



The Society for Modeling and
Simulation International

SCS Founder: John McLeod

SCS Executive Committee

- John (Drew) Hamilton Jr., President
Auburn University
François Cellier, Past President
University of Arizona
Mohammad S. Obaidat, Senior Vice President
Monmouth University
David Cook, V.P., Conferences
Aegis Technologies Group, Inc.
Michael J. Chinni, V.P., Membership
U.S. Army - RDECOM - ARDEC
Bernard Zeigler, V.P., Publications
University of Arizona
William V. Tucker, V.P., Education
The Boeing Company
Levent Yilmaz, Secretary
Auburn University
Richard Mac Donald, Treasurer
RAM Laboratories, Inc.

SCS Board of Directors

- Ralph Huntsinger, F.S.C.S., Historian
California State University, Chico;
Technical University of Bialystok, Poland

Regional Council Directors:

Eastern Council

- Taieb Znati, *University of Pittsburgh*

Midwestern Council

- William McQuay, *WPAFB*

Southeastern Council

- Ratan Guha, *UCF*

Southwestern Council

- David Cook, *Aegis Technologies*

Western Council

- Richard MacDonald, *RAM Laboratories*

European Council

- Rainer Rimane, *University of Erlangen*

Chinese Council

- Tianyuan Xiao, *Tsinghua University*

Pacific Rim Council

- Tag Gon Kim, *KAIST*

Latin-American

- Miguel Angel Piera, *Autonomous University of Barcelona*

International Director

- Sergio Junco, *National University of Rosario*

Directors at Large

- David Cook, *Aegis Technologies Group, Inc.*
Claudia Frydman, *Université Paul Cézanne*
Tuncer Ören, *University of Ottawa*
Jerzy Rozenblit, *University of Arizona*
William V. Tucker, *The Boeing Company*
Bill Waite, *Aegis Technologies Group, Inc.*
Levent Yilmaz, *Auburn University*
Bernard Zeigler, *University of Arizona*
Richard MacDonald, *RAM Laboratories, Inc.*

SCS Staff

- Stephen Branch, *Executive Director*
Christine Wasserman, *Managing Editor*
Brandon LaFlair, *Shipping/Receiving*
Oletha Darensburg, *Accountant*
Suzette McLeod, *Assistant Managing Editor*

SIMULATION

Transactions of The Society for Modeling and Simulation International

January 2007

Volume 83 Number 1

Special Issue on Modeling and Simulation for and in Service-Orientated Computing Paradigm

Modeling and Simulation for and in Service-Orientated Computing Paradigm <i>Y. Chen</i>	3
Modeling and Simulation in Service-Oriented Software Development <i>W. T. Tsai, Z. Cao, X. Wei, R. Paul, Q. Huang, and X. Sun</i>	7
A Petri Net-based Approach for Automated Goal-Driven Web Service Composition <i>D. Zhovtobryukh</i>	33
CSP-Based Verification for Web Service Orchestration and Choreography <i>W. L. Yeung</i>	65
Modeling the Measurements of QoS Requirements in Web Service Systems <i>W. D. Yu, R. B. Radhakrishna, S. Pingali, and V. Kolluri</i>	75
Modeling and Simulation of QoS-aware Web Service Selection for Provider Profit Maximization <i>D. Tsesmetzis, I. Roussaki, and E. Sykas</i>	93
Quality of Service Measures of Mobile Ad-hoc Wireless Network using Energy Consumption Mitigation with Asynchronous Inactivity Periods <i>C. X. Mavromoustakis and H. D. Karatza</i>	107
A Pipeline Virtual Service Pre-Scheduling Pattern and its Application in Astronomy Data Processing <i>M. Wang, Z. Du, Z. Cheng, and S. Zhu</i>	123

<http://sim.sagepub.com>

Free access to tables of contents and abstracts. Site-wide access to the full text for members of subscribing institutions.

Modeling and Simulation for and in Service-Oriented Computing Paradigm

Yinong Chen
Arizona State University
Tempe
Arizona
United States of America
yinong@asu.edu

1. Service-Oriented Computing Paradigm

Service-Oriented Computing (SOC), including Service-Oriented Architecture (SOA), Web services (WS), and their data and processing models, emerges as a new computing and development paradigm that changes the way we develop and use computer software. Major computer corporations, including HP, IBM, Intel, Microsoft, Oracle, Sun Microsystems, and SAP, just to mention a few, are moving into and supporting this new paradigm by offering service-oriented software development tools, languages, protocols, and services. Standard organizations, such as ISO, OASIS, and W3C, have recommended numerous standards to make services platform-independent and interoperable. Under this new paradigm, all computer programs or components can be given a standard programming interface and be stored in an internet-searchable repository as reusable units for application composition. The accumulation of available services makes it possible to compose systems and applications completely based on existing services.

SOC is different from object-oriented and component-based computing in several significant ways. SOC explicitly differentiates service providers, service brokers, and service requesters (application builders). As a result of the independence among these three parties, standard-based interfacing and interoperability become a necessity. SOC does not require the integration of components and code into one computing environment. The relationships among the constituting components are service exchanges, which deliver the data required, instead of importing the code that processes the data. This loosely coupled architecture makes platform-independent computing

possible. Furthermore, SOC uses Extensible Markup Language (XML) and the derived ontology languages, such as RDF (Resource Description Framework) and OWL (Web Ontology Language), to describe data and metadata, which makes information not only readable but also understandable by machines. These features further enable the dynamic architecture of service-oriented software, in which coordination from the central process to the constituent services, also known as orchestration, the coupling topology among services, also known as choreography, as well as binding protocols and addresses, can be redefined and changed dynamically at runtime.

2. Modeling, Simulation and Service Orientation

Why would modeling and simulation of SOC systems and applications differ from that of the current distributed systems? SOC systems and applications belong to distributed systems and their modeling and simulation certainly fall into the research and practice of distributed system modeling and simulation. However, unique issues and problems arise with the new computing paradigm, as mentioned in Section 1. The current research in distributed modeling and simulation needs to be extended to address these issues and problems. Distributed computing is challenging because of the parallelism, non-determinism, communication, synchronization, and interlocking problems. SOC adds a number of dimensions to these already complex problems. On the one hand, modeling and simulation are perfect techniques to address such complex problems, where analytic or prototyping techniques are either infeasible or too expensive. On the other hand, the current modeling and simulation techniques are limited in capacity and complexity, which need to be addressed or reinvented using a service-oriented paradigm. Thus, the research questions can be categorized into two directions: Using modeling and simulation techniques to address the design and development problems in SOC systems and us-

ing a SOC paradigm to extend the capacity of modeling and simulation techniques and frameworks.

2.1 Modeling and Simulating Service-Orienting Systems and Applications

Due to these unique features in SOC, current modeling and simulation techniques need to be extended to address these new issues.

Modeling and simulation of systems of systems. This is a general question that exists not only in SOC systems. However, SOC facilitates the nested composition of services and systems, which makes large assemblies of systems easier to compose and to manage. Modeling and simulation techniques need to be scaled to beyond their current limits.

Dynamic architecture is a unique feature in SOC systems. This feature meets the dynamic requirements of current business models and empowers the integration of their models with the underlying computer models. Simulation can be used to enumerate the foreseeable and to generate unforeseeable scenarios of dynamical behaviors. These problems are beyond the traditional reconfiguration problems, where functionally equivalent components are used to replace malfunctioned components. Dynamic architecture allows services with different functions to substitute existing services, new services to be added into an application, and restructuring among the services in terms of orchestration or choreography.

Dependability and quality of services both refer to the property of a system by which reliance can justifiably be placed on the services it delivers. However, dependability is more from the developer's point of view, while the quality of services is more from the user's point of view. A service-oriented system is developed independently by parties involved in service provision, service brokerage, and application building, and thus the independent control of dependability and quality of services is more critical. Simulation can play a much more important role in estimating the dependability and quality of services. Since a service-oriented system can be dynamically composed or modified, dynamic simulation that reflects the changing system is necessary.

2.2 Modeling and Simulation in Service-Orienting Computing Paradigm

Service-oriented computing inherits all the features from object-oriented and component-based computing. Adding the concept of service orientation into modeling and simulation may significantly improve the capacity and the effectiveness of modeling and simulation techniques.

Service-oriented simulation components. Many simulation components and systems have been implemented. The concept of service orientation can help the heterogeneous components to interoperate and to cooperate in a loosely coupled manner. Wrapping or servicetizing a standard interface on each component can quickly achieve a basic level of interoperation, where data and results can be exchanged. However, manual interpretation and application of the data and results are necessary at this basic level. To automate the cooperation, metadata for markup needs to be added so that the cooperation can be established based on the semantics defined in the metadata. Ontology languages such as RDF and OWL and their development frameworks can facilitate such research and implementation.

Service-oriented simulation composition. Once the components are servicetized, simulation applications can be composed using the services. Current composition languages are domain specific. For example BPEL (Business Process Execution Language) is widely used for composition in service-oriented business applications. Microsoft VPL (Visual Programming Language) is a drag-and-drop enabled graphic language used for composing service-oriented robotics applications. A visual composition language facilitating service-oriented simulation composition and application building can help simulation builders, who may not be software developers, to build their simulations without learning detailed programming.

Service-oriented simulation framework. Using a SOC paradigm, one can build frameworks and development environments that support dynamically reconfigurable and recompose-able simulations. It is most likely that only the frameworks and development environments implemented in a SOC paradigm can adequately support the simulation of service-oriented systems and applications. Such frameworks and environments should also support the testing, data collection, and the evaluation of dependability and quality of services.

3. Coverage of the Special Issue

This is the first special issue of the journal dedicated to the modeling and simulation of service-oriented systems and applications. A specific review and evaluation form has been used to help the reviewers to select the papers in the target domain. As the guest editor, I have read every paper submitted to the special issue and rigorously selected the papers that fit well in the specified topic areas based on the reviewers and my own judgment. The selected papers do not cover all the areas that overlap with modeling/simulation and service orientation, but present an excellent coverage in both breadth and depth, which can

serve as a starting point to inspire the research in the areas as discussed in Section 2. The following subsections scan the papers in this special issue. The first two papers focus on the compositional development processes of service-oriented software in model- and goal-driven approaches, followed by a paper that verifies the model features of workflows. Then, the next two papers study the evaluation of quality of services and quality-aware composition in service-oriented software. The last two papers explore the applications of quality of services in energy-sensitive wireless networks and in time-consuming astronomy data processing systems. The summaries of these papers serve as a quick guide to this special issue and are purely based on my reading and understanding of the papers. They may not correctly or accurately reflect the authors' work.

3.1 Modeling and Simulation in Service-Oriented Software Development

In this paper, Tsai, et al. first give a fairly extensive overview to the role of modeling and simulation in service-oriented software development, by comparing and contrasting traditional distributed simulation and service-oriented simulation. Then a model-driven approach is proposed to dynamically generate simulation based on the specification written in the Process Specification and Modeling Language (PSML). Dynamic architecture in SOC is in fact a model-driven approach, and thus the proposed simulation approach can potentially support the simulation of dynamic architecture. Finally, the paper presents a case study that demonstrates the entire development process consisting of modeling, automated simulation code generation, simulation execution, and simulation results.

3.2 A Petri Net-based Approach for Automated Goal-Driven Web Service Composition

In this paper, Zhovtobryukh starts with an introduction to service-oriented architecture, Web services, Web services composition, and the related technologies. In the main part of the paper, the workflows of the composite Web services are modeled by Petri nets graphically and their mathematical meaning in algebra. The properties, such as parallelism, mutual exclusion, order of execution, liveness, and attainability are studied in detail. These studies lead to the further exploration of goal-driven composition of Web services, aimed at achieving on-demand composition, context-awareness, verifiable composition, and automated composition. Finally, the paper presents the development process using the modeling and goal-driven approaches and its application in an illustrative example.

3.3 CSP-Based Verification for Web Service Orchestration and Choreography

In Zhovtobryukh's paper, Petri nets are used to model and verify the dynamic behaviors of composite Web services. In this paper, Yeung studies the behavioral consistency and the verification of Web services modeled in CSP (Communicating Sequential Processes), which is a language used to describe the process in terms of possible interactions with its environment. Web services composed in orchestration in BPEL and in choreography in WS-CDL and their verification processes are also studied, in a way that these processes are mapped to CSP and verified in CSP. Orchestration in BPEL and choreography in WS-CDL are the two major compositional styles used in business composition. A digital library example is used throughout the paper to illustrate the modeling and verification processes in BPEL, WS-DSL, and CSP.

3.4 Modeling the Measurements of QoS Requirements in Web Service Systems

Yu and his colleagues present a variety of techniques that can be used to measure, to optimize, and to tradeoff the major Quality of Service (QoS) requirements in composite Web services, which can be incorporated into the planning, design, implementation, deployment, operation, and maintenance phases during the Web service development lifecycle. The paper studies the ten major QoS requirements defined by W3C, including performance, reliability, scalability, robustness, accuracy, integrity, availability, accessibility, interoperability, and security. A software model is designed and implemented to test Web services, to perform the experiments, and to quantitatively measure the QoS requirements involved in the Web services. For each of the QoS requirements, a list of available improvement techniques is suggested to enhance the level of QoS in Web services.

3.5 QoS-aware Web Service Selection for Provider Profit Maximization

In this paper, Tsesmetzis et al. assume the QoS parameters have been measured and propose to include these measures as a part of the Web services description, which enables the service requesters to consider QoS factors when dynamically discovered and bind services into a live application. To standardize the QoS terminologies, the authors suggest the definition of a set of QoS elements and attributes to form a QoS ontology in RDF and to develop an ontology framework for service requesters to access the services with QoS indicators. Once such an ontology is available, service requesters can select the Web services in their application building process based on the optimization among the QoS parameters such as price and bandwidth. An efficient algorithm is presented in the paper and

extensive simulation results are given to illustrate the performance of the algorithm.

3.6 Quality of Service measures of Mobile Ad-hoc Wireless Network

This is a paper applying QoS measures in energy-sensitive wireless networks. Mavromoustakis and Karatza study in this paper the QoS measures and their impacts to an ad hoc wireless network. They use the Adaptive Energy Conservation (ADEC) protocol to manipulate the network states in a distributed and an on-demand form, to control each node's inactive state duration adaptively, and thus to mitigate the energy consumed for the peer-to-peer communication among mobile terminals. This is a cost-effective scheme of sending delay-sensitive packets or streams of packets over a network, which guarantees an error free transmission by buffering of data and overcomes the loss of data packets in an energy-efficient way. The QoS of the scheme in terms of the performance is simulated and the results show that, if a node has relatively large caching capacities, the system will behave significantly better in

terms of the node's utilization for both periodic and non-periodic caching schemes.

3.7 A Pipeline Virtual Service Pre-Scheduling Pattern and Its Application

This is another application paper. In this paper, Wang et al. study the optimum composition of the workflows. A pipelined scheduling algorithm is proposed, which overlaps the execution of independent services in the workflows in the Open Grid Service Architecture environment. The performance and the application flexibility of this algorithm are studied before simulation results are presented. Finally, the paper presents the prototype which implements the pipelined scheduling algorithm and the application of the prototype in an Astronomy Data Processing system. The experiments and data collected after the prototype is installed in the telescope data process system show that the total time of processing data packets is significantly reduced in comparison with that before the prototype is installed.