A Discrete-Events Simulation Approach for Evaluation of Service-Based Applications

Maha Driss[1,2], Yassine Jamoussi[1], Jean-Marc Jézéquel[2] and Henda Hajjami Ben Ghézala[1]
[1]National School of Computer Sciences (ENSI)
RIADI-GDL Laboratory
University of Manouba, Tunisia
yassine.jamoussi@ensi.rnu.tn
henda.benghezala@ensi.rnu.tn
[2]IRISA/INRIA, Rennes, France
maha.driss@irisa.fr
jezequel@irisa.fr

Abstract

One of the promises of the Service-Oriented Architecture (SOA) is to build complex added-value services in order to enhance and extend existing ones. Service-Based Applications (SBAs) are asked not only to perform required functionalities, but also to deliver expected level of Quality of Service (QoS). Dealing with QoS management of such distributed applications, which are executed in dynamic environments, raises the need to consider context characteristics. This paper proposes a discrete-events simulation approach which assures the evaluation of SBAs performance under different context status. The main contributions of this paper are: (i) the discrete-events modeling approach for SBAs, (ii) the context-based model for SBAs considered in the discrete-events simulation model, and (iii) the evaluation of a set of QoS metrics by simulation.

1. Introduction

Web services (WSs) have emerged as a major technology for deploying automated interactions between heterogeneous systems. They are based on a collection of standards and protocols that allow processing requests to remote systems by speaking a common, non-proprietary language and using common transport protocols. WSs are becoming a popular vehicle for organizations that need to integrate their applications within and across organizational boundaries. The process-based composition of WSs is gaining a considerable momentum as an approach for the effective integration of distributed, heterogeneous, and autonomous applications [12]. One of the major requirements for such Service-Based Applications (SBAs) is the Quality of Service (QoS). QoS for WSs refers to various non-functional characteristics such as response time, throughput, availability and security [8]. Since many WSs provide overlapping or identical functionality, different WS processes can be composed satisfying the same user requirement. A choice needs to be made to determine which process is to use to provide the more beneficial QoS. Also, providing the acceptable QoS is really a challenging task due to the dynamic and unpredictable nature of the execution environment (e.g. network resources, devices characteristics, etc.) [7]. The above factors will force service providers to understand and analyze WSs QoS in order to achieve the fulfilment of quality guarantees. Different analytical quality assurance techniques are proposed. The goal of these techniques is to evaluate QoS and to uncover faults in the composed applications after they have been created. An example for analytical quality assurance techniques is simulation. The goal of the simulation is to emulate the conversational behavior of atomic WSs participating in the composition.

In this paper, we adopt a discrete-events simulation to test and verify WS compositions. We introduce a modeling approach for SBAs. This approach enables analytical description of SBAs and allows QoS predictions in different status and conditions of the execution context. We define a lightweight quality model for WSs focusing on essential properties of QoS that play a critical role for the effective management of WSs and that can be measured by simulation technique. We propose also a context model that supports explicit description of the execution environment of tested SBAs. This model is depicted into the simulation model in order to provide a context-based approach for evaluating SBAs.

978-0-7695-3399-5/08 $25.00 © 2008 IEEE
DOI 10.1109/ECOWS.2008.19

The rest of the paper is organized as follows. Section 2 reviews related work. In Section 3, simulation issues are addressed. We describe, in Section 4, the proposed context model. In Section 5, we introduce our quality model and explain metrics used to measure considered QoS properties. Section 6 describes a case study. Finally, Section 7 provides conclusions and outlines future works.

2. Related Work

To achieve the desired quality of an SBA, different analytical quality assurance techniques can be employed. The goal of these techniques is to evaluate QoS and to uncover faults in the SBA after it has been created. Among these techniques, we distinguish testing approach. The goal of testing is to (systematically) execute SBA with concrete inputs and to observe the produced outputs. Analyzing observed outputs allows determining and evaluating specific QoS properties.

In this work, we adopt simulation as a testing technique for analyzing the efficiency of SBAs. Simulation allows us to predict system performance in different status and load conditions of the execution environment. The predicted results are used to provide feedback on the efficiency of the system. Simulating Web processes for performance evaluation is a research area with little previous work. Works in simulation that are the closest to ours are described by [9] [3] [6].

[9] proposes a model-theoretic semantics as well as distributed operational semantics that can be used for simulation, validation, verification, automated composition and enactment of DAML-S-described SBAs. To provide a fully service description, Narayanan and McIlraith use the machinery of situation calculus and its execution behaviour described with Petri Nets. They use the simulation and modeling environment KarmaSIM to translate DAML-S markups to situation calculus and Petri Nets. The KarmaSIM tool allows for interactive simulation and supports various verification and analysis techniques to test functional correctness as well as quality performances. In this work, three verification problems are simulated and analyzed: reachability, liveness and existence of deadlocks.

Use of simulation for WSs is described also in [3]. This work focused on problems related to SBA specification, evaluation and execution using Service Composition and Execution Tool (SCET). SCET allows to compose statically a WS process with WSFL and to generate simulation model that can be processed by the JSIM simulation environment. JSIM is a Java-based simulation and animation environment that contains several features to support simulation of WS processes. After its simulation run, JSIM generates statistical information about the completed simulation. This statistical information provides feedback on the process performance. In this work, Chandrasekaran et al. have enhanced WSFL to include QoS estimates (e.g. execution time, cost and reliability) obtained by performing simulation tests. SCET manages also WS process execution; it is capable of automatically generate Perl executable code from WSFL based process description. In [11], SCET was integrated with METEOR-S system [1] in order to validate the QoS model developed by Cardoso et al. [2]. Only the QoS response time dimension is supported by the implementation of JSIM in both works [3] [11].

In contrast to our approach, both [9] [3] don't consider execution environment information in simulation model.

[6] presents a framework, named MAWeS (MetaPL/HeSSE Autonomic Web Services) whose aim is to support the development of self-optimizing predictive autonomic systems for WS architectures. It adopts a simulation-based methodology, which allows predicting system QoS in different status and load conditions. MAWeS integrated the simulation tool HeSSE. HeSSE allows the user to simulate the performance behavior of a wide range of distributed systems for a given application, under different computing and network load conditions. MetaPL is an XML-based metalanguage for parallel programs description. Starting from a MetaPL program description, a set of filters produce different program views, among which are the trace files that can be used to feed the HeSSE simulation engine. HeSSE and MetaPL allow users to perform SBA QoS optimization, without actual execution of the application on the target environment. In contrast to [9] [3], this work considers execution environment information in simulation model.

This work adds autonomic behavior to WSs, but it proposes only one possible optimization target that is response time minimization. Enhancement of MaWeS implementation is needed to add more optimization rules for QoS parameters. The contributions of this work on this topic are complementary to ours, insofar as they do not deal with QoS evaluation.

3. Discrete-events simulation modeling

Simulation is an important testing technique, as it allows users to tune and to evaluate different WS composition processes that provide identical functionality without experiencing the cost of enacting them. In this work, we propose an SBA modeling approach that is oriented towards performance evaluation through discrete-events simulation. To elaborate our simulation model for SBAs, we are based on the work presented in [10] that focus on modeling distributed applications.

Discrete-events simulation is frequently used to analyze and predict the performance of real word systems. In discrete-events simulation, a system is represented as a collection of entities that interact over time by exchanging events. Given the evolution of the operation of a system, we can analyze its behavior and evaluate appropriate quality measures.

We model an SBA as a combination of two types of entities: distributed application and network infrastructure entities. Our simulation model is shown in Figure 1.

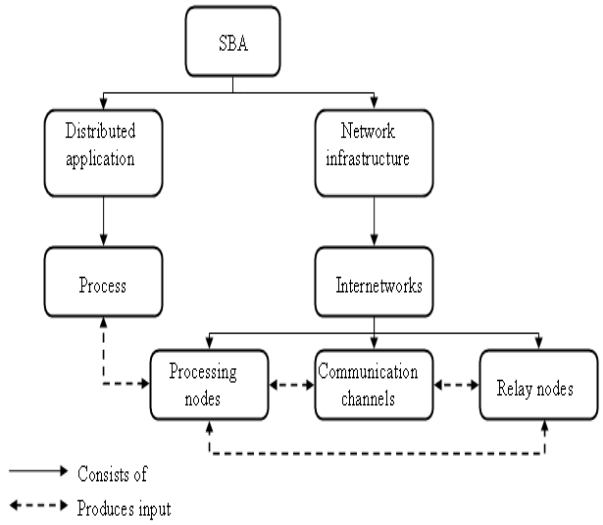

Figure 1. Simulation model for SBAs

The operation of distributed application is based on the client-server model. This model consists of two kinds of interacting processes: clients, which are invoked by users requesting service, and servers, which provide services and are invoked by other processes.
The operation scenario, retrieved from the BPEL document, is supported through specifying groups of actions:

- Processing: indicating data processing
- Request: indicating invocation of a server process
- Write: indicating data storage
- Read: indicating data retrieval
- Transfer: indicating data transfer between client and server processes
- Synchronize: indicating synchronization

Each process is executed on a processing node. Processing action indicates invocation of the processing unit of the corresponding node and is characterized by the amount of data to be processed. According to the SOA, communication between processes is performed through exchanging SOAP messages. Request action indicates invocation of a server process and is characterized by the name of the server, the name of the WS, its invoked interface and the required inputs. Forking processes are modeled as a parallel execution of multiple requests. There are two actions available for data storing, read and write, which are characterized by the amount of data stored and retrieved, respectively, and the server invoked. Transfer action is used to indicate SOAP messages exchange between processes. Synchronize action is needed since replication of processes and data is a common practice in such distributed applications. Synchronize action parameters include the process replicas that must be synchronized and the amount of data transferred.
In the proposed modeling scheme, the network infrastructure is considered as a collection of internetworks, exchanging messages through relay nodes (active communication devices e.g. routers). Communication element entity represents protocol suites (i.e. routing protocols (OSI layers 2 and 3) and peer-to-peer protocols (OSI layers 4-7)).
We have conducted simulation experiments using NS-2 simulator [4]. NS-2 is a discrete-events simulator; its code is written in C++ with an OTcl interpreter as a front end. NS-2 is targeted at networking research. It provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless networks. The main advantages of such an object-oriented simulator are reusability and easy maintenance. To support SBA simulation, we have extended the C++ class hierarchy in order to implement HTTP, SMTP and SOAP protocols.

4. Context model

By the term ”context”, we mean ”information utilised by the web service to adjust execution and output to provide the client with a customised and personalised behaviour” [5].
Since SBAs are operating in dynamic environments, variations of execution context lead to variations in QoS expectations. In this work, we propose a context-based approach for evaluating SBAs performances. We consider a context model, described in Figure 2, which consists of a set of elements grouped in 2 axes.

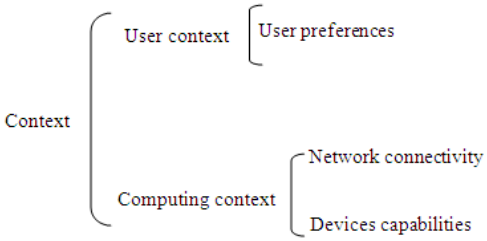

Figure 2. Context model

- User context: describes user preferences. To each QoS property, user attributes a weight. He chooses the value of this weight according to the level of the QoS property he needs(e.g. execution time←0.8).

- Computing context: describes network connectivity (e.g. Internet connexion, locality, bandwidth, etc.) and devices capabilities (e.g. memory capacity, CPU speed, etc.).

Context information is described in the simulation model. Context changes are modeled as discrete events.

5. QoS model

In this approach, we define a light-weight quality model focusing on essential properties of QoS that play critical role for the effective management of WSs and that can be measured by simulation technique. The QoS properties detailed above are defined in the context of elementary WSs. They are also used to evaluate the QoS of composite WSs. We use the QoS computation model described by Cardoso et al. [2] to provide aggregation functions for computing the QoS of composite WSs.

- Response Time: it corresponds to the total time needed by a WS to transform a set of inputs into outputs. Response Time (RT) for a service (s) can be computed as follows:

$$RT(s) = ST(s) + DT(s)$$

  - Service Time (ST) is the time that the WS takes to perform its task.
  - Delay Time (DT) is the time taken to send/receive SOAP messages.

- Reliability: it corresponds to the likelihood that the service will perform for its users on demand. Reliability (R) of a service (s) is function of the Failure Rate (FR):

$$R(s) = (1\text{-}FR(s))*100$$

  - FR= successful executions/scheduled executions

- Availability: it refers to the rate of Service Activity (SA). Availability (A), during a time Interval (I), for a service (s) corresponds to:

$$A(s) = SA(s)/I$$

- Scalability: it computes service's capacity to manage loads. To test the scalability of a WS, we conducted the simulation while changing the number of concurrent clients.

6. Case Study

In this section, we describe two WS processes which provide the same required functionality. We use our simulation approach to evaluate each of these processes. We propose a validation method that help user to choose the appropriate process that fit the best to his preferences and his execution environment. Both WS Process 1 and 2 (WSP1, WSP2), described in Figure 3 and Figure 4, ensure on-line books acquisition.

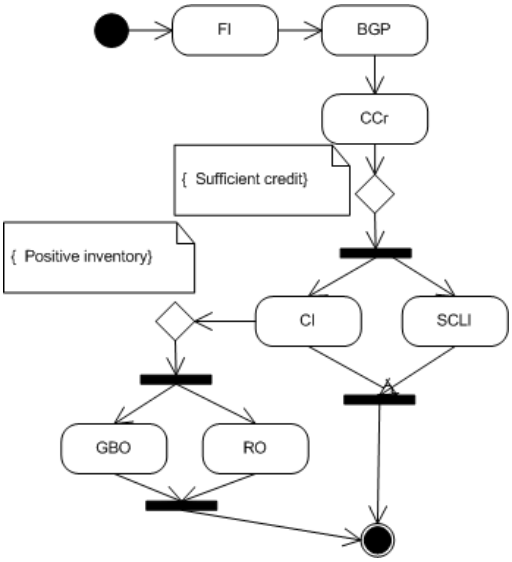

Figure 3. Activity diagram of WSP1

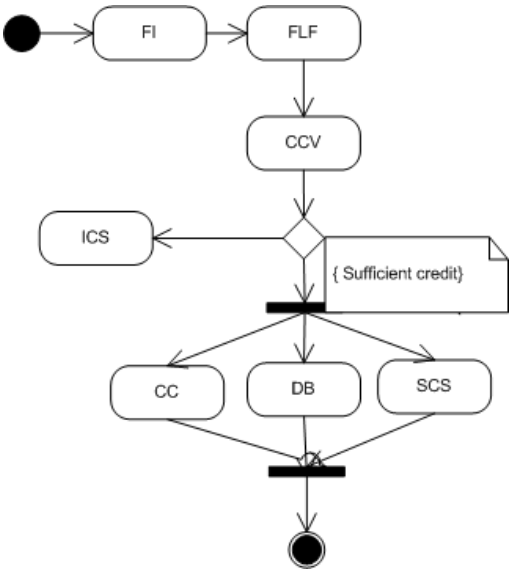

Figure 4. Activity diagram of WSP2

WSP1 begins by invoking FindISBN (FI) service. This service allows the customer to find the ISBN code of the wanted book. The price of this book is retrieved using the BarnesGetPrice (BGP) service. The user's account is then checked for sufficient funds using the CheckCredit (CCr) service. If the user has sufficient credit, the CheckInventory (CI) service is invoked; otherwise, the SendCreditLowInfo (SCLI) service is invoked. If the CheckInventory Web service returns true the ReleaseOrder (RO) service is invoked in order to send the book; otherwise, the GenerateBackOrder (GBO) service is invoked.
In WSP2, the customer begins also by searching ISBN code of the book that he wants to buy. The FindLowestFare (FLF) service allows him to find the cheapest bookstore. The user's credit card is then checked by CreditCardValidator (CCV) service. If the user has sufficient credit, ChargeCard (CC), DispatchBook (DB) and SendConfirmationSMS (SCS) are invoked. Otherwise, InsufficientCreditSMS (ICS) is executed in order to inform customer of his insufficient credit.
Figure 5 shows the execution context of the user.

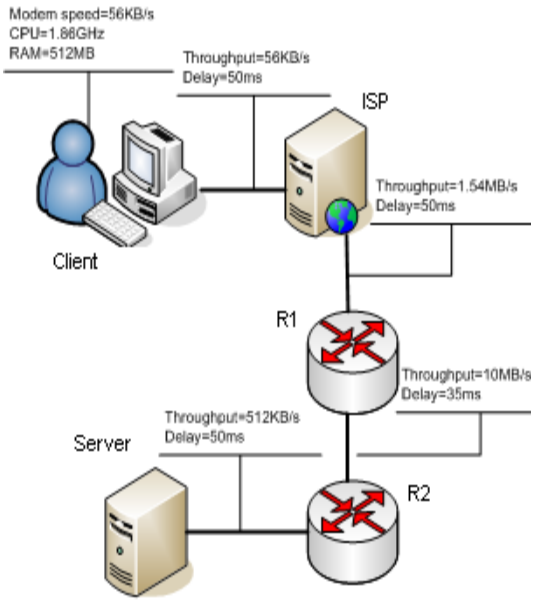

Figure 5. Context scenario

- Devices: PC, CPU: 1.86GHz, RAM: 512MB.
- Internet connexion: NUMERIS, Modem speed: 56KB/s.
- Network: topology: Client is connected to an Internet Server Provider (ISP), which is in its turn connected to Server through two Routers (R1) and (R2).
  - Link Client_ISP: throughput: 56KB/s, delay: 50ms.
  - Link ISPs_R1: throughput: 1.54MB/s, delay: 25ms.
  - Link R1_R2: throughput: 10MB/s, delay: 35ms.
  - Link R2_Server: throughput: 512KB/s, delay: 50ms.

Table 1 illustrates user's preferences. For each QoS property i, user attribute a weight w_i ($0 \leq w_i \leq 1$ and $\sum_{n}^{i=1} w_i = 1$).

Table 1. User's QoS preferences

QoS property	Weight
Response Time	0.35
Reliabiliy	0.3
Availability	0.2
Scalability	0.15

The configuration of the simulation platform is four Pentium IV-based Windows 2000 systems. For each QoS property, we performed a set of simulation experiments, and we have considered the average value. Simulation results in Figure 6 show that WSP1 is more reliable, available, and scalable than WSP2, but WSP2 is faster than WSP1.

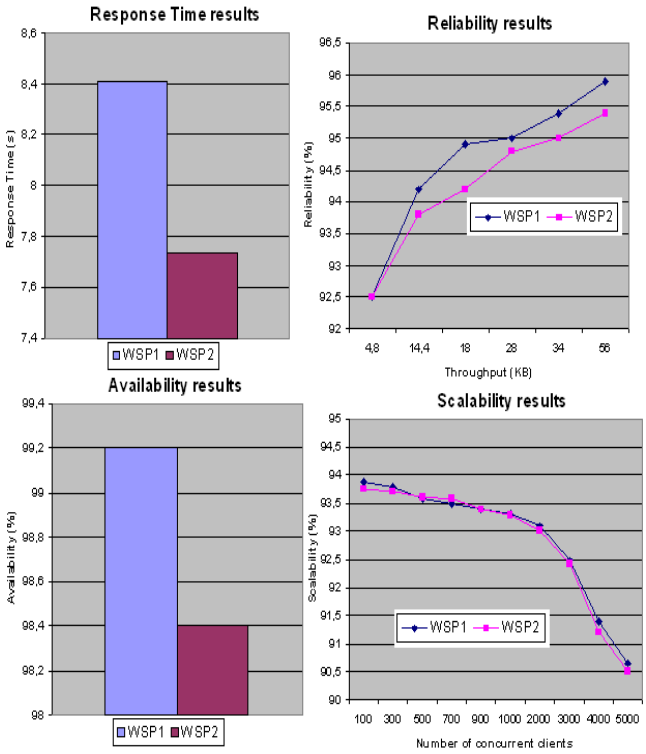

Figure 6. Simulation results

To validate these results in order to help the user to choose the appropriate WS process, we use a Benefit

Function (BF). BF is computed as follows:

$$\text{BF} = \sum_{n}^{i=1} d'_i * w_i \text{ with } \sum_{n}^{i=1} w_i = 1$$

Where d'_i is a normalized value of a QoS dimension d_i and w_i denotes the client's assigned relative importance to the dimension. As dimensions can be of different units (e.g., response time is in second and availability in percentage), in order to allow for a uniform measurement of WS QoS independent of units, data normalization is applied, which essentially transforms values of different units into comparable ones. By considering a 75% confidence interval, the dimensions that are stronger with larger values (e.g., reliability, availability and scalability) are normalized according to the following equation:

$$f(n) = \begin{cases} d'_i = 1 & \text{if } d_i \text{ - m(d)} > 2 * \delta\text{(d)} \\ d'_i = 0 & \text{if } d_i \text{ - m(d)} < 2 * \delta\text{(d)} \\ d'_i = \frac{d_i - m(d)}{4 * \delta(d)} + 0.5 & \text{otherwise} \end{cases}$$

While for QoS dimensions that are stronger with smaller values (e.g., response time), they are normalized according to the following equation so that smaller values contribute more to the user benefit:

$$f(n) = \begin{cases} d'_i = 0 & \text{if } d_i \text{ - m(d)} > 2 * \delta\text{(d)} \\ d'_i = 1 & \text{if } d_i \text{ - m(d)} < 2 * \delta\text{(d)} \\ d'_i = 0.5 - \frac{d_i - m(d)}{4 * \delta(d)} & \text{otherwise} \end{cases}$$

where d_i is the value of dimension d for the service instance *i*, and m(d) and δ(d) are the mean and standard deviation values for dimension d respectively.
Validation of WSP1 and WSP2 has given the results shown in Table 2.

Table 2. Validation results

	WSP1	**WSP2**
BF	0.33	0.228

This validation proves that WSP1 is more appropriate for user's expectations than the WSP2 is.

7. Conclusion

In this work, we adopt a discrete-events simulation approach to evaluate QoS of SBAs. We present a simulation modeling approach. This approach enables analytical description of SBAs and allows QoS predictions in different status and conditions of the execution context. We define a light-weight quality model considering a set of QoS properties that can be measured by simulation technique. We propose also a context model that describes execution environment and user's preferences. This model is depicted into the simulation model in order to provide a context-based approach for evaluating SBAs.
To show the effectiveness of our approach, we have conducted a set of simulation experiments in order to evaluate and to validate two WS processes that provide the same required functionality.
Future work is devoted to extend both our context and QoS model and to perform extensive experiments to evaluate more complex composite WSs.

References

[1] R. Aggarwal, K. Verma, J. Miller, and W. Milnor. Constraint driven web service composition in meteor-s. *SCC '04: Proceedings of the 2004 IEEE International Conference on Services Computing*, pages 23–30, 2004.

[2] J. Cardoso. Quality of service and semantic composition of workflows. *PhD thesis*, 2002.

[3] S. Chandrasekaran, J. A. Miller, G. A. Silver, I. B. Arpinar, and A. P. Sheth. Performance analysis and simulation of composite web services. *Electronic Markets*, 13(2), 2003.

[4] K. Fall and K. Varadhan. The *ns* manual. *http://www.isi.edu/nsnam/ns/doc/ns-doc.pdf*.

[5] M. Keidl and A. Kemper. Towards context-aware adaptable web services. *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 55–65, 2004.

[6] E. Mancini, U. Villano, M. Rak, and R. Torella. A simulation-based framework for autonomic web services. *ICPADS '05: Proceedings of the 11th International Conference on Parallel and Distributed Systems - Workshops (ICPADS'05)*, pages 433–437, 2005.

[7] A. Mani and A. Nagarajan. Understanding quality of service for web services. *http://www.ibm.com/developerworks/library/ws-quality.html*, 2002.

[8] D. A. Menascé. Qos issues in web services. *IEEE Internet Computing*, 6(6):72–75, 2002.

[9] S. Narayanan and S. A. McIlraith. Simulation, verification and automated composition of web services. *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 77–88, 2002.

[10] M. Nikolaidou and D. Angnostopoulos. An application-oriented approach for distributed system modeling and simulation. *ICDCS '01: Proceedings of the The 21st International Conference on Distributed Computing Systems*, page 165, 2001.

[11] G. Silver, A. Maduko, R. Jafri, J. A. Miller, and A. P. Sheth. Modeling and simulation of quality of service for composite web services. *Proceedings of the 7-th World Multiconference on Systemics, Cybernetics and Informatics (SCI'03)*, 1:420–425, 2003.

[12] L. Zeng, B. Benatallah, A. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. Qos-aware middleware for web services composition. *IEEE Trans. Softw. Eng.*, 30(5):311–327, 2004.