Non-Functional Requirements for Service-Based Applications: A Systematic Review

Plácido A. Souza Neto[1] **, Martin A. Musicante**[2]
Genoveva Vargas-Solar[3] **, Valeria de Castro**[4]
Umberto Souza da Costa[2]

[1] Instituto Federal do Rio Grande do Norte
Campus Natal-Central – RN - Brazil

[2]DIMAp - Universidade Federal do Rio Grande do Norte
Campus Universitário – Natal – RN – Brazil

[3]Université de Grenoble
Saint Martin d'Hères – France

[4]Universidad Rey Juan Carlos
Móstoles – Spain

placido.neto@ifrn.edu.br, {umberto, mam}@dimap.ufrn.br

genoveva.vargas-solar@imag.fr, valeria.decastro@urjc.es

Abstract. *This paper presents a systematic literature review of non-functional requirements (NFRs) for service-based applications. The main goal of the review is to identify the most common terms used to refer to NFRs. We also propose a classification of the most common terms, as well as, a model to integrate those terms.*

1. Introduction

Functional properties of a computer system are characterized by the effect produced by the system when given a defined input. Functional properties are not the only crucial aspect in the software development process. Other properties need to be addressed to fit in the application with its context. These other aspects are called Non-Functional Properties.

Non-Functional Requirements (NFRs) specify those properties that are not addressed by the functional specification. They are often called *qualities* of the software system. Non-Functional Requirements may specify response time, security constraints or quality of the solution, among others.

Service-Oriented Computing [22], is a software development paradigm where pre-existing services are combined to produce more complex applications. The development of service-based applications can benefit from the inclusion of NFRs to the software process from its early stages. Failure to comply with this inclusion means that the final application is obtained from a partial specification, making the deployment a difficult task. The adoption of non-functional specifications from the early states of development can help the developer to produce applications that are capable of dealing with their context.

Non-functional properties of service oriented applications have been addressed in academic works and standards [5, 8, 2]. Different proposals [3, 1, 7, 12, 30, 13, 28] support non-functional requirements in the context of web service development.

Most software development methods define software processes that use the notion of refinement. Software process begins with the formulation of an abstract specification, which is successively refined to yield the implementation of the system. Methods for the development of web service applications are no exception to this rule. At least two levels of abstraction can be distinguished: a *Business Level*, including the abstract specification, and a *System Level*, including actual computer programs that implements the system.

In the case of web service applications we will distinguish two separate layers of the implementation. The *Composition Layer* is the upper layer of the implementation. It defines the workflow of the system, in terms of individual service calls. The *Service Layer* defines those services that are called by the composition.

In this work we investigate the extent in which NFRs are considered by the development methods proposed in the literature. We have conducted a systematic review [16] to summarize the approaches that support NFRs. As a result, we propose a classification of NFRs for web services.

This paper is organized as follows: Section 2 presents a systematic literature review of non-functional requirements for service-based applications. The findings of the systematic review are presented in Section 3 and used in Section 4 to propose a classification and a model for NFRs in the context of web service development. Section 5 concludes the paper.

2. A Systematic Review

In this work we develop a systematic literature review of NFRs for web services. Our review considers the abstraction levels and implementation layers defined above. The purpose of our analysis is to: *(i)* Identify concepts, properties and notations related to NFRs and used in service-based system development; and *(ii)* Define a classification for these concepts, properties and notations, at different levels of abstraction.

We propose seven research questions (RQ_1 to RQ_7) to guide our analysis about non-functional requirements for web services. For each question we define a set of possible answers in order to guide the analysis. The questions are:

- RQ_1 : How NFRs are modelled by existing methodologies for Web services? Possible answers: *Answer is specific to each proposal.*
- RQ_2 : Which kinds of NFRs are considered more frequently? Possible answers: *Security / availability / portability / . . . / reliability / performance.*
- RQ_3 : What is the main underlying approach used by the proposal? Possible answers: *Model driven approach (MDD) / Ontologies (Ont) / Formal methods (FM) / Artificial intelligence (AI) / Business Process Modeling (BP) / Traditional (TDT).*
- RQ_4 : What is the scope of the proposal? Possible answers: *Software architecture / QoS model / Language definition / Methodology / etc.*

- RQ_5 : Does the paper propose a (meta)model for NFRs? Does the Business Level specification include NFRs? Possible answers: *yes / no – yes / no.*
- RQ_6 : Are non-functional aspects considered at the composition or single service level? Possible answers: *single / composition.*
- RQ_7 : What is the publication year of the paper? Possible answers: *Year of publication.*

We use the approach described in [16] for searching, collecting and selecting works related with NFRs for service-based applications. The results are used to identify and compare key concepts related to NFRs. The search engines used in our review include journals, conferences and workshops of recognized quality. We used the following search engines: *(i) IEEE Computer; (ii) ACM Digital Library;* and *(iii) Science Direct.*

The query used to perform the search is defined as:

```
(((non functional properties) OR (non functional
requirements)) AND web service AND composition))
```

The results of our query for each search engine were filtered in accordance to the following criteria:

1. Papers written in other languages than English were excluded.
2. Papers published in non-international conferences and workshops were also excluded.
3. Papers whose title and abstract do not refer to NFRs were excluded.

After this filtering process, the remaining papers were analyzed in full. A brief account of our findings is given in the next section.

3. Our Findings

In this section we analyze the results of the search query for our systematic review. The search was performed on three different scientific sources, namely IEEE Computer, ACM Digital Library (only journal papers) and Science Direct. We first show the contribution of each search engine, in terms of the number of relevant papers and year of publication. Next, we summarize the answers we have obtained to the questions of our systematic review.

	IEEE	ACM	Science Direct	Total
Total	76	88	248	412
Title/Abstract	25 (32.9%)	12 (13.6%)	30 (12.1%)	67 (16.3%)
Full text	11 (14.5%)	3 (3.4%)	12 (4.8%)	26 (6.3%)

Table 1. Number of papers per source.

Table 1 shows the number of papers obtained at each stage of the selection procedure, for each source. The first row shows the total number of papers returned by each engine, for the search query (Section 2). The next row of the table shows the results obtained after filtering the papers by considering their titles and abstracts. The percentage figures indicate the proportion of remaining papers, for each source. The last row of the table shows the number of relevant papers obtained, after considering the complete text

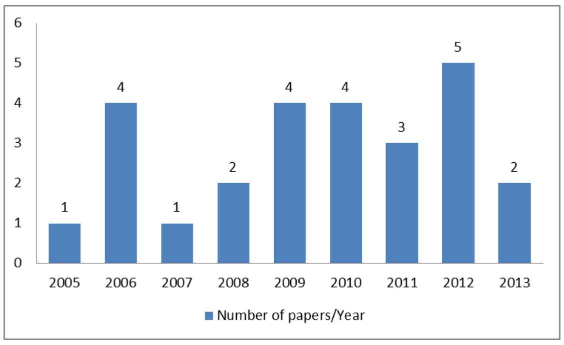

Figure 1. Publications per year.

of each paper. The percentages are relative to the total, for each source. The distribution of publication year for the 26 selected papers is shown in Figure 1.

For each of the 26 papers, we obtained answers for each research question. Our findings are presented in Tables 2 and 3.

Reference	RQ_1:***NFR concepts***	RQ_2:***Properties***	RQ_3:***Approach***	RQ_4:***Domain / Scope***
Babamir et al. [3]	property / category / constraint	responsiveness / availability performance / sla properties	TDT	Software architecture
Yeom et al. [31]	category / sub-category / property	business value / performance / stability / manageability /security / business processing interoperability	TDT	QoS model
Xiao et al. [30]	NF attribute	time / cost / resource	BP	Business processes modeling
D'Ambrogio [9]	characteristics / category / dimension	availability / reliability / access control	MDD	WSDL extension
Chollet et al. [7]	activity / NF attribute	security	MDD	Orchestration Framework
Schmeling et al. [25]	NF concern / NF attribute / NF action / NF activity	security	MDD	Web service composition process
Thißen et al. [26]	NF value	performance / reliability / cost / availability	FM	Software architecture
Zhang et al. [32]	attribute / predicate	security	FM	Access control
Basin et al. [4]	attribute	security	MDD	System architecture
Ceri et al. [6]	police / rule / condition / action	n.a.	TDT	Context-aware applications
Fabra et al. [11]	property	storage / processing	MDD	Web service methodology
Modica et al. [19]	quality level	sla properties	TDT	Service oriented architecture
Ovaska et al. [21]	attribute / category	security / reliability	MDD	Model development
Agarwa et al. [1]	property / policy / function	*not explicitly defined*	Ont	Policy language
Jeong et al. [13]	NF attribute	operation cost / performance / availability / accessibility / security / interoperability / usability / user satisfaction	AI	Service oriented architecture
Pastrana et al. [23]	NF property / contract / assertion / NF behaviour	performance / reliability / scalability / capacity / robustness / precision / security / accessibility / availability / interoperability	Ont	Web service methodology
Diamadopoulou et al. [10]	NF characteristic	user' subjective perception	TDT	Web service selection
Gutierrez et al. [12]	NF factor / NF sub-factor	Security	BP	WS development process
Mohanty et al. [20]	NF attribute / NF factor	reliability / performance / integrity / usability / response time / documentation	AI	Artificial intelligence / Web services classification
Karunamurthy et al. [15]	Non-functional parameters	cost / response time / availability / security / availability / reliability / reputation	BP	DSL (NFSL)
Liu et al. [18]	QoS parameter	cost / execution duration / accuracy / security / integrity / availability / reliability	FM	QoS model
Tran et al. [27]	QoS policies	performance / availability / security / response time / SLA aspects	MDD	Language definition / QoS model
Wang et al. [29]	Non-functional properties	response time / price / reliability / availability / platform / location / provider	N/A	Genetic Algorithm / QoS model
Li et al. [17]	Dimensions / QoS parameters	execution time / storage / reliability / service cost / communication time / message length / availability	TDT	QoS model
Kamalabad et al. [14]	QoS Attributes	reliability / availability / response time / performance / stability / accuracy / capacity / robustness / cost / scalability / throughput / efficiency / accessibility / successability / reputation / consistency / delivery time	Ont	Business Specification
Rumpel et al. [24]	Quality Requirement / Quality Property	runtime / cost management / security	N/A	Requirement definition

Table 2. Research question results - RQ_1, RQ_2, RQ_3, RQ_4.

Let us now present a brief description of the terminology used by each of the analyzed papers to refer to the different aspects of NFRs. In [3, 31] non-functional properties of web services are classified according to three points of view, namely, *service level*, *system level* and *business level*. In [3] NFRs are denoted as *quality constraints*, which are expressed as logic formulae. In [31] authors classify NFRs into *category*, *sub-category* and *property*. Categories include *business*, *service* and *system*. Possible *sub-categories* are *security*, *value* or *interoperability*. The work also defines a *web service quality model*, which considers non-functional properties.

In [30] the authors use the terms *non-functional attributes*, *composition model entity* and *model entity* to classify different concepts related to NFRs. The notion of non-functional attribute is used to describe NFRs of the abstract process model. In the lower level, the composition is annotated with non-functional attributes.

D'Ambrogio [9] uses the term *quality category* to group similar *quality characteristics*. *Quality dimensions* are used to quantify an individual characteristic. For instance, the quality category *performance* groups characteristics such as *latency* and *throughput*. The development process is based on MDA and the authors also present a WSDL extension for describing the QoS of web services. A catalog of *QoS characteristics* is provided for the web service domain, including properties as *availability*, *reliability* and *access control*.

Schmeling et al. [25] present an approach and a toolset for specifying and implementing web service compositions with support to several NFRs. The term *non-functional concern* (NFC) is used to denote NFRs. *Non-functional concern* is a general term used to describe non-functional requirements. For instance, *security*, *reliability*, *transactional behavior* are non-functional requirements. A *non-functional action* represents some behavior that implements *non-functional attributes*. An example of *non-functional action* is *encryption*, which provides the implementation of the *non-functional attribute confidentiality*. Non-functional actions related to a common concern are grouped into *non-functional activities*.

Pastrana et al. [23] use the term *contract* to describe non-functional requirements. *Contracts* may have pre-conditions, post-conditions and invariants. Each contract defines *assertions* associated with *quality properties*. Each service may have as many associated *contracts* as needed.

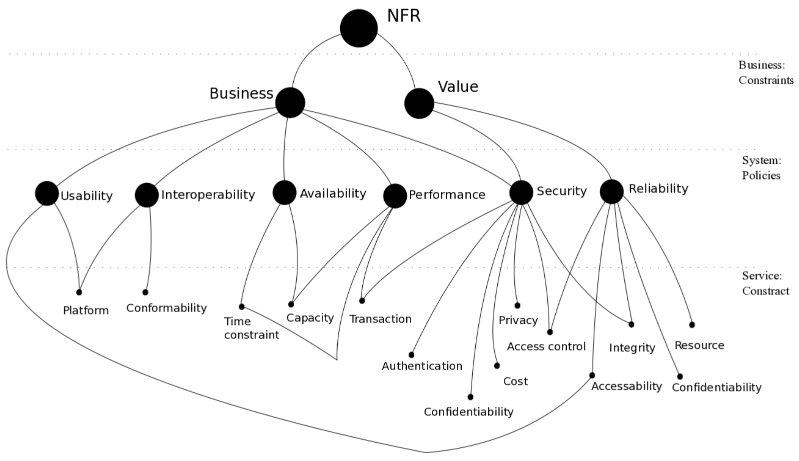

Figure 2. Relationship between NFR Concepts.

Reference	RQ_5:***WS model – Business services***	RQ_6:***Service type***	RQ_7: ***Year***
Babamir et al. [3]	no – yes	composition	2010
Yeom et al. [31]	yes – yes	single	2006
Xiao et al. [30]	no – no	composition	2008
D'Ambrogio [9]	yes – no	composition	2006
Chollet et al. [7]	yes – yes	composition	2009
Schmeling et al. [25]	no – no	composition	2011
Thißen et al. [26]	no – yes	composition	2006
Zhang et al. [32]	no – no	single	2005
Basin et al. [4]	yes – no	single / composition	2006
Ceri et al. [6]	no – no	single	2007
Fabra et al. [11]	yes – yes	composition	2011
Modica et al. [19]	no – no	composition	2009
Ovaska et al. [21]	yes – no	single	2010
Agarwa et al. [1]	yes – no	single / composition	2009
Jeong et al. [13]	no – no	composition	2009
Pastrana et al. [23]	yes – no	composition	2011
Diamadopoulou et al. [10]	no – no	composition	2008
Gutierrez et al. [12]	no – no	single / composition	2010
Mohanty et al. [20]	no – no	single	2010
Karunamurthy et al. [15]	no – no	composition	2012
Liu et al. [18]	no – no	single / composition	2012
Tran et al. [27]	yes – yes	composition	2012
Wang et al. [29]	no – no	composition	2012
Li et al. [17]	no – no	composition	2013
Kamalabad et al. [14]	yes – no	composition	2013
Rumpel et al. [24]	yes – no	composition	2012

Table 3. Research question results - RQ_5, RQ_6, RQ_7.

Chollet et al. [7] associate (non-functional) *quality properties* to (functional) activities. The authors present a security meta-model for web service composition. The NFRs considered are *authentication*, *integrity* and *confidentiality*. Each NFR is associated with a service activity.

Ceri et al.[6] uses the notions of *policy*, *rule*, *condition* and *action model* to specify NFRs. Agarwal et al. [1] associate *service policies* to services. Each service may also have *properties*, such as *security* and *reliability*. Ovaska et al. [21] use the terms *quality attribute*, *category*, *conceptual layer* and *importance* to organize and classify NFRs. Other authors do not define specific terms to refer to NFRs. They use terms such as *attribute* [32, 4, 13], *property* [11], *factor* [20, 12], *characteristic* [10], *quality level* [19], and *value* [26, 4].

Despite of the different notations found in the literature for classifying NFRs, some non-functional requirements are frequently considered, such as *security*, *performance*, *reliability*, *usability*, and *availability*. However, distinct hierarchies and models are proposed for NFRs, according to different points of view. We have identified a number of approaches [9, 7, 25, 4, 11, 21] that use MDD (Model Driven Development) for designing and developing applications.

Fabra *et al.* [11] also describes the importance of MDD for service-oriented applications. This work presents a complete development methodology, although this methodology is not centered on NFRs. The authors in [26, 32] use formal methods to define a service-based development process that takes NFRs into account. In [1, 23] ontologies are used to define and model NFRs, whereas in [30, 12] Business Process Modeling (BPM) is used for system specification, including NFRs. The majority of the authors concentrate on the modeling of service compositions, although a significant number of approaches is focused on the definition of NFR models.

In the method defined in [30], tasks in the process model can be annotated with *non-functional attributes* (NFAs). NFAs are defined apart and are concerned with data items or tasks. NFAs for data considers *value* and *range*, whereas NFAs for tasks include *cost*, *time*, *resources* and *expressions*.

The proposal in [26] presents steps for selecting services by taking QoS information into account. The proposed steps are: *(i)* identification of relevant QoS information; *(ii)* identification of basic composition patterns and QoS aggregation rules for these patterns; and *(iii)* definition of a selection mechanism of services. The QoS properties considered are *performance*, *cost*, *reliability* and *availability*.

Karunamurthy et al. [15] use the term *non-function parameters* to define NFRs, such as *cost*, *response time*, *availability*, *security*, *reliability* and *reputation*. The *Non-Functional Specification Language* (NFSL) is proposed as a domain specific language (DSL) to express *non-function parameters*.

Liu et al. [18] use the term *QoS parameter* to describe non-functional requirements such as *cost*, *execution duration*, *accuracy*, *security*, *integrity*, *availability* and *reliability*. In the same way, Tran et al. [27] use the term *QoS policies* to classify similar non-functional requirements.

Li et al. [17] associate *dimensions* to *QoS parameters* to classify NFRs. For instance, the *time* dimension is associated to the *execution time* and *communication time* parameters; the *spatial* dimension is associated to the *storage capacity* and *message length* parameters; the *reliability* dimension is associated to the *availability* and *reliability* parameters and the *cost* dimension is associated to the *service cost* parameter. Rumpel et al. [24] associate *quality requirements* to *quality properties*. Quality requirements are intended to be specified as constraints.

4. Classification of NFR

Our systematic review is useful to identify those aspects of the NFR treatment in web service development that are consensual among authors. These aspects include terminology to denote concepts as well as the relationships among them.

Most works agree on distinguishing three points of view, namely the point of view of the organization (or Business view), of the individual service providers (or Service view) and of the composition designer (or System view). The Business view is concerned with the business logic (*i.e.*, an abstract level of tasks, defined by the guidelines and constraints imposed by the organization). Service and System views are concerned with the implementation of the software solution: The Service level is concerned with the building blocks of the application. It may use web services provided by third party sources. The System level is concerned with the coordination of services, to implement the business logic.

Table 4 shows the NFR classification we propose, in accordance to these points of view. The second column of the table presents the term we suggest for NFRs at each level. The last column contains the most common terms found in the literature for specific classes of NFRs at each level.

Viewpoint	***Term***	***Concepts***
Business View	Constraint	Business Constraint, Value Constraint
System View	Policy	Security, Performance, Interoperability, Scalability, Reliability, Usability, Transactional Behaviour, Availability
Service View	Contract	Integrity, Transaction, Accessibility, Encryption, Cost, Time Constraint, Encryption, Platform, Privacy, Authentication, Resource, Capacity, Privacy, Confidentiability

Table 4. NFR Classification.

The relationship between concepts of the three levels is depicted in Figure 2. The *Contracts* bound to services should be devised to implement *Policies* (at the System level). These policies are used to guarantee business *Constraints*.

Another classification of NFRs can be found in [31], which does not considers NFRs over data (but just over functions and service performance). We consider that it is important to classify the requirements of business and data (value) restrictions, since data processing is an essential part of the web service execution.

In the next subsection we present a NFR-centred meta-model for web service applications.

4.1. NFR Meta-Model

The model presented in Figure 3 shows the relationship between those concepts we consider important for quality requirements used in service-based development.

In this model, one `Requirement` (functional or non-functional) can be represented by one or more use cases. Each use case represents a `Service Activity`. Each use case has *business* or *value* constraints. *Business constraints* are restriction on functions and how they may be implemented. The *value constraints* are restrictions on the service interface, which the desired values for input and output data. Each constraint is associated with NFAs.

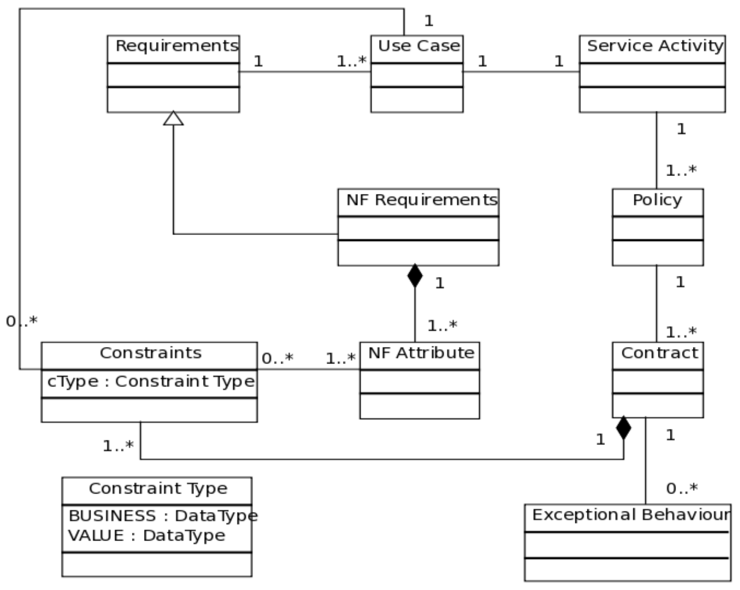

Figure 3. NFR Model.

A Contract is a set of constraints for the same function. For example, a contract for the payment operation. The constraints for payment are: (i) the value amount should not be less than 10 euros and (ii) the user should always receive a purchase confirmation by phone message. This restrictions are grouped into a single contract for payment verification.

An Exceptional Behavior happens when a contract is not respected. When this happens a new function is called or the process is stopped. For example, if the bank does not authorize the payment, the system offers alternative forms of payment such as PayPal.

Finally a Policy groups similar contracts. For example, security contracts are grouped into a security policy and performance contracts are grouped into a performance policy.

5. Conclusions

Notice that *security* and *performance* are the most frequently considered NFRs. *Reliability* is also present in most proposals.

This paper presented a systematic review of NFRs associated with web services. We grouped the NFRs according to their characteristics and analyzed them in each context of application. We identified the main NFR concepts associated with service-based development. We also presented a synthesis of the most common concepts found in the literature. We suggest a characterization of NFR-related concepts and a classification by considering three points of view: *business*, *services* and *system*.

Our analysis and classification can help to improve the development of service-based applications, by focusing on the guarantee of quality requirements.

We believe that the specification of NFRs from the early stages of the software process can improve the quality of the solution.

References

[1] Sudhir Agarwal, Steffen Lamparter, and Rudi Studer. Making web services tradable: A policy-based approach for specifying preferences on web service properties. *J. Web Sem.*, 7(1):11–20, 2009.

[2] Assaf Arkin, Sid Askary, Scott Fordin, Wolfgang Jekeli, Kohsuke Kawaguchi, David Orchard, Stefano Pogliani, Karsten Riemer, Susan Struble, Pal Takacsi-Nagy, Ivana Trickovic, and Sinisa Zimek. Web service choreography interface. Technical report, World Wide Web Consortium, 2002.

[3] S.M. Babamir, S. Karimi, and M.R. Shishechi. A broker-based architecture for quality-driven web services composition. In *Computational Intelligence and Software Engineering (CiSE), 2010 International Conference on*, pages 1 –4, dec. 2010.

[4] David A. Basin, Jürgen Doser, and Torsten Lodderstedt. Model driven security: From uml models to access control infrastructures. *ACM Trans. Softw. Eng. Methodol.*, 15(1):39–91, 2006.

[5] F. Cabrera, G. Copeland, T Freund, J. Klein, D. Langworthy, D. Orchard, J. Shewchuk, and T. Storey. Web services coordination (ws-coordination). Technical specification, BEA Systems, International Business Machines Corporation, Microsoft Corporation, Inc, November 2004.

[6] Stefano Ceri, Florian Daniel, Maristella Matera, and Federico Michele Facca. Model-driven development of context-aware web applications. *ACM Trans. Internet Techn.*, 7(1), 2007.

[7] Stéphanie Chollet and Philippe Lalanda. An extensible abstract service orchestration framework. In *ICWS*, pages 831–838, 2009.

[8] W. Cox, F. Cabrera, G. Copeland, T. Freund, J. Klein, T. Storey, and S. Thatte. Web services transaction (ws-transaction). Technical specification, BEA Systems, International Business Machines Corporation, Microsoft Corporation, Inc, November 2004.

[9] Andrea D'Ambrogio. A model-driven wsdl extension for describing the qos ofweb services. In *ICWS*, pages 789–796, 2006.

[10] Vassiliki Diamadopoulou, Christos Makris, Yannis Panagis, and Evangelos Sakkopoulos. Techniques to support web service selection and consumption with qos characteristics. *J. Network and Computer Applications*, 31(2):108–130, 2008.

[11] J. Fabra, V. De Castro, P. Álvarez, and E. Marcos. Automatic execution of business process models: Exploiting the benefits of model-driven engineering approaches. *Journal of Systems and Software*, (0):–, 2011.

[12] Carlos Gutiérrez, David G. Rosado, and Eduardo Fernández-Medina. The practical application of a process for eliciting and designing security in web service systems. In *JISBD*, pages 143–143, 2010.

[13] Buhwan Jeong, Hyunbo Cho, and Choonghyun Lee. On the functional quality of service (fqos) to discover and compose interoperable web services. *Expert Syst. Appl.*, 36(3):5411–5418, 2009.

[14] M.A. Kamalabad, F. Mardukhi, N. Nematbakhsh, and M.N. Dehkordi. Evaluating the similarity of web service policies using flexible parameter matching. In *Measurement, Information and Control (MIC), 2012 International Conference on*, volume 2, pages 1000–1005, 2012.

[15] R. Karunamurthy, F. Khendek, and R. H. Glitho. A novel architecture for web service composition. *Journal of Network and Computer Applications*, 35(2):787 – 802, 2012. ¡ce:title¿Simulation and Testbeds¡/ce:title¿.

[16] Barbara A. Kitchenham, Hiyam Al-Kilidar, Muhammad Ali Babar, Mike Berry, Karl Cox, Jacky Keung, Felicia Kurniawati, Mark Staples, He Zhang, and Liming Zhu. Evaluating guidelines for reporting empirical software engineering studies. *Empirical Software Engineering*, 13(1):97–121, 2008.

[17] L. Li, M. Rong, and G. Zhang. A web service composition selection approach based on multi-dimension qos. In *Computer Science Education (ICCSE), 2013 8th International Conference on*, pages 1463–1468, 2013.

[18] M. Liu, M. Wang, W. Shen, N. Luo, and J. Yan. A quality of service (qos)-aware execution plan selection approach for a service composition process. *Future Generation Computer Systems*, 28(7):1080 – 1089, 2012. ¡ce:title¿Special section: Quality of Service in Grid and Cloud Computing¡/ce:title¿.

[19] Giuseppe Di Modica, Orazio Tomarchio, and Lorenzo Vita. Dynamic slas management in service oriented environments. *Journal of Systems and Software*, 82(5):759–771, 2009.

[20] Ramakanta Mohanty, V. Ravi, and Manas Ranjan Patra. Web-services classification using intelligent techniques. *Expert Syst. Appl.*, 37(7):5484–5490, 2010.

[21] Eila Ovaska, Antti Evesti, Katja Henttonen, Marko Palviainen, and Pekka Aho. Knowledge based quality-driven architecture design and evaluation. *Information & Software Technology*, 52(6):577–601, 2010.

[22] M. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. Service-Oriented Computing: State of the Art and Research Challenges. *IEEE Computer*, 40(11), 2007.

[23] Jose Luis Pastrana, Ernesto Pimentel, and Miguel Katrib. Qos-enabled and self-adaptive connectors for web services composition and coordination. *Computer Languages, Systems & Structures*, 37(1):2–23, 2011.

[24] A. Rumpel and K. Meissner. Requirements-driven quality modeling and evaluation in web mashups. In *Quality of Information and Communications Technology (QUATIC), 2012 Eighth International Conference on the*, pages 319–322, 2012.

[25] Benjamin Schmeling, Anis Charfi, and Mira Mezini. Composing non-functional concerns in composite web services. In *ICWS*, pages 331–338, 2011.

[26] Dirk Thißen and Pimjai Wesnarat. Considering qos aspects in web service composition. In *ISCC*, pages 371–377, 2006.

[27] H. Tran, U. Zdun, T. Holmes, E. Oberortner, E. Mulo, and S. Dustdar. Compliance in service-oriented architectures: A model-driven and view-based approach. *Information and Software Technology*, 54(6):531 – 552, 2012.

[28] Anargyros Tsadimas, Mara Nikolaidou, and Dimosthenis Anagnostopoulos. Extending sysml to explore non-functional requirements: the case of information system design. In *SAC*, pages 1057–1062, 2012.

[29] H. Wang, P. Ma, and X. Zhou. A quantitative and qualitative approach for nfp-aware web service composition. In *Services Computing (SCC), 2012 IEEE Ninth International Conference on*, pages 202–209, 2012.

[30] Hua Xiao, Brian Chan, Ying Zou, Jay W. Benayon, Bill O'Farrell, Elena Litani, and Jen Hawkins. A framework for verifying sla compliance in composed services. In *ICWS*, pages 457–464, 2008.

[31] Gwyduk Yeom, Taewoong Yun, and Dugki Min. Qos model and testing mechanism for quality-driven web services selection. In *Proceedings of the The Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, and the Second International Workshop on Collaborative Computing, Integration, and Assurance (SEUS-WCCIA'06)*, pages 199–204, Washington, DC, USA, 2006. IEEE Computer Society.

[32] Xinwen Zhang, Francesco Parisi-Presicce, Ravi S. Sandhu, and Jaehong Park. Formal model and policy specification of usage control. *ACM Trans. Inf. Syst. Secur.*, 8(4):351–387, 2005.