[ESR+08] *Julian Eckert, Stefan Schulte, Nicolas Repp, Rainer Berbner, Ralf Steinmetz*; **Queuing-based Capacity Planning Approach for Web Service Workflows Using Optimization Algorithms.** In: IEEE International Conference on Digital Ecosystems and Technologies 2008 (IEEE DEST 2008), February 2008. S 313-318.

2008 Second IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST 2008)

Queuing-based Capacity Planning Approach for Web Service Workflows Using Optimization Algorithms

Julian ECKERT, Stefan SCHULTE, Nicolas REPP,
Rainer BERBNER, and Ralf STEINMETZ, Fellow, IEEE
Multimedia Communications Lab (KOM), Department of Electrical Engineering and Information Technology,
Technische Universität Darmstadt, Germany
e-mail: julian.eckert@kom.tu-darmstadt.de

Abstract—**One major challenge for service-oriented workflows in digital ecosystems is capacity planning for cross-organizational Web service workflows in order to avoid performance degradation. In order to analyze the execution capacity of Web service workflows and to plan the workflow control, queuing theory can be used to describe the average performance behavior of a workflow.**

By addressing capacity planning of Web service workflows, resource usage becomes more and more important. Capacity planning and performance measurement are crucial to ensure that the workflow execution remains feasible and SLA violations due to overload are avoided. Thus, this paper presents a capacity planning approach for Web service workflows based on queuing theory to support capacity planning decisions. Further we describe an optimization algorithm how to achieve an optimal utilization of the invoked Web services at minimal costs.

Index Terms—**Capacity Planning, Quality of Service, Service-oriented Architecture, Queuing Theory, Web service Workflow.**

I. Introduction

In service ecosystems, the collaboration of business partners became more and more important. The reason for this is the globalization and deregulation of markets, which create the need for enterprises to be able to react to their changing environment and therefore also adapt their business processes continuously [12]. Concerning business processes, cooperation with external business partners forms the advantage that enterprises are able to buy and invoke external services which are less expensive and/or have a better performance in order to reduce costs or increase the execution capacity of a business process.

A continuous business process management is needed in order to realize the requested flexibility of enterprises [3] which allows to react flexible and to manage the business process to avoid performance problems during the process execution. The challenges for the business process management are on the one hand to meet customer expectations, i.e. Quality of Service (QoS) and on the other hand to minimize costs to stay competitive. Thus, a holistic process management includes key issues as capacity, reliability, availability, scalability, and security [1].

In order to realize the invocation of services from external partners as well as the internal legacy systems, the Service-oriented Architecture (SOA) paradigm is often recommended to enable agile business processes [18]. Self-contained loosely coupled services can be composed and orchestrated to cross-organizational business processes in order to react fast and flexible to changing business needs and to optimize the business process concerning costs and QoS parameters. Workflows, which are typically the automation of a business process, may use Web services as an open standard technology [2], [19]. Reference architectural styles for Service-oriented Computing as the matchmaker style or the broker style are described in [7].

A detailed workflow execution method within one workflow in digital ecosystems can be found in our previous work [4], [5]. A continuous workflow management for workflows with a high amount of execution requests e.g., claims handling, loan handling, and accounting, requires capacity planning strategies and a performance analysis in order to be able to serve all execution requests and even peaks. Capacity planning can be described as the process of predicting when future load levels of a business process will saturate the system and determine the most cost-effective way of delaying system saturation as much as possible [17]. Performance prediction, cost model development, cost prediction, and cost/performance analysis are amongst others typical steps concerning a complete business process management [16].

The aim of a capacity planning approach, in the context of Web service workflows and business process management, is the avoidance of poor performance and Service Level Agreement (SLA) violations before those impact the business.

Considering business processes with high repetition rates as e.g., a generic credit process, the high number of execution requests of this process has to be processed by a composed Web service workflow. In order to ensure the complete execution of all credit requests a continuous capacity planning is required. It is indispensable to achieve a proactive and continuous capacity planning procedure to guarantee that all requests of a workflow can be served. Due to the necessity of a holistic workflow control for workflows with high repetition rates, we propose a capacity planning and cost-effective approach for Web service workflows. In [8] we show how to plan the workflow execution by workflow parallelization in order to serve a large amount of incoming workflow execution requests in a specific time period.

A worst-case performance analysis for Web service workflows using network calculus is shown in [9]. In order to analyze the average performance behavior of Web service workflows, this paper focuses on capacity planning as-

2008 Second IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST 2008)

pects by using queuing theory.

The remainder of this paper is structured as follows. The next section introduces queuing theory basics for the M/M/1 queue. In Section III the system model is described followed by the application of queuing theory to Web services in Section IV. The throughput analysis of a Web service workflow is discussed in Section V. Section VI describes an optimization algorithm how to maximize the utilization of all invoked Web services at minimal costs and minimum average response time of the entire workflow. The paper closes with a conclusion and an outlook on future work.

II. Queuing Theory Basics For The M/M/1 Queue

The principle of queuing models is well studied in the literature [13], [14]. Queuing occurs because the number of arrivals of requests to a resource varies in time. An overview about a basic queuing model is shown in Fig.1. The M/M/1 queuing model assumes that the arrivals are distributed in a Poisson manner (equation 1) with the arrival rate λ. The average number of arrivals in a specific time period t is $E(k) = \lambda t$ and the variance v is $v = \lambda t$. At the server the service times are exponentially distributed.

$$p_i(t) = \frac{(\lambda t)^i}{i!} e^{-\lambda t}, t \geq 0 \quad \forall i \in N. \tag{1}$$

In a Poisson process the customer interarrival times are exponentially distributed. This memoryless characteristic implies that the next state x_{n+1} does not depend on the time the process has spent in state x_n.

The context between a birth-death process, which is a special case of a Markov process, and the M/M/1 queue is that a birth reflects the arrival of a new customer, either to the queue or directly to the server (depending on the status of the queue) and a death represents that a customer has already been serviced and has left the system.

The queue is usually modeled as a black box where λ represents the requests respectively jobs that arrive per time unit on average. The number of incoming jobs in a specific time period λ is called the *arrival rate* or the *arrival intensity*. The order of processing the jobs is in the same order as they arrive, i.e. in a first come first serve discipline [6]. If there are already jobs in the queue, the incoming job has to wait in the queue until the server is empty and the job can be processed at the server.

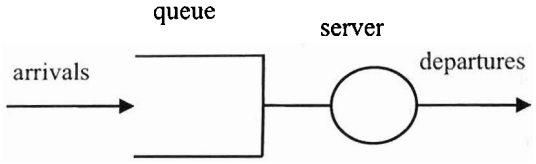

Fig. 1. Basic qucuing model

A. Arrival rate and service rate

At the queuing station the jobs arrive with a negative exponential interarrival time distribution, i.e. the rate at which they arrive at the server is denoted by λ with the interarrival time $t = 1/\lambda$. Considering several parallel incoming Poisson processes $\lambda_1, \ldots, \lambda_n$, the resulting process λ_{result} is also a Poisson process which can be calculated as shown in equation (2).

$$\lambda_{result} = \sum_{i=0}^{\infty} \lambda_i . \tag{2}$$

At the server, the service times are also negatively exponentially distributed with the mean service time $x = 1/\mu$, where μ denotes the rate at which the jobs are processed.

B. Overload avoidance and utilization

In order to avoid overload the average number of arrivals per time unit may not exceed the average number of jobs the server is able to process per time unit, i.e. $\lambda \leq \mu$. This precondition ensures that the system is stable and an accumulation of an infinite queue is avoided. The utilization ρ_i of the server i can be calculated as shown in equation 3.

$$\rho_i = \frac{\lambda_i}{\mu_i} .$$

This implies that the system (server) can only be in stable state if $\rho_i \leq 1$. Otherwise there would be more incoming jobs than the server is able to serve per time unit.

C. Average number of customers in the system and average system time

The *average number of customers* N in the system represents the number of jobs that are waiting in the queue and the number of jobs that are in the server at a specific time. The average number of customers N in the system yields to:

$$N = \frac{\rho}{1-\rho} \tag{4}$$

The *average system time* T is the time a job spends in the system (queue and server) until the job is processed and has left the system. The calculation is shown in equation (5).

$$T = \frac{1/\mu}{1-\rho} \tag{5}$$

2008 Second IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST 2008)

D. Burke's Theorem

The departure process γ of a stable single server M/M/1 queue with an arrival and a service rate λ and μ respectively, is a Poisson process with the rate λ [13].

III. System Model

In [12] a business process is defined as a consecution of activities which creates a value to the customer. Typically a business process can be decomposed into several basic activities, which can be executed by a specific Web service WS_i $(i=1,...,m)$ which fulfils the required functionality of the considered task. When talking about Web service workflows in digital ecosystems and service-oriented environments it is quite important that the selection and composition of Web services can be done from internal services as well as from external partners [20]. The focus of our research are workflows with high repetition rates as the previously mentioned generic credit processes, which can be decomposed into the subprocesses loan request, credit assessment, servicing, and workout.

The considered workflows in this paper, with a sequential execution of the tasks, are only a subset of the workflows encountered in reality. However, queuing theory allows the analysis of complex networks (and therefore complex workflows) containing loops, iterations, forking and joining paths, etc. [13]. In our further analysis we consider workflows with a sequential execution of the tasks, respectively Web services. These tasks are, dependent on the granularity of the task decomposition, decomposed in such a way that there exist ranges of Web services $WS_{i,j}$ which fulfill the required functionality of the considered task i.

The orchestrator as the workflow controller acts as an intermediary and is responsible for the execution of a workflow. It has to handle all execution requests and has to ensure that all requests can be served within a specific time period. The aim of the workflow controller has to be to serve all incoming requests at minimal costs and at minimal time.

Assuming a huge amount of incoming requests, the incoming arrivals can be modeled as a Poisson process as described in Section II. The assumption that the arrivals can be modeled as a Poisson process can be assumed in various scenarios [10]. In order to execute all incoming requests the workflow controller has to stores all incoming requests before respectively during the workflow execution. The workflow is executed by invoking several Web services and composing them to a workflow. For each task the workflow controller has to choose the appropriate Web service which fulfills the required functionality which is described in the SLAs. An example of the considered system is shown in Fig.2. In the workflow consisting of m different activities the orchestrator respectively workflow controller has to ensure that task i $(i=1,...,m)$ has to be executed before task i' $(i'=1,...,m)$ if $i<i'$. The workflow controller has to create an execution plan in order to optimally use the capacities of each Web service to increase the throughput by an optimal utilization of the invoked Web services, i.e. to select the most efficient Web services. Furthermore, it has to maximize the throughput, and realize an optimal resource usage in order to use the capacities of the Web services optimally.

IV. Application of Queuing Theory to Web Services

The workflow, as mentioned in Section III consists of several Web services. The number of incoming requests of the workflow in a specific time period can be modeled as a Poisson process with a specific arrival rate which is referred to as λ_w. The workflow controller is faced with this arrival rate and has to invoke the first Web service out of a range of n Web services, which fulfills the required functionality, in order to execute the first task of the workflow, denoted by $WS_{1,i}$. After invoking a Web service for the first task, the workflow controller has to invoke a Web service for the second task out of a range of n, Web services which fulfill the required functionality of task two, denoted by $WS_{2,i}$. The specific aspect in the consideration concerning queuing theory is that due to a large amount of incoming requests there will be a queue for the execution of the first Web service, the second Web service et cetera.

A. Throughput and utilization analysis of a Web service

Considering a Web service which is faced with a specific arrival rate λ and a service rate μ the throughput, respectively output γ in the long run, if it holds that $\lambda < \mu$, i.e. $\rho < 1$, will be the same as the input rate λ, i.e. $\gamma = \lambda$.

The queue which is accumulated at a Web service occurs due to random arrivals and the fact that requests also may occur in a burst. The workflow controller stores the requests in the queue and has to forward them to the subsequent Web service as soon as the preceding job is processed. The higher the utilization of the Web service ρ is the better the execution capacity of the considered Web service is used. The calculation of the average number of customers at the Web service and the average system time which represents the average response time can be calculated with equation (4) and (5).

B. Cost model for the Web service usage

In the literature several pricing models are discussed, as pay-per-use or flat-rate models for Web services [11], [15].

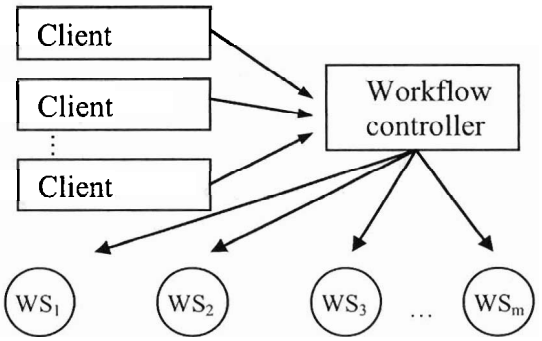

Fig. 2: Considered scenario

2008 Second IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST 2008)

In our consideration a pricing model is used, which is a combination of a pay-per-use and a flat-rate model.

A service provider offers a Web service that provides a specific service rate μ_i which represents the amount of requests it is able to process within a specific time period. The Web service provider charges the customer the amount c for the usage of the Web service. The incidental costs are independent of the real usage of the Web service. The usage is determined by its execution capacity, which is the number of requests it is able to process within a specific time period. This cost model is referred to a volume rate, which implies that it is the interest of the service customer to use the offered execution capacity efficiently.

C. Capacity-usage of a single Web service

In order to use the execution capacity of one single Web service WS_i the most efficient way the utilization ρ has to be nearly 1. As depicted in equation (4) the average number of customers in the system increases by an increasing value of the utilization ρ. This implies that the average response time of a Web service execution increases as well which is shown in Fig. 3.

V. Throughput Analysis of a Web Service Workflow

The first consideration focuses on workflows in which the tasks are decomposed into m subtasks which have to be executed sequentially. The challenge is how to measure the overall response time and the overall throughput of the workflow.

The sequential workflow, as described in the system model, can be described and analyzed as a Feed-forward queuing network (FFQN) [13], which implies that the outgoing flow of queue i is the incoming flow of queue $i+1$, i.e. the workflow behaves as a acyclic queuing network. The assumption is that each invoked Web service acts as a M/M/1 queue in a stable state ($\lambda < \mu$). The relation between the considered Web service workflow and the FFQN can be seen in Fig. 4.

As precondition for the stability of the workflow, the utilization ρ_i for each invoked Web service WS_i the service rate μ_i has to be smaller than 1, i.e. $\rho_i = \lambda / \mu_i < 1$.

The incoming workflow execution requests have a specific arrival rate λ_w, thus, the arrival rate λ_1 at WS_1 equals $\lambda_1 = \lambda_w$. According to Burke's theorem, mentioned in Section II, the output γ_1 at WS_1 is the same as the input rate λ_w, if $\rho_1 \leq 1$. The invoked Web service WS_2 of task 2 has the input rate $\lambda_2 = \gamma_1 = \lambda_1 = \lambda_w$ if $\rho_2 \leq 1$. By applying this consideration to all invoked Web services the overall throughput γ_w of the entire workflow is the same as the input rate, i.e. $\gamma_w = \lambda_w$ if for all tasks i $(i=1,...,m)$ holds $\rho_i \leq 1$.We assume that the workflow is a closed system. The invoked Web services are not used by any other workflow, i.e. at any Web service the arrival rate at each Web service equals λ_w

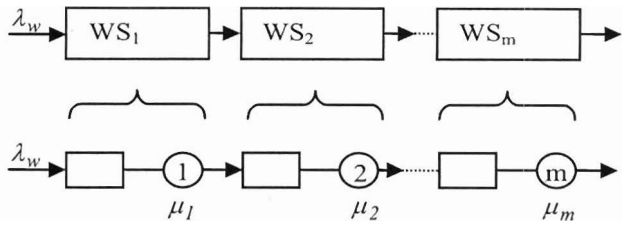

Fig. 4. FFQN model for workflows

In the case of $\rho_i > 1$, an infinitely large waiting queue and an infinitely large response time would be accumulated. In this case the Web service should be replaced by other Web services with a higher service rates or a Web service should be instantiated in order to serve all incoming requests.

VI. Web Service Workflow Utilization Optimization

This section presents our optimization approach in order to use the execution capacities of all invoked Web services of the workflow the most efficient way. Thus, some constraints as cost, response time, and number of customers in the queue are considered as well.

The considered workflow consists of m different tasks which have to be executed sequentially. For each task i the workflow controller has to choose a Web service which fulfills the required functionality out of $j=1,...,n$ available Web services per task i. All of these Web services fulfill the required functionality of task i. The Web services differ in the provided service rates $\mu_{i,j}$ and the costs $c_{i,j}$. The higher the service rate $\mu_{i,j}$ is the higher are the costs $c_{i,j}$. For all n Web services $(j=1,...,n)$ of task i a binary variable $x_{i,j}$ is introduced to model whether a Web service $WS_{i,j}$ is used for the workflow execution or not. In order to avoid that more than one Web service of one task is used at the same time, for the binary variable should hold:

$$\sum_{j=1}^{n} x_{i,j} = 1 \quad \forall i = 1,...,m. \tag{6}$$

The following sections describe the optimization approach in more detail.

A. Objective function

In order to maximize the utilization of all invoked Web services of the workflow, the workflow controller has to invoke those Web services, which maximize the overall utilization. Thus, it has to calculate all potential utilizations $\rho_{i,j}$ of all Web services $WS_{i,j}$ given the arrival rate of the incoming requests of the requestors λ_w and the individual service rates $\mu_{i,j}$. With these utilization parameters, the

2008 Second IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST 2008)

workflow controller is able to compute the overall maximum utilization with equation (7).

$$F(\vec{x}) = \frac{1}{n} \sum_{i=1}^{m} \sum_{j=1}^{n} \delta_{i,j} x_{i,j} . \quad (7)$$

As shown in Fig. 3 the higher the utilization of a Web service is the higher is the average response time. In order to avoid an increase in response time and to border the costs for the workflow execution some constraints are introduced in the following section.

B. Constraints

In a sequential workflow the overall average response time T_w can be calculated by the summation of all average response times $T_{i,j}$ of all invoked Web services of the composed workflow.

The aim of the optimization algorithm is to maximize the throughput, thus, in order to avoid an infinite increase of the overall average response time, the overall average response time of the workflow has to be restrained with a maximum average response time T_{max} as shown in equation (8). In this constraint $x_{i,j}$ represents the binary variable for each category which indicates whether the Web service $WS_{i,j}$ is invoked or not.

$$\sum_{i=1}^{m} \sum_{j=1}^{n} T_{i,j} x_{i,j} \le T_{max} . \quad (8)$$

As mentioned in Section III the cost model of the Web service usage is a volume rate, i.e. the service consumer has to pay a fixed amount $c_{i,j}$ for the usage of Web service $WS_{i,j}$ with the service rate $\mu_{i,j}$. The computation of the overall costs C_w for the entire workflow execution can be done by the summation of all costs for each invoked Web service $WS_{i,j}$ as shown in equation (9).

$$C_w = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{i,j} x_{i,j} . \quad (9)$$

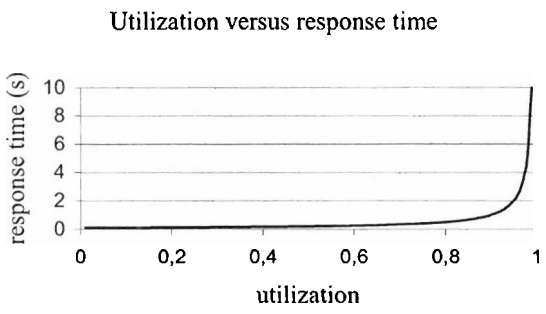

Fig. 3. Utilization versus response time

An important constraint in this optimization algorithm is that the overall costs of the workflow execution are constrained by a certain boundary C_{max}, which is shown in equation (10) in order to realize a cost-efficient Web service composition.

$$\sum_{i=1}^{m} \sum_{j=1}^{n} c_{i,j} x_{i,j} \le C_{max} \quad (10)$$

The fact that the queues of requests at each Web service can be handled as autonomous queues, the number of jobs the workflow controller has to store during the execution of all workflow execution requests can be done by the summation of all average customers at the Web services, denoted by N_w that is shown in equation (11).

$$N_w = \sum_{i=1}^{m} \sum_{j=1}^{n} N_{i,j} x_{i,j} \quad (11)$$

In order to constrain this overall queue length the appropriate constraint with the boundary N_{max} is as described in equation (12).

$$\sum_{i=1}^{m} \sum_{j=1}^{n} N_{i,j} x_{i,j} \le N_{max} . \quad (12)$$

The remaining but nevertheless important constraint in order to avoid overload of the system is that the utilization of each invoked Web service is smaller than 1 as shown in equation (13). All the invoked Web services have to be in stable state as mentioned earlier.

$$\rho_{i,j} \le 1 \quad \forall i = 1,\ldots,m . \quad (13)$$

This proposed optimization approach, consisting of an objective function and some constraints, facilitates that the workflow controller is able to optimize the execution capacity of all invoked Web services. The constraints (8) and (9) allow the workflow controller to optimize the Web service utilization given a fixed cost level and a fixed overall response time that he wants to offer his customers, i.e. those requestors that want to execute the workflow. The workflow controller acts in this context as an intermediary that composes the workflow out of a given subset of Web services and sells the workflow execution as a service to its customers.

VII. Conclusion

In this paper we propose a capacity planning approach for Web service workflows in order to maximize the utilization of the invoked Web services. The rationale of this model is to apply results from queuing theory to Web service workflows. The concept of Poisson arrivals is used to describe the arrivals of requests. The concept of exponentially distributed service times at each invoked Web service

is well-suited to capture the service offered by a service provider. It is shown how the execution capacity of Web services of the entire workflow can be maximized by using an optimization algorithm.

Our further research aims at extending our approach with simulations and enhancements of the proposed optimization problem. Further, we will enhance the approach considering workflow parallelization and other resource planning problems.

VIII. ACKNOWLEDGMENTS

This work is supported in part by the E-Finance Lab e.V., Frankfurt am Main, Germany (http://www.efinancelab.com).

IX. REFERENCES

[1] V. A. F. Almeida and D. A. Menascé, "Capacity Planning: An Essential Tool for Managing Web Services," *IT Professional*, vol. 4, no. 4, 2002, pp. 33-38.

[2] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, *Web Services. Concepts, Architectures and Applications*, Springer, Berlin: 2004.

[3] J. Becker, M. Kugeler, and M. Rosemann, *Process Management. A Guide for the Design of Business Processes*, Springer, Berlin: 2004.

[4] R. Berbner, O. Heckmann, and R. Steinmetz, "An Architecture for a QoS driven composition of Web Service based Workflows," in *Proceedings of the Networking and Electronic Commerce Research Conference*, 2005.

[5] R. Berbner, T. Grollius, N. Repp, O. Heckmann, E. Ortner, and R. Steinmetz, "An approach for the Management of Service-oriented Architecture (SoA) based Application Systems," in *Proceedings of Enterprise Modelling and Information Systems Architectures*, 2005.

[6] J. P. Buzen, "Computational algorithms for closed queuing networks with exponential servers," *Communications of the ACM*, vol. 16, no. 9, 1973, pp. 527-531.

[7] T. S. Dillon, C. Wu, and E. Chang, "Reference Architectural Styles for Service-Oriented Computing", in *Proceedings of the IFIP International Conference of Networked and Parallel Computing*, 2007, pp. 543-555.

[8] J. Eckert, N. Repp, S. Schulte, R. Berbner, and R. Steinmetz, "An Approach for Capacity Planning for Web Service Workflows," in *Proceedings of the 13th Americas Conference on Information Systems*, 2007.

[9] J. Eckert, K. Pandit, N. Repp, R. Berbner, and R. Steinmetz, "Worst-Case Performance Analysis of Web Service Workflows," in *Proceedings of the 9th International Conference on Information Integration and Web-based Application & Services*, 2007.

[10] M. Gillmann, G. Weikum, and W. Wonner, „Workflow management with service quality guarantees," in *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, 2002.

[11] D. Gouscos, M. Kalikakis, and P. Georgiadis, "An Approach to Modeling Web Service QoS and Provision Price," in *Proceedings of the 4th International Conference on Web Information Systems Engineering Workshops*, 2003, pp. 121-130.

[12] M. Hammer and J. Champy, *Reengineering the Corporation: A Manifesto for Business Revolution*, HarperBusiness, New York: 2003.

[13] B. R. Harverkort, *Performance of Computer Communication Systems: A Model-Based Approach*, John Wiley & Sons Inc., New York: 1998.

[14] E. Lazowska, J. Zahorjan, S. Graham, and K. Sevcik, *Quantitative System Performance: Computer System Analysis Using Queuing Network Models*, Prentice Hall, Englewood Cliffs: 1984.

[15] G. E. Mathew, J. Shields, and V. Verma, "QoS Based Pricing for Web Services," in *Proceedings of the 5th International Conference on Web Information Systems Engineering Workshops*, 2004, pp. 264–275.

[16] D. A. Menascé and V. A. F. Almeida, *Capacity Planning for Web Services: Metrics, Models, and Methods*, Prentice Hall, Upper Saddle River: 2002.

[17] D. A. Menascé, V. A. F. Almeida, and L. D. Dowdy, *Capacity Planning and Performance Modeling: From Mainframe to Client-Server Systems*, Prentice Hall, Upper Saddle River: 1994.

[18] M. P. Papazoglou, "Service-Oriented Computing: Concepts, Characteristics and Directions," in *Proceedings of the 4th International Conference on Web Information Systems Engineering*, 2003, pp. 3-12.

[19] Workflow Management Coalition, "Workflow Management Coalition – Terminology & Glossary," *Technical Report WFMC-TC-1011*, 1999.

[20] L. Zeng, B. Benatallah, A. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-aware Middleware for Web Service composition," *IEEE Transactions on Software Engineering*, vol. 30, no. 5, 2004, pp. 311-328.