AN AUTONOMIC COMPUTING FRAMEWORK FOR SELF-MANAGED EMERGENCY DEPARTMENTS

Serene Almomen and Daniel Menascé
Volgenau School of Information Technology and Engineering, George Mason University
4400 University Drive, Fairfax, VA 22030, U.S.A.
salmome1@gmu.edu, menasce@cs.gmu.edu

Keywords: Autonomic Computing, Healthcare, Emergency Department, Quality of Service, Self-Managed System.

Abstract: The delivery of cost-effective and quality Emergency Department (ED) services remains an important and ongoing challenge for the healthcare industry. ED overcrowding has become a common problem in hospitals around the world, threatening the safety of patients who rely on timely emergency treatment. Despite numerous advances in medical procedures and technologies, EDs continue to experience overcrowding problems. The combination of increased demand and diminished resources makes optimizing emergency departments a difficult problem for healthcare decision makers. We examine this problem by applying an autonomic computing framework for self-managed emergency departments to maintain optimal Quality of Service (QoS) during its operation. Our work has potential implications in guiding a hospital's effort to optimize their emergency department system.

1 INTRODUCTION

In hospitals all over the country, healthcare emergency departments (ED) are severely overcrowded resulting in delays in care, difficulty in providing quality care, patient discomfort and dissatisfaction, and higher service cost. In addition, overcrowding also leads to staff burnouts and inefficient utilization of resources. Many ED nurses leave for other departments or units as a result of getting overwhelmed with the ED workload (ACEP, 2010). The challenges of the ED, including overcrowding and boarding, have been a subject of a great deal of discussion. Several meetings, reports, and research studies have been conducted to understand the causes, implications and possible solutions to ED overcrowding and boarding issues. The ED is one of the most critical units in any healthcare organization. Consequently, improving performance of this unit is vital to the success of the healthcare organization.

Due to the dramatic increase in the cost of healthcare over the past few decades, researchers and healthcare professionals examined new ways to improve efficiency and at the same time reduce healthcare costs. Simulation tools have assisted healthcare decision-makers in this endeavour (Hashimoto & Bell, 2007). Another attempt to improve the ED system relies in capturing ED workflow patterns and analyzing these patterns to create an automated and enhanced ED system design (Moss & Xiao, 2004). Another approach discussed the use of workflow technologies and web services to automate emergency healthcare processes (Poulymenopoulou, Malamateniou, & Vassilacopoulos, 2008). That work discussed the need to provide an appropriate technological infrastructure for automating and managing the emergency healthcare processes in both intra- and inter-organizational services. The implementation of this approach involves capturing process logic requirements for healthcare workflow systems with a view to design a system that is easily adjustable to process changes and to evolving organizational structures. Some tools have also been developed for hospital capacity planning simulation to conduct both process flow analysis and capacity forecasting (Mengwasser & Berger, 2009).

The dynamic nature of the ED adds to the complexity of the problem. Sudden changes to the workload due to emergencies such as fire, natural disasters, and terrorist attacks are difficult, if not impossible, to predict.

Furthermore, the ED environment is complex in nature. ED systems are composed of a collection of

resources including both humans (e.g., doctors, nurses, and technicians) and equipments (e.g., X-ray machines and CT-Scan). ED systems also involve human processes and decision making where humans in the loop determine how the application evolves based on their awareness of the situation and infrastructure. Consequently, there is a need for self-managing EDs. We examine this problem by applying an autonomic computing framework for self-managed EDs to maintain optimal operational Quality of Service (QoS). Our work has potential implications in guiding a hospital's effort to optimize their emergency department systems.

The rest of the paper is organized as follows. Section 2 discusses background information on the proposed approach. Section 3 discusses the environment of EDs. The next section describes the autonomic computing framework for EDs. Section 5 discusses different implementation techniques for the autonomic framework. Finally, Section 6 presents some concluding remarks.

2 BACKGROUND

Computing systems have reached a level of complexity where traditional IT support that involves human effort to maintain the systems and keep them operational is becoming increasingly challenging. A similar problem was experienced in the 1920s in telephony before automatic branch exchanges were introduced to eliminate human intervention (Mainsah, 2002).

Autonomic computing seeks to enhance the performance or QoS and at the same time minimize human intervention. The autonomic computing paradigm has been inspired by the autonomic function of the human central nervous system (Kephart & Chess, 2003). It is the body's master controller that monitors changes inside and outside the body, integrates sensory inputs, and effects appropriate response (Ashby, 1960). Autonomic controls in the human body use motor neurons to send indirect messages to organs at a sub-conscious level. These messages regulate temperature, breathing, and heart rate without conscious thought (Ashby, 1960). The implications for computing are immediately evident; a network of organized, smart computing components that give us what we need, when we need it, without a conscious mental or even physical effort (IBM, 2010).

Autonomic computing attempts to intervene in computing systems in a similar fashion to its biological counterpart. There has been significant research to create autonomic systems. An example of such effort is IBM's MAPE-K (Monitor, Analyze, Plan, Execute, Knowledge) reference model (Huescher & McCann, 2008). A similar model is proposed by Russel and Norvig (2003) in which an intelligent agent monitors its environment through sensors and uses the collected data to determine actions to be performed in the environment (Russel & Norvig, 2003).

In either model, an autonomic system has a managed element (such as software or hardware resources), the organ in the human body, that is given an autonomic behaviour and an autonomic manager, the nervous system in the human body, that monitors the managed element and specifies actions to be executed by the managed element (Huescher & McCann, 2008) (Russel & Norvig, 2003). There have been several different implementations of the MAPE-K model including autonomic toolkit, ABLE, Kinesthetics eXtreme (KX), and self-management tightly coupled with application. The latter implementation is closely related to the proposed approach discussed in this paper. Such implementation involves either using an autonomic middleware framework that offers self-management properties to applications built on top of this middleware or through encapsulating tasks in components and defining self-management and adaptation in terms of these components (Huescher & McCann, 2008).

IBM also portrayed four fundamental properties of self-management: self-configuration, self-optimization, self-healing, and self-protection. Briefly, these properties mean that an autonomic computing system configures itself according to high-level goals, optimizes its use of resources, detects and diagnose problems, and protects itself against malicious attacks and end users who inadvertently make changes to the system components such as its software (Huescher & McCann, 2008). Consequently, any implementation of autonomic computing systems should realize these properties.

In addition, studies in the autonomic systems field describe approaches to plan the changes or actions to be effected on the managed element of an autonomic system. Some propose policy-based adapting planning, architectural models, or process-coordination approach (Sloman, 1994) (Wise, Cass, Lerner, Call, Osterweil, & Jr., 2000) (Huescher & McCann, 2008).

With recent advances in embedded computing, networking, and related information technologies, it is now feasible to deploy a variety of sensing

devices, communication networks and IT services in the real world. These physical spaces include a variety of sensors such as optical sensors, RFIDs, as well as specialized sensors such as people-counters and load-cells that enable monitoring the state of the physical world and its activities. These sensors are connected to communication networks such as Ethernet, cellular, Bluetooth, and WiFi. (Kim, et al., 2008). These sensors provide a mechanism to monitor the different resources of a system. Such technology makes it even easier to implement autonomic computing systems in real world environments.

Another concept that can facilitate the implementation of autonomic systems in real world environments is *utility functions*. Utility functions express the usefulness of a system to one or more stakeholders as a function of the attributes of a system. The concept of utility is one of the methods used to represent Knowledge in autonomic systems. A utility function is written as follows:

$$U = f(x_1, x_2, ..., x_n) \quad (1)$$

where x_1, …, x_n are attributes and the function f combines these attributes in way that expresses the usefulness of a system as a function of these attributes. In general, utility functions are normalized in the [0,1] range with zero representing the lowest utility and one representing the highest utility. It is generally easier to specify a utility function as a function of several utility functions, one for each attribute. An example would be where

$$U = \sum_{i=1}^{n} w_i U_i(x_i) \quad (2)$$

the global utility function is a weighted sum of all the individual utility functions.

Autonomic computing systems use utility functions as the goal to be optimized. The attributes in this case are several Quality of Service (QoS) metric of interest such as response time, throughput, and availability of the computing resources (Menasce, Bennani, & Ruan, 2005). As failures and performance degradations occur, the autonomic computing system automatically changes its configuration parameters in a way that maximizes the utility function for the system. Consequently, the utility function of an autonomic system can be written as follows:

$$U = f(QoS_1, QoS_2, ...QoS_n) \quad (3)$$

where both the utility function and QoS metrics are defined by domain experts.

3 EMERGENCY DEPARTMENT ENVIRONMENT

We interviewed a Director of Emergency Services and a staff nurse at a Pediatric Emergency Department to gain a better understanding of the ED environment. Our findings are summarized in what follows.

The ultimate goal of an ED is patient satisfaction, which is normally measured by the length of stay at the ED. ED length of stay is the patient time in the ED as follows (Welch, Augustine, Camargo, & Reese, 2006):

- For admitted patient: arrival time to conversion time
- For discharged patients: arrival time to discharge time
- For transferred patients: arrival time to transfer conversion time

Many QoS metrics are collected and analyzed at EDs to determine areas of improvement that are necessary to meet this goal. These QoS metrics can be *time measures* or *proportion measures* (Welch, Augustine, Camargo, & Reese, 2006). The time measures include arrival time to first seen by a doctor, doctor to discharge time, doctor to decision to admit time, arrival time to rooming, disposition to discharge, and many others. The proportion measures include number of patients who left before they were supposed to, complaints, hospital diversion, and ED patient flow to name a few. These measures are commonly referred to as *Core Measures* and are often specified by a healthcare governing body serving local or nationwide hospitals. These Core Measures data are saved in an advanced analytical tool where comparative reports taking into account national averages can then be generated allowing hospitals to proactively assess performance and identify opportunities for quality improvement including potentially preventable readmissions and complications. These Core Measures also depend on several Census and utilization metrics including pediatric patients per day, high-acuity patients per day, number of self-paid patients, medication doses administered per 100 patients seen, and service hours per day of physicians.

There are many critical issues in EDs that contribute to an unsatisfactory level of these QoS metrics, many of which are demography-dependent. Inadequate patient beds always lead to long arrival to discharge time for example. The lack of an ultrasound machine may also be critical in an ED

that expects many pregnancy-related emergencies. This is an example of a demography-related issue. Other issues include lack of outpatient psychiatric service and lack of outpatient programs for referral. However, we found that lack or resources, specifically nurse shortage, is a major contributor to not meeting the overall goal of quick patient turnaround.

There are compelling reasons to collect and control ED QoS metrics. Regulatory burdens, ED operations management, and ED body of knowledge expansion are some (Welch, Augustine, Camargo, & Reese, 2006). The Joint Commission on Accreditation of Healthcare Organizations (JCAHO), for example, is pursuing clinical quality improvement (QI) data in the form of Core Measures. Any facility that does not have in place the infrastructure to track these data risks its accreditation. In addition, to determine whether ED process innovations are effective, quality measures will be required. To date, much QI work goes unpublished, and therefore ED QI workers are failing to build a body of research that is pertinent to operational efficiency.

In an effort to control QoS metrics within an acceptable range, hospitals in collaboration with nurse and the physician groups document their ED standard processes, treatment protocols, and regulations and orient new staff on them. The processes take into account national averages of ED patients' length of stay and try to stay within or below that range. In addition, these processes are evaluated often to accommodate technology changes, equipment increase, and changes in practices. Pilots and time audits are sometimes used to determine the compliance to those processes and protocols.

To maximize efficiency at the ED, the Emergency Severity Index (ESI) to classify patients coming into the ED is used. ESI is a five-level emergency department (ED) triage algorithm that provides clinically relevant stratification of patients into five groups from 1 (most urgent) to 5 (least urgent) on the basis of acuity and resource needs. The Agency for Healthcare Research and Quality (AHRQ) funded initial work on the ESI (AHRQ, 2010).

A *triage nurse* is responsible for ESI level assignment to ED patients. The ESI level determines the waiting time of patients. ESI level 1 patients have no waiting time for example. In addition, EDs are divided into care areas or zones based on the severity of the case treated. Consequently, the ESI level also determines the zone the patient will occupy. Level 4-5 patients are often assigned to the 'fast track' zone since they usually do not require many resources before they can be discharged. The triage nurse is also responsible for zone assignments.

A *charge nurse* uses the ESI level and zone assignment for each patient in the ED to determine the most efficient workflow of the ED. The charge nurse's role, consequently, is to run the ED as efficiently as possible to help minimize patients' length of stay. In the EDs we visited, the charge nurse uses a computer application that collects the QoS metrics as well as patient status to assist in making decisions on the most efficient workflow at the ED at any given time. These decisions may include changing a nurse's assignment to balance the workload of ED nurses, task a nurse to dispense medication if a doctor's order is ready, start the hospital admission process for a patient after doctor's diagnosis is complete, and request an on-call nurse to come to the ED if the workload is high. The *staff nurses* are assigned patients and take care of patients once they are in a room at the ED.

It is also important to note that there is no cost constraint that limits the operation of an ED. In other words, no patient will ever be turned away because of insufficient resources. EDs normally have working agreements with other hospitals in their area where they can quickly transfer patients to due to heavy workload or resource shortage. EDs also use what they call a 'float pool' of nurses who can be used to staff EDs if needed. Nevertheless, maximizing the efficiency of an ED does reduce the cost of service.

4 AN AUTONOMIC COMPUTING FRAMEWORK FOR ED

Our framework involves developing an autonomic computing system or a self-managed ED system that can regulate and maintain itself without human intervention. This is ideal in an ED environment since the goal is to create a system that will be able to adapt to a constantly changing environment (such as patient flow, workload, and resource availability) in a way that preserves given operational goals (such as performance goals or QoS goals).

To achieve that, the proposed approach attempts to implement the MAPE-K (Kephart & Chess, 2003) model in an ED environment. Within the ED context, autonomic managers define a control loop (MAPE-K loop), as shown in Figure 1. Changes are made through action operations. Sensors look at the

state of the managed ED resources, and action operations can change the current state. The entire ED environment is a set of managed resources. Autonomic managers, just like a charge nurse, continuously monitor the system and handle events that need action to be taken. They monitor the ED environment using inputs from the sensors installed in the environment and analyze what is found. Based on the defined utility function of the ED, the autonomic manager then plans and executes any specific actions needed to maximize the utility function. The steps of monitoring, analyzing, planning, and executing may be executed concurrently. For example, if the X-ray machine in an ED is experiencing high utilization, the system could decide to provision an additional machine. The system can then return to monitoring, and if utilization drops, the X-ray machine can be deprovisioned and made available to other departments in the hospital.

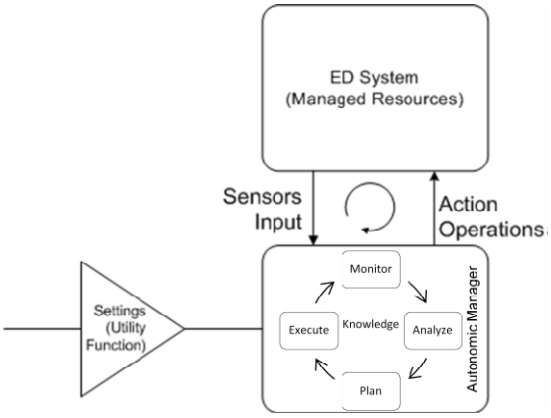

Figure 1: Autonomic Computing Model for ED.

The ED system depicted in Figure 1 can have a variety of architectures. It can consist of integrated applications or components within the ED such as the pharmacy dispensing application, the scheduling application, and the resource allocation application. We also consider a cyber-physical ED system as including smart devices such as patient wrest sensors, nurse PDAs, bed sensors, and other devices.

Our approach to self-managed ED systems does not address the various ED system architectures. Rather, it assumes a cutting edge system such as a cyber-physical environment that collects data or metrics through different sensors and devices to send to the autonomic manager. Contrary to common systems in the autonomic computing environments, the ED system in the diagram consists of human resources in addition to the hardware and software resources. This means that doctors and nurses are considered resources of the system as well as CT-Scan, X-ray machines, bed sensors, and pharmacy dispensing application for example.

The ED autonomic manager attempts to optimize pertinent QoS metrics. One of the metrics, called wait time ratio and defined as W/T, combines the average length of stay (T) with the average time (W) spent by the patient in the waiting area. Other relevant metrics are patient throughput (X_0), and resource utilization (U_i) for resource i.

The values of these QoS metrics depend on several of the census and utilization parameters discussed in section 3 which can be categorized as workload intensity parameters (e.g., the arrival rate of patients of a given group) and the service demands parameters of each group at each resource (e.g., the average time spent by a patient using the CT-Scan).

The goal of the ED autonomic system is to find settings of the managed resources that optimize a given utility function provided by the domain expert, which depends on the values of several QoS metrics. This is important in order to realize the self-optimization property of an autonomic system. The autonomic manager uses the provided utility function to plan appropriate changes or actions to be effected on the managed resources of the ED. As mentioned before, utility function may be obtained by combining utility functions for the different QoS metrics, such as:

- Utility function for the throughput of the ED: $U_X(X_{ED})$
- Utility function for the average length of stay, T, in the ED: $U_T(T)$
- Utility function for the average time spent by a patient waiting in the ED: $U_W(W)$
- Utility function for the wait time ratio W/T: $U_{W/T}(W/T)$

As an example, Figure 2 shows two utility functions: one for $U_{W/T}(W/T)$ in Figure 2(a) and the other for $U_X(X_{ED})$ in Figure 2(b). A global utility function U_g is a function of the individual utility functions $U_X(X_{ED})$, $U_T(T)$, $U_W(W)$, and $U_{W/T}(W/T)$:

$$U_g = f(U_X(X_{ED}), U_T(T), U_W(W), U_{W/T}(W/T)). \quad (4)$$

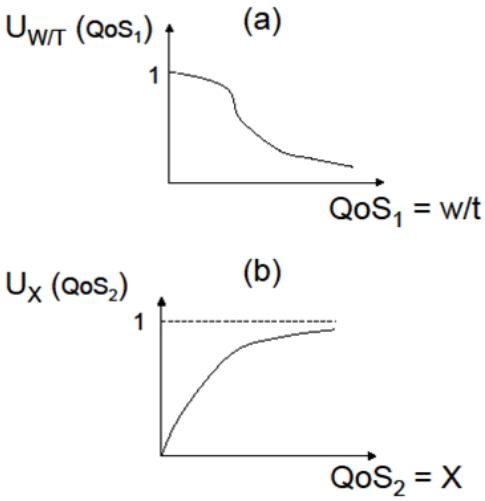

Figure 2: Example of Utility Functions.

5 IMPLEMENTATION TECHNIQUES

There are two main approaches to implementing autonomic computing systems; model-driven based on system performance models and data-driven based on reinforcement learning. In what follows, we describe the two approaches to autonomic computing implementation and illustrate how one model-driven technique, the queuing-theoretic model, can be applied in the context of the ED.

5.1 Model-driven Approach

The model-driven approach relies on being able to build models that can be used to predict the values of a system's QoS metrics as a function of its configuration parameters and resource sharing policies. Parameter and policy optimization techniques need to be defined in this approach. The parameter and policy optimization techniques map system states to action operations, hence, are used to plan the changes to the autonomic system to maximize its utility function (Gracanin, Bohner, & Hinchey, 2004).

Model-driven approaches focus on algorithms that make use of explicit system performance models such as queuing-theoretic or control-theoretic models.

Using a model-driven approach to autonomic computing makes it possible to generate run-time models that reflect the current state of the system without the unnecessary dependency on the system platform (Rohr, Boskovic, Giesecke, & Hasselbring, 2006). This capability makes this approach possible to adapt in implementing dynamic and complex autonomic systems such as the ED system. However, the design and implementation of accurate performance models of complex computing systems can be highly knowledge-intensive and labour-intensive and may require original research (Tesauro, Jong, Das, & Bennani, 2006).

5.2 Data-driven Approach

The data-driven approach focuses on knowledge-free trial-and-error methodology in which a learner tries various actions in numerous system states, and learns from the consequences of each action (Tesauro, Jong, Das, & Bennani, 2006). This approach is also referred to as Reinforcement Learning (RL). RL has successful applications in Markov Decision Process (MDP) in which RL can potentially learn decision-theoretic optimal policy in dynamic environments where the effects of actions are stationary and history-independent. RL has successful implementation in real-world problems such as helicopter control and financial markets trading (Tesauro, Jong, Das, & Bennani, 2006).

Contrary to the model-driven approach, RL does not require an explicit model of the computing system. In addition, RL has the capability to properly react to dynamical phenomena in an environment due to its roots in sequential decision theory. Other methods tend to treat dynamical effects only approximately or ignore them all together, or deal with the decision making problem as a series of unrelated instantaneous optimization (Tesauro, Jong, Das, & Bennani, 2006). Thus, RL is a possible implementation approach in autonomic computing.

The use of RL in real-world applications such as an ED system, however, can suffer from poor scalability in large state spaces since a lookup table is used to store a separate value for every possible state-action pair. The size of such a table increases exponentially with the number of state variables of the system making it challenging to use in real applications (Tesauro, Jong, Das, & Bennani, 2006). In addition, poor performance in live systems implementing this approach can be observed due to the long learning periods that may be necessary.

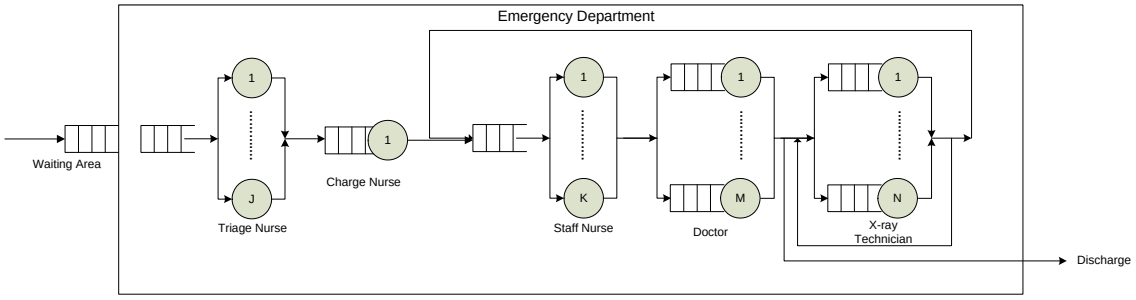

Figure 3: Example of ED QN Model.

5.3 Model-driven Implementation in the ED

The work by Menascé and Bennani (2003) describes creating autonomic computing models for computer systems based on the notion of queuing network (QN) models (Menascé & Bennani, 2003). Much like computer systems, an ED has many shared resources (e.g., beds, X-ray machines, doctors, and nurses). The performance of such a system can, hence, be conveniently represented by a Queuing Network (QN) model that represents the flow of patients and their contention for these resources. When a patient arrives at the ED for treatment, the patient alternates using the different resources in the ED such as the nurse, bed, and X-ray machine, quite likely more than once. At any point in time, a patient can be treated by a nurse and another using the X-ray machine, while other patients are waiting to be treated by the nurse or use the X-ray machine. Thus, the nurse and X-ray machine can each be characterized as a queue with a waiting line. An example of a QN model of an ED system is illustrated in Figure 3. In the diagram, the different resources of the system are represented by queues. This is a common representation in computer systems that our approach may adapt for self-managed ED systems including humans as resources. For example, nurses, doctors, and X-ray technicians are resources used by patients who flow through the system as indicated in Figure 3. The flow of a patient from one resource to another gives this model the network nature.

Some parameter and policy optimization approaches commonly used for QN models are hill climbing and beam-search algorithm among others. These approaches are referred to as combinatorial search techniques. The use of exhaustive searches of all possible configurations of the ED system is not feasible due to the complexity of such systems. Consequently, using a combinatorial search technique will find a close-to-optimal configuration so that the utility function of the new configuration is as close as possible to the desired QoS level (Babaoglu, et al., 2005) (Menasce, Bennani, & Ruan, 2005). In this case, the ED autonomic manager will use combinatorial search techniques such as hill-climbing to find the close-to-optimal configuration. A state space represents possible configurations of the system, as shown in Figure 4. Each point in the space represents a configuration of controlled parameters and the numerical value associated with each point represents the value of the utility function.

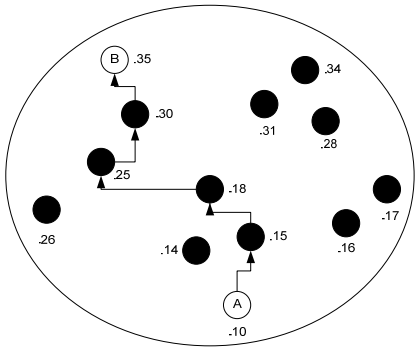

Figure 4: Example of State Space Search.

The figure shows that the current configuration is point A with value .10, which is obtained by computing the utility function using the measurements obtained from the sensors in the ED system. Through hill-climbing search, all 'neighbour' configurations are examined and a new configuration with the highest value of the utility function is selected. The search is repeated at each new point visited until either the value of the QoS does not improve, or a threshold on the number of points traversed has been exceeded. Through hill-climbing, a new close-to-optimal configuration, point B with value .35, is found. The value of any point in the search space can be computed through the use of QN models that can predict the value of the ED QoS metrics for configurations different than

the current one. The QoS values are then used to compute the value of the utility function for that point in the search space.

The configuration parameters for each point in the space can be represented as $v=(v_1, \ldots, v_m)$. As an example, in the ED, possible configuration parameters to be changed by the autonomic controller are: the maximum number of doctors (M_D), the maximum number of nurses (M_N), the queuing discipline at the X-ray machine ($d_{X\text{-}ray}$), and the queuing disciplines at the CT-Scan ($d_{CT\text{-}Scan}$). Thus, the configuration point can be defined as $v=(M_D, M_N, d_{X\text{-}ray}, d_{CT\text{-}Scan})$. Based on these parameters, the combinatorial search technique of choice will start at the current configuration C_0, examine all the neighbour configurations to C_0 and move to the one with the highest QoS value. A neighbour configuration is defined as one in which the parameters values of M_D, and M_N changes by ± 1 and the parameter values of $d_{X\text{-}ray}$, and $d_{CT\text{-}Scan}$ changes from First Come First Served (FCFS) or Priority Queuing, for example. The search is repeated at each new point until either the value of utility function does not improve, or a threshold on the number of points traversed has been exceeded.

The autonomic manager will then send an action operation comprised of the new optimal configuration to the ED system to change the ED's current configuration in order to achieve improved operations and decreased service cost.

6 CONCLUDING REMARKS

Inspired by biology, autonomic computing has evolved as a discipline to create software systems and applications that self-manage in an attempt to overcome the complexities and inability to maintain current and emerging systems effectively.

The implementation of autonomic computing has been increasingly emerging in fields including power management, data centers, clusters, and GRID computing systems, and ubiquitous computing. These applications are already demonstrating their feasibility and value (Huescher & McCann, 2008). However, there is no evidence that autonomic computing has been implemented in healthcare Emergency Department (ED) systems where not only hardware and software comprise the system resources, but also human beings. Our framework extends the autonomic computing concepts to create a self-managed ED system to reduce the dependency on human intervention to maintain such a complex system, thus, improve the ED operations, and decrease the ED service cost.

We are currently in the process of investigating a live implementation of the framework proposed in this paper in an ED environment. Specifically, we plan to compare the results on an ED with and without an autonomic computing system in order to investigate whether performance and cost improvements can be obtained.

REFERENCES

ACEP, 2010. *Meeting the Challenge of Emergency Department Overcrowding/Boarding.* Washington: American College of Emergency Physicians.

AHRQ. (2010). *Emergency Severity Index, Version 4*. Retrieved October 20, 2010, from Agency for Healthcare Research and Quality: http:// www.ahrq.gov/research/esi/esi1.htm

Ashby, W. R., 1960. *Design for a Brain.* Chapman & Hall Ltd.

Babaoglu, O., Jelasity, M., Montresor, A., Fetzer, C., Leonardi, S., Moorsel, A. v., et al., 2005. *Self-Star Properties in Complex Information Systems.* Springer Verlag: Lecture Notes in Computer Science.

Gracanin, D., Bohner, S. A., & Hinchey, M., 2004. Towards a Model-Driven Architecture for Autonomic Systems. *Proc. 11th IEEE Intl. Conf. Engineering of Computer-Based Systems*, (pp. 500-505).

Hashimoto, F., & Bell, S., 2007. Improving Outpatient Clinic Staffing and Scheduling with Computer Simulation. *Journal of General Internal Medicine* , 182-184.

Huescher, M. C., & McCann, J. A., 2008. *A survey of Autonomic Computing—Degrees, Models,and Applications.* New York: ACM.

IBM, 2010. *Autonomic Computing - The Solution.* Retrieved June 30, 2010, from Autonomic Computing: http://www.research.ibm.com/autonomic/index.html

Kephart, J., & Chess, D. (2003). The Vision of Autonomic Computing. *IEEE Internet Computing* , 41-50.

Kim, M., Massaguer, D., Dutt, N., Mehrotra, S., Ren, S., Stehr, M.-O., et al. (2008). *A Semantic Framework for Reconfiguration of Instrumented Cyber Physical Spaces.* St. Louis: IEEE.

Mainsah, E., 2002. Autonomic computing: The next era of computing. *Electronic Communications Engineering Journal* , 14, 1, 2–3.

Menascé, D. A., & Bennani, M. N., 2003. On the Use of Performance Models to Design Self-Managing Computer Systems. *Computer Measurement Group Conference.* Dallas: Computer Measurement Group.

Menascé, D. A., Bennani, M. N., & Ruan, H., 2005. *On the Use of Online Analytic Performance Models in Self-Managing and Self-Organizing Computer Systems.* Springer Verlag: Lecture Notes in Computer Science.

Menascé, D., Almeida, V., & Dowdy, L., 2004. *Performance by Design: Computer Capacity Planning by Example.* Upper Saddle River: Pearson Education, Inc.

Mengwasser, M. J., & Berger, M. A., 2009, February. Hospital Capacity Planning Model. *Noblis Sponsered Research Projects* , pp. 20-21.

Moss, J., & Xiao, Y., 2004. Improving Operating Room Coordination: Communication Pattern Assessment. *The Journal of Nursing Administration* , 93-100.

Poulymenopoulou, M., Malamateniou, F., & Vassilacopoulos, G., 2008. *Emergency healthcare process automation using workflow technology and web services.* Athens: Informatics for Health and Social Care.

Rohr, M., Boskovic, M., Giesecke, S., & Hasselbring, W., 2006. Model-driven Development of Self-managing Software Systems. *Proc. 9th Intl. Conf. Model-Driven Engineering Languages and Systems.* Springer.

Russel, S., & Norvig, P., 2003. *Artificial Intelligence: A Modern Approach 2nd Ed.* Prentice Hall.

Sloman, M., 1994. *Policy driven management for distributed systems.* J. Netw. Syst. Manag.

Tesauro, G., Jong, N. K., Das, R., & Bennani, M. N., 2006. A Hybrid Reinforcement Learning Approach to Autonomic Resource Allocation. *3rd IEEE International Conference on Autonomic Computing (ICAC)*, (pp. 65-73). Dublin, Ireland.

Welch, S., Augustine, J., Camargo, C. A., & Reese, C. (2006). Emergency Department Performance Measures and Benchmarking Summit. *Academic Emergency Medecine* , 1074–1080.

Wise, A., Cass, A. G., Lerner, B. S., Call, E. K., Osterweil, L. J., & Jr., S. M., 2000. Using Little-JIL to coordinate agents in software engineering. *Automated Software Engineering.* ASE.