MODEL BASED FRAMEWORK FOR MEASURING SERVICE LEVEL AGREEMENT PERFORMANCE IN SERVICE ORIENTED ARCHITECTURE

ALAWI ABDULLAH AHMAD AL-SAKKAF

UNIVERSITI TEKNOLOGI MALAYSIA

MODEL BASED FRAMEWORK FOR MEASURING SERVICE LEVEL AGREEMENT PERFORMANCE IN SERVICE ORIENTED ARCHITECTURE

ALAWI ABDULLAH AHMAD AL-SAKKAF

A thesis submitted in fulfilment of the
requirements for the award of the degree of
Doctor of Philosophy (Computer Science)

Faculty of Computing
Universiti Teknologi Malaysia

AUGUST 2013

For my mother and father

To my supervisors and sponsor

ACKNOWLEDGEMENT

In preparing this thesis, I was in contact with many people, researchers, academicians. They have contributed towards my understanding and thoughts. In particular, I wish to express my sincere appreciation to my main thesis supervisor, Dr. Dayang Norhayati Binti Abang Jawawi, for encouragement, guidance, critics and friendship. I am also very thankful to my co-supervisor Professor Dr. Robert Colomb for their guidance, advices and motivation. Without their continued support and interest, this thesis would not have been the same as presented here.

Do not forget all my life, my beloved friend Abdolgaffar Hamed Ahmed, that the impact of the significant impact in my life. He provided me with deep to understanding of specialization. as well as the impact in the understanding of the way to the knowledge of the Creator. Where I am touch from his characteristics high-level and sophistication.

I am also indebted to Universiti Teknologi Malaysia (UTM) for funding my Ph.D. study. librarians at UTM and for all services. My fellow postgraduate students should also be recognised for their support. My sincere appreciation also extends to all my colleagues and others who have provided assistance at various occasions. Their views and tips are useful indeed. Unfortunately, it is not possible to list all of them in this limited space. I am grateful to all my family members.

ABSTRACT

Service-Oriented Architecture (SOA) which manages remote service under a third party or provider is a new paradigm for building IT systems. In SOA, the increasing demand for cross-organizational services has highlighted the need for Service-level Agreement (SLA) and monitoring of its service level (performance). Although the role of machine-readable SLA languages like Web Service Level Agreement (WSLA) is recognized, but, the engineering of monitors is complex because it uses the code-based approach. Therefore, research on effective designs of monitors for SOA environment and providing standards in the instrumentation process would improve SOA. This thesis proposed a model-based engineering approach to raise the abstraction and re-use levels for designing standard monitors with automation support. Model Driven Architecture (MDA) was used to automate the development of the software product (monitor). This was done by mapping a business model called Platform Independent Model (PIM) into Platform Specific Model (PSM) using Query View Transform (QVT) as the standard language. In this approach the PIM metamodel is stemmed from WSLA while the PSM is borrowed from SEI framework. Model-based testing was used to generate tests as an artifact which is a requirement for the 6-element framework. As a design science research, an email system case study was used to evaluate the framework. The results showed that Model-based engineering provided a standard method for developing monitors that has raised the abstraction and eventually led to a maintainable and reusable framework. PSM would also act as the standard implementation model for configuring monitors using QVT because it is effective and could configure a number of monitors by reusing the same artifacts (proposed PIM and PSM) requiring less human intervention. Besides that, the PIM metamodel can be extended to accept different SLA languages. The research has proven that the proposed models are not only the best means of communication between SLA stakeholders, but are the core engineering assets for both human and machine because they could reduce engineering effort.

ABSTRAK

Reka bentuk Berorientasikan Perkhidmatan (SOA) yang menguruskan kawalan perkhidmatan di bawah pihak ketiga atau pembekal adalah satu paradigma baharu untuk pembangunan sistem IT. Dalam SOA permintaan yang semakin meningkat kepada perkhidmatan merentas organisasi telah meningkatkan keperluan untuk Perjanjian Tahap Perkhidmatan (SLA) dan pemantauan tahap perkhidmatan (prestasi). Walaupun peranan bahasa SLA boleh dibaca oleh mesin seperti Perjanjian Tahap Perkhidmatan Laman Sesawang (WSLA) diiktiraf tetapi kejuruteraan monitor adalah kompleks kerana pendekatannya berasaskan kod. Justeru itu penyelidikan mengenai reka bentuk yang efektif untuk memantau persekitaran SOA dan menyediakan standard dalam proses instrumentasi akan meningkatkan SOA. Tesis ini mencadangkan pendekatan kejuruteraan berasaskan model dan tahap penggunaan semula untuk mereka bentuk monitor dengan sokongan automasi. Senibina Berpandukan Model (MDA) digunakan untuk mengautomasikan pembangunan produk perisian (monitor). Ini dilakukan dengan memetakan model perniagaan yang dikenali sebagai Model Paltform Bebas (PIM) dalam Model Platform Khusus (PSM) menggunakan Permintaan Paparan Berubah (QVT) sebagai bahasa standard. Dalam pendekatan ini metamodel PIM berasal daripada WSLA manakala PSM dipinjam daripada kerangka kerja SEI. Pengujian berasaskan model telah digunakan untuk menjana ujian sebagai artifak yang menjadi keperluan kepada kerangka kerja enam elemen. Sebagai reka bentuk penyelidikan sains kajian kes melalui e-mel telah digunakan untuk menilai kerangka kerja tersebut. Hasil kajian menunjukkan bahawa kejuruteraan berasaskan model menyediakan satu kaedah standrad dalam pembangunan monitor yang meningkatkan pengabstrakan dan menghasilkan kerangka kerja yang mudah diselenggara dan digunakan semula. PSM juga akan bertindak sebagai model pelaksanaan standard untuk mengkonfigurasi monitor menggunakan QVT kerana PSM berkesan dan boleh menetapkan beberapa monitor menggunakan semula artifak yang sama (yang dicadangkan PIM dan PSM) dengan sedikit campur tangan manusia. Selain itu metamodel PIM boleh dilanjutkan untuk menerima bahasa SLA yang berbeza. Kajian telah membuktikan bahawa model yang dicadangkan bukan sahaja cara terbaik komunikasi antara pemegang saham SLA tetapi merupakan aset kejuruteraan teras kepada manusia dan mesin disebabkan boleh mengurangkan usaha kejuruteraan.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LEST OF ABBREVIATIONS

MBT	-	Model Base Testing
FPA	-	Function Point Analysis
SLA	-	Serves Level Agreement
SOA	-	Serves Oriented Architecture
WSDL	-	Web Services Description Language
WSPOLICE	-	Web Services Police
UML	-	Unified Modeling Language
SUT	-	System Under Testing
IT	-	Information Technology
QOS	-	Quality of Service
CBSE	-	Component – Based Software Engineering
EML	-	Extensible Markup Language
OMG	-	Object Management Group
PIM	-	Platform Independent Model
PSM	-	Platform Specific Model
SQL	-	Structured Query Language
WSLA	-	Web Service Level Agreement
QVT	-	Query View Transform

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Overview

In internet computing, both internal and external customers presume that business services have to always be accessible and offer disruption-free performance. Any problem that interrupts the performance of internet-speed business will lead to the loss of business in the real world. Therefore, it is extremely important to detect, diagnose and correct any performance problems before the service is deployed.

The current business requirements depict a complex map for IT infrastructure because it is generally very large and quickly expanding. Hence, it is important for modern enterprises to have a mechanism to monitor the quality of service provided by their IT infrastructure. This mechanism is generally called a dashboard, which allows users to monitor, detect and correct the infrastructure (Swebok, 2004) in order to increase the value of their business processes. Technologies and mechanisms to monitor the performance constraints are necessary in order to achieve business goals.

The function of these automatic monitoring mechanisms became necessary by maintaining it within the size of complex systems. An effective practical engineering approach is needed to improve the quality of a monitor's design. Model-based engineering is a new promising trend such as model-driven architecture

(MDA) and model-based testing (MBT). MDA is OMG initiative, which separates the conceptual Platform Independent Model (PIM) from Platform Specific Model (PSM). It uses mapping as an ultimate goal of automating product engineering. The capability of raising abstraction level encourages the reusing of artefacts such as PIM, PSM and even the mapping rules at a certain domain. Software testing and test cases are common practices in engineering task (Mark, 2007) (Bill, 2008). One of the common software testing methods under the model-based initiative is called Model-Based Testing (MBT). MBT is a form of black-box testing that uses behavioral and structural models such as UML diagrams to generate test cases automatically. MBT is strongly suitable for Service-Oriented Architecture (SOA) environment because the test cases generators are able to cover almost all model-related features such as states in UML state machine and boundary values for the data coverage (Wieczorek, et al, 2008). Therefore, the philosophy of MBT follows the same trend that automates test cases generation (not as before like script-based testing) (Mark, 2007) and execution which starts testing from specification of the system under test. MBT toolset makes it easy to verify system functionality. Due to performance being an execution property, test cases execution is used as a means of telling about system behavior at run time.

The current trend in the modeling and designing of service-oriented systems follows a new paradigm called SOA (Thomas, 2007) (Nicolai, 2007). In this approach, the functionality of the system is assigned to loosely coupled services where integration between heterogeneous systems is possible, thereby reusing increased agility to adapt to changing business requirements. Service Level Agreement (SLA) is an obligation between the service provider and service consumer in which services and the level of quality are specified. SLAs have been used in IT organizations and departments for many years. The definition of SLAs in SOA framework is still new; however, it has become extremely important in recent times due to the high demand on services in SOA systems that cross over the organizational boundaries and a lot of third-party service providers (Philip, et al., 2008). Therefore, it is required to measure and ensure quality of service from both the service provider and service consumer prospective.

Generally, SOA software systems have a different life cycle than traditional software which consists of analysis, design, implementation and testing. The new dimension of SLA as a new artifact in this paradigm requires several engineering sub-tasks such as specification using languages (i.e. WSLA) (Diana and Boyan, 2009), measuring and evaluation. From an engineering point of view, assessing the performance of service-oriented system and checking its compliance with the specified parameters in SLA is a very significant task in SOA. This is because performance is the critical factor that affects the global Quality of Service (QoS) that is expected by the end-users of the systems. Therefore, by using model-based approach, the complexity of the large engineering activities involved will be minimized.

1.2 Background of the Problem

SLA is an essential artifact in realistic SOA systems, especially for the service providers in a large-scale of software such as (Amazon). Building SLA monitors is the interest of both perspectives: the end user or service consumer, and the provider. Since both need to specify a certain level of quality, such as specifying the metrics and parameters of performance of the service-based system, an SLA language will be described first. The background of SLA management, evaluation and life cycle will be presented next, and finally, the general approaches to monitoring SLA will be defined.

1.2.1 SLA languages

In the history of IT computing, English or some other natural language has been used to describe SLA elements of agreement for service levels. An example of this is one of the SLA documents namely the Amazon S3 Service Level Agreement (Amazon, 2008). This document includes a section which declares the company's commitment in providing the Simple Storage Service (S3) that is available with a monthly uptime hour of at least 99.9%. The SLA includes a service credit where users who experience unavailability of service could demand for monetary credit as compensation.

The new trend towards SLA is for it to be in standard form and machine-readable by formalizing it. This direction is new and a few standards exist with these properties in the literature review (Ed et al., 2010), for example, IBM's WSLA framework and the WS-Agreement specification. These XML-based languages can be used to create machine-readable SLAs for services implemented using web services technology which define service interfaces using WSDL (W3C,2001). WSLA is an extensible language that can be extended to adopt other technical or service-based technologies. A machine-readable SLA is better than text for reasons identified in the literature review and is discussed in Chapter 2 under Section 2.9. Therefore, if SLA is machine-readable, the measurement could be automated easily.

1.2.2 SLAs management and evaluation

As mentioned before, the development of SLA is the main difference between the traditional system and SOA systems. It is an engineering task that consists of a number of sub-activities such as instrumentation and measurement which assign values to SLA parameters (Keller, et al., 2003). The important parts for current studies are the evaluation of the subsystem; it takes input from SLA metrics, parameters and checks the values against the guaranteed conditions of SLA. In case of violation, certain actions should be triggered and this process is expected to be relying on tools so it can be performed automatically. This process is also called monitoring. Generally, there is no standard way in executing this step. There are different practices most of which are low level tasks that involve substantial effort and tedious work.

1.2.3 SLA life cycle

The following are the phases in an SLA life cycle as defined in (TMForum, 2008). It involves four phases: a) Service and SLA template development, b) Negotiation, c) Preparation and d) Execution. The focus of current studies is on Execution, which mainly involves the assessment or evaluation of SLA and QoS that are provided to an individual or group of customers.

As stated by SEI (Philip, et al., 2008), this is an active research area because in extreme cases there is a need to automate the SLA life cycle in order to enable the dynamic provisioning of service between organizations. Here, all of the steps are done at runtime. Moreover, the assessment of QoS is seizing the attention of researchers and large scale organizations (Philip, et al., 2008) (OMG, 2009).

1.2.4 The need for measurement and assessment of SLA

In traditional software engineering practices, software testing is used as a common tool to verify the functionality of systems. With the advent of SOA, verifying the QoS aspects of SLA becomes an issue as there are many practices of cross-boundaries services emerging. Two examples of this are an online storage web service offered by Amazon Web Services, and an exchange server provider hosting customers' emails (i.e. Microsoft live outlook). In both examples, performance is a critical QoS need, which must be verified by the end users or third parties at provisioning time. This is due to many factors; if we look carefully at current service-based systems, services are able to communicate because they are independent of technology. Apart from that, service is allowed to grow dynamically. In this case, a service provider could enhance the quality of functionality provided by their systems such as increasing the resources available to the service. This causes a variation in the service's non-functional properties. For instance, optimization could improve a non-functional (i.e. performance) property while worsening another (i.e.

availability). Therefore, this may lead to the violation of the SLA obligations as was the case with Amazon S3 where its availability became lower than 95%.

Therefore, it is clear that there is a high demand for automated monitors that help taking a decision by large-scale service providers like Amazon and service consumers like UTM in outsourced email service.

SLA is usually expressed in statistical terms over large numbers of service instances (average latency, small percentage above a certain threshold; need to specify the demand pattern, and so on). The instrumentation (monitor) measures performance of service instances in order to be compared with the expected results. This step requires considerable processing before they can be related to the SLA terms. It is made complicated by the need for transparency. Both the service provider and consumer need less design or configuration effort given the fact that this process has a high frequency of execution (consider Amazon customers). In addition, it also needs business models instead of low level languages to enable easier communication among stakeholders and to increase the effectiveness of the monitoring process (re-using among different clients). For example, most of the current SLA machine-readable languages and parameters are XML-based language. This situation increases the complexity of the monitoring process and making it more tedious.

Having that knowledge, there exist now a need to reduce engineering efforts by increasing the automation of this process and putting the artifacts in high quality. Model-based approaches like model driven architecture and MBT (see Chapter 3 under Section 3.8) are common trends which have added many values to the engineering of systems such as reducing human intervention for the purpose of development in the case of MDA and generating tests automatically in the case of MBT.

Moreover, the monitor will be working in SOA environment which gives rise to new dimension (will be elaborated in detail in Chapter 3). For example, the testing of challenges in SOA environment can be categorized based on roles or perspective (developer, provider, end user), kinds (functional, non-functional), time (design time, provisioning time), scope (SOA infrastructure, web service, end-to-end thread), and testing level (individual service or business process).

However, studying the problem with models could be made easier due to a high level of abstraction and the ability of being a re-usable artifact between service provider and requestors (Colomb, et al., 2006).

1.3 Statement of the Problem

The fundamental question to address the research problem is:

How SLA monitor design of performance can be automated for SOA-based systems?

This study is about designing a monitoring system for performance parameters gathered from the end user at provisioning time. So far contribution was proprietary solutions with more engineering efforts. In addition, although there is a standard for SLA like the standard language developed by IBM - WSLA (Ludwig, 2005), it does not show any details on how the monitor design will turn out to be. However, this step from IBM and others is a progress towards this kind of problem because there is no standard so far for SLA. In addition, the focus of literature was on assessment at design time and SOA infrastructure (Domenico, et al, 2009) (Alin, 2009). Furthermore, most of the monitor designs are not environments for service-based (Thomas, 2007). As shown in the introduction, SOA is a paradigm shift for most IT-enabled information systems.

Thus, this question would be better addressed by presenting an effective framework to help SOA engineers evaluate the performance through monitor design at SOA-based systems.

The main issue in this study could be divided into several research questions as the following:

A. What is the trend in SLA and more importantly specifying performance parameters for SOA-based systems?

The huge definition of internet service-based systems perspectives, environment and engineering stages leads to the need to firstly identify the domain of the problem. It is important to sort out the differences between the views of the client, the provider and/or the service integrator in SOA. This will help the researcher to choose the type of operational performance's metrics. For example, the throughput can be measured in terms of request per second or in data rate per second as average data rate includes latency as well (Ed, et al., 2010). In addition, performance can be measured for infrastructure components (BPEL engines, parser, etc.) as well as from the developer perspective at design time (Domenico, 2009).

The history of SLA was based on telecom practices so most of the terms are low level as well as metrics. The SLA is specified manually in the form of templates. The instrumentation process itself involves a lot of human effort which consists of component configuration and deployment plan. Recently, the need for high level machine readable language became more apparent, but it still needs to be studied under the SOA context.

B. What is the appropriate framework for measurement in SOA context?

There are a number of frameworks working in this dimension, but they are basically lacking in longevity and a standard method of developing monitors that help achieve re-usability at different scales such as the reusing of SLA aritfacts which is not addressed. Most of the contributions are proprietary solutions as will be seen in the litature review. However, the monitors are rewritten many times for a similar class of problems due to low level abstraction used in the designs. This situation becomes worse with increases in service-based systems on the internet which would require a greater amount of time and effort from the developer's point of view (Amazon with ten thousand and more customers utilizing Amzone's different web services). In addition, deploying the monitoring infrastructure (Simon, 2011) requires effort and time.

A more efficient approach is the one that reduces human effort by increasing the automating of different activities as well as keeping communication between stakeholders easier. Here, experiences and practices from state-of-the-art model-based engineering will be considered because models are like a coin which has two faces: one for human, and the other for machine. The tools behind this framework are needed for the measurement of SLA performance parameters which will be addressed from three dimensions:

a) SLA language representation and manipulation.
b) Transformation mechanisms.
c) Generation of measurements result.

The goal of the first dimension is to comprise a formal and expressive language that has the ability to encode the quality attribute specifications (QoS) which also gives the ability to describe different levels of quality provided by a single service. Next, the goal of the second dimension is to be able to relate the terms of SLA with the capabilities of available instrumentation (monitoring). Finally, the goal of the last dimension is to prepare execution environment in order to implement and execute complex functions to calculate the SLA performance parameters terms by considering the workload.

C. What are the components to be considered in the monitoring of SLA performance parameters?

The components of frameworks vary in their contributions, but have some degree of similarities. The central point is the concept of load which is used in a non-realistic way in many cases (java methods for simulating load) (Antonia, 2008). The concept of test cases or tests is used as a body from the system showing time behaviour. Testing is used but in current cases there are various reasons that make testing SOA applications a challenging task. In most cases, services are often outside the control of the organization that is using the service (service developers are independent from service providers and service consumers). As a result, potential mismatches and misunderstandings between parties can occur. Additionally, SOA applications are highly dynamic with frequently changing services and service consumers, changeable load on services, SOA infrastructure, and basic network. Consequently, it is normally impossible to capture all possible configurations and loads during the testing process. Therefore, what could possibly be done as from many common practices and literature is to identify and maintain a set of configurations that are considered important and use it as the base to define the test environments (Philip, et al., 2008).

It turns out that the measurement or testing environment will have an influence on the result. Thus, it must be as similar as possible to the deployment environment. This means that the simulation environment should include complete and realistic elements such as workload. This will allow measurement metrics such as latency and response time to be more realistic because it is performed under real working load.

1.4 Objective of the Study

The main goal is to propose a framework for monitoring performance parameters in the context of SLA in SOA systems. In order to achieve this, the following objectives are required:

i. To identify the SLA performance parameters and design SLA elements of performance monitor in the context of SOA.

ii. To propose a model-based monitor framework for monitoring SLA performance parameters under the context of SOA.

iii. To evaluate the framework of SLA performance parameters using a case study.

1.5 Scope

- The scope of this research is mainly covers the monitor of performance parameters in SLA but not the performance engineering and development of metrics.
- The monitor engineering is the main issue in this research where an effective framework is need.
- Model-based approach will be followed in the design of monitor or architecture.
- The SOA environment and service-based systems is the main environment where the problem is studied.

1.6 Motivation

Service-oriented systems have recently grown and started to cross over the organization boundaries. One of the huge examples is Amazon web services - services[1] for businesses, consumers, Amazon associates, sellers, and other developers.

Service-based systems currently are engineered by SOA principles in which software functionality is outsourced by one or more providers; hence, the program is not entirely under the control of clients as traditionally experienced. For example, the UTM email system is outsourced by Microsoft Outlook Live, the third party which is the exchange server hosting company.

Performance of services is among the most important concerns in an organization because of the dynamic nature of service-based systems. Service providers usually enhance services for many reasons such as optimization and improvements of services. Therefore, there are demands on automated systems and frameworks to manage the whole SLA life cycle. Performance assessment of a service component, at a minimum consists of the execution of a series of tests, each one with a specific workload for the component and the collecting and aggregating of some performance metrics that characterize the system. In the distributed systems and service-oriented systems, the components can be deployed on different machines over different networks and it may need to be stimulated by different remote clients. However, this task when performed manually or with a limited amount of automation can be problematic and error-prone (Domenico, et al, 2009).

1.7 Theses Organization

In Chapter 1 the context of the research has been setup which involves problem description and objectives. It supports with terminologies and background of the problem in SLA space in general. Chapter 2 shows us the dimension and new directions of the literature where the problem is stemmed. The pivot of this Chapter is the theory and trend from where framework concept is stemmed like MDA and MBT are investigated. The explanation of SLA, standard language like WSLA and monitors in SOA environment is covered to show the current limitations of current designs of performance monitors. Chapter 3 describes the research methodology and this sort of a research where its main feature is discussed through design science filed of research. The innovation and proposed solution to the problem of this research is explained in details at Chapter 4. It is a new framework for designing SLA Monitor of performance using model-based approach. Chapter 5 is about the evaluation of the proposed framework using case study. It explains an email case study where a concrete SLA is used between service requestor a provider to show how monitor concepts and strategies discussed at Chapter 4 are implemented. In addition in this chapter each step is linked with Chapter 4 through principles named A-D based on Chapter 4 philosophy. Chapter 6 draws a conclusion where results generated by the research are restated and discussed within the scope and research question addressed at Chapter 1.

REFERENCES

AIS, Association for Information System. (2010). http://desrist.org/design-research-in- information-systems. Date access 24-11-2009.

Ajaya, K., Manas, R. (2011). Modeling and Monitoring SLA for Service Based Systems, ISWSA, Amman, Jordan. Copyright, pp. 924-931.

Alin, S., Sebastian, W., Andrei, K. (2009). A Model-Based Testing Approach for Service Choreographies, ECMDA-FA '09 Proceedings of the 5th European Conference on Model Driven Architecture - Foundations and Applications. pp. 313 - 324.

Amazon Web Services. (2008). Amazon S3 Service Level Agreement, www. aws.amazon.com/s3-sla.

André, V., Jan, W., and Wilhelm, H., (2012). Kieker: A Framework for Application Performance Monitoring and Dynamic Software Analysis, ICPE'12, April 22–25, 2012, Boston, Massachusetts, USA. pp. 247-248.

Andrew, D., Marek, J., Mathias, S., Claudio, B. (2004). Performance Monitoring of Service-Level Agreements for Utility Computing Using the Event Calculus, Electronic Contracting, 2004. Proceedings. First IEEE International Workshop on . 2004 , pp 17-24.

Anneliese, A., Robert, F., Sudipto, G., and Gerald, C., (2003). Test adequacy criteria for UML design models. Journal of Software Testing, Verification and Reliability, pp. 95-127.

Antonia, B., Guglielmo De Angelis, Lars, F, Andrea, P. (2008). Model-Based Generation of Testbeds for Web Services, IFIP International Federation for Information Processing 2008. K. Suzuki et al. (Eds.): TestCom/FATES 2008, LNCS 5047, Springer, pp. 26-282.

Antoniol, G., Briand, L. C., Di Penta, M., Labiche, Y., (2002). A case study using the round-trip strategy for state-based class testing, IEEE Computer Society. In 13th International Symposium on Software Reliability Engineering, pp. 26-279.

Arilo, C., Rajesh, S., Marlon, V., Guilherme, H., (2007). A Survey on Model-based Testing Approaches: A Systematic Review, WEASELTech'07, November 5, 2007, Atlanta Georgia, USA,Copyright ACM, pp. 31-36.

Bass, L., Clements, P., Kazman, R., (2003). Software architecture in practice, Boston Addison Wesley.

Bertolino, A., De Angelis, G., Frantzen, L., Polini, A., (2008). Model-based generation of testbeds for web services. In Proc. of TESTCOM/FATES, volume 5047 of LNCS, Springer, pp. 266-282.

Bill, H., Helmut, G., Klaus, B., (2008). Model-Based Testing of System Requirements Using UML Use Case Model, Software Testing, Verification, and Validation, International Conference on, 9-11 April 2008, Siemens AG, pp. 367 - 376.

Bruno, M., Canfora, G., Di Penta, M., Esposito, G., Mazza, V., (2005). Using Test Cases as Contract to Ensure Service Compliance Across Releases. RCOST-Research Centre on Software Technology, University of Sannio, Palazzo ex Poste, Via Traiano 82100, Benevento, Italy. In Benatallah et al. pp. 87-100.

Carlos, M., Santosh, S., Jon, C., Panos, G. (2004). On the Monitoring of Contractual Service Level Agreements, UK, Proceedings of the First International Workshop on Electronic Contracting (WEC'04), IEEE. 6 July 2004, Sch. of Comput. Sci, Newcastle Univ., UK, pp.1-8.

Chow, T., (1978). Testing software design modeled by finite-state machines. IEEE Transactions on Software Engineering, SE-4(3), pp.178-187.

Christoph, B., Hannes, K., Michael, K., Riccardo, G., Andreas, R.(2009) An Extensible Monitoring Framework for Measuring and Evaluating Tool Performance in a Service-Oriented Architecture, Vienna University of Technology, Vienna, Austria. ICWE Springer-Verlag Berlin Heidelberg 2009, LNCS 5648, pp. 221–235.

Colomb, R., (2009). Metamodelling and Model-Driven Architecture. Unpublished.

Colomb, R.M., Raymond, K., Hart, L., Emery, P., Welty, C., Xie, G.T., Kendall, E., (2006). The Object Management Group Ontology Definition Metamodel In: Calero, C., Ruiz, F., Piattini, M. (eds.). Ontologies for Software Engineering and Software Technology. Berlin Heidelberg: Springer, pp. 217-247.

Compuware, (2006). Applied Performance Management. Survey. Date Access 23-04-2009.

David, A., Xavier, F., (2008). Service Level Agreement Monitor (SALMon), Composition-Based Software Systems, ICCBSS 2008. Seventh International Conference on. 25-29 Feb. 2008, pp. 224-227.

Decker, G., Von Riegen, M., (2007). Scenarios and techniques for choreography design. In Proc. of BIS'08, LNCS 4439, Springer, pp. 121-132.

Di Modica, G., Orazio, T., Lorenzo, V. (2009). Dynamic SLAs management in service oriented environments. Journal of Systems and Software archive Volume 82 Issue, Elsevier Science Inc. New York, NY, USA, pp. 759-771.

Diana B., Boyan, B. (2009). Design of Service Level Agreements for Software Services, International Conference on Computer Systems and Technologies. ACM New York, NY, USA, Article No.26, pp. 13-1-13-6.

Domenico, B., Walter, B., Mauro, D., (2009). Automated Performance Assessment for Service-Oriented Middleware, USI-INF Technical Report, on site at NASA Ames Research Center. WWW '10 Proceedings of the 19th international conference on World wide web, ACM New York, NY, USA ©2010, pp. 141-150.

Dušan, O., André, H., Zora, K., Milan, V., (2013) SLA-Driven Adaptive Monitoring of Distributed Applications for Performance Problem Localization, Computer Science and Information Systems 2013,Published by ComSIS Consortium, Volume 10, Issue 1, pp. 25-50.

Ed, M., William, A., Sriram, B., David, C., John, M., Patrick, P., Soumya, S., (2010). Testing in Service-Oriented Environments, Software Engineering Institute. the U.S. Department of Defense Research, Technology, and System Solutions (RTSS) www.sei.cmu.edu.

Edward, W., (2002). Sun Professional Services Sun BluePrints OnLine – April 2002 Service Level Agreement in the Data Center, www.sun.com/blueprints. Access date 09-05-2009.

Freeman, P., (1987). Software Perspectives: The System is the Message. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA ©1987.

Fujiwara, S, Von Bochmann, F. Khendek, (1991). Test selection based on finite state models. IEEE Transactions on Software Engineering, pp. 591-603.

Glinz, M., (2007). On Non-Functional Requirements. In: 15th IEEE International Requirements Engineering Conference, pp. 21-26.

Guan, M, (1962). Graphic programming using odd and even points. Chinese Mathematics, pp. 273-277.

Harel, D., (1987). Statecharts: A visual approach to complex systems. Science of Computer Programming, pp. 231-274.

Harel, D., Naamad, A., (1996). The STATEMATE semantics of Statecharts. ACM Transactions on Software Engineering and Methodology, pp. 293-333.

Hasselmeyer, P., Koller, B., Kotsiopoulos, I., Kuo, D., Parkin, M., (2007). Negotiating SLAs with Dynamic Pricing. Policies, www.hasselmeyer.com /pdf/socinside07.pdf. In Proceedings of the SOC@ Inside'07. Vienna, Austria, September 2007. pp. 28-32.

Heiko, L., Alexander, K., Asit, D., Richard, P., (2003). Web Service Level Agreement (WSLA) Language Specification, Version: 1.0. IBM Web Services Toolkit;www.alphaworks.ibm.com/tech/webservicestoolkit, and dwdemos. alphaworks. ibm.com/wstk/common/wstkdoc/ services/utilities/Wslaauthoring / WebServiceLevelAgreementLanguage.html. Date access 10-05-2010.

Hennie, F.(1964). Fault detecting experiments for sequential circuits. In Proceedings of the 5th Annual Symposium on Switching Circuit Theory and Logical Design. pp. 95-110.

Henver, A.R., March, S.T., Park, J., Ram, S., (2004). Design Science in information Systems Research. MIS Quarterly, Vol. 28, No. 1, pp. 75-105.

Hoorn, A., Hasselbring, W., Waller, J., (2012). Kieker: A Framework for Application Performance Monitoring and Dynamic Software Analysis. Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering (ICPE 2012).ACM, Boston, Massachusetts, USA. pp. 247-248.

Hueppi, F., Wrage, L., Lewis, G., (2008). T-Check in Technologies for Interoperability: Business Process Management in a Web Services Contex. P 45.

Jianmin, D., Zhuo, Z., (2012). Towards Autonomic SLA Management: A Review. International Conference on Systems and Informatics (ICSAI). IEEE, pp. 2552-2555.

Juszczyk , L., Truong, H-L., Dustdar, S., (2008). GENESIS - a framework for automatic generation and steering of testbeds of complex web services. In Proc of ICECCS, IEEE Computer Society. Belfast, Northern Ireland. http://www.infosys.tuwien.ac.at/prototype/Genesis/. Date access 12/03/2010.

Kaner, C., Falk, J., Nguyen, H., (1999). Testing Computer Software, 2nd ed. Hoboken, NJ, USA: John Wiley & Sons, Inc.

Keller, A., Ludwig, H., (2003). The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services, Journal of Network and Systems Management. pp.5-13.

Keller, R., (1976). Formal verification of parallel programs. Communications of the ACM, pp. 371–384.

Kerber , L., (2001). Scenario-Based Performance Evaluation of SDL/MSC-Specified Systems, in Performance Engineering: State of the Art and Current Trends, Springer, LNCS vol. 2047.

Konig, H., (2004). Protocol Engineering Prinzip, Beschreibung und Entwicklung von Kommunikationsprotokollen (XLeitfäden der Informatik), German Edition. Amazon.com. Date access 16/05/2009.

Kronlöf, K., (1993). Method integration: Concepts John Wiley & Sons. Published by John Wiley and Sons Ltd., Chichester, U.K. Software Testing, Verification and Reliability, Volume 3, Issue 2, pp. 113-11.

Larus, R., Ball, T., (1994). Rewriting executable files to measure program behavior. Software: Practice and Experience 24(2), pp. 197-218.

Lawrence, C., Julio, C., do Prado, L., (2009). On Non-Functional Requirements in Software Engineering. A.T. Borgida et al. (Eds.): Mylopoulos Festschrift, LNCS 5600, Springer-Verlag Berlin Heidelberg, pp. 363-379,

Len, B., (1998). Software Architecture in Practice, Second Edition. Carnegie Mellon, Software Engineering Institute (SEI).

Len, B., Paul, C., Rick, K., (2003). Software Architecture in Practice, 2 ed.: Addison-Wesley.

Liang, Z., Sherif, S., Anna, L., (2013), SLA-Based and Consumer-Centric Dynamic Provisioning for Cloud Databases. Fifth International Conference on Cloud Computing 24-29 June 2012. IEEE. pp. 360 - 367.

Ludwig, H., Keller, A., Dan, A., King, R., Franck, R. (2005).Web Service Level Agreement (WSLA) language specification.*www.research.ibm.com/wsla/ WSLASpecV1-20030128.pdf*.

Mark. U., Legeard B., (2007). Practical model-based testing, a tools approach. Morgan Kaufmann Publishers. San Francisco, CA 94111.

Martin, G. (2007). On Non-Functional Requirements, Delhi, India. Proceedings of the Requirements Engineering Conference, 2007. RE '07. 15th IEEE International, pp. 21 - 26.

Matinlassi, M., Quality-driven Software Architecture Model Transformation. PhD Dissertation. University of Oulu. (2006).

Matinlassi, M., Niemela, E., (2003). The impact of maintainability on component-based software systems. In: Proceedings of the 29th Euromicro Conference. Antalya, Turkey. IEEE. pp. 25 - 32.

Mitschele, A., Muller, B., (1999). Performance Engineering of SDL/MSC Systems, Journal on Computer Networks and ISDN Systems.

Msexchange, (2002). Blueprint for an Exchange Service Level Agreement, www.msexchange.org. Date access 19-05-2009.

Naito, S., Tsunoyama, M., (1981). Fault detection for sequential machines by transitiontours In Proceedings of the IEEE International Symposium on Fault Tolerant Computer Systems, World Enformatika Society. pp. 238–243.

Nethercote, N., Seward, J., (2007). Valgrinda framework for heavyweight dynamic binary instrumentation. SIGPLAN Not. pp. 89–100.

Nicolai, M., Josuttis., (2007). SOA in Practice. Publisher O'Reilly Media.

OASIS., (2000). Organization for the Advancement of Structured Information, Standard http://www.oasis-open.org/home/index.php. Date access 11-12-2008

OMG (2006a). MOF Core specification version 2.0 OMG formal/2006. Date access 15-08-2009.

OMG (2007b). Unified Modeling Language (OMG UML), Infrastructure, V2.1.2 OMG Document Number: formal/2007-11-04.

OMG, (2003). MDA Guide V1.0.1. OMG document omg/03-06-01.

OMG, (2007). OMG Unified Modeling Language (OMG UML), Superstructure, V2.1.2, OMG Document Number, A user-oriented software reliability model. IEEE Trans. Softw. Eng. 6(2), 118–125.

OMG.(2009). Handling Non-Functional Properties in a Service Oriented Architecture Request for Information. Date access 2011-02-12.

Opdahl, A., (1995). Sensitivity analysis of combined software and hardware performance models: Open queuing networks, Performance Evaluation. pp. 75-92.

Paul, B., Zhen, R., Jens, G., Øystein, H., Ina, S., Clay, W., (2007). Model-Driven Testing Using the UML Testing. Software Engineering, Springer. pp. 185.

Paul, R., (2005). DoD towards software services in WORDS '05: Proceedings of the 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems, pp. 3-6.

Peltz, C., (2003). Web services orchestration and choreography, Computer, vol. 36, pp. 46-52.

Penta, M., Bruno, M., Esposito, G., Mazza, V., Canfora, G., (2007) Web Services Regression Testing, in Test and Analysis of Web Services. Heidelberg: Springer Berlin. pp 205-234.

Petter,L. Eide, H., (2005) Quantification and Traceability of Requirements. Ph.D. Thesis, Norweiag University of Science and Technology.

Philip, B., Lewis, A., (2008). Service Level Agreements in Service-Oriented Architecture Environments. Carnegie Mellon, Technical Report CMU/SEI-2008-TN-021.

Platzer, C., Rosenberg, F., Dustdar, S., (2007).Enhancing Web Service Discovery and Monitoring with Quality of Service Information. In: Securing Web Services: Practical Usage of Standards and Specifications, Idea Publishing Inc. Politècnica de Catalunya. Securing Web Services, pp. 19.

Sabnani, K., and Dahbura, A.,(1988). A protocol test generation procedure. Computer Networks and ISDN Systems, pp. 285-297.

Sebastian, W., Alin, S., Großmann, J. (2008). Enabling Model-Based Testing for SOA Integration testing, Proc. of Workshop on Model-based testing in practice Fraunhofer IRB Verlag, 2008, pp. 73-82.

SEI, (2010). www.sei.cmu.edu/index.cfm. Date access 15-03-2010.

Sherif, S., Anna, L., (2012). SLA-Based and Consumer-Centric Dynamic Provisioning for Cloud Databases. IEEE Fifth International Conference on Cloud Computing. Hawaii-USA, pp. 8.

Sholl, H., Booth T., (1975). Software Performance Modeling Using Computation Structures , IEEE Transactions, Software Engineering, pp. 414 – 420.

Simon, E., Graham, M., (2011). Toward reusable SLA monitoring capabilities, School of Computing Science, Newcastle University, U.K.

Smith , C., Williams, L., (2002). Performance Solutions, Addison-Wesley.

Steinberg, D., Budinsky, F., Paternostro, M. & Merks, E.,(2008). EMF: Eclipse Modeling Framework.2nd Edition. Addison-Wesley Professional.

SWEBOK, (2004). Guide to the Software Engineering Body of Knowledge, Version, executive eds. A.Bran and A.W. Moore, eds. P. Borque, R. Dupuis,project champion L.Tripp, the IEEE Computer Society, Accessed May, Browse Journals & Magazines .17, 2008.

Thbs , www.thbs.com/services-oriented-architecture.html. Date access 02-01-2010.

Thimbleby, H., (2003). The directed Chinese postman problem. Software: Practice and Experience, pp.1081-1096.

Thomas, E., (2007) SOA Principles of Service Design. Book Soa: principles of service design First,Prentice Hall Press Upper Saddle River, NJ, USA.

Thomas, G., (2011). A typology for the case study in social science following a review of definition, discourse and structure. Qualitative Inquiry. pp. 511-21.

TMForum,(2008), Telemanagement (TM) Forum. SLA Handbook Solution Suite V2.0.www.tmforum.org/DocumentLibrary/SLAHandbookSolution/2035/Home. htmlH, Date access 22-04-2010.

Tretmans, J. (2004). Model-based testing: Property checking for real. Keynote address at the International Workshop for Construction and Analysis of Safe Secure and Interoperable Smart Devices, Available. sop.inria.fr/everest/events/cassis04. Date access 03-07-2009.

Vasco, A., Anacleto, C., Fernando, B., (2011), SLALOM: a Language for SLA Specification and Monitoring, 1 CITI/FCT/UNL, 2829-516 Caparica, Portugal 2 IPS/EST, 2910-761 Setúbal, Portugal 3 DCTI, ISCTE-IUL, 1649-026 Lisboa, Portugal. pp. 556-567, 8-9.

Vasilevskii, M., (1973) Failure diagnosis of automata. Cybernetics, 9:653–665. Translated from Kibernetika 4, 98–108.

Vinod, M., Hans, J., Tony, C., Allen, C., Phil, C.,(2009), SLA-Driven Business Process Management in SOA, Paper appears in CASCON 2009, Richmond Hill, Ontario, Canada.

Vittorio, C., Vincenzo, G., (2007). Reliability modeling and analysis of service-oriented architectures. Test and Analysis of Web Services , 2007, pp. 339-362.

W3C, (2001), Web Services Description Language (WSDL) 1.1, http://www.w3.org/TR/wsdl. Date access 08-05-2010.

Whittaker, J., Whittaker and Thomason, M., (1994). A Markov chain model for statistical software testing. IEEE Transactions on Software Engineering. pp. 812 - 824.

Wieczorek, S., Stefanescu, A., and Großmann, J., (2008). Enabling model-based testing for SOA integration testing, In Proc. of Model-based testing in practice (MOTIP'08), satellite workshop of ECMDA, pp. 77-82.

Woodside M., Petriu D.C., Petriu D.B., Shen H., Israr T., Merseguer J.,(2005). "Performance by Unified Model Analysis (PUMA)", Proc. WOSP. pp. 1-12.

Yang, B., Ural, H., (1990). Protocol conformance test generation using multiple UIO sequences with overlapping. In Proceedings of the ACM Symposium on Communications Architectures & Protocols, ACM Press, pp. 118-125..

Yin, R., (2009), Case study research: Design and methods, Sage publications.

Zain, B., Mohd, B., (2010), Agent based Monitoring Framework for SOA Applications Quality, Computer and Information Science Dept.Universiti Teknology Petronas, IEEE, pp. 1124-1129.

Zhang, S., Song, M., (2010), An Architecture Design of Life Cycle Based SLA, Advanced Communication Technology (ICACT), International conference pp. 1351-1355.