CERVANTES: A Model-Based Approach for Service-Oriented Systems Development

L. García-Borgoñón *laurag@itainnova.es*
ITAINNOVA Aragon Institute of Technology / IWT2 Research Group University of Seville
Zaragoza / Seville, Spain

M.A. Barcelona *mabarcelona@itainnova.es*
ITAINNOVA Aragon Institute of Technology / IWT2 Research Group University of Seville
Zaragoza / Seville, Spain

J.I. Calvo *jicalvo@itainnova.es*
ITAINNOVA Aragon Institute of Technology
Zaragoza / Seville, Spain

I. Ramos *iramos@us.es*
IWT2 Research Group University of Seville
Zaragoza / Seville, Spain

M.J. Escalona *mjescalona@us.es*
IWT2 Research Group University of Seville
Zaragoza / Seville, Spain

Abstract

Context: The benefits of the Model-Driven Software Engineering application for Service-Oriented Computing.
Objective: This paper proposes a Model-Based approach for Service-Oriented Systems Development.
Method: Following the Model-Driven Reverse Engineering process, from the models discovery, to the generation of the current Model-Based Approach for Service-Oriented Systems Development.
Results: The CERVANTES metamodel is presented and compared to other initiatives.
Conclusions: This study shows how Model-Driven Engineering can be used to develop Service-Oriented Systems in practice.
Keywords: Service-Oriented System Engineering, Model-Driven Engineering, Service-Oriented Computing.

1. Introduction

Computer systems have had a great evolution in the latest years, and software researchers and developers have been creating abstractions that help them. From the first monolithic programs, which had been developed with defined functions and autonomous behavior, to current systems with greater functionalities which operate in heterogeneous and distributed environments by interacting with other systems, many approaches have been proposed to manage the system and software development complexity.

In the area of distributed systems, Service-Oriented Computing (SOC) is the computing

paradigm that utilizes services as the basic constructs to support the development of rapid, low-cost and easy composition of distributed applications even in heterogeneous environments [15]. SOC proposes to develop technology neutral and loosely coupled systems[14].

Moreover, Model-Driven Software Engineering (MDE) is a software development paradigm which focuses on creating and exploiting domain models (that is to say, abstract representations of knowledge and activities that govern a domain specific application) as a means of alliviating the systems complexity and express domain concepts effectively [16]. Models, metamodels and transformations are the key elements in MDE [5].

In 2001 we set out to create a methodological approach, named CERVANTES, that could be reused in the design and development of distributed systems. This approach was based on these assumptions: i) to follow an architecture-driven development: current and future functionalities should be considered in the architectural design; ii) the design of the subsystems should be independent of technology, platforms, programming languages or messaging protocols; iii) the internal behavior of each subsystem must be semi-formally described giving rise to subsequent validation, and; iv) runtime logs will have valuable information about usage of the system in order to reconfigure a deployment or redesign to improve performance, even dinamically.

Since its definition, CERVANTES has been widely used for the systems design and development within the group, being present in many solutions, which are currently deployed in different domains and real environments.

With the appearance of MDE, we proposed to use this paradigm to facilitate the CERVANTES use. First of all, we needed to change to a model-based perspective, a higher-level representation of our legacy system, as has been recently proposed in Model-Driven Reverse Engineering (MDRE) [4]. MDRE aims to create a set of models that represent the system, and includes three main phases: model discovery, model understanding and model (re)generation.

In this paper, we present the MDRE process for CERVANTES, from the models discovery, to the generation of the current model-based approach for service-oriented systems development. To this end, at the beginning we present the original runtime framework that manages the message interchange and the internal business process, and then, as a result of the MDRE process, the metamodel for distributed systems design and a tool that generates the source code will be shown.

As we said above, this paper is structured following the three main phases of a MDRE process: First, in Section 2, we talked about the Model Discovery phase where the previous components of CERVANTES are presented . Section 3 shows the result of the Model Understanding pha se by introducing the CERVANTES metamodel and, in Section 4, the IDE followed by an example of application in practice is presented as the result of the Model Generation phase. A review of related work and a comparison with other alternatives is provided in Section 5. Finally Section 6 concludes and outlines future work.

2. CERVANTES Framework: The origins

In MDRE, the idea is to switch as soon as possible from the heterogeneous real world to the homogeneous world of models. To achieve this goal, it will be detailed the CERVANTES framework, a basic infrastructure that supports the system of systems execution.

This runtime allows the deployment of distributed systems designed as exchanging information through asynchronous *Messages* that are processed by *State Machines*. The framework is based on the following components, as described in the Figure 1a): i) *Core Framework*: contains the common elements to all infrastructure like activation and shutdown mechanisms, metrics, exceptions and distributed logger; ii) *Control Framework*: it encapsulates the logic when a *Message* is received; iii) *Boundary Framework*: responsible for establishing communication with other subsystems by asynchronous messages using various protocols such as Java RMI, .NET Remoting, XML-RPC or SOAP, and; iv) *Entity Framework*: stores the information of the var-

ious *State Machines* in execution on a separate shared memory space for each subsystem and maintains a mirror of the same in a persistent medium.

Based on the previously exposed architecture, the dynamic behavior of the framework follows a sequence of steps, as seen in Figure 1b): 1. A Request is received by the *Server* following a concrete protocol; 2. The *Server* translates to the internal Messaging format and sends to the *Manager*; 3. The *Message* is stored in a priority sorted queue; 4. The Producer-Consumer pattern is used to process the *Message* by the *Manager*; 5. If it was a Request, a *State Machine* to process it is created. If it was a Response, it's automatically sent to the *State Machine* which was waiting for it; 6. The *State Machine* waits for the Token to restart its execution (to avoid race conditions all inter-dependent *State Machines* are managed by a single Thread); 7. When it is the turn of execution, the logic of the workflow starts; 8. Usually every request is associated with a response, so that the end result of the execution of the *State Machine*, the invoked subsystem transmits a final answer. To send the results the same Messaging system is used, so the result is transferred to the *Manager* as a proxy to the *Boundary Framework*, and; 9. The Response is sent by the *Sender* and confirms to finalize the *State Machine*.

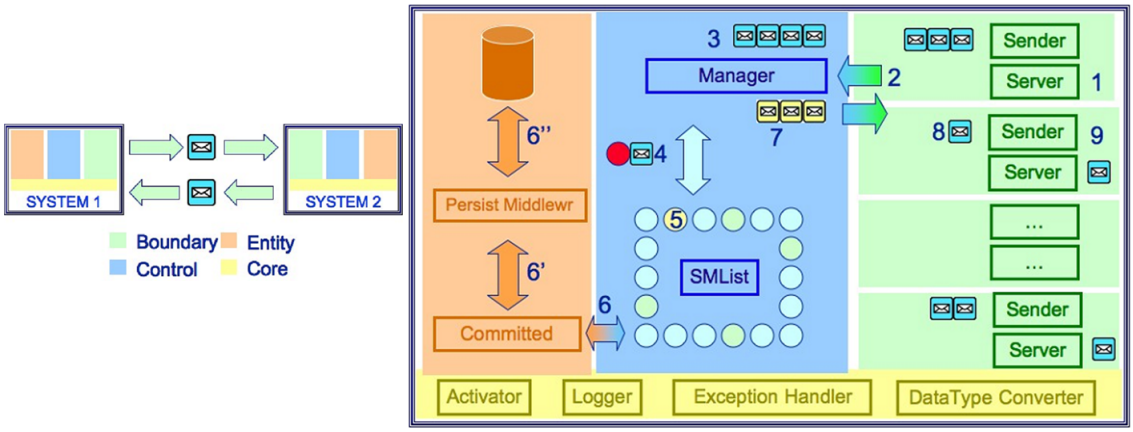

Figure 1. a) CERVANTES Framework Components; b) Runtime steps

This Framework has been the starting point for the Model Discovery Phase according to MDRE process.

3. CERVANTES Metamodel

The following phase is Model Understanding. The goal of this phase is to obtain a kind of manageable representation in MDE. The way we have worked this phase is by inferencing of the support CERVANTES metamodel. This metamodel will be described below.

A CERVANTES system is composed by a set of *CervantesSubsystems* which can be deployed in a distributed and heterogeneous enviroment. The behaviour of every *CervantesSubsystem* is modelled by a set of *StateMachines* which are composed by a set of *States* and their *Transitions*. A *StateMachine* has at least a *initial* and *finalOK States* mandatory, and it maybe a *finalError State* also. A *Transition* is triggered by a *Message* sending. *Messages* are divided into *Requests*, *Responses* and *Events*. A *Response* can be successful (*OK* response) or *error*. The CERVANTES metamodel can be shown in Figure2.

4. Model Generation

The following phase is Model Generation where the models obtained at the Model Understanding Phase are finally used to generate the expected outcome. In this Section the Integrated Development Environment (IDE) tool kwown as CERVANTES Studio is presented and a practical

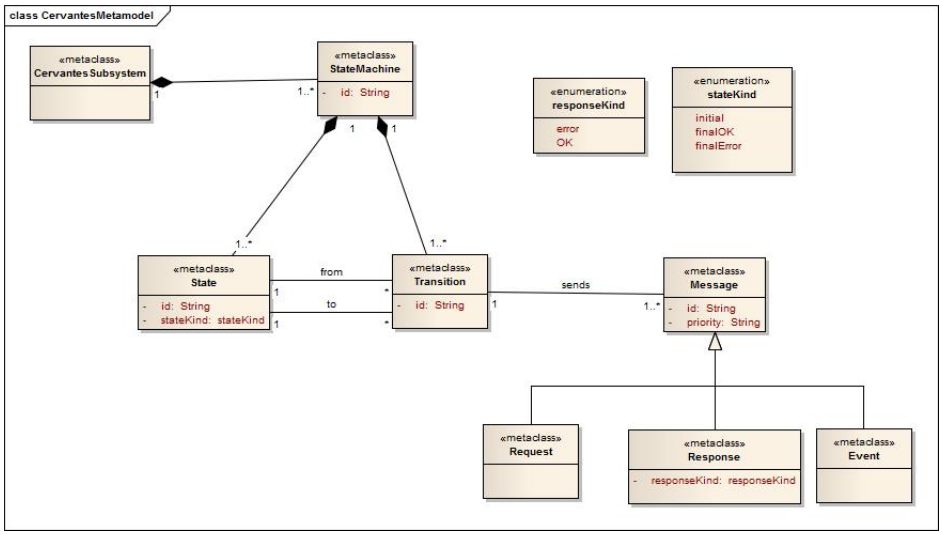

Figure 2. CERVANTES metamodel

case study is exposed.

In order to make use of the metamodel CERVANTES Studio was developed. Based on a Eclipse Modeling Framework (EMF)[6] and Graphical Modeling Framework (GEF)[7], this tool allows the software engineer to model the system of systems, by creating the *State Machines* and the *Messages* and generates the source code for specific platforms, in particular, for Java and C#.net. A screenshot of the *State Machine* Editor is shown in Figure 3.

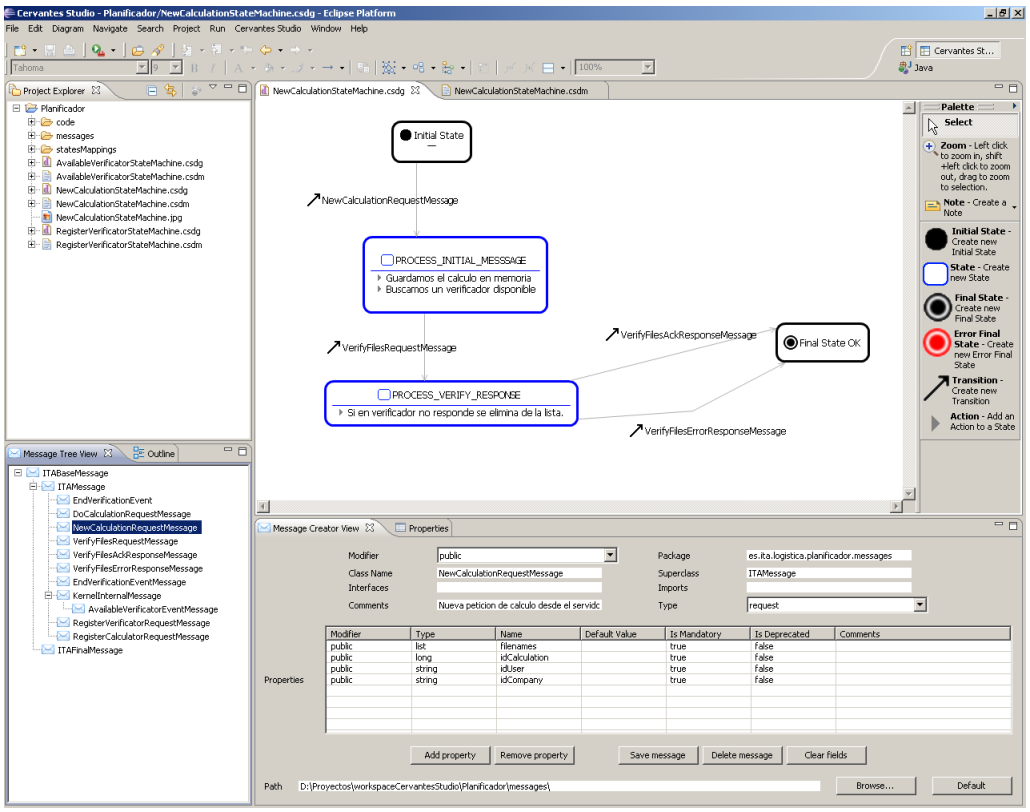

Figure 3. CERVANTES Studio Screenshot

The IDE also includes a metrics monitor. Once selected the metrics server of a CERVANTES Framework instance (namely, server and port), the user can navigate through the

model and for every *State Machine* monitor: i) execution Time (latest, on average, maximum); number of class instances (current, maximum); average execution time per state. This tool is very useful in order to improve the performance of the system by taken usage information.

Since its creation, the CERVANTES framework has been widely used for the design and development of systems within the group, being present in many solutions currently deployed in real environment in different domains and technologies, like telecomunications, healthcare, tourism or logistics, by usign C++, Java and C#.NET programming languages.

As an example of its application in practice, we will describe how CERVANTES was used for the design and development of an Intelligent Taxi Fleet Management System, which was composed of a set of *Cervantes Subsystems* as it is shown in Figure 4: i) *Comm*: the module to bridge connections to the fleet by using different protocols; ii) *Kernel*: the core of the system which maintains the main connection between the rest of the modules; iii) *PBX*: the module to integrate a telephone server; iv) *GIS*: the module which maintains the geographical information system and calculates the best route; v) *Database*: the core of the persistence data of the system; vi) *Operator*: the module to achieve the human interaction.

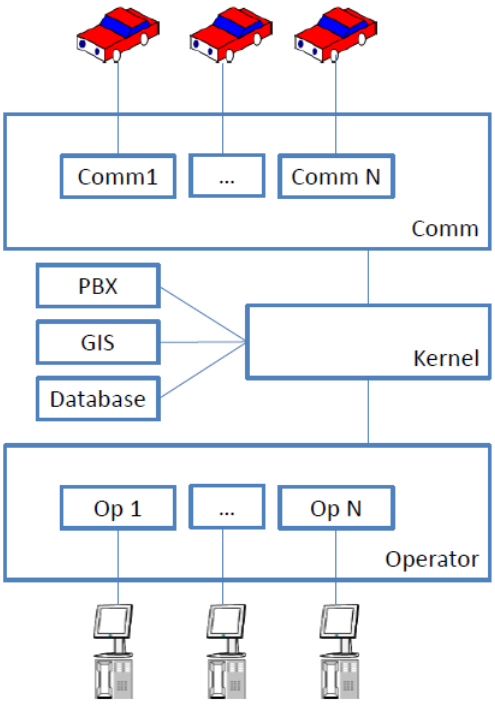

Figure 4. Intelligent Transportation System Architecture

Every subsystem can be deployed in a distributed system and, according to the metamodel, their interfaces are described by using *Messages* and their internal behavior by *State Machines*. An example of the State Machine which processes the Request of an Incoming Call is shown in Figure 5.

5. Related Work and Comparison

Once the MDRE process was finalized, we have a model-based approach for Service-Oriented Systems Development. In this section we have included a general review of similar proposals in the literature, with the aim of knowing them and comparing the main characteristics with our approach.

The Model-Based approach for development of Service-Oriented systems have been tackled by plenty of authors in the last few years. Some of most relevant proposals are:

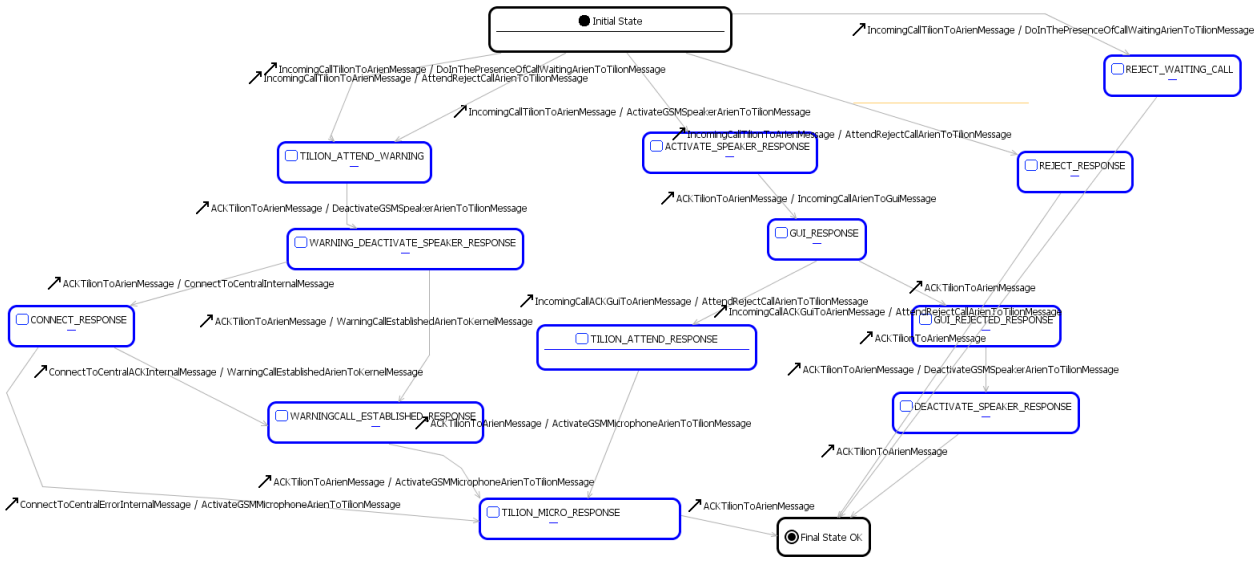

Figure 5. New Incoming Call StateMachine

- a proposal for modelling Service-Oriented Architectures with UML[10];
- a service-oriented modeling framework that employs an agile and universal business and technology language to facilitate analysis, design, and architecture initiatives[3];
- a service-oriented modeling framework that employs an agile and universal business and technology language to facilitate analysis, design, and architecture initiatives[3];
- a service-oriented modeling framework that employs an agile and universal business and technology language to facilitate analysis, design, and architecture initiatives[3];
- an arquitectural-model based approach for web applications[2];
- a UML profile to graphically design the non-functional aspects in SOA[18];
- a UML profile to graphically design the non-functional aspects in SOA[18];
- a UML profile for software services which allows for the modeling of services, service-oriented architecture (SOA), and service-oriented solutions, by IBM[9];
- a Service-Oriented Modeling Framework (SOMF) as a Model Driven Engineering Methodology that Offers Used-to-Be, As-Is, and To-Be Business and Technological Perspectives of Service-Oriented Modeling[17];
- a UML model to represent SOA static concepts ang graph transformations for the behavior[1], and;
- a UML profile for SOA modelling following MDA principles[8];

The main distinguishing factor of our approach regarding other proposals is that CERVANTES has been validated widely by industry, because it has been included in multiple projects in real environments. These real environments are source information for the feature of metrics in execution time allows us to learn and improve the approach.

Regarding related standards, in 2012 the OMG[11] published the Service oriented architecture Modeling Language(SoaML)[13] specification which provides a metamodel and a UML

profile for the specification and design of services within a service-oriented architecture. Its profile provides the flexibility for tool vendors having existing UML2 tools to be able to effectively develop, transform, and exchange services metamodels in a standard way. Its metamodel can be extended for specific purposes. It seems that this proposal has become a standard for SOC modeling. The relationship between SOC and business process seems to be fully addressed by using the OMG Business Process Model and Notation (BPMN)[12] specification. Both are standards and, as such, they try to address the higher quantity of concepts as possible. This fact produces that the learning curve for using in the industry will be increased significantly. Our proposal is focused on being easy-to-learn, simple and common to the service-oriented systems. Again, the practice application in the industry supports our proposal.

6. Conclusions and Future work

This paper presents a model-based approach for Service-Oriented Systems development. This approach is the result of a Model-Driven Reverse Engineering, and the three main phases of the MDRE process have been exposed. The process starts with the CERVANTES framework, which has been widely used for systems development since 2001. According to the process, the paper introduces a metamodel to supports CERVANTES and an integrated development environment (IDE) for using in a MDE way.

From this work and due to CERVANTES has been used in several real projects, some relevant conclusions can be deduced from this experience.

Firstly, a model-based mechanism can be very useful in a Service-Oriented development environment but it should be easy-to-learn for its application in the industry. A simple abstract syntax but tool-supported is a better option than a more sophisticated syntax that incorporates a lot of concepts. However, the usability of the tool by non-experts software developers is very important too, so we are working in a better solution for the MDE tool, as could be Enterprise Architect, widely used in the IWT2 group, where this paper has been developed.

Secondly, the MDRE process is a very useful proposal for including a model-based approach in legacy systems, with the benefits of complexity management that MDE incorporates. We are working in defining a method to automate the three phases and systematize the use in legacy systems.

Finally, we are trying to improve the CERVANTES framework and incorporate new features, as the definition a methodology that convers the entire development lifecycle of a service-oriented systems, or the use of the metrics of usage to support automatic reconfiguration at runtime, as a first implementation of Self-configurement services which configures themselves automatically to adapt to different environments in which they can be installed and can operate to optimize for paticular kinds of their use[15].

Acknowledgements

This work has been partially funded by the projects ITCHAIN (Ministry of Industry, Energy and Tourism of Spain through the AVANZA Plan TSI-020302-2010-80), the NDTQ-Framework project of the Junta de Andalucía, Spain (TIC-5789) and MEGUS (Ministry of Economy and Competitiveness of Spain TIN2013-46928-C3-3-R).

References

1. Baresi, L., Heckel, R., Thöne, S., Varró, D.: Modeling and validation of service-oriented architectures: application vs. style. In: ACM SIGSOFT Software Engineering Notes, vol. 28, pp. 68–77. ACM (2003)
2. Beigbeder, S.: Websa: un método de desarrollo dirigido por modelos de arquitectura

para aplicaciones web (2007)
3. Bell, M.: Service-oriented modeling. John Willey & Sons, Inc (2008)
4. Brambilla, M., Cabot, J., Wimmer, M.: Model-driven software engineering in practice. Synthesis Lectures on Software Engineering **1**(1), 1–182 (2012)
5. Cetinkaya, D., Verbraeck, A.: Metamodeling and model transformations in modeling and simulation (2011)
6. EMF: Eclipse modeling framework project. Eclipse Foundation (2014). Http://www.eclipse.org/modeling/emf/?project=emf, last accessed 04/2014
7. GMF: Graphical modeling project. Eclipse Foundation (2014). Http://eclipse.org/gmf-tooling/, last accessed 04/2014
8. Heckel, R., Lohmann, M., Thöne, S.: Towards a uml profile for service-oriented architectures. In: Model Driven Architecture: Foundations and Applications, p. 115. Citeseer (2003)
9. Johnston, S.: Uml 2.0 profile for software services. IBM developerWorks http://www.ibm.com/developerworks/rational/library/05/419_soa/ Accessed 2014-04-10 (2005)
10. López-Sanz, M., Acuña, C.J., Cuesta, C.E., Marcos, E.: Modelling of service-oriented architectures with uml. Electronic Notes in Theoretical Computer Science **194**(4), 23–37 (2008)
11. OMG: Object management group, http://www.omg.org, accessed 2014-04-10
12. OMG: Omg's business process model and notation (bpmn), http://www.bpmn.org, accessed 2014-04-10
13. OMG: Omg's service oriented architecture modeling language (soaml), http://www.omg.org/spec/soaml, accessed 2014-04-11
14. Papazoglou, M.P.: Service-oriented computing: Concepts, characteristics and directions. In: Web Information Systems Engineering, 2003. WISE 2003. Proceedings of the Fourth International Conference on, pp. 3–12. IEEE (2003)
15. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-oriented computing: a research roadmap. International Journal of Cooperative Information Systems **17**(02), 223–255 (2008)
16. Schmidt, D.C.: Model-driven engineering. COMPUTER-IEEE COMPUTER SOCIETY- **39**(2), 25 (2006)
17. SPARX: Service-oriented modeling framework (somf) , http://www.sparxsystems.com.au/somf, accessed 2014-04-11
18. Wada, H., Suzuki, J., Oba, K.: Modeling non-functional aspects in service oriented architecture. In: Services Computing, 2006. SCC'06. IEEE International Conference on, pp. 222–229. IEEE (2006)