16 Jul 2007

Specification of Non-Functional Requirements for Contract Specification in the NGOSS Framework for Quality Management and Product Evaluation

Manooch Amoozdeh

Nektarios Georgalas

Xiaoqing Frank Liu
Missouri University of Science and Technology, fliu@mst.edu

Recommended Citation

M. Amoozdeh et al., "Specification of Non-Functional Requirements for Contract Specification in the NGOSS Framework for Quality Management and Product Evaluation," *Proceedings of the 5th International Workshop on Software Quality (WoSQ'07)*, Institute of Electrical and Electronics Engineers (IEEE), Jul 2007.
The definitive version is available at https://doi.org/10.1109/WOSQ.2007.12

Specification of Non-functional Requirements for Contract Specification in the NGOSS Framework for Quality Management and Product Evaluation

Xiaoqing (Frank) Liu
University of Missouri-Rolla
Computer Science Dept.
fliu@umr.edu

Manooch Azmoodeh
BT GCTO
Adastral Park
Martlesham Heath IP5 3RE
manooch.azmoodeh@bt.com

Nektarios Georgalas
BT GCTO
Adastral Park
Martlesham Heath IP5 3RE
nektarios.georgalas@bt.com

Abstract

The community of Operation Support Systems (OSS) for telecom applications defined a set of fundamental principles, processes, and architectures for developing the Next Generation OSS through the TeleManagement Forum TMF. At the heart of NGOSS lies the notion of a "Contract" which embodies the specification of services offered by an OSS component for quality management and product evaluation. However, TMF does not provide any method (or process) for specification of the non-functional part in the NGOSS contract specification. In this paper, we develop a systematic approach for specifying non-functional requirements of telecom OSS applications for contracts in the NGOSS framework for quality management and evaluation. Specifically, two categories of non-functional specification techniques are explored: qualitative and quantitative. Furthermore, we introduce two quantitative non-functional requirements specification methods: crisp and elastic to expand the capability of the current NGOSS contract specification method since only qualitative non-functional specification is currently available from TMF.

1. Introduction

Software industry and in particular Information and Communication Technology (ICT) Service Providers (SPs) is facing a formidable challenge in the face of immense competition in the marketplace. These challenges include improving quality, continual reduction in cost and time to market, as well as increasing business agility by developing, integrating, deploying and adapting their Operational Support Systems (OSS). OSS of an SP comprise service surround capabilities which span all business processes from service fulfillment (ordering, service provisioning, etc) to service assurance (fault and performance management) to billing for service usage as well as key business functions such as supply/partner chain management.

Currently, SPs typically own ~1000 disparate OSS systems which are developed and built using different middleware and software platforms and often integrated using a plethora of methods (EAI, service bus, integration hub, etc.). Furthermore, for each service/product, bespoke integration of OSS applications is used to provide OSS functions, leading to much duplication of OSS capabilities and adding to the complexity of the overall system architecture. As service/product offerings evolve, the OSS applications need to be (re)configured to react to such changes. All these activities are often error-prone, expensive, manual, and time consuming. To complicate matters further, as SPs also use many legacy and COTS application packages, little 'standards' is used to ease the so called 'integration tax'.

The Telco industry is responding to these challenges by specifying a set of fundamental principles for architecting the Next Generation OSS – NGOSS OSS through the TeleManagement Forum TMF [7, 11]. These principles are further elucidated in section 2. In particular interest, NGOSS has defined a lifecycle methodology so that OSS application development can be traced from business requirements right down to deployed systems [7]. At the heart of NGOSS lies the notion of a "Contract" which embodies the specification of services offered by an OSS component. These "contracts" are to be defined

Fifth International Workshop on Software Quality (WoSQ'07)
0-7695-2959-3/07 $20.00 © 2007 IEEE

independent of technologies/platforms used to implement them, so that they will capture 'business' needs of an SP and thus be preserved as software techniques change.

A key challenge in specifying 'contracts' in the NGOSS framework is how Non Functional Requirements - NFRs are specified. This paper examines the current approach to NFR specification in the NGOSS framework and its weakness, and proposes a novel approach based on an elastic quantitative specification technique. These are discussed in sections 2, 3, and 4. A detailed example of non-functional requirements specification of a 'contract', based on the NGOSS framework is presented in section 5. Finally section 6 provides concluding remarks and impacts of our approach and our future work.

2. NGOSS - Life Cycle Methodology and Contract Specification

The OSS community in the global telecom industry has defined a set of fundamental principles for architecting the Next Generation OSS – **NGOSS** through the **TMF** [11]. In a nutshell, NGOSS [7] applies a top-level approach for the specification of an OSS architecture where:

- **Technology Neutral** and **Technology Specific** Architectures are separated.
- The more dynamic "**business process**" logic is separated from the more stable "**component**" logic.
- **Components** present their services through well defined "**contracts**" with clear semantics.
- **Policies** are used to provide a flexible control of behavior in an overall NGOSS system.
- The **infrastructure services** such as naming, invocation, directories, transactions, security, persistence, etc are provided as a common deployment and runtime framework for use by all OSS components and business processes over a service bus.
- A common **Shared Information and Data Model** – **SID**, where all data used by components, processes and policies will follow an agreed standard format.
- A business process framework **eTOM** [12] is a framework where business processes encompassing al aspects of operating an IT enterprise from fulfillment to assurance and billing activities are mapped from top level abstract description to more detailed decompositions.

Furthermore, NGOSS specifies a rigorous methodology for architecting an OSS. The NGOSS life cycle is depicted in figure 1. There are four views of an OSS. The Business view captures business requirements irrespective of how automated computerised system will realise them. The System view describes the automated system capabilities in a technology neutral manner. The Implementation view describes technology specific system capabilities; and finally the deployment view captures the run-time components of the system.

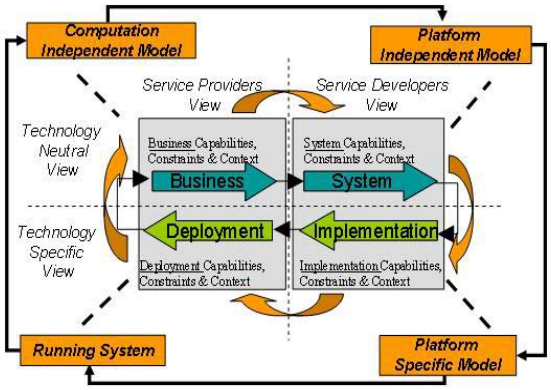

Figure 1. NGOSS life cycle

The key to a NGOSS architecture is the notion of "contract" in each viewpoint of the lifecycle. A contract specification includes the functional aspect of an OSS capability (such as billing, trouble ticketing, order handling, etc.) as well as non-functional requirements to aid procurement of third party components as well as guiding design decisions for developing an OSS application. While this approach to automation of life cycle of an OSS can be applied to functional aspects of the overall system, non-functional requirements of an OSS systems are largely expressed as rather vague qualitative statements which are not amenable to further analysis and have little value while making design decisions in subsequent phases in the life cycle [4, 5].

3. Approaches to Non-Functional Requirements Specification for NGOSS CONTRACTS for Quality Management and Product Evaluation

Unlike Functional Requirements (FR) whose significance has been widely recognized, Non-Functional Requirements (NFRs) are poorly understood [10]. Research of non-functional requirements has focused on their analysis instead of

specification [1, 6]. In fact, neglecting non-functional requirements has been counted as one of the top risks of requirement engineering. The problem of incorrect specification of non-functional requirements often leads to disputes in business contracts, wrong design and implementation trade-off decisions, poor customer satisfaction, and loss in competition.

The basic description of a NGOSS contract, no matter the view being represented, is made up of five main parts: general contract view, functional part, non-functional part, management part, and view specific model part. The non-functional part defines aspects which govern or restrict the bounds of operation of the capabilities specified by a contract [8].

No examples of non-functional parts of a contract have been given in the NGOSS contract specification yet [8, 9]. TMF does not provide any method (or process) for specification of a non-functional part in the NGOSS contract specification except the exemplary list of fields [8]. In the exemplary list, all fields are in text, and they are qualitative. The reason that qualitative non-functional requirements specification is often used in practice is that it is easy to develop. It is sometimes difficult to identify metrics to quantify a non-functional characteristic, or it is too complex and time-consuming to use metrics to quantify them. However, qualitative non-functional requirements specified in text sometimes may be hard to use for making design decisions and selecting reusable components in the product development process, and very difficult to validate since they may be ambiguous and subject to different interpretations by different stakeholders. For example, assume that we have a qualitative non-functional requirement for a billing system in a telecom company:

R_1 "the performance of the billing system shall be high".

Firstly, how to measure the performance may be unclear to developers. Secondly, "high" is qualitative and can be interpreted differently by different people.

Therefore, we propose that quantitative non-functional requirements specification needs to be used instead if precise requirements specification is needed in NGOSS specifications. Below is an example of quantitative performance requirements specification:

R_2: The response time of search of a customer account in a billing system is no more than one (1) second.

This requirement is precise and can be easily validated. An implementation of the billing system either satisfies it or not since it is crisp. However, in telecom industry, a billing system which slightly violates the requirement is usually considered to be fine. It leads to the development of elastic quantitative non-functional requirements specification in NGOSS discussed below.

4. Towards Elastic Quantitative Non-Functional Requirements Specification in NGOSS for Quality Management and Product Evaluation

In this section, we further propose to enhance capability of existing NGOSS contract specification which currently is limited to be only qualitative by developing elastic quantitative non-functional requirements specification which enables trade-offs in a development process. Non-functional requirements enforce constraints on a system or service. Clarity of non-functional requirements, such as availability, is vital to for efficient business operation and product development.

We propose two methods for quantitative specification of a non-functional requirement in NGOSS: 1) crisp, and 2) elastic. A crisp quantitative non-functional requirement imposes a rigid constraint on a non-functional characteristic of a system or service. It is either satisfied or dissatisfied. Considering the following crisp quantitative non-functional requirement:

R_3: The worst-case latency of billing must be less than one (1) second.

If the billing of a system takes 1.05 seconds for a test case in the testing process, it does not satisfy the above requirement, and the system realization is not acceptable. Crisp quantitative requirements are easy to validate. The crisp quantitative non-functional requirement specification is used widely in industry. Actually, BT has adopted it in specification of performance of operations for its telecom capabilities.

Elastic quantitative non-functional requirements specification for NGOSS is based on works on imprecise requirements specification [2]. An elastic quantitative non-functional requirement imposes an elastic constraint on a non-functional characteristic of a system or service using a membership function of a qualitative term to characterize its satisfaction. Below is an example of an elastic quantitative non-functional requirement:

R_4: The worst-case latency of billing must be SHORT,

where SHORT is a linguistic term in fuzzy logic whose membership function characterizes satisfaction of the above requirement as shown in Figure 2. In the

figure, one (1) represents the highest level of requirement satisfaction by a realization of system or service, and zero (0) represents the lowest level of requirement satisfaction by a system realization or service.

If the billing of a system realization takes 0.8 seconds in the worst case in the testing process, its satisfaction degree is one which is the highest. It indicates that it completely satisfies the requirement. If it takes 1.5 seconds in the worst case, its satisfaction degree is around 0.5 and it partially satisfies the requirement although it is acceptable. In the elastic non-functional requirement specification, a minimal threshold for its metric value is usually specified. It indicates that a system realization whose metric value is below this threshold is not acceptable. For example, if the billing of a system realization takes three (3) seconds in the worst case which is greater than the threshold of two (2) seconds, its satisfaction degree is zero (0) and it is completely unacceptable.

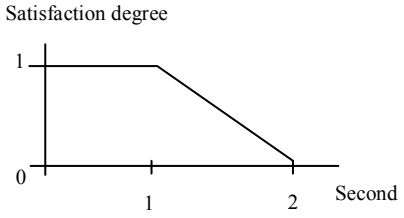

Figure 2. Satisfaction function for requirement R_4

Elastic quantitative non-functional requirements specification enables trade-offs in the design of a system and selection of reusable components, which is impossible if crisp quantitative non-functional requirements specification method is used. This is absolutely important when a design trade-off decision for resolving a conflict among non-functional requirements, which often exists in many applications, needs to be made. For example, suppose that we need to select a reusable billing component for a new product. Assume that there is a reusable component $COMP_1$ which provides the functionality needed for the new product, and its worst case latency of $COMP_1$'s billing is 1.01 seconds. Different results can be obtained using crisp and elastic requirements specification techniques.

We discuss crisp non-functional requirements specification for the new product first. Assume that crisp requirement R_3 is its latency requirement. $COMP_1$ can not be reused for the new project since it violates the above requirement.

Now we discuss how to use elastic non-functional requirements specification technique for the new product. Assume that the elastic requirement R_4 is its latency requirement. Based on this requirement, $COMP_1$ can be reused for the new project since it has a satisfaction level which is close to one (1) which is the highest and far greater than the minimal threshold of satisfaction based on its satisfaction function in Figure 2. This result is much more desirable and practical than the one obtained using crisp non-functional requirement specification discussed above in many applications. In addition, the elastic quantitative non-functional requirements specification also makes non-functional requirements easily evaluated and validated than qualitative non-functional requirements specification.

To overcome problem of the lack of guidance and example for specification of non-functional requirements for a NGOSS contract in NGOSS standard documents, a complete contract example, CRM-SM&O Customer Problem Handling, is developed by adding its non-functional part in a incomplete contract provided by NGOSS [9] using qualitative, crisp quantitative, and elastic quantitative non-functional requirements specification techniques discussed above.

5. An Example of Non-functional Specification for a NGOSS Contract

TMF is working on a draft of examples of a NGOSS contract which contains no examples of non-functional parts [9]. In this draft, a contract in NGOSS business view contains multiple capabilities. A capability in turn contains multiple processes.

We now extend a contract example, CRM-SM&O Customer Problem Handling, in the draft [9] by adding its non-functional part to illustrate the above framework. It deals with both customer order and service order handling. Here is a description of its business capabilities [9]:

This Contract defines interaction between Customer Relationship Management and Service Management areas within an enterprise (as represented by the relevant eTOM CRM and SM&O processes [12].

It directly interacts with two processes in eTOM for this Contract, at eTOM Level 2 [12]:

- Order Handling (in OPS-CRM)
- Service Configuration & Activation (in OPS-SM&O).

They are decomposed into eTOM level 3 processes in a CRM and SM&O fulfillment process flow, such as *validate customer order* and *activate*

service [9]. Next, we are going to complete the non-functional part for the contract example. It will contain examples of three specification techniques: qualitative, crisp quantitative, and elastic quantitative non-functional requirements specification. It must be noted that it is deliberately restricted to a simple scenarios and simple data since it is intended to illustrate principals. Actual requirements may vary from company to company.

In the specification of non-functional part for the following example of contract, we still use the categorization of non-functional requirements recommended by TMF [8] although we would suggest replacing the category of deployment with category of quality in a business view contract since it is supposed to be deployment independent.

NGOSS Contract Example – CRM-SM&O Fulfillment Information Handling

Other parts in the contract [9]
Business View – Non-Functional

Deployment-Related

- **Availability**

ER_1: Availability of all business capabilities specified in the contract must not be lower than 99.99%.

- **Performance**

ER_2: Performance of all business capabilities specified in the contract must be very good.

This requirement is specified qualitatively, and in many cases it may be appropriate although it may have different interpretations from different stakeholders and may be hard to validate. We can transform it into more precise requirements if the qualitative requirements specification is not appropriate. Performance is usually characterized by sub-characteristics, such as time-efficiency and resource-efficiency. ER_2 can be transformed into lower level requirements based on its sub-characteristics. An example of such requirements may look as follows:

$ER_{2,1}$: The time-efficiency of all business capabilities specified in the contract must be very good.

Once again, this requirement is qualitative and is not precise. The time-efficiency is usually characterized by several metrics, such as latency and throughput. Non-functional requirements can be derived from $ER_{2,1}$ based on these metrics. It can be illustrated by using latency as an example. Before transformation, we need to extend the definition of metric *latency* of an operation to a capability. We can define latency of a capability of handling a customer order to be the time needed for completing it after a customer request is received. An example of non-functional requirements for capability *handling a customer order* can be derived from $ER_{2,1}$ as follows:

$ER_{2,1,1}$: The latency of handling a customer order should be SHORT.

An example of satisfaction function can be defined for SHORT of $ER_{2,1,1}$ as follows:

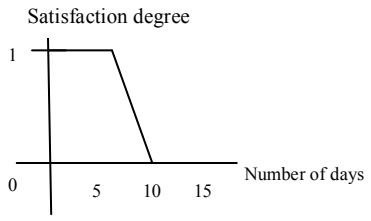

Figure 3. Satisfaction function for latency requirement $ER_{2,1,1}$

In this figure, one (1) represents total satisfaction and zero (0) represents total dissatisfaction. Basically, it indicates that if a capability realization takes more than ten days to complete a customer order, it is totally unacceptable; if it takes no more than five days, it achieves the highest level of satisfaction; and if it takes between five and ten days, its satisfaction level is gradually decreased as number of days is increased. The numbers used in this example are for illustration only.

- **Safety**

There is no safety requirement for business capabilities in this contract.

Organization-Related

- **Business Environment**

ER_3: Some of business processes may be business environment specific.

- **Organization Limitations**

ER_4: Some of business processes may be organization specific

- **Market Limitations**

ER_5: Some of business processes may be market specific

- **Financial Limitations**

ER_6: Financial loss from cancellation of customer orders due to service delay must be MINIMAL.

An example of satisfaction function can be defined for MINIMAL of ER_4 in Figure 4. In this figure, one (1) represents total satisfaction and zero (0) represents total dissatisfaction. Basically, it indicates that if percentage of revenue lost from cancelled orders due to service delay is no more than two (2)

percent, it achieves the highest level of satisfaction; if percentage of revenue lost from cancelled orders due to service delay is equal to or more than five (5) percent, it is totally unacceptable; and its satisfaction level is gradually decreased when percentage of revenue lost from cancelled orders due to service delay is increased from two (2) to five (5) percent. The numbers used in this example are for illustration only.

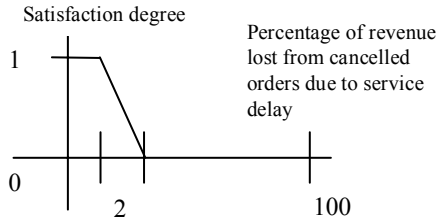

Figure 4. Satisfaction function for minimal financial loss due to service delay

Legal-Related

- **Regulatory Limitations**
  None identified
- **Legal limitations**

ER_7: Customer order data must not be released for public usage without consent.

Miscellaneous

None identified

6. Concluding Remarks

The quality problems, increased cost, and lack of agility in building OSS applications are caused by lack of standards, methodologies, and data / component / process description languages and implementations that are often technology specific and are constantly subject to change, and hence creating a barrier to business agility and quality management as requirements change as well as new middleware's are introduced. NGOSS of TMF has already laid out the foundations for a rigorous methodology which enables high level and more abstract business focused models of OSS applications be designed for a given software platform. A key enabler for increased use of NGOSS is the concept of a "contract".

In this paper, we developed a framework for contract based non-functional requirements specification using qualitative, crisp quantitative, and elastic quantitative non-functional requirements specification techniques. An example of non-functional specification of a NGOSS contract has been presented based on these techniques. The quantitative non-functional requirements specification techniques, especially elastic non-functional requirements specification which enables trade-offs in a development process, enhance capability of current NGOSS contract specification which is only qualitative currently.

Future works include aggregation of non-functional requirements and integration of the proposed approaches with other methods in valued based software engineering [13].

7. References

[1] L. Cysneiros and J. Leite, "Nonfunctional requirements: from elicitation to conceptual models," *IEEE Tran. Software Engineering,* vol. 30, pp.328-350, May 2004.

[2] Xiaoqing (Frank) Liu and John Yen, "An Analytic Framework for Specifying and Analyzing Imprecise Requirements", *Proc. of the 18th IEEE International Conference on Software Engineering (ICSE-1996)*, pp. 60-69, Berlin, Germany, March, 1996.

[3] Xiaoqing (Frank) Liu, "Specification of Non-Functional Requirements and Its Application for NGOSS Contract Specification", *BT Internal report*, July 2005.

[4] Georgalas N, Azmoodeh M, "Using MDA in Technology-independent Specifications of NGOSS Architectures", *1st European Workshop on MDA (MDA-IA 2004)*, Enschede, The Netherlands, March 2004.

[5] Georgalas N., Azmoodeh M., Clark T., Evans A., Sammut P., Willans J., "MDA-Driven Development of standard-compliant OSS components: the OSS/J Inventory Case-Study", *Proceedings of the 2nd European Workshop on Model Driven Architecture with emphasis on Methodologies and Transformations (EWMDA 2004)*, Canterbury, UK, 7-8 September 2004.

[6] J. Mylopoulos, L.Chung, and B. Nixon, "Representing and using non-functional requirements: a process-oriented approach," *IEEE Trans. Software Engineering,* vol. 18, pp.483-497, June 1992.

[7] TeleManagement Forum - New Generation Operations Systems and Software, http://www.tmforum.org/browse.asp?catID=1911.

[8] TeleManagement Forum, NGOSS Architecture Technology Neutral Specification: Contract Description: Business and System Views, *TMF 053B*, Feb., 2004.

[9] TeleManagement Forum, NGOSS Contract Examples: Examples of the NGOSS Lifecycle and Methodology for NGOSS Contract Definition, GB-921 Addendum N, Team Draft 4, 2005.

[10] B. Paech and D. Kerkow, "Non-Functional requirements engineering - Quality is essential," *REFSQ'04*, 2004.

[11] www.tmforum.org.

[12] TMF eTOM process framework, from www.tmforum.org.

[13] Liguo Huang and Barry Boehm, "How Much Software Quality Investment Is Enough: A Value-Based Approach", IEEE Software, September/October 2006.