Performance Evaluation of Dynamic Scheduling in a Hospital Environment

Conference Paper · July 2011

CITATION
1

READS
20

3 authors, including:

Michael Harrison
Newcastle University
238 PUBLICATIONS **2,379** CITATIONS
SEE PROFILE

Nigel Thomas
Newcastle University
136 PUBLICATIONS **833** CITATIONS
SEE PROFILE

Performance Evaluation of Dynamic Scheduling in a Hospital Environment

Xiao Chen, Michael Harrison and Nigel Thomas,
School of Computing Science, Newcastle University, UK
{xiao.chen1, michael.harrison, nigel.thomas}@ ncl.ac.uk

Abstract. In this paper, the performance of different scheduling policies in a smart environment is considered. Models are generated for a hospital scenario, and these model three scheduling policies within a hospital patient flow scenario. The main modelling techniques used are Performance Evaluation Process Algebra (PEPA) and Chemical Model Definition Language (CMDL). Fluid flow approximation is adopted in the model analysis by solving a set of ordinary differential equations (ODEs). Furthermore, discrete event simulation is used to verify the results generated from the PEPA and CMDL models. The findings show that dynamic scheduling policies yield a performance superior to static policy; and the two types of dynamic scheduling policies each have particular features so that one policy may be more appropriate than the other, depending on actual conditions. Finally, more work should be planned to build a capacity model based on a new clinical scenario.

Keywords: Dynamic Scheduling; Performance Evaluation; PEPA; CMDL; ODEs;

1 Introduction

Smart environments, ubiquitous computing, pervasive computing, context-awareness, ambient intelligence, among others, are related concepts and technologies that aim to help people complete their tasks and activities without explicit interaction with computers, a vision that was first introduced by Weiser [1].

In a smart environment, a ubiquitous system is embedded in order to provide relevant information to individuals in a physical environment to guide and support their experience of that environment. In the real world, such smart environments can be applied in various physical environments: in a hospital, patients are directed to have a test, or surgery, or to see a doctor, or attend for a checkup, according to their appointments or decisions made by doctors or nurses; in an airport, passengers are notified which queue is for check-in, baggage screening or 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 cards, pay fees, or meet their tutors at the beginning of the academic year; in a sports centre or similar large buildings, visitors are guided to a specific location or office with the 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 many simple inexpensive sensor technologies to facilitate activities, enhancing the convenience to, and well-being of, citizens by providing them with information they need, when and how they need it.

Smart environments can improve the experience and collective behaviour of individuals within a physical space. To design an efficient and effective smart environment, techniques are required to model and analyze the effects of design decisions early in the design process. Such early modeling and analysis aims to predict before deployment the impact that the system will have and to provide information that can be used formatively in later refinements or versions of the design. As per the previous work of Harrison [2], before fielding the system, model based analysis techniques can be employed in analyzing usability aspects of smart environments where many people are present. Moreover, an approach based on ordinary differential equations (ODEs), derived from process algebraic specifications modeling both large groups of users and the environment, may provide a useful analysis of the collective behavior and the more quantitative aspects of usability of smart environments [2]. Thus, in this paper, system models will be achieved with PEPA modeling and CMDL modeling, and analysis will be handled with fluid flow analysis (ODEs). The automatic derivation of systems of ODEs from PEPA specifications, the algorithms to solve ODE and the generation of the numerical results are also supported by PEPA Eclipse Plug-in.

The goal of this paper is to model different scheduling policies which are employed by a smart system to carry out the routing service in a physical environment. The scenario is based on patient flow scheduling in a hospital. Various scheduling policies will be modeled and analyzed to assess the performance of each. It is worthy mentioning that a key point is to explore the performance of the dynamic scheduling policy. To carry out the dynamic scheduling analysis, CMDL modeling is adopted which is transformed from an equivalent PEPA model. More detailed information will be found in the case study section.

Section 1 introduces the scenario and the main goal of the research. Section 2 describes the case study, including three sorts of scheduling policies. Section 3 briefly introduces the model techniques used; namely, PEPA, CMDL and discrete event simulation. Section 4 gives the original PEPA models of two scheduling policies and the transformed CMDL models. Section 5 demonstrates the related analysis of CMDL models. Section 6 states the results of discrete event simulation. Section 7 briefly concludes the analysis.

2 Case Study

In previous work [11], the performance was explored of two types of scheduling policies in a scenario in which all patients had to take three tests. In this work, the scenario is updated to a new situation so as to model three scheduling policies.

2.1 Scenario

As proved before [11], 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 model assumptions, firstly, it is assumed only two tests will be operated for patients: blood test and X-ray scan. Instead, 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. A 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.

2.2 Scheduling Policies

Based on the above patient flow model, three scheduling policies will be embedded in the model, and these three policies have similar functionality to those used before. All these three scheduling policies are noted as:

Static Scheduling Policy:

The static scheduling policy is a method which directs incoming clients to the next destination with a fixed probability. The set of probabilities can be easily optimised to balance average load, but this policy will intuitively perform poorly under fluctuating conditions.

Dynamic Scheduling Policy 1:

Dynamic Scheduling Policy 1 is policy using transient queue length to determine scheduling rate so as to manage the scheduling process dynamically.

Dynamic Scheduling Policy 1 uses transient queue length to determine scheduling rate so as to manage the scheduling process. According to this policy, when the queue length increases, this means that a growing number of clients are waiting for service and 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 this function rate is the transient queue length, and the function value is 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 going to this component. The expression of function rate for Policy 1 is:

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

Dynamic Scheduling Policy 2:

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 the response for blood and X-ray tests respectively, and then compares these 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 current 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 current response time varies with time. The value of current response time can be calculated during the run of the model, too. Details about how to calculate current response time and average response will be introduced later. According to the formula, as average response time is fixed, if current 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 current response time becomes less (P increases). Hence, in order to manage the patient flow in the scheduling process, the functional 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.

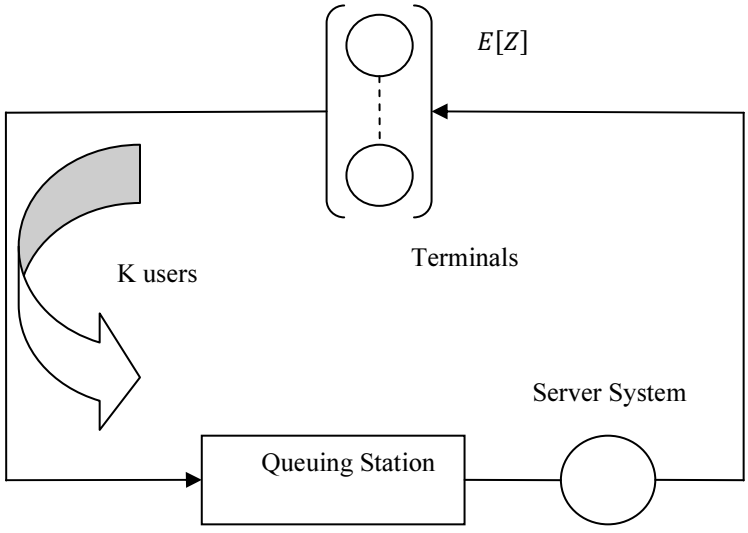

Figure 2.1 Simple Terminal Model

Before finding out current response time and average response time, it is necessary to introduce a concept which will be used to calculate response time, $M|M|1||K$ queue model (or terminal model, see Figure 2.1).

As introduced in Haverkort's book [12], in the $M|M|1||K$ queue 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 modeled as infinite servers; where each user has its 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, current response time and average response time can be found in terms of the terminal model.

In this case, a single server system was modelled, which means each type of server only serves one customer 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 customer 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:

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

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 customers actually present. For processing the server system, however, the average response time depends on the number of K customer.

To compute $E[R_s(K)]$, the 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 customer to go once through the cycle "think-serve". The throughput $X[K]$ can now be represented as $\frac{K}{E[C(K)]}$, 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:

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

From which we derive the *response time law*:

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

As we know, the throughput $X[K]$ equals the product of the server-busy probability and the service rate of the system, as $X[K] = (1 - p_0) \times \mu = \frac{1-p_0}{E[S]}$. 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:

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

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:

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

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 $T[R]$ and $E[R_s(K)]$. Overall, the rate of "decide" action may be expressed as:

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

if and only if

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

3 Modelling Techniques

3.1 PEPA

In this paper, the prototype model is created with PEPA. Performance Evaluation Process Algebra (PEPA) is a stochastic process algebra introduced by Jane Hillston in the 1990s [3] and designed for modeling computer and communication systems.

In PEPA, systems can be described as interactions of components. Such components represent the related entities in the system, and their interactions reflect

behaviours of relevant parts of the system. A component itself may be composed of more components. The specification of a PEPA activity consists of action type and action rate, represented as *(action type, rate)*, in which action type denotes the type of action, and the rate is drawn from negative exponential distribution.

PEPA has only four combinators, which are, *prefix*, *choice*, *cooperation* and *hiding*. Prefix 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 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. The cooperation operator joins two processes together, in which these two processes may share some actions: the process $P<a, b>Q$ *or* $P \bowtie_L Q$, $L = \{a, b\}$ denotes that processes *P* and *Q* must cooperate with the shared actions *a* and *b*, while any other action is performed independently. Hiding: the process $P/\{a\}$ hides the action *a* from view and prevents other processes from joining with it.

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

$$P \stackrel{def}{=} (a, \lambda).P \mid P + Q \mid P <L> Q \mid P/L \mid A$$

This PEPA statement involves all four combinators mentioned in the previous paragraph. The last part of this statement $P \stackrel{def}{=} A$ is to identify component *P* with *A*. When the rate of the action is passive, we use the symbol $\top$ or *infty* is used.

PEPA was chosen here as the main modelling language owing to several of its attractive features; which are, its compositionality, formality, and abstraction. Its compositionality can facilitate the model disintegration and model composition and provide the modeller with enough flexibility in the modelling process. Formality guarantees the precision of the model and preserves its semantics even taking different analysis or evaluation techniques. Abstraction makes the modelling process more flexible, which means components of PEPA models can be decomposed into more detailed components, or amalgamated to form a new one. More features and explanations of PEPA are given in Hillston's paper [3].

3.2 CMDL

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.

The reason for the choice of the CMDL model in this case study lies in the custom reaction rate. PEPA Plug-in cannot support the analysis with a varying action rate, while the CMDL model analysis has the ability to perform such a varying reaction rate. As the PEPA model can be transformed into a CMDL model, dynamic scheduling policy can be modeled by employing a function rate which varies with the change of system state.

Moreover, it is convenient to transform a PEPA model into a CMDL model via Eclipse PEPA Plug-in.

4 Model Constructions

According to the preceding model specification, the updated model has three patient classes: PatientB (blood only), PatientX (X-ray only) and PatientBX (both blood and X-ray). Each group of patients has its own arrival rate. Group PatientB will be sent to the blood test directly and the PatientX group will be directed to the X-ray test on arrival. For group PatientBX, the system will check the current situation of blood and x-ray departments, and then decide which department the present patient should be sent to. The detailed PEPA model states:

$$
\begin{aligned}
\text{PatientB} &\stackrel{\text{def}}{=} (\text{arriveB}, \text{r\_arrive\_b}).\text{PatientB1};\\
\text{PatientB1} &\stackrel{\text{def}}{=} (\text{register}, \text{r\_register}).\text{PatientB2};\\
\text{PatientB2} &\stackrel{\text{def}}{=} (\text{blood}, \text{r\_blood}).\text{PatientB3};\\
\text{PatientB3} &\stackrel{\text{def}}{=} (\text{wait}, \text{r\_wait}).\text{PatientB};\\
\\
\text{PatientX} &\stackrel{\text{def}}{=} (\text{arriveX}, \text{r\_arrive\_x}).\text{PatientX1};\\
\text{PatientX1} &\stackrel{\text{def}}{=} (\text{register}, \text{r\_register}).\text{PatientX2};\\
\text{PatientX2} &\stackrel{\text{def}}{=} (\text{xray}, \text{r\_xray}).\text{PatientX3};\\
\text{PatientX3} &\stackrel{\text{def}}{=} (\text{wait}, \text{r\_wait}).\text{PatientX};\\
\\
\text{PatientBX} &\stackrel{\text{def}}{=} (\text{arriveBX}, \text{r\_arrive\_bx}).\text{PatientBX1};\\
\text{PatientBX1} &\stackrel{\text{def}}{=} (\text{register}, \text{r\_register}).\text{PatientBX2};\\
\text{PatientBX2} &\stackrel{\text{def}}{=} (\text{decideB},\text{T}).\text{PatientBX3} + (\text{decideX}, \text{T}).\text{PatientBX4};\\
\text{PatientBX3} &\stackrel{\text{def}}{=} (\text{blood}, \text{r\_blood}).\text{PatientBX5};\\
\text{PatientBX5} &\stackrel{\text{def}}{=} (\text{xray}, \text{r\_xray}).\text{PatientBX6};\\
\text{PatientBX6} &\stackrel{\text{def}}{=} (\text{wait}, \text{r\_wait}).\text{PatientBX};\\
\text{PatientBX4} &\stackrel{\text{def}}{=} (\text{xray}, \text{r\_xray}).\text{PatientBX7};\\
\text{PatientBX7} &\stackrel{\text{def}}{=} (\text{blood}, \text{r\_blood}).\text{PatientBX8};\\
\text{PatientBX8} &\stackrel{\text{def}}{=} (\text{wait}, \text{r\_wait}).\text{PatientBX};\\
\\
\text{Blood} &\stackrel{\text{def}}{=} (\text{blood}, \text{r\_blood}).\text{Blood};\\
\text{Xray} &\stackrel{\text{def}}{=} (\text{xray}, \text{r\_xray}).\text{Xray};\\
\text{Reception} &\stackrel{\text{def}}{=} (\text{register}, \text{r\_register}).\text{Reception};\\
\\
\text{DecideB} &\stackrel{\text{def}}{=} (\text{decideB}, \text{r\_db}).\text{DecideB};\\
\text{DecideX} &\stackrel{\text{def}}{=} (\text{decideX}, \text{r\_dx}).\text{DecideX};\\
\\
\text{Wait} &\stackrel{\text{def}}{=} (\text{wait}, \text{r\_wait}).\text{Wait};
\end{aligned}
$$

$$
\begin{aligned}
&\text{PatientB[N1]}||\text{PatientX[N2]}||\text{PatientBX[N3]} \bowtie_L\\
&\quad \text{Blood}||\text{Xray}||\text{Reception}||\text{DecideB}||\text{DecideX}||\text{Wait}\\
&\text{L = \{ register, blood, xray, decideB, decideX, wait \}}
\end{aligned}
$$

The above PEPA model clearly presents the updated hospital scenario. In addition, a periodic function is also applied for arrival rate and service rate to model the

varying system arrivals and service ability. Here, two trigonometric functions, sine and cosine, are introduced to fluctuate the arrival rate and service rate. In contrast to the dynamic scheduling, the static scheduling uses the same functional arrival and service rate.

5 Model Analyses

This section will analyze three scheduling policies in the new model scenario. Analysis focuses on the comparison between the 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 that come from the original PEPA model. The analysis places extra emphasis on a large number of customers. In this situation, the system keeps running to serve the 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 number of instances guarantee the accuracy of fluid flow analysis. 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 world situation, the following analysis sets up four condition groups based on arrival rate and service rate: stable arrival rate and service rate, varying arrival rate and stable service rate, stable arrival rate and varying service rate, and varying arrival rate and service rate. 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 sin() and cos() function are adopted for arrival rate and service rate to alter their values over time.

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 is the same for constant service rate which stands for the serving ability being kept at a constant mean value.

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. So, 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 a blood test is faster than an x-ray scan, so 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 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 expression which were introduced in the previous section, 2.2, and initialize them with suitable values.

Initialization for Static Scheduling Policy

$$Rate\ of\ decide\ for\ Blood: \quad r_{db} = 1.0$$
$$Rate\ of\ decide\ for\ Xray: \quad r_{dx} = 1.0$$

Here, the rate of decide for both Blood and Xray are initialized with a small constant value 1.0. This is because the decide action is defined with a passive action rate. This 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

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

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

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 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:

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

As per the model, K is the number of patients at each service component, such as Blood and Xray; $E[S]$ 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 the 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.

5.1.2 Analysis Results

As per the previous condition setup, the following figures were obtained from a single run under the condition that both arrival rate and service rate are constants.

Figure 5.1 displays the average queue length for an 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 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, 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.

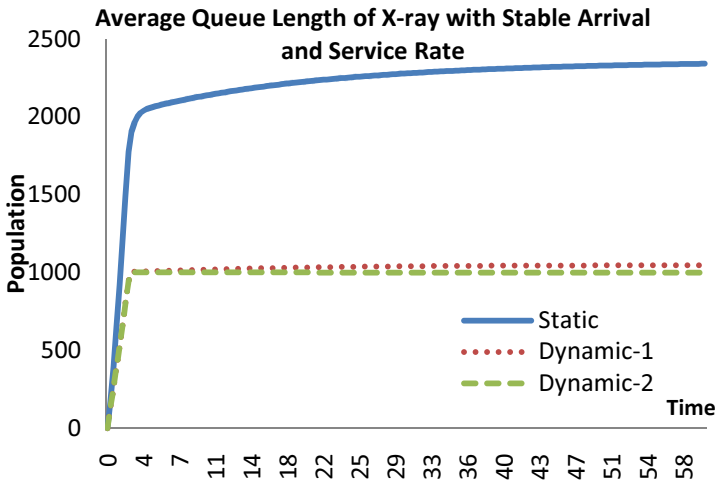

Figure 5.1 Average Queue Length of X-ray with Stable Arrival Rate and Service Rate

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* 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 static.

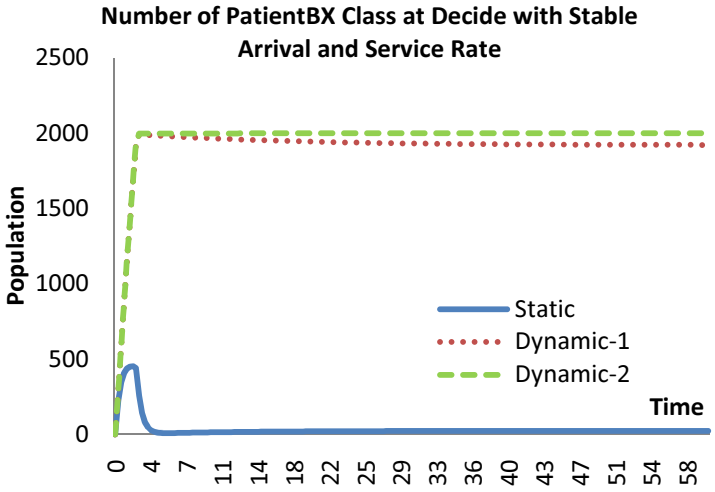

Figure 5.2 Population of PatientBX Waiting at Decide Component with Stable Arrival and Service Rate

Figure 5.2 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 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.

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 5.1.1. 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:

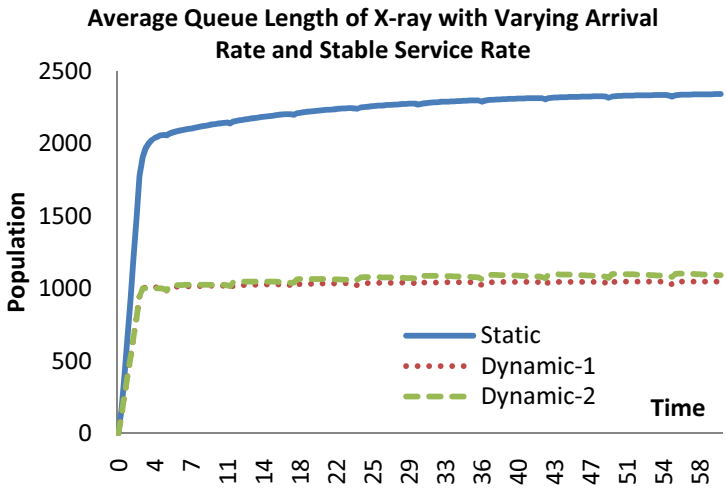

Figure 5.3.a Average Queue Length of X-ray with Varying Arrival Rate and Stable Service Rate

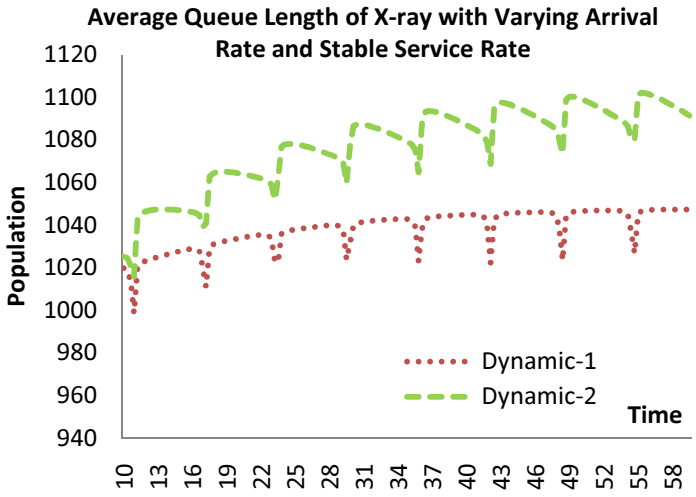

Figure 5.3.b Average Queue Length of X-ray with Varying Arrival Rate and Stable Service Rate (average queue length for Dynamic 1 and Dynamic 2, time: 10~60)

As can be seen from Figure 5.3.a, dynamic scheduling policy has far fewer patients in the queue waiting at x-ray in contrast to the static. In Figure 5.3.a, the queue length for static policy is about 2340, which is the same as that in Figure 5.1. However, the queue length for dynamic policy 1 and dynamic policy 2 are roughly 1050 and 1090 respectively. Figure 5.3.b shows 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 policy 2, as in 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 5.1, all lines in Figure 5.3.a have very small and periodical fluctuations, which are caused by the varying arrival rates.

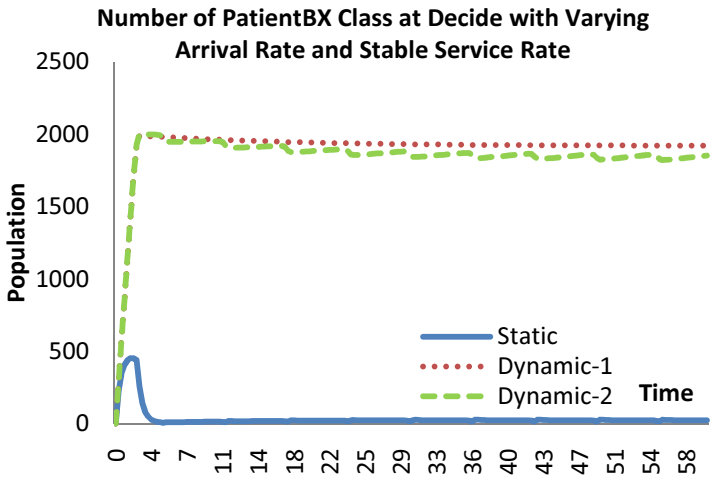

Figure 5.4 Population of PatientBX Waiting at Decide Component with Varying Arrival and Stable Service Rate

Figure 5.4 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 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.

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 5.1.1. 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 interesting results emerged from fluid ODE analysis. The figures are:

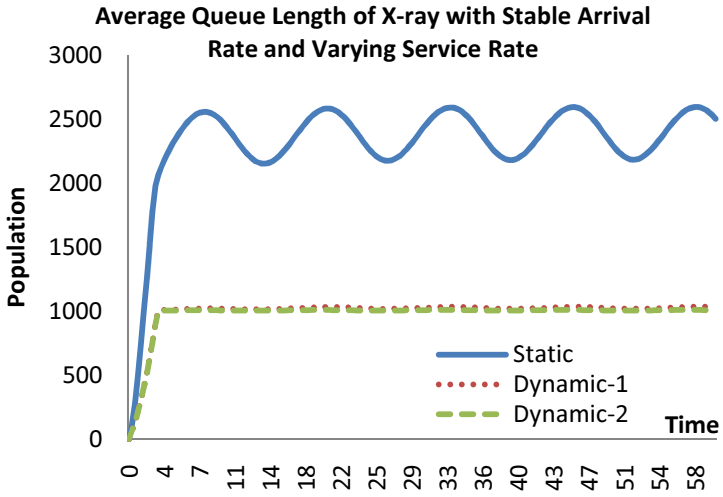

Figure 5.5.a Average Queue Length of X-ray with Stable Arrival Rate and Varying Service Rate

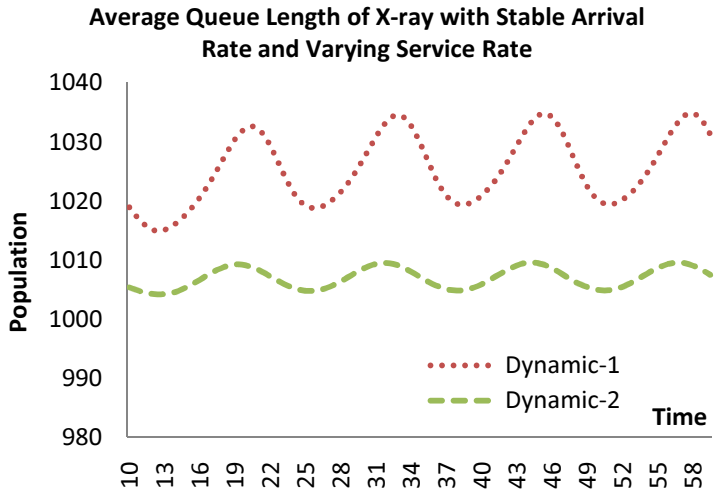

Figure 5.5.b Average Queue Length of X-ray with Stable Arrival Rate and Varying Service Rate (average queue length for Dynamic 1 and Dynamic 2, time: 10~60)

As displayed in Figure 5.5.a, static policy at X-ray component creates distinct waves between 2100 and 2600. However, dynamic policies appear quite smooth and steady. From Figure 5.5.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.

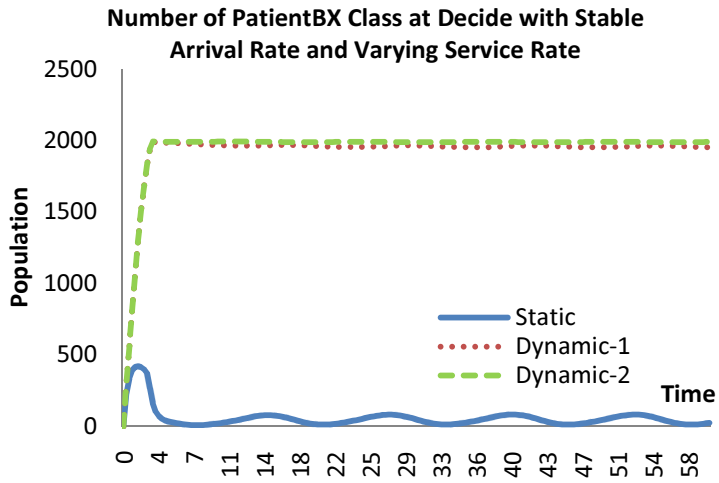

Figure 5.6 Population of PatientBX Waiting at Decide Component with Stable Arrival and Varying Service Rate

As per Figure 5.6, 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.

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 5.1.1 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:

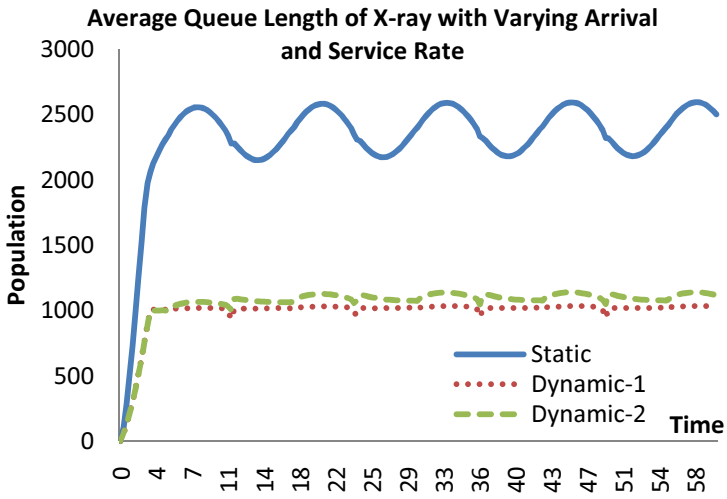

Figure 5.7 Average Queue Length of X-ray with Varying Arrival and Service Rate

Figure 5.7 depicts the average queue length of three scheduling policies at component X-ray. According to the figures, it is obvious that dynamic policies have a 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 5.7, 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 1 appears more smooth and stable than the green line for dynamic 2. However, the static policy is still the worst of all having an average queue length fluctuating between 2100 and 2600. In this figure, all lines have small sharp waves caused by the varying arrival rates, as appeared in section 5.2.

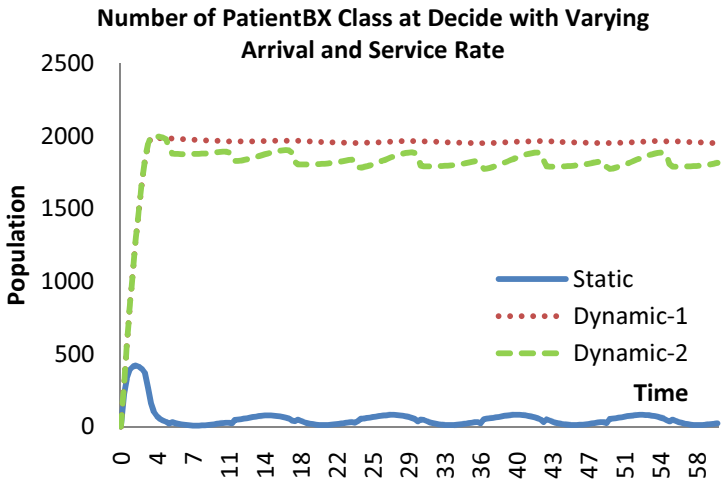

Figure 5.8 Population of PatientBX Waiting at Decide Component with Varying Arrival and Service Rate

Figure 5.8 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. The number of patients blocked by policy 1 is about 1950 on average. For dynamic policy 2, its line periodically undulates around 1800. However, the static policy prevents no more than 100 patients from entering hospital.

From the above figures, it can be declared that dynamic policies have a much more efficient and stable ability in scheduling than the static policy. Of the two dynamic policies, especially 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 much greater than service rate, dynamic policy 1 should be better than dynamic policy 2. Conversely, if service rate has a greater variation, dynamic policy 2 will yield a better performance.

6 Conclusions

This paper aims to explore the performance of dynamic scheduling policies by comparing these with static scheduling policy. The main method used for measurement is fluid flow approximation, solving a set of ordinary differential equations generated from an original PEPA model. The ODE analysis is carried out with an equivalent CMDL model converted from the PEPA model. Thus, we can conclude the performance of the three policies as follows:

- 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.

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

References

1. W. Mark, "The computer for the 21 century," SIGMOBILE Mob. Comput. Commun. Rev., vol.3, pp.3-11, 1999.
2. M. Harrison, M. Massink, D. Latella, "Engineering Human Flows in Smart Environments using Formal Techniques", Forth International Workshop on Practical Application of Stochastic Modelling, September, 2009.
3. J. Hillston, "A Compositional Approach to Performance Modelling", Cambridge University Press, 1996.
4. S. Gilmore and J. Hillston, "The PEPA Workbench: A Tool to Support a Process Algebra-based Approach to Performance Modelling", In Proceeding of the Seventh Interantional Conference on Modelling Techniques and Tools for Computer Performance Evaluation, number 794 in Lecture Notes in Computer Science, pp.353-368, Springer-Verlag, 1994.
5. M.Tribastone, "The PEPA Plug-in Project", in M. harchol-Balter, M. Kwiatowska, and M.Telek, editors, Proceedings of QEST'07, pp.53-54, IEEE Computer Society Press, 2007.
6. J. Hillston, "Fluid Flow Approximation of PEPA models", Proceeding of the Second International Conference on the Quantitative Evaluation of Sytems (QEST'05), 2005.
7. M. Calder, S. Gilmore, and J. Hillston, "Automatically deriving ODEs from process algebra models of signalling pathways", In Proceeding of CMSB'05, 2005.
8. J.Bradley, S. Gilmore, and N. Thomas, "Performance Analysis of Stochastic Process Algebra Models Using Stochastic Simulation", In Proceeding of 20th IEEE International Parallel and Distributed Processing Symposium, IEEE Computer Society, 2006.
9. 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, October 2006, pp. 60-67.
10. M. Cooke, J. Fisher, J.Dale, E.Mcleod, A.Szczepura, P.Walley, S.Wilson, "Reducing Attendences and Waits in Emergency Departments - A systematic review of present innovations", National Co-ordinating Centre for NHS Service Delivery and Organisation R & D (NCCSDO), January 2004.
11. X. Chen, and N. Thomas, "Performance Evaluation of Scheduling in a Smart Environment", 26th Annual UK Performance Engineering Workshop, July 2010.
12. R. Haverkort, "Performance of Computer Communication Systems – A Model-Based Approach", pp. 86-89, John Wiley & Sons Ltd, 1998.