Assessing Relations between Non-Functional Requirements

Michael C. Jaeger and Anne Hoffmann

Siemens Corprate Research and Technology, Software & Engineering (CT SE)
Otto-Hahn-Ring 6, D-82000 München, Germany
michael.c.jaeger@siemens.com
anne.hoffmann@siemens.com

Abstract: The development of requirements involves also necessary checks for consistency, coherency and identification of dependencies between them. While approaches for requirements development exist that cover the functional characteristics of a system, the support for non-functional requirements is not well elaborated today. This work proposes to assess non-functional requirements (NFRs) by their relations between them in order to provide a reference for a systematic analysis of NFRs.

1 Introduction

Generally, a non-functional characteristic describes – contrary to a functional characteristic – the behaviour of system during its use. Accordingly, during the engineering process of a system, the requirements are separated into two major parts: functional requirements and non-functional requirements. Functional requirements constitute the nature of the desired system. In addition to that, non-functional requirements (NFRs) express requirements in order to enable the use of such system. Thus, requirements engineering must consider NFRs as important as the function of a system.
This work is driven by the fact that an NFR implies different actions and decisions during the subsequent specification and design of a system. Consider the example that a system must provide extraordinary reliability. In this case, a possible implication on the design is the redundant set up of critical components of that system. However, this also implies a higher operational cost, because the footprint of the system increases leveraging a redundant design. Thus, in this simple example, the NFR referring to the reliability interferes with a possible NFR on operational cost. The important point for the requirements engineering is: *What are the relations between different NFRs and what are the resulting implications for the requirement engineering phase?* In order to answer to this question, this work aims to identify the dependencies among NFRs.
To answer this question, we must determine the relevant NFRs for the engineering of software-intensive systems. The discussion on relations among NFR cannot be general or on an abstract level, but must consider the nature of individual requirements in a specific domain. For example, implications on capacity when constructing buildings are different as when considering software systems. Thus, this work entirely focuses on the requirements engineering of software-intensive systems. It lists relevant non-functional characteristics of these and it explains their interpretation today.

Finally, we can analyse how these NFRs interrelate with each other. Based on the interpretation of the NFRs, the implications are explained. Then, this discussion puts these implications in a common context in order to identify conflicts or – what is also considered possible – favouring effects. The rest of this work is organised in the following way: the next section gives a brief overview on requirements engineering. Section 3 discusses relevant NFR for software-intensive systems and leads to a rough overview of dependencies among these. Finally, Section 4 presents our conclusions so far and our next steps in this field.

2 Brief Overview on Requirements Engineering

In general, dealing with requirements can be separated into the development and the management of requirements. In requirements development, the main activities are elicitation, analysis, and specification. Beginning with the elicitation, an appropriate scope of the desired system is defined that limit the extent of the requirements. Then requirements are collected and a common understanding among the stakeholders of the development process needs to be ensured. In multi-party software development projects, the elicitation also involves the negotiation of requirements, for example between client and developers. As the next step, the analysis of the requirements takes place: Requirements are classified and checked for consistency and completeness. Also, dependencies among requirements would be identified in order to cover potential conflicts. Subsequently, the third step is about the specification of requirements, which includes their presentation, their structuring and the setting of criteria for determining under what conditions a product meets the requirements.
Besides the development of requirements, the management of requirements is also necessary, because in a software development process, subsequent steps, such as the design or the implementation take place beyond the developed requirements. Thus, a change management of the requirements is necessary that allows to apply and track modifications. Our work proposes to enhance the development of requirements: Our aim is to bring the focus of requirements engineering to checking consistency and dependencies with in particular with NFR. Our anticipated result is a comprehensive reference that provides requirements engineers with a guide when structuring NFRs. This reference enables the requirements engineer to:

- **Define implications of particular NFRs:** Using this framework, the requirements engineer can pick up relevant implications for the entire set of particular NFRs.
- **Discover possible conflicts between NFRs:** Knowing about implications of particular NFRs enables the requirements engineer to get a more comprehensive image of the desired system. Within this image, conflicting NFRs become explicit.
- **React to conflicts in the requirements phase:** With the identification of conflicts, the requirements engineer in able to apply appropriate corrections in an early phase of the system engineering process. This improves the control on the entire engineering process and reduces the overall efforts.

3 The Non-Functional Requirements and their Dependencies

The term non-functional requirement refers to the non-functional characteristics as requirements on the desired system. The main question that rises with non-functional characteristics is what their difference to functional characteristics are. Cases exist where a characteristic is functional in one application and is non-functional in another. For example, ensuring the provision of a communications link with a particular bandwidth for voice transmissions, the provided bandwidth is actually the provided function. However, with a Web server application, the bandwidth is clearly a non-functional requirement since the application would work with almost any bandwidth characteristics.
The research consensus in order to answer this question is to define the functional core of a provided function as discussed by Werner [Wer07]. Then, a characteristic that is not covered by the functional core is named non-functional. Therefore, for a precise definition, it depends on the application to unambiguously define which characteristics are considered non-functional. However, in most cases, the non-functional characteristics can be easily identified. In this area, several characteristics exist as a group. A large group is comprised by the quality-of-service (QoS), which are considered a subset of non-functional characteristics. Different works state that QoS characteristics are non-functional characteristics that are quantifiable (e.g. [OMG04]). In other words, the QoS refers to the measureable characteristics when a system provides its functionality.
In addition, to the cluster of QoS characteristics, other groupings exist such as the dependability or the RAMSS characteristics (Reliability, Availability, Maintainability, Security and Safety). Both are umbrella terms as well that comprise different individual characteristics. For this work, different work in the domain of distributed software systems are considered: the ISO QoS reference model as a supplemental proposal for the ISO Reference Model for Open Distributed Processing (RM-ODP) [ISO98] and the OMG UML QoS Profile [OMG04]. In addition, we consider the works of Colouris [CDK02], Tanenbaum and Goodman [TG01], Ran [Ran03] and Buschmann et al. [BMR$^+$96] as references for non-functional characteristics in the area of distributed software systems in an informal way. Our selection includes the following characteristics:

- Organisational characteristics:
  - Location of provision: This characteristic refers to the location where the system is provided. This could involve for example different end user devices.
  - Cost (-): The cost for either realising or using the provided system.
  - Affiliation: The affiliation denotes the organisation that provides the system or its parts. For example, legal issues might be relevant with specific affiliations.
- Characteristics from the dependability area:
  - Reliability (+): The reliability of a system denotes the ratio of successful systems execution according to the specification.
  - Availability (+): The availability denotes a ratio on how many times the system's functionality can be accessed.
  - Safety (+): The safety is meant to cover the operational safety.

    - Security (+): This umbrella term refers to different particular characteristics, such as confidentiality. In some works, the safety is also included.

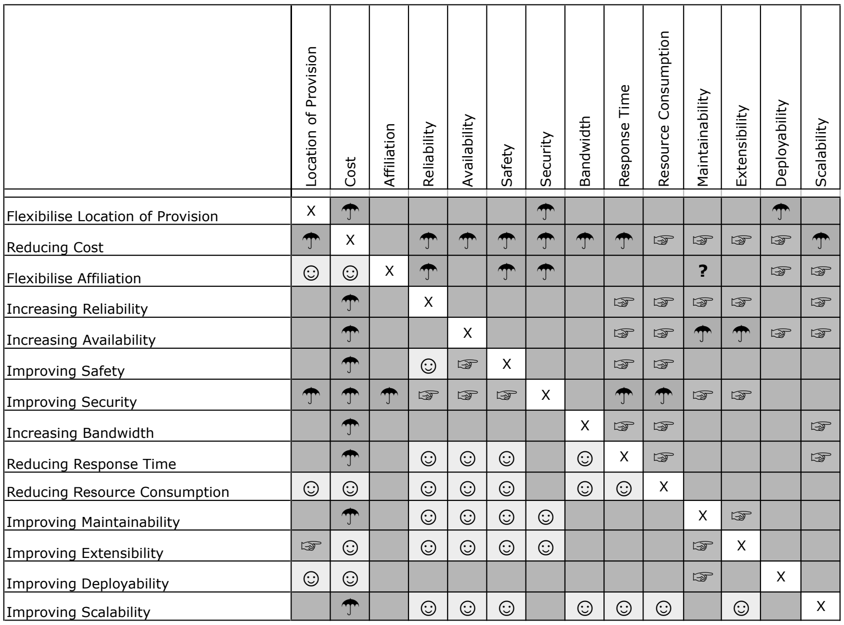
- Performance characteristics:
    - Bandwidth (+): The bandwidth denotes the amount of data that a system can process within a defined time interval.
    - Response Time (-): The amount of time to process a particular request.
    - Resource Consumption (-): This characteristic denotes the consumption of technical resources, for example, CPU time or main memory.
- Technical characteristics:
    - Maintainability: During its lifetime, the software must be updated, corrected or data must be maintained. In order to preserve the operation of the software, efficient mechanisms for maintenance are desired.
    - Extensibility: This characteristic refers to the suitability of the software for applying extension for the support of new functionality.
    - Deployability: Describes how easy or difficult a software is to deploy.
    - Scalability: Denotes how well a system performs with a rising amount of use.

In this list, a plus or minus sign indicates that the particular characteristic is quantifiable and thus has a direction. A minus indicates that lower values are considered better; a plus vice versa. A systematic selection could be based on assessing actual occurrences in the requirement engineering of industrial projects. However, this would exceed the available space for this work and is, thus, postponed as further work.

These characteristics result in particular implications on the requirements engineering when involved as non-functional requirements. Such implications result in new requirements or complement existing ones. For example, requirements on the scalability supplement requirements on the response time. A requirement on a high level of extensibility implies modular system architecture. With maintenance, the implications are more complicated as this is understood as a characteristic in order to improve the general availability, or reliability. However, modularity can result in a performance penalty, as this, for example, is an issue with the design of operation system kernels. But, to show the difficulty of this interpretation, modularity can also lead to the solution on the response time: For example, throughput-intensive application servers are modularised in order to balance the load in an optimal way.

In order to establish a starting point with the goals of in this work, an empirical assessment serves as a first proposal. This proposal is summarised in Figure 1. In this figure, a relation denotes that improving a particular characteristic:

1. worsens the other, depicted by an umbrella,
2. has no general impact on the other, shown in the table as grey box,
3. implies the improvement of another characteristic,
4. results in improving another characteristic, indicated by green boxed smiley.



	Location of Provision	Cost	Affiliation	Reliability	Availability	Safety	Security	Bandwidth	Response Time	Resource Consumption	Maintainability	Extensibility	Deployability	Scalability
Flexibilise Location of Provision	X	☂					☂						☂	
Reducing Cost	☂	X		☂	☂	☂	☂	☂	☂	☞	☞	☞	☞	☂
Flexibilise Affiliation	☺	☺	X	☂		☂	☂				?		☞	☞
Increasing Reliability		☂		X					☞	☞	☞	☞		☞
Increasing Availability		☂			X				☞	☞	☂	☂	☞	☞
Improving Safety		☂		☺	☞	X			☞	☞				
Improving Security	☂	☂	☂	☞	☞	☞	X		☂	☂	☞	☞		
Increasing Bandwidth		☂						X	☞	☞				☞
Reducing Response Time		☂		☺	☺	☺		☺	X	☞				☞
Reducing Resource Consumption	☺	☺		☺	☺	☺		☺	☺	X				
Improving Maintainability		☂		☺	☺	☺	☺				X	☞		
Improving Extensibility	☞	☺		☺	☺	☺	☺				☞	X		
Improving Deployability	☺	☺									☞		X	
Improving Scalability		☂		☺	☺	☺		☺	☺	☺		☺		X

Figure 1: Summary on Relations

This figure represents a first result on our personal assessment. Some of the mentioned authors (e.g. [BMR+96]) already discuss implications of particular NFRs. However, to our best knowledge, we are not aware of work that provides a comprehensive presentation. Certainly, the particular relations as given by Figure 1 could be discussed in further detail. However, due to space limitations, we can give only a brief summary on our assumptions:

- Location of provision implies a modularity to some certain extend, because specification of special end user devices or particular location of provision improve with a separation of the business logic from presentation concerns.
- A cost improvement generally reduces other QoS characteristics. However, some non-functional characteristics result in improvement of the cost, such as low resource consumption or extensibility.
- To bring the affiliation into relation to other non-functional characteristics is generally difficult. It is listed here to demonstrate an unusual candidate in the discussion.
- The reliability implies some characteristics such as good response time or good safety. The availability is related to the reliability w.r.t. the other characteristics.
- The safety implies the proper working of the system and also benefits from a good maintainability.

- The security forms a trade off with performance characteristics and thus has also a negative on resource consumption.
- The bandwidth and response time are performance characteristics and thus result in good availability and reliability but worsen resource consumption and cost. They both benefit from a good scalability. The scalability is a hidden characteristic that results in other observable characteristics, such as reliability or response time.
- The maintainability has effects on most of the other characteristics. It results in good reliability but can also result in more cost. The same applies to the extensibility.
- The deployability has an effect on the cost and on the location of provision. However regarding the rest of the characteristics it develops neutral.

4 Conclusions and Future Work

This work has presented an overview on non-functional characteristics relevant for the requirements engineering. In addition, it has presented the role and handling of non-functional requirements in the requirements engineering field. Based on this, an outlined overview on how NFRs could depend on each other was given. Our future aim is to elaborate on this work in iterations in order to provide a mature reference about dependencies between NFRs. As the next steps in our future work we consider relevant an assessment of the importance and popularity of different non-functional characteristics for requirements engineering. This could involve *1)* the analysis of projects where the results of requirements engineering is accessible, *2)* the conduction of questionnaires and interviews, and *3)* further research in books and articles where particular implications are already discussed for individual NFRs.

References

[BMR$^+$96] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. *Pattern-Oriented Software Architecture – Volume 1.* Wiley &Sons, 1996.

[CDK02] George Colouris, Jean Dollimore, and Tim Kindberg. *Verteilte Systeme.* Pearson Education Deutschland, München, Deustchland, 2002.

[ISO98] ISO/IEC. CD 15935 Information Technology: Open Distributed Processing - Reference Model - Quality of Service. (CD Ballot), October 1998.

[OMG04] Object Management Group OMG. UML Profile for Modelling Quality of Service and Fault Tolerance Characteristics and Mechanisms. ptc/2004-06-01, June 2004.

[Ran03] Shuping Ran. A Model for Web Services Discovery with QoS. *SIGecom Exch.*, 4(1):1–10, 2003.

[TG01] Andrew S. Tanenbaum and James Goodman. *Computerarchitektur.* Prentice Hall, Upper Saddle River, New Jersey, USA, 2001.

[Wer07] Matthias Werner. Eigenschaften Verlässlicher Systeme. Habilitationsschrift der Technischen Universität Berlin, 2007.