Workshop Summary
First International Workshop on Requirements Engineering: Foundation of Software Quality (REFSQ'94)[1]

Klaus Pohl, Gernot Starke, Peter Peters
RWTH-Aachen, Informatik V,
52056 Aachen, Germany
{pohl, peters}@informatik.rwth-aachen.de

1 Introduction and Workshop Structure

As achieving high quality means the realization of customers needs, requirements engineering (RE) is the most crucial phase within software development. In the RE process not only the functional requirements but also the so-called 'non-functional' or 'quality' requirements of the planned software system have to be elicited from the customer and represented in a requirements document in order to provide the software designer a complete and correct specification. Conventional RE methods usually s=mupport only parts of this process or help stating only specific kinds of requirements.

These methodological problems were the prime motivation for the REFSQ '94 workshop held in conjunction with the CAiSE '94 Conference on Advanced Information Systems Engineering in Utrecht, The Netherlands on June 6th and 7th 1994. In order to find solutions which handle the described deficiencies, it was the goal of the workshop to improve the understanding of the relations between both areas of research in software engineering and not to give new definitions of either requirements engineering or software quality.

On the Call for Papers addressing the above problems we received 24 submissions. After the reviewing process, we accepted 14 papers which have high quality and cover the research areas related to the workshop. Finally, 13 full and position papers were presented at REFSQ '94 and discussed with the 23 participants (including the organizers) coming from eleven different countries.

Workshop Structure

With respect to the described research problems and the contents of the accepted papers, we divided the workshop into three sessions:

- Non-functional requirements (chair: Gernot Starke)
- Requirements traceability (chair: Klaus Pohl)
- Structuring requirements engineering (chair: Peter Peters)

[1] The proceedings of the workshop where published by Augustinus Verlag and can be ordered by sending an email to {pohl,peters}@informatik.rwth-aachen.de.

To ensure the effectiveness of the workshop, we defined a strict organization scheme: Each talk was restricted to 15 minutes and followed by a short discussion and a talk summary. This summary was supported by three means.

Firstly, the speakers had to describe their research context by drawing one or more labeled arrows between a set of given concepts (cf. figure 1). This set is divided into the *agents* which mainly contribute to the software development process and the *documents* that must be produced during that process. The arrows should describe the kind and direction of the relation between those concepts. While the arrows between people show some kind of social interaction, the arrows between documents describe technical and logical relationships. Arrows between both levels mainly focus on production/usage relationships.

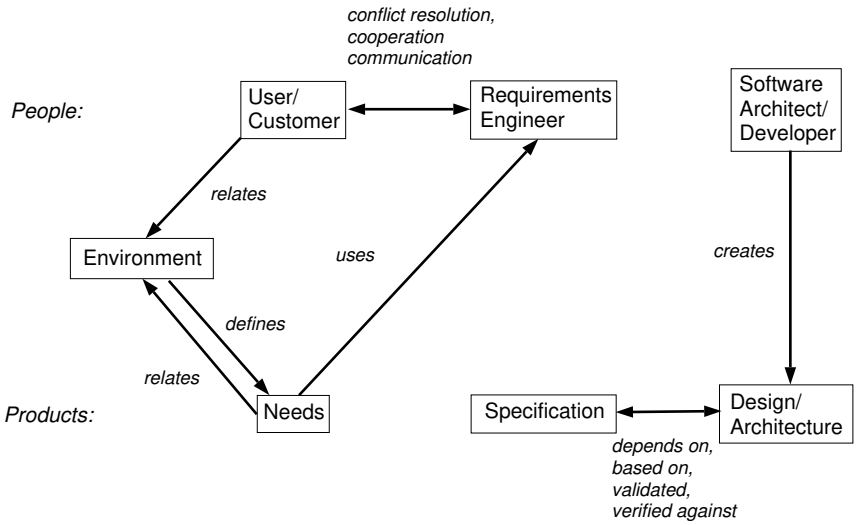

Fig. 1.1 Possible arrows on the contextual slide

Secondly, they had to pinpoint in short terms their main idea, the approach used as well as its possible benefits and shortcomings. These results built the basis for the talk summaries given below.

Thirdly, the participants had to respond to three session specific questions in order to formulate some general opinion on important topics in the field covered by the session (cf. figure 1.2).

Session 1: Non-functional requirements	1. What is your definition of non-functional? 2. Whay are the general relationships between typical quality attributes and features? 3. How can non-functional attributes/features be allocated to development activities?
Session 2: Requirements traceability	1. What should be traced? 2. How and when to trace it? 3. How can tracing improve software quality?
Session 3: Structuring the RE process	1. Is requirements formalization necessary to improve software quality? 2. What's the scope of the user? 3. How to deal with requirements overload?

Fig. 1.2 Session specific questions

The presentation of each session was followed by a discussion of related major topics raised by the talks or during the talk discussions. The results of the discussion (the viewpoints of the participants and open research problems) were summarized in a short session summary.

At the end of the workshop a general discussion was outlined with the aim of integrating the session summaries and identifying important research topics in the intersection of requirements engineering and software quality. This led to a final workshop summary and a research agenda that comprises topics worth elaborating to enable higher software quality in the future.

2 Session I: Non-Functional Requirements

2.1 Overview of Talks

This session comprised five presentations centered around non-functional or quality requirements. Several of the speakers presented a kind of domain-model, e.g. for performance, security or timing. Therefore they concentrate on refinements and extensions of certain single quality factors instead of addressing software quality in general. This allows a very precise identification of possible benefits and improvements. The presented approaches focussed mainly on pure requirements engineering, leaving room for additional research and improvement concerning the integration with the later phases of software engineering, like design and implementation.

In the first talk *Andreas Opdahl* refined performance needs with respect to the user by early quantification and validation of performance demands. The resulting performance model later is checked against the architectural model of hard- and software. Benefits of this approach are the prevention of performance bugs and the improvement of the performance specification. Quantification of user-related psychological performance aspects pose a problem. Furthermore, the mapping between organizational and hardware/software metric is difficult.

Tereza Kirner (co-author *Alan M. Davis*) presented a concise model of time behavior by formalizing real-time requirements. This approach employs algebraic specifications based on a fixed set of timing constraints, resulting in a more detailed and testable specification of time behavior. The main problem is the lack of a technique or method implementing this approach.

Hubert Hofmann (co-author *Ralph Holbein*) related security requirements to a given environment by refining the notion of security into several sub-factors to cover both social and economic dimensions. This provides the foundation for developing a security framework, although the lack of appropriate metric is regarded as a problem.

Jos Trienekens explained how to determine quality factors by using a checklist-based structured communication between users and requirements engineers. The resulting prioritization of the quality factors is beneficial, whereas the overall approach is very static in the sense that only a fixed and given set of quality factors can be handled.

The last speaker, *Rui Crespo*, identified and proposed quality requirements on requirements documents by introducing non-decomposition relations between quality factors. He related several of those factors to requirements documents, resulting in improved RE documents. The purely theoretic nature of this approach is currently its main disadvantage.

Figure 2.1 depicts the main focus of the talks given in this session. Obviously, all talks focus more on the user involvement for producing high quality specifications and (nearly totally) ignore the later development phases.

2.2 Discussion Summary

The discussion following this session concentrated on the question, how *non-functionality* can be characterized. The term itself was disputed by the workshop participants, as their understanding

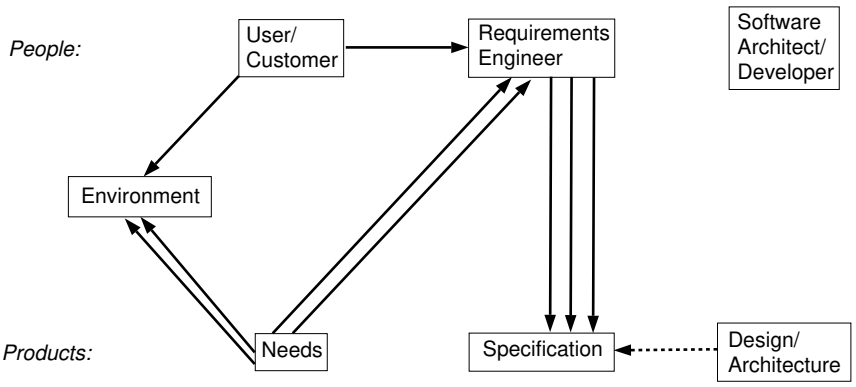

Fig. 2.1 The working context of the talks at session 1

of non-functionality varied considerably. The definitions or characterizations proposed during the discussion are explained below:

- Non-functionality can be characterized by the quality feature and attribute graph as proposed by Barry Boehm (discussed in [Boehm 78], Within those graphs, higher-level quality features, like maintainability, are decomposed into lower-level quality attributes like modularity.
- An alternative definition, as introduced by Ian Sommerville in [Sommerville 89, p. 88], are the constraints under which a system must operate.
- Another characterization is the distinction between *quantifiable* and *qualifiable* factors. The latter are also termed *what* – or functional issues.
- The distinction between functional and non-functional factors could be replaced by the distinction of *global* and *local* factors. The former denote quality factors, which should be global for any system, the latter denote local aspects of functionality and system behavior.
- Several participants proposed to generally avoid any distinction between functional and non-functional factors. In that case, no definition of non-functionality is needed.

Regardless of the diversity of those characterizations it was under agreement that it may shift over time wether any quality requirement is functional or non-functional. Possible reasons for this effect is an improved or modified domain understanding or advances in underlying technology. Furthermore it went undisputed that the role or viewpoint of the person responsible determines wether any requirement is functional or non-functional.

3 Session II: Requirements Traceability

3.1 Overview of Talks

Within this workshop session four talks were given.

The first talk by *Dieter Landes* dealt with the role of non-functional requirements in the development of knowledge-based systems. He addressed software quality aspects by focusing on the relation between requirements specification and design. His approach enables the modelling of relations between (non-functional) requirements and design activities by capturing design decisions together with their rationales and their inputs and outputs. The main benefits expected are support for maintenance and reuse as well as traceability from the requirements specification down to the design documents.

The second talk by *Orlena Gotel* (co-author: *Anthony Finkelstein*) focused on the contribution structure underlying requirements. The problem of tracing a particular requirement back to its contributors was identified as an existing problem in industry. Based on the PAD (principal, author, documentor) contribution structure a framework was proposed by which the various contributions

can be related to the different agents and their roles. Furthermore, firmer foundation for building software quality can be gained and the final authority is handed over to the people involved in the process. Problems with the approach are the high costs for capturing information, the scale and ownership of information and the non conformity to the philosophy dominant in today's organizations.

In the third talk by *Philip Morris* (co-authors: *Andrew Coombes and John McDermid*) a method for specifying safety critical applications was outlined. Given a model of the application and a particular goal to be satisfied, first a strategy is chosen, by which the goal is either integrated in the existing model or decomposed into sub-goals. By integrating the goal in the existing model a new model of the application is created, which provides the starting point for the integration of other goals, etc. Relationships between goals are currently not treated by the proposed method. Furthermore, the definition of strategies as well as the selection of a strategy by which a goal is either decomposed or integrated in the application model is not supported. Moreover, the method was not yet evaluated against a full scale application. Nevertheless, the presented approach creates useful traceability relationships between the various models and the stated goals.

The last talk of this session was given by *Tsuyoshi Nakajima* (co-author: *Alan M. Davis*). Two models for capturing and classifying errors detected during software specification reviews were presented: the *domain of error* model and the *source of error* model. The classification and the traceability of the detected errors enables an improvement of the activities performed during the review and thereby ensures a better error detection process. However, more abstract user needs are not captured by the approach and no validation by using real world application had been made so far.

The main focus of the talks given in this session is depicted in figure 3.1. As in the first session, the main focus of the contributions was on traceability needs between requirements engineers, the users, their needs and the specification.

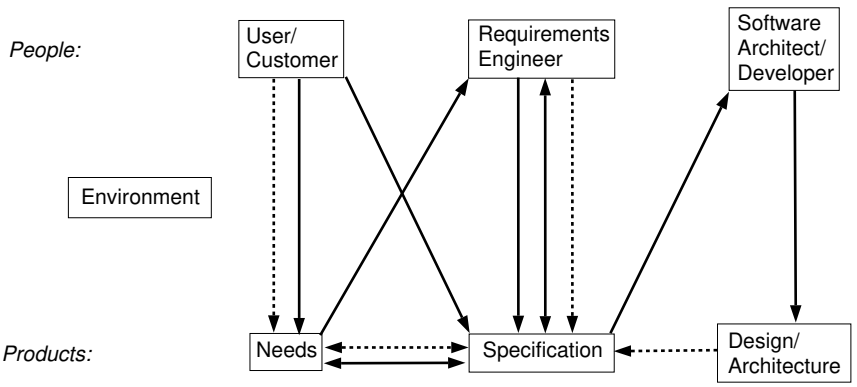

Fig. 3.1 The working context of the talks at session 2

3.2 Discussion Summary

The first part of the discussion was organized around the questions *'What to trace?'* and *'How, when and by whom to trace?'*. The second part dealt with research topics related to requirements traceability and software quality.

All participants agreed that there exists no unique answer to the questions. Capturing of traces must always be related to the usage and the expected pay-offs. Only if the potential usage of the trace information is identified, the information to be captured can be defined. Nevertheless, overall software quality could be improved by capturing traces about:

- requirements changes and their evolutions
- decisions made during the specification process and their rationale
- contributors (people and their roles)

- interrelation of different documents and representations
- activities together with their input and output
- relations between non-functional requirements (decompositions, correlations, consequences, contradictions, conflicts, etc.)

The answer to the question *'How, when and by whom to trace?'* is threefold:

Firstly, trace information can be captured manually or automatically. The participants agreed that capturing trace information should be mostly automated to avoid additional workload for the people responsible for capturing the traces.

Secondly, trace information can either be captured during process execution or by applying later reviews. Again, there was a common agreement that the information should be captured as a by-product during process execution.

Thirdly, the notion of *traceability jobs* was introduced. Since producing suitable traces is an essential activity during system development there should be people involved, who are only responsible for the capturing of correct and sufficient traces. Like quality assurance or process modeling activities are performed by a special group of people, also traceability should be seen as special task, which requires certain needs, training and responsibilities.

Research Topics related to Traceability

The discussion on the research topics for requirements traceability in relation to software quality can be described by six topics:

(1) *Costs, usage and pay-offs of traces*
A classification is needed by which the kind of information to be traced can be identified dependent on the usage. Moreover, guidelines for calculating the costs raised by the trace and the expected pay-offs are needed. In other words, since traceability always depends on the usage and causes additional costs, some evaluation criteria are needed to define information to be traced during process performance with respect to the expected pay-offs gained.

(2) *Ownership of trace information and misuse*
Capturing trace information raises two problems which are currently not considered in traceability research. First, access and usage of trace information have to consider the ownership, both personal and organizational, of the trace information. Second, possible misuse of restricted trace information must be avoided by relating the access of trace information to certain roles in the development process.

(3) *How to produce trace information retrospectively?*
Little is currently known how traceability can be added to existing systems. Approaches are needed, which allow to introduce traceability to existing systems, e.g. during reverse or reengineering activities.

(4) *Traceability needs required by software quality standards or CMM*
The relationship between traceability and existing standards and guidelines should be analyzed. Traceability needs implicitly stated in the CMM or ISO 9000 and the influence of trace information on the maturity levels should be named, i.e. which tracing capability is needed on which maturity level and which information could support which improvement activity.

(5) *Learning from traces*
Process traces are a valuable resource for experienced based process improvement. Approaches are needed that support the learning (generating) of new process chunks from captured traces.

(6) *Structuring traces*
Trace information is often used by people who were not involved in the tracing. Therefore, capturing traces in an arbitrary manner is not sufficient. Traces must

be recorded using user independent trace models which enable a broad use of the information, i.e. which are understandable to all persons using the trace.

4 Session III: Structuring Requirements Engineering

While the first two sessions dealt with special topics (non-functional requirements, requirements traceability) the third session should indicate requirements for performing requirements engineering processes in order to enable the production of high quality software. The contributions ranged from requirements and process formalization to user involvement.

The first talk, presented by *Michael von der Beeck*, described a method that enriches Structured Analysis with finite automata in order to provide a more precise formulation of informal requirements. Because of the stronger semantics of these automata less errors in specifications with respect to correctness and completeness can be made. But at this moment no general procedure for the design of these automata is available and it is unclear if this technique of method integration is generally applicable.

In the next talk, *Veronique Plihon* put the RE process in the center of her analysis. She wants to provide guidance in requirements elaboration by formalizing the way—of—working and modeling situation dependencies by means of process meta models. This provides the possibility to develop guidance tools, but it is still a topic of on-going research if applicable process models for every RE task in every scenario exist.

The third talk, given by *Matthias Rauterberg* (co-author: *Oliver Strohm*) focused on the benefits of user involvement in RE. In order to optimize the requirements specification process he proposed continuous participation of the user in cycles of requirements optimization, because empirical studies showed that cost and time of the development process can be reduced this way. But there is still a need for empirical studies in differing software development communities or projects of different sizes and no tools supporting user involvement are available by now.

In the final talk, *Peter Holm* (co-author: *Jan Ljungberg*) explained how it will be possible to achieve a compact and task adequate requirement specification by following frameworks for 'reality construction' adapted from several theories of speech philosophy. These frameworks could help structuring documents and processes and could support requirements identification and the selection of representations. But at the moment, these frameworks are too theoretical and not validated by empirical studies.

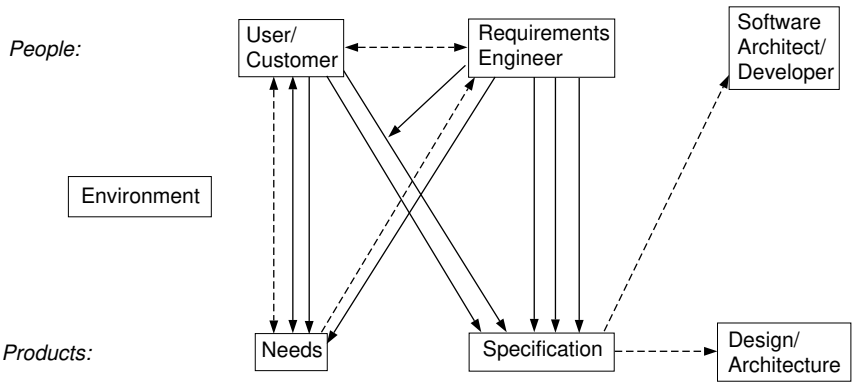

Fig. 4.1 The working context of the talks at session 3.

Figure 4.1 summarizes that even in this more global session the talks focused mainly on the requirements elicitation and elaboration parts of the software development process. The problems identified within this session led directly to the general discussion summarized below.

5 General Discussion

The general discussion at the end of the workshop focused on four main topics.

- *Crystallization of quality needs*
  Software quality needs and requirements evolve over time. There is no clear understanding of the needs at the beginning of the requirements engineering process. Along optimization cycles the requirements change frequently, are elicited again and crystallize during the process itself.
- *Participation of later development stages in the RE phase*
  With respect to the first topic, all participants agreed that the consideration of design, implementation, maintenance and other system development phases in the requirements phase are as important as the participation of users.
- *Informal versus formal languages*
  The discussion about informal versus formal specification languages and the *gap* between them ended up in a differentiation between the *specification* which captures the current system understanding and the *representation* used to express this specification. A precise understanding can be represented using informal languages, e.g. laws are defined in natural languages, but also using a formal language. On the other side, pure system understanding can be hidden using a lot of formalism but also represented in natural language. Hence, the language does not imply if a specification is precise or not (cf. [Pohl 1994]). By making a difference between the specification and its representation , the *gap* between informal and formal languages does not exist at all. Moreover (cf. next point) participants agreed, that informal, semi-formal and formal representations should be used in parallel.
- *Formalization of specifications*
  a. Formalization of requirements often goes along with better system understanding. Correct formalization adds semantic, which was not specified before.
  b. Total formalization of a requirements specification is impossible. Dependent on the information and the expected usage, many representations (languages) are needed. There was a common agreement, that each kind of language has its advantages and disadvantages. For example, an explanation of a particular requirement could be easy represented using natural language, whereas proves of the specification are enabled by using First Order Predicate Logic. The final specification should therefore be represented using different languages. But this causes the need of smooth integration of the different languages

Research Agenda

From the insights gained during the workshop as well as from other topics not covered in the presentations and discussions a research agenda was defined at the end of the workshop, extending the research agenda identified in the *traceability* session (cf. page 8).

- *Involvement of affected people*
  As mentioned in many discussions and in contrast to most of the presentations in the workshop, the involvement of designer, maintenance staff, financial officer, etc. in the requirements engineering phase is an important research issue. It is obvious, that each stakeholder is involved in only some requirements engineering activities. Research should point out for which kind of activities which kind of stakeholder should be involved.
- *Empirical studies*
  Requirements engineering is an activity which had been performed infinite times in industry. Despite this, little is known about the RE process itself. Empirical studies are needed to clarify on the one hand why certain software quality features are not met and on the other to validate research ideas (results) to improve software quality

in real settings. This was seen as the most important research topic in the intersection of requirements engineering and software quality.

- *Combination of different representations*
  As mentioned above, many representations are used during the RE process and for representing the final specification. Research effort is needed for identification and clarification of the advantages of the different languages, with respect to a given scenario. Second, the different representations must be coupled smoothly, to enable traceability and consistency among them.
- *Focus on the process to improve overall quality*
  In industrial engineering the process is seen as central for improving product quality. The positive experience with process oriented quality improvement has also raised strong interest in the software community and many approaches to improve software processes were proposed. However, in the case of RE little is known about the processes. Research effort is needed to elicit reusable process chunks by which high quality specifications can be produced.
- *Role of scenarios and software architectures*
  Quality needs can be better understood if they are explained by scenarios. Hidden needs can be elicited by using scenario based requirements techniques. Furthermore, domain specific quality needs can be integrated in domain specific software architectures. The use of such architectures during the requirements engineering phase could avoid, that well known quality needs remain undetected. However, it is an open research topic, what kind of scenarios and architectures are needed, how they should be defined and used to improve software quality in the first place.

The workshop summaries presented above give only a coarse reflection of the talks given in the workshop itself. However, more insight into the topics covered by the talks could be gained by reading the papers included in this proceedings.

We would like thank all workshop participants for enabling an interesting and valuable workshop and the organizers of CAiSE '94 for providing a pleasant working environment and organizational help whenever we needed it.

Bibliography

[Boehm 1978] Barry Boehm. *"Characteristics of Software Quality"*; North Holland, 1978.

[Pohl 1994] Klaus Pohl; *The Three Dimensions of Requirements Engineering: A Framework and its Applications"*; Information Systems, Vol. 19, No. 3, 1994, pp. 243–258.

[Sommerville 1989] Ian Sommerville. *"Software Engineering";* Addison-Wesley, 1989.