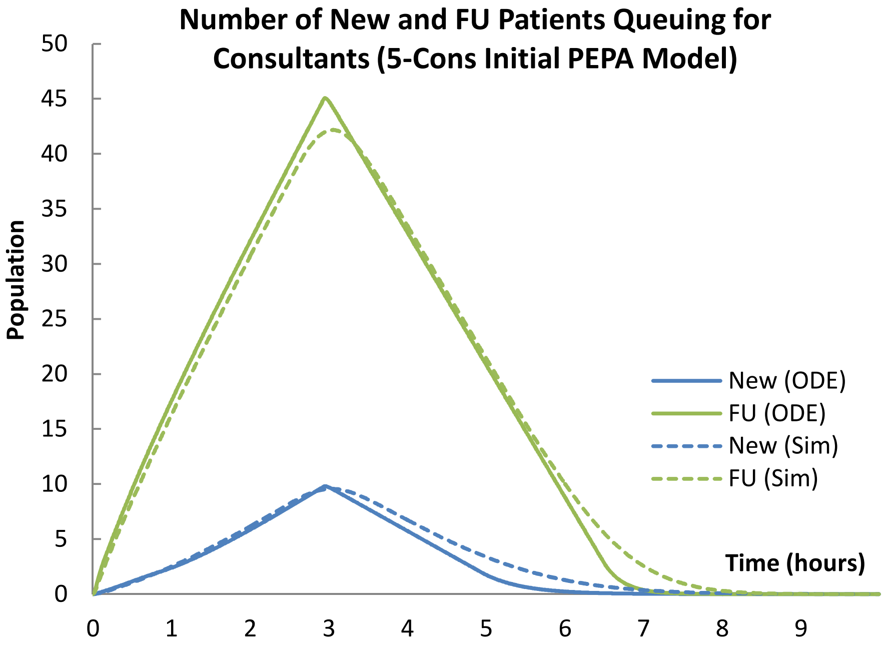


Figure 5.2 PEPA model: population of patients queuing for consultants

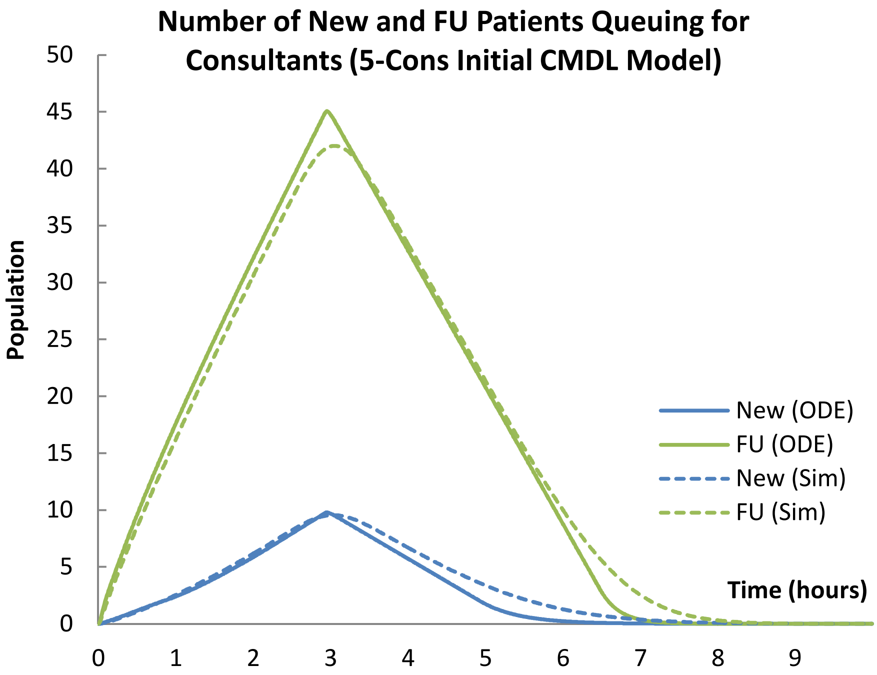


Figure 5.3 CMDL model: population of patients queuing for consultant

Figure 5.2 and Figure 5.3 depict the population of new patients and follow-up patients queuing for consultants in both models, and the solid lines represent the results from stochastic simulation and the dotted lines represent the results from fluid flow approximation. The upper lines describe the transient population of follow-up patients, while the lower lines display the transient population of new patients.

From both figures, it is really hard to distinguish between the two sets of results. This verifies that the transformed CMDL model is equivalent to the original PEPA model. Furthermore, as can be seen in figures, dotted lines always follow the solid lines with some slight difference at the peak and tail of lines. However, this difference lies in the acceptable error band. Hence, the initial PEPA model and its equivalent CMDL model have achieved its functionality and accomplished its requirements in system.

5.3.4 Model analysis

The analysis in this section aims to investigate the minimum number of consultants required for 100% capacity situation in the department, and the performance of both the static scheduling policy and dynamic scheduling policy applied in the model. The analysis is conducted on the initial validated CMDL model using fluid flow approximation. Parameters used here are those initialized in section 4.2.

The analysis will first investigate the queue for the consulting service when there are 5 consultants working in departments, 2 for new patients and 3 for follow-up patients. The first step is to run the model without using a scheduling policy for the patient flow. The results obtained for the scheduling free situation are shown in preceding Figure 5.3. In the Figure 5.3, the upper solid line represents the number of follow-up patients queuing for consultants and the lower solid line shows the queuing new patients. The upper line has a sharp increase and reaches the peak around 45 patients after 3 hours, and the lower line also grows to the top point about 10 patients after 3 hours. Thereafter, both queues come to decrease to zero within 8 hours. This figure reveals that the queue will be unacceptably long if no micro scheduling process is employed.

According to the results in the first step, it is necessary to apply a scheduling policy to address the queuing issue. In the second step, a static scheduling policy is created by adding a "factor" parameter in the model. This parameter is a constant used to manage the registration process via multiplying the registration rate by the "factor". The value of the factor lies in the interval $(0, 1)$, depending on other system parameters and the average queue length we expect. The factor can be calculated in terms of a formula:

$$Static\ Factor = \frac{\mu_{consultant} \times N_{consultant}}{\mu_{register}}$$

On the basis of the initialized parameters, the factor has a value roughly equaling 0.33 in this analysis step. Thus, the results from the model with the static scheduling policy are displayed in Figure 5.3.

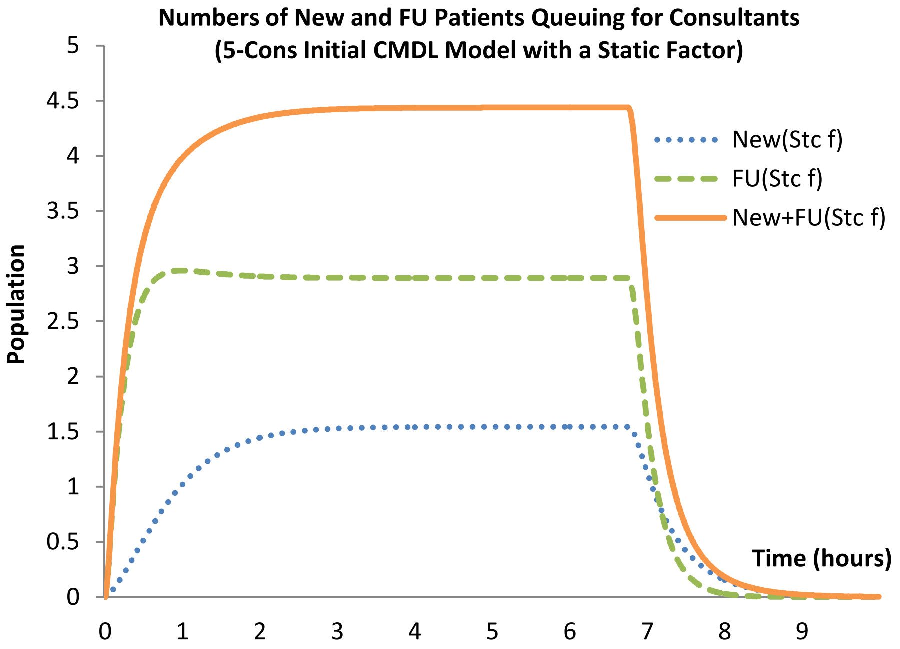


Figure 5.4 Population of patients queuing for consultants (×5) with static scheduling policy

Figure 5.4 depicts the number of patients waiting for consultants in the static scheduling process. In the Figure 5.4, a dotted line, representing the new patients, increases to a stable level around 1.5 patients on average in the queue after 2 hours. The dashed line, describing the follow-up patients, experiences a faster growth with 2 hours and reaches the stable situation around 4.5. The follow-up patients have faster increase in contrast to the new patients at the beginning two hours, because the number of follow-up patients is much greater than the new patients. This means that more follow-up patients arrive for consultation during the same period, so the queue of the follow-up patients must increase faster. The solid line is the total average population of both patient classes. These three lines all have the stable process, which means that the queues keep the length to a stable and lower level in the whole period. The total average queue length is about 4.5, not exceeding the number of consultants; this is because the patients are rightly scheduled at registration step via using a factor to accomplish it. This stable and short situation is really a good queuing status.

However, problems still exist, which are the slow increase of queue during the first two hours, especially the queue of the new patients. Before the length of queue reaches a stable level, the utilization of consulting process stays in a low level. Thus, it is necessary to cut down the increasing process especially the new patients during the first two hours so as to make the queue length reach the stable level soon. Thus, a dynamic scheduling policy is introduced in the next step.

In the third step, a dynamic scheduling policy is used to address the efficiency problem. The policy is to use a function to alter the registration process instead of the static factor. In other words, in the dynamic policy, a dynamic factor is applied to the model by multiplying the registration rate by the dynamic factor. The expression of the dynamic factor is:

$$Dynamic\ Factor = Max(Static\ Factor, 1 - aT)$$

This dynamic factor is a Max function involving the static factor and a formula $1 - aT$. T is the instant model run time, and a is a coefficient used to altering the value of the formula to achieve the best situation in modelling. With this function, a dynamic factor obtains a value greater than the static factor and close to 1 at the start of run, namely the value of the formula $1 - aT$. At the moment, patients are scheduled faster than the normal pace, namely the static scheduling. As time elapses, the value of the formula $1 - aT$ decreases until less than the static factor. Now the dynamic factor has its value equaling to the static factor. The results of the analysis using the dynamic scheduling policy are shown in Figure 5.5.

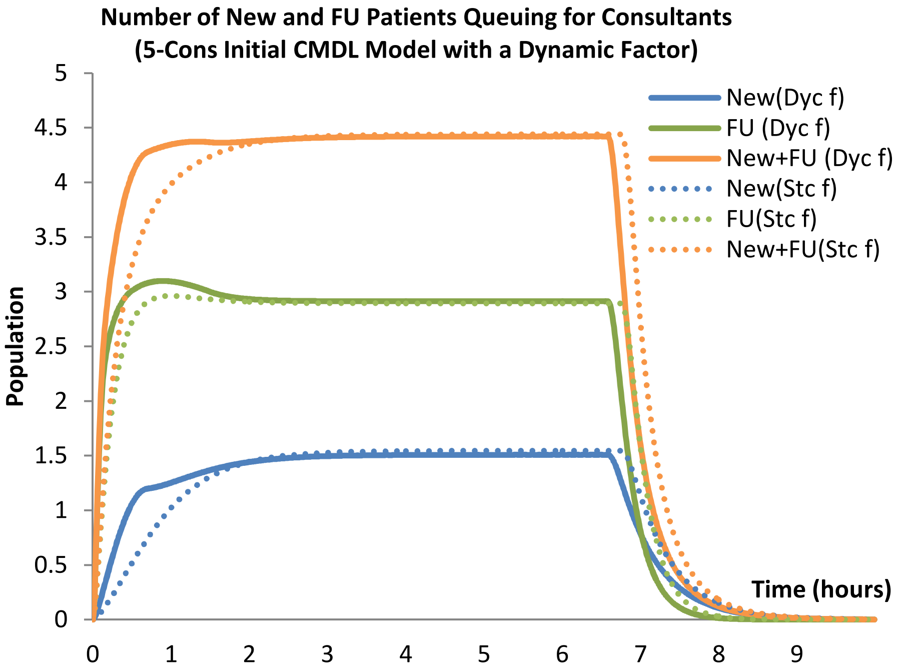


Figure 5.5 Population of patients queuing for consultants (×5) with dynamic scheduling policy

Figure 5.5 demonstrates the difference between dynamic scheduling policy and static scheduling policy. In the figure, solid lines exhibit the length of consulting queue with dynamic scheduling policy. The dotted lines, indicating for the length of queue using the static policy, are exactly the same as those lines in Figure 5.4. As shown in the figure, it is clear that solid lines have faster increase at the beginning in contrast to the dotted lines, and they also terminate earlier that the dotted lines. This means that patients are scheduled to the consultants in a shorter period by using a dynamic scheduling policy in contrast to the previous Figure 5.4. Thus, the utilization in the consulting process is improved during the beginning hours due to the fast scheduling.

The dynamic policy improves the efficiency of scheduling process. However, the tails of all lines, in Figure 5.5, continue over 8 hours. This means that 5 consultants cannot complete all patients in 8 hours even if they work with full efficiency. To reduce the whole consulting time span, the department needs more consultants to serve patients.

In the fourth step, it is assumed that one more consultant is added to serve the follow-up patients, because there are more follow-ups waiting for the service compared with the new. All other parameters, except factor, are the same as before.

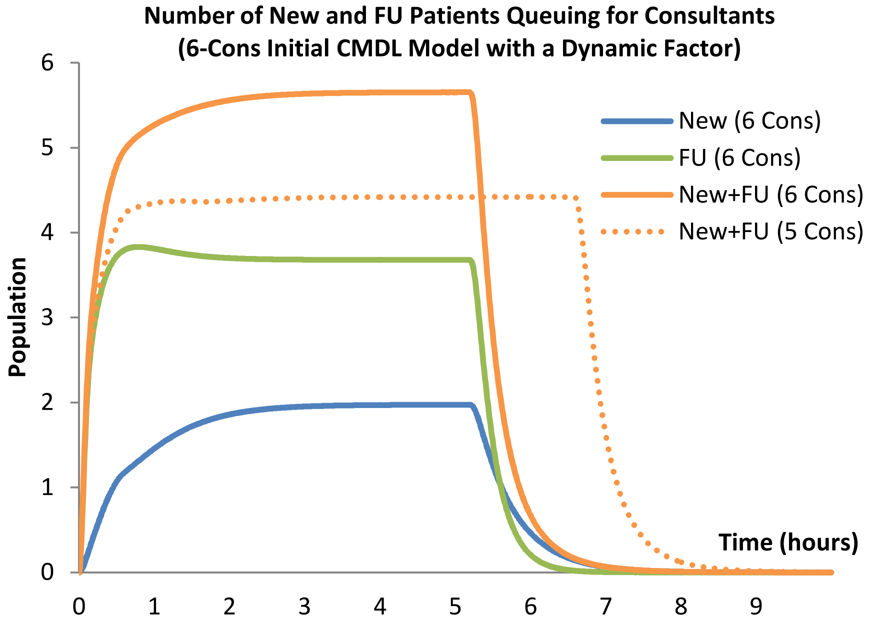


Figure 5.6 Population of patients queuing for consultants (×6) with dynamic scheduling policy

Figure 5.6 presents the queue length of patients for 6 consultants. Solid lines in the figure have similar characters with those in the previous figure, but the length of queue has some growth compared with Figure 5.5 because of the change in the dynamic factor. The queue length of each patient group keeps its value close to the number of consultants, which makes sure the number of queuing patients for each consultant is around 1. The key point of the step is the end of lines. In contrast to the dotted line representing the model of 5 consultants, all solid lines end in 8 hours; and this demonstrates that 6 consultants are enough to end the service of all patients in 8 hours.

So far, the capacity planning for the rheumatology department is complete. The minimum requirements of the department for consulting are 2 consultants for the new patients and 4 consultants for the follow-up patients. With the appropriate scheduling policy, the number of queuing patients in the department can be controlled around only 1 for each individual consultant.

5.4 Model update and analysis

This section plans to update the model with an alternative workflow in order to minimize the length of the waiting queue in the system. The updated model aims to refine the workflow by changing the order of activities in the flow and the allocation of consultants for different kinds of patients. Thereafter, the length of the waiting queue can be reduced and the efficiency of the workflow can be improved. The analysis is conducted by comparing the updated workflow model with the initial model via the fluid flow approximation.

5.4.1 Scenario description and model construction

In contrast to the initial scenario, the workflow of the new patient is changed to a new order, which is to bring forward the x-ray inspection and the blood test before the test, and the follow-up workflow remains the same. The consultants for the new patients also serve the follow-up patients while the consultants issued for the follow-up patients only serve the follow-up patients. This is because the new patients must have the blood test and the x-ray scan in another department, so bring forward these tests before seeing a consultant is able to save time.

Meanwhile, when the new patients are doing their x-ray and blood tests, the idle consultants for the new patients can be scheduled to serve the follow-up patients rather than just waiting there. Such updated workflow ideally improves the work efficiency. Thus, with the limited number of consultants, more patients can be served or the waiting queue can be reduced. This will be certified in the following model analysis. The detailed changes in the updated workflow are shown in Figure 5.7.

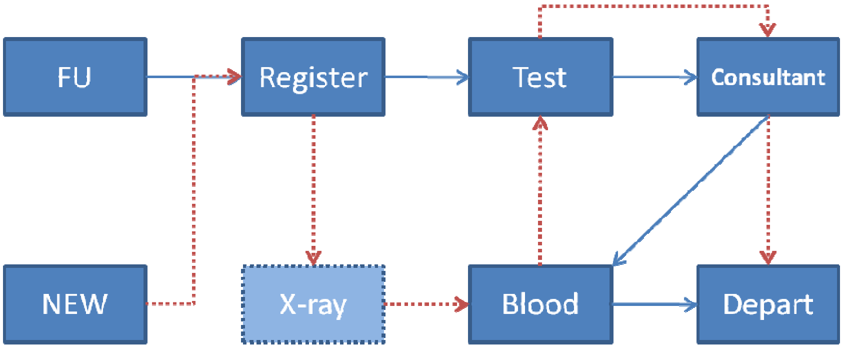


Figure 5.7 Updated workflow of rheumatology department

In Figure 5.7, the dotted line clearly depicts the flow of the new patients in the department. As the x-ray inspection must be handled in another department, it is more efficient for the new patients to complete the x-ray before entering the rheumatology department rather than leaving for x-ray halfway and coming back.

Furthermore, bringing forward the x-ray and the blood test for the new patients is to schedule more consultants from the new patients to the follow-up patients, when the new patients are leaving for the x-ray and doing the blood test. Thus, in the new model, consultants previously working for the new patients now must work for the both. When the new patients leave for x-ray or carry out the blood test, these consultants serve the follow-up patients; once the new come back, these consultants go to serve the new.

As per the model description, the initial PEPA model should be modified as follows:

$$
\begin{aligned}
\mathit{New} &\overset{def}{=} (\mathit{arriveNew},\ r_{\mathit{arriveNew}}).\mathit{New\_reg}\\
\mathit{New\_reg} &\overset{def}{=} (\mathit{register},\ r_{\mathit{register}}).\mathit{New\_xray}\\
\mathit{New\_xray} &\overset{def}{=} (\mathit{xray},\ r_{\mathit{xray}}).\mathit{New\_blood}\\
\mathit{New\_blood} &\overset{def}{=} (\mathit{blood},\ r_{\mathit{blood}}).\mathit{New\_test}\\
\mathit{New\_test} &\overset{def}{=} (\mathit{test},\ r_{\mathit{test}}).\mathit{New\_con}\\
\mathit{New\_con} &\overset{def}{=} (\mathit{newCon},\ r_{\mathit{newCon}}).\mathit{New\_depart}\\
\mathit{New\_depart} &\overset{def}{=} (\mathit{depart},\ r_{\mathit{depart}}).\mathit{Stop}
\end{aligned}
$$

$$
\begin{aligned}
\mathit{FU} &\overset{def}{=} (\mathit{arriveFU},\ r_{\mathit{arriveFU}}).\mathit{FU\_reg}\\
\mathit{FU\_reg} &\overset{def}{=} (\mathit{register},\ r_{\mathit{register}}).\mathit{FU\_test}\\
\mathit{FU\_test} &\overset{def}{=} (\mathit{test},\ r_{\mathit{test}}).\mathit{FU\_con}\\
\mathit{FU\_con} &\overset{def}{=} (\mathit{fuCon},\ r_{\mathit{fuCon}}).\mathit{FU\_blood}\\
&\quad + (\mathit{newCon\_fu},\ r_{\mathit{fuCon}}).\mathit{FU\_blood}\\
\mathit{FU\_blood} &\overset{def}{=} (\mathit{blood},\ r_{\mathit{blood}}).\mathit{FU\_depart};\\
\mathit{FU\_depart} &\overset{def}{=} (\mathit{depart},\ r_{\mathit{depart}}).\mathit{Stop}
\end{aligned}
$$

……

$$
\begin{aligned}
\mathit{Consultant\_New} &\overset{def}{=} (\mathit{newCon},\ r_1).\mathit{Consultant\_New}\\
&\quad + (\mathit{newCon\_fu},\ r_2).\mathit{Consultant\_New}\\
\mathit{Consultant\_FU} &\overset{def}{=} (\mathit{fuCon},\ r_{\mathit{fuCon}}).\mathit{Consultant\_FU}
\end{aligned}
$$

……

$$
\begin{gathered}
(\mathit{New}[m]||\mathit{FU}[n]) \bowtie_L\\
(\mathit{Register}[n_1]||\mathit{Consultant\_New}[n_3]||\mathit{Consultant\_FU}[n_4]||\mathit{Test}[n_2]||\mathit{Blood}[n_5]\\
||\mathit{Xray}[n_6]||\mathit{Depart}[n_7]||\mathit{Stop}[n_8])\\
L=\{\mathit{register},\ \mathit{test},\ \mathit{newCon},\ \mathit{newCon\_fu},\ \mathit{fuCon},\ \mathit{blood},\ \mathit{xray},\ \mathit{depart},\ \mathit{stop}\}
\end{gathered}
$$

The above segments are just the modified parts in the updated model in contrast to the initial model used in section 4.1. The workflow for the new patients is altered as that in the first block: registration, x-ray, blood, test, consultant and departure. The consultant for the new patients is defined with two actions which are consulting the new patients, namely *newCon*, and consulting the follow-up patients, namely *newCon_fu*. For the follow-up patients, the workflow remains the same; however, at the state *FU_con*, two actions appear there, standing for the process that the follow-up are served by two types of consultants.

It is worth mentioning that a function rate must be used when two or more actions are defined for a single component, such as, *Consultant_New*. Hence, the action rates for component *Consultant_New* is specified as below:

$$r_1 = \frac{New\_con}{New\_con + FU\ con} \times \mathrm{r}_{\mathrm{newCon}} \times \mathrm{Min}(\,New\_con + FU\ con, \qquad Consultant\_new\,)$$

$$r_2 = \frac{FU\ con}{New\_con + FU\ con} \times \mathrm{r}_{\mathrm{fuCon}} \times \mathrm{Min}(\,New\_con + FU\ con, \qquad Consultant\_new\,)$$

The formal proof of using this function rate can be found in [71]. As explained previously, the function rates cannot be processed with the PEPA Plug-in based analyzer, the following analysis will be conducted with its corresponding CMDL models.

5.4.2 Model validation and analysis

The main analysis techniques used in the section is the fluid flow approximation, so it will be validated with the stochastic simulation. Validation is carried out by running the CMDL model which is developed from the PEPA model. This conversion from PEPA to CMDL is completed with PEPA Plug-in automatically. Conditions of analyzing methods and parameters of the model used in the section remain the same as previous. However, the numbers of consultant instances are changed to: 4 consultants for the new and 2 consultants for the follow-up. This is because consultants for the new also serve the follow-up patients at the same time. Thus, the new patients need more consultants in contrast to the previous case.

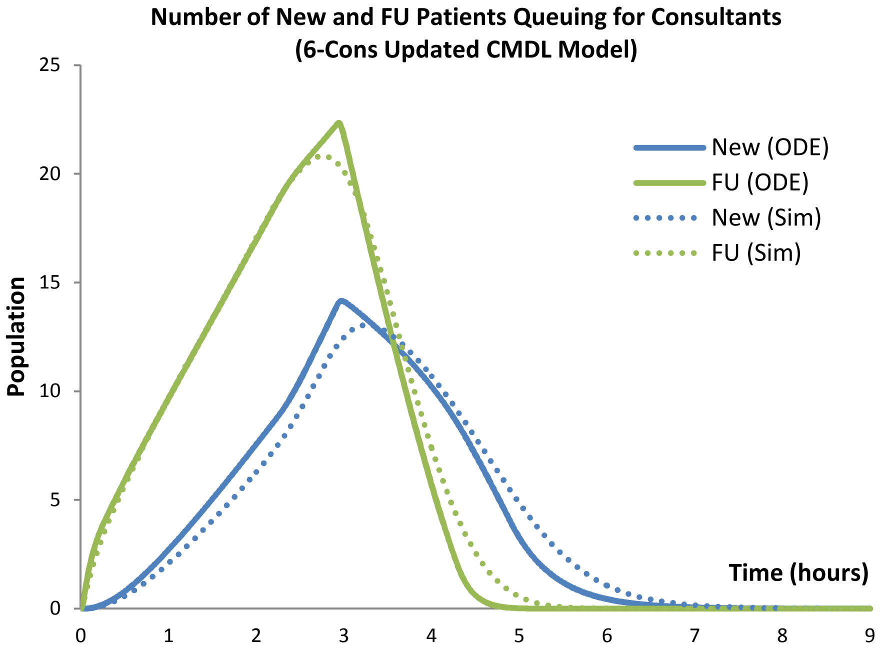


Figure 5.8 Population of patients queuing for consulting under the updated model

Figure 5.8 indicates the number of patients queuing for consulting service without using any scheduling policy. In the figure, solid lines are the results from the fluid flow approximation and the dotted lines are those from the stochastic simulation. It is clear that the dotted lines are always close to the solid lines, which means the fluid flow approximation generates similar results in the stochastic simulation. Thus, after validating the results of the fluid flow approximation, the next step will generate the analysis in the condition of using the previous introduced dynamic scheduling policy to the updated model, and then compare the results with those obtained from the initial model.

This step is to continue the analysis of the updated workflow model. In order to compare the updated model with the initial models, the conditions and parameters must be unified. The fluid flow approximation are applied to run the analysis with the same condition setting, and parameters used remain the same except for the number of consultants for each patient group, which is introduced in section 5.4.2. As the dynamic scheduling policy is the most efficient policy compared with the static policy, the analysis is directly conducted based on the case using the dynamic scheduling policy.

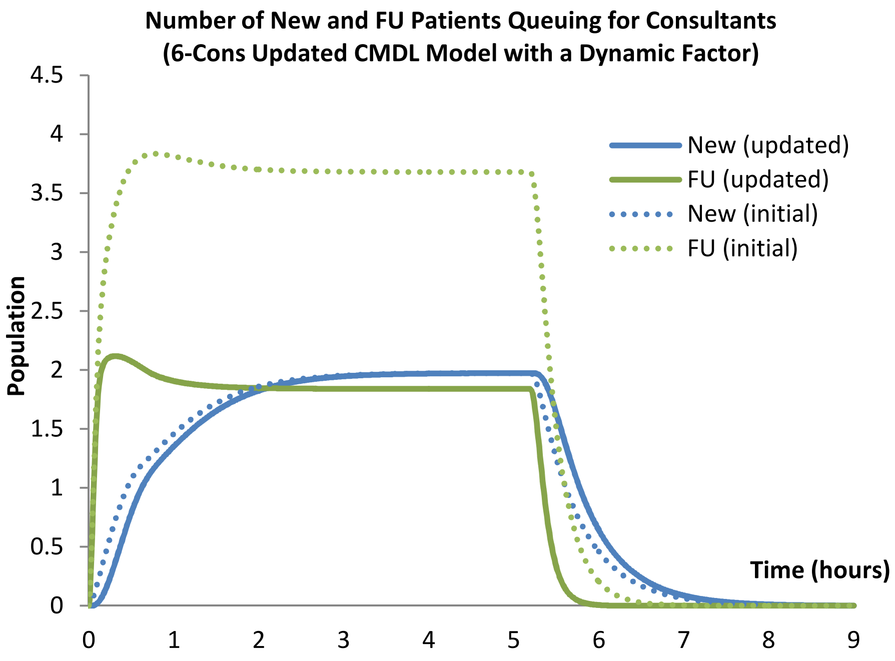


Figure 5.9 Population of patients queuing for consulting under the updated model with dynamic scheduling policy

Figure 5.9 displays the number of patients queuing for consultants of the updated model in contrast to the number in the initial model. In the figure, the solid lines represent the queue length in the updated model, and the dotted lines represent the queue length in the initial model. As can be seen from the figure, the lines, representing the new patients of the both models, are close to each other in the start and the end areas, and keep overlapped and stable in the middle. However, for the follow-up lines, the solid line from the updated model has its stable value around 2.0; the dotted line from the initial model locates above the solid line around 4.0, which is doubled compared with the solid line. This difference demonstrates that the updated model generates a shorter queue for follow-up patients than the initial model. The reason is to do with the changed workflow in the updated model. When the new patients leave for the x-ray and blood test, the consultants for the new go to serve the follow-up first until the new patients return.

Consequently, according to the model results, there is no doubt that the updated workflow is more efficient than the initial workflow, and it is able to minimize the queue length of the follow-up patients approaching the level of the new patients.

5.5 Modelling and analysis of an alternative grouping scheme

The section will investigate a model with an alternative grouping scheme based on the updated model, and analyze its performance by fluid flow approximation. This model is explored to approach the real situation of the workflow in the rheumatology department. Actually, people cannot work like machines, so consultants also cannot serve patients for full working hours without break. However, the previous study has the assumption that consultants work without any break until completing all patients. However, in the rheumatology department, consultants cannot work continuously. It is assumed that consultants usually have two or three short breaks during a working day. Thus, if there are three breaks in a whole working day, the whole day can be separated into four working time slots. Thus, all patients should be grouped in terms of the number of slots, and then scheduled to the department; each group of patients is served in a single time slot, and the following group always starts at the end of previous group. Between any two time slots, consultants have a break before which all patients in the group should be completed. This section aims to apply a discontinuous grouped patient flow to the updated workflow model.

5.5.1 Scenario description and model construction

According to the previous specification, this model will represent the workflow with three breaks for consultants, and patients will be scheduled in four groups. To achieve this, all patients, including the new and the follow-up patients, must be split into four equivalent groups. In the model, each group is registered to enter the department after an equal time period. For example in this case, the time period is assumed to be 2 hours. Thus, Group 1 starts entering the department from time point zero; Group 2 starts from time 2; similarly, Group 3 and Group 4 starts from time 4 and time 6 respectively. Once the patient group starts, they will be scheduled under the static policy first, and then follow the updated workflow until departure.

To achieve this scenario, first of all, the updated model type must be changed to the type with four sub-groups, which is described as follows:

$$\begin{aligned}
New_1 &\overset{def}{=} (arriveNew_1,\ r_{arriveNew1}).New_1\_reg \\
New_1\_reg &\overset{def}{=} (register,\ r_{register}).New_1\_xray \\
New_1\_xray &\overset{def}{=} (xray,\ r_{xray}).New_1\_blood \\
New_1\_blood &\overset{def}{=} (blood,\ r_{blood}).New_1\_test \\
New_1\_test &\overset{def}{=} (test,\ r_{test}).New_1\_con \\
New_1\_con &\overset{def}{=} (newCon,\ r_{newCon}).New_1\_depart \\
New_1\_depart &\overset{def}{=} (depart,\ r_{depart}).Stop \\
&\dots\dots
\end{aligned}$$

$$\begin{aligned}
FU_1\_CON &\overset{def}{=} (arriveFU1,\ r_{arriveFU1}).FU_1\_CON\_reg \\
FU_1\_CON\_reg &\overset{def}{=} (register,\ r_{register}).FU_1\_CON\_test \\
FU_1\_CON\_test &\overset{def}{=} (test,\ r_{test}).FU_1\_CON\_con \\
FU_1\_CON\_con &\overset{def}{=} (fuCon,\ r_{fuCon}).FU_1\_CON\_blood \\
&\quad + (newCon\_fu,\ r_{fuCon}).FU_1\_CON\_blood \\
FU_1\_CON\_blood &\overset{def}{=} (blood,\ r_{blood}).FU_1\_CON\_depart \\
FU_1\_CON\_depart &\overset{def}{=} (depart,\ r_{depart}).Stop \\
&\dots\dots
\end{aligned}$$

$$\begin{gathered}
(New_1[m_1] || New_2[m_2] || New_3[m_3] || New_4[m_4] || \\
FU_1[n_1] || FU_2[n_2] || FU_3[n_3] || FU_4[n_4]) \bowtie_L \\
(Register[n_1] || Consultant\_New[n_3] || Consultant\_FU[n_4] || \\
Test[n_2] || Blood[n_5] || Xray[n_6] || Depart[n_7] || Stop[n_8]) \\
L = \{register,\ test,\ newCon,\ newCon\_fu,\ fuCon,\ blood,\ xray,\ depart,\ stop\}
\end{gathered}$$

The above model is created to model the process of Group 1 involving the new patients and the follow-up patients respectively. From the model details, the workflow is completely the same as the previous updated model. For the other groups, they all have the same process.

Although the above model represents four groups of patients, the key point is to make each group start in different time slots. Therefore, a function rate must be used for the arrival rate of each group, which is expressed as follows:

$$\lambda = \begin{cases} C, & t > T \\ 0, & t \leq T \end{cases}$$

In above formula, λ is the arrival rate, t is the transient time; C is a constant value; and T is the specified timeout. This formula means the arrival rate remains as zero until time point T; and after T, the rate is C. The function accomplishes the process that each patient group starts from different time. If patients cannot be completed in this time slot, they will be moved to the next time slot

5.5.2 Model analysis

In the analysis, it is assumed that the time span to start each sub-group is 2 hours. Based on the previous function, the arrival rates of each sub-group can be found in Table 5.4. The other parameters remain the same as those in previous sections.

$\boldsymbol{\lambda_{New1}} = \begin{cases} \mathbf{2.5}, & t > 0 \\ \mathbf{0}, & t \leq 0 \end{cases}$	$\boldsymbol{\lambda_{New2}} = \begin{cases} \mathbf{2.5}, & t > 2 \\ \mathbf{0}, & t \leq 2 \end{cases}$	$\boldsymbol{\lambda_{New3}} = \begin{cases} \mathbf{2.5}, & t > 4 \\ \mathbf{0}, & t \leq 4 \end{cases}$	$\boldsymbol{\lambda_{New4}} = \begin{cases} \mathbf{2.5}, & t > 6 \\ \mathbf{0}, & t \leq 6 \end{cases}$
$\boldsymbol{\lambda_{FU1}} = \begin{cases} \mathbf{10}, & t > 0 \\ \mathbf{0}, & t \leq 0 \end{cases}$	$\lambda_{FU2} = \begin{cases} 10, & t > 2 \\ 0, & t \leq 2 \end{cases}$	$\lambda_{FU2} = \begin{cases} 10, & t > 4 \\ 0, & t \leq 4 \end{cases}$	$\lambda_{FU4} = \begin{cases} 10, & t > 6 \\ 0, & t \leq 6 \end{cases}$

Table 5.4 Functional arrival rates of each patient sub-group

As the PEPA Plug-in does not support the analysis of function as rate and CMDL cannot model a piecewise function, the fluid flow approximation will be processed with MatLab in the following analysis. Here, ODE45 in MatLab is adopted to solve ODEs; time span is from 0 to 11; step size is 0.01.

Figure 5.10 clearly shows the queue length for consulting of each patient group. The dotted lines represent the population of the new patients, and the dashed lines stand for the population of the follow-up patients. From the figure, it is easy to distinguish four groups of patients arriving in waves. The subsequent groups always start at the end of the previous group. Owing to a static factor and the updated workflow are used in the model, the queue length of the new patients is just around 3.5 at the peak, and the queue of the follow-up is about 2.2 for the peak value. The population waiting for consultants fluctuates like waves. The wave peak means that more patients are waiting for service, which means that at this point consultants are busy. However, at the end of each sub-group, there are fewer patients queuing for service, thus, the consultants have some free time for a break.

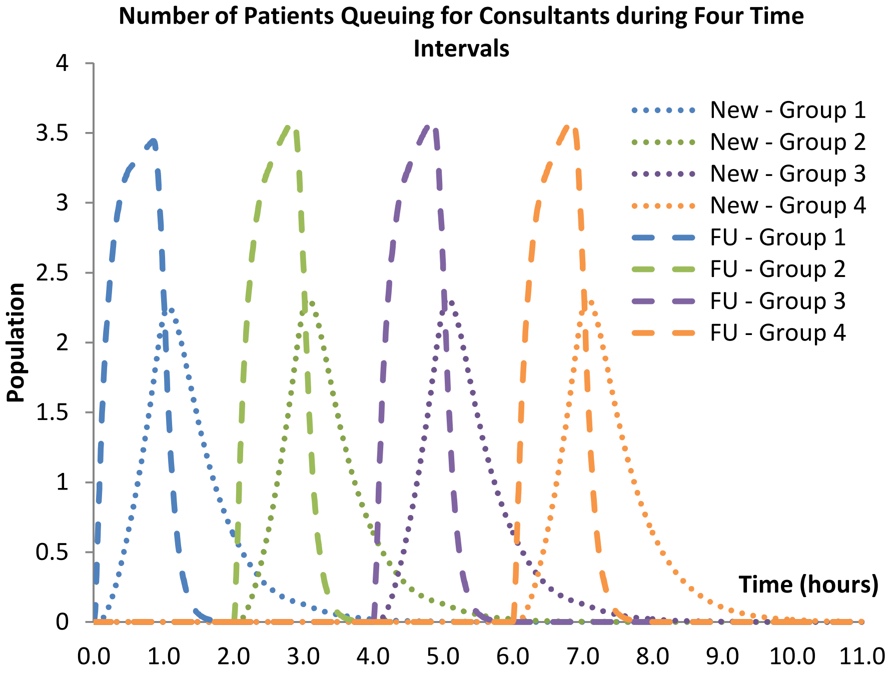


Figure 5.10 Population of patients queuing for consultants with the grouping scheme

According to the time span of each wave, each follow-up patient group can be completed within 2 hours, and the total follow-up can be done within 8 hours. However, the new patient group cannot finish their consultations with 8 hours, because each follow-up sub-group uses more than 3 hours to complete.

Compared with follow-up patients, the population of the new patients has a slower increase after the start in each sub-group. This is because of the workflow, in which the new patients need complete x-ray and blood first before seeing the consultants. This is the first reason for the new patients' late completion.

The second reason is the number of consultants serving the new. In this case, there are 4 consultants serving both the new and the follow-up. However, there is no consultant serving the new only. Thus, to solve this problem, more consultants can be added to serve the new patients only.

The third possible reason is the time span set before starting each group. Here, 2 hours is used as a slot. To reduce the total time span, each time span for an individual sub-group can be set less than 2, for example, 1.5 hours.

In this section, the workflow remains static. Hence, to cut down the total time used by the new patients, two possible approaches can be attempted based on the first and the second reasons. Firstly, the model is changed by adding one more consultant which is used to serve the new patients only. Now there are three kinds of consultants in the model. It is assumed that there is 1 consultant serving the new only, 2 consultants serving the follow-up only, and 3 consultants serving both. All other parameters and conditions remain the same. The results in terms of the new model are displayed in the following Figure 5.11.

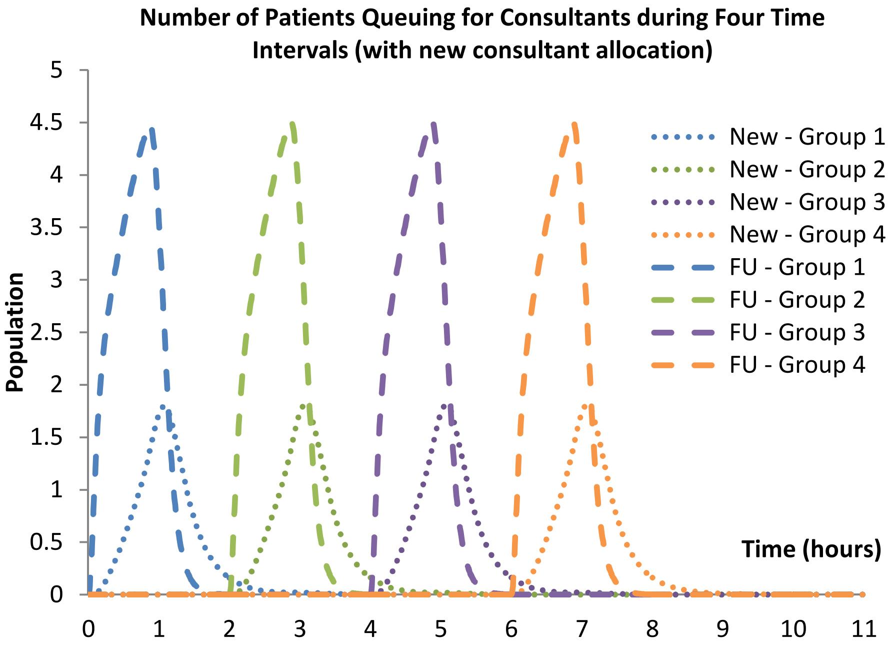


Figure 5.11 Population of patients queuing for consultants in the grouping scheme with updated consultant allocation

In contrast to the Figure 5.10, the time span of completing all the new, displayed in Figure 5.11, reduces from nearly 10 hours to about 8.5 hours. Although it is still over 8 hours, the new consultant allocation plan really generates active effects for time compression. Furthermore, the queue length for either the new or the follow-up is altered because of the change of consultant allocation.

Finally, as suggested in the third reason, it is necessary to cut the individual time span from 2.0 hours to 1.5 hours. Then, the following figure is obtained.

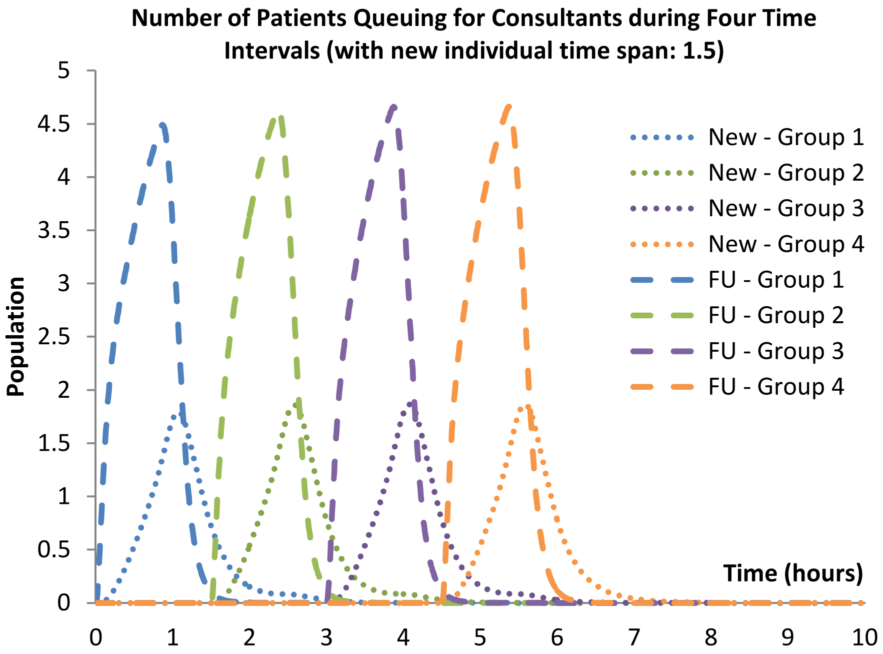


Figure 5.12 Population of patients queuing for consultants in the grouping scheme with updated consultant allocation and individual time span

After altering the time span to a lower value, Figure 5.12 obviously shows that the consultants can complete all new patients within 8 hours, and the queue length is quite close to that in Figure 5.11.

In conclusion, the grouping scheme is a good scheduling policy for the rheumatology department. It can combine with the static policy to minimize the queue, and create several breaks for consultants between any two sub-groups. Hence, the grouping scheme is an effective scheduling policy facing the realistic demands.

5.6 Conclusions

This chapter aims to explore the capacity planning based on the workflow model of the rheumatology department, and also to investigate the scheduling schemes that could be used for patient flow. As previously, the main modelling techniques used in the work are PEPA and CMDL. PEPA is used to create the original model, and CMDL is generated from the initial PEPA model and applied to handle the analysis using functional rates. The main analysis methods are fluid flow approximation and stochastic simulation.

In this research, real statistical data is used in the model so as to obtain an efficient capacity scheme for consultants. At the same time, both the static scheduling policy and the dynamic scheduling policy are adopted in the model to minimize the waiting queue by comparing their performance. The findings show that the dynamic policy is more efficient than the static, but it is more complex to be run without a smart system in the real department. The smart system is able to undertake patient scheduling and consultant allocation process by a designed scheduling program. Finally, the paper explores an alternative scheduling scheme, namely grouping scheme. According to the analysis results, the grouping scheme has an excellent performance combining with the static scheduling policy. Another advantage of the grouping scheme is that this scheme is easy to apply in the real department, when a smart system is not available.

Overall, this chapter successfully demonstrated the potential of the model of the rheumatology department by investigating the capacity and the scheduling based on it.

Chapter 6 Simulation of capacity and scheduling for a healthcare department

6.1 Introduction

This chapter aims to investigate the model of the healthcare department via the discrete event simulation, which is researched in the preceding chapter through PEPA based model. The main target of the chapter is to validate the previous PEPA model with a different modelling technique.

In Chapter 5, capacity planning work is performed via a formal modelling language that is the Performance Evaluation Process Algebra (PEPA), and then the model is analyzed in terms of the ODEs based fluid flow analysis. In this chapter, the main purpose is to apply a discrete event simulation to the same capacity model and generate the analysis from the discrete event simulation so as to compare the results with the original PEPA. The discrete event simulation techniques are used to create an equivalent model in order to validate the measurement results from PEPA model, as discrete event simulation is able to generate the analysis in a discrete situation rather than the fluid flow. The simulation tool used in this research is OMNeT++ IDE that will be detailed later.

Section 5.2 describes the model scenario based on PEPA. Section 5.3 demonstrates a powerful simulation tool OMNeT++. Section 5.4 illustrates the design scheme through OMNeT++. Section 5.5 summarizes the measurement results and completes a validation with the PEPA model. Section 5.6 concludes the analysis briefly.

6.2 Model scenario

In Chapter 5, the model scenario depends on the workflow of rheumatology department in the Royal Hallamshire hospital. This research uses the same scenario as that in Chapter 5. Rheumatology department runs the service for a number of patients with a fixed procedure. The work flow in rheumatology department is depicted in Figure 6.1.

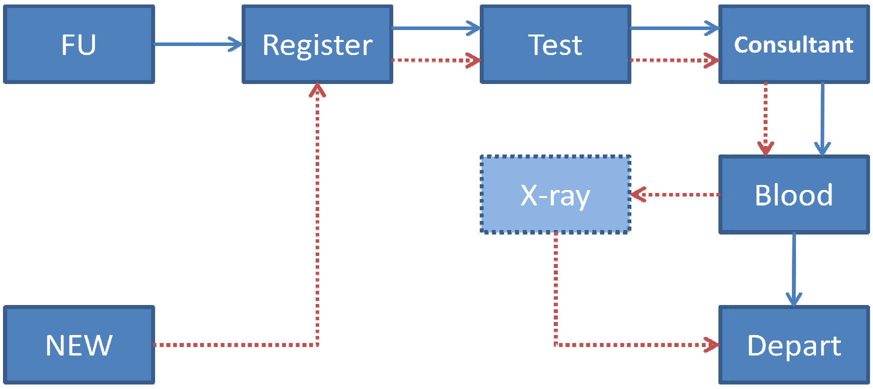


Figure 6. 1 Rheumatology department workflow

As displayed in Figure 6.1, patients can be generally grouped in two classes in terms of the research objects, namely follow-up patients (FU) and new patients (NEW). These two patient flows are indicated in Figure 1 with a solid line for the follow-up and a dotted line for the new. Each rectangle represents a step point of the whole process. The both groups of patients must register for an appointment (Register) with the department first, then each patient has a general check up (Test) before meeting the consultants (Consultant). After seeing the consultants, follow-up patients go for a blood test (Blood) and leave the department (Depart). However, new patients must go to another department for an X-ray inspection (X-ray) after the blood test, and then leave the department.

6.3 Simulation tool

The main technique used in this work refers to a useful simulation tool named OMNeT++ to generate the queue network depending on the previous scenario.

According to the definition of OMNeT++ in the OMNeT++ community, “OMNeT++ is an extensible, modular, component-based C++ simulation library and framework, primarily for building network simulators.” OMNeT++ is developed to solve different kinds of network models, such as wired and wireless communication networks, queueing networks, and distributed networks, and so on. To facilitate simulation, OMNeT++ offers an Eclipse-based IDE, a graphical runtime environment, and a host of other tools. Therefore, the simulation is easy to generate and execute. OMNeT++ IDE extends Eclipse platform with some new functional modules: an editor with both graphical view and command lines (an NED file), a functional module for parameter setup and simulation configuration (an INI file) and an analysis module for the simulation output (an ANF file). More details of OMNeT++ and OMNeT++ IDE can be found in the documentation of OMNet++ which is available on the OMNeT++ community site.

In this chapter, OMNeT++ is used due to its powerful functionality in simulating a queuing net work.

6.4 Model re-construction based on PEPA model

In this section, the capacity model will be rebuilt in the OMNeT++ IDE in terms of the PEPA based model. In previous chapter, this capacity model has been completed and analyzed based on PEPA. Here, the capacity simulation model must be equivalent to the previous PEPA model in order to verify the results obtained from the fluid flow analysis in PEPA model.

6.4.1 PEPA based capacity model

According to the model scenario, this capacity model is defined with PEPA in previous chapter. The model definition in PEPA is:

$$New \stackrel{def}{=} (arriveNew,\ r_{arriveNew}).New\_reg$$

$$New\_reg \stackrel{def}{=} (register,\ r_{register}).New\_test$$

$$New\_test \stackrel{def}{=} (test,\ r_{test}).New\_con$$

$$\begin{aligned}
New\_con &\stackrel{def}{=} (newCon, r_{newCon}).New\_blood \\
New\_blood &\stackrel{def}{=} (blood, r_{blood}).New\_xray \\
New\_xray &\stackrel{def}{=} (xray, r_{xray}).New\_depart \\
New\_depart &\stackrel{def}{=} (depart, r_{depart}).Stop
\end{aligned}$$

$$\begin{aligned}
FU &\stackrel{def}{=} (arriveFU, r_{arriveFU}).FU\_reg \\
FU\_reg &\stackrel{def}{=} (register, r_{register}).FU\_test \\
FU\_test &\stackrel{def}{=} (test, r_{test}).FU\_con \\
FU\_con &\stackrel{def}{=} (fuCon, r_{fuCon}).FU\_blood \\
FU\_blood &\stackrel{def}{=} (blood, r_{blood}).FU\_depart \\
FU\_depart &\stackrel{def}{=} (depart, r_{depart}).Stop
\end{aligned}$$

$$\begin{aligned}
Register &\stackrel{def}{=} (register, r_{register}).Register \\
Test &\stackrel{def}{=} (test, r_{test}).Test \\
Consultant\_New &\stackrel{def}{=} (newCon, r_{newCon}).Consultant\_New \\
Consultant\_FU &\stackrel{def}{=} (fuCon, r_{fuCon}).Consultant\_FU
\end{aligned}$$

$$\begin{aligned}
Blood &\stackrel{def}{=} (blood, r_{blood}).Blood \\
Xray &\stackrel{def}{=} (xray, r_{xray}).Xray \\
Depart &\stackrel{def}{=} (depart, r_{depart}).Depart \\
Stop &\stackrel{def}{=} (stop, r_{stop}).Stop
\end{aligned}$$

$$\begin{gathered}
(New[m]||FU[n]) \\
\bowtie_L \\
(Register[n_1]||Test[n_2]||Consultant\_New[n_3]||Consultant\_FU[n_4]||Blood[n_5] \\
||Xray[n_6]||Depart[n_7]||Stop[n_8]) \\
L=\{register, test, newCon, fuCon, blood, xray, depart, stop\}
\end{gathered}$$

6.4.2 Capacity model re-construction in OMNeT++ IDE

In order to transfer the PEPA based capacity model to a queue network in OMNeT++ IDE, the number of server components must be specified first. In preceding chapter, all parameter values are obtained from the rheumatology department of the Royal Hallamshire Hospital in Sheffield. All parameters used in the simulation remain the same as these used in the PEPA model.

According to the statistic data, the service rate at each step of the workflow can be addressed. In all situations, the time unit is unified as hour. Thus, the rates in each step can be obtained as follows:

Rate of Registration	$\mu_{register} = 44.44$	Statistic cycle time of registration is 81 seconds; 44.44 patients arrive each hour.
Rate of Test	$\mu_{test} = 33.96$	Statistic cycle time of test 106 seconds; 33.96 patients complete tests each hour.
Rate of Consultant	$\mu_{conFU} = 4.0$ $\mu_{conNew} = 2.0$	Statistic cycle time of meeting consultant is 19 minutes on average, 30 minutes for the new and 15 minutes for the follow-up; each consultant can serve 2 .0 news and 4.0 FUs each hour.
Rate of Blood	$\mu_{blood} = 14.29$	Statistic cycle time of blood test is 252 seconds; 14.29 patients test blood each hour.
Rate of Depart	$\mu_{depart} = 14.81$	Statistic cycle time of departure is 243 seconds; 14.81 patients depart each hour.

Table 6.1 Statistic parameters used in the model

In addition to the statistic parameters, a set of hypothetic parameters are used in the model. It is assumed that the department runs at 100% capacity in the model, which means 100 new patients and 400 follow-up patients come to the department each week. If the department works 8 hours each day and from Monday to Friday each week, thus in each single working day, 20 new patients and 80 follow-up patients must be served. Thus, there are, on average, 2.5 new patients and 10 follow-up patients arriving in the department each hour. According to these hypothetic conditions, the following parameters can be taken:

Arrival rate of New	$\lambda_{NEW} = 2.5$
Arrival Rate of FU	$\lambda_{FU} = 10.0$
Takt Time at Register	$Takt_{register} = 210\ seconds$
Takt Time at Test	$Takt_{test} = 210\ seconds$
Takt Time at Consultant	$Takt_{conNEW} = 24\ minutes$ $Takt_{conFU} = 6\ minutes$
Takt Time at Blood	$Takt_{blood} = 288\ seconds$
Takt Time at Depart	$Takt_{depart} = 210\ seconds$

Table 6.2 Hypothetic parameters used in the model

In addition, this model has a component x-ray inspection which is an external test conducted in another department. Its rate is used with the value 12.0 slots/hour. The last component "Stop" is specified with a large value 100.0, as its value does not matter the analysis results.

In terms of the cycle time and takt time in Table 6.1 and 6.2, the instance number of each component can be determined. For example, the cycle time for consulting a new patient is 30 minutes; however, the takt time for consulting a new patient need 24 minutes. Thus, at least 2 consultants for new patients are required to insure that all new patients can be severed on time. In the same way, the number of consultants serving the follow-up patients should be 3 at least. Hence, the number of each component used in the model is shown in the below table.

N_{new}	N_{FU}	$N_{register}$	N_{test}	N_{conNew}
20	80	1	1	2
N_{conFU}	N_{blood}	N_{xray}	N_{depart}	N_{stop}
3	2	2	2	1

Table 6.3 Number of components used in the model (100% capacity)

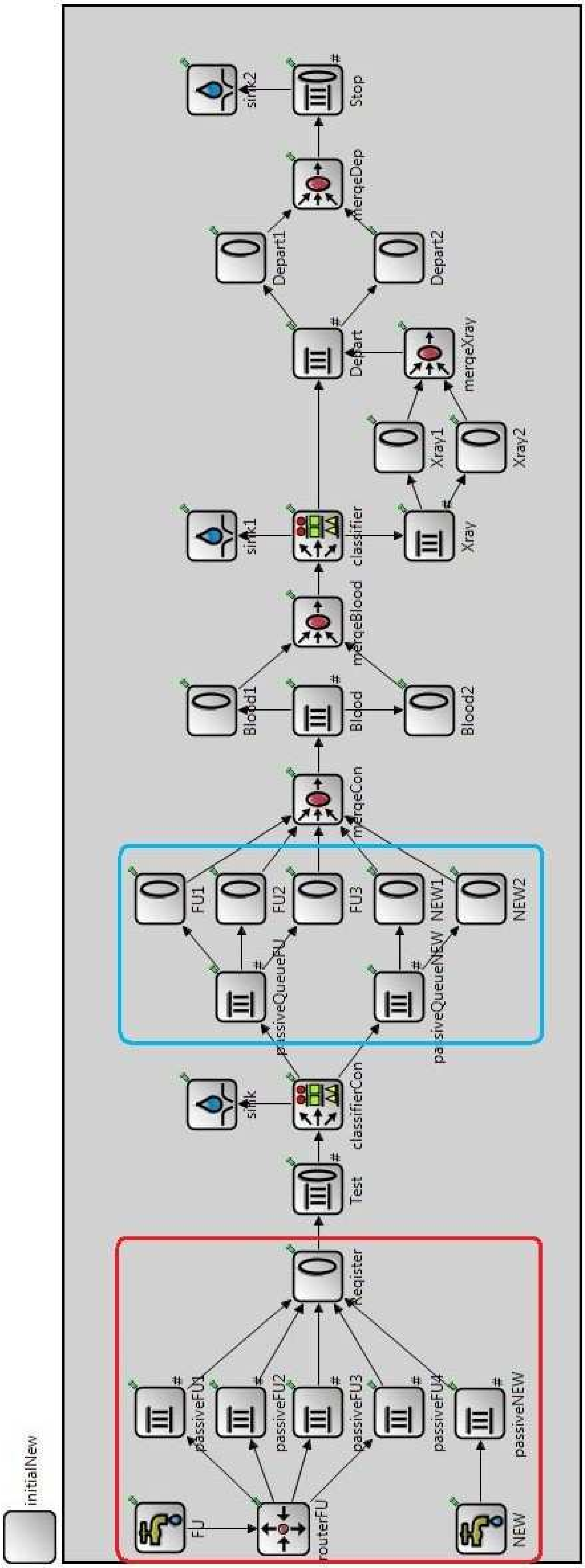


Figure 6. 2 Capacity model scheme in OMNeT++ IDE

According to the above parameters, the simulation based queue network should be a multi-server network, and the number of servers equals the number of components used in the PEPA model. Therefore, a simulation based model is created, which is completely equivalent to the PEPA based model. Figure 6.2 displays the design scheme of the capacity in the OMNeT++ IDE.

In the Figure 6.2, the area in the red line circle is designed to create two classes of patients that are the new arrivals and the follow up arrivals in terms of the PEPA model. The logo NEW is the source of the new arrivals which can generates a number of events representing patients and forward them to a passive queue with a fixed arrival rate. Similarly, the logo FU is the source of the follow up patients, but the different point is that all generated FU events go a router first; thereafter, the router dispatch each event to one of the connected passive queues at random, which means that the four passive queues have equivalent probability to obtain the next FU event. Then all passive queues for both NEW and FU connect to a server representing a register. The register server is able to obtain event from those five connected passive queues in terms of the random algorithm. Thereby, when the register server obtains the next event, it has 80% probability to get a FU event and just 20% probability to get a NEW event. This because there are four FU queues and just one NEW queue sending events to the register server, and the register server always receives events at random. This design for the initial sourcing process is to model the sourcing process in PEPA model. According to the arrival rates in the PEPA model, the rate of the incoming follow up patients is four times faster than the new patients.

In the next step, all patient events leave register server and go a single queue server that is the TEST which has been noted in the Figure 6.1. Thereafter, these events, both FU and NEW, come to a classifier in which two types of patients are classified and dispatched to two different connections, and one connection is to the consultants of the follow up patients and the other one is the to the consultants for the new patients as displayed in the blue circled area. In this area, there are three servers used as consultants for the follow up patients and two servers standing for the consultants of the new patients. All the FU servers are connected with a passive queue of the FU patients; meanwhile, all the NEW servers are connected with a passive queue of the NEW. Both passive queues apply the random algorithm to dispatch the event to one of the connected servers. This means the events in each queue have the equivalent probability to obtain the service. The number of servers here depends on the parameters used in the PEPA model.

After leaving the consultant servers, all events are merged and sent to a passive queue for the blood test. Here, there are two servers used for the blood test. The passive queue for the blood test still sends events using a random algorithm.

Subsequently, a classifier is employed to dispatch the follow up patients to depart; however, the new patients are dispatched to take the x-ray test before being sent to depart. After the Depart server, all patient events reach the Stop server and terminate there. Up to now the workflow completes, and finally all events are collected in the Sink.

In the OMNeT++ based capacity model, each part is designed equivalently to the original PEPA model including all parameters.

6.5 Measurement and validation

In this section, a serious of results is obtained from the simulation in order to validate the previous measurements from PEPA model. As introduced in preceding sections, the model in the OMNeT++ IDE has equivalent action flow and rates compared with the PEPA model.

6.5.1 Measurement with a low number of patient events

Following the PEPA model, the results based on the average queue length have been obtained in previous chapter. To facilitate the validation, only the queue length values at integer time points are picked up for the comparison. Similarly, for the results from the simulation, just the value at an integer time point is used to validate the PEPA based results. In the simulation, the value at each time point is the mean of a number of replications generated from the simulation run in OMNeT++ IDE, which guarantees the accuracy of the simulation results. Hence, the comparison between the fluid flow analysis and the discrete event simulation is displayed in Figure 6.3.

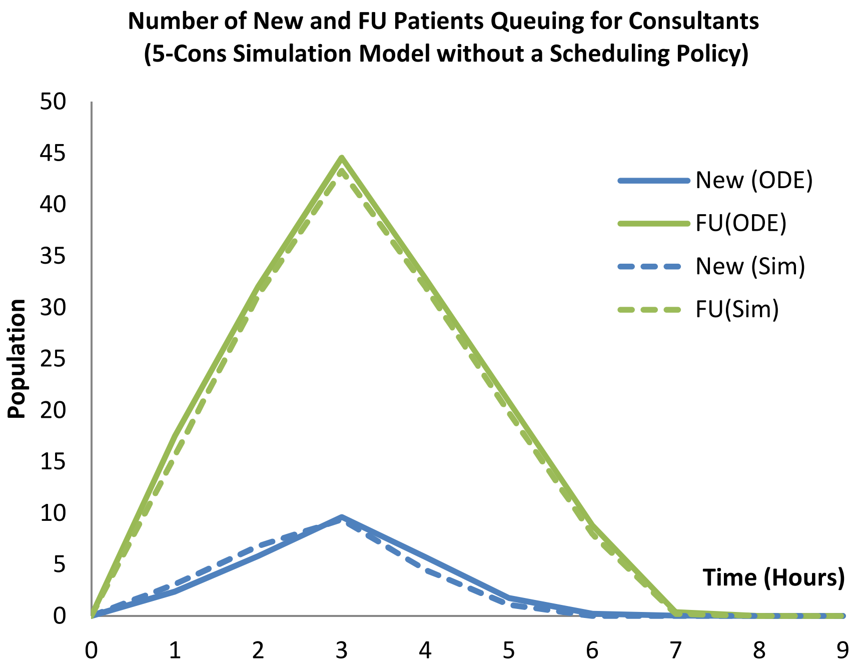


Figure 6. 3 PEPA vs. simulation - population of patients queuing for consultants

From Figure 6.3, it is very clear that the dashed for the simulation measurements plot close to the real line representing the ODEs. Thus, a preliminary conclusion is obtained that is the preceding fluid flow analysis from PEPA model has the high accuracy when the patient flow model has no scheduling policy used in it.

However, when a scheduling policy, such as a constant factor used in the registration rate to restrain the overall arrival rates in the following model parts, is used in the PEPA model, a different queue length plot is obtained from the PEPA model, see Figure 6.4.

In Figure 6.4, due to the use of the "factor", the rates of the FU and NEW arrivals after the registration are reduced to a low level which is $\lambda_{NEW} = 2.5$ for the new arrivals and $\lambda_{FU} = 10.0$ for the follow up arrivals. Thus, the queue length of the new patients and the follow up patients waiting for the consultants are measured as displayed in Figure 6.4 which is the same as the figure in the previous chapter. In this work, the main aim is to validate this result in the simulation model. Figure 6.4

approximates to be a continuous time plot; however, in order to compare with the simulation outputs easily, the values only at the integer time points are used to the validation. Hence, after running the simulation, the following diagram is obtained namely Figure 6.5.

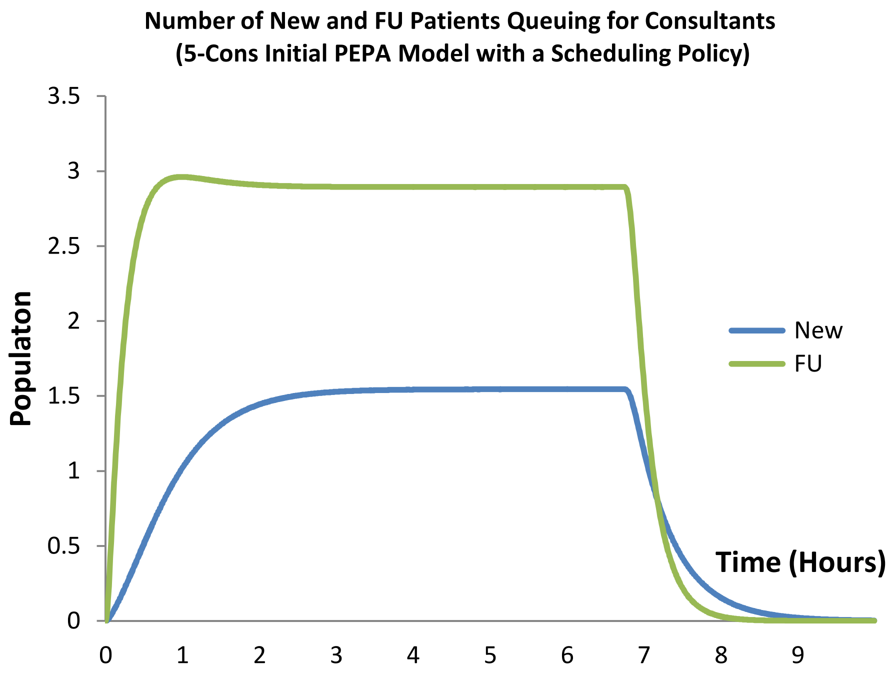


Figure 6. 4 PEPA model: population of patients queuing for consultants with a scheduling policy

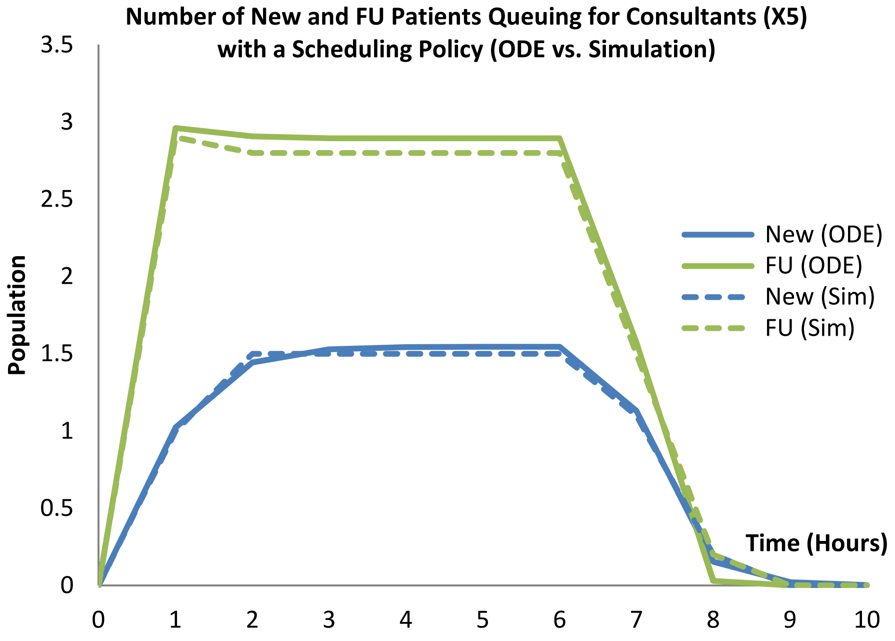


Figure 6. 5 PEPA vs. simulation - population of patients queuing for consultants with a scheduling policy

Figure 6.5 displays the transient queue length of the new patients and the follow up patients. The dashed blue line represents the transient queue length of the new patients obtained from the simulation with the mean value around 1.5. The dashed green line is the transient queue length of the follow up patients from the simulation and has a mean value close to 3.0. It is clear that both dashed lines for the simulation results approximate to the both solid lines which are the results from the PEPA based fluid flow approximation.

However, due to the fewer number of patient events existing in the system, the simulation results have lower accuracy. Hence, the queue length is hard to achieve a stable situation and the cause the low accuracy. To solve this problem, there must be sufficient number of events existing in the network system or a huge number of replications generated from the simulation. In fact, when a factor is applied in this model, the number of patient events existing in the system must be in a small range. This means it is impossible to obtain a fluid plot from the simulation. Thus, in order to guarantee the accuracy of the simulation, a huge number of replications must be measured in the simulation. To verify this conclusion, a large number of patient events will be applied in the model in the next sub-section.

6.5.2 Measurement with a large number of patient events

In order to verify the previous conclusion, a large number of patient events are used in the simulation which means the number of the new patients $\boldsymbol{N_{new}}$ is 100 and $\boldsymbol{N_{FU}}$ is 400. These values stand for the total amount of patients during five working days. All other parameters and conditions remain the same. Firstly, the measurement is carried out when the simulation model does not use a factor. Figure 6.6 represents the queue length of the new and the follow up from the simulation in contrast to the plots from the PEPA model.

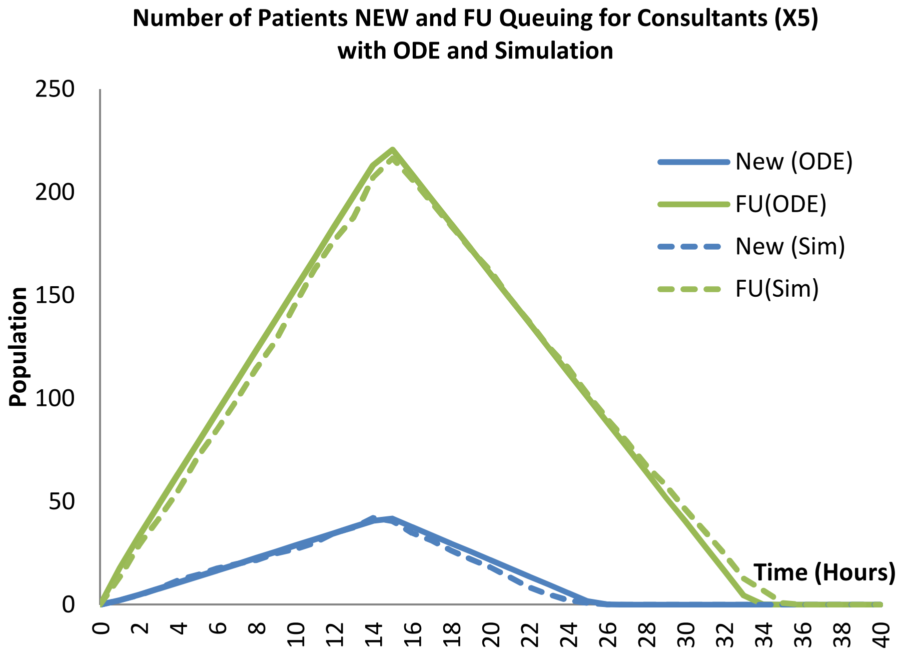


Figure 6. 6 PEPA vs. simulation - population of patients queuing for consultants (100×New and 400×FU)

Figure 6.6 illustrates the queue length of two types of patients waiting for the consultants. The real lines are the results from the PEPA model with the fluid flow analysis. The dot lines are obtained from the simulation. It is clear that both the real lines and the dot lines are close to each other. Thus, when the factor is not used in the model, the simulation results are roughly equivalent to the results from PEPA model.

Thereafter, the factor is applied in the model again. Figure 6.7 describes the queue length of the new and the follow up when a factor is employed for the registration. The lines displayed in the figure is similar with that in the Figure 6.4 except the terminate time point which is due to the larger number of patient events. From the figure, the queue length of the new still keeps stable around 1.5 for the most period; meanwhile, the queue length of the follow up is about 3.0 on average in the most time span. This means that the increased patient amount just extends the service period when a scheduling factor is applied in the model. The queue length of each patient group is not related to the total patient amount, as the existing number of patient events in the system is fixed after using a factor.

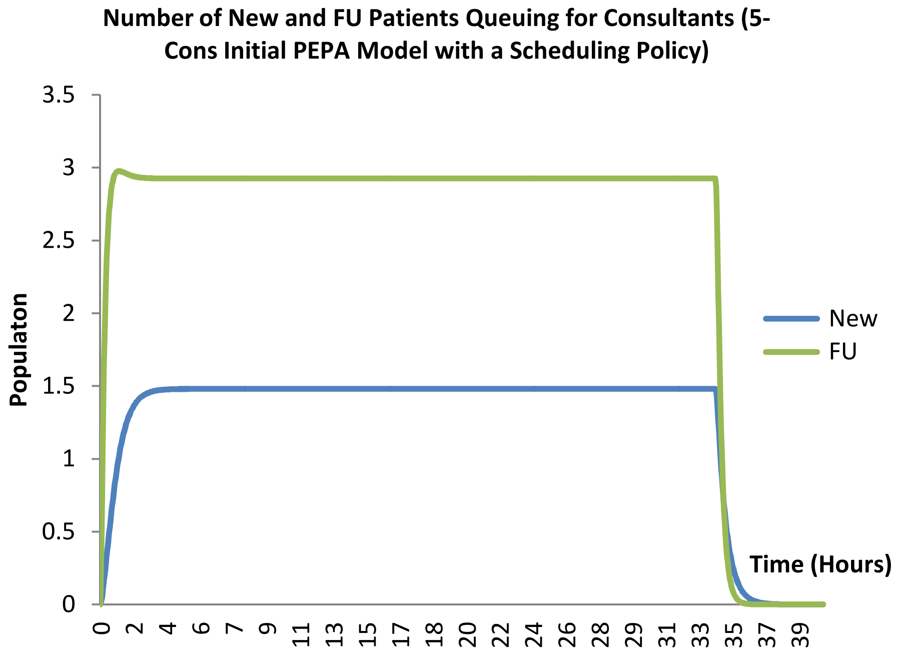


Figure 6. 7 PEPA model: population of patients queuing for consultants with a scheduling policy (100×New and 400×FU)

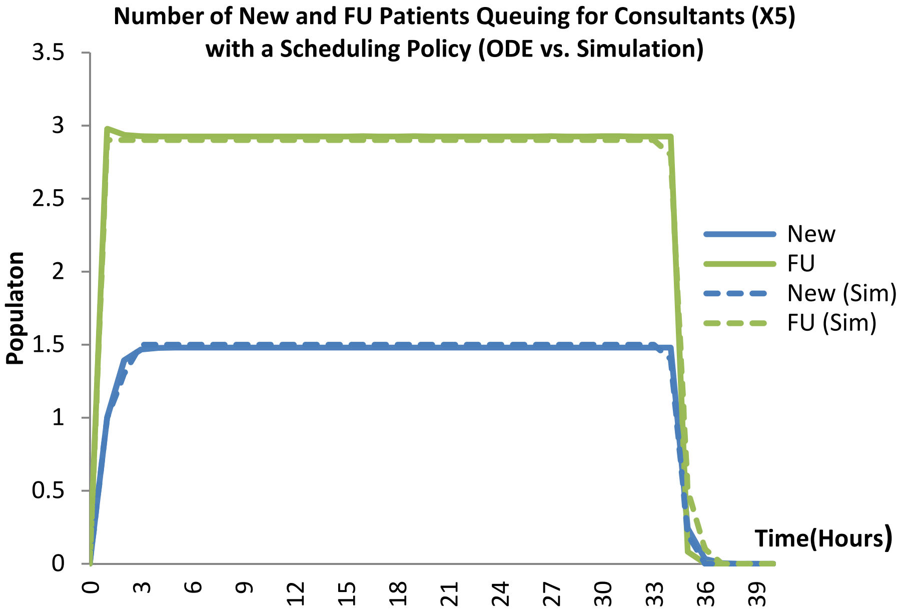


Figure 6. 8 PEPA vs. simulation - population of patients queuing for consultants with a scheduling policy (100×New and 400×FU)

Figure 6.8 collected from the simulation shows the queue length of the new and the follow waiting for consulting. In terms of the dashed lines in the Figure 6.8, the mean of the queue length for the new patients is around 1.5; and the mean queue length for the follow up patients is about 3.0. Obviously, the mean values (dashed lines) from the simulation are very close to the values from the PEPA model (solid lines).

Therefore, when a scheduling policy (factor) is used in the model, it is difficult to obtain the accurate results from the discrete event simulation due to the fewer amounts of events existing in the system.

6.5.3 Measurement with one more consultant

This sub-section aims to explore the patient queue length in the condition of one extra consultant for the follow up patients, so there are 4 consultants working for the follow up patients and 2 consultants for the new patients. Except the one more consultant for the follow up, all other conditions remain the same as before. Here, the scheduling policy is not used in the model. The measurement is aimed to compare the difference of the queue length between five-consultant and six-consultant schemes.

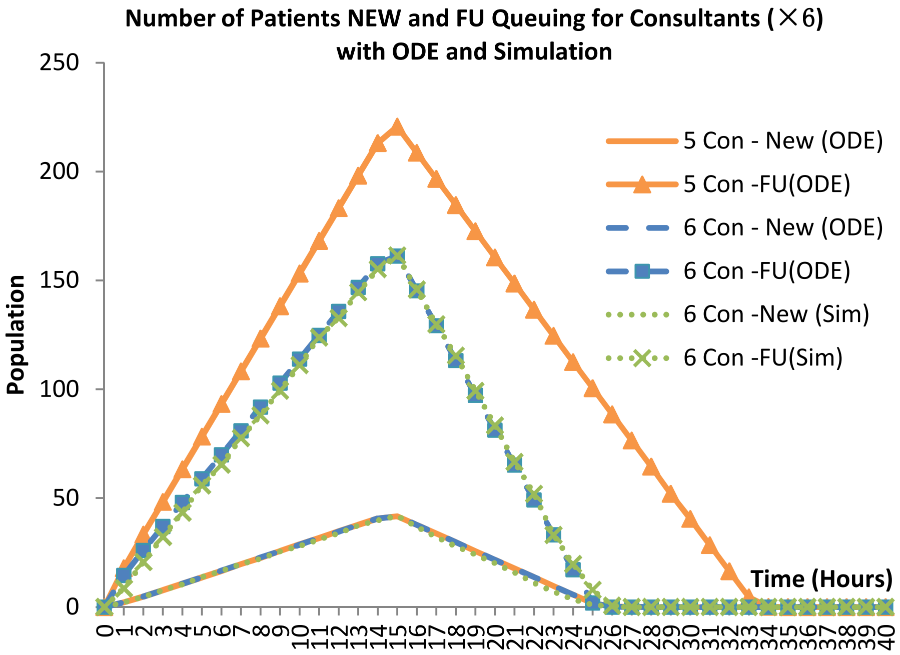


Figure 6. 9 Cons×6 vs. Cons×5: population of patients queuing for consultants (100×New and 400×FU)

Figure 6.9 displays the queue length of both the new and the follow in terms of the number of the consultants. In the figure, the real yellow lines represent the queue length in the condition of five consultants. The blue dash lines represent the queue length in the condition of six consultants via ODEs; meanwhile, the green dot lines are obtained in the same condition six consultants but from the simulation.

From the Figure 6.9, two key features can be found. The first one is that the figure from the simulation is quite close to that from the ODEs. The blue dash lines and the green dot lines are almost overlapped. The second feature is the queue length of the follow up patients in the condition of 6 consultants is much less than the queue length in the condition of 5 consultants; moreover, when the model uses 6 consultants, the total time of completing all follow up patients saves about 8 hours in contrast to the condition of 5 consultants. However, for the new patients, the queue length and the terminate time point have no change, as the number of consultants serving the new remains the same.

6.5.4 Summary

To sum up, this section successfully validates the previous analysis using the fluid flow analysis in the PEPA model through an equivalent discrete event simulation model based on the OMNeT++ IDE. The simulation result is almost identical with the results of fluid flow analysis when no scheduling policy (factor) is applied in the model; nevertheless, when a scheduling policy (factor) appears in the model, simulation result has a low accuracy due to quite few events existing in the system, but the approximate mean in the simulation still approaches the fluid flow result.

6.6 Conclusion

The main goal of this chapter is validate the measurement from PEPA model in preceding chapter. The main technique used for the validation is the discrete event simulation based on the OMNeT++ IDE. Moreover, another target in this chapter is to explore whether an open queue network can be modelled successfully with the discrete event simulation tool OMNeT++ IDE.

In this work, an equivalent simulation model is successfully built in terms of the original PEPA model, and a serious of measurements is obtained from the simulation. According to the measurements, the following conclusions are obtained:

- When the model scheme does not have a scheduling policy (factor), the results from the simulation are almost identical comparing with the PEPA model.
- When the model scheme applies a scheduling policy (factor), it is difficult to obtain the stable figure, as the "factor" restrains the amount of events to a low level in the system and the measurement cannot approach a stable situation in the simulation run. However, the mean value from the simulation output roughly equals the mean value in the PEPA model.

Consequently, this work obtains accurate results from the simulation and validates the PEPA model successfully. The fluid flow analysis generates accurate analysis in the capacity model. Furthermore, there are still some problems which are due to the features of diffident modelling techniques. PEPA model applies fluid flow analysis which is able to obtain a smooth serious of figure in the condition of low event amount; however, in the discrete event simulation, the generated figures has obvious variance around the mean by reason of the less event amount in the network. Thus, to obtain the accurate measurements, the discrete event simulation should run sufficient replications.

Chapter 7 Conclusion

7.1 Aims

This thesis, firstly, aims to build the patient flow model and capacity model depending on a hospital scenario with Performance Evaluation Process Algebra (PEPA), and then equivalently simulate the stochastic PEPA model with discrete event simulation for validation purpose. Secondly, it also uses several novel techniques in the modelling, such as the function rate in the PEPA model, and the open queue network with finite arrivals in the simulation. Finally, the thesis has a core aim to investigate the performance of the scheduling policies (dynamic scheduling policy and static scheduling policy) and the capacity planning strategy.

Through the research, the previous mentioned objectives in Chapter 1 have been achieved:

- ❖ PEPA is successfully used to build a large scale model which has the complex structure and a large number of events in the model system.
- ❖ Fluid flow approximation is certified to be efficient and reliable in the analysis of large scale models.
- ❖ Function rates are successfully used to model some complex process, such as, scheduling process. The use of function rate extends the functionality of PEPA method.
- ❖ Discrete event simulation is used to validate the fluid flow analysis results from PEPA and CMDL models. The findings prove that the fluid flow approximation can generate reliable analysis results in contrast to the results from the simulation.

In addition, this thesis has achieved the objectives for evaluating the performance of several scheduling policies, such as the static scheduling policies and two dynamic scheduling polices; and the objective for the exploration for capacity planning. The achievements of this performance evaluation work will be detailed in the next contribution section.

7.2 Contributions

Chapter 3 and 4 research the performance of scheduling policies based on the patient flow model.

In Chapter 3, the dynamic scheduling policy and the static scheduling policy are modelled with PEPA and CMDL, and analyzed with the fluid flow analysis. The final outcomes demonstrate that dynamic scheduling policies have more powerful ability than the static policy under all kinds of environment conditions. Furthermore, to achieve the dynamic scheduling, function rate is successfully adopted in the model and is transformed to the equivalent CMDL for further analysis.

In Chapter 4, the Java based discrete event simulation is used to rebuild the patient flow model to measure the performance of the scheduling policies in order to verify the findings from the PEPA based model analysis. According to the outputs, the results from the discrete event simulation are almost identical to those from the fluid flow analysis.

Chapter 3 and 4 explore the use of function rates in the PEPA model which is challenging point in this part, and also promote a simulation of the closed network which is equivalent to the PEPA model. Meanwhile, this stage proves the optimal performance of dynamic scheduling policies which is useful to aid the development of the smart environment. This part mainly considers the improvement of custom scheduling in a system, which is meaningful for improving the efficiency.

Chapter 5 and 6 explore the efficient capacity planning based on the work flow of the Rheumatology department.

In Chapter 5, the work flow capacity is modelled with PEPA and CMDL, and the analysis is carried out with the ODEs based fluid flow analysis. This part generates the efficient capacity plan in terms of the requirements which are the less resource usage and less queuing time. In addition, it models different type work flows in order to compare the efficiency of different types of work flow, and find an optimal scheme. It worth to mention that the function rate is successful used again to achieve the grouping process to refine the work flow. Finally, as the figures used in the model are genuine statistics from the Rheumatology department of Royal Hallamshire Hospital, the conclusions are quite helpful in refining the current work flow and improve the efficiency and quality of the service.

In Chapter 6, an OMNeT++ discrete event simulation is adopted to model an equivalent capacity model which is actually an open queue network with limited number of customer events. This simulation cannot reach a steady state in the run. The key point is to model this situation without a steady state and carry out the measurement. This part succeeds in modelling this open queue network and generates a set of results which effectively validate the results from the PEPA model in Chapter 5.

The findings of Chapter 5 and 6 mainly demonstrate the importance of capacity planning for the efficiency work flow especially having limited serving resource. Furthermore, it is also challenging to complete the simulation of an open queue network with limited number of customer events and the accurate analysis based on the simulation run.

7.3 Further work

This thesis demonstrates the performance evaluation of scheduling policies and capacity plans with both formal modelling techniques and discrete event simulation. For the further work, these aspects following the current work should be considered:

- ❖ The first half of this thesis focuses on modelling dynamic scheduling policies. The complex dynamic scheduling process is achieved by using function rates in the models. The function rate is used to simplify the complex model behaviours to extend the ability of formal methods. However, such simple function rates representing complex behaviours are difficult to validate with the discrete event simulation. In the further work, a more simulation techniques should be explored to achieve flexible function rates in order to minimize the time cost of simulation.

- The model scenario used for the scheduling case just considers the basic process in the hospital. However, in the real hospital, many human factors also affect the scheduling process, such as, patient no-show rates, lateness and interruption of doctors, and so on. In the following work, more human factors can be considered; thus, the scheduling process can be modelled in a larger scale.
- The second half of the thesis models the capacity of a hospital department based on its workflow. In this part, capacity plans are evaluated with both formal methods and discrete event simulation. As this model is defined as an open queue network with limited events in the network, it is difficult to obtain high accuracy of the simulation. For the further work, more efficient simulator should be adopted to facilitate the measurement process.
- For the capacity models, this thesis considers the allocation strategies of medical staff in a department. Moreover, there are many other hospital resources which can be considered in the capacity scenario. These resources consist of examination equipments, operating rooms, beds, and so on. In the further work, capacity planning for other hospital resources can be researched in order to better support the efficiency of healthcare system.

In addition, a more complex scenario can be proposed for industry purpose. For example, we can explore a comprehensive model including patient scheduling process, capacity calculation for necessary resources, and resource allocation. Such complex process is a big challenge to the formal modelling. More modelling techniques may be used such as model component substitution technique. Real statistical data can be used to test the model design. According to the reliable results, the models can be used as the prototype of industry applications. Whatever, the further work should aim to improve the utilization of the service and facilitate individuals in the hospital.

Reference

[1] A. Caragliu, C. Del Bo and P. Nijkamp, "Smart Cities in Europe", Journal of Urban Technology, 18(2), 2011.

[2] R. Giffinger, F. Christian, K. Hans, K. Robert, PM. Nataša, M. Evert, "Smart Cities - Ranking of European Medium-sized Cities", Centre of Regional Science, Vienna University of Technology, 2007.

[3] R. Gold, J. S. Brown, "The Origins of Ubiquitous Computing Research at PARC in the Late 1980s", IBM Systems Journal, pp. 693-696, 1999.

[4] D. Cook, S. Das, "Smart Environments: Technology, Protocols and Applications", Wiley-Interscience, 2005.

[5] S. Poslad, "Ubiquitous Computing Smart Devices", Smart Environments and Smart Interaction, Wiley, 2009.

[6] S.W.M.Au-Yeung, P.G.Harrison, W.J.Knottenbelt, "A Queuing Network Model of Patient Flow in an Accident and Emergency Department", 20th Annual European and Simulation Modelling Conference, pp. 60-67, October 2006.

[7] The King's Fund, Has the government met the public's priorities for the NHS? A King's Fund briefing for the BBC, available at http://news.bbc.co.uk/nol/shared/bsp/hi/pdfs/18_03_04_kings_fund_audit.pdf, last accessed 20/02/2013.

[8] Healthcare Commission, NHS performance ratings 2004/2005, Commission for Healthcare Audit and Inspection, 2005.

[9] Healthcare Commission, Acute hospital portfolio review, accident and emergency, London: Commission for Healthcare Audit and Inspection, 2005.

[10] M.D. Harrison and M. Massink. "Modelling interactive experience, function and performance in ubiquitous systems", Electronic Notes in Theoretical Computer Science 261: 23-42, 2010.

[11] R. W. Hall, “Patient Flow: Reducing Delay in Healthcare Delivery”, volume 91, Springer, 2006.

[12] E. W. Woodcock, "Mastering Patient Flow: More Ideas to Increase Efficiency and Earnings", Medical Group Management Association, 2003.

[13] D. Ouelhadj, S. Petrovic, “A Survey of Dynamic Scheduling in Manufacturing Systems”, Journal of Scheduling 12(4): 417-431, 2009.

[14] E. Wolstenholme, “A Patient Flow Perspective of U.K. Health Service: Exploring the Case for New ‘Intermediate Care’ Initiatives”, John Wiley and Sons, 1999.

[15] X. Chen and N. Thomas, Performance Evaluation of Dynamic Scheduling in a Environment, in: Proc. 27th UK Performance Engineering Workshop, University of Bradford, 2011.

[16] J. A. Hillston, “A Compositional Approach to Performance Modelling”, Cambridge University Press, 1996.

[17] J. A. Hillston, “Fluid Flow Approximation of PEPA Models”. In: Proceedings of QEST05, pp. 33-43. IEEE Computer Society, 2005.

[18] B. Jeremy, S. Gilmore, "Stochastic Simulation Methods Applied to a Secure Electronic Voting Model", Electronic Notes in Theoretical Computer Science, 2005.

[19] J. Bradley, S. Gilmore and N. Thomas, Performance analysis of Stochastic Process Algebra models usinig Stochastic Simulaing, in: Proceedings of 20th IEEE International Parallel and Distributed Processing Symposium, IEEE Computer Society, 2006.

[20] M. H. MacDougall, Simulating Computer Systems: Techniques and Tools. MIT Press, 1987.

[21] B. R. Haverkort, “Performance of Computer Communication Systems – A Model Based Approach”, John Wiley, 1999.

[22] Ò Miró, M Sánchez, G Espinosa, B Coll-Vinent, E Bragulat, J Millá, “Analysis of Patient Flow in the Emergency Department and the Effect of an Extensive Reorganisation”, Emergency Medicine Journal, 20(2): 143-148, 2003.

[23] T. J. Coats, S. Michalis, “Mathematical Modelling of Patient Flow Through an Accident and Emergency Department”, Emergency Medicine Journal, 18(3): 190-192, 2001.

[24] M. J. Côté, W. E. Stein, “An Erlang-based Stochastic Model for Patient Flow”, Omega, International Journal of Management Science, 28(3), pp. 347–359, June 2000.

[25] B. R. Asplin, T. J. Flottemesch, and B. d. Gordon, “Developing Models for Patient Flow and Daily Surge Capacity Research”, in: Proceedings of the Academic Emergency Medicine Consensus Conference, ‘‘Establishing the Science of Surge’’, San Francisco, CA, May 17, 2006.

[26] A. Marshall, C. Vasilakis, and Elia El-darzi, “Length of Stay-based Patient Flow Models: Recent Developments and Future Directions”, Health Care Management Science 8(3): 213-220, 2005.

[27] G. R. Hung, S. R. Whitehouse, C. B. O’Neill, A. P. Gray, and N. M. Kissoon, “Computer Modelling of Patient Flow in a Paediatric Emergency Department Using Discrete Event Simulation”, Paediatric Emergency Care, 23(1):5-10, January 2007.

[28] D. J. Medeiros, E. Swenson, and C. DeFlitch, “Improving Patient Flow in a Hospital Emergency Department”, in: Proceedings of the 2008 Winter Simulation Conference, pp. 1526-1531, 2008.

[29] A. Kolker, “Process Modelling of Emergency Department Patient Flow: Effect of Patient Length of Stay on ED Diversion”, Journal of Medical Systems, 32(5): 389-401, 2008.

[30] M. Gunal, and M. Pidd, “Understanding Accident and Emergency Department Performance Using Simulation”, in: Proceedings of the 2006 Winter Simulation Conference , pp. 446-456, IEEE, 2006.

[31] M. Guo, M. Wagner and C. West, “Outpatient clinic scheduling – a Simulation Approach”, in: Proceedings of the 2004 Winter Simulation Conference, pp.1981-1987, 2004.

[32] DN. Pham, A. Klinkert, “Surgical Case Scheduling as a Generalized Job Shop Scheduling Problem”, European Journal of Operational Research, 185: 1011-1025, 2008.

[33] F. Hashimoto, S. Bell, “Improving Outpatient Clinic Staffing and Scheduling with Computer Simulation”, Journal of General Internal Medicine, 11(3): 182-184, Mar, 1996.

[34] J. C. Lowery, “Design of Hospital Admissions Scheduling System Using Simulation”, in: Proceedings of the 1996 Winter Simulation Conference, pp.1199-1204, 1996.

[35] R. Kopach, PC. DeLaurentis, M. Lawley, K. Muthuraman, L. Ozsen, R. Rardin, H. Wan, P. Intrevado, X. Qu and D. Wills, “Effects of Clinical Characteristics on Successful Open Access Scheduling”, Health Care Manage Science, 10: 111-124, 2007.

[36] T. Cayirli and E. Veral, “Outpatient Scheduling in Health Care: a Review of Literature”, Production and Operations Management Society, 12(4), 2003.

[37] E. Rising, R. Baron and B. Averill, “A System Analysis of a University Health Service Outpatient Clinic”, Operations Research, 21(5), pp. 1030-1047, 1973.

[38] T. F. Cox, J. F. Birchall, and H. Wong, “Optimizing the Queuing System for an Ear, Nose and Throat Outpatient Clinic”, Journal of Applied Statistics, 12, pp. 113-126, 1985.

[39] J. R. Swisher, S. H. Jacobson, J. B. Jun and O. Balci, “Modelling and Analyzing a Physician Clinic Environment Using Discrete-Event (Visual) Simulation”, Computers & Operations Research, 28(2), pp. 105-125, 2001.

[40] M. Babes and G. V. Sarma, “Out-Patient Queues at the Ibn-Rochd Health Centre”, Journal of the Operational Research Society, 42(10), pp. 845-855, 1991.

[41] L. Liu and X. Liu, “Block Appointment Systems for Outpatient Clinics with Multiple Doctors”, Journal of the Operational Research Society, 49(12), pp. 1254-1259, 1998.

[42] L. Liu and X. Liu, “Dynamic and Static Job Allocation for Multi-Server Systems,” IIE Transactions, 30(9), pp. 845–854, 1998.

[43] J. Vissers, “Selecting a Suitable Appointment System in an Outpatient Setting,” Medical Care, 17(12), pp. 1207–1220, 1979.

[44] D. J. Heaney, J. G. Howie, and A. M. Porter, “Factors Influencing Waiting Times and Consultation Times in General Practice”, British Journal of General Practice, 41, pp. 315–319, 1991.

[45] J. P. Meza, “Patient Waiting Times in a Physicians’ Office,” The American Journal of Managed Care, 4(5), pp. 703–712, 1998.

[46] J. D. Welch, and N. Bailey, “Appointment Systems in Hospital Outpatient Departments,” The Lancet, 259(6718): 1105-1108, pp. 1105–1108, 1952.

[47] J. Vissers AND J. Wijngaard, “The Outpatient Appointment System: Design of a Simulation Study,” European Journal of Operational Research, 3(6), pp. 459–463, 1979.

[48] C. Ho, and H. Lau, “Minimizing Total Cost in Scheduling Outpatient Appointments,” Management Science, 38(12), pp. 1750–1764, 1992.

[49] A. Mercer, “A Queuing Problem in Which Arrival Times of The Customers are Scheduled,” Journal of the Royal Statistical Society Series B, 22: 108–113, 1960.

[50] A. Mercer, “Queues with Scheduled Arrivals: A Correction, Simplification and Extension,” Journal of the Royal Statistical Society Series B, 35(1), pp. 104–116, 1973.

[51] N. Bailey, “A Study of Queues and Appointment Systems in Hospital Outpatient Departments with Special Reference to Waiting Times,” Journal of the Royal Statistical Society 14, pp. 185–199, 1952.

[52] J. F. Rockart, and P. B. Hofmann, “Physician and Patient Behavior Under Different Scheduling Systems in a Hospital Outpatient Department,” Medical Care, 7(6), pp. 463–470, 1969.

[53] M. J. Blanco White and M. C. Pike, “Appointment Systems in Outpatients’ Clinics and the Effect on Patients’ Unpunctuality,” Medical Care, 2, pp.133–145, 1964.

[54] K. J. Klassen and T. R. Rohleder, “Scheduling Outpatient Appointments in a Dynamic Environment,” Journal of Operations Management, 14(2), pp. 83–101, 1996.

[55] B. Denton and D. Gupta, “A Sequential Bounding Approach for Optimal Appointment Scheduling,” unpublished working paper, DeGroote School of Business, McMaster University, Hamilton, Ontario, DEPARTMENT OF HEALTH (1991), The Patient’s Charter, H. M. S. O., London, U.K., 2001.

[56] S. D. Walter, “A Comparison of Appointment Schedules in a Hospital Radiology Department,” British Journal of Preventive and Social Medicine, 27, pp. 160–167, 1973.

[57] R. Fetter, and J. Thompson, “Patients’ Waiting Time and Doctors’ Idle Time in the Outpatient Setting,” Health Services Research, 1, pp. 66–90, 1996.

[58] A. R. Mahachek, AND T. L. Knabe, “Computer Simulation of Patient Flow in Obstetrical/Gynecology Clinics,” Simulation, 43 (August), pp. 95–101, 1984.

[59] B. Lehaney, S. A. Clarke, and R. J. Paul, “A Case of Intervention in an Outpatients Department,” Journal of the Operational Research Society, 50(9), 877–891, 1999.

[60] R. K. Kanter, and J. R. Moran, “Hospital Emergency Surge Capacity: An Empiric New York Statewide Study”, Annals of Emergency Medicine, 50(3), pp. 314-319, 2007.

[61] W. S. Lovejoy, and Y. Li, “Hospital Operating Room Capacity Expansion”, Management Science, 48(11), pp. 1369-1387, 2002.

[62] J. M. Nguyen, P. Six, D. Antonioli, P. Glemain, G. Potel, P. Lombrail, and P. Le Beux, “A Simple Method to Optimize Hospital Beds Capacity”, International Journal of Medical Informatics, 74, pp.39-49, 2005.

[63] S. Gallivan, M. Utley, T. Treasure, and O. Valencia, “Booked Inpatient Admissions and Hospital Capacity: Mathematical Modelling Study”, British Medical Journal, 324: 280-282, 2002.

[64] J. M. H. Vissers, “Patient Flow-based Allocation of Inpatient Resources: A Case Study”, European Journal of Operational Research 105: 356-370, 1998.

[65] J. M. H. Vissers, “Patient Flow based Allocation of Hospital Resources”, PhD Thesis, Eindhoven University of Technology, Eindhoven, 1995.

[66] S. C. Kim, I. Horowitz, K. K. Young, T. A. Buckley, “Analysis of Capacity Management of the Intensive Care Unit in a Hospital”, European Journal of Operational Research 115: 36-46, 1999.

[67] S. G. Elkhuizen, G. Bor, M. Smeenk, N. S. Klazinga, and P. J. M. Bakker, “Capacity Management of Nursing Staff as a Vehicle for Organizational Improvement”, BMC Health Service Research , 7: 196, 2007.

[68] J. E. McGowan, J. D. Truwit, P. Cipriano, R. E. Howell, M. VanBree, A. Garson Jr, and J. B. Hanks, “Operating Room Efficiency and Hospital Capacity: Factor Affecting Operating Room Use During Maximum Hospital Census”, Journal of the American College of Surgeons, 204: 865-872, 2007.

[69] D. Delia, “Annual Bed Statistics Give a Misleading Picture of Hospital Surge Capacity”, Annals of Emergency Medicine, 48: 384-388, 2006.

[70] W. Mark, "The computer for the 21 century," SIGMOBILE Mob. Comput. Commun. Rev., 3: 3-11, 1999.

[71] Y. Zhao, and N. Thomas, “Comparing Models for the Efficient Analysis of PEPA Model of Non-repudiation Protocols”, in: Proceedings of the 15th Int'l Conference on Parallel and Distributed Systems, IEEE Computer Society, 2009.

[72] X. Chen, M. Harrison and N. Thomas, Performance evaluation of dynamic scheduling within a smart hospital environment, in: Proc. 1st Int'l Workshop on Healthcare Systems Engineering, IEEE Computer Society, 2011.

[73] M. Massink, A. Bracciali, D. Latella, M. Harrison, A Scalable Fluid Flow Process Algebraic Approach to Emergency Egress Analysis. in: Proceedings of the 8th IEEE International Conference on Software Engineering And Formal Methods (SEFM2010). pp. 169-180. IEEE, 2010.

[74] M. Massink, D. Latella, A. Bracciali, M. Harrison, A scalable fluid flow process algebraic approach to emergency egress analysis. in: Proceedings of the 8th International Conference on Software Engineering and Formal Methods, pp. 169-180, IEEE, 2010.

[75] M. Massink, M. Harrison, D. Latella, Scalable analysis of collective behaviour in smart service systems, in: Proceedings of the 2010 ACM Symposium on Applied Computing (SAC), pp. 1173-1180, ACM, 2010.

[76] M. Harrison and M. Massink, Modelling interactive experience, function and performance in ubiquitous systems, Electronic Notes in Theoretical Computer Science, 261, pp. 23-42, Elsevier B.V., 2010.

[77] X. Chen and N. Thomas, Performance Evaluation of Scheduling in a Smart Environment, in: Proc. 26th UK Performance Engineering Workshop, University of Warwick, 2010.

[78] M. Harrison, M. Massink and D. Latella, Engineering Crowd Interaction within Smart Environments, in: Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems, ACM Press, pp. 117-122, 2009.

[79] M. Massink, D. Latella, M.H. ter Beek, M. Harrison, and M. Loreti, A Fluid Flow Approach to Usability Analysis of Multi-user Systems, in: Engineering Interactive Systems 2008 - Proceedings of the 2nd Conference on Human-Centered Software Engineering (HCSE'08), Lecture Notes in Computer Science 5247, Springer-Verlag, pp. 166-180, 2008.

[80] J. Karnon, M. Mackay and T. M. Mills, Mathematical Modelling in Health Care, in: Proceedings of the 18th World IMACS/MODSIM Congress, Caims, Australia, 13-17, July, 2009.

[81] F. Gorunescu, S. I. McClean and P. H. Millard, A Queueing Model for Bed-occupancy Management and Planning of Hospitals, Journal of the Operational Research Society, 53, 19-24, 2002.

[82] A. W. Roscoe, the Theory and Practice of Concurrency, Prentice Hall, 1997.

[83] C. A. R. Hoare, Communicating sequential processes, in: Communications of the ACM, 21 (8): pp.666–677, 1978.

[84] R. Milner, A Calculus of Communicating Systems, Springer Verlag, 1980.

[85] R. Milner, Communication and Concurrency, Prentice Hall, International Series in Computer Science, 1989.

[86] G. D. Plotkin, The Origins of Structural Operational Semantics, in: Journal of Logic and Algebraic Programming, Vol. 60-61, pp. 3-15, 2004.

[87] G. D. Plotkin, A Structural Approach to Operational Semantics, in: Journal of Logic and Algebraic Programming, Vol. 60-61, pp. 17-139, 2004.

[88] S. Devaraj, T. T. Ow, and R. Kohli, Examining the Impact of Information Technology and Patient Flow on Healthcare Performance: A Theory of Swift and Even Flow (TSEF) Perspective, in: Journal of Operations Management, Volume 31, Issue 4, pp. 181-192, 2013.

[89] J. L. Wiler, R. T. Griffey, and T. Olsen, Review of Modelling Approaches for Emergency Department Patient Flow and Crowding Research, in: Academic Emergency Medicine, Volume 18, Issue 12, pp. 1371-1379, 2011.

[90] M. G. Klein, and G. Reinhardt, Emergency Department Patient Flow Simulations Using Spreadsheets, in: Simulation in Healthcare: The Journal of the Society for Simulation in Healthcare, Volume 7, Issue 1, pp. 40-47, 2012.

[91] D. Shah, and J. Shin, Randomized Scheduling Algorithm for Queueing Networks, in: The Annals of Applied Probability, Vol. 22, No. 1, pp. 128-171, 2012.

[92] M. Cheng, M. Kanai-Pak, N. Kuwahara, H. I. Ozaku, K. Kogure, and J. Ota, Dynamic Scheduling-based Inpatient Nursing Support: Applicability Evaluation by Laboratory Experiments, in: International Journal of Autonomous and Adaptive Communications Systems, Volume 5, Number 1, pp. 39-57, 2012.

[93] S. K. Nayak, S. K. Padhy, and S. P. Panigrahi, A Novel Algorithm for Dynamic Task Scheduling, in: Future Generation Computer Systems, Volume 28, Issue 5, pp. 709-717, 2012.

[94] N. Vansteenkiste, C. Lamote, J. Vandersmissen, P. Luysmans, P. Monnens, G. De Voldere, J. Kips, and F. Rademakers, Reallocation of Operating Room Capacity Using the Due-time Model, in: Medical Care, Volume 50, Issue 9, pp. 779-784, 2012.

[95] C. Rau, P. Tsai, S. M. Liang, J. Tan, H. Syu, Y. Jheng, T. Ciou and F. Jaw, Using Discrete-event Simulation in Strategic Capacity Planning for an Outpatient Physical Therapy Service, in: Health Care Management Science, 2013.

[96] HJ. Schutz and R. Kolisch, Approximate Dynamic Programming for Capacity Allocation in the Service Industry, In: European Journal of Operational Research, Volume 218, Issue 1, pp. 239-250, 2012.

Appendix

Appendix 1 – Source code of the static scheduling policy

Source class

```
package s1.Static.Policy;

import eduni.simjava.Sim_entity;
import eduni.simjava.Sim_port;

public class Source extends Sim_entity{

      private Sim_port out;
      private int patientAmount;
      private int type;


      public Source(String name, int patientAmount, int type){

            super(name);
            this.patientAmount = patientAmount;
            this.type = type;

            out = new Sim_port("out");
            add_port(out);
      }

    public int getAmount(){
      return this.patientAmount;
    }

      public void body(){
            for (int i = 1; i <= this.patientAmount; i++){
                  sim_schedule(out, 0.0, type);
            }
      }
}
```

Patient class

```
package s1.Static.Policy;

import eduni.simjava.*;
import eduni.simjava.distributions.*;

public class Patient extends Sim_entity{

      private Sim_port in, out;
      private Sim_negexp_obj delay;
      private double arrivalRate;
      private int tag;

      public Patient(String name, double arrivalRate){

            super(name);
            this.arrivalRate = arrivalRate;

            in = new Sim_port("in");
            out = new Sim_port("out");
            add_port(in);
            add_port(out);

           delay = new Sim_negexp_obj("delay", 1/this.arrivalRate);
            add_generator(delay);
      }

    public double getRate(){
      return this.arrivalRate;
    }

      public void body(){
            while(Sim_system.running()){
                  Sim_event e = new Sim_event();
                  sim_get_next(e);
                  this.tag = e.get_tag();
                  sim_schedule(out, delay.sample(), this.tag);
            }
      }
}
```

Register class

```
package s1.Static.Policy;

import eduni.simjava.*;
import eduni.simjava.distributions.*;

public class Register extends Sim_entity{
      private Sim_port in, outS, outB, outX;
      private double registerRate;
      private Sim_negexp_obj delay;

      private int tag;
      private Sim_stat stat;
      private static final int PATIENT_B = 0;
      private static final int PATIENT_X = 1;
      private static final int PATIENT_BX = 2;

      public Register(String name, double registerRate){

            super(name);
            this.registerRate = registerRate;

            in = new Sim_port("in");
            outS = new Sim_port("outS");
            outB = new Sim_port("outB");
            outX = new Sim_port("outX");
            add_port(in);
            add_port(outS);
            add_port(outB);
            add_port(outX);

            delay = new Sim_negexp_obj("delay",
1/this.registerRate);

            add_generator(delay);


      stat = new Sim_stat();
      stat.add_measure(Sim_stat.QUEUE_LENGTH);
      set_stat(stat);
      }

    public double getRate(){
      return this.registerRate;
    }

    public void body(){
      while(Sim_system.running()){
            Sim_event e = new Sim_event();
            sim_get_next(e);
            tag = e.get_tag();

            switch(tag){
            case PATIENT_B: //Patient B
                  sim_process(delay.sample());
                  sim_completed(e);
                  sim_schedule(outB, 0.0, tag);
                  break;
            case PATIENT_X: //Patient X
                  sim_process(delay.sample());
                  sim_completed(e);
                  sim_schedule(outX, 0.0, tag);
                  break;
            case PATIENT_BX: //Patient BX
                  sim_process(delay.sample());
                  sim_completed(e);
                  sim_schedule(outS, 0.0, tag);
                  break;
            default:
                  break;
            }
      }
    }
}
```

Scheduler class

```
package s1.Static.Policy;
import Customer.Distributions.*;
import eduni.simjava.*;
import eduni.simjava.distributions.*;

public class Scheduler extends Sim_entity {

    private Sim_port in, out, outSB, outSX;
    private double decisionB, decisionX;
    private Sim_random_obj prob;
    private double random_No, ratio;
    private Sim_negexp_dyn delayB, delayX;
    private int tag;

    public Scheduler(String name, double decisionRateB, double
decisionRateX){

      super(name);
      this.decisionB = decisionRateB;
      this.decisionX = decisionRateX;
      in = new Sim_port("in");
      out = new Sim_port("out");
      outSB = new Sim_port("outSB");
      outSX = new Sim_port("outSX");
      add_port(in);
      add_port(out);
      add_port(outSB);
      add_port(outSX);
      delayB = new Sim_negexp_dyn("delayB", 1/this.decisionB);
      delayX = new Sim_negexp_dyn("delayX", 1/this.decisionX);
      add_generator(delayB);
      add_generator(delayX);

      prob = new Sim_random_obj("probility");
      add_generator(prob);
    }

    public void body(){
      while(Sim_system.running()){
            Sim_event e = new Sim_event();
            sim_get_next(e);
            tag = e.get_tag();

            this.ratio = this.decisionB/(this.decisionB +
this.decisionX);

            this.random_No = prob.sample();
                  if (random_No < ratio){
                  //sim_pause(delayB.sample());
                  //sim_completed(e);
                  sim_schedule(outSB, delayB.sample(), tag);
                  }else{
                  //sim_pause(delayX.sample());
                  //sim_completed(e);
                  sim_schedule(outSX, delayX.sample(), tag);
                  }
      }
    }
}
```

Test Class

```
package s1.Static.Policy;

import eduni.simjava.*;
import eduni.simjava.distributions.*;

public class Test extends Sim_entity {

    private Sim_port in, outCont, outB, outX, outBX;
    private Sim_negexp_obj delay;
    private double serviceRate;
    private Sim_stat stat;
    private int tag;
    private static final int PATIENT_B = 0;
    private static final int PATIENT_X = 1;
    private static final int PATIENT_BX = 2;
    private static final int PATIENT_BX_CONT = 3;

    public Test(String name, double serviceRate){

      super(name);
      this.serviceRate = serviceRate;

            in = new Sim_port("in");
            outB = new Sim_port("outB");
            outX = new Sim_port("outX");
            outBX = new Sim_port("outBX");
            outCont = new Sim_port("outCont");
            add_port(in);
            add_port(outB);
            add_port(outX);
            add_port(outBX);
      add_port(outCont);

      delay = new Sim_negexp_obj("delay", 1/this.serviceRate);
      add_generator(delay);
      stat = new Sim_stat();
      stat.add_measure(Sim_stat.QUEUE_LENGTH);
      stat.add_measure(Sim_stat.SERVICE_TIME);
      stat.add_measure(Sim_stat.WAITING_TIME);
      set_stat(stat);
    }

    public double getRate(){
      return this.serviceRate;
    }

    public Sim_negexp_obj getDelay(){
      return this.delay;
    }

    public void body(){

      while(Sim_system.running()){
            Sim_event e = new Sim_event();
            sim_get_next(e);
            tag = e.get_tag();

            switch(tag){
            case PATIENT_B:
```

```
                        sim_process(delay.sample());
                        sim_completed(e);
                        sim_schedule(outB, 0.0, tag);
                        break;
                  case PATIENT_X:
                        sim_process(delay.sample());
                        sim_completed(e);
                        sim_schedule(outX, 0.0, tag);
                        break;
                  case PATIENT_BX:
                        sim_process(delay.sample());
                        sim_completed(e);
                        sim_schedule(outCont, 0.0, tag+1);
                        break;
                  case PATIENT_BX_CONT:
                        sim_process(delay.sample());
                        sim_completed(e);
                        sim_schedule(outBX, 0.0, tag-1);
                        break;
                  default:
                        break;
                  }
            }
      }
}
```

Wait Class

```
package s1.Static.Policy;

import eduni.simjava.*;
import eduni.simjava.distributions.*;

public class Wait extends Sim_entity{
      private Sim_port in, outB, outX, outBX;
      private double waitRate;
      private Sim_negexp_obj delay;
      private int tag;
      private static final int PATIENT_B = 0;
      private static final int PATIENT_X = 1;
      private static final int PATIENT_BX = 2;

      public Wait(String name, double waitRate){

            super(name);
            this.waitRate = waitRate;

            in = new Sim_port("in");
            outBX = new Sim_port("outBX");
            outB = new Sim_port("outB");
            outX = new Sim_port("outX");
            add_port(in);
            add_port(outBX);
            add_port(outB);
            add_port(outX);

            delay = new Sim_negexp_obj("delay", 1/this.waitRate);
            add_generator(delay);
      }

    public double getRate(){
      return this.waitRate;
    }

      public void body(){
            while(Sim_system.running()){
                  Sim_event e = new Sim_event();
                  sim_get_next(e);
                  tag = e.get_tag();

            switch(tag){
            case PATIENT_B:
                  sim_schedule(outB, delay.sample(), this.tag);
                  break;
            case PATIENT_X:
                  sim_schedule(outX, delay.sample(), this.tag);
                  break;
            case PATIENT_BX:
                  sim_schedule(outBX, delay.sample(), this.tag);
                  break;
            default:
                  break;
            }
            }
      }
}
```

Main Class

```
package s1.Static.Policy;
import eduni.simjava.Sim_system;

public class Main {

      private static int total_patientB = 30;
      private static int total_patientX = 30;
      private static int total_patientBX = 32;

      private static double r_arrival = 10.0;
      private static double arrivalRate_B = r_arrival;
      private static double arrivalRate_X = r_arrival;
      private static double arrivalRate_BX = r_arrival;

      private static double decisionRate_B = 1.0;
      private static double decisionRate_X = 1.0;

      private static double serviceRate_Blood = 8.0;
      private static double serviceRate_Xray = 4.0;

      private static double registerRate = 2000.0;
      private static double waitRate = 2000.0;

      private static final int type_patientB = 0;
      private static final int type_patientX = 1;
      private static final int type_patientBX = 2;

      public static void main(String[] args) {

            System.out.println("step1--Static: " + " B-" +
total_patientB + " X-"+total_patientX + " BX-" + total_patientBX);
          System.out.println("                  " + " Arrival
Rate_B---" + arrivalRate_B);
          System.out.println("                  " + " Arrival
Rate_X---" + arrivalRate_X);
          System.out.println("                  " + " Arrival
Rate_BX---" + arrivalRate_BX);
          System.out.println("                  " + " Service
Rate_Blood---" + serviceRate_Blood);
          System.out.println("                  " + " Service
Rate_Xray---" + serviceRate_Xray);

            Sim_system.initialise();

            Source sourceB = new Source("SourceB", total_patientB,
type_patientB);
            Source sourceX = new Source("SourceX",  total_patientX,
type_patientX);
            Source sourceBX = new Source("SourceBX",
total_patientBX, type_patientBX);

            Patient patientB = new Patient("PatientB",
arrivalRate_B);
            Patient patientX = new Patient("PatientX",
arrivalRate_X);
            Patient patientBX = new Patient("PatientBX",
arrivalRate_BX);

            Test blood = new Test("Blood", serviceRate_Blood);
            Test xray = new Test("Xray", serviceRate_Xray);

            Scheduler scheduler = new Scheduler("Scheduler",
decisionRate_B, decisionRate_X);

            Register register = new Register("Register",
registerRate);
            Wait wait = new Wait("Wait", waitRate);

      Sim_system.link_ports("SourceB", "out", "PatientB", "in");
      Sim_system.link_ports("SourceX", "out", "PatientX", "in");
      Sim_system.link_ports("SourceBX", "out", "PatientBX", "in");

      Sim_system.link_ports("PatientB", "out", "Register", "in");
      Sim_system.link_ports("PatientX", "out", "Register", "in");
      Sim_system.link_ports("PatientBX", "out", "Register", "in");

      Sim_system.link_ports("Register", "outB", "Blood", "in");
      Sim_system.link_ports("Register", "outX", "Xray", "in");
      Sim_system.link_ports("Register", "outS", "Scheduler", "in");

      Sim_system.link_ports("Scheduler", "outSB", "Blood", "in");
      Sim_system.link_ports("Scheduler", "outSX", "Xray", "in");

      Sim_system.link_ports("Blood", "outCont", "Xray", "in");
      Sim_system.link_ports("Blood", "outB", "Wait", "in");
      Sim_system.link_ports("Blood", "outBX", "Wait", "in");

      Sim_system.link_ports("Xray", "outCont", "Blood", "in");
      Sim_system.link_ports("Xray", "outX", "Wait", "in");
      Sim_system.link_ports("Xray", "outBX", "Wait", "in");

      Sim_system.link_ports("Wait", "outB", "PatientB", "in");
      Sim_system.link_ports("Wait", "outX", "PatientX", "in");
      Sim_system.link_ports("Wait", "outBX", "PatientBX", "in");

      Sim_system.set_trace_detail(false, false, false);
      Sim_system.set_transient_condition(Sim_system.TIME_ELAPSED,
5000);

      Sim_system.set_termination_condition(Sim_system.TIME_ELAPSED,
10000, true);

      Sim_system.set_output_analysis(Sim_system.IND_REPLICATIONS,
30, 0.95);
      Sim_system.set_report_detail(false, false);
      //Sim_system.generate_graphs(true);
      Sim_system.run();
      }
}
```

Appendix 2 – Source code of the dynamic scheduling policy 1

Scheduler class

```
package s1.Dynamic.Policy1;

import Customer.Distributions.*;
import eduni.simjava.*;
import eduni.simjava.distributions.Sim_random_obj;

public class Scheduler extends Sim_entity {

    private Sim_port in, out, outSB, outSX;
    private double decisionB, decisionX, tranDecisionB,
tranDecisionX;
    private Sim_random_obj prob;
    private double random_No, ratio;
    private Test blood, xray;
    private Sim_negexp_dyn delayB, delayX;
    private int tag;

    public Scheduler(String name, double decisionRateB, double
decisionRateX, Test blood, Test xray){

      super(name);
      this.decisionB = decisionRateB;
      this.decisionX = decisionRateX;
      this.blood = blood;
      this.xray = xray;
      this.tranDecisionB = 0;
      this.tranDecisionX = 0;

      in = new Sim_port("in");
      out = new Sim_port("out");
      outSB = new Sim_port("outSB");
      outSX = new Sim_port("outSX");
      add_port(in);
      add_port(out);
      add_port(outSB);
      add_port(outSX);

      delayB = new Sim_negexp_dyn("delayB", 1/this.decisionB);
      delayX = new Sim_negexp_dyn("delayX", 1/this.decisionX);
      add_generator(delayB);
      add_generator(delayX);

      prob = new Sim_random_obj("probility");
      add_generator(prob);
    }

    public void body(){
      while(Sim_system.running()){
            Sim_event e = new Sim_event();
            sim_get_next(e);
            tag = e.get_tag();

            this.tranDecisionB =
this.decisionB/this.blood.sim_waiting();
            this.tranDecisionX =
this.decisionX/this.xray.sim_waiting();
```

```
             delayB.set_mean(1/this.tranDecisionB);
             delayX.set_mean(1/this.tranDecisionX);

             this.ratio = this.tranDecisionB/(this.tranDecisionB +
this.tranDecisionX);

             this.random_No = prob.sample();
                    if (random_No < ratio){
                    sim_pause(delayB.sample());
                    sim_schedule(outSB, 0.0, tag);
                    }else{
                    sim_pause(delayX.sample());
                    sim_schedule(outSX, 0.0, tag);
                    }
       }
     }
}
```

Appendix 3 – Source code of the dynamic scheduling policy 2

Scheduler class

```
package s1.Dynamic.Policy2;

import Customer.Distributions.*;
import eduni.simjava.*;
import eduni.simjava.distributions.*;

public class Scheduler extends Sim_entity {

    private Sim_port in, out, outSB, outSX;
    private double decisionB, decisionX, tranDecisionB,
tranDecisionX;
    private Sim_random_obj prob;
    private double random_No, ratio;
    private Test blood, xray;
    private Sim_negexp_dyn delayB, delayX;
    private Sim_negexp_obj delayBlood, delayXray;
    private int tag;
    private double tranRespTimeB, tranRespTimeX;

    public Scheduler(String name, double decisionRateB, double
decisionRateX, Test blood, Test xray){

      super(name);
      this.decisionB = decisionRateB;
      this.decisionX = decisionRateX;
      this.blood = blood;
      this.xray = xray;
      this.tranDecisionB = 0;
      this.tranDecisionX = 0;
      this.tranRespTimeB = 0;
      this.tranRespTimeX = 0;

      this.delayBlood = blood.getDelay();
      this.delayXray = xray.getDelay();

      in = new Sim_port("in");
      out = new Sim_port("out");
      outSB = new Sim_port("outSB");
      outSX = new Sim_port("outSX");
      add_port(in);
      add_port(out);
      add_port(outSB);
      add_port(outSX);

      delayB = new Sim_negexp_dyn("delayB", 1/this.decisionB);
      delayX = new Sim_negexp_dyn("delayX", 1/this.decisionX);
      add_generator(delayB);
      add_generator(delayX);

      prob = new Sim_random_obj("probility");
      add_generator(prob);
    }

    public void body(){
      while(Sim_system.running()){
            Sim_event e = new Sim_event();
            sim_get_next(e);
            tag = e.get_tag();

            this.tranRespTimeB = (this.blood.sim_waiting() + 1) *
(this.delayBlood.sample());
            this.tranRespTimeX = (this.xray.sim_waiting() + 1) *
(this.delayXray.sample());

            this.tranDecisionB = this.decisionB/this.tranRespTimeB;
            this.tranDecisionX = this.decisionX/this.tranRespTimeX;
            delayB.set_mean(1/this.tranDecisionB);
            delayX.set_mean(1/this.tranDecisionX);

            this.ratio = this.tranDecisionB/(this.tranDecisionB +
this.tranDecisionX);

            this.random_No = prob.sample();
                  if (random_No < ratio){
                  sim_pause(delayB.sample());
                  sim_schedule(outSB, 0.0, tag);
                  }else{
                  sim_pause(delayX.sample());
                  sim_schedule(outSX, 0.0, tag);
                  }
      }
    }
}
```