A New Classification of Non-Functional Requirements for Engineering the Development of Software Services and Service-Oriented Applications

Yousra Odeh
Mohammed Odeh
A NEW CLASSIFICATION OF NON-FUNCTIONAL REQUIREMENTS FOR SERVICE-ORIENTED SOFTWARE ENGINEERING

YOUSRA ODEH ¹ and MOHAMMED ODEH ²

1, 2 in Software Engineering Research Group, Centre for Complex Cooperative Systems, Department of Computer Science and Creative Technologies, Faculty of Environment and Technology, University of the West of England (UWE), Coldharbour Lane, Frenchay Campus, Bristol, BS16 1QY, UK.

Abstract:

The service-oriented model of computing is increasingly becoming the mainstream for developing complex software systems and in particular highly distributed and web-based systems. However, the classification and the specification of non-functional requirements (NFRs) for software services and also for service-oriented systems have not been addressed to the level that NFRs’ classification has been attempted for non service-oriented systems. In this paper, we introduce a new framework for classifying non-functional requirements in relation to engineering software services and service-oriented systems. In addition, this new classification is anticipated to be of significant contribution in facilitating the identification and specification of NFRs for service engineering and service-oriented software engineering.

Keywords: Non-Functional Requirements, SOA, Service-Oriented Systems, NFRs Classification, Service Engineering, and QoS.

1. INTRODUCTION

Service-oriented systems are composed of reusable, dependable and distributed software services. Summerville defined a software service as “A loosely coupled, reusable software component that encapsulate discrete functionality, which may be distributed and programmatically accessed” [19] and he also defined a web service as “a service that is accessed using standard Internet and XML-based protocols” [19].

And, if we consider his definition of non-functional requirement being a constraint on the software product, the software development process, and externally related constraints such as interoperability and adherence to standards and regulations, there appears to be much of knowledge required regarding the requirements specifications for software services not only from the functionality view, but also from the non-functional view. In this regard, we aim in this paper to devise a new classification of non-functional requirements for software services (as part of the notion of service engineering) and also for service-oriented systems - as part of the notion of service-oriented software engineering. Such a classification is aimed at identifying and guiding the specifications of NFRs for software services, for example the key quality attributes for a software service.

In this paper, we first proceed by presenting in Section 2 a brief review of related work regarding service-oriented software engineering and the classification of NFRs. Then, in Section 3, we introduce the new NFR classification for service-oriented software engineering. In Section 4, we conclude the discussion on this new classification with some directions for further research work.

2. RELATED WORK
2.1 SERVICE-ORIENTED COMPUTING AND ARCHITECTURE

Service-Oriented Computing (SOC) is a new paradigm for developing a network of services that are reusable and loosely coupled in order to create flexible agile applications. Erle [10] has defined SOC as “a new generation computing platform that encompasses the service-oriented paradigm and service-oriented architecture with the ultimate goal of creating and assembling one or more service inventory”. These services are both language and platform independent [10]. Papazoglou et al [18] consider SOC as a logical separation of functionality through three main architectural layers: (1) service foundation, (2) service composition and (3) service management and monitoring taking into account semantics, QoS and NFR service properties. They also considered the Service-Oriented Architecture (SOA) as a logical approach to realize SOC [18]. Sommerville [19] regards SOA as a new technology for developing distributed applications, where fundamental components are stand-alone services as shown in Figure 1.

Web services are promising technology as an implementation of SOA that use Internet as the medium for distributed computing and offering interoperability as a major advantage. Web services are implemented using open standards such as WSDL (Web Service Description Language), UDDI (Universal, Description, Discovery and Integration) for allowing service requestors - that could be a user or a service - to browse and find the required service, and SOAP (Simple Object Access Protocol SOAP) as illustrated in Figure 1 [19]. Designing service interface takes into consideration service operations, input, output, exceptions, and message format that are all specified following WSDL. However, WSDL lacks describing NFRs and service semantics [19].

![Figure 1: Service-Oriented Architecture (Adopted from [19]).](image)

Furthermore, business process driven software development entails innovating Service Identification (SI) methods - for example BPAOntoSOA [22] - that rely on business process understanding and analysis taking into account functional requirements and NFRs [10]. Also, composition of services may be required to enact complex business processes (modeled for example using BPMN [17]) as part of a certain organizational business workflow and/or the overall business process architecture.

2.2 A SHORT REVIEW OF NON-FUNCTIONAL REQUIREMENTS CLASSIFICATIONS

Nowadays, customers are not only concerned about functionality embedded in software services but they are also paying attention to the desired quality of developed services, for example reliability, security, and efficiency, which are non-functional requirements.

In his attempt to develop a framework for software quality characteristics, Boehm proposed [4] some form of NFRs classification as depicted in Figure 2. In addition, Boehm proposed quality evaluation metrics to assess adherence to NFRs. For example, low-level quality attributes (in Figure 2) are logically implied as the guarding conditions for such adherence. Also, using this tree, three questions are easily deduced: (i) Can the developed software change its environment? (ii) How adhering this software to quality requirements
To what extent such software can be maintained? In spite of such worthy classification by Boehm, this is still not complete and lacks more reflective quantification techniques of modern software development such as the service-oriented paradigm, let alone recommendations for quality tradeoffs between NFRs. However, this classification may be considered as a landmark for further refinements [4].

Davis [7] considers NFRs as non-behavioral requirements and identified seven qualities in this regard as depicted in the third tier of Figure 2.

Chung et al [6] introduced a process and goal based NFR-framework, where goals are decomposed and refined to sub goals in order to arrive at low-level operationalisations soft goals using the Soft Goal Interdependency Graph (SIG). These low level operationalisations denote additional functional requirements in order to satisfy the high level NFRs via the “Label Propagation Algorithm”. Satisfaction of NFRs is not discrete but can vary in range as “satisfied”, “weakly satisfied”, “unknown”, “conflict”, “weakly denied” or “denied”. Also, this approach illustrates tradeoffs between soft goals. This framework may be considered as the most comprehensive approach for determining the satisfaction and tradeoffs of and between low-level NFRs satisfaction. Burgess et al [5] optimised SIG to develop Soft goal Interdependency Rule set Graph (SIRG) as an automated technique for determining the optimal set of the low-level operationalization's to attain better NFRs’ satisfaction. However, SIRG requires some comprehensive case studies to inform its applicability and tackle shortcomings.

The IEEE Standard and namely “IEEE Std-830-1993” is another notable example of attempting to classify and specify NFRs [14].

In 1997, Gilb classified requirements to functions, qualities, costs and constraints [12]. The last three can be regarded as NFRs. Qualities denote “How well the function will perform” and “any restrictions on the freedom of requirements or design” relates to constraints. Gilb’s classification emerged due to the presence of unwanted or undesirable requirements - specified in the Software Requirement Specification (SRS) document - that are false, unclear, and/or not possible to assess their satisfaction.

Sommerville and Kotonya consider NFRs as “restrictions and constraints among
system services” [15]. This classification may be considered the most comprehensive in terms of coverage of NFRs types under three key categories: product, process, and externally related NFRs. The first category relates to the possible or desired attributes that a system may possess. Any constraints and restrictions on the development process over the system relate to the second category of this classification. Finally, externally related NFRs are concerned with organisational regulations, national or international standards, and interoperability requirements.

Glinz [13] classified NFRs as performance and quality related requirements that could be described using four facets: kind, representation, satisfaction, and role.

Lamsweerde classified NFRs according to Quality of Service (QoS), compliance, architectural constraints and development constraints as shown in Figure 3 [21]. QoS is similar to quality requirements such as security, performance…etc. Compliance implies confirming to standards, organizational regulations and external standards and laws. Architectural constraints include structural constraints in relation to the developed software and its operational constraints. Finally, development constraints are concerned with what governs developing the anticipated software. Examples of these constraints are maintainability, delivery schedules, etc.

2.3 CURRENT WORK ON SERVICE-ORIENTED RELATED NFRS

Relating NFRs to service engineering and service-oriented development has been mainly concerned with identifying QoS attributes, tradeoff between them if needed, and how they affect developing service-oriented applications. However, no classifications attempts appear to have been reported in the literature building on earlier classifications to specialise or arrive at a new classification of NFRs for service engineering and also service-oriented software engineering with the exception of the work of Galster and Bucherer who proposed NFRs classification regarding services and service-oriented systems [11].

O’Brien et al [16] reported on...
determining different quality attributes that affect SOA and discussed ten of these attributes observing related issues with recommended solutions to satisfy quality characteristics. Also, they discussed tradeoffs between quality attributes, for instance: developers may need to be aware of the negative effect of security on performance and interoperability, and how the absence of some quality attributes such as availability could significantly harm the overall operational requirements of service oriented systems. Also, they observed the limitation of current standards [16] in addressing quality attributes to attain the appropriate NFR specification before stepping forward.

A number of standards have been developed for QoS such as WS-Reliable. However, if a service is provided from an outside party then a Service Level Agreement (SLA) is established between the parties in order to attain an agreement on the desired QoS. Despite O’Brien et al worthy attempts in this regard, they did not provide an explicit classification of NFRs in relation to service engineering and service oriented software engineering [16].

Ameller and Franch suggested Service Level Agreement Monitor (SALMon) as a monitoring technique on SLA [2]. This technique monitors services to carry out the appropriate decisions when unexpected actions occur while adopting SOA in order to meet SLA requirements. This work reported on the ISO/IEC 9126 based classification of the desired characteristics in relation to web services as shown in Figure 4 with emphasis on the technical and non-technical characteristics of such services. This quality model was chosen due to its generic features and ability to develop characteristic hierarchies. In this quality model, characteristics such as efficiency, portability, etc are refined and decomposed to sub levels. A further refinement and decomposition is applied in order to arrive at some measurable attributes in support of the SALMon approach, for example response time in relation to efficiency [2].

A taxonomy for NFRs along with service-centric systems is presented by Galster and Bucherer [11]. Their taxonomy is more likely of a checklist in relation to the three categories: process requirements, service
requirements and external requirements. This work denoted the ability to relate the taxonomy to the service level as well as the system level. Moreover, it is suggested to be formally or informally integrated to the requirements engineering process as a template [11].

Since NFRs or QoS attributes demand specific knowledge in the associated application domain and also the associated requirements engineering process, there have been attempts to develop the associated ontology to provide the necessary metadata about the desired QoS attributes. Dobson and Sommerville [9] have demonstrated an attempt at developing QoS ontology, namely the QoSOnt for service-centric systems. Dobson and kotonaya reported in [9] on this ontology specified using OWL-DL, but disregarding domain knowledge.

3. The NEW NFRs
CLASSIFICATION FOR SERVICE ENGINEERING AND SERVICE-ORIENTED SOFTWARE ENGINEERING

This new proposed classification depicted in Figure 5 has been inspired by Kotonya and Sommerville, ISO/IEC 9126, and Van Lam sweerde NFR classifications. In this classification, we address service engineering and service-oriented software engineering as two different high-level categories in driving further sub classifications.

3.1 NFRS FOR SERVICE ENGINEERING

These are defined as constraints on the desired QoS characteristics for a single service, its compliance to requirements and development standards, and SOA architectural constraints. This will, however, lead to having a service as a reusable, and loosely coupled unit with independent operational requirements, but adhering to related organizational, national and international standards and regulations. QoS requirements relate to the desired characteristics of a given service running independently and also while being integrated with other services at run-time. Development requirements are constraints on the service engineering process stages starting with the service identification stage, and then moving to design and service implementation and deployment stages. Compliance requirements relate to standards, laws and rules, organisational regulations, political constraints, etc that the developed service must conform to e.g. WS-Privacy. Architectural constraints, are concerned with structural constraints in relation to service design, for example adhering to Erl’s SOA design principles [10]: services share standardised contracts, abstract underlying logic, and are: loosely coupled, reusable, autonomous, stateless, discoverable and composable.

3.2 NFRS IN RELATION TO SERVICE-ORIENTED SOFTWARE ENGINEERING

These are the constraints on the QoS attributes, standards’s compliance requirements, the service-oriented development process, and the architectural constraints in relation to the desired product developed using reusable services provided by service providers. While service engineering NFRs are limited to a single service, service-oriented software engineering NFRs are concerned with the overall behavior of the service-oriented application developed from the composition of reusable services. For example, the QoS attributes, and compliance to standards and regulation are related to the overall behavior of the service-oriented application. However, the NFRs related to the service-oriented development process and architectural constrains are highly likely to be influenced by the service oriented software engineering process.

For example, if construction of service-oriented applications is achieved through a services’ composition process, then the NFRs for the development process will be concerned with ensuring compliance with requirements for workflow development, discovering services, selection of services, refinement of the modeled workflow, enactment of the refined workflow (in a particular implementation environment), and
testing of both individual services and the overall service-oriented application.

4. CONCLUSION

The new classification introduced in this paper has explicitly addressed non-functional requirements for both service engineering and service-oriented software engineering with the latter being dependent on the former in terms of classifying related NFRs. Rather than starting from scratch, this newly introduced classification benefitted from two well know classifications, namely Sommerville and Lamsweerde, but more specifically for software service oriented development. One corollary of such new classification is that it sets the basis for a shared understanding of NFRs in relation to both of these areas i.e. service engineering and service-oriented engineering. Also, being an NFR classification facilitates developing the semantics behind the identified set of NFRs in each category and sub-categories to the leaf nodes of the hierarchy. The authors are currently working on completing an OWL-DL [3] specification of this classification to serve as the general ontology for the engineering of services and service oriented applications with further reference to domain specific models. Finally, this new classification paves the way for advancing the state of the art in detailing the quantification, where possible, of satisfying NFRs for particular application domains.

REFERENCES:


