

[Session #]

Usability and Other Quality Aspects Derived from Use Cases

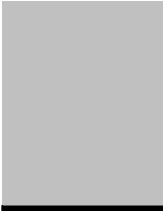
D. Kerkow, K. Kohler, J. Dörr

Abstract

Usability is one of several quality aspects (also named non-functional requirements) according to ISO 9126. The elicitation of those during an early phase of the development is crucial in information systems as well as in embedded systems. Despite the practical importance of usability and additional aspects like performance and maintainability, and so forth, there is rarely any guidance on the elicitation of these requirements. We present an approach to elicit usability requirements in concert with supplementary requirements based on a use case based specification of functional behavior.

1. Introduction

Most software development approaches like the Rational Process (Kruchten, 1998) address the elicitation and realization of functional requirements systematically but disregard the importance of quality aspects like usability. Indeed these quality aspects have to be considered right from the beginning of a project. Only if they are well understood and stated very precisely in an early stage, the right design decisions and trade-offs can be made while creating the users experience during the



development of the project in the subsequent phases. User centered approaches (Mayhew, 1999; Rosson and Carroll, 2002) recommend to set usability goals early, but don't provide guidance on how to elicit them. In addition, they completely disregard other quality aspects like performance, maintainability, and so on.

The main goal of our approach is to achieve a minimal, complete and focused set of measurable and traceable (IEEE, 1998) quality aspects (also named non-functional requirements; NFR for short) of the software at an early state of the development project and in this manner set the preconditions for a successful software product. The main contribution of our approach is to provide guidance during the elicitation process and by this ensure, that the NFRs are derived in the required quality as mentioned before.

This guidance is achieved by the following key elements:

- *A reference model* that captures knowledge and experience about quality attributes and dependencies among them. It also captures knowledge about constructive pattern to satisfy the quality demands. This model improves continuously with subsequent projects and represents a comprehensive view on quality attributes and their refinements.
- *A quality model* that captures general characteristics of usability and other NFRs. In the quality model, quality attributes are refined into more fine-grained quality attributes. The quality model also includes metrics to measure the NFRs (for example “time to perform a task” for efficiency), and means to achieve them. In particular, this model reflects views of different stakeholder roles, such as customer and developer. The quality model supports measurability, completeness as well as focussedness due to the views.
- *A prioritization questionnaire* is used to prioritize high-level quality attributes (i.e., maintainability, efficiency, reliability, usability). This helps to narrow down the comprehensive view on quality aspects and to focus only on those that are important for the concrete project. The prioritization questionnaire supports a minimal and focused set of NFRs.
- *Checklists* provide detailed elicitation guidance. The checklists are derived from the quality model. They help to elicit usability NFRs in concert with an user model, use cases and a high-level

architecture. The checklists support the completeness and measurability of the derived NFRs.

- A *template*, which embeds FRs as use cases into a full-fledged requirements document and provides specific places for documenting NFRs. This template supports traceability from NFRs to FRs, completeness and focussedness.

Our approach can be applied as extension to any established software development processes that accomplish the following precondition:

- Characteristics of the users have are understood and described.
- Functional requirements are captured in a task-oriented manner. The approach consists of two basic steps:
 - The tailoring of the quality model: During this step we tailor the experience-based “reference model” to the need of the customer and it’s project
 - The elicitation process: During this step the values that instantiate the quality attributes in a measurable way are defined.

The paper is structured as follows. In Section 2, we sketch our terminology and explain the notation of the quality model. Section 3 includes the tailoring process as described above. Then, we present in section 4 the elicitation process by way of an example. Section 5 summarizes our experience and Section 6 discusses related work. We finally conclude in Section 7 with an outlook on future work.

2. Main Concepts

This section describes the fundamental concepts and terminology of our approach

2.1 Quality model

A quality model describes typical refinements of high-level quality attributes. Quality attributes (QAs for short) are non-functional characteristics of a user task, system task, system component or organization that have an influence on the product. The distinction between different types of quality attributes is important for our elicitation process. Each type of quality attribute is elicited differently (see Section 3).

The quality model shows the refinement of high-level QAs into more fine-grained QAs and related metrics. The idea of the quality model is to come up with QAs that are measurable, i.e., to QAs to which a metric can be associated. In addition, it describes relationships between different

QAs. Therefore, it captures experience of previous projects. Our quality model is similar to the goal graphs of for instance (Chung et al., 2001), but emphasizes dependencies, and distinguishes between different types of QAs.

Figure 1 shows an excerpt of the quality model (QM) for usability. It is based on several usability standards (for example ISO 9126, 9241-10, 9241-11, 14598-1) and was tailored to the needs of the stakeholders of

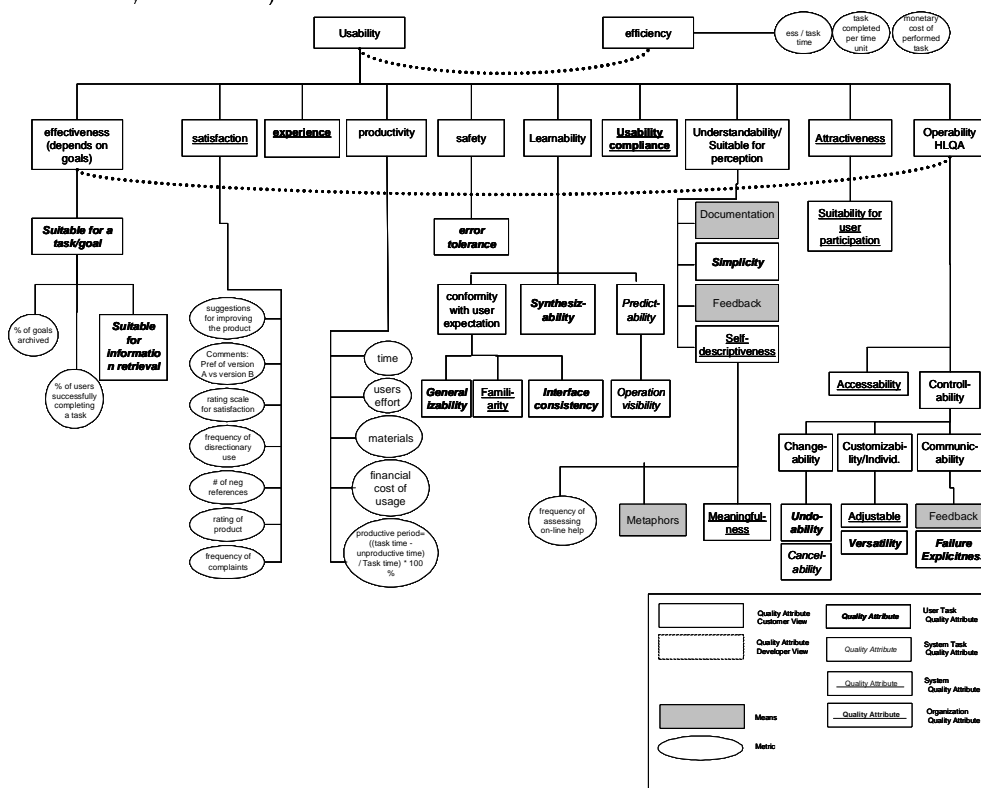
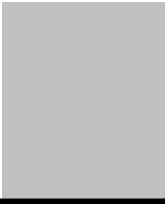


Figure 1: Excerpts of the quality model for the QA “usability”

our case study. In the QM, QAs are represented by white rectangles. Ovals are metrics to measure the related quality attribute. There are four different types of QAs in this quality model, they refer to different artefacts in the process or structure the hierarchy:

- *General* QAs such as “Understandability” are used to structure the QAs on lower levels.

- 
-
- *System QAs*, such as “Accessibility”, are QAs related to the system and its subsystems (A *system* (for example, “wireless control and monitor system”) can be refined into a set of subsystems (for example, “mobile device”). These subsystems have input controls as well as output controls. *Environmental factors* (for example temperature, brightness) constrain the system.
 - *User Task QAs and System Task QAs*. We distinguish between two types of *tasks*: *user tasks* and *system tasks*. *User tasks* are tasks, a certain user has to perform. They are supported by the system (for example, “monitoring of certain machines”), but include some user involvement. *System tasks* are tasks the system performs. In contrast to user tasks, the user is not involved in system tasks. With this distinction we adapt Constantine’s separation of user actions and system response introduced with the essential use cases (Constantine and Lockwood, 1999).
 - *Organizational QAs* concern the organizational aspects. This also includes development process related aspects, such as required documentation, reviews etc as well as standards, legal issues.

These QAs are constrained by non-functional requirements (NFRs). A *NFR* describes a certain *value* for a QA that should be achieved in a specific project. The NFR constrains a QA by determining a *value* for a *metric* associated with the QA. For example, the NFR “Similar alarms have to be acknowledged in a similar way (number of different actions = 0)” constraints the QA “generalization”. The value is determined based on an associated metric “number of different actions”. For each NFR, a *rationale* states reasons for its existence (for example, “The actor shall concentrate on removing the cause of the alarm and not on acknowledging the alarm”).

The first type of QA serves to structure the hierarchical decomposition of the more fine-grained QAs. This structure is also used in the template to document the NFRs. How the NFRs for the QAs are elicited, depends on the type of the QA they constrain. This is described in Section 3.

A QA can be positively or negatively *influenced by* another QA. If the “operability”, for instance, is higher, the “effectiveness” will increase (positive influence). Dependencies exist within one quality model as well as between quality models of different QAs, as explained in the following section.

2.2 Dependency graph

Usability as it is understood in ISO 9126 (2001), mainly addresses user interface QAs. Quality in use as it is defined in ISO 9126 addresses more than that. The users experience with the system can only be expressed accurately, if all other quality aspects are taken into account. In our process we have currently developed refinement methods for the high-level QAs “efficiency”, “reliability”, “portability”, and “maintainability”, based on ISO 9126. During the derivation of the usability QM, the QMs for the other QAs are used to identify dependencies towards them. Figure 2 shows an excerpt of the QM for “efficiency” (Dörr et al, 2003).

It is obvious that there is a dependency between the general QAs “usability” and “efficiency”. The identification of more fine grained dependencies such as between the efficiency QA “usage time” and the usability QA “satisfaction” is a necessary step to identify NFRs and conflicts among them. The identified dependencies are integrated in the checklists and used in the consolidation step described in section 4.

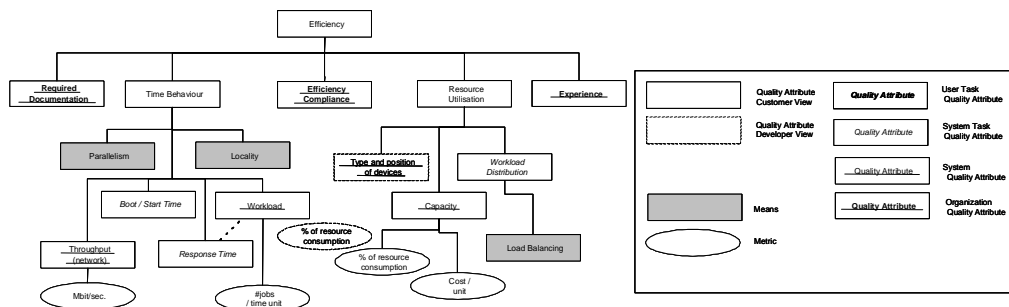


Figure 2: Excerpts of the Efficiency QM used to identify dependencies

3. The Tailoring Process

Our approach provides a default reference model that can be used as a starting point to tailor the QM to the context of each company and project. In addition, a company might have an own QM that shall be used. In this case, it is very important to agree on the meaning of the different QAs in the QM. Our recommendation is to build a QM together with the company in a workshop. By doing so, the QM benefits from the already integrated experience of our reference model and it is tailored to the project and company.

Figure 3 describes the process of tailoring the reference model to the project and company. The tailored QM is used as input to derive the checklists and templates for documenting NFRs. Experience made during the project is incorporated into the reference models. This feedback loop ensures continuous improvement of the reference artifacts and ensures a subsequently more comprehensive understanding of the quality attributes and their dependencies.

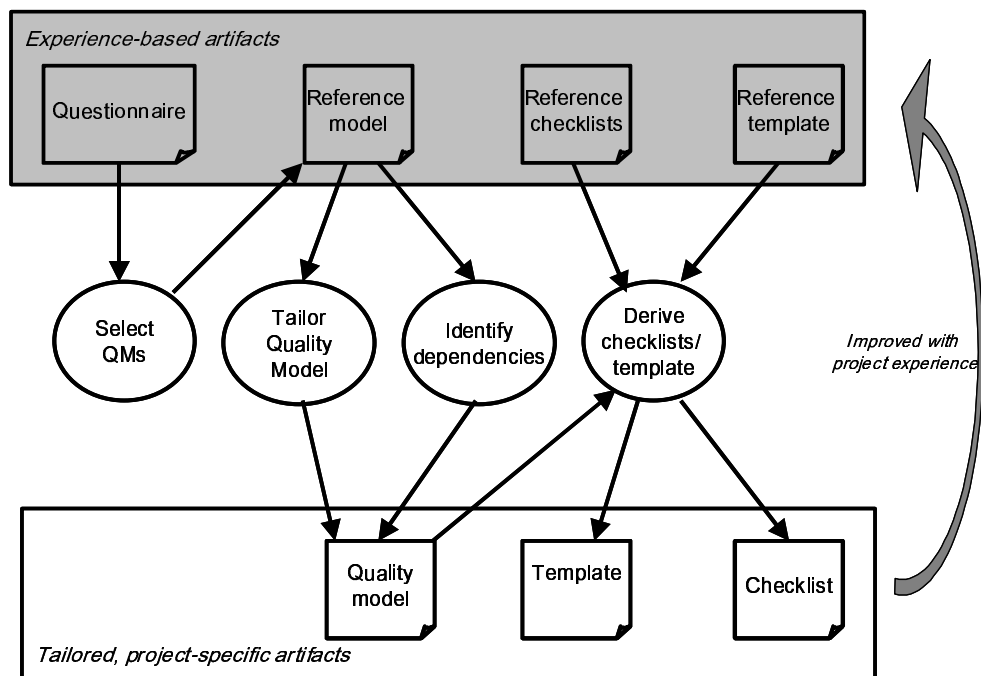


Figure 3: [Process of tailoring the quality models, checklists and the template](#)

The hierarchy of the QM gives the structure of the checklists. General QAs (for example, Operability) are, therefore, a means for structuring the checklist, while the QAs at the lowest level (for example, Undoability) are directly used to elicit the NFRs constraining them. The type of the QA influences the way the questions in the checklist are phrased:

- Organizational QAs are used in initialization checklists that focus on general aspects in contrast to the concrete system or its task.

- User task QAs are iterated over the use cases (for example, use case 1, then use case 2)
- System task QAs are iterated over the use case steps (for example, step 1, then step 2)
- System QAs are iterated over the various subsystems in the system (for example, input controls of mobile devices first, then output controls) and the user model.

The structure of the template is strongly influenced by the quality model. The NFRs constraining the different types of QAs are denoted at different places in the template:

- NFRs constraining the organizational QAs are documented in an organizational requirements section.
- NFRs constraining user task QAs are attached to the use case diagrams and are, therefore, documented in a use case diagram section.
- NFRs constraining system task QAs are directly attached to each use case in the textual use case description section. Therefore, the use cases have a field “NFRs”, where each system task oriented QA is listed. Below such a system task oriented QA, there is a list of the use case steps that express system tasks (for example, familiarity: step2, step4). The NFRs for each system task are then expressed at this use case step (for example, feedback step2 - “While the system performs this step, the user must be informed about the state of the system (time of asynchronous state transparency=0)”, step4 - “...”) and so on.
- NFRs constraining system QAs are denoted at two places in the template. First, if a NFR constrains a system QA of a subsystem (for example, “the input control must be able to operate wearing gloves”) that is used in a use case, the NFR is attached to that use case. Therefore, each use case also includes a list of system QAs in the field NFRs. Below such a system QA, there is a list of all subsystems (for example, Accessibility: mobile device input control, mobile device output control, stationary terminal input control, stationary terminal output control). The NFRs for each subsystem are then expressed at this subsystem (for example, Accessibility: mobile device output control - “the display of the PDA must be illuminated in order to be accessible in the darkness”). Second, the system NFRs are documented in the section of task overspanning NFRs. The structure is similar to the structure in the use cases (i.e.,

there is a list of all system QAs, below each system QA there is a list of all subsystems), but it aggregates the NFRs from all use cases and the ones that are not specific for one use case. This is done because a consolidation step searches for dependencies between NFRs concerning one subsystem.

As a result of the “tailoring” process the requirements template is derived. Table 1 shows a subset of this template.

Table 1: Subset of the requirements template

1. Organizational requirements
1.1 Process requirements
1.2 Stakeholder requirements
2. Task descriptions
2.1 UC diagram
2.2 Textual UC description
3. Task over spanning requirements
3.1 Textual description of task over spanning NFRs

4.2 The Elicitation Process

In the following sections, we describe the activities to be performed within the elicitation process. We use examples from a case study, provided by an industrial partner about a wireless framework for mobile services. The application enables up to eight users to monitor production activities, manage physical resources, and access information within an industry plant. The user can receive state data from the plant on his mobile device, send control data from the mobile device to the plant components, position the maintenance engineer and get guidance to fix errors on machines. The case study is based on a real system and was provided by an industrial partner in the context of the ITEA-EMPRESS project.

4.1 Prerequisites

The elicitation process is based upon the following artefacts:

- A *prioritization questionnaire*: Many times, budget and time limitations oblige to prioritize and select the subset of high-level QAs most important for a project. This activity is supported by a prioritization questionnaire developed at Fraunhofer IESE. It builds a ranking order for the QAs described in ISO 9126 (for example, maintainability, efficiency, reliability, and usability). The questionnaire is described in more detail in (Paech et al., 2003). Based on this ranking order, quality models for the most important high-level QAs relevant for the project can be chosen.
- A *user model* (compare Paech & Kohler, 2003), representing assumptions about the users of the system, their goals, responsibilities, capabilities and skills.
- *The system's functionality* (behavior) described by use cases (UCs),
- *The physical architecture*, if available, and further implementation constraints (some NFRs can only be elicited if the detailed physical system architecture is known).

As described above, some of the QAs are associated to user tasks and system tasks. Therefore, we recommend use cases to describe the FRs. This seems to be beneficial, because QAs associated to user tasks can directly be related to use cases. QAs associated to system tasks can directly be related to use case steps. However, we believe that our approach can be applied to other notations as well.

4.2 Elicitation and documentation of NFRs

This section describes the activities and documents needed to elicit, document and consolidate NFRs. A checklist that is derived from the quality model as described in Section 2 guides each activity. The activities are explained in more detail in the following. We distinguish between different elicitation activities: user task NFR elicitation, system task NFR elicitation and system NFR elicitation and organizational NFR elicitation.

Activity "Elicit organizational NFRs"

In this activity (see figure 4), NFRs are elicited that constrain QAs of the organization. The customer, for example, might have certain requirements concerning the organizational structure and experience of a supplier. This process is guided by a set of clues in form of a checklist. These clues suggest thinking about domain-experience, size, structure or age of the supplier organization, as well as required standards (for example RUP),

activities (for example inspections), documents or notations (for example Statecharts).

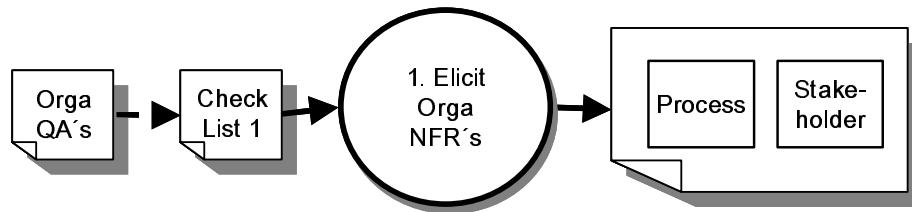


Figure 4: Elicit organizational NFRs

Activity “Elicit user task NFRs”

In this activity, NFRs are elicited that constrain QAs of user tasks (see Figure 5 for a representation of this activity). In our case study, the QA “simplicity” included in the quality model is a user task QA. These QAs are documented for each use case included in the use case diagram, because each use case represents a user task. As shown in Figure 6, NFRs are added to use cases with the help of notes.

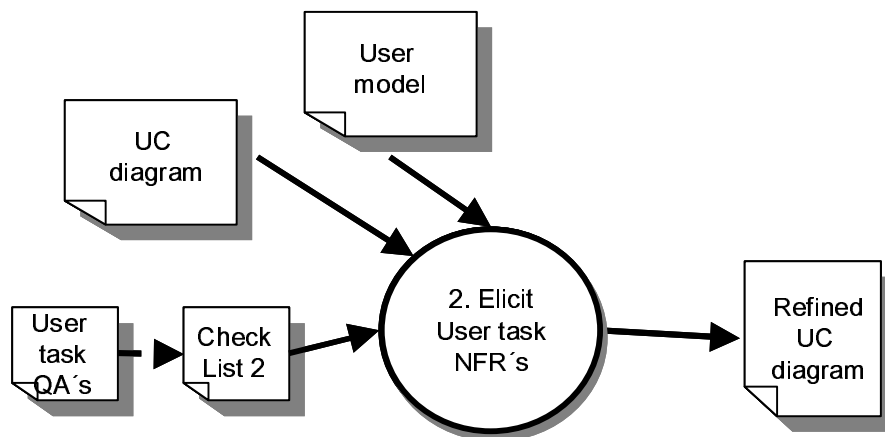


Figure 5: Elicit user task NFRs

In our case study the requirement “To monitor the status of the machine, only necessary information about fill status, chocolate

temperature and activation status shall be provided and only minimal functions to fulfill the tasks provided.” was attached to the use case “Monitor status of the machine”. To prevent premature design decisions, a justification from the stakeholder is required. We recommend the Socratic Dialogue as a method to reveal rationales and question decisions. In the Socratic Dialogue (Kahn, 1998), a NFR is questioned over and over again and the phrasing stakeholder must have a good reason for phrasing it, otherwise it won't be accepted. If the stakeholder can't express the rationale for a NFR, an alternative NFR can be offered. Now the stakeholder will have to make a decision between two alternatives and justify the decision. Implicit goals and criteria can be discovered that way. The resulting rationale for the NFR mentioned above was phrased as “reduce the cognitive load, so actor has a good overview of status” and was documented in parenthesis behind the NFR.

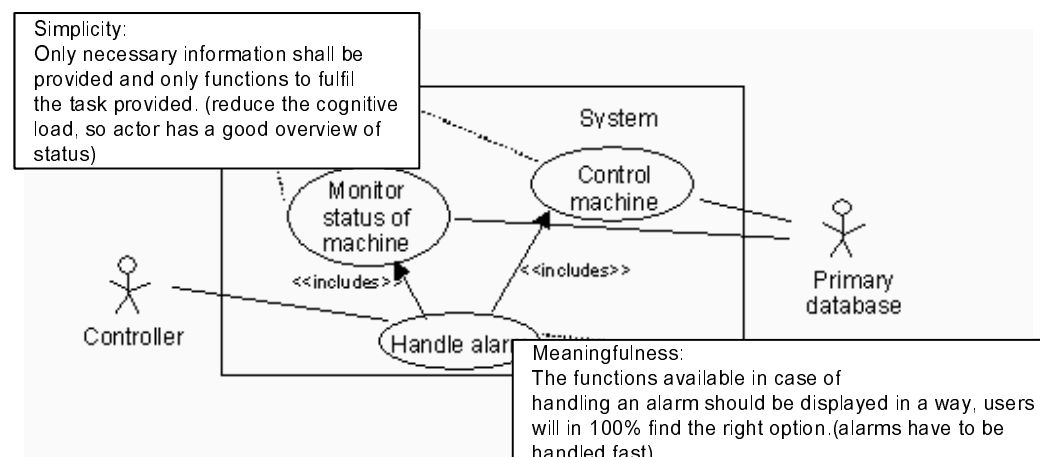


Figure 6: Use cases with attached user task NFRs

Activity “Elicit system task NFRs”

In this activity, NFRs are elicited that constrain QAs of system tasks (see Figure 7 for a representation of this activity). The elicitation is based on the detailed interaction sequence (also called flow of events) documented in the use case. Table 2 shows the textual description of the use case

“handle alarm”. It describes that the system shows an alarm and where the alarm was produced. As reaction to this, the user acknowledges the alarm, so other users know s/he is taking care of it.

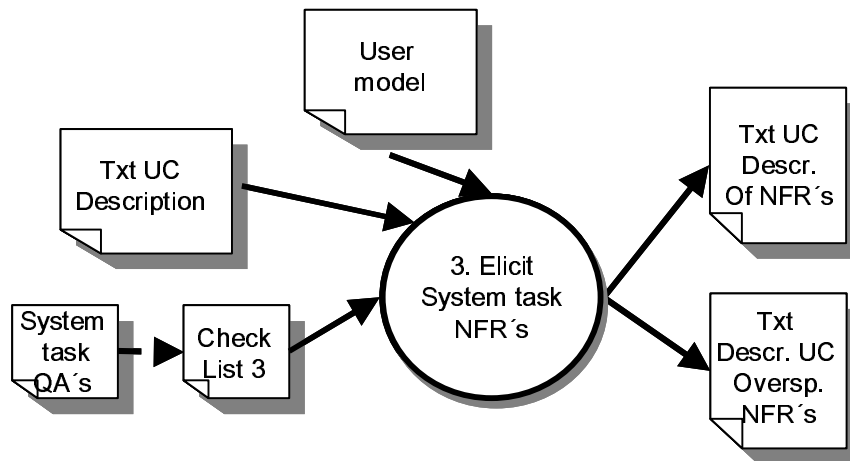


Figure 7: Elicit system task NFRs

As a result of the elicitation and documentation process, NFRs that constrain the user task QA “Undoability” were documented. The NFR “the user must be able to undo the notification.” was attached to the use case step 4 “The system informs other users about acknowledgement” and the same NFR was attached to use case step 10. Both requirements were documented in the NFRs field within the textual description of the use case, after being justified by the customer in the Socratic dialogue. The rationale elicited was “(in case the user is hindered to finish the task)”. As shown in Figure 6 and Figure 7, the rationale was documented in parenthesis.

Table 2 : UC steps with attached system task NFRs

UseCase	Handle alarm
Actors	Controller, Guest, Administrator, Machine
Intent	User removes a warning send by a certain machine
Preconditions	Use Case “Boot up system”

Flow of events	<ol style="list-style-type: none"> 1. The system requests data from the primary database regularly 2. The system shows alarm and where the alarm was produced. [Exception: Multiple alarms] 3. The actor acknowledges alarm to let others know he/she is going to take care of it. [Exception: Another actor has acknowledged the alarm] 4. The actor moves to the machine following the path displayed by the system. 5. During the walk, the actor monitors the status of this machine by the system (-> use case "monitor status of machine") 6. The actor removes the problem by controlling the machine (-> use case "control machine") 7. The system sends control data to first database.
Exceptions	<p>1.1 Multiple alarms: System opens a window for each alarm message.</p> <p>3.1 Another user has acknowledged the alarm: System removes alarm message from the screen and shows acknowledgement.</p>
Rules	The alarm warning will always have the highest priority.
NFRs	<p>...</p> <p>Undoability:</p> <ol style="list-style-type: none"> 1. the user must be able to undo the notification (in case the user is hindered to finish the task 2. ... <p>Feedback</p> <ol style="list-style-type: none"> 1. While the system performs this step, the user must be informed about the state of the system 2. ...

Activity "Elicit system NFRs"

In this activity, NFRs are elicited that constrain QAs of the system and subsystems (see Figure 8 for a representation of this activity). In this

activity, the architecture of the physical subsystems is used, if available. The subsystems and architecture constraints on our case study are shown in Table 3.

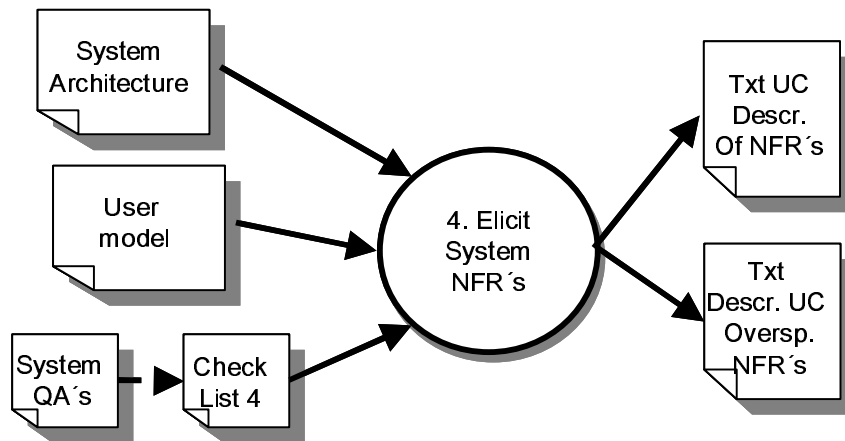


Figure 8: Elicit System NFR's

The checklist gives instructions on how to consider the user model while phrasing NFRs for each use case description and physical subsystem of the system architecture. Table 4 shows an excerpt of the user model that is considered in relationship to the system. A user model describes assumptions about typical users of the system. Here we capture user roles, their expected capabilities and limitations in an physical environmental and in a physiological sense. The user roles are a subset of the actors (only those actors, that are human) mentioned in the description of the functional requirements.

Table 3: Constraints on system-architecture

Constraints on overall architecture
A.1 Windows CE OS must be run on the PDA
A.2 Standard PDAs must be used (replaceable)
A.3 Standard Network components must be used (replaceable)
A.4 Throughput: WLAN 11 Mbit/sec

A.4 Server: Windows 2000 must be used
A.5 Secondary Database -> PDA: Wired Network required
A.6 Downloading and monitoring at the same time is not possible

Table 4: User models for Controller and Administrator

Actors	Controller	Administrator
Goals	Minor system downtime	Minor maintenance effort
Environment	Wears gloves, noisy, space between the engines not very well illuminated	
Skills	Mechanical engineer, working in shifts	Technician,...
Personal Attributes	20-60 years	20-60 years
Equipment	Mobile phone, screw driver	-

As Table 5 shows, the NFR field of the use case description is segmented into NFRs related to every physical subsystem.

Table 5: UC with attached system NFRs

NFRs
<p>...</p> <p>Accessibility requirements:</p> <p>mobile device input control: The subsystem shall be able to be operated wearing gloves. (the actor “controller” usually wears gloves whenever he manipulates a machine)</p> <p>mobile device output control : the display of the PDA must be illuminated in order to be accessible in the darkness</p>

In the use case “handle alarm”, NFRs for the QA “accessibility” could be phrased for the physical subsystem “PDA- input control”. The subsystem shall be able to be operated wearing gloves. The rationale for this NFR is the practice of the actor “controller” to wear gloves whenever he manipulates a machine.

The elicited NFRs for single subsystems are documented within the textual use case description as well as in the section “use case overspanning textual description of NFRs”. This is done to be able to consolidate the requirements over several use cases.

Activity “Consolidate”

In this activity, the NFRs are analyzed for conflicts. This activity includes two sub-activities (see Figure 9 for a representation of this activity). In the first, NFRs of one kind, i.e., of one quality attribute, for example, usage time is analyzed over all use cases. The checklist gives hints on how to identify conflicts and how to solve them. It has to be checked, for example, whether NFRs can be achieved if use cases are executed in parallel. In the second sub-activity, NFRs that constrain different QAs are validated under consideration of the dependencies documented within the quality model.

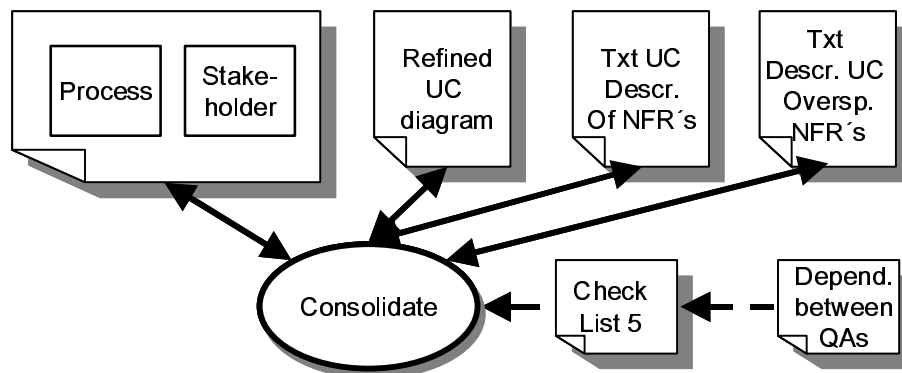


Figure 9: Consolidate

The consolidation activity discovered an important conflict between the determined accessibility requirements and the defined hardware constraints. As shown in Figure 9 one of the accessibility requirements stated:

“The subsystem shall be able to be operated wearing gloves.”

The hardware constraints shown in Figure 8 constrained the PDA to standard PDAs, which do not support users wearing gloves. This conflict between the requirements can be found in an early phase by making the

relationships between requirements explicit. The complexity of dependencies between NFRs ascends if QAs of different QMs are taken into consideration.

The dependency relationships make explicit that the QA workload from the efficiency QM has an influence on the communicability aspect of the usability QM. It does not take much to imagine that a freezing system lacks of communicability.

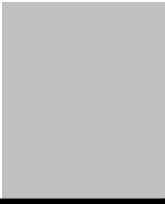
Another relationship is the one between throughput from the efficiency QM and feedback from the usability QM. Again, to receive permanent feedback about the state of the system, the network components must provide enough resources.

5. Experience

We have used this approach for a subset of the quality attributes so far in a case study with an industrial partner in the ITEA-EMPRESS project and in a workshop with 10 practitioners. In the case study, we spent half a day with the customer in discussing and tailoring the default quality model and half a day in eliciting NFRs for some of the Use Cases. The customer acknowledged that the time was very worthwhile as he discovered many new NFRs he had not been aware of before. Also, it helped him to specify them more precisely. In the workshop, we spent one hour explaining our method and then within another two hours we interactively went through the checklists and filled the template. Again, the feedback was very positive as the participants acknowledged that this was the first systematic method they had seen to elicit NFRs. They particularly liked the idea of the quality model, checklists, and template to capture experience on NFRs. In addition, they liked the use of use cases and the architecture to ensure completeness and ease traceability.

6. Related work

The current paper addresses on the one hand usability as a high level QA, and on the other hand the concept of quality in use, which can only be understood as an integration of usability QAs with other QAs such as efficiency, reliability, portability and maintenance. In contrast other approaches like (Bevan, 1995; Dzida et. al., 2001) that support the measurement of usability, are restricted on usability only and don't provide this comprehensive view on the quality attributes. Additionally these



approaches measure only a predefined, fixed set of metrics that is not tailored to the needs of the project.

Further related work on NFRs can be found in the communities of requirements engineering, architecture design and performance engineering (Cysneiros & Leite, 2001).

Another approach developed for security requirements is the elicitation of NFRs by using Mis-Usecases (Alexander, 2001). Those are an excellent means to analyze security threats, but inappropriate to express security requirements explicitly in a measurable way, as discussed in (Firesmith, 2003). The approaches presented by (Firesmith, 2003) and (Sindre and Opdahl, 2000) are very similar to our activities of user task level elicitation. Our results show, that not every quality attribute can be expressed by documenting a task. There are NFRs that can only be elicited and expressed using descriptions of system components and flows of events not explicitly related to the quality attribute. We have also learned, that a quality attribute such as efficiency can be understood in many different ways by different stakeholders. The definition of the concept and finding dependencies to other attributes are also part of the elicitation process.

As exemplified by last year's STRAW workshop, in the architecture community several approaches rely on goal graphs for specifying NFRs and FRs and their dependencies (Boehm et. al, 2001; Egyed et al., 2001; Gross and Yu, 2001, In et al, 2001; Liu and Yu, 2001). In these approaches, the graph captures the actual FRs and NFRs. In contrast, we only use the graph to represent dependencies between quality attributes and we place the NFRs in the requirements document template.

7. Conclusion

In this paper, we have presented an approach for eliciting and documenting usability requirements in concert with use cases and a high-level architecture. There are three major innovations. One is the use of a quality model and quality attribute types to capture general knowledge on NFRs, while specific NFRs are captured in a template. The other are detailed checklists on how to elicit NFRs in concert with user models, use cases and architecture. With this approach, we achieve a minimal, complete and focused set of measurable and traceable NFRs. This leads to the applicability in early stages of the software development process and differs from testing approaches. The dependencies to other QAs than

usability extends the approach to support quality in use as it is addressed in ISO 9126.

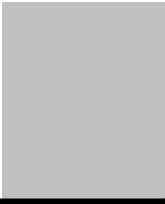
There is first evidence from practitioners that this approach valuable for the QAs usability, efficiency, reliability and maintainability. We have extended the validated approach to other QAs, such as the one presented in this paper. We assume that the defined concepts, such as the different types of QAs and metrics can be applied to every high-level quality attributes. We have experienced, that only the necessary context factors are variant, such as the granularity of the system- architecture and the usage of artifacts such as user models instead of data assumptions. The main open issue is improving the reference quality model with subsequent project. In that way the tailoring of the quality model will be more and more simplified until it finally can be cut down to the selection of the relevant “branches” of the QA model and does not need any further adaptation of checklists.

8. Acknowledgements

We thank our colleagues for fruitful discussion within the Fraunhofer IESE-EMPRESS-Team. We acknowledge the EUREKA-ITEA project “EMPRESS” (ITEA 01003) for partly funding our research. Furthermore, we want to thank all partners in the ITEA project EMPRESS that contributed to our research.

References

- Alexander, I. (2001) Misuse Case Help To Elicit Nonfunctional Requirements, IEE CCEJ
- Bevan, N. (1995) Measuring Usability as Quality of Use, Software Quality Journal, 4, p.115-150
- Dörr, J., Kerkow, D., von Knethen, A., Paech, B. (2003) Eliciting Efficiency Requirements with Use Cases, Proceedings of REFSQ Workshop
- Dzida, W. et al. (2001) Gebrauchstauglichkeit von Software, Schriftenreihe der Bundesanstalt für Arbeitsschutz und Arbeitsmedizin
- Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J., (2000) Non-Functional Requirements in Software Engineering”, Kluwer Academic Publishers
- Constantine, L., Lockwood, L. (1999) Software for Use, ACM Press
- Cysneiros, L.N., Leite, J.C.S.P (2001) Driving Non-Functional Requirements to Use Cases and Scenarios, XV Brazilian Symposium on Software Engineering
- Egyed, A., Grünbacher, P., Medvidovic, N., refinement and evolution issues in bridging requirements and architecture – the CBSP approach, STRAW 2001

- 
-
- Firesmith, D., "Security Use Cases", in Journal of Object Technology, vol. 2, no. 3, May-June 2003, pp. 53-64.
- Gross, F., Yu, E., "Evolving system architecture to meet changing business goals: an agent and goal-oriented approach", STRAW 2001
- IEEE Recommended Practice for Software Requirements Specifications, IEEE Std. 830-1998
- In, H., Boehm, B.W., Rodgers, T., Deutsch, W., "Applying WinWin to Quality Requirements: A Case Study", ICSE 2001, pp. 555-564, 2001
- In, H., Kazman, R., Olson, D., From requirements negotiation to software architectural decisions, STRAW 2001
- ISO/IEC 9126-1:2001(E), "Software Engineering - Product Quality - Part 1: Quality Model", 2001
- Kahn, C.H., Plato and the Socratic Dialogue: The Philosophical use of a Literary Form, Cambridge University Press, 1998
- Kruchten, P., The Rational Unified Process. An Introduction, Addison-Wesley, 1998
- Liu, L., Yu, E., From requirements to architectural design – using goals and scenarios, STRAW 2001
- Mayhew, D.J., The Usability Engineering Lifecycle, Morgan Kaufmann Publishers, 1999
- Paech, B., Kohler, K., *Task driven requirements in object-oriented development*, in Leite, J., Doorn, J.,(eds.) *Perspectives on Requirements Engineering*, Kluwer Academic Publisher, 2003
- 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 architecture and requirements engineering“, ICSE-workshop STRAW, May 2003
- Rosson, M.B., Carroll, J.M., Usability Engineering, Morgan Kaufmann Publishers, 2002
- Sindre, G. & Opdahl, A., „Eliciting Security Requirements by Misuse Cases, Proc. TOOLS Pacific 2000, pp 120-131, 20-23 November 2000