Eliciting Efficiency Requirements with Use Cases

J. Dörr, D. Kerkow, A. von Knethen, B. Paech
Fraunhofer IESE, Sauerwiesen 6, 67661 Kaiserslautern, Germany
{doerrj, kerkow, vknethen, paech}@iese.fraunhofer.de

Abstract

Non-functional requirements provide the glue between functional requirements and architectural decisions. Thus, it is important to elicit and specify the non-functional requirements precisely. In practice, however, they are mostly neglected. In this paper, we sketch an approach developed in the context of the EMPRESS project, which allows efficiency requirements to be elicited in conjunction with use case. This is part of a more general, experience-based approach to elicit and specify non-functional requirements in concert with functional requirements and architecture.

1. Introduction

The last few years have seen a growing awareness of the requirements engineering community for architectural issues and vice versa. Several authors argued convincingly for the tight interdependencies between functional requirements (*FRs*), non-functional requirements (*NFRs*) and architectural options (*AOs*) that need to be made explicit early, for example, [1], [2].

While there are many established methods for the specification of FRs, for instance, use cases [3], and several approaches for specifying AOs, for example, patterns [4], there is little guidance available on how to elicit and specify NFRs in concert with FRs and AOs. The problem is that different kinds of NFRs, such as efficiency or security requirements, need to be treated differently. The different communities concentrating on the different NFRs exemplify this. Thus, it seems difficult to define one method to cope with all NFRs.

In this paper, we propose an approach for specifying efficiency requirements in concert with use cases and, if available, a high-level architecture. This method is so far tailored to efficiency requirements, but we believe that it can be generalized also to other NFRs, such as reliability requirements. We believe this because our approach is based on some general characteristics that can then be used for each type of requirement (e.g., efficiency, reliability, maintainablity requirements).

The main goal of our approach is to achieve a minimal, complete and focused set of measurable and traceable NFRs. The quality criteria on NFRs mentioned are a subset of the general quality criteria on requirements defined by the IEEE-Std. 830 [5].

- *Minimal* means that only necessary NFRs are stated so that the design space is not restricted prematurely.
- *Complete* means that all NFRs of the stakeholders (e.g., customer and developer) are captured.
- *Focused* means that the impact of the NFRs on the solution is clear. A NFR, for example, may concern the system context (namely the customer processes), the system, a FR, or an AO. This supports unambiguity in the sense of the IEEE Std. 830.
- *Measurable* means that a metric is given on how to verify that the system satisfies the NFRs. This supports verifiability and unambiguity in the sense of the IEEE Std. 830.
- *Traceable* means that rationales are given that describe why the NFR is necessary and how it is refined into subcharacteristics. This also supports modifiability in the sense of the IEEE Std. 830.

Our main focus has not been on eliciting consistent NFRs so far. However, our approach includes a consolidation step, where dependencies between elicited NFRs are checked. When specifying means to achieve certain NFRs, consistency has to be treated with more attention.

To accomplish the different quality criteria of the IEEE Std. 830, our approach provides:

- a quality model that captures general characteristics of efficiency (quality attributes), metrics to measure these quality attributes, and means to achieve them. In particular, this model reflects views of different stakeholder roles, such as customer and developer. This quality model supports measurability, completeness as well as focussedness due to the views.
- a distinction of different types of quality attributes, which gives guidance on how to elicit NFRs. This specific treatment for the various types supports focussedness of the NFRs.
- detailed elicitation guidance in terms of checklists and a priorisation questionnaire. The former are derived from the quality model and the types of quality attributes and help to elicit efficiency NFRs in concert with use cases and a high-level architecture. The latter is used to prioritize high-level quality attributes (i.e., maintainability, efficiency, reliability, usability). The

checklists support completeness, the priorization questionnaire supports the focussedness of the NFRs.

- a template, which embeds use cases into a full-fledged requirements documents and provides specific places for documenting NFRs. This template supports traceability from NFRs to FRs, completeness and focussedness.
- the use of rationales to justify each NFR. Using rationales supports minimality of the set of NFRs.

The paper is structured as follows. In Section 2, we sketch our terminology and explain the notation of the quality model. Then, we present our approach by way of an example. Section 4 summarizes our experience and Section 5 discusses related work. We conclude with an outlook on future work.

2. Terminology

This section describes the foundation of our approach. Subsection 2.1 points out a metamodel that describes the basic concepts of our approach. Subsection 2.2 gives an overview on the "quality model", which instantiates parts of the metamodel.

2.1. Metamodel

The metamodel describes the main concepts we are dealing with (see Figure 2). In the following, we explain the most important ones.

- A *quality attribute* (*QA*) is a non-functional characteristic of a *system*, *user task, system task*, or *organization.* Quality attributes of the organization include development process specific aspects.

  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). QAs can be *refined into* further QAs. In addition, QAs can have positive or negative *influences* on each other. A more detailed description of the types of QAs and their relationships can be found in Section 2.2.

- A *system* (e.g., "wireless control and monitor system") can be refined into a set of subsystems (e.g., "wireless network", "mobile device"). *Architectural requirements* (e.g., "the system shall have a database") constrain the system.

- We distinguish between two types of *tasks: user tasks* and *system tasks*. *User tasks* are tasks, a certain users has to perform. They are supported by the system (e.g., "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. Tasks can be refined into further tasks. User tasks can be refined into more fine-grained user tasks. Furthermore, user tasks can be refined into parts carried out by the user and system

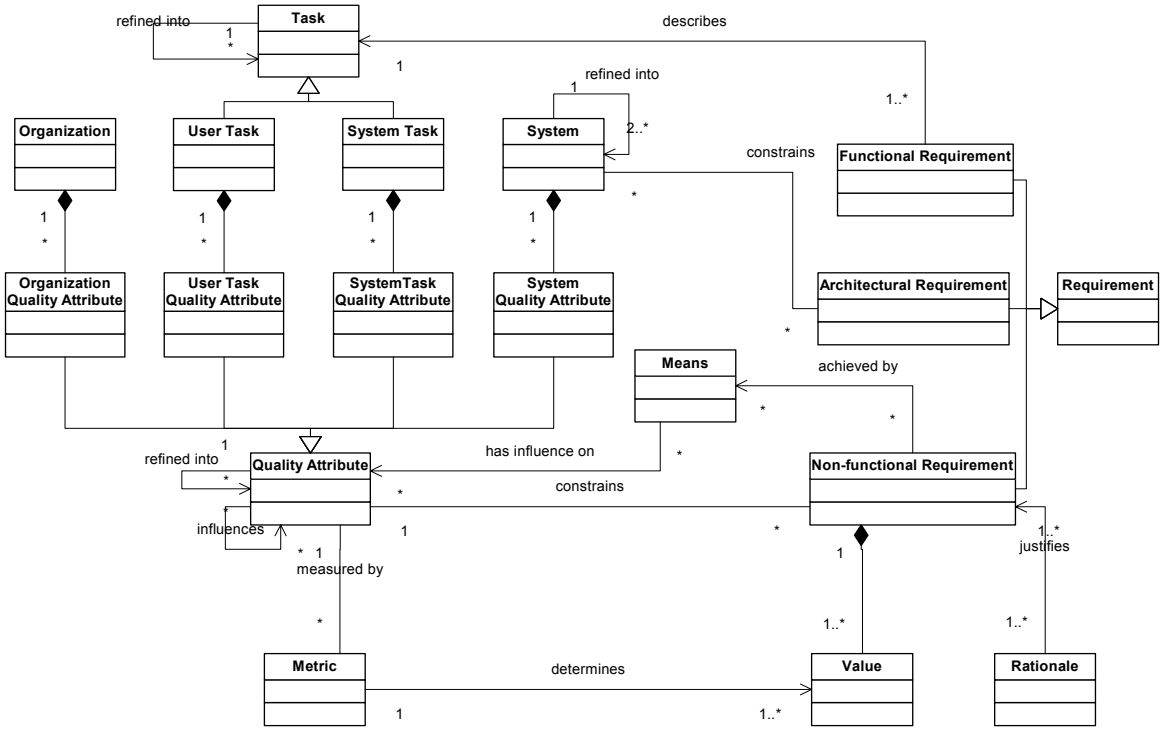

Figure 2: The metamodel

tasks (e.g., a user task "monitoring machine x" is refined into a set of system tasks such as "system displays alarm message if machine runs out of filling"). A *task* is described by one or more *FRs*.

- A *NFR* describes a certain *value* (or *value domain*) for a QA that should be achieved in a specific project. The NFR constraints a QA by determining a *value* for a *metric* associated with the QA. For example, the NFR "The database of our new system shall handle 1000 queries per second." constraints the QA "workload of database". The value is determined based on an associated metric "Number of jobs per time unit". For each NFR, a *Rationale* states reasons for its existence (e.g., "the user will be unsatisfied if it takes more than 2 seconds to display alarm message").
- We distinguish problem-oriented refinement (refinement of NFRs according to the constrained QAs) from solution-oriented refinement of QAs. The latter is made explicit in terms of *means*. A *means* is used to achieve a certain set of NFRs. In many cases, a means describes an AO that can be applied to the architecture to achieve a certain QA (e.g., "load balancing" is used to achieve a set of NFRs concerning the QA "workload distribution"). However, a means can also be process related (e.g., the means "automatic test case generation" is used to fulfill NFRs regarding "reliability").

2.2. Quality model

A quality model instantiates parts of our metamodel. It describes typical refinements of high-level QAs into more fine-grained QAs, metrics, and means. The idea of the quality model is to refine QAs into 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, [7], but emphasizes dependencies, and distinguishes between different types of QAs. Figure 4 gives an example for such a quality model for the QA "efficiency".

In Figure 4, QAs are represented by white rectangles. Grey rectangles are means that have influence on the related QA and ovals are metrics to measure the related quality attribute. There are five different types of QAs in this quality model (see also metamodel):

- General *QAs* such as "Time Behaviour" are used to structure the QAs on lower levels.
- *Organizational QAs,* such as "Experience", concern the organizational aspects. This also includes development process related aspects, such as required documentations, reviews, etc.
- *System QAs,* such as "Capacity", are QAs related to the system and its subsystems (e.g., related to the database, secondary storage or network).
- *User Task QAs,* such as "Usage Time", are related to tasks the system and the user are involved.
- *System Task QAs,* such as "Response Time", are related to system tasks, i.e., tasks that are carried out by the system, not including the user any more (e.g., calculation of results).

Only the latter four QAs are constraint by NFRs. The first type of QA serves as a structuring for the hierarchical decomposition of the more fine-grained QAs. This structure is also used for the template for documenting 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.

Four types of relationships can be found in such a quality model that relates the various kinds of QAs, means and

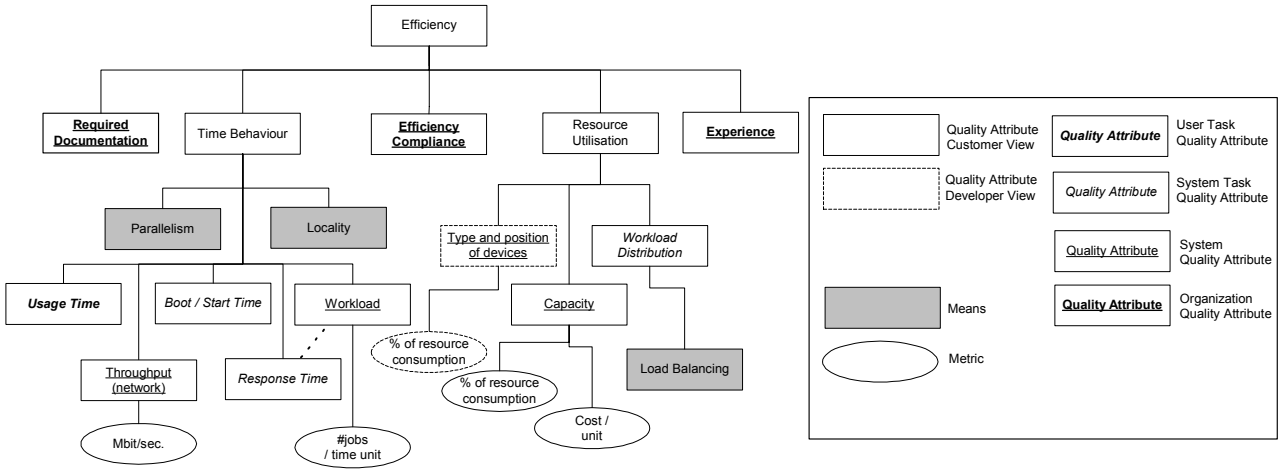

Figure 4: Quality model for efficiency

metrics. The metamodel in Figure 2 describes the general types of relationships.

- A QA, such as "efficiency", is *refined into* more detailed QAs, such as "time behaviour" and "resource utilization".
- A means *has influence on* a QA, i.e., it is used to achieve the NFRs constraining the QA. "Load balancing", for example, is influencing "workload distribution" and used to achieve the constraining NFRs (e.g., "The workload for computing the results must be equally distributed on the two processors").
- A QA is *measured by* a metric. The "workload" can, for example, be measured by the metric "number of jobs per time unit".
- A QA can be positively or negatively *influenced by* another QA. If the "workload", for instance, is higher, the "response time " will increase (negative influence).

Our approach provides a default quality model that can be used without adaptations by a company. Reasons for doing so can be a lack of time or money. We recommend tailoring the quality model to the context of each company and project. In addition, a company might have an own quality model that shall be used. In this case, it is very important to agree on the meaning of the different QAs in the quality model. Our recommendation is to build a quality model together with the company in a workshop. By doing so, the quality model benefits from the already integrated experience of our reference quality model and it is tailored to the project and company.

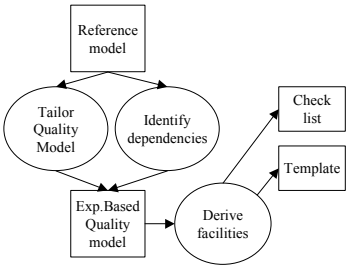

Figure 5: Experience based creation of a quality model

Figure 5 describes the process of tailoring the quality model to the project and company. The tailored quality model (experience based quality model) is used as input to develop checklists and templates for documenting NFRs.

The structure of the checklists is given by the hierarchy of the quality model. General QAs (e.g., time behaviour) are, therefore, a means for structuring the checklist, while the QAs at the lowest level (e.g., usage time) 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 at general aspects in contrast to the concrete system or its task.
- User task QAs are iterated over the use cases (e.g., use case 1, then use case 2)
- System task QAs are iterated over the use case steps (e.g., step 1, then step 2)
- System QAs are iterated over the various subsystems in the system (e.g., database first, then network1)

The structure of the template is also 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 (e.g., response time: step2, step4). The NFRs for each system task are then expressed at this use case step (e.g., response time: step2 - "The system has to respond within 2 seconds", step4 - "…").
- NFRs constraining system QAs are denoted at two places in the template. First, if a NFR constrains a system QA of a subsystem (e.g., "the database has to store 100000 entries") 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 (e.g., capacity: database, memory). The NFRs for each subsystem are then expressed at this subsystem (e.g., capacity: database – "the system has to store 100000 entries", memory – "…"). 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.

3. The elicitation process

As a result of the process "derive facilities" described above, the requirements template is created. Figure 6 shows a subset of this 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 overspanning requirements
   3.1. Textual description of Task overspanning NFR´s

Figure 6: Subset of the requirements template

The elicitation process is guided by our experience that various entities (e.g., user task, system task) have different types of QAs. Each NFR has to be elicited under consideration of this entity. In addition, if an entity is described by one or a set of documentation elements (e.g., a user task is described by a use case, a system task is described by a step of a use case), the NFR has to be documented together with this entity.

In the following sections, we describe the activities to be performed within the elicitation process. We use examples from a case study that deals with a mobile and interactive application. The application allows users to monitor production activities, manage physical resources, and access information. This case study is based on a real system and was provided by Siemens in the context of the Empress project.

3.1. Prerequisites

The elicitation process is based upon the documentation of

- the systems functionality (behavior) described by use cases (Ucs),
- the physical architecture, if available, and further implementation constraints (e.g. constrained HW-resources or constraints derived from the operating systems), and
- assumptions about the average and the maximum amount of data used in the system. The amount of data for each use case is determined under consideration of the amount of data for the entire system.

Since some activities of the elicitation and documentation process are closely related to the functionality, the completeness of the NFRs is limited by the completeness of the FRs.

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.

Figure 7 shows the pre-required documents and the activities to create them.

- *Activities "Prioritize" and "Chose quality models":* Many times, budget and time limitations oblige to prioritize and select a subset of high-level QAs most important for a project. This activity is supported by a prioritisation questionnaire developed at IESE. It builds a ranking order for the QAs described in ISO 9126 (e.g., maintainability, efficiency, reliability, and usability). The questionnaire is described in more detail in [6]. Based on this ranking order, quality models for certain high-level QAs relevant for the project can be chosen.
- *Activity "Elicit FRs":* In this step, the FRs are elicited and documented in form of a graphical use case-diagram. Each use case included in the diagram is later associated to NFRs that constrain QAs of user tasks. In addition, each use case is described textually. The textual description includes an interaction sequence between actor and system. This description allows us later to associate NFRs that constrain QAs of system tasks to use case steps.

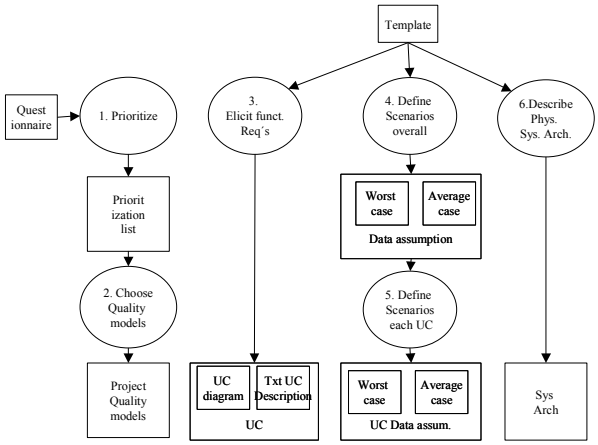

Figure 7: Development of prerequisites

- *Activities "Define scenarios" and "Define scenarios for each UC":* In order to be able to imagine NFRs, maximum and average usage data for the overall system, as well as for each use case are elicited and documented.
- *Activity "Describe physical system architecture":* Some NFRs can only be elicited if the detailed physical system architecture is known. So the architecture

must be elicited and documented, whenever it is available.

3.2. Elicitation and documentation of NFRs

Figure 8 shows the activities and documents needed to elicit and consolidate NFRs. A checklist that is derived from the quality model as described in Section 2.2 guides each activity. 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. Each activity focuses on eliciting NFRs that constrain one certain type of QA (i.e., organization QA, user task QA, system task QA, and system QA). The user task NFR elicitation is based on use cases. The system task NFR elicitation is based on the interaction sequence described for each use case. The system NFR elicitation is based on physical subsystems and interaction sequences.

Activity "Elicit organizational NFRs"

In this activity, 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. The customer is asked to phrase these requirements. 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 (e.g. RUP), activities (e.g. inspections), documents or notations (e.g. statecharts). In our case study, some of the requirements phrased were:

- "The supplier needs at least three years of experience in the domain of access-control."
- "The supplier has to create a specification document."

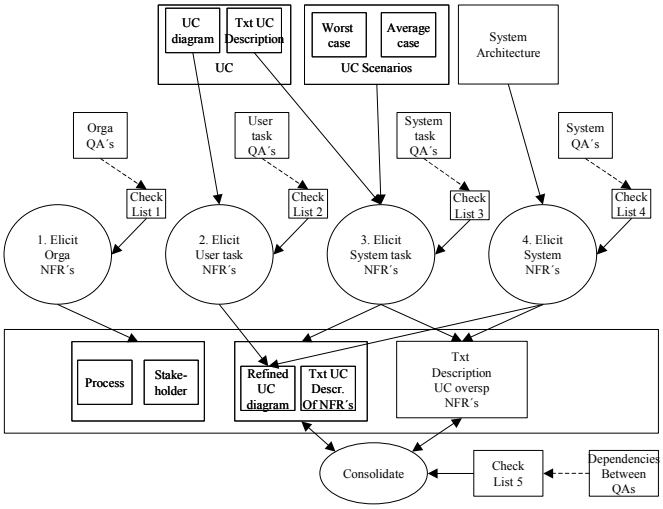

Figure 8: Elicitation process for NFRs

To avoid unnecessarily design decisions, the customer is instructed to scrutinize this NFR again, just as Socrates used to try to get to the bottom of statements over and over. This form of Socratic dialogue serves to uncover the rationale behind that NFR and bewares the customer from constraining the system unnecessarily. NFRs are reformulated until they reflect the rationale. It is a good practice to document the rationale as well [18].

As soon as the now elicited and justified NFRs are phrased in a measurable way (this is the case if the metric attached to the QA in the quality model can be applied to the requirement), it is documented in the chapter "organizational requirements" of the template.

Activity "Elicit user task NFRs"

In this activity, NFRs are elicited that constrain QAs of user tasks. In our case study, the QA "usage time" 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 9, NFRs are added to use cases with the help of notices.

In our case study the requirement "the use case shall be performed within 30 min." was attached to the use case "Handle alarm". Again, a justification as described above is performed to prevent unnecessary anticipated design decisions. The resulting rationale "breakdown of plant longer than 30 min. is too expensive" is documented in parenthesis behind the NFR.

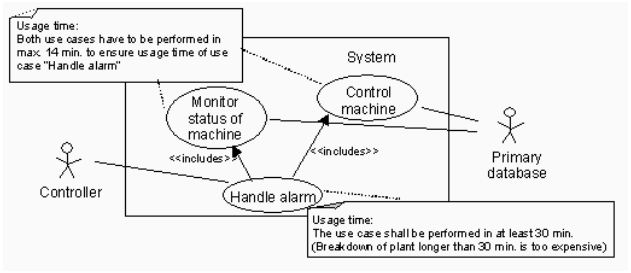

Figure 9: Use cases with attached user task NFRs

Activity "Elicit system task NFRs"

In this activity, NFRs are elicited that constrain QAs of system tasks. The elicitation is based on the detailed interaction sequence (also called flow of events) documented in the use case. For this activity, maximum and average usage data (Figure 7 shows the development process of this information) are needed. The checklist gives clues of thinking of scenarios where the maximum and the average amount of data are processed in the system. With these scenarios in mind, every step and every exception described by the use case description are checked. Elicited NFRs are documented. Figure 10 shows the textual de-

scription 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.

UseCase	Handle alarm
Actors	Controller
Intent	Actor removes a warning send by a certain machine
Preconditions	Use Case "Boot up system"
Flow of events	1. System requests alarm messages from the first database regularly 2. System shows alarm and where the alarm was produced. [Exception: Multiple alarms] 3. Actor acknowledges alarm to let others know he/she is going to take care of it. [Exception: Another actor has acknowledged the alarm] 4. Actor moves to the machine following the path displayed by the system. 5. During the walk, actor monitors the status of this machine by the system. (-> use case "monitor status of machine") 6. Actor removes the problem by controlling the machine (-> use case "control machine") 7. System sends control data to first database.
Exceptions	1.1 Multiple alarms: System opens a window for each alarm message. 3.1 Another user has acknowledged the alarm: System removes alarm message from screen and shows acknowledgement.
Rules	The alarm warning will always have the highest priority.
NFRs	**Response time (assumed times):** 1. every 5 sec. 2. at least in 5 sec. 3. just one click 4. at least 15 min. (worst case) 5. -> usage time of use case "monitor status of machine" 6. -> usage time of use case "control machine" 7. at least 5 sec.

Figure 10: UC steps with attached system task NFRs

As a result of the elicitation and documentation process, NFRs that constrain the system task QA "response time" were documented. The NFR "at least in 5 sec." was attached to the use case step 2 "System shows alarm and where the alarm was produced" and the NFR "just one click" was attached to the users reaction described in use case step 3. 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 lead to the statement, that the NFRs elicited were assumed times only and could be changed, if necessary. As shown in figure 10, the rationale was documented in parenthesis.

Activity "Elicit system NFRs"

In this activity, NFRs are elicited that constrain QAs of the system and subsystems. In this activity, again maximum and average usage data is needed. Additionally, the architecture of the physical subsystems is used, if available. The subsystems and architecture constraints on our case study are shown in Figure 11.

Constraints on overall architecture:

- Windows CE OS at PDAs
- Standard PDAs (replaceable)
- Standard Network components (replaceable)
- Throughput: WLAN at least 11Mbit/sec
- Server: Windows 2000
- Secondary Database -> PDAs: Wireless Network required
- Downloading and monitoring at the same time is not possible

Figure 11: Constraints on system-architecture

The checklist gives instructions on how to consider the scenarios while phrasing NFRs for each use case description and physical subsystem of the system architecture. As Figure 12 shows, the NFR field of the use case description is segmented into NFRs related to every physical subsystem.

Throughput requirements:
Network between secondary database and PDA:

- shall be able to deal in average case with 2 alarms every 10 minutes with 16 machines (assumed average number of alarms)
- shall be able to deal maximal with 8(1/PDA) * 60 alarms at the same time (assumed maximal number of alarms)
- shall be able to deal maximal with 8 people that download 1 doc (size of 8 docs constrained to: <55MBit) /person within 5-10 secs (assumed maximal number of downloads)

Capacity requirements:
PDA:

- shall have a maximum capacity of 64 MB (standard components shall be used to reduce costs)
- shall be able to handle up to 60 alarms at the same time (assumed maximal number of alarms)

Workload requirements:
PDA:

- shall allow 5 programs to be opened at the same time (assumed maximal number of programs that will be opened by the user)

Figure 12: UC with attached system NFRs

In the use case "handle alarm", NFRs for the QA "capacity" could only be phrased for the physical subsystem "PDA". The subsystem shall have a maximum capacity of 64 MB and shall be able to handle up to 50 alarms at the same time. The rationale for this NFR is the need for usage of standard components available at the consumer market. This rationale is documented as well.

The QA "throughput" does only apply to the subsystem "Network" by definition. Our experience shows, that some QAs are related to only a subset of subsystems. This relationship is documented in the quality model.

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 analysed for conflicts. This activity includes two sub-activities. In the first, NFRs for one physical subsystem are analysed 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.

The consolidation activity discovered an important conflict between the determined throughput requirements and the defined hardware constraints. As shown in figure 12 one of the throughput requirements stated:

- "The network between secondary database and PDA shall be able to deal in worst case with 8 people that download 1 doc (size of 8 docs constrained to <55Mbit) / person within 5-10 secs."

The restriction of the total size of 8 documents to 55 Mbits was added because the hardware constraints shown

in Figure 11 constrained the network to a 11Mbit/sec WLAN. The additional requirement would not been found without the consolidation activity.

4. Experience

We have used this approach so far in a case study with Siemens in the 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 to the case study project and half a day in eliciting the NFRs. 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 efficiency 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. They also pointed out the need for capturing the rationale and a supporting tool environment.

5. Related work

At last years' REFSQ we presented the following challenges for a method for the integrated elicitation and specification of FRs, NFRs and AOs [1]:

- Issue 1: Adequate abstraction levels for the elicitation and alignment of FR, NFR and AOs
- Issue 2: Views of different stakeholders in the elicitation of NFRs, FRs and AOs
- Issue 3: Identification of dependencies among FRs, NFRs and AOs
- Issue 4: Compact description of the solution space

In this paper, we concentrate on the first two issues The quality model contains abstract descriptions of NFRs (in terms of QAs) and AOs (in terms of means). Thus, to solve issue 1, we provide two main levels of abstraction. Within the quality model, QAs are refined on as many levels as necessary to distinguish different aspects. With respect to issue 2 (views), we distinguish developer and customer view. We do no support negotiation explicitly. However, by providing a standardized terminology in terms of the quality model, we help reducing conflicts and misunderstanding. The checklists make sure that all relevant aspects are considered.

We also give some hints on how to deal with issue 3 and 4. For dependencies again the quality models helps identifying typical dependencies. This is elaborated in the checklists. With respect to issue 4 (assessment), we use rationale techniques to capture decision making. The framework for the full-fledged method is described in [6]. The main achievement of this paper is a detailed description of the elicitation of efficiency requirements with the help of the checklists.

Further related work can be found in the communities of requirements engineering, architecture design and performance engineering:

Within requirements engineering, [10] provides a general method for specifying NFRs. It also gives specific advise for how to capture performance requirements with goal graphs. However, the emphasis is on the satisfycing step where means are elicited to achieve performance. In contrast, we focus on using use cases to elicit the customer view. [11] seems to be most similar, since it also combines use cases and NFRs. There are, however, essential differences. While we focus on elicitation of NFRs, Cysneiros and Leite focus on *satisfycing* NFRs. This term was coined in [10] to describe the fact that NFRs are not satisfied, but there are several ways to achieve them. Thus, in [11] use cases and NFRs are elicited separately and then combined to make sure that the use cases satisfice the NFRs. For example, because of an NFR new functionality is added into the use case diagram or into the steps of the use case description. In contrast, we use the use cases to elicit measurable NFRs. The same comment applies to [12] which also relates use cases and NFRs after both have been elicited. Furthermore, they only use high-level quality attributes, such as efficiency.

As exemplified by last years STRAW workshop, in the architecture community several approaches rely on goal graphs for specifying NFRs and FRs and their dependencies. [2][13][14][15]0. 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 template.

In the performance community it is emphasized, that performance issues are not suitably integrated in regular software engineering processes[16]. This is attributed to education issues, single-user and small database mindsets and in particular, lack of scientific principles and models. The main emphasis of this community is to create just these models, e.g., queuing models. So, for example [17] also uses use cases in the representation of use case maps in combination with efficiency NFRs. As for [11], however, it is already presupposed that the NFRs have been elicited adequately. The main emphasis is then to create a queing network reflecting the paths of the use case maps and the NFRs.

6. Conclusion

In this paper, we have presented an approach for eliciting and documenting efficiency requirements in concert with use cases and a high-level architecture. There are two 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 use cases and architecture. With this approach, we achieve a minimal, complete and focused set of measurable and traceable NFRs. There is first evidence from practitioners that this approach is worthwhile.

While so far we have concentrated on efficiency, we believe that this approach can be generalized to other high-level quality attributes, such as reliability or maintainability. This is because of the use of our meta model and our quality model. We assume that the defined concepts, such as the different types of QAs, metrics, and means can be applied to other high-level quality attributes as well. The main open question is whether the distinction between task and system-oriented QAs also gives helpful guidance for eliciting specific NFRs for other quality attributes. This question is the focus of our current work. After that, we will continue working on the other issues mentioned above, for example, notations that support the identification of dependencies between NFRs.

7. Acknowledgements

We thank our colleagues for fruitful discussion within the IESE_Empress-Team. We acknowledge the ITEA project EMPRESS for partly funding our research. Furthermore, we want to thank all partners in the ITEA project EMPRESS that contributed to our research. In particular, we want to thank Ricardo Jimenez Serrano (Siemens) for providing a case study to validate our approach and for taking over the role of the customer.

References

[1] . B. Paech, A. Dutoit, D. Kerkow, A. von Knethen: „Functional requirements, non-functional requirements and architecture specification cannot be separated – A position paper", REFSQ 2002

[2] In, H., Boehm, B.W., Rodgers, T., Deutsch, W., "Applying WinWin to Quality Requirements: A Case Study", ICSE 2001, pp. 555-564, 2001

[3] Cockburn A., *Writing Effective Use Cases*, Addison Wesley 2001

[4] Shaw, M., Garlan, D., "Software Architecture – Perspectives on an emerging discipline" Prentice Hall, 1996

[5] IEEE Recommended Practice for Software Requirements Specifications, IEEE Std. 830-1998

[6] 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 ", accepted for ICSE-workshop STRAW 2003

[7] ISO/IEC 9126-1:2001(E), "Software Engineering - Product Quality - Part 1: Quality Model", 2001

[8] von Knethen, A., Paech, B., Houdek, F., Kiediasch, F., "Systematic Requirements Recycling through Abstraction and Traceability", RE 2002

[9] Trochim, W. M. K., "The Research Methods Knowledge Base", Atomic Dog Pub Inc., Cincinnati, 2001

[10] Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J.,"Non-Functional Requirements in Software Engineering", Kluwer Academic Publishers, 2000

[11] Cysneiros, L.N., Leite, J.C.S.P, "Driving Non-Functional Requirements to Use Cases and Scenarios", XV Brazilian Symposium on Software Engineering, 2001

[12] Ana Moreira, Isabel Brito, João Araújo, "A Requirements Model for Quality Attributes", Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design, workshop da 1st International Conference on Aspect-Oriented Software Development, University of Twente, Enschede, Holland, 22-26 April , 2002

[13] In, H., Kazman, R., Olson, D., From requirements negotiation to software architectural decisions, STRAW 2001

[14] Egyed, A., Grünbacher, P., Medvidovic, N., refinement and evolution issues in bridging requirements and architecture – the CBSP approach, STRAW 2001

[15] Liu, L., Yu, E., From requirements to architectural design – using goals and scenarios, STRAW 2001

Gross, F., Yu, E., Evolving system architecture to meet changing business goals: an agent and goal-oriented approach, STRAW 2001

[16] Menasce, D.A., "Software, Performance or Engineering", Workshop on Software and Performance, pp. 239-242, 2002

[17] Petriu, D., Woodside, M., "Analysing Software Requirements Specifications for Performance", Workshop on Software and Performance, p.1-9, 2002

[18] A. H. Dutoit, B. Paech, "Rationale Management in Software Engineering. In: S.K. Chang (Ed.), "Handbook of Software Engineering and Knowledge Engineering. World Scientific, December 2001.