Developing Service-based Software Systems with QoS Monitoring and Adaptation

S. S. Yau, N. Ye, H. Sarjoughian and D. Huang
Arizona State University, Tempe, AZ 85287-8809, USA
{yau,nongye, hessem.sarjoughian, dazhi.huang}@asu.edu

Abstract

The rapid adoption of SOA in many large-scale distributed applications requires the development of adaptive service-based software systems (ASBS), which have the capability of monitoring the changing system status, analyzing and controlling tradeoffs among multiple QoS features, and adapting its service configuration to satisfy multiple QoS requirements simultaneously. In this paper, a performance-model-oriented approach to developing ASBS is presented. This approach consists of the establishment of performance models for SBS through controlled experiments, the development of QoS monitoring and adaptation (M/A) *modules based on the performance models, and the validation of ASBS design through simulations. In our approach, four QoS features: timeliness, throughput, accuracy and security, which are important for many critical applications, are considered.*

Keyword: Adaptive service-based software systems, QoS monitoring, QoS adaptation, performance modeling, SOA-based simulation

1. Introduction

Recent development of service-oriented computing and grid computing has led to rapid adoption of Service-Oriented Architecture (SOA) in distributed computing systems, such as enterprise computing infrastructures, grid-enabled applications, and global information systems. One of the most important advantages of SOA is the capability that enables the rapid composition of the needed services provided by various service providers through networks for distributed applications and integration of the "system of systems". Software systems based on SOA are called *service-based software systems (SBS).* Late binding with services is a fundamental characteristic of SOA, and facilitates adaptation of SBS in run-time.

Fundamental changes to current software engineering techniques are needed for designing high-quality SBS due to the unique characteristics of SBS. A major problem to achieve this goal is how to manage the quality of service (QoS) of SBS, which may demand satisfaction of multiple QoS requirements simultaneously, such as timeliness, throughput, accuracy, security, dependability, survivability and availability. Because the satisfaction of requirements in one QoS feature often requires certain sacrifice in other QoS features, tradeoffs of requirements among multiple QoS features must be taken into account in the design of SBS. However, existing software engineering techniques do not support the analysis and adaptive control of such tradeoffs due to lack of comprehensive understanding of such tradeoffs and their relations to service operations on computers and networks. Furthermore, SBS often comprise services owned by various providers. Each service may support various service configurations, each of which defines the runtime properties of the service, such as authentication mechanism, priority, and maximum bandwidth reserved for the service. Different service configurations will result in different QoS in runtime. The services in SBS often operate in highly dynamic environments, where the services may become temporarily unavailable due to various system and network failures, overloads or other causes. Hence, high-quality SBS need to have the capability to monitor the changing system status, analyze and control tradeoffs among multiple QoS features, and adapt their service configurations accordingly. Such SBS are referred as *Adaptive SBS (ASBS)*.

The conceptual view of ASBS is depicted in Figure 1, in which functional services used to compose the ASBS and the modules for QoS M/A form a closed control loop [1]. The QoS monitoring modules collect the measurements of various QoS features as

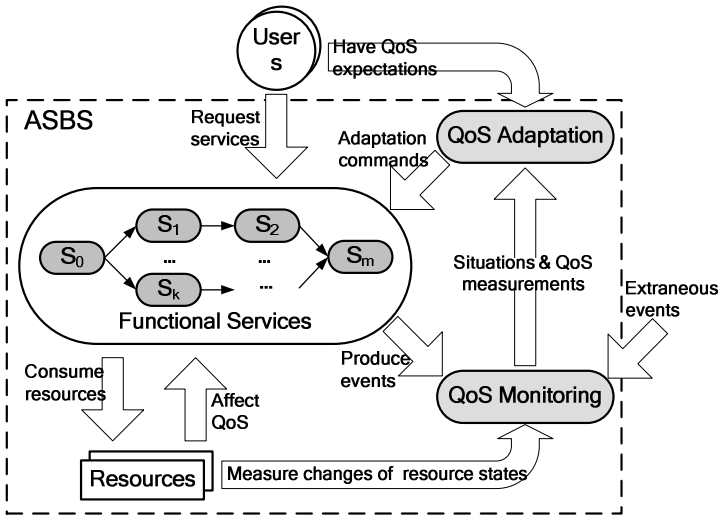

Figure 1. A conceptual view of ASBS

well as system status concerning QoS. Based on the system status and QoS measurements, decisions on QoS adaptation are made to adjust the configurations and service operations of ASBS to satisfy various QoS requirements simultaneously. In this paper, we will present a performance-model-oriented approach to developing ASBS. Our approach includes the performance models for SBS based on controlled experiments, development of QoS M/A modules based on the performance models, and validation of ASBS design through simulations. In our approach, we consider four QoS features: timeliness, throughput, accuracy and security, which are important for many critical applications.

2. Related Work

Currently, software design is usually based on logic-based operational models. However, performance models linking service operations to resource states and QoS performance are needed for QoS monitoring and adaptation in ASBS. Individual QoS features related to various activities and resource states in computing and communication systems have been investigated both empirically and theoretically [2-4]. However, how system activities and resource states affect the tradeoffs among various QoS features has not been thoroughly investigated [5, 6]. The lack of such essential knowledge makes it difficult to design ASBS.

To provide a cost-effective way for validating the design of an ASBS with QoS M/A capabilities to meet the QoS requirements for the ASBS, it is necessary to have proper modeling and simulation support for ASBS, which cannot be provided by existing simulation approaches [7-12]. The simulation environment DEVS/DOC [7], which supports distributed object computing concepts, may be used to design distributed software systems, but it is not suitable for ASBS due to lack of modeling constructs which are needed for describing QoS features in SBS. Other simulation approaches and tools, such as HLA [8], SimEvents [9] and OPNET [10], are not suitable for developing ASBS unless they are extended with new abstractions that can describe key characteristics of service-oriented computing. The UML 2.0 has been suggested to support simulating software systems, but its concept of time and execution protocol is limited [11]. The process specification and modeling language (PSML) [12] is useful in designing, implementing, and testing services with simulation support. But it lacks direct support for simulating time-based services, which is important for describing dynamics of ASBS. In our approach, simulation models that are grounded in service-oriented computing concepts will play a central role in validating the dynamics of ASBS with multiple QoS features.

3. Overview of Our Approach

Our approach to developing ASBS consists of the following three major steps:

S1) Gather the knowledge of the underlying cause-effect dynamics that drive the performance and tradeoffs among the four QoS features (timeliness, throughput, accuracy and security) based on controlled experiments and data analysis.

S2) Develop QoS M/A capabilities in ASBS based on the knowledge gathered in S1).

S3) Validate the QoS M/A capabilities developed in S2) through SOA-based simulations.

Although our approach aims at providing a general methodology for developing ASBS, it is obvious that ASBS in various application domains usually have different QoS requirements, resource requirements, and workload patterns. Taking these domain-specific characteristics into consideration, especially in controlled experiments and data analysis in S1), will effectively reduce the effort for knowledge gathering in S1) and improve the quality of ASBS being developed. Currently, we are developing and evaluating our approach with a group of applications utilizing various multimedia services, including video streaming, voice communication, and motion detection in real-time video streams. In Sections 4-6, we will present each step in our approach respectively.

4. Controlled Experiments for Modeling QoS-related Cause-Effect Dynamics

As shown in Figure 1, the QoS M/A modules are at the core of ASBS because they provide the crucial capabilities, including measuring QoS performance and system status, making decisions for adapting the configurations and service operations in ASBS. However, these modules cannot be developed without knowing the QoS-related cause-effect dynamics in ASBS, especially the following three basic aspects:

i. *QoS composition*: How the overall QoS of an ASBS is determined by the QoS of individual services used to compose the ASBS.
ii. *QoS tradeoff*: How a QoS feature of an ASBS affects other QoS features of the ASBS.
iii. *Environment and configuration impact*: How the QoS of a service varies when the configuration of the service or the status of the execution environment changes.

Hence, it is necessary to have a systematic way to gather the knowledge on the underlying cause-effect dynamics driving the performance and tradeoffs among the four QoS features being considered in our approach. Figure 2 depicts the cause-effect chain of user activities, system resource states, system events and QoS performance. A service request from a user calls for a system process, which will utilize certain resources, change the states of the resources and generate certain events in the system environment. The changes of the resource state, the generated events and other extraneous events in the system environment in turn affect the QoS performance of the process. The events in the system environment, especially the extraneous events like network failures and malicious attacks, reflect the disturbances on the process, and hence also affect the QoS performance of the process. The performance models revealing such cause-effect dynamics in SBS are called the *Activity-State-Event-QoS (ASEQ)* models. Constructing the ASEQ models is challenging because the differences in the types of services, ways of service composition, and even service implementations will have different characteristics of resource consumptions which in turn produce different characteristics of various QoS features and the tradeoffs among them.

We have developed an approach to constructing the ASEQ models. Our approach consists of the following three steps:

A1) Design controlled experiments in SBS to collect data related to user activities, system resource states, system events, and QoS performance under various experimental conditions.

A2) Implement and execute the experiments.

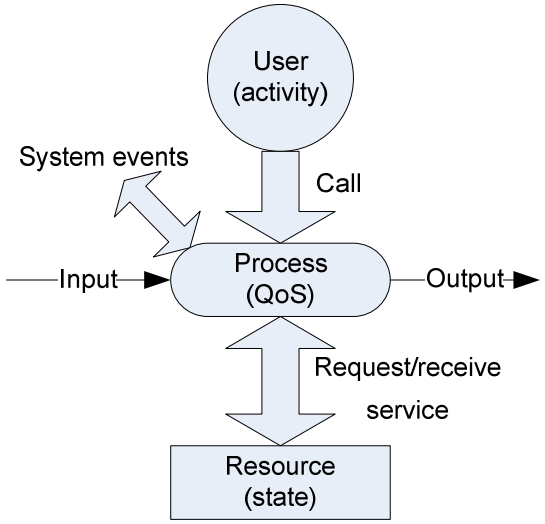

Figure 2. The cause-effect chain of activity-state-event-QoS in SBS

A3) Analyze the collected data to identify the cause-effect chains among activities, states, events, and QoS performance in SBS.

A2) is straightforward, and will not be discussed here. We will first discuss ASEQ models for the four identified QoS features, and then A1) and A3).

4.1 ASEQ Models for Timeliness, Throughput, Accuracy, and Security

Since ASEQ models should reveal the cause-effect chains of user activities, system resource states, system events and QoS performance, we need to define the following components before constructing ASEQ models for the four identified QoS features:

(1) Abstractions of user activities
(2) Abstractions of system events
(3) Abstractions of system resource states
(4) Evaluation of QoS performance

The first three components are common for the four QoS features, whereas the fourth component is distinct for them due to their different nature.

Because user activities in ASBS are carried out through service invocations, users' service requests are used directly in our ASEQ models as the abstractions of user activities. In particular, parameters for service invocations will be used as the input parameters for ASEQ models.

System events are signals indicating certain system behavior or activities not initiated by legitimate users, such as service failure, and attacks on networks and services, and should be captured by monitoring modules. Due to the message-based nature of SBS,

system events in our ASEQ models are represented as messages containing the descriptions of the events, including what, when and where an event has happened. These descriptions will also be used as the input parameters for ASEQ models.

A system resource state is represented by a set of state variables, where state variables are properties of system resources that can be directly acquired or derived from runtime monitoring, such as committed memory, CPU utilization, and the amount of data transferred to/from physical disks. Specifically, a system state is a certain combination of particular ranges of state variables. For example, a simple system resource state can be "the committed memory is between 100 and 200 Mbytes, and CPU utilization is between 10% and 20%". Note that although the state variables are pre-selected by system designers, the definitions of specific system resource states are not given and will be generated during the construction of ASEQ models.

Evaluation of QoS performance is the most important part for constructing ASEQ models. There are well-defined metrics for timeliness, accuracy, and throughput based on which analytical models can be constructed to estimate their performance based on user activities, current system resource states, and event history in the system. Table 1 shows the metrics used for timeliness, accuracy and throughput in our approach. These metrics are selected based on the multimedia services and applications we are currently focusing on. For other services and applications, different metrics may be used. For instance, the accuracy of a web search service will be measured by the precision and recall of the search results, rather than the loss rate and error rate in Table 1.

However, it is very difficult to evaluate the security of a service or a system quantitatively [13]. Hence, in the ASEQ models for security, we focus on the performance impact on other QoS features when security-related service configurations, such as authentication mechanisms and encryption key length, are changed [14], which can be quantified through controlled experiments. Such ASEQ models for security will be used to optimize the performance of ASBS without violating users' security requirements.

4.2 Design of Experiments

To achieve correctness, generality and effectiveness of our experimental results, the following two aspects of our experiment design are most important:
i) Identify an appropriate set of functional services and construct representative application scenarios using these services for our experiments. First, the number of identified services and application scenarios being constructed should avoid excessive cost for performing experiments. Second, the services and application scenarios being constructed should provide a good coverage of various types of services and service composition patterns so that the experimental results will be sufficiently general.
ii) Identify the resource and QoS requirements of the services in various application scenarios to be used in the experiments, and the corresponding experiment conditions and control variables determining these conditions. Similar to i), the resource and QoS requirements of the services in various application scenarios should provide a good coverage of various characteristics of the QoS features and system resources for the generality of the experiment results.

We have identified a set of services, including voice communication, video streaming, and motion detection in video streams, which have well-understood QoS requirements [5]. Table 2 provides an overview of these identified services. Using these services, five different application scenarios, including

Table 1. QoS Metrics

QoS Features	Metrics	Definition
Accuracy	Loss rate	The number of bits lost between two points in telecommunications after transmission.
	Error rate	The bit error rate is the frequency of erroneous bits between two points in telecommunication after transmission.
Timeliness	Service delay	The time elapsed between a user submitting a service request and receiving a service response
	Network transmit delay	The time elapsed between the emission of the first bit of a data block by the transmitting end-system, and its reception by the receiving end-system
Throughput	Data rate	The rate in which data are encoded
	Bandwidth	The data transfer rate, measured in bits per second

online radio broadcasting, motion analysis, online video sharing, real-time surveillance camera, and net-meeting, have been designed and identified as an initial set of our experiments to cover various composition patterns, QoS and resource requirements. As shown in Table 3, online radio broadcasting, motion detection, and online video sharing scenarios only use one service each. These three scenarios are mainly designed for constructing performance models for the three identified services. Real-time surveillance camera and net-meeting scenarios use different compositions of services, and are mainly designed to acquire knowledge on QoS composition.

For each experiment, a set of experiment conditions are identified based on QoS and resource requirements, and user activities in each application scenario. These experimental conditions are represented by various value assignments for experiment control variables. There are two types of experiment control variables: one reflects the initial system state, such as the current CPU workload and network bandwidth utilization, and the other reflects certain characteristics of user activity, such as the number of concurrent users for a service and the quality of audio or video streams requested by users. Usually, these experiment control variables are parameters of users' service requests as well as certain workflow generation tools, such as SWORD and StreamGen, which can be used to simulate or create various operational environments for our experiments.

4.3 Construction of the ASEQ models in ASBS

We have developed a methodology for analyzing the experimental data generated above to construct the ASEQ models in ASBS. We summarize our methodology as follows:

B1) Data reduction: Identify the data items having significant changes of values when the system changes from idle condition to an experimental condition using a statistical test, the Mann-Whitney test [15].

B2) Data clustering: The data items are grouped together based on the similarity in their statistical behavior using the following method. Apply the hierarchical clustering method [16] to generate clusters of data items under each experimental condition, and examine the generated clusters by counting the co-occurrences of data items in the same cluster across different experimental conditions to find the data items with similar statistical behaviors

B3) Analysis of the impact of experimental condition changes on data items: Experimental conditions are determined by a set of experiment control variables. The statistical significance of experiment control variables for data items is analyzed using ANOVA [17].

B4) Establishment of the relationships among activities, states, events, and QoS performance in SBS: The data items are first categorized into activity, state, event, and QoS data. Then, use the Classification And Regression Tree (CART) analysis [18] to identify the cause-effect chains among activity, state, event, and QoS performance data. We have developed a tool to transform the decision trees generated from CART analysis automatically into conditional evaluation statements, which will later be used in the QoS monitoring modules.

Note that a part of the experimental data will be reserved for validating simulation models for SBS as indicated in Section 6.

5. Development of QoS M/A Modules

For large-scale ASBS, QoS M/A tasks often need to be partitioned and distributed in a hierarchical manner for better performance and manageability. We have developed a three-layered intelligent control architecture for ASBS [1], in which monitoring and adaptation tasks are distributed to user, workflow, and service management layers. Here, we will discuss the

Table 2. An overview of the services identified for our experiments

Services	*Functional Descriptions*	*Resource Characteristics*
Voice communication	Provide voice-based communication over the network	Network-intensive
Video streaming	Provide the live video feed from a web camera	Network-intensive
Motion detection	Identify the differences in two images	Computation-intensive

Table 3. A summary of service compositions in the five identified scenarios

Scenarios	*Services Used*	*Composition Patterns*
Online radio Broadcasting	Voice communication	None
Motion analysis	Motion detection	None
Real-time surveillance camera	Video streaming, motion detection, voice communication	Sequential, Exclusive Choice
Video sharing	Video streaming	None
Net-meeting	Voice communication, video streaming	Parallel

development of QoS M/A modules in ASBS based on the ASEQ models generated in Section 4.

❖ **QoS monitoring modules.** As shown in Figure 1, functional services in ASBS are often used to compose complex workflows to perform high-level tasks for users. Such workflows can be viewed as *composite services*, and often need to satisfy the overall QoS of the workflow, rather than the QoS of the individual services used in the workflows. QoS monitoring module needs to keep track of the workflow execution, and detect various execution problems, such as service failure, network disconnection, and QoS degradation. In [19], we have developed an adaptive execution monitoring approach in ASBS, in which distributed monitoring modules are automatically generated based on the knowledge of the workflows to be executed, the systems executing the workflows, and the possible failure in workflow execution. The generated monitoring modules will proactively acquire the status information of computing resources to be used in future workflow execution for early detection of potential execution problems, and self-reconfigure or load new monitoring modules to acquire additional information if any problem occurs. Currently, we are improving this approach by incorporating the knowledge generated in our data analysis (see Section 4.3). Specifically, the hierarchical clustering results will be used in the indicator selection process to reduce the number of targets to be monitored. The decision trees generated from the CART analysis will be embedded in the monitoring modules for estimating near-future changes in QoS performance and the execution environment.

❖ **QoS adaptation module.** To develop the QoS adaptation module in ASBS, we first define QoS expectation functions to quantify gains and losses for satisfying and violating users' QoS requirements, respectively. Then, the ASEQ models and QoS expectation functions will be used to derive the optimization and control synthesis algorithms for generating adaptive control commands. QoS adaptation in ASBS is formulated as a multi-objective optimization problem [20], in which the ASEQ models and QoS expectation functions are used as the constraints and objective functions respectively. The solution of this optimization problem will determine system and service configurations which result in the desired QoS tradeoffs. Hence, the optimization problem includes the representation of the QoS tradeoffs, and the solution reflects the result from the tradeoff analysis performed while solving the optimization problem. The QoS adaptation module in ASBS will be then developed based on the mathematical methods to solve this optimization problem.

6. Validation of ASBS Design through SOA-based Simulations

Running extensive experiments on a real testbed to collect a large amount of data can be quite costly, and it is intrinsically difficult to derive internal dynamics of services for generality analysis from experimental data. We have developed simulation models for SBS, and are using them to validate and improve our research results as follows:

(1) Develop simulation models for services and application scenarios in our experiments based on the service specifications and ASEQ models.
(2) The simulation models developed in (1) will be used to conduct simulations using the same experiment conditions that generate the experimental data. Data collected in simulations will be compared with the reserved experimental data to validate the simulation models.
(3) The validated simulation models will be integrated with the real M/A modules to support the rapid development for simulation and evaluation of ASBS with M/A modules.
(4) The validated simulation models will be used to perform simulations under additional experiment conditions, which incorporate more service activities, system events and control structures in service composition, to obtain more simulated data for further improvement and generalization of our results.

We have developed an SOA-compliant simulation framework based on the DEVS (Discrete Event System Specification) formalism for modeling SBS [21]. In this framework, a set of abstract component models based on DEVS formalism and conforming to SOA principles, such as autonomy and composability, is developed to enable modeling and simulating primitive and composite services. In particular, atomic DEVS models for publisher, subscriber, and broker in SOA are developed. The input/output ports with couplings of DEVS models are extended to provide a messaging framework supporting various message-based interactions among publisher, subscriber, and broker services in SOA. Generic message classes are developed as abstractions of WSDL and SOAP specifications used in SOA. Coupled DEVS models are used to represent composition of services in SOA. To model network communication effect on the QoS of SBS, a simple model representing the network with stochastic delay is developed. The network model allows the publisher, subscriber, and broker services to communicate with one another. For each of the publisher, broker, subscriber, and network models, a

transducer model is developed to measure throughput and timeliness QoS.

Based on this simulation framework, an SOA-compliant DEVS (SOAD) simulator has been designed and implemented based on DEVSJAVA [22]. Using this simulator, common variations of SBS can be systematically conceptualized, modeled, and efficiently simulated given the inherent support for component-based modeling and parallel execution. Two preliminary simulation experiments for a voice communication service and a travel agent service have been conducted to validate and demonstrate the capability and efficiency of SOAD framework and its support for SOA-based simulations.

7. Conclusions and Future Research

In this paper, we have presented a performance-model-oriented approach to developing ASBS. The current research progress is discussed, including constructing the ASEQ models through controlled experiments, developing QoS M/A capabilities in ASBS based on the ASEQ models, and validating ASBS design through SOA-based simulations. Future research includes development of methods for solving the optimization problem for QoS adaptation, improvement of our SOA-based simulation environment to support the integration of real and simulated SBS, and integration and evaluation of our research results. We also need to improve our design of experiments and data analysis methodology for constructing ASEQ models.

Acknowledgment

This work is supported by National Science Foundation under grant number CCF-0725340.

References

[1] C.-H. Jiang, H. Hu, K.-Y. Cai, D. Huang, and S. S. Yau, "An Intelligent Control Architecture for Adaptive Service-based Software Systems with Workflow Patterns," *Proc. 5th IEEE Int'l Workshop on Software Cybernetics,* in conjunction with COMPSAC 2008, pp. 824-829.

[2] M. Reisslein, K.W. Ross and S. Rajagopal, "A Framework for Guaranteeing Statistical QoS", *IEEE/ACM Trans. on Networking*, vol. 10(1), 2002, pp. 27-42.

[3] X. Xiao, T. Telkamp, V. Fineberg, C. Chen and L. M. Ni, "A Practical Approach for Providing QoS in the Internet Backbone", *IEEE Communications*, vol. 40(12), 2002, pp. 56 – 62.

[4] Z. Yang, N. Ye and Y.-C. Lai, "QoS Model of a Router with Feedback Control", *Quality and Reliability Engineering Int'l,* vol. 22(4), 2006, pp. 429-444.

[5] Y. Chen, T. Farley and N. Ye, "QoS Requirements of Network Applications on the Internet", *Information, Knowledge, Systems Management*, vol. 4(1), 2004, pp. 55-76.

[6] J. Zhou, K. Cooper, I. Yen and R. Paul, "Rule-Base Technique for Component Adaptation to Support QoS-based Reconfiguration", *Proc. 9th IEEE Int'l Symp. Object-oriented Real-time Distributed Comp.*, 2005, pp. 426–433.

[7] D. R. Hild, H. S. Sarjoughian and B. P. Zeigler, "DEVS-DOC: A Modeling and Simulation Environment Enabling Distributed Codesign," *IEEE Trans. on Systems, Man and Cybernetics*, Part A, vol. 32(1), 2002, pp. 78 -92.

[8] J. S. Dahmann, "High Level Architecture for Simulation," *Proc. 1st Int'l Workshop on Distributed Interactive Simulation and Real-time Applications (DIS-RT'97),* 1997, pp. 9-14.

[9] MathWorks, "MATLAB/Simulink", 2005. Available at: http://www.mathworks.com.

[10] OPNET, "OPNET Modeler", December 2005. Available at: http://opnet.com.

[11] D. Huang, and H. Sarjoughian, "Software and Simulation Modeling for Real-Time Software-Intensive Systems," *Proc. 8th IEEE Int'l Symp. on Distributed Simulation and Real-time Applications*, 2004, pp. 196-203.

[12] W. T. Tsai, Y. Chen, R. Paul, X. Zhou and C. Fan, "Simulation Verification and Validation by Dynamic Policy Specification and Enforcement", *SIMULATION*, vol. 82(5), 2006, pp. 295-310.

[13] D. Lie, and M. Satyanarayanan, "Quantifying the Strength of Security Systems," *Proc. 2nd USENIX Workshop on Hot Topics in Security,* 2007, article no. 3.

[14] S. S. Yau, M. Yan, and D. Huang, "Design of Service-based Systems with Adaptive Tradeoff between Security and Service Delay," *Proc. 4th Int'l Conf. on Autonomic and Trusted Computing*, 2007, pp. 394-401.

[15] H. B. Mann, and D. R. Whitney, "On a Test of Whether One of Two Random Variables is Stochastically Larger than the Other," *Annals of Mathematical Statistics*, vol. 18, 1947, pp. 50-60.

[16] S. C. Johnson, "Hierarchical Clustering Schemes," *Psychometrika*, vol. 2, 1967, pp. 241-254.

[17] D. C. Montgomery, G. C. Runger and N. F. Hubele, *Engineering Statistic (4th ed.)*, John Wiley, 2006.

[18] L. Breiman, J. Friedman, R. A. Olshen and C. J. Stone, *Classification and Regression Trees*, Wadsworth Int'l Group, 1984.

[19] S. S. Yau, D. Huang and L. Zhu, "An approach to adaptive distributed execution monitoring for workflows in service-based systems," *Proc. 31st Annual Int'l Computer Software and Application Conf.*, vol. 2, 2007, pp. 211-216.

[20] R. L. Rardin, *Optimization in Operations Research*, Prentice Hall, 1998.

[21] H. Sarjoughian, S. Kim, M. Ramaswamy, and S. S. Yau, "A simulation framework for service-oriented computing," *Proc. 2008 Winter Simulation Conf.,* 2008, to appear.

[22]Arizona Center for Integrative Modeling and Simulation, *http://www.acims.arizona.edu/SOFTWARE*., 2006