The Bad Conscience of Requirements Engineering: An Investigation in Real-world Treatment of Non-functional Requirements

Andreas Borg[A], Angela Yong[B], Pär Carlshamre[C], Kristian Sandahl[A]
[A]*Dept. of Computer and Information Science, Linköping University, {andbo, krisa}@ida.liu.se*
[B]*Swedish Meteorological and Hydrological Institute, angela.yong@smhi.se*
[C]*Ericsson AB, par.carlshamre@ericsson.com*

Abstract

Even though non-functional requirements (NFRs) are critical in order to provide software of good quality, the literature of NFRs is relatively sparse. We describe how NFRs are treated in two development organizations, an Ericsson application center and the IT department of the Swedish Meteorological and Hydrological Institute. We have interviewed professionals about problems they face and their ideas on how to improve the situation.

Both organizations are aware of NFRs and related problems but their main focus is on functional requirements, primarily because existing methods focus on these. The most tangible problems experienced are that many NFRs remain undiscovered and that NFRs are stated in non-measurable terms. It became clear that the size and structure of the organization require proper distribution of employees' interest, authority and competence of NFRs. We argue that a feasible solution might be to strengthen the position of architectural requirements, which are more likely to emphasize NFRs.

1. Introduction

Requirements Engineering (RE) as a whole is a maturing research area and many of its ideas and techniques are well established nowadays. However, Chung et al. [7] point out that so far non-functional requirements (NFRs) have received relatively little attention in the literature and that other, less critical factors in software development are significantly better understood. An influential example of existing literature is Davis's [8] description of how to specify NFRs ("nonbehavioral requirements" in his terminology), based on the "Software Quality Characteristics Tree" presented by Boehm et al. [4]. Other work is much concentrated on quantification (e.g. [2, 3, 9]). The only major work on NFRs in recent years is the work of Chung et al. [7], which provides a framework for letting NFRs drive the overall systems design process.

Real-world examples describing how NFRs are treated in software-developing organizations are particularly rare. In this paper we provide that kind of information by presenting a case study involving two organizations, Ericsson and SMHI. The contexts of the cases differ in several ways. Ericsson is a global telecommunications company with a long tradition in large-scale software development, the organization is hierarchically structured and the overall product management is market-driven. In contrast, SMHI normally develops software in relatively small projects and in cooperation with known and available stakeholders.

The overall purpose of the case study was to document state-of-the-practice of NFR management in industry and to corroborate or challenge the sparse literature. Our assumption was that software developing organizations are aware of NFRs and NFR-related problems, but that the main focus is on functional requirements (FRs). We base this assumption on the fact that most processes and techniques used in industry today are oriented towards FRs, e.g. the Rational Unified Process (RUP) [13]. We also wanted to know what problems practitioners face that can be related to NFRs, and if these problems can be addressed using existing knowledge from literature.

The remainder of the paper is organized as follows. The contexts of the cases are described in Section 2. Section 3 describes the NFR management of each case respectively and in Section 4 the main problems and difficulties found are listed and analyzed. A more general discussion is provided in Section 5 and, finally, the overall conclusions are summarized.

2. Context descriptions

2.1. Ericsson

The Ericsson office in this case study develops technically administrative systems for mobile telecommunication. Like other Ericsson offices it obeys company policies and action programs. Yet it is autonomous to a great extent since the company is heterogeneous with many products

and offices world-wide. The organization studied could be described as mature and has significant system development experience, is ISO 9001 certified and also deploys CMM at level 3. The employees are generally well-educated (typically M.Sc.).

The respondents are associated to a specific part of a larger product. This part is called OSS (Operation Support System) and is the support system of the BSS (Base Station System) product. From an organizational point of view (see Figure 1), OSS is located on the same level as the systems it supports (Base Station Controller and Base Transceiver System). It is relatively small compared to the entire mobile communications system but in absolute figures, however, the product is rather large involving three different development categories (performance management, configuration management and fault management) with more than a dozen design units (DUs) associated with each one of them. The number of people working with it was close to 400 at the time of the case study.

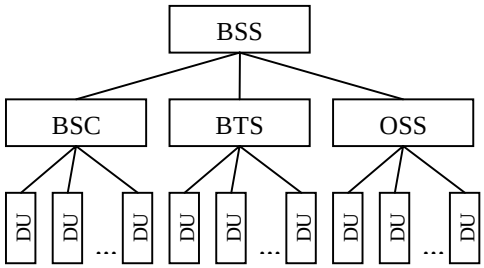

Figure 1. The Ericsson BSS organization

The OSS product has been generally available for more than a decade and the current version is the tenth release. The product is therefore to be considered as mature.

2.2. SMHI

SMHI, Swedish Meteorological and Hydrological Institute, is an authority with extensive service and business operations within the areas of meteorology, hydrology and oceanography. The workplace comprises a variety of professions and employs around 600 persons. The organization also conducts its own research in several disciplines, e.g. meteorology and climatology. However, the case of interest for this paper is the central IT department, in which around 90 people work. Many employees hold some kind of university degree, and both organization and many individuals have extensive system development experience.

The main activity of the IT department is software development, including developing new systems and improving and maintaining existing systems. The department is also responsible for purchasing standard systems from external suppliers.

Customers are either from within the organization or collaborating partners (both government agencies and private companies). Consequently, end users and customers are often involved in the requirements work and contracts normally exist for the systems being developed.

2.3. Methodology issues

In order to understand the current industry situation regarding NFR management and NFR-related problems, a series of interviews were carried out within each organization. Our belief was that the situation was best assessed asking the practitioners themselves.

A total of 14 professionals, seven in each case, with varying experiences and responsibilities were interviewed. The respondents represented different roles and perspectives, but all of them were involved with product development in some way. Thus, the product management perspective on NFRs is not represented in the case study.

A number of questions were prepared and used throughout the interview series. However, the exact focus of specific interviews varied slightly with each respondent's area of expertise.

The interview series at Ericsson was completed before the series at SMHI took place. The respondents in the first case (Ericsson) were selected from the OSS system level and from several of its design units. In the latter case (SMHI) respondents were selected to make the results from the cases as comparable as possible. These results are presented and analyzed in the following sections.

3. Understanding and managing NFRs

3.1. Ericsson

As already mentioned, OSS is a mature product developed by a mature organization. In theory, this situation presents good opportunities to focus on system quality, i.e. NFRs, since the most critical functionality is already released and efforts can be devoted to improving various quality aspects of the system. On the other hand, it can be argued that the organization's high maturity level and skilled designers might make explicit NFR focus unnecessary; participating people and the organization they represent know what is needed to produce good system quality, and do so as a natural part of work.

Since OSS is a support system, OSS requirements originate to a great extent from requirements on the systems it supports (BSC and BTS). Adding a new BSS feature might generate new requirements to all subsystems, including OSS. This circumstance, OSS being part of a hierarchical organization and product structure,

means that end user contact is limited. Requirements elicited directly from end users are thus very rare.

Even though several subsystems are part of the same system, the processes and routines within the subsystems vary to a great extent. The situation at OSS can be completely different from the others and the respondents have for example stated that the BSC organization has better structure and better requirements practices than OSS. One respondent said that "*it surprises me that we are allowed to develop this system as spontaneously as we do*". He also added that a possible explanation could be that "*So far no one has bought a mobile telecommunications system because of its support system. This might change in the future, and then the interest for non-functional requirements at OSS is likely to increase*." This change will probably be noticeable when new operators, with less technical and more business-oriented background, enter the market.

Nevertheless, the general NFR knowledge at OSS is good. All respondents know the basic difference between NFRs and FRs and are able to adequately define the meaning of NFRs. Most respondents think of NFRs as "constraining requirements", "system properties" or "*how*-aspects on *what*-requirements". As expected, the NFR examples given by respondents reflect their role within the organization. Respondents working directly with requirements at the system level provide more comprehensive definitions, while e.g. respondents primarily working with integration and verification mostly reason in terms of NFR types like performance and availability. Characteristic for the entire organization is that NFR types that are typical for traditional telecommunication systems (capacity, performance, availability and reliability) gain more attention than others and these are mentioned by most respondents. These types are frequently referred to as *system characteristics* and seem to have precedence over other types. In fact, a system level respondent claimed that "*it is difficult to make people adopt non-functional requirements that are not system characteristics*".

No NFR-dedicated activities are carried out within the development process at OSS. Typically NFRs are added to requirements documents as they occur. However, there are two fundamental and business-critical NFRs given to the OSS system level, constraining the development on a high level:

1. A given number of mobile network cells must be supported (capacity).
2. The performance of the system must be at least as good as in the previous release.

Lower level NFRs are mainly elicited by a group consisting of system level personnel and representatives from involved design units, based on requirements from the related organizations. It is important to note, though, that these design units are considered rather autonomous and relatively strong compared to the system level, which means that many decisions are taken on the lowest hierarchical level.

Most respondents have worked with OSS for many years. They testify that NFRs have received varying level of attention during the years, and that functionality and technical solutions dominate today. One way of forcing attention to NFRs earlier was that types considered most important had their own headings in the requirements specification template. This meant that people had to consider these types when writing the specifications.

Performance requirements are considered properly tested within the OSS organization, and likewise system characteristics NFRs in most cases. However, NFRs of types that are not included among system characteristics types, e.g. usability, are often poorly tested. A common mistake is to equate a system behavior to a system property: "*We have tried to break the system for half an hour and we haven't succeeded. That means that the system is secure!*" The concept of usability is widely spread within the organization in the sense that many people talk about it, but proper testing or evaluation is rarely performed.

3.2. SMHI

The IT department at SMHI is a project-oriented organization. Several projects coexist and development of new systems is mixed with operating, maintaining and improving existing systems. The work at the department is governed by an overall policy called the *technical plan*. This plan includes architectural concerns as well as the use of models and processes. The project model gives guidelines on how work should be conducted in a project and the routines for operating and maintaining systems are defined in a software maintenance model. When it comes to processes, the IT-department is about to become ISO 9001 certified, and the Rational Unified Process (RUP) [13] is under evaluation for regular use. Much of requirements management and NFRs are related to and expressed using RUP terminology, artifacts and approaches.

All interview respondents have some understanding for NFRs (or "quality attributes" that some of the respondents prefer to use). Their orientation towards RUP is revealed by the most common single explanation to what NFRs are: "the requirements that can not be covered with use cases". Most frequent examples given are availability and performance.

Some years ago, before involving use cases in the development process, all requirements were assembled in a long list. The attention NFRs receive has increased over the years, sometimes by NFR-dedicated activities and sometimes by considering NFR aspects at requirements workshops. However, focus is on use cases and thus on

functionality. Even though this condition is likely to last at SMHI there are some possible NFR-related improvements:

1. Good requirements processes consider all requirements, both functional and non-functional. Even though focus will remain on functionality, NFRs will receive an acceptable level of attention.
2. RUP emphasizes that architectural requirements should be captured first. NFRs often influence architecture and are thus likely to gain more attention.

A new project typically schedules two or three days for requirements workshops involving various stakeholders (normally including customers and end users). One of the respondents noted that if RUP procedures are followed, three separate workshops should be held: a requirements workshop, a use case workshop and a use case analysis workshop. This is, however, not yet standard procedure within the department. If generalizing, NFRs are in most cases elicited as they appear when system functionality is considered or when use cases are modeled. Discovered NFRs are added to a specific artifact called *supplementary specification*. When possible and appropriate, NFRs are also mapped to one or several use cases. In some projects prioritization of NFR types are performed during the workshops, but generally availability is the most important type. Scalability, maintainability and portability are also often important.

The supplementary specification artifact contains a group of NFR descriptions. This grouping is predefined in an artifact template and may be adapted by each project. In one of the ongoing projects, a defined set of attributes accompanies each NFR description. It includes for instance priority of each NFR, connection (per NFR) to relevant use cases (one or several) and a verification attribute (stating verification method to be adopted). Verification methods are often test or formal inspection, but more informal methods like "ocular inspections" and conformance to policies are sometimes used as well.

The test activities at the IT department generally work well. There is always a person responsible for testing and there exists to a great extent an accompanying set of defined test cases. According to some of the respondents, performance requirements are considered relatively easy to test and verify, while NFR types like availability and security are more difficult to deal with. Since availability is one of the most important NFR types at SMHI, extensive efforts are made to test these requirements, and load tests for several weeks are normally carried out.

The requirements work in a project at SMHI is usually non-iterative, meaning that the most crucial parts of the requirements specification should be completed before the start of design. Most of the requirements, including NFRs, are elicited directly from discussions with customers and end users. In addition, some overall NFRs that constrain a project, e.g. economical matters and security policies, are occasionally given from management level. In some projects, new NFRs that are discovered after having finished the requirements phase (mostly from within the organization) are taken care of with formal change management procedures.

4. Findings regarding NFR-related problems

4.1. Problem overview

Table 1 assembles problems expressed directly by practitioners, which means that they can be of different abstraction levels. Furthermore, some problems are overlapping and some are probably consequences of other problems. For each problem identified, the table also includes which organization(s) who has stated it. Problems that are suggested by only one of the organizations are not necessarily invalid for the other organization.

The problems in Table 1 reflect the fact that NFR-related problems occur throughout the entire development process. Roughly, problems occur at four stages in our study:

1. **Elicitation**. Both organizations find it difficult to elicit NFRs. The main reason is probably due to FR focus, both as a consequence of the organizations deployed and in the methods used within the organizations.
2. **Documentation**. The case study shows that practitioners think that NFRs are often described in too vague terms.
3. **Management**. NFRs are often not sufficiently considered and prioritized and are sometimes even ignored.
4. **Test**. Most NFR types are difficult to test properly due to their nature, and when expressed in non-measurable terms testing is time-consuming or even impossible.

The remainder of this section is devoted to analyzing the suggested problems and corresponding suggested remedies. Since test problems mainly are related to documentation problems, stage 2 and 4 are analyzed together. Problems that are applicable to more than one phase are referred to as "General" in Table 1.

To be able to produce high-quality systems NFRs need to be sufficiently managed at all stages. An early incongruity is likely to propagate through the entire development process. Consider the following example from Ericsson (problems from Table 1 italicized):

Many NFRs are never discovered and *NFRs are often discovered too late* in the process, which causes an incomplete requirements set. The designers of the *too independent design units* get too much freedom even though they may lack the required system overview. This is likely to affect system architecture negatively which can result in poor runtime system properties.

Table 1. Experienced NFR-related problems

	Problem	Source
General	FR focus	Both
	Too independent design units	Ericsson
	Weak system design organization	Ericsson
	Limited knowledge of dependent systems	Ericsson
	Vague architectural responsibility	Ericsson
	Too far from customer	Ericsson
	Insufficient NFR knowledge	SMHI
	No NFR method corresponding to use cases	SMHI
	NFRs especially complicated in complex systems	SMHI
	Communication when several organizations interact	SMHI
Elicitation	NFRs discovered too late	Ericsson
	Many NFRs never discovered	Ericsson
	Users' real needs not in focus	Ericsson
	Difficult to elicit NFRs	SMHI
	Users do not know what they want	SMHI
	Users have no idea about NFR cost	SMHI
Doc and test	Vague NFRs	Both
	Difficult to quantify	Both
	Language problems	Ericsson
	Difficult to estimate cost and measures	Ericsson
	Difficult to test/verify	Both
	No clear rules how to verify NFRs	SMHI
Mgmt	Risk of misunderstanding NFRs	Ericsson
	Risk of ignoring NFRs	Ericsson
	Conflicting NFRs	SMHI
	Difficult to prioritize	SMHI

The most obvious and appealing remedy to many NFR-related problems would be to use methods and tools that address the problems in all stages. According to Kotonya and Sommerville [12] most existing RE methods do not adequately cover NFRs simply because it is very difficult to do so. Reasons are for instance that certain constraints are unknown at the requirements stage, that some constraints need very complex empirical evaluations to be determined and that NFRs tend to conflict each other. Furthermore, they argue that separating NFRs and FRs makes it difficult to see dependencies between them, whereas functional and non-functional considerations are difficult to separate if all requirements are stated together. Finally they claim that it is difficult to determine when NFRs are optimally met, since it is almost always possible to refine solutions.

4.2. Elicitation

Many NFRs are never discovered at Ericsson, probably due to the lack of interest for NFRs that are not system characteristics. Respondents' explanations are that the department is too far from the customers and that the users' real needs are not known. Furthermore, the vague architectural responsibility is a consequence of the weak system level design organization, which leaves many architectural considerations to the design units that do not have the required overview. The most effective solution to many of these problems, as suggested by several respondents, would be to strengthen the system level architectural design unit and make NFRs part of the agenda.

SMHI also find it difficult to elicit NFRs, but not primarily for the same reasons as Ericsson. Since SMHI develops new systems to known customers to a great extent, their elicitation problems are different. Well-known elicitation problems like "users do not know what they want" and "users have no idea about what NFRs cost" are applicable, and communicative problems when several organizations interact are also faced. In this situation it is vital to clearly understand the customer's problem domain. Respondents regard the interplay between different stakeholders as a key factor for success. Both NFRs and FRs benefit from close and regular contact with customers and stakeholders that are represented by the most appropriate individuals. With optimal composition of participants the RUP requirements workshops are helpful when eliciting NFRs. However, workshops concentrate primarily on use cases and FRs. One respondent proposed the use of scenarios as in SEI's Quality Attribute Workshops [1] to take care of quality aspects. The method consists of NFR-dedicated workshops and is built on scenarios that are used to analyze a system's architecture (or architecture alternatives) against a set of critical NFRs.

To effectively elicit NFRs, methods based on user concerns have been proposed (see [12] for a summary). Users' main concerns may be ease of use and preventing unauthorized access, corresponding to NFR types usability and security. The concerns can be broken down into NFRs and questions that need to be answered during the requirements engineering process. An attempt to provide a more comprehensive solution is the NFR Framework introduced by Chung et al. [7]. The framework is based on the same idea, but includes breaking down NFRs (so called softgoals) into FRs and even design decisions.

There is current research regarding elicitation techniques and quality requirements (i.e. NFRs) [10]. This extensive research project aims at assembling all known elicitation techniques, to find out why so few of them are used in practice and to assist analysts in determining which techniques are "right". However, the contribution is limited by existing elicitation techniques. Even if all existing elicitation techniques are considered inadequate, though, it is still interesting to know which one is best in different situations.

4.3. Management

The case study shows that primary NFR management problems are difficulties to prioritize NFRs, managing conflicting NFRs and the risk that NFRs are ignored. Generally, prioritization is a well-recognized part of requirements engineering [11]. At SMHI, NFRs are sometimes prioritized when starting a new project, and the types that are considered most important have been pointed out within the organization. However, some NFRs are conflicting due to their nature and trade-offs are often necessary [5, 15]. A suggested solution to minimize conflict-related problems, also recommended by Davis [8], is to focus on the most important NFR types. Again, user concerns may be useful when determining the most important NFR types. Furthermore, the NFR Framework and the SEI Quality Attribute Workshops mentioned in the previous section both provide means to deal with prioritization and conflicting NFRs to some extent; in the first case by a prioritization mechanism built into the framework and by special attention during the workshops in the latter case.

System characteristics are considered most important at Ericsson OSS and included NFR types gain most of the NFR attention. Limited time and competence often cause other NFRs to be ignored, since people do not realize that these NFRs add any value to the product. Thus, system characteristics have been implicitly prioritized simply because involved personnel understand them better and have an idea about how to deal with them. Therefore, both explicit NFR type trade-offs and improved NFR competence are needed to solve the problem.

Several respondents think that good overview is vital when managing NFRs and suggest many improvements; connections between NFRs and architectural decisions and between NFRs and design patterns should be more visible. Furthermore, architectural guidelines should be available. All this knowledge should be stored in some kind of knowledge base for reuse purposes in future projects and releases. We are, however, fully aware of the practical difficulties of setting up and maintaining such a knowledge base.

4.4. Documentation and test

Documenting and testing NFRs are closely related. In both cases, practitioners find it more difficult to describe the properties of a system than its functionality. This means that NFRs are often described in non-measurable terms and with vague wordings. Moreover, requirements are documented in English within Ericsson. However, the majority of the practitioners are native Swedish speakers and have difficulties being as precise in English as in Swedish. Altogether, NFRs are often vaguely stated and thus easy to misinterpret and difficult to test.

From a tester's perspective, things would be easier if NFRs were properly quantified. There are metrics [9] for most NFR types that can be used when describing NFRs (see [12] for an overview), e.g. "processed transactions per second" to quantify performance and "time to learn 80% of the facilities" to quantify usability. However, it is difficult for a requirements engineer or end user to define an appropriate level. At OSS the problem is sometimes attacked by using prototypes, while theoretical models to calculate measures and cost are rarely used and could be tried more. A somewhat different approach to the problem would be the concept of design space analysis [16]. Design decisions are taken based on comparisons with similar systems, and the approach could be used when determining, for instance, performance levels in requirements. However, respondents in both cases agree upon that testers should be involved when defining the requirements. SMHI has also reduced the problem by stating verification method to each requirement.

Even if NFRs are properly quantified, they might require a lot of time, effort and competence to test and verify. A good example is the usability measure above: "time to learn 80% of the facilities". However, performance is considered relatively easy to measure in both cases since good tools and methods are available. Still improvements can be made. One of the forcing requirements at OSS was that performance shall be at least as good in the next release as in the previous release, and verifying that would benefit from more cooperation with customers using the system. If possible, performing benchmarking at a customer's site would reveal what performance level that actually needs to be achieved [14].

4.5. Organization structure

As have been shown NFR-related problems are present in both cases. However, problems as well as reasons are somewhat different between the cases. The project-oriented IT department at SMHI faces mostly problems that are related to the vague nature of NFRs (e.g. NFR trade-offs), deficiencies in the used methods or individual lack of competence. Nevertheless, the moderate size of

projects and the availability of stakeholders facilitate the avoidance of other potential problems in a way that is not applicable at Ericsson.

Due to its size and structure Ericsson has to cope with another dimension of NFR-related problems. The large organizational hierarchy at Ericsson corresponds well to the structure of the system itself, i.e. a certain system module belongs to a specific department. This structure, although suitable for FRs, is somewhat problematic when dealing with NFRs. NFRs must be considered on a high enough hierarchical level in order to align all departments and design units and to be able to take in the whole picture when designing the system. The current situation is such, as several respondents claim, that the design units on the lowest hierarchical level are too autonomous and thus that NFRs can not be properly dealt with. These respondents think that NFR management would benefit from a stronger system level design unit making the lower level design units serve more like "code factories". However, system characteristics NFRs fit well enough into the structure, thanks to good understanding for those NFR types and a skilled and experienced integration and verification department. Nevertheless, NFR types like usability can hardly be successfully dealt with unless the decisions are moved upwards in the hierarchal structure. Such concerns must be considered on a higher level to have impact on all relevant departments and system parts. A trivial example is that user interfaces should be consistent across related system parts.

A less trivial example showing a problem with a functionally oriented system and organization structure is the potential conflict between building and using a system in the most efficient way. Consider an administrative system providing support for automated salary reports. Building the system might benefit from having one design unit responsible for the salary database, another design unit for the vacation database and a third design unit for the report generator. Thus, the organization is well suited for *functional* breakdown and allocation of requirements. In contrast, a user of the system will need the services of all these parts but cares of nothing but generating a report containing the desired data. In such a case, the design work will benefit from managing NFRs like usability on a higher hierarchical level.

5. Discussion

The influence of quality methods over technology has varied during the past decades. Quality problems forced the development of good methods and techniques and the interest for ISO certification, Cleanroom, CMM etc. was growing at Ericsson in the beginning of the 1990's. However, the use of methods became too extensive and more technology-oriented (incremental) approaches were introduced as a reaction in the second half of the decade. The trade-off between technology and quality methods today is, however, further complicated by the weak market (at least in the Ericsson case), and it is very hard to tell right now where the trend is heading.

The hierarchical structure as described in the previous section (see 4.5) is based on the assumption that the people in charge (system level) can use their market knowledge and provide the design units with correct directives. However, reality shows that these directives are often hard to interpret and hard to realize, primarily due to lack of technical knowledge on the system level. Thus, the system level is dependent on the design units and their technical skills. Incomplete or erroneous directives are likely to make the technical staff question and challenge the system level staff with opinions like "*they don't know what they talk about*". The situation that the design units ask the system level for directives and that the system level asks the design units for advice influences NFR aspects negatively. In a short perspective this problem must be solved by negotiations between involved units and the system level but in the long run significant knowledge transfer is needed. In a hierarchical organization like Ericsson it is crucial that interest, authority and competence of NFRs are moved as high in the hierarchy as possible. We believe that such knowledge transfer is necessary to make a hierarchical organization efficient for NFR aspects. Based on our earlier experience with Ericsson, we know that in crisis projects, certain "war rooms" are deployed. This means that people with all types of knowledge and authority are working together in physically the same office. It is amazing that this solution is saved only for crisis.

For organizations like SMHI that often elicit requirements directly from end users and customers, an obstacle is that customers must learn to express themselves in terms of requirements. It is a well-known problem (and experience at SMHI) that customers have a hard time articulating what they want, and we believe that NFRs are less in focus in this perspective too. Both the requirements engineer and the customer probably find it easier to reason in terms of features and FRs than in terms of system properties and NFRs.

RUP has influenced both organizations in the case study. Ericsson played an important part when developing RUP and SMHI evaluates it for regular use. The use case focus in RUP leads to FR focus in the software developing organization. The opening for NFR management is that RUP also point out architecture as important. We believe that the growing interest for architecture is beneficial for quality aspects, since architectural decisions have consequences on quality to a great extent. Here the Requirements Engineering community has a true challenge: to provide the techniques necessary for successful NFR management that can be understood and deployed by

software architects. Without good requirements, the NFR attention will vanish. Carlshamre and Rantzer [6] corroborates this when analyzing the work with deploying a usability-method in more than 100 projects, that things are not done unless they are stated among the requirements: "*… the smallest and single-most important piece of information is the requirement, and the requirements are what drives the whole project towards a complete product. The requirements are followed to the point; nothing much is done unless there is a requirement stating so*."

6. Conclusions and future work

In the introductory chapter of "Non-functional Requirements in Software Engineering", Chung et al. [7] claim that "*As far as software engineering practice is concerned, they [NFRs] are generally stated informally during requirements analysis, are often contradictory, difficult to enforce during software development and to validate, when the software system is ready for delivery*." This case study provides empirical evidence showing that the statement is congruent with industrial reality.

Even though there are several differences between the cases in the case study, we also believe that our main assumption has proven to be right: Software developing organizations are well aware of NFRs and NFR-related problems but their main focus is on functional requirements. Primarily, the reasons are that existing methods and tools focus on FRs, that practitioners are more used to describe systems in terms of functionality than in terms of system properties and, in the Ericsson case, that the hierarchical organization structure to some extent counteracts successful treatment of NFRs.

In more concrete terms, the two most tangible problems identified in the case study are that many NFRs remain undiscovered and that NFRs tend to be vaguely stated. Both problems partly derive from the nature of NFRs but can be reduced by increased efforts and competence. However, a more general remedy would be to provide methods and tools that support NFRs throughout the entire development process. As of today, such processes and tools do not exist to a great extent, and the area can be further investigated in future research projects.

Some of the elicitation problems at SMHI are mitigated by the presence of customers and end users. In contrast, at Ericsson, where direct user involvement is rare, many NFRs remain undiscovered due to the hierarchical structure combined with too much power on a low hierarchical level. The overall constraints mentioned in section 3.1 are good examples showing that guidelines provided from system level have impact on all subsystems. We believe that the only chance to perform successful treatment of NFRs in a large hierarchical organization is that even other types of NFRs are considered on the system level. Thus, for large systems, overall quality may be a function of organizational power structures.

Acknowledgements

The authors would like to thank the anonymous respondents for participating. This project was funded by the Swedish Foundation for Strategic Research and SMHI.

References

[1] Barbacci, M. R., R. Ellison, A. J. Lattanze, J. A. Stafford, C. B. Weinstock and W. G. Wood. *Quality Attribute Workshops, 2nd Edition* (CMU/SEI-2002-TR-019). Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2002.
[2] Basili, V. R. and J. D. Musa. "The Future Engineering of Software: A Management Perspective", *IEEE Computer*, Vol. 24, No 9, pp. 90-96, September 1991.
[3] Boehm, B., J. R. Brown, H. Kaspar, M. Lipow, G. J. Macleod and M. J. Merrit. *Characteristics of Software Quality*, North-Holland, Amsterdam, 1978.
[4] Boehm, B., J. R. Brown and M. Lipow. "Quantitative Evaluation of Software Quality", In *Proceedings of 2nd International Conference on Software Engineering*, San Fransisco, pp 592-605, 1976.
[5] Boehm, B. and H. In. "Identifying Quality-Requirement Conflicts", *IEEE Software*, Vol. 13, No 2, pp. 25-35, March 1996.
[6] Carlshamre, P. and M. Rantzer. "Dissemination of Usability: Failure of a Success Story", *ACM Interactions*, Vol. 8, No 1, pp. 31-41, Jan/Feb 2001.
[7] Chung, L., B. A. Nixon, E. Yu and J. Mylopoulos. *Non-functional Requirements in Software Engineering*, Kluwer Academic Publishers, Boston, 2000.
[8] Davis, A. M. *Software Requirements: Objects, Functions and States*, Prentice Hall, Upper Saddle River, NJ, 1993.
[9] Fenton, N. E. and S. L. Pfleeger. *Software Metrics: A Rigorous and Practical Approach*, Second ed, PWS Publishing Company, Boston, 1997.
[10] Hickey, A. M. and A. M. Davis. "The Role of Requirements Elicitation Techniques in Achieving Software Quality," in *REFSQ'02*. Essen, Germany, 2002.
[11] Karlsson, J. and K. Ryan. "A Cost-value Approach for Prioritizing Requirements", *IEEE Software*, Vol. 14, No 5, pp. 67-74, Sep/Oct 1997.
[12] Kotonya, G. and I. Sommerville. *Requirements Engineering: Processes and Techniques*, John Wiley & Sons, Chichester, 1998.
[13] Kruchten, P. *The Rational Unified Process: An Introduction*, Second ed, Addison-Wesley, Reading, MA, 2000.
[14] Moe, J. and D. Carr. "Using Execution Trace Data to Improve Distributed Systems", *Software - Practice & Experience*, Vol. 32, No 9, pp. 889-906, July 2002.
[15] Wiegers, K. E. *Software Requirements*, Microsoft Press, Redmond, 1999.
[16] Åberg, J. *Live Help Systems: An Approach to Intelligent Help for Web Information Systems*, Doctoral thesis, Dissertation No. 745, Linköpings universitet, 2002.