Performance Analysis of Web Applications

M. Elena Gómez Martínez [*]

May 31, 2005

1 Introduction

Performance evaluation focusses on the analysis of the dynamic behavior of a system and the prediction of indices or measures such as its throughput, utilization or response time [17].

Most current practices in performance evaluation of software systems are based on the well-know *fix-it-later* approach [32]. That is, the functional design and the implementation of the system are accomplished and performance techniques are applied only in the last stages of the software lifecycle. Then, when performance metrics of these systems are poor, the redesign of the software system turns on to a very expensive task.

Software Performance Engineering (SPE) [32] has been proposed as a systematic, quantitative approach to construct software systems that meet performance objectives. SPE applied in early stages of the software development process helps to overcome these mentioned problems.

The growth of the Internet has led the emergence of a wide variety of Web-based applications, such as e-business, e-commerce, digital libraries or video on-demand. A Web application can be defined as an application delivered from a Web server to users over a network, such as the World Wide Web or an intranet.

A common characteristic of Web applications is that they receive millions of requests per day, that must be attended in a short term, since they are forced to provide the satisfaction requirements by their users.

Web applications, in which Web Services are included, are considered among the most difficult software systems to measure and estimate in general. Since the software components are distributed across a network without regard to their physical locations or the characteristics of the communication.

Quality of Service (QoS) can be defined as a set of perceivable characteristics expressed in an user-friendly language with quantifiable parameters that may be subjective or objective [37]. Examples of objectives parameters are startup delay or data sizes. Subjective factors are the overall cost or the factors of importance of other parameters. Examples of QoS parameters for system

[*]This report has been developed for the ”Tecnologías Middleware” PhD course.

resources are jitters, delays, blocking times and size of buffers [28]. The QoS of Web applications plays a crucial role in attracting and retaining users, and therefore, in their satisfaction level.

The goal of this work is to present a review of the works in the literature devoted to the performance evaluation of Web applications. This work will carry out an special attention about performance issues on two specific models of Web applications development: CORBA-based applications and Web Services.

The paper is structured as follows: in the next section, very basic concepts about Web applications are recalled. Section 3 introduces the main performance evaluation metrics that use to be applied for Web applications measurement. In Section 4 we summarize the state of the art. A brief introduction to SPE for Web Application is carried out in Section 5.

2 Web applications

A Web application is similar to a traditional client/server application, but it runs on a Web site [30]. Keys of their proliferation are the easiness to be updated and maintained, just by adding new functionalities. These processes use to be transparent to the user.

In the following, the basic concepts concerning the two keys models studied are recalled: CORBA-based distributed-object systems and Web Services.

2.1 CORBA-based distributed object systems

Distributed-object technology is the result of merging object-oriented techniques with distributed systems technology. Distributed systems are composed of several computing entities (distributed objects) to run a single task in a transparent way, so that they appear to users as a single, centralized system [30].

Interoperability between software systems, based on distributed-object technology, is enabled and supported by communication middleware mechanisms such as the Common Object Request Broker Architecture (CORBA) [27] standard, defined by the Object Management Group (OMG) [26]. The middleware provides a layer of services between the application and the underlying platform (operating system and network software).

Figure 1 depicts, the OMG's Object Management Architecture (OMA) that is composed of four elements [30]:

- *Object Request Broker* (ORB) is responsible for managing communication between objects without regard to: object location, implementation and state, or inter-object communication mechanisms. It is the core of this architecture. Some aspects of the ORB includes:
  - The interface definition language (IDL): declares the interfaces and types of operations and parameters.
  - Language mappings: specifies how the IDL features are mapped to various programming languages.

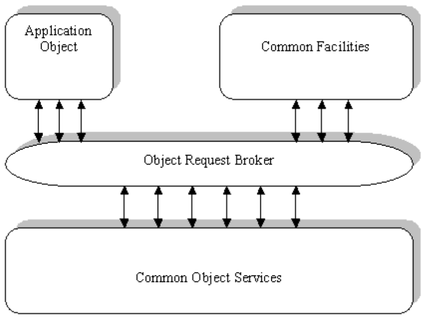

Figure 1: The OMG Object Management Architecture.

   - Client stubs and server skeletons: provide mechanisms for interacting with the ORB to convert (static) request invocations from the programming language into a form for transmission to the server object and to similarly handle the response.
   - Dynamic invocation and dispatch: a generic mechanism for dynamic request invocations without compile-time knowledge of object interfaces.
   - Protocols for inter-ORB communication: the mechanism for ORB-to-ORB handling of request invocations.

- *Common Object Services* are collections of system-level services packaged as components with IDL-specified interfaces. They augment and complement the functionality of the ORB.
- *Common Facilities* are collections of IDL-defined components that provide services of direct use to application objects.
- *Application Objects* are components specific to end-user applications.

2.2 Web Services

The term Web service describes specific business functionality offered by a company, usually through an Internet connection, for the purpose of providing a way for another company or software program to use the service [23].

Web Services are universally accessible software components deployed on the Web, therefore in a Web service architecture clients and services are loosely coupled and geographically distributed. One important characteristic is the heterogeneous and architecture-neutral of the computing platform. Therefore, one of the key ideas is that a Web service's implementation and deployment platform are not relevant to the application that is invoking the service [1].

A refined definition is supplied by the World Wide Web Consortium (W3C) [39]: *A software system identified by a URI*[1]*, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols.* [41].

A Web service is a collection of protocols and standards used for exchanging data between applications. Software applications written in various programming languages and running on various platforms can use Web Services to exchange data over computer networks like the Internet, in a manner similar to inter-process communication on a single computer. This interoperability is due to the use of open standards such as the following and others:

- The *Simple Object Access Protocol* (SOAP) [42] is a W3C draft note describing a way to use XML and HTTP to create information delivery and remote procedure mechanisms. It is a lightweight protocol for the exchange of information decentralized, distributed environment; therefore, it is a specification that defines a uniform way of passing XML-encoded data. Examples of transport protocols that may be used are HTTP, SMTP, FTP or POP3. Currently, the most used is HTTP over port 80, originally reserved by Web browsers.

- The *Web Service Description Language* (WSDL) [40] is an XML format for describing network services as a set of endpoint operating on messages containing either document-oriented or procedure-oriented information. WSDL is used to describe what services does, where they resides and how to invoke them.

- The *Universal Description Discovery and Integration* (UDDI) [36] specifications define a way to publish and discover information about Web Services. It consists of several related documents and an XML schema that defines a SOAP-based programming protocol for registering and discovering Web Services. UDDI is a framework for Web Services integration. From a performance perspective, the discovery process involves the time to look up the service in the Web Services directory using UDDI. It is possible to assume a more simplistic approach, where the client has prior knowledge of the service addressed.

An advantage of Web Services is a disadvantage at the same time. By utilizing HTTP (over TCP port 80), Web Services can work through many common firewall security measures without requiring changes to the filtering rules; since port 80 is always open, because it is used for Web browsers. However, this means that Web Services can evade existing firewall security measures whose rules are intended to block or audit communication between programs on either side of the firewall.

[1]Uniform Resource Identifier

3 Characterizing Performance and QoS of Web applications

The most Web environments are complex software systems. They are composed of firewalls, dynamic services or security techniques. Therefore, a myriad of options should be taken into account in order to carry out their performance analysis: kind of Web application (distributed object-oriented vs service-oriented), implementation technologies used in their development(e.g. Enterprise Java Beans or ActiveX controls) or design patterns used in their conception. All of these choices directly affect their performance as well as their QoS attributes.

The *Service-level agreements* [23] (SLAs) are defined as legally binding contracts that establish bounds on various QoS metrics. Performance metrics are matched against the SLAs to determine if the capacity of the system is adequate. In other words, the values of SLAs determine the level of acceptable service from users perspective.

The most important performance parameters commonly used to measure Web applications are the following [23]:

Response time is the time a system takes to respond to various types of requests. It is a function of load intensity, which can be measured in terms of arrival rates (such as request per second) or number of concurrent requests. QoS uses to take into account not only the average response time, but also the percentile of the response time.

Throughput is the rate at which a system or service can process requests, i.e. the request is completely processed by a computer system. It is measured in number of operations by unit of time. The throughput is a function of the load assigned to a system and of the maximum capacity of the system to process work.

Availability is the fraction of time that a system is operating and available to its customers. The two main reasons for systems to be unavailable are failures and overloads.

Cost is usually associated with some measure of performance, i.e. response time or throughput, as a price-performance ratio.

The performance perception of the users of the Web applications differs from the providers perception. In the case of the users, it has to do with fast response time or not connections refused. Service providers also concern about high throughput and high availability.

Web applications have some unique characteristics that distinguish them from traditional distributed systems. Some of these characteristics have a profound impact on its performance. First, the number of users can be constantly increasing. On the other hand, the randomness associated with the way that users visit sites makes the problem of workload forecasting and capacity planning very difficult.

The QoS of a Web application uses to be indicated by a combination of the parameters and qualities previously defined. Besides of the above-mentioned performance metrics, there are qualities or properties that characterize QoS, such as [23, 19, 21, 24]:

Security properties include the existence and type of authentication mechanisms; confidentiality and data integrity of messages exchanged; non - repudiation of requests or messages, and resilience to denial-of-service attacks.

Reliability of a system is the probability that it works properly and continuously over a fixed period of time. When the time period during which the reliability is computed becomes very large, the reliability tends to the availability.

Scalability of a system is defined when its performance does not degrade significantly as the number of users, or equivalently, the load on the system increases.

Extensibility is the property of a system to easily evolve to cope with new functional and performance requirements.

3.1 Performance Issues in CORBA-based distributed object systems

Concerning CORBA-based distributed object systems and focussing on the performance characteristics of the ORB, there are two main issues: those that do not need to be explicitly modeled and those that do [33].

Aspects of the ORB that do not require explicit modeling include: IDL, language mappings, client stubs and server skeletons, dynamic invocation and dispatch and protocols for inter-ORB communication, such as Internet-ORB protocol (IIOP). These features affect the overall performance of the system, they are modeled implicitly by measuring their resource requirements and including them as *processing overhead* for each invocation of a server object. This overhead, that includes delays due to CORBA-based process coordination, is a significant portion of the total end-to-end time to process an operation [33].

However, there are five aspects of the ORB that have to be explicitly modeled in order to calculate performance measures: object location, process composition, request scheduling, request dispatching and coordination mechanisms. Object location and process composition determine the processing steps assigned to each performance scenario. Request scheduling and request dispatching are partially determined by contention for called processes which is determined by the coordination mechanism and other processing requirements of performance scenarios.

3.2 Performance Issues in Web Services

When a Web service is measured a number of topics have to be taken into account. In the following, the most important are remarked [10, 4, 5]:

- Discovering the Web Service. This involves the time to look up the service in the Web service directory using UDDI. It can include the authentication process of the users.
- Transporting SOAP messages. They are independent of the transport protocol, but they introduces a great overhead.
- Extracting the XML-defined Web service request from SOAP envelope (conversely, at the client side this involves packing the XML request in the SOAP envelope).
- Parsing the XML document. Due to SOAP messages must be parsed and interpreted before they can be invoked, it is necessary a XML parser, such as DOM or SAX. Different implementations of the parsers may also have an impact.
- Usually, Web Services lure users with contents, such as graphics, audio and video, which alter the distribution patterns of network traffic. They affect negatively the Web performance [23].
- Composing Web Services. Dynamic composition of Web services and its corresponding invocation method provides new functionalities, but it modifies the performance study [6, 21, 20]
- Web applications, and specially Web Services, exhibit high variability in workload characterization. This means that the workload seems to be well modeled by a distribution with infinite variance. Such distributions are called heavy-tailed [23, 7].

As a conclusion, it could be stated that the flexibility and interoperability of the XML based protocol has been achieved partly at the expense of performance.

4 Performance Works about Web applications measurements

In the last years, a vast literature has arisen related to performance studies of Web applications and the platforms where they are developed and executed.

In this section, a number of these studies are revised in order to offer a comprehensive and up-to-date background about the techniques, models, methods, problems and challenges reported in them and currently present in this area. Since these great number of works has been accomplished using different techniques and methods, then in the following two classifications, based on what and how to measure, are established. The classification will be useful to conduct the rest of the section and it is summarized in the Table 1.

	Architecture and communication protocols	Software	
		SPE	Others
No formal models	[13, 14, 2, 3, 7, 8, 35, 10, 25, 43, 15, 12]	[6]	
Formal models	[22, 31, 23, 19, 4, 11, 24, 21, 20]	[33, 34, 38, 5]	[18]

Table 1: Works concerning Web measurement

- **What to measure**:
  - Architecture and communication protocols: These works measure different aspects of middleware technologies (mainly CORBA) and communications protocols (such as IIOP or SOAP).
  - Software systems: They try to measure specific Web applications considering the overload introduced by the architecture and the communication protocols. A few of them follow the techniques proposed by the SPE [32].
- **How to measure**:
  - No model is used: Usually, they directly measure the system and/or use benchmarks over the network.
  - A model is used: They model the system (or part of it) and/or the execution platform. The model is said formal when a modeling formalism is chosen as target specification. Among relevant formalisms are Queueing Networks (QN), Layered Queueing Networks (LQN), Stochastic Petri Nets (SPN) and Stochastic Process Algebra (SPA).

4.1 Evaluation of architecture and communication protocols without modeling techniques

The works of Crovella et al.[2, 3, 7, 8] study the Web workload characterization, just as its implications in performance. They conclude that most of the Web workloads exhibit high variability. They can be well modeled by a distribution with infinite variance, called *heavy-tailored* in contrast to distributions commonly encountered in computer and telecommunication systems modelling. The obtained results are compared with empiric values measured over the network, showing the validity of these approaches.

The next works are focussed on different implementation and improvements of the CORBA from a performance point of view. A methodology for improving the performance of an implementation of the Internet Inter-ORB Protocol (IIOP) is described by Gokhale and Schmidt in [13]. The same authors in [14] measure the performance of CORBA in High-Speed Networks. Also, they study the performance on dependable CORBA middleware [12]. Testbed environments are used to model Web behavior and in order to measure performance in all of these approaches.

A benchmarking methodology for evaluating performance of CORBA middleware is presented by Tuma and Buble [35]. They compare different implementations of CORBA and their performance over the network.

Mishra and Shi [25] propose techniques to improve the performance of distributed CORBA applications. They design and implement a middleware system, called *CORBA-as-needed* that provides all necessary CORBA functionality except interoperability. The proposed implementation uses TCP for communication instead of CORBA communication methods. Using this technique the performance extra overhead in interoperability is decreased.

Elfwing et al. [10] compare SOAP performance to CORBA from the part of the communication between the provider and the consumer of a Web service. Taking into account the SOAP protocol characteristics, they propose improvements in its implementation, such as usage of SOAP over HTTP 1.0 or 1.1. The main conclusion from this work is that SOAP is significantly slower compared to CORBA.

The approach of Zou et al. [43] investigates the overhead of real-time CORBA invocations against TCP/IP socket programs and end-to-end predictability or real-time CORBA. For this aim, they configure a testbed environment in order to measure the throughput and latency of CORBA invocations.

Harkema et al. [15] focus on the development of a quantitative performance model for CORBA-based method invocations. They observe that the model features such as a finite thread pool and deterministic service times prohibit a straightforward application of analytic techniques, such as LQN. So, they model by means of simulation over a test lab environment.

4.2 Evaluation of architecture and communication protocols with modeling techniques

Menascé and Almeida [22, 23, 24] study the factors that affect the Web performance and Web services, between them the workload to attain the optimum capacity planning. The Web services are modeled by using the QN formalism. Also, Menascé study both the response time analysis of composite of Web service and their QoS issues in [21, 20] and [19]. The interest of these approaches consist in the modeling of the workload and the entire software system.

Petriu et al. [31] apply the LQN model to predict middleware performance of CORBA-based applications. They base the study on the middleware interaction architectures; i.e., the way a request from the client is routed to the server.

Brodo et al. [4] model a Web Service by using PEPA nets to analyze the dynamic behavior and to predict performance indices such as throughput, response time or utilization.

The approach of Fernandes et al. [11] evaluate Message-Oriented Middleware protocols, more specifically an implementation called WebSphere [16], through Generalized-SPN (GSPN) and their impact in the performance. Moreover, they present an alternative model in order to improve the overall behavior. Based on these models, numerical results are obtained by a simulation tool.

4.3 Evaluation of software systems without modeling techniques

From the best of our knowledge, only one work [6] focusses on this category. Chandrasekaran et al. They discuss techniques for process execution time analysis that can be used to evaluate the performance of individual Web services and how to integrate them in composition. Taking into account these techniques, they implement a tool which automatically generates an execution code. This tool is integrated in a Java-based simulation and animation environment that supports several features to simulate Web processes.

4.4 Evaluation of software systems with modeling techniques

Among these approaches, the most relevant works are those proposed by Smith and Williams [33, 34]. They establish the main guidelines to apply SPE techniques in the Web applications development process by means of extensions of the proper SPE. Due to the importance of the proposed modelling guidelines, a more detailed explanation is shown in the next section.

Liu et al. [18] propose an approach to predict performance metrics for a middleware-hosted application. They produce a QN model, in order to predict the performance values.

Catley et al. present in [5] a case study of Web services applying SPE techniques and based on the UML Profile for Schedulability, Performance and Time (UML-SPT) [29]. The performance analysis is made by using LQN.

Verdickt et al. [38] propose an Model Driven Architecture (MDA) model transformation for Platform Specific Model (PSM), including middleware performance details. It is based on SPE and the UML-SPT profile. The transformation process is made by an automatic tool which generates LQN performance models.

5 SPE for Web applications

The process of constructing software systems that meet performance objectives is called Software Performance Engineering (SPE). For this reason, the SPE techniques should be integrated into the early stages of the software development life cycle for evaluating performance metrics before constructing them.

The SPE techniques for Web applications [34] are similar to those that used for CORBA-based distributed-object systems [33]. Web applications use different implementation technologies than other distributed systems, but implementation details do not affect the SPE models during their life cycle.

The first step in the SPE process is to establish the performance objectives, i.e. the quantitative criteria for evaluating the performance characteristics of the system under development. Performance objectives should avoid vague statements such as *the system response time should be satisfactory to end users*, they should be stated in a precise manner, such as *the response time of the patient information system should not exceed one second for local users.*

The next step is to identify important Uses Cases, i.e., those that are critical to the operation of the system or which are important to responsiveness as seen by the user. Next, it is necessary to select a set of performance scenarios that represent the important (Web) application interactions, those which are executed frequently or those which are critical to the perceived performance of the system. They also include the important functions that must perform well if the application is to meet its business objectives. After that, an end -to -end performance scenario is created that represents (at a high level) the processing steps in each of the performance scenarios. Extended UML sequence diagrams are used to represent the system interactions.

After representing the overall flow, the processing steps that have the greatest impact on performance are identified and detailed. The next step, it is to convert the sequence diagram into a software execution model, add performance specifications, and solve the software execution model to determine the end-to-end response time (without contention delays). The software execution model often identifies problems with Web applications, particularly when they need to transfer large amounts of data over relatively slow communication lines.

Due to the Web execution environment is typically complex, at the architectural level of design, first we will use deliberately simple models of software processing that are easily constructed and solved to provide feedback. After correcting any problems, the delays to interact with another scenarios in the model are introduced.

The model solutions are iterative, since the solution of each independent performance scenario quantifies its processing time, which serves as the delay for system interactions in the subsequent model.

References

[1] Gustavo Alonso, Fabio Casati, Harumi Kuno, and Vijay Machiraju. *Web Services. Concepts, Architectures and Applications.* Springer, 2004.

[2] Paul Barford and Mark Crovella. Generating Representative Web Workloads for Network and Server Performance Evaluation. In *Joint International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS '98 / PERFORMANCE '98. Performance Evaluation Review 26(1)*, pages 151–160, Madison, Wisconsin, USA, June 22-26 1998.

[3] Paul Barford and Mark Crovella. Measuring Web performance in the wide area. *SIGMETRICS Performance Evaluation Review*, 27(2):37–48, 1999.

[4] Linda Brodo, Pierpaolo Degano, Stephen Gilmore, Jane Hillston, and Corrado Priami. Performance Evaluation for Global Computation. In Corrado Priami, editor, *Global Computing*, volume 2874 of *Lecture Notes in Computer Science*, pages 229–253. Springer, 2003.

[5] Christina Catley, Dorina C. Petriu, and Monique Frize. Software Performance Engineering of a Web service-based Clinical Decision Support infrastructure. In Dujmovic et al. [9], pages 130–138.

[6] Senthilanand Chandrasekaran, John A. Miller, Gregory S. Silver, Ismailcem Budak Arpinar, and Amit P. Sheth. Performance Analysis and Simulation of Composite Web Services. *Electronic Markets*, 13(2), 2003.

[7] Mark Crovella. Performance Characteristics of the World Wide Web. In Günter Haring, Christoph Lindemann, and Martin Reiser, editors, *Performance Evaluation*, volume 1769 of *Lecture Notes in Computer Science*, pages 219–232. Springer, 2000.

[8] Mark Crovella. Performance Evaluation with Heavy Tailed Distributions. In Boudewijn R. Haverkort, Henrik C. Bohnenkamp, and Connie U. Smith, editors, *Computer Performance Evaluation / TOOLS*, volume 1786 of *Lecture Notes in Computer Science*, pages 1–9. Springer, 2000.

[9] Jozo J. Dujmovic, Virgílio A. F. Almeida, and Doug Lea, editors. *Proceedings of the Fourth International Workshop on Software and Performance, WOSP 2004, Redwood Shores, California, USA, January 14-16, 2004*. ACM, 2004.

[10] Robert Elfwing, Ulf Paulsson, and Lars Lundberg. Performance of SOAP in Web Service Environment Compared to CORBA. In *APSEC*, pages 84–. IEEE Computer Society, 2002.

[11] Stênio F. L. Fernandes, Wellington J. Silva, Mauro J. C. Silva, Nelson S. Rosa, Paulo R. M. Maciel, and Djamel F. H. Sadok. On the Generalised Stochastic Petri Net Modelling of Message-Oriented Middleware Systems. In *IPCCC*. IEEE, 2004.

[12] Aniruddha S. Gokhale, Balachandran Natarajan, Douglas C. Schmidt, and Joseph K. Cross. Towards Real-Time Fault-Tolerant CORBA Middleware. *Cluster Computing*, 7(4):331–346, 2004.

[13] Aniruddha S. Gokhale and Douglas C. Schmidt. Measuring the performance of communication middleware on high-speed networks. In *SIGCOMM*, pages 306–317, 1996.

[14] Aniruddha S. Gokhale and Douglas C. Schmidt. Principles for Optimizing CORBA Internet Inter-ORB Protocol Performance. In *HICSS '98: Proceedings of the Thirty-First Annual Hawaii International Conference on System Sciences-Volume 7*, page 376, Washington, DC, USA, 1998. IEEE Computer Society.

[15] Marcel Harkema, Bart M.M. Gijsen, Robert D. van der Mei, and Y. Hoekstra. Middleware Performance: A Quantitative Modeling Approach. In

Proc. of the international Symposium of Performance Evaluation of Computer and Telecommunication Systems (SPECTS), San Jose, USA, July 2004.

[16] IBM, `http://www-306.ibm.com/software/websphere/`. *WebSphere*, May 2005.

[17] E.D. Lazowska, J. Zahorjan, G. Scott, and K.C. Sevcik. *Quantitative System Performance: Computer System Analysis Using Queueing Network Models.* Prentice-Hall, 1984.

[18] Yan Liu, Alan Fekete, and Ian Gorton. Predicting the performance of middleware-based applications at the design level. In Dujmovic et al. [9], pages 166–170.

[19] Daniel A. Menascé. QoS Issues in Web Services. *IEEE Internet Computing*, 6(6):72–75, 2002.

[20] Daniel A. Menascé. Composing Web Services: A QoS View. *IEEE Internet Computing*, 8(6):88–90, 2004.

[21] Daniel A. Menascé. Response-Time Analysis of Composite Web Services. *IEEE Internet Computing*, 8(1):90–92, 2004.

[22] Daniel A. Menascé and Virgilio A. F. Almeida. *Capacity planning for Web performance: metrics, models, and methods.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1998.

[23] Daniel A. Menascé and Virgilio A. F. Almeida. *Capacity Planning for Web Services: metrics, models, and methods.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001.

[24] Daniel A. Menascé, Virgilio A. F. Almeida, and Lawrence W. Dowdy. *Performance by Design: Computer Capacity Planning By Example.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004.

[25] Shivakant Mishra and Nija Shi. Improving the performance of distributed corba applications. In *IPDPS*. IEEE Computer Society, 2002.

[26] Object Management Group, `http:/www.omg.org`.

[27] Object Management Group, `http:/www.omg.org`. *The Common Object Request Broker: Architecture and Specification*, December 2002. version 1.0.

[28] Object Management Group, `http:/www.omg.org`. *UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms.*, September 2004. version 1.0.

[29] Object Management Group, `http:/www.omg.org`. *UML Profile for Schedulabibity, Performance and Time Specification*, January 2005. version 1.1.

[30] Robert Orfali, Dan Harkey, and Jeri Edwards. *The Essential Distributed Objects Survival Guide.* John Wiley, 1996.

[31] Dorina C. Petriu, Hoda Amer, Shikharesh Majumdar, and Istabrak Abdul-Fatah. Using analytic models predicting middleware performance. In M. Woodside, H. Gomaa, and D. Menascé, editors, *Workshop on Software and Performance*, pages 189–194, Ottawa, Canada, September 17-20 2000. ACM.

[32] Connie U. Smith. *Performance Engineering of Software Systems.* The Sei Series in Software Engineering. Addison–Wesley, 1990.

[33] Connie U. Smith and Lloyd G. Williams. Performance Engineering Models of CORBA-based Distributed-Object Systems. In *Int. CMG Conference*, pages 886–898. Computer Measurement Group, 1998.

[34] Connie U. Smith and Lloyd G. Williams. Building Responsive and Scalable Web Applications. In *Int. CMG Conference*, pages 127–138. Computer Measurement Group, 2000.

[35] Petr Tuma and Adam Buble. Overview of the CORBA Performance. In *Proc. of the International Symposium of Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*, Orlando, USA, July 2001.

[36] Universal Description, Discovery, and Integration of Business for the Web (UDDI), `http://www.uddi.org`.

[37] Nalini Venkatasubramanian and Klara Nahrstedt. An integrated metric for video QoS. In *MULTIMEDIA '97: Proceedings of the fifth ACM international conference on Multimedia*, pages 371–380, New York, NY, USA, 1997. ACM Press.

[38] Tom Verdickt, Bart Dhoedt, and Frank Gielen. Incorporating SPE into MDA: including middleware performance details into system models. In Dujmovic et al. [9], pages 120–124.

[39] World Wide Web Consortium (W3C), `http://www.w3.org`.

[40] World Wide Web Consortium (W3C), `http://www.w3.org/TR/wsdl`. *Web Services Description Language (WSDL)*, March 2001.

[41] World Wide Web Consortium (W3C), `http://www.w3.org/TR/wsa-reqs`. *Web Services Architecture Requirements*, October 2002.

[42] World Wide Web Consortium (W3C), `http://www.w3.org/TR/soap`. *Simple Object Access Protocol (SOAP)*, June 2003.

[43] Jianfan Zou, David Levy, and Anna Liu. Evaluating Overhead and Predictability of a Real-Time CORBA System. In *HICSS '04: Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 9*, page 90276.1, Washington, DC, USA, 2004. IEEE Computer Society.