Non-Functional Requirements Engineering - Quality is essential

Invited Anniversary Paper

Barbara Paech[1], Daniel Kerkow[2]

[1] Institute for Computer Science, University of Heidelberg,
69120 Heidelberg, Germany
Paech@informatik.uni-heidelberg.de
[2] Fraunhofer Institute Experimental Software Engineering,
67661 Kaiserslautern, Germany
Kerkow@iese.fraunhofer.de

Abstract. *The purpose of this paper is to review the state-of-the-art in the engineering of NFR, and to define an agenda for future NFR research. Therefore we define the requirements on a NFR method, compare this with current approaches and sketch ideas how to fill the gap between the current methods and our requirements. The main challenge for future research in our view is to improve the understanding of the notion quality.*

1 Introduction

Since the early nineties requirements engineering (RE) has matured from a discipline focusing mainly on the specification of formal requirements to a discipline which constitutes the foundation for the whole software development process.

A major step in this development has been the wide-spread usage of business processes, scenarios and use cases in dealing with functional requirements. This trend had been initiated by a general focus on processes, particularly in economy.

In contrast to this, non-functional requirements (NFR) are still poorly understood. This is exemplified by the superficial treatment of NFR in all major RE text books. The only comprehensive NFR approach is the NFR framework described in [11]. This situation is particularly unsatisfying as today's development processes put particular emphasis on quality. Neglecting NFR is counted as one of the top ten risks of requirements engineering [41]. Given the high competition on the market, companies need to thoroughly understand and meet the quality needs of their customers. Component developers and buyers need to understand and negotiate the quality of the components in different contexts. As the society as a whole depends more and more on information technology (IT), its need to rely on the quality of the technology is growing dramatically.

The purpose of this paper is to review the state-of-the-art in the engineering of NFR, and to define an agenda for future NFR research. The review of the state-of-the-art is split in two parts. First we identify criteria for evaluating the state-of-the-art, then we evaluate the available literature according to the requirements. For the former we have identified requirements on a NFR method. We call this ideal NFR method *Non-functional Requirements Engineering (NFRE for short).*

In the next section we sketch the important terminology. Then we present the NFRE requirements, namely how we want to work with NFR in the future. In section 4 we discuss what we can do with NFR today, that is the state-of-the-art of NFR. Section 5 discusses how to fill the gap between presence and future, suggesting a research agenda.

2 Terminology

The term "non-functional requirement" is used to delineate requirements focusing on "how good" software does something as opposed to the functional requirements, which focus on "what" the software does. As exemplified already in the session summary on NFR at the first REFSQ, 1994, there are diverging views whether it is possible to make this delineation or not, as NFR may become functional through refinement. In the following we define NFR as any requirement describing the quality of the system.

The notion of quality is made precise in several newly evolving standards on quality, notably ISO/IEC 9126-1:2001[31]. In this standard, 4 types of quality levels are distinguished:

Quality in use refers to the perception of software *quality attributes (QA)* through specific users in specific contexts. This comprises the use of the software product to achieve the user goals with the following attributes

- Effectiveness: accuracy and completeness
- Productivity: resources appropriate in relation to effectiveness
- Safety: acceptable levels of risk or harm to people, business, software, property or environment
- Satisfaction.

External and internal quality corresponds to QA of the software as a black box or as a white box. This comprises

- Efficiency: appropriate performance, relative to the amount of resources used, under stated conditions.
- Portability: transfer from one environment to another.
- Maintainability: modifications may include corrections, improvements or adaptations of the software products to changes in environment, and in functional specifications.
- Functionality[1]: functions, which meet stated and implied needs when the software is used under specified conditions.
- Usability: understandability, learnability, operability, attractiveness.
- Reliability: maintainance of a specified level of performance when used under specified conditions.

Process quality refers to the quality of the product development process and thus influences all the other quality attributes.

[1] Functionality is, in fact, the label of a set of NFR in the ISO standard. It is defined by the sub quality attributes accuracy, compliance, interoperability, suitability and security.

NFR specify the quality of the software on any of these 4 levels. We distinguish a concrete NFR from the QA it belongs to. For example, the NFR “The database of our new system shall handle 1000 queries per second.” belongs to the QA “performance” or more specifically the QA “workload of database”. In the NFR-framework QA are called *NFR-type*.

As NFR characterize the software product as a whole, they heavily depend on the functional and architectural characteristics of the software. *Functional requirements* (FR) are concerned with *tasks* performed by the user, by the system or by both in an interaction. *Architectural requirements* (AR) constrain the physical or logical structure of the system.

3 Criteria for evaluating the state-of-the-art

In this section we define the requirements on NFRE. For their identification we have used the task-oriented requirements engineering method TORE developed at Fh IESE [48]. This method is designed to deal with requirements on IT-systems and the related business and user processes. Thus, we view NFRE as an IT-based process to support different stakeholders dealing with NFR.

TORE describes the decisions to be made when defining functional requirements. It does not prescribe any particular representation for these decisions. The following decisions are distinguished:

- The first decisions concern the stakeholders to be supported and their tasks (*task level*).
- This is followed by the decision on the to-be-activities of these tasks, the relevant domain concepts and the decision which of these tasks and concepts should be supported by IT (*domain level).*
- Furthermore, the interaction between users and the system has to be defined: in particular the system functions, the user interface concept in terms of data exchanged in the interaction and the grouping of the data and system function in the user interface (*interaction level*).
- On the last level the system details, namely the system architecture, internal functions, and the user interface details, have to be decided (*application core and GUI level*).

In this paper we only concentrate on the task, domain and part of the interaction level. These define the features (functions and concepts) of NFRE and how they are derived from user tasks and to-be-activities.

Table 1 shows the method features and the quality aspects of the features we have identified by looking at the different stakeholders, their tasks and to-be-activities. We have not elicited these from real stakeholders, but from our own experience in and with the different roles.

Table 1: NFRE method features

	method feature	method feature quality	to-be-activity	user task
1	support identification of quality aspects in a specific context	completeness, correct- ness, creativity,standards	specify NFR	manage, use
2	support identification of NFR relative to given FR, AR	completeness, correct- ness,creativity, standard, adequate level of detail, no premature design decisions, feasability	elicit NFR,AR,FR	spec
3	support identification of GUI relevant NFR	completeness, correctness,standards	design GUI	spec
4	identify conflicts	reflection of means	elicit NFR,AR,FR	spec
5	support negotiation of conflicts	explicit rationale	elicit NFR,AR,FR	spec
6	support prioritization	explicit rationale	prioritize NFR	spec
7	identify dependencies	completeness	document NFR,AR,FR	spec
8	support to make dependencies between AR,NFR,FR explicit	good visualization, overview	manage change of AR, NFR, FR, estimate and im- plement change	spec, change
9	support documentation	adequate structure, standards	document NFR, AR, FR	spec
10	support wording	adequate precision, standard terminology	document NFR, AR, FR	spec
11	support identification of defects in NFR spec.	IEEE830 quality criteria	inspect NFR, AR, FR	spec
12	support identification of metrics	standards, adequate precision	define test cases for NFR	V&V
13	support identification of means and patterns	completeness, correct- ness, creativity, standards	design detailed architecture	design
14	support evaluation of means (architecture, functional RE) against a set of NFR	good visualization, overview	document NFR, AR, FR, design detailed architecture	spec , design
15	support discussion and documentation of different options	explicit rationale	document NFR, AR, FR	spec
16	support trade-off decisions	explicit rationale	design detailed architecture	design
17	support cost estimation of NFR	correctness	cost estimation	manage
18	support to estimate change impact within requirements	completeness, correct- ness, good visualization, overview	manage change of NFR, AR, FR	spec
19	support to estimate change impact on the whole system	completeness, correctness, good visualization, overview	estimate and implement change	change
20	support controling the status of NFR	completeness	project controlling	manage

The roles that have a stake on the NFRE method are all roles involved in software engineering. The following list sketches the roles and their main tasks:

- "customers" carry out the task "buy or procure system", "product and project management" carry out the task "manage". We subsume both under (manage) as their needs concerning NFR are roughly similar.
- "users" carry out the task "use system" (use)
- "requirements engineers" carry out the task "specify system" (spec)
- "designers" carry out the task "design system" (design)
- "testers" carry out the task "verify system against requirements " (V&V)
- "maintainers" carry out the task "maintain system" (change).

Only the activities dealing with NFR are described. The method features describe how the method should support the activities. They are of different granularity and precision. The wording "support for…" indicates that it is not quite clear what the features should look like, while e.g. "identify conflicts" requests a method feature to identify conflicts between given NFR. In describing the method features we have used the term *means* for any possible solution (FR, QR or AR) to achieve a specific NFR. In the NFR-framework this is called operalization.

The quality aspects capture aspects which are not inherent in the feature, but important to achieve a high-quality method. First, they consider standards and experience, as NFR are very context dependent and therefore heuristics need to be applied. The other important quality aspect is to achieve specific quality of the requirements dealt with in the feature (e.g. completeness and consistency). This is due to the fact that NFR are typically very vague (often due to their global nature) and interdependent (again due to their global nature).We will go into the details of the features in the next section when we discuss the state-of-the-art concerning the features.

4 State of the art in NFR engineering

In this section we give a short survey on NFR literature published at all RE-specific conferences and journals, in particular REFSQ, IEEE and ACM RE-conferences, Springer RE Journal and IEEE Software.

Quality has been a focus of REFSQ, since its start in 1994. REFSQ'94 had a whole section on NFR. This contained methods for capturing specific QA like performance or security, and some general statements on NFR [45][37][27]. REFSQ'95 did not present any specific NFR paper. In REFSQ'97 a classification of NFR was presented [26]. REFSQ'98 saw papers on further specific QA like usability, an industry statement on the need for NFR approaches, and an approach to use scenarios to elaborate NFR [40][39][549. There were no specific NFR papers in REFSQ'99 and REFSQ'01. In REFSQ'00 an extension of the NFR framework to incorporate patterns was presented [23]. In REFSQ'02 the authors presented a position paper emphasizing the importance of architecture in NFR considerations [46]. REFSQ'03 saw again a session on NFR dealing with specific QA [16][25][53]. So altogether NFR have been a topic almost continuously, but mostly on the level of individual QA. In the following we do not consider the specific QA, but only the general NFRE features elicited in the last section.

The features in Table 1 can be roughly grouped into

- The identification of NFR from different viewpoints and different levels of detail (1-3)
- The support for uncovering dependencies and conflicts between them, and to discuss and prioritize them accordingly (4-8).
- The documentation of NFR and the evaluation of this documentation (9-12).
- The support for identifying means to satisfy the NFR, to evaluate and discuss means, and to make trade-off decision accordingly. This includes cost estimation (13-17)
- The support for change and project management (18-20).

4.1. Identification

The identification of NFR involves elicitation from the different stakeholders. Here, of course general requirements elicitation techniques like workshops and interviews can be used. These methods should emphasize creativity, as suggested in [43]. There are three main difficulties specific for NFR:

- Stakeholders perceive the quality of a product differently, e.g. the QA usability has very different connotations. Many different aspects should be considered.
- Typically, NFR are expressed on a very high-level only, and must be refined. During the refinement it is important to avoid premature design decisions.
- NFR can often only be stated in relation to given FR and AR.

The first issue can be dealt with by methods focusing on particular *stakeholders* and supporting *context rich* descriptions of NFR. Scenarios as proposed in [54] are very helpful for this purpose. [2] gives detailed hints which elicitation technique is useful for which QA. The first and the second issue can be supported by *collecting knowledge* on the different QA. This knowledge can be used as a checklist during the elicitation. A first step in this direction was [4]. The NFR-framework includes catalogues for the major QA. They have been refined in the EMPRESS project [15]. A practical template asking for specific NFR is VOLERE [57]. Another important help is the identification of *typical kinds of refinements* for NFR. These can be specific to a QA, generic or even specific for the project. Again the NFR-framework offers catalogues of typical refinement methods. These catalogues also help to think abstractly and avoid premature design decisions. The third difficulty requires an *iterative* approach to NFR identification. NFR are identified based on the given FR and AR. During this identification process typically new FR and AR arise, either because of the general progress of the requirements process or because identified NFR give rise to new FR and AR. An example for such interdependencies is given in [46]. Another example is given in [49] which argues for early architecture consideration similar to trade-off-analysis in systems engineering. Based on the new insights concerning FR and AR, new NFR have to be identified. [13] describes an approach that combines NFR and use cases. Use cases and NFR are first elicited separately and then combined to make sure that the use cases satisfy the NFR.

Another reason for iteration is the prioritization of specific NFR and the consideration of different means for achieving the NFR. This is discussed in the following sections.

4.2. Negotiation

Like other requirements, NFR elicited from different stakeholders often are in conflict with each other. Thus, it is important

- to make these conflicts explicit. This should include derived dependencies.
- to make decisions based on the rank of priorities. The rationale for these decisions should be explicit. Prioritization is enhanced through good overview of the dependencies.

The WinWin-approach offers NFR-specific support for the latter [8][29] in terms of a tool which alleviates *group decisions*. As described in [19][30] the resolution of such conflicts typically involves the *reflection of architectural options and other means*. This is discussed in the following sections. To make the conflicts explicit, *dependencies* need to be made explicit and evaluated. The NFR-framework uses specialized goal graphs, called *softgoal interdependence graph*, to describe dependencies. They capture positive and negative influences between NFR together with the corresponding argumentation. Again to ensure completeness, *knowledge* on typical influences between QA should be collected. This approach has been enhanced with conceptual models which capture the terminology common to the different softgoals [12]. The conceptual model helps to detect dependencies.

Aspect-oriented approaches also try to analyze the dependencies between FR and so called concerns in more detail [44][50]. They use *matrices* to make the dependencies between NFR and from NFR to FR and AR explicit. In addition they present a detailed catalogue of types of interdependencies.

Based on the explicit description of dependencies between a few NFR, one can use *algorithms and tools* such as QARCC [8] or the evaluation procedure of the NFR-framework to detect more dependencies.

4.3. Documentation

Because of complexity and longevity, requirements need to be documented. This is particularly true for NFR. The main issues here are

- to integrate NFR into requirements documents as used in industry
- to state NFR precisely.

The first can be dealt with by providing *templates and rules* on where to capture NFR. An example for this is [16] and [36] which describe how to elicit and document precise NFR together with use cases and architectural descriptions. The goal graphs proposed by the NFR framework support the documentation of dependencies, but do not give advice on how to integrate the NFR in a full-fledged requirements documentation. Furthermore, no advice for the respective wording is given. To the formulation of specific NFR the *usual quality criteria* on requirements documents (see IEEE-830 [28]) apply. To achieve precision, metrics should be used for different QA. The EMPRESS-project has collected the available knowledge on metrics in [15]. These can also be used as checklists during inspection. In the context of architecture evaluation the formulation of NFR as scenarios has been advocated [34]. These scenarios, however, are only one-sentence descriptions, and thus correspond to usual textual requirements. In addition preliminary approaches to a formal specification of NFR exist [51].

4.4. Identification and Evaluation of Means

A major step in the understanding of NFR *is the distinction between the NFR and means to achieving it*. A means describes how to achieve a specific NFR. So e.g. performance can be enhanced through better hardware or better software structures. This distinction is called phenotype vs. genotype in [14]. The means can either be further FR (e.g. authorization as a means to achieve security) or AR (e.g. a second data storage for reliability), and they can also relate to the development process (e.g. maintainability can be supported by regular code reviews). It is important to bring in means as late as possible because otherwise specific solutions are introduced too early. This typically leads to a suboptimal solution where only few NFR, FR and AR can be satisfied. Typical means can only be described on the basis of a high-level software architecture. There are three major issues in handling means:

- identification of possible means
- evaluating the NFR compared to the means
- discussing and deciding the trade-offs between several means. This should include cost considerations.

The identification requires considerable amount of *creativity*, but can also be supported by *experience*. As means often refer to architecture, *patterns* are a good way to describe them [23]. One problem is that there is little agreement on the description of architecture. General approaches are collected in [52][1][5][6]. Further possibilities are, e.g., use case maps [429, agent-oriented goal graphs [24], the CBSP approach [19], or social organizations [38]. Thus, there is no general scheme of describing architectural means.

Similarly, there is no standardized approach to evaluate an architecture against NFR. The most advanced method is from SEI: The Architecture Tradeoff Analysis Method (ATAM) captures criteria (quality attributes, business goals), issues (risks), options (architectural views), and assessments (utility tree) [35]. The Cost Benefit Analysis Method (CBAM) is used to refine the ATAM results with cost, benefit (criteria, options) [30] .

The patterns, in particular, should include the positive and negative influences of the means on different QA. In the NFR framework this is handled similarly to the overall dependencies between NFR. Both are captured in the goal graphs together with their arguments. While this is effective for a machine-based evaluation, we believe that humans need better visualizations for the discussion and evaluation of different means. In [47] we propose to use general *matrices for questions, option and criteria*. This is adapted from general rational management methods [17]. Other approaches advocate the use of general *multi-criteria decision methods* such as AHP for finding a set of means best suited to satisfy a given set of NFR [55].

As for dependencies, tools can be used to compute aggregations based on explicitly described evaluations. Again, iteration is very important, as the decision for a specific (architectural) means might induce further NFR and FR.

4.5. Change and project management

As discussed above the specification of NFR is inherently iterative. Therefore, the identified NFR, FR and AR may change in the process. Especially, if a change is triggered by a major change in the system context, it is important that the impact of a change on a current set of requirements can be specified explicitly. For project and change management of NFR the usual methods can be used. There are almost no change approaches specific to NFR. In the NFR framework changes are captured with special labels in the goal graphs [10]. While these changes are simple to perform, we again feel that this is not adequate for human reasoning as the graphs get very complicated.

4.6. Summary

The discussion above has shown that many approaches exist to deal with the many aspects of NFR. All issues have been approached at least in some way. Still, it is the fact that these approaches have not yet been established as standards, e.g. in RE textbooks. In particular, we see the following open questions:

Identification. It seems to be inherent to human requirements negotiation that it is easier to state requirements in terms of concrete FR and AR than in terms of QA. Thus, it is important to better understand and cope with this phenomenon. Furthermore, we have to take into account the diversity of quality concepts and the necessity to understand the context of a specific quality concept. Quality is relative to the person expecting it and to the possibilities to implement it.

Negotiation. We need to deal with complex dependencies (not only bilateral ones). The current algorithms treat the definition of a consistent set of NFR as a configuration problem, similar to feature-oriented approaches in the area of product-lines. However, because of their subjective nature, this is not so easy for NFR as the priority and even the definition of a specific NFR might change when confronted with several other NFR. So it seems important to provide support for exploration of the consequences of the combination of different NFR.

Documentation and wording. Existing approaches are often not yet suitable for application in industry as they require very specific notations and tools not integrated into the mainstream RE-tools. This is particular true for the notation of goal graphs and the evaluation tools. [18] and [32] describe in detail how NFR are handled at Alcatel Telecom and Ericsson. These companies mainly apply general methods for requirements and quality management and in particular make use of knowledge from earlier projects. They do not apply any specific notation or tool. Furthermore, we need a systematic approach and a standardized vocabulary for quality issues. The standards are not detailed enough. In particular, the different levels of quality have to be made more precise. In addition, specific documentation styles for different QA are needed which are best suited for the communication with the users. Sub-communities, such as security, came up with specific styles such as mis-use-cases [3][20]. For usability, prototypes are very helpful.

Means and trade-off-decisions. There is still no specific support to evaluate how well a given means satisfies a set of NFR. The algorithms such as [55] proposed for aggregating and propagating individual evaluations during decision-making are only as good as the initial evaluations. For the initial evaluation (i.e. how well a specific means satisfies a specific NFR), experience (personal or organizational in terms of e.g. patterns) is very important. We also feel that it is necessary to better visualize architectural options and their relevance to specific NFR. This requires to visualize different aspects of an architecture and to break down a specific measurable NFR into smaller parts belonging to different parts of the architecture. This would in particular, help the designers, because they get more specific guidelines on how to choose the right means.

Change and project management. The great disadvantage of the traceability approaches is their need for a complete conceptual model of all specification elements. In NFR research we are far away from knowing every type of NFR and the relationships between these types. Another challenge is the intertwined nature of NFR. The current traceability approaches are based on a set of discrete elements that are related to each other. NFR, are rather crosscutting aspects that have an interweaving impact on different requirements.

5 How to fill the gap

Looking at the summary of the last section it is evident that future NFR research should not so much focus on methods for systematic treatment of NFR (as important parts already have been provided, e.g. through the NFR-framework). It seems to be much more urgent *to get hold of the notion of quality*. This would provide a common ground to talk about quality in different contexts. And then the systematic treatment can help to manage this common ground within different projects.

To focus on quality as such we see different research directions:

Perform more empirical research.

Empirical studies on how NFR impact the performance of projects could help to better understand how NFR evolve during projects. How are the schedules impacted by wrong or imprecise NFR? How is the satisfaction of all process participants impacted by wrong or imprecise NFR? This data could also be used to collect experiences on how well specific NFR combine, which development costs are induced by specific NFR and so on.

Understand subjective vs. objective quality

As a starting point, speech-psychology and its methods (card sorting, cluster analysis or semantic differentials[22][33][56][9]) can be applied to create common ground and to standardize semantic spaces. Furthermore, one can apply systematic construct operationalization as for example GQM (goal/question/metric).

Apply ethnographical methods

Ethnographic methods have been successfully used to understand different contexts of IT-technology-development [7][21]. They could specifically be used to explore the understanding of quality in these contexts. These methods could on the one hand produce general insights on quality. On the other hand these methods could be packaged for requirements engineers so that they can use them for elicitation in a particular project.

Apply graphical design methods

Visualization of NFR, means and their dependencies have so far concentrated on typical computer science structures such as graphs. They are well-suited for tools, but do not support humans very well in thinking creatively. It would be interesting to investigate graphic design methods which take into account human characteristics such as cognition.

Acknowledgements

We thank our colleagues for fruitful discussion within the IESE-EMPRESS-Team. We acknowledge the ITEA project EMPRESS for partly funding our research.

References

1. Abowd, G., Bass, L., Clements, P., Kazman, R., Northrop, L. & Zaremski, A.. "Recommended Best Industrial Practice for Software Architecture Evaluation". CMU/SEI-96-TR-025, Software Engineering Institute, Carnegie Mellon University, 1997
2. Andreou, A.S., "Promoting software quality through a human, social and organizational requirements elicitation process", Requirements Engineering Journal, no.8, pp85-101, 2003
3. Alexander, I., "Misuse Case Help To Elicit Nonfunctional Requirements", Institution of Electrical Engineers (IEE) Computing & Control Engineering Journal (CCEJ), 2001
4. Barbacci, M., Klein, M.H., Longstaff, Th.A., Weinstock, Ch.B., "Quality attributes", CMU/SEI report CMU/SEI-95-TR-021, 1995
5. Barbacci, M. R., Klein, M. H. & Weinstock, C. B. "Principles for Evaluating the Quality Attributes of a Software Architecture". CMU/SEI-96-TR-036, Software Engineering Institute, Carnegie Mellon University, 1997
6. Bass, L., Clements, P. & Kazman, R. "Software Architecture in Practice". Addison-Wesley, 1998
7. Blomberg, J. et al., Ethnographic Field Methods and Their Relation to Design, Participatory Design: Principles and Practices, from CPSR First Participatory Design Conference 1990, pp. 123-155, 1993.
8. Boehm, B., In, H., "Identifying quality requirements conflicts", IEEE Software, March 1996
9. Bowker, G.C., and Star, S.L. (1999). Sorting things out: Classification and practice. Cambridge MA USA: MIT Press.
10. Chung, L., Nixon, B.A., Ye, E., „Using non-functional requirements to systematically support change", ISRE'95, pp. 132-139, 1995
11. Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J.,"Non-Functional Requirements in Software Engineering", Kluwer Academic Publishers, 2000
12. Cysneiros, L.N., Leite, J.C.S.P, "A Framework for Integrating Non-Functional Requirements into conceptual models", Requirements Engineering Journal, no. 6, pp. 97-115, 2001
13. Cysneiros, L.N., Leite, J.C.S.P, "Driving Non-Functional Requirements to Use Cases and Scenarios", XV Brazilian Symposium on Software Engineering, 2001
14. Davis, A.,M., "System phenotypes", IEEE software, july/august 2003, pp. 54-56, 2003
15. Dörr, J., Punter, T., Bayer, J., Kerkow, D., Kolb R., König, T., Olsson, Th., Trendowicz, A., „Quality Models for Non-functional Requirements", IESE_Report, Nr. 010-04/E, 2004
16. Dörr, J., Kerkow, D., von Knethen, A., Paech, B., „Eliciting efficiency requirements with use cases", REFSQ'03, Essener Informatik Beiträge,, Band 8, pp.37-46, 2003
17. Dutoit, A. H., B. Paech, P., "Rationale Management in Software Engineering. In: S.K. Chang (Ed.), "Handbook of Software Engineering and Knowledge Engineering. World Scientific, December 2001.

18. Ebert, Ch. “Putting requirements management into praxis: dealing with non-functional requirements”, Information and Software technology, no. 40, pp. 175-185, 1998
19. Egyed, A., Grünbacher, P., Medvidovic, N., “Refinement and evolution issues in bridging requirements and architecture – the CBSP approach”, From Software Requirements to Architectures (STRAW) Workshop held at ICSE 2001
20. Firesmith, D., “Security Use Cases”, in Journal of Object Technology, vol. 2, no. 3, May-June 2003, pp. 53-64.
21. Goguen J.A. & Linde, C., Techniques for Requirements Elicitation, Proceedings of IEEE International Symposium on Requirements Engineering, p. 152-64, January 1993.
22. Gordon, A.D., "Classification, Second Edition", Chapman & Hall/CRC, 1999
23. Gross, D., Yu, E., “From non-functional requirements to design through patterns”, REFSQ’00, Essener Informatik Beiträge,, Band 5, pp.86-98, 2000
24. Gross, D., Yu, E., “Evolving system architecture to meet changing business goals: an agent and goal-oriented approach”, From Software Requirements to Architectures (STRAW) Workshop held at ICSE 2001
25. He, Qu., Anton, A.I., „A framework for modeling privacy requirements in role engineering”, REFSQ’03, Essener Informatik Beiträge,, Band 8, pp.137-146, 2003
26. E. Hochmüller, “Requirements Classification as a first step to grasp quality requirements”, REFSQ’97, Presses universitaeires de Namur, pp. 133-144, 1997
27. Hofmann, H., F., Holbein, R., „Seven Ways to quality: a framework for specifying security requirements“, REFSQ’94, ‚Aachener Beiträge zur Informatik, Band 6, pp. 45-54, 1994
28. IEEE Recommended Practice for Software Requirements Specifications, IEEE Std. 830-1998
29. In, H., Boehm, B.W., Rodgers, T., Deutsch, W., "Applying WinWin to Quality Requirements: A Case Study", ICSE 2001, pp. 555-564, 2001
30. In, H., Kazman, R., Olson, D., “From requirements negotiation to software architectural decisions”, From Software Requirements to Architectures (STRAW) Workshop held at ICSE 2001
31. ISO/IEC 9126-1:2001(E), “Software Engineering - Product Quality - Part 1: Quality Model”, 2001
32. Jacobs, St., “Introducing measurable quality requirements: a case study”, RE’99, pp. 172-179, 1999
33. Kahneman, D., "The semantic differential and the structure of inferences among attributes". American Journal of Psychology, 76, 554-567, 1963.
34. Kazman, R., Abowd, G., Bass, L. & Clements, P. “Scenario-Based Analysis of Software Architecture”. IEEE Software, November , pp.47-55, 1999
35. Kazman, R., Klein, M. & Clements, P. “ATAM: Method for Architecture Evaluation”. CMU/SEI-2000-TR-004, Software Eng. Inst., Carnegie Mellon University, 2000
36. Kerkow, D., Dörr, J., Paech, B., Olsson, Th., König, T., “Elicitation and documentation of non-functional requirements for socio-technical systems”, in Silva, A., Mate, J.L., (eds.) Requirements Engineering for Socio-Technical Systems, to appear 2004
37. Kirner, T.G., Davis, A.M., „Timing-Constraints for requirements specification of hard real-time systems“, REFSQ’94, Aachener Beiträge zur Informatik, Band 6, pp. 33-45, 1994
38. Kolp, M., Castro, J., Mylopoulos, J., “A social organization perspective to software architectures”, From Software Requirements to Architectures (STRAW) Workshop held at ICSE 2001
39. Landes, D., “Requirements Engineering for quality requirements”, REFSQ’98, Presses universitaeires de Namur, pp. 185-186, 1998
40. Lauesen, S., Younessi, H., “Six styles for usability requirements”, REFSQ’98, Presses universitaeires de Namur, pp. 155-166, 1998
41. Lawrence, B., Wiegers, K, Ch. Ebert, “The top ten risks of requirements engineering”, IEEE Software, November/December, pp. 62-63, 2001

42. Liu, L., Yu, E., “From requirements to architectural design – using goals and scenarios”, From Software Requirements to Architectures (STRAW) Workshop held at ICSE 2001
43. Maiden, N., Gizikis, A., “Where do requirements come from?”, IEEE Software, September/October, pp.10-12, 2001
44. Moreira, A., Brito, I., Araújo, J., "A Requirements Model for Quality Attributes", Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design, workshop at AOSD 2002
45. Opdahl, A, “Requirements Engineering for Software performance”, REFSQ’94, Aachener Beiträge zur Informatik, Band 6, pp. 16-32, 1994
46. Paech, B., Dutoit, A., Kerkow, D., von Knethen, A.: „Functional requirements, non-functional requirements and architecture specification cannot be separated – A position paper”, REFSQ’2002
47. Paech, B., von Knethen, A., Doerr, J., Bayer, J., Kerkow, D., Kolb, R., Trendowicz, A., Punter, T., Dutoit, A., „An experience based approach for integrating archiecture and requirements engineering “,From Software Requirements to Architectures (STRAW) Workshop held at ICSE, May 2003
48. Paech, B & Kohler, K “Task-driven Requirements in object-oriented Development.” Leite, J. & Doorn, J. (eds.). Perspectives on RE. Kluwer Academic Publishers, 2003
49. Pasternak, T., “Using Trade-Off Analysis to uncover links between functional and non-functional requirements in use-case analysis”, Int. con. On software-science, technology and engineering, IEEE, 2003
50. Rashid, A., Moreira, A., Araujo, J., “Modularisation and composition of aspectual requirements”, Int. Conf. on aspect-oriented software development, pp. 11-20, IEEE2003
51. Rosa, N.S., Cunha, P.R.F. “Process-NFL: a language for describing non-functional properties”, Int. conf. HICSS, IEEE; 2002
52. Shaw, M., Garlan, D., “Software Architecture – Perspectives on an emerging discipline”, ISBN: 0131829572, Prentice Hall, 1996
53. Sindre, G., Firesmith, D., Opdahl, A., “A reuse-based approach to determining security requirements”, REFSQ’03, Essener Informatik Beiträge,, Band 8, pp.127-136, 2003
54. Sutcliffe, A. & Minocha, S. “Scenario-based Analysis of Non-Functional Requirements”, REFSQ’98, Presses universitaeires de Namur, pp. 219-234, 1998
55. Svahnberg, M., Wohlin, C., Lundberg, L., Mattson, M., „A method for understanding quality attributes in software architecture structures“, SEKE’02, pp.819-826, 2002
56. Tudor, L.G., Muller, M.J., Dayton, T., and Root, R.W. (1993). A participatory design technique for high-level task analysis, critique, and redesign: The CARD method. In Proceedings of HFES’93. Seattle WA USA.
57. VOLERE-Template, Atlantic System Guild, http://www.atlsysguild.com