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

Plácido A. Souza Neto
Federal Institute of Rio Grande do Norte (IFRN),
Federal University of Rio Grande do Norte (UFRN)
Natal-RN, Brazil
placido.neto@ifrn.edu.br

Genoveva Vargar-Solar
French Council of Scientific Research
LIG-LAFMIA
Saint Martin d'Hères, France
Genoveva.Vargas-Solar@imag.fr

Valeria de Castro
Universidad Rey Juan Carlos
Móstoles, Spain
Valeria.deCastro@urjc.es

Martin A. Musicante
Federal University of Rio Grande do Norte (UFRN)
Natal-RN, Brazil
mam@dimap.ufrn.br

ABSTRACT

This paper presents a systematic literature review of non-functional requirements (NFRs) for service-based applications. We propose the classification of non-functional requirements into 3 levels: *business*, *service* and *system*. During the development process, the NFRs can be refined from one level to a more concrete level (*i.e.*, from business to service level or from service to system level). Along with the classification, we also propose a specific nomenclature for this type of modelling.

Keywords

Non-functional requirements, Classification, Sevice-based development, Contraints, Contract, Policy

1. INTRODUCTION

Non-functional aspects concerning services and applications semantics are often expressed as requirements and constraints. It is not uncommon to these aspects not not to be fully considered during the development of applications. In many cases, they are considered once the application has been implemented, in order to ensure some level of reliability (e.g., data privacy, exception handling, atomicity, data persistence). This situation leads to service-based applications that are partially specified and that are thereby partially compliant with their requirements.

In systems and requirements engineering, a non-functional requirement (NFR) specifies criteria about the behaviour of a system. These criteria may be not related to the results produced by the system, but to other conditions of its performance. Non-functional requirements are often called qualities of a system. They are also referred as "constraints", "quality attributes", "quality goals", "quality of service requirements" and "non-behavioural requirements" [23]. In the case of service-based applications, non-functional requirements concern the application itself as well as its component services.

As services are independent components, ensuring non-functional properties is a challenge. Different studies [2, 1, 5, 13, 26, 14, 25] try to associate non-functional requirements to services using different approaches. Associating non-functional requirements to services compositions can help to ensure that the resulting application is compliant to the user requirements and also with the characteristics of the services it uses.

Attempts to solve these problems have led to the development of a great deal of work in non-functional attributes and non-functional requirements. Although they are used as synonyms in most NFR approaches [16], we will distinguish the concepts of non-functional requirements and non-functional attributes. NFRs will be defined as a group of semantically correlated non-functional attributes (NFA). For example, *security* is an non-functional requirement that comprises attributes such as *confidentiality* and *integrity*. A non-functional attribute describes the characteristics of a functional requirement.

In this context, this paper presents *(i)* a systematic review of the treatment given to non-functional requirements for web service based applications, and *(ii)* proposes a classification of the non-functional requirements for these kind of applications.

The analysis will be conducted in the form of a systematic review [15]. The result of this review can be useful to summarize the existing treatment of NFRs and to provide a background to new research activities.

The paper is organized as follow: Section 2 presents a systematic literature review of non-functional requirements for service-based application. In Section 3 we propose a classification of the NFRs considering the analysis of works made in the systematic review section. Section 4 concludes the paper.

2. NON-FUNCTIONAL REQUIREMENTS FOR SERVICE-BASED APPLICATIONS

This section presents a systematic review of the non-functional requirements and properties used in the context of service-oriented development. The purpose of our analysis is:

- to identify the concepts, properties and notations used in the service-based systems development;
- to find if there is any pattern or relationship between non-functional requirement concepts in different modelling levels.

For each phase of the development, the requirements and non-functional requirements are refined. As a result of this process, the granularity of the concepts described by the requirement becomes thinner and more precise.

We propose 7 research questions (RQ_1 to RQ_7) to guide our analysis about non-functional requirements. For each question we define a set of possible answers in order to guide the analysis. These possible answers are defined from our knowledge on each topic. The questions are closely related to service-oriented development with a NFR focus. They are:

- RQ_1: How are NFR modelled by existing methodologies for developing reliable Web services?
  - *Answer is specific to each proposal*
- RQ_2: Which are the NFR that are more frequently by methodologies developing web services?
  - *Possible answers: security / availability / portability / . . . / reliability / performance*
- RQ_3: What is the software development approach used in the paper?
  - Possible answers: *Model driven approach (*MDD) / Ontology (*Ont) / Formal method (*FM) / Artificial intelligence (*AI) / Business Process Modeling (*BP) Traditional (*TDT)*
- RQ_4: What is the discipline (application domain) of the "*non-functional requirements*" / "*non-functional properties*" used in the work?
  - Possible answers: *Software architecture / QoS model / Language definition / Methodology / etc*
- RQ_5: Does the paper proposes a (meta)model describing and analyzing NFR? Is there any relationship between the non-functional requirements (meta)model proposed and business services?
  - Possible answers: *yes / no – yes / no*
- RQ_6: Do the non-functional aspects are treated in an independent way?
  - Possible answers: *single / composition*
- RQ_7: Which is the publication year of the paper?
  - Possible answers: *Year of publication*

The research questions identify characteristics of the service-oriented application development, especially the modelling of non-functional requirements/properties, how they are addressed and if there is any classification.

We used the approach described in [15] for searching, collecting and selecting works related with non-functional requirements for developing service-based applications. Based on these data we analyzed concepts used in each work for the description of NFRs and the difference among them.

We selected the sources proposed by [15] for searching primary studies. These sources contain the works published in journals, conferences and workshops which are of recognized quality within the research community. The search engines are: *(i) IEEE Computer; (ii) ACM Digital Library;* and *(iii) Science Direct.* In the selected sources, we used the following search query criteria:

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

After define the sources' selection, we apply the process used to identify those works that provided direct evidence with regard to the research questions. Deciding for the inclusion and exclusion criteria for filtering the corpus works selection, we selected those related to non-functional requirements/properties, and quality for web service based applications. Initially, the selection criteria were interpreted liberally and clear exclusions were only made with regard to title, abstract and introduction.

Based on the guidelines mentioned in [15], we established a multi-step process made up of three steps with different selection criteria:

- Step 1 - the search string must be run on the selected search engine. An initial set of studies was obtained by filtering of title, abstract, and if necessary, introduction. All the studies were selected according to the inclusion and exclusion criteria. Studies which were not clearly related to any aspect of the research questions were excluded.
- Step 2, the exclusion criteria were based on the following practical issues: non-English papers, non- International Conference papers and non-International Workshop papers. Specifically in the case of ACM library, we considered only the transaction journal works.
- Step 3, the papers selection process was based on detailed research questions.

The information for each step was collected considering the 3 searchers and the the query presented previously. the results of each step were: *(i)* for each source a list of all the studies that fulfilled the query; *(ii)* a list of studies for each source which contained all the works that did not fulfill the second stage inclusion criteria; and *(iii)* the last step produced a list of works for each source which contained all the studies that fulfilled the second step.

The notation used to refer non-functional requirements will be highlighted in *italic*, as well as the set of values that can be associated with each concept. For example, a notation used by a work can be *quality property* or *non-functional* *concern*, and the values associated with its notation are *security*, *reliability*, *transaction*, etc. Thus, considering the research question, we analyze a set of works in order to identify any pattern or divergence between the analyzed literature.

Babamir et al.[2] ranks services *quality properties* in three categories (business level, service level, system level). *Quality properties* are associated with *quality constraints*, that are defined as assertions or propositional logic formulas. *Non-functional attributes, composition model entity* and *model entity* are the notations used by Xiao et al. [26] for classifying the different concepts for non-functional requirements modeling. Non-functional attributes (NFAs) describe the non-function aspects in the abstract process models. The framework to model NFRs proposes to annotate composition models with NFAs.

D'Ambrogio [9] uses the term *quality characteristics* to group similar characteristics into *quality categories.* Each *quality characteristic* is quantified by *quality dimensions. Quality characteristic* is a quantified QoS aspect, for example *latency, throughput, reliability, availability*, etc. *Quality characteristics* of a common subject are grouped into abstract *quality categories*, for example *performance* (for latency and throughput characteristics) and *dependability* (for reliability and availability characteristics).

The terms *category, sub-category* and *property* are used by Yeom et al.[27] to classify non-functional requirements. *Business*, *service* and *system* are the possible values to be associated with the proposed terms, and a *sub-category* can be *security, value, interoperability*, etc. The work in [27] defines a *web services quality model*, which considers non-functional properties in several aspects. In this model, web services qualities are classified in categories and sub-categories. Chollet et al.[5] uses only two terms to classify and relate quality properties with services, they are: *activity* and *quality property.* Each activity represents a functional property that can be divided in sub-activities, depending on its granularity. For non-functional requirements, the work in [5] describes the possibility of creating different meta-models for each *quality property*, and then relate them with activities.

Schmeling et al.[21] uses the *non-functional concerns (NFC)* notation to describe NFRs. This term encompasses two aspects: the specification of NFCs and their realization. [21] defines that a *functional concern (FC)* is a consistent piece of functionality that is part of a software system. *Non-functional concern (NFC)* is a general term describing a matter of interest or importance that does correspond to a non-functional requirement of a system, for example, *security, reliability, transactional behavior*, etc. A *non-functional action* represents some behavior that results in a *non-functional attribute.* An example of *non-functional action* is *encryption*, which realizes the *non-functional attribute*, *confidentiality.* A *non-functional activity* is also used as a term, which means to encapsulate the control flow of *non-functional action* that apply to the same subject. The term is used in analogy to activity and action in UML2.

Ceri et al.[4] uses the terms *policy, rule, condition* and *action model* to specify NFRs, and in a similar way, Agarwal et al.[1] also uses the concept of *service policy* associated with the concept of service. Each service is also associated with a *service property*, which may have a specific value (*security, reliability*, and so on). Each service is also associated with a *unit function*, that represents one or more requirements.

Ovaska et al.[19] uses *quality attribute, category, conceptual layer* and *importance* to organize and classify the NFRs, Pastrana et al.[20] uses the term *contract* to describe non-functional requirements. In a *contract* it is possible to define pre-conditions, post-conditions and invariants. A *web service* can have many *contracts*, that defines many *assertions* and are associated with *quality properties.*

Some related papers do not have any nomenclature to classify non-functional requirements. However, some of them uses the *attribute* notation [28, 3, 14], other uses *properties* [11], other *factors* [18, 13], *characteristics* [10], *quality level* [17] or *values* [24, 3].

2.1 Analysis

Although there exist many different types of notation used for classify non-functional requirements, in general, the values associated to them are the same, *i.e. security, performance, reliability, usability, availability*, etc. What distinguishes the different approaches is the adoption of different NFR hierarchies in order to prioritize or classify the quality requirements. Another interesting factor is that the each work uses different approaches to model these requirements. Thus, there are different kinds of notations for non-functional requirements.

A number of approaches use [9, 5, 21, 3, 11, 19] MDD (Model Driven Development) for designing and developing systems. Fabra *et al.* [11] presents a complete methodology that does not mainly focus on non-functional requirements. Fabra *et al.* [11] also describes the importance of the use of MDD in the development of service-oriented applications. [24, 28] use formal methods to define a development process based on NFR for web services, [1, 20] use ontologies for the definitions and modeling of non-functional requirements and [26, 13] use Business Process Modeling (BPM) for system specification, including NFR. Most works focus on composition service modeling while others define requirements models to represent the properties used.

In [2, 27] non-functional properties for web services are classified into three categories such as *service level view, system level view* and *business level view.* In the *business* level the quality properties are: *service charge, compensation rate, penalty rate* and *reputation*; at the *service* level the quality properties are: *performance* and *stability*; and at the *system* level the quality properties are classified into: *manageability, interoperability, business processing* and *security.*

In the method defined in [26], each task in the process model is annotated with the NFAs (*non-functional attributes*). During the design phase, the service composition and the definition of NFAs are separated. Then, each task in the process model is annotated with the corresponding NFA. The attributes are related with tasks or data item. For data, the NFAs are: *value* and *range*; and for tasks the NFAs are: *cost, time, resources* and *expression.*

D'Ambrogio [9] presents a WSDL extension for describing the QoS of web services. The process is based on MDA. The work presents a catalog of *QoS characteristics* for the web service domain and the Q-WSDL (Quality WSDL) meta-model for modeling QoS properties in web services. The properties presented are: *availability, reliability* and *access control.* [5] presents a security meta-model for web service composition. The non-functional requirements considered concern *authentication, integrity* and *confidentiality.* Each

Reference	*NFR concepts*	*NFR values*	*Approach*	*Domain / Scope*
Babamir et al. [2]	property / category / constraint	responsiveness / availability performance / sla properties	TDT	Software architecture
Yeom et al. [27]	category / sub-category / property	business value / performance / stability /manageability / security / business processing interoperability	TDT	QoS model
Xiao et al. [26]	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. [5]	activity / NF attribute	security	MDD	Orchestration Framework
Schmeling et al. [21]	NF concern / NF attribute / NF action / NF activity	security	MDD	Web service composition process
Thißen et al. [24]	NF value	performance / reliability cost / availability	FM	Software architecture
Zhang et al. [28]	attribute / predicate	security	FM	Access control
Basin et al. [3]	attribute	security	MDD	System architecture
Ceri et al. [4]	police / rule condition / action	n.a.	TDT	Context-aware applications
Fabra et al. [11]	property	storage / processing (*case study)	MDD	Web service methodology
Modica et al. [17]	quality level	sla properties	TDT	Service oriented architecture
Ovaska et al. [19]	attribute category	security / reliability	MDD	Model development
Agarwa et al. [1]	property / policy / function	*not explicitly defined*	Ont	Policy language
Jeong et al. [14]	NF attribute	operation cost / performance / availability / accessibility / security / interoperability / usability / user satisfaction	AI	Service oriented architecture
Pastrana et al. [20]	NF property / contract / assertion / NF behaviour	performance / reliability / scalability / capacity / robustness / precision / accessibility / availability / interoperability / security	Ont	Web service methodology
Diamadopoulou et al. [10]	NF characteristic	user' subjective perception	TDT	Web service selection
Gutierrez et al. [13]	NF factor NF sub-factor	Security	BP	Web service development process
Mohanty et al. [18]	NF attribute or NF factor	reliability / performance / integrity / usability response time / documentation	AI	Artificial intelligent / Web services classification

Table 1: Research question results - RQ_1, RQ_2, RQ_3, RQ_4.

property is related with a service activity. In [21], authors present an approach and a toolset for specifying and implementing the composition of several non-functional properties of Web services. The non-functional attributes described in [21] are: *confidentiality, integrity (security concern)* and *response time (performance concern)*.

The work presented in [24] describes steps to design a selection mechanism of services identified as candidates for participating in a composition, considering their quality properties. The 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 service candidates. As QoS properties considered in [24] are: *performance, cost, reliability* and *availability.*

Among the properties presented we highlight *security* and *performance*. Users usually need to access data securely and quickly. Most studies have both properties as the most important non-functional requirements. *Reliability* is also an important non-functional requirement presented in some works and required by end users.

We can now relate the research questions presented in section 2, with the works described above. Tables 1 and 2 show some results.

The vocabulary used for naming and characterizing NFR is not stable, 5 works out 360 (1.6%). 19 of the total of works chosen (26.31%) propose a classification of non-functional requirements.

Reference	*Service model – Business services*	*Service type*	*Year of publication*
Babamir et al. [2]	no – yes	composition	2010
Yeom et al. [27]	yes – yes	single	2006
Xiao et al. [26]	no – no	composition	2008
D'Ambrogio [9]	yes – no	composition	2006
Chollet et al. [5]	yes – yes	composition	2009
Schmeling et al. [21]	no – no	composition	2011
Thißen et al. [24]	no – yes	composition	2006
Zhang et al. [28]	no – no	single	2005
Basin et al. [3]	yes – no	single / composition	2006
Ceri et al. [4]	no – no	single	2007
Fabra et al. [11]	yes – yes	composition	2011
Modica et al. [17]	no – no	composition	2009
Ovaska et al. [19]	yes – no	single	2010
Agarwa et al. [1]	yes – no	single / composition	2009
Jeong et al. [14]	no – no	composition	2009
Pastrana et al. [20]	yes – no	composition	2011
Diamadopoulou et al. [10]	no – no	composition	2008
Gutierrez et al. [13]	no – no	single / composition	2010
Mohanty et al. [18]	no – no	single	2010

Table 2: Research question results - RQ_5, RQ_6, RQ_7.

There are other works [7, 8, 6, 12, 22] that consider and propose non-functional requirements classifications. A number of software attributes are defined as requirements in [22, 7]. According to [7], the most common non-functional properties are: *performance*; interface; operational; resource; verification; acceptance; *maintainability*; documentation; *security*; *portability*; quality; *reliability*; *usability*; and *safety.* We highlight those that are used in most analyzed classifications. [7] proposes an NFR specification and a template for specifying requirements, considering both, functional and non-functional properties.

The NFRs are classified in 4 main concepts by [8]. The classification uses *quality properties*. These properties are: *performance; security; cost;* and *usability*. Pastrana [20] describes an ontology based methodology and uses a NFR classification based in some properties, most already described in other works. However that work is the only one to use *scalability, capacity* and *precision* properties. Considering web services development, these properties are not very frequently used, however, considering data processing, these properties can be important in cases of large data processing requests through services.

Sommerville [22] classifies requirements, either functional or non-functional, in three main blocks: *process, product* and *external*. [22] also defines a sub-classification considering the system domain. Software requirements are classified as: *usability; reliability; safety; efficiency; performance;* and *capacity*.

Some requirements may determine the system design, for example, *(i)* Keep certain functions in separate modules; *(ii)* check data integrity for critical variables; and *(iii)* permit only limited communication (requester / provider), are examples of restrictive requirements [7]. These examples are related with some NFR concepts as: *security, reliability* and *availability*.

In the service-based development there is a clear difference between the business, service and system levels. The quality requirements are treated differently in each of these levels. Despite the different nomenclatures they can be described in general as *non-functional concern/requirement* and *non-functional attribute*.

Considering [7] and [22], we notice that restrictions are usually related to use cases and functional requirements, and in most cases, a service activity can be represented as an use case. It is in this way that non-functional requirements with web services are related. They don't propose a specific way to design constraints to services, but assuming that a service is modeled as a use case, and it has quality requirements.

At the service level, each service activity is related in some way with the concept of contracts. Contracts can be grouped into policies and their rules. Policies are directly related to concepts like quality, security, performance and availability; contracts are associated with non-functional attributes.

	IEEE	ACM	Science Direct	Total
Total results	65	271 (75 [1])	166	502 (306 [2])
Step one	19	10	20	49
Step one (%)	29.23%	13.33%	12%	16%
Step two	7	3	9	19
Step two (%)	10.76%	4%	5.42%	6.20%

Table 3: Summary of the studies selected at each step.

The extraction of information was based on the research questions, and each work extraction question included the following items: *(i)* where the paper was found; *(ii)* identification of the title and main subjects; *(iii)* summary of the research; *(iv)* inclusion and exclusion criteria; *(v)* objective and result; and *(vi)* subjective results.

Table 3 shows a summary of the studies selected in each stage of the selection procedure for each source. The "Total results" were obtained by running the search string on the selected sources. The next four rows show the results obtained after applying stages one (2 first rows) and two (2 last rows) of the studies selection procedure.

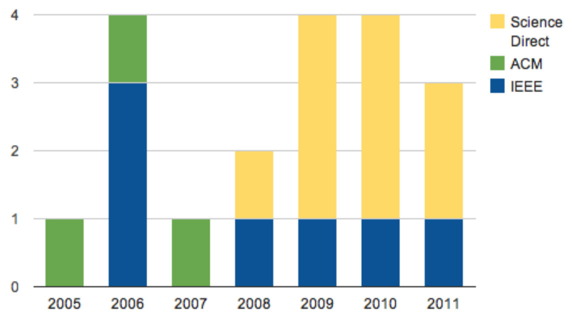

Figure 1: Publications per year.

In the first step, respecting the filters described, the 65 articles collected from IEEE, only 29.23% of them were in accordance with the criteria described early, representing 19 articles. In ACM Library, from 75 works collected, only 13.33% passed in the first stage filter, representing 10 articles. In Science Direct had the lowest percentage, totaling 20 of the 166 articles collected by the query, thus representing 12% of the total. Despite being the lowest relative value, the Science Direct had the largest absolute result, with 20 works in the first step. In the second stage, the percentage dropped further, and the relevant works and with accordance to the criteria have been collected as the final result. The results were respectively, 10.76%, 4% and 5.42% of total from the IEEE, ACM and Science Direct. The highest percentage was among the works from IEEE, while the largest number od results, in absolute terms, was collected from Science Direct. The approaches resulting from this last stage were studied in depth and information concerning the detailed research questions and other fields of the extraction forms was extracted from each eork we selected. 49 works were selected in the first stage, and, only 19 works in the second stage. It represents 6.20% of the total amount of works.

Figure 1 shows the publications per year, from 2005 to 2011. 12 of 19 articles were selected in the systematic review published in 2006, 2009 and 2010, being four in each year and 6 in Science Direct source, 5 in IEEE, and only 1 at the ACM. All nine of Science Direct were published in the last 4 years. Figure also shows that the number of publications that consider classification of NFR once again increased from 2008.

3. CLASSIFICATION OF NON-FUNCTIONAL REQUIREMENTS

According to the previous analysis, we defined the main non-functional requirement concepts that are associated with service-based development. We also present a synthesis of the concepts common to the analyzed approaches used for modelling non-functional requirements of service-based applications. The NFRs have been classified into three levels, *business, services* and *systems*. According to each modeling level, non-functional requirements are being refined from the business level to system level.

One classification of NFRs can be seen in [27]. In that classification, NFRs are not applied on data but just to functions and service performance. We consider that it is important to classify the requirements of business and data (value) restrictions, since data accessible to web services depend on the application contexts.

In sections 3.1 and 3.2 we present which concepts (NFR meta-model) and values (NFR classification) we consider important to a reasonable modeling of non-functional requirements for service applications.

3.1 NFR Meta-Model

The model present in figure 2 shows the relationship between those concepts we consider important for quality requirements used in service-based development.

One `Requirement`, whether functional or non-functional, can be represented by one or more use cases. Each use case represents a `Service Activity`. For example, a paying process can be modeled through the use cases that represent withdrawal and deposit of money in transaction of customer and service provider accounts. A payment process also requires the guarantee of a complete transaction and data security. Thus several use cases are modeled to a single requirement. All these use case can be implemented as services.

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.

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.

3.2 NFR Classification

Table 4 shows the NFR classification we propose, organized as follows (by column): (i) the level of abstraction, (ii) the proper term for this kind of abstraction and (iii) possible values to be used. The rows represent the abstraction level for modeling the NFR. The highest level is the *Business level*, the intermediary one is the *Service level*; and finally the lowe level, called *System level*.

At the business modeling level, non-functional requirements are classified into business restrictions. We have adopted, *business* and *value* constraints to address the most abstract levels of restrictions in business modeling.

The relationship between non-functional requirements and attributes help identify groups of restrictions and contracts to develop specific policies. The level of service shows non-functional requirements, and system level presents a finer granularity, describing the non-functional attributes. Thus, a set of contracts that are related to the same attribute form specific policies for the system. The relationship between the concepts in business, service and system levels are presented in figure 3.

Modeling Level	*Concept / Notation*	*NFR / NFA*
Business Level	Constraint	Business Constraint, Value Constraint
Service Level	Contract	Integrity, Transaction, Accessibility, Encryption, Cost, Time Constraint, Encryption, Platform, Privacy, Authentication, Resource, Capacity, Privacy, Confidentiability
System Level	Policy	Security, Performance, Interoperability, Scalability, Reliability, Usability, Transactional Behaviour, Availability

Table 4: Non-Functional Requirements Classification.

In our work we adopt *security, performance, availability, interoperability, usability* and *reliability* as NFRs (these are system level properties), and we consider as NFAs *access control, time constraint, privacy, accessibility* and so on (which are service level properties).

4. CONCLUSIONS

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 also reviewed some existing NFR classification for service-based development.

To the best of our knowledge, there are no proposals that define a service-oriented approach for the *complete* development of systems, considering non-functional requirements. Although many efforts have been made to support the new technological proposals for the Web such as web services, in general, existing approaches focus their processes on traditional software engineering methodologies, with emphasis to the functional aspects of the application.

Thus, our analysis and classification can help to improve the development of service-based applications which have as their main effort the guarantee of quality requirements and the generation of reliable systems, considering NFRs. We believe that the early modelling of NFRs during the software process helps users and developers to improve the quality of the solution.

5. REFERENCES

[1] S. Agarwal, S. Lamparter, and R. 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] S. Babamir, S. Karimi, and M. 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.

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

[4] S. Ceri, F. Daniel, M. Matera, and F. M. Facca. Model-driven development of context-aware web applications. *ACM Trans. Internet Techn.*, 7(1), 2007.

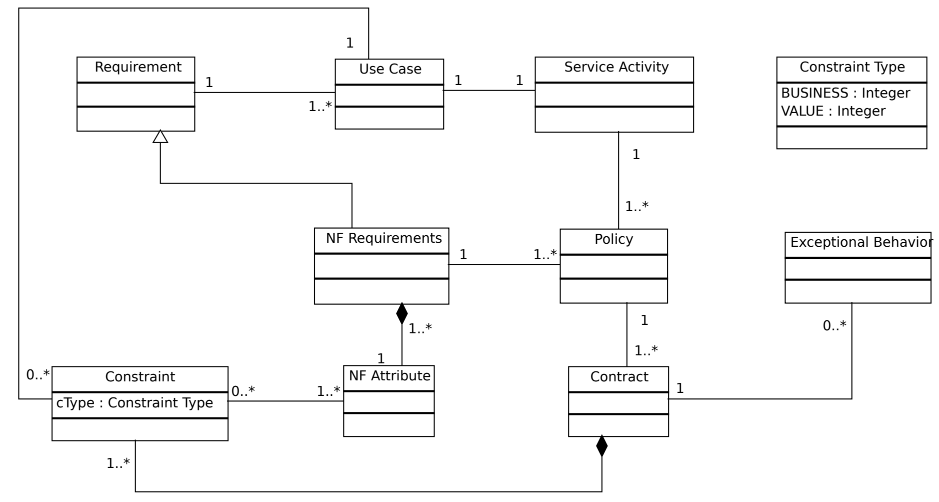

Figure 2: Service-Based Non-Functional Requirement Model.

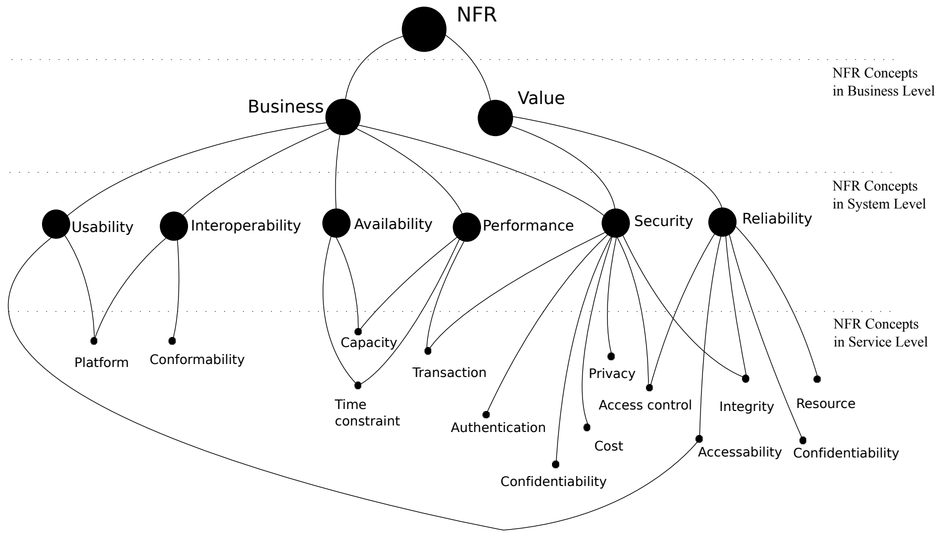

Figure 3: Relationship of the NFR Concepts.

[5] S. Chollet and P. Lalanda. An extensible abstract service orchestration framework. In *ICWS*, pages 831–838, 2009.

[6] J. Cleland-Huang, R. Settimi, X. Zou, and P. Solc. The detection and classification of non-functional requirements with application to early aspects. In *RE*, pages 36–45, 2006.

[7] S. Committee. Ieee recommended practice for software requirements specifications. *Practice*, 1998(October):37.

[8] L. M. Cysneiros, J. C. S. do Prado Leite, and J. de Melo Sabat Neto. A framework for integrating non-functional requirements into conceptual models. *Requir. Eng.*, 6(2):97–115, 2001.

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

[10] V. Diamadopoulou, C. Makris, Y. Panagis, and E. 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. D. Castro, P. ï£¡lvarez, 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] M. Glinz. Rethinking the notion of non-functional requirements. In *in Proceedings of the Third World Congress for Software Quality (3WCSQ'05*, pages 55–64, 2005.

[13] C. Gutiérrez, D. G. Rosado, and E. Fernández-Medina. The practical application of a process for eliciting and designing security in web

service systems. In *JISBD*, pages 143–143, 2010.

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

[15] B. A. Kitchenham, H. Al-Kilidar, M. A. Babar, M. Berry, K. Cox, J. Keung, F. Kurniawati, M. Staples, H. Zhang, and L. Zhu. Evaluating guidelines for reporting empirical software engineering studies. *Empirical Software Engineering*, 13(1):97–121, 2008.

[16] D. Mairiza, D. Zowghi, and N. Nurmuliani. An investigation into the notion of non-functional requirements. In *SAC*, pages 311–317, 2010.

[17] G. D. Modica, O. Tomarchio, and L. Vita. Dynamic slas management in service oriented environments. *Journal of Systems and Software*, 82(5):759–771, 2009.

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

[19] E. Ovaska, A. Evesti, K. Henttonen, M. Palviainen, and P. Aho. Knowledge based quality-driven architecture design and evaluation. *Information & Software Technology*, 52(6):577–601, 2010.

[20] J. L. Pastrana, E. Pimentel, and M. Katrib. Qos-enabled and self-adaptive connectors for web services composition and coordination. *Computer Languages, Systems & Structures*, 37(1):2–23, 2011.

[21] B. Schmeling, A. Charfi, and M. Mezini. Composing non-functional concerns in composite web services. In *ICWS*, pages 331–338, 2011.

[22] I. Sommerville. *Software Engineering 6th Edition*. Addison Wesley, 2008.

[23] A. Stellman and J. Greene. *Applied software project management*. O'Reilly, 2005.

[24] D. Thißen and P. Wesnarat. Considering qos aspects in web service composition. In *ISCC*, pages 371–377, 2006.

[25] A. Tsadimas, M. Nikolaidou, and D. Anagnostopoulos. Extending sysml to explore non-functional requirements: the case of information system design. In *SAC*, pages 1057–1062, 2012.

[26] H. Xiao, B. Chan, Y. Zou, J. W. Benayon, B. O'Farrell, E. Litani, and J. Hawkins. A framework for verifying sla compliance in composed services. In *ICWS*, pages 457–464, 2008.

[27] G. Yeom, T. Yun, and D. 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.

[28] X. Zhang, F. Parisi-Presicce, R. S. Sandhu, and J. Park. Formal model and policy specification of usage control. *ACM Trans. Inf. Syst. Secur.*, 8(4):351–387, 2005.