DESY 04-233

Requirements Engineering Patterns

An Approach to Capturing and Exchanging Requirements Engineering Experience

Working Group on Requirements Engineering Patterns (WGREP)
of Section 2.1.6 “Requirements Engineering” in the German Informatics Society (GI)

Kathrin Lappe[1] (Speaker), Thorsten Cziharz[2], Heinrich Dreier[3], Ralf Fahney[4], Dorina Gumm[5], Lars Hagge[1], Gerit Höhne[6], Frank Houdek[7], Jan Ittner[8], Dirk Janzen[9], Barbara Paech[10]

[1] Deutsches Elektronen-Synchrotron, Hamburg, Germany
[2] SOPHIST GmbH, Nürnberg, Germany
[3] innovative systems GmbH, Hamburg, Germany
[4] Independent consultant, München, Germany
[5] University of Hamburg, Germany
[6] Deutsche Post ITSolutions, Dresden, Germany
[7] DaimlerChrysler Research Center, Ulm, Germany
[8] method park, Erlangen, Germany
[9] Harman/Becker, Karlsbad, Germany
[10] University of Heidelberg, Germany

Abstract

The special interest group on requirements engineering[1] of the German Informatics Society has established a working group on “Requirements Engineering Patterns”. The group has demonstrated the applicability of the concept of patterns for collecting and exchanging requirements engineering (RE) experience, especially for project teams which are in the process of adopting RE. This report presents an overview of the work and a summary of the results of the working group. It describes a method for comparing case studies and extracting patterns of recurring successful requirements engineering activities, and it contains a catalogue of fourteen RE patterns which have been identified by that method. A Web-based RE pattern repository is being developed to make patterns available for project teams on the job.

[1] “Requirements engineering” describes a systematic approach to obtaining a complete project specification. It involves requirements elicitation, analysis, negotiation and verification and covers requirements management and documentation.

Table of Content

About the Members of the ”Working Group on Requirements Engineering Patterns”

Kathrin Lappe (Speaker) is an engineer in the information management department at the Deutsches Elektronen-Synchrotron (DESY). She is currently working as a requirements engineer in a plant construction project.

Contact her through the requirements engineering pattern repository, http://repare.desy.de, or by e-mail at kathrin.lappe@desy.de for questions about and feedback on the WGREP.

Thorsten Cziharz is a consultant and trainer at SOPHIST GmbH. His main focus lies in supporting clients in their system analysis with methods and procedures from requirements engineering and business process analysis.

Heinrich Dreier is a quality manager at innovative systems GmbH, part of the Harman Becker Automotive Group. In addition he was leader of the process action team that worked out the CMMI requirements engineering processes of the entire group. Innovative systems is developing navigation software for embedded systems.

Ralf Fahney offers his consulting services as an independent consultant throughout Germany and internationally since 2003. Before, he was employed at several IT consultancies. He has been engaged in requirements and project management for more than 15 years. His main focus is in logistics, tourism, banking and insurance.

Dorina Gumm is a scientific assistant and Ph.D. student at the department of Computer Science, University of Hamburg. Her research fields are requirements engineering and distributed project settings.

Lars Hagge is a senior scientist at the Deutsches Elektronen-Synchrotron (DESY) in Hamburg, Germany. He is heading the information management department, which supports process optimization and provides information systems for lifecycle management.

Gerit Susann Höhne is a graduate in information systems at DP ITSolutions GmbH. She has been collecting practical experiences in software engineering and consulting for some years. Her assignments are consulting, business process and system analysis in various IT projects and the establishment of methods in the requirements engineering domain.

Frank Houdek is a senior researcher at the DaimlerChrysler research centre in Ulm, Germany. He is currently leading a large requirements engineering research and transfer project with the aim of introducing and improving requirements engineering practices in various DaimlerChrysler automotive business units.

Jan Ittner is a computer scientist at method park Software AG in Erlangen. At present, he is working part-time on his doctorate in the field of requirements engineering.

Dirk Janzen is an electrical engineer at Harman/Becker. He has worked on database design, technical documentation, information architecture and requirements engineering for the last years in the area of automotive electronic control units.

Barbara Paech is a professor at the Institute of Computer Science at the University of Heidelberg. Her research area is Software Engineering, especially methods and processes for achieving quality with reasonable effort. She has been active in the field of Requirements Engineering for many years and has conducted many national and international industrial research and transfer projects with her group.

1 Introduction

At the 2002 annual meeting of the special interest group on requirements engineering (FG-RE) of the German Informatics Society (GI) in Ulm, four presentations reported experience of introducing and promoting requirements engineering in different projects. The projects covered [SIG04] [STT03]

- Embedded real-time systems in an automotive multi-supplier environment
- A management information system product line of a global IT company
- The planning of an interdisciplinary plant construction project
- A control software for conveyor belt systems

Although the presented projects were of different nature, it turned out that all of them were facing similar challenges, e.g.

- Stakeholders had to be convinced of the necessity and benefits of RE
- Cultural differences of stakeholders had to be accommodated
- RE methods and tools had to be established
- Suspicions against processes and transparency had to be overcome

The subsequent discussion revealed that the challenges were resolved with comparable measures, and it was concluded that sharing experience of successful RE activities would be feasible across projects and business sectors. Furthermore, it turned out that there is a demand for such an exchange, especially from small and medium organizations which do not afford dedicated resources for RE.

One year later, a working group was initiated with the goal of developing a mechanism for facilitating the exchange of RE experience between projects. The “Working Group on Requirements Engineering Patterns (WGREP)” selected the format of patterns, as they are established and successfully used by a variety of communities for collecting and exchanging engineering experience. The intended work packages included:

- Develop a concept for capturing engineering experience using patterns
- Collect RE case studies from different domains
- Identify and publish recurring observations from different case studies as patterns
- Investigate the feasibility of the approach
- Publish a final report after a one year period

The WGREP started its work in January 2004 with members from academia, industry and consulting. During nine meetings, the WGREP has developed a method for comparing practitioners’ reports and extracting patterns of recurring successful RE activities. More than a dozen RE patterns have been collected from fourteen case studies which contain descriptions of more than eighty individual project situations. Based on these results, two workshops on requirements engineering patterns were organized at the IEEE Conference on Requirements Engineering (REP’04 [HHP04]) and at the 2004 annual FG-RE meeting.

This report presents an overview of the work and a summary of the results of the WGREP. Chapter 2 introduces the developed pattern format and the procedure for collecting patterns from case studies, and chapter 3 presents the identified requirements engineering patterns. The report concludes with experience gained and perspectives for future work.

2 Collecting RE Patterns: Method and Example

2.1 The Need for Making RE Experience Accessible

Practitioners often encounter project teams or environments with no or only little experience in requirements engineering. Especially in small and medium enterprises introducing and gaining acceptance for RE methods and tools is a prevalent challenge for practitioners. Typical characteristics of such environments include:

- The customer organization is not used to the interplay of IT systems and business processes and does not have established engineering processes
- The resources for RE are limited, and customers and their users have to be convinced of the necessity of RE
- RE has to be adopted within the project as the project manager generally has no or only little influence on the higher-level customer organization

Methods and tools for handling these challenges are available, but they are difficult to access for newcomers or "part time" requirements engineers who need advice in a specific project situation:

- RE draws from various areas of knowledge, including organizational, methodological and technological subjects.
- Knowledge sources are hardly ever organized in a way that enables information access according to a project's current circumstances.
- RE experts to ask for advice are rare, and project teams too often have to find out on their own which method or tool to choose and how to implement it.

A collection of RE patterns could bridge this gap between existing RE knowledge and its successful application in different kinds of projects by improving accessibility.

2.2 Why Patterns?

Patterns are an established and well-known format for exchanging experience. They have been used in different disciplines for capturing engineering knowledge and for providing rules for generating successful engineering solutions:

- The idea originated in civil engineering in a series of books from Christopher Alexander [AIS77] [Ale79]. He observed that well-accepted buildings – he calls those buildings "alive" – have common characteristic structures, and from this observation he developed a set of rules for architects on how to construct buildings which are alive.
- The pattern format has been extended to software design by the "Gang of Four" [GHJV95]. These patterns had a tremendous success in the software design community as they formalized implicit design knowledge that had not been documented before and thus made it available for communicating insight about design problems and solutions.
- Using patterns for knowledge transfer spread to further domains, including software architecture [BMR+96], business processes [EP00] and many more. The adopted formats to describe patterns vary slightly with the habits of the particular communities.
- On the Internet, pattern collections can be found for various domains [CH01] [BE00]. The pattern homepage (see http://hillside.net/patterns/) contains exhaustive material on patterns.

Patterns are characterized by presenting engineering knowledge in self-contained units that address a particular context. They provide guidelines for good engineering, engineering being

the “art of applying scientific and methodological knowledge to practical problems”. This easily adoptable engineering knowledge is what practitioners are interested in [Kau04].

2.3 Requirements on Requirements Engineering Patterns

The requirements engineering patterns are intended for project teams that are in the process of adopting RE. It is assumed that those teams have a basic background in RE (e.g. from text books), but need to build up practical knowledge from projects. Most of the WGREP members have personal experience in working in, for or with such teams, and the requirements on RE patterns were collected based on this experience.

Patterns shall provide proven practical experience from projects. They are expected to be relevant, reliable and applicable, meaning

- Relevant: patterns should refer to “typical” project situations which are “frequently” encountered by the intended target group, rather than dealing with exotic situations.
- Reliable: patterns should provide guidelines which have proven successful under several comparable circumstances.
- Applicable: patterns should be written in an instructive and intuitive way and contain guidelines which the project managers can implement within their responsibility.

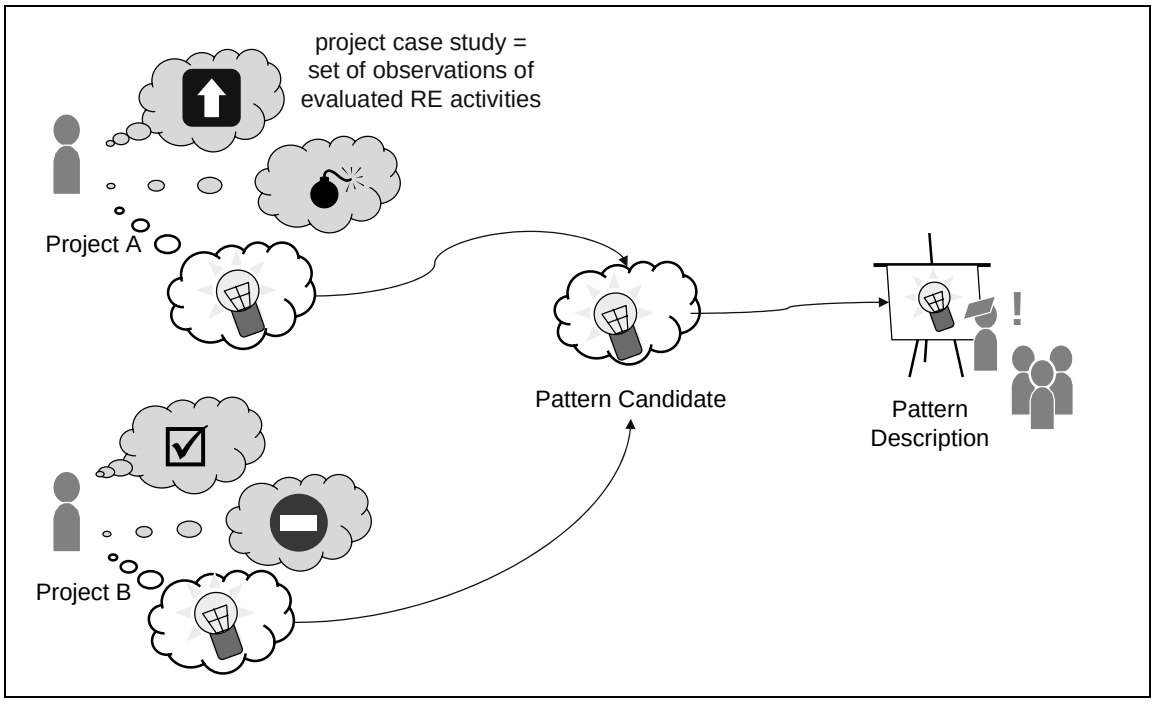

Figure 1: Similar observations from different projects yield advice for future projects.

Consequently, the working group had to develop, use and evaluate a procedure for eliciting, recording and comparing RE experience from different projects, and to propose a publication format for RE patterns. The envisioned pattern extraction procedure is built around four basic activities (Figure 1):

- Collect case studies from practitioners
- Extract relevant observations (of e.g. decisions, actions …) from the case studies
- Identify similar observations from different projects as pattern candidates
- Generalize, elaborate and publish patterns from pattern candidates

The resulting patterns are expected to shorten the RE learning curve for newcomers. Ideally, reading a pattern would correspond to a discussion with an (absent) RE expert. In particular, the following requirements were found for RE patterns:

Content: RE Patterns shall describe well-proven solutions for typical RE tasks and thus help to overcome entry barriers and to reduce initial reluctance against RE introduction.

- RE patterns shall describe solutions and experience from several projects.
- RE patterns shall focus on the introduction of RE in the different project phases..
- RE patterns shall address typical barriers for RE introduction like e.g. limited resources, limited experience, cultural diversity, or suspicions against the method.
- RE patterns shall address different facets of requirements engineering, including process, organisational or technical issues.
- RE patterns shall cover different levels of abstraction, ranging from organisational issues down to questions of tool configuration.

Form: RE patterns shall provide advice and guidance for project teams on the job, provide usable instructions and explanations on how to react on frequently occurring conflicts, and present guidelines for the concrete implementation.

- RE patterns shall provide guidelines on how to obtain solutions, rather than deliver blue prints of ready-made solutions. In addition, optional anti-patterns should describe approaches that have repeatedly delivered negative results.
- RE patterns shall help generating short-term benefits and acceptance for the proposed measures.
- RE patterns shall contain a context description and proposed solutions. They should take into account organisational, political and psychological constraints that might have impact on the usefulness and acceptance of a measure.

Access: RE patterns shall be easily accessible for practitioners with basic RE know-how.

- Practitioners should be able to select, adapt and implement single RE patterns during project set-up or execution according to their specific situation.
- RE pattern collections shall support a context based selection of applicable patterns under given project conditions. Ideally, a pattern repository offers access to other patterns addressing related issues.

2.4 The RE Pattern Structure

The structure which has been developed for the RE patterns puts special emphasis on the conditions when a pattern should be used. The structure is best explained using “A Window Place”, a frequently quoted pattern from Christopher Alexander’s “The Timeless Way of Building” [Ale79], as an example. It recommends that:

> *In living rooms where people want to be comfortable, a sitting area should be located close to the windows. In rooms where the sitting area is not placed near to the windows, people would be caught in a conflict: they would be drawn to the chairs to sit down and relax, but at the same time they would also be drawn towards the windows where the light is. Using the window place pattern would resolve and prevent the stress situation.*

The general idea of this paragraph is that patterns help resolving conflicts or stress situations. Stress stands in this context for “difficulties that cause worry or emotional tension.”

An analogy from physics is used to further analyze the pattern description. In physics, the term stress signifies “a pressure or a force that produces strain on a physical body”. It can be illustrated e.g. with a spring of a certain length and two forces, $F^{\leftarrow}$ and $F^{\rightarrow}$, which are applied

to its two ends (Figure 2). As a response to the forces, the spring will stretch by a certain distance $\Delta\ell$ and react with a stress force $F_s = k\,\Delta\ell$ (k is a constant describing the spring).

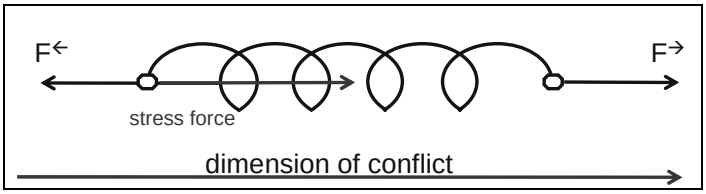

Figure 2: Spring metaphor for stress from conflicting forces.

With some imagination one could now see the spring corresponding to the person in the window place pattern: $F^{\leftarrow}$ and $F^{\rightarrow}$ would stand for the forces pulling the person towards the chairs and the window, respectively, and $F_s \sim \Delta\ell$ would be a measure for the stress being – literally – proportional to the distance between window and chairs. Being comfortable would correspond to the spring being relaxed, i.e. $F_s \sim 0$, and the pattern would simply recommend to keep $\Delta\ell \sim 0$, i.e. move the seats to the window to achieve $F_s \sim 0$.

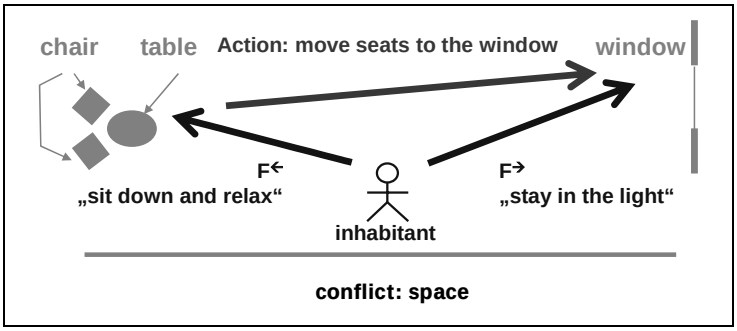

Figure 3: Christopher Alexander's "A Window Place" in the proposed pattern format.

Figure 3 illustrates the "Window Place" using the picture of forces. The stress situation has been efficiently characterized by a quality goal ($F_s \sim 0$) and two opposing forces. Taking this as general reasoning, it has been proposed that a pattern should be written as a vector [HL05],

$$\boldsymbol{P = (T, F^{\leftarrow}, F^{\rightarrow}, A)},$$

where T is a task with a quality goal, $F^{\leftarrow}$ and $F^{\rightarrow}$ are the opposing forces generating the stress force, and A is an action compensating the difference of $F^{\leftarrow}$ and $F^{\rightarrow}$, the stress. This "pattern vector" is supposed to cover the pattern essence. For the "Window Place",

- T = "design a comfortable living room"
- $F^{\leftarrow}$ = "people are drawn towards the chairs to sit down and relax"
- $F^{\rightarrow}$ = "people are drawn towards the windows to where the light is"
- A = "move seats to the window"

A generic pattern statement can be created from the pattern vector using

IF $F^{\leftarrow}$ **BUT** $F^{\rightarrow}$ **THEN** A **TO** T.

It provides a short description of patterns that helps readers to decide on the pattern's relevance and applicability for their purposes at a glance. For the "Window Place", the description reads:

IF	people are drawn towards the chairs to sit down and relax
BUT	people are drawn towards the windows where the light is
THEN	move seats to the window
TO	design a comfortable living room

The general nature of the approach is suggested by looking at two further examples from the RE pattern collection, which can be expressed in the same format without difficulty[1]:

IF	a new technology is available
BUT	stakeholders have limited technological experience
THEN	introduce a prototype
TO	create a specification for innovative functionality

IF	high-level requirements are needed for negotiation at management level
BUT	detailed requirements are needed for contracts and development
THEN	bundle requirements to features
TO	create a specification usable for different stakeholder groups

The other RE patterns further encourage the approach. For publication purposes, patterns of course need to be elaborated similar to the established pattern collections. The full versions of these and more patterns are presented in Section 3, but their general idea can already be grasped form the short form. Final RE patterns contain the following elements:

- An **abstract** providing a brief overview of the pattern
- A statement of **objective** explaining the pattern's intention
- A **context** description of the conditions when the pattern is applicable
- A **problem** statement explaining the generalized conflict
- The **forces** characterizing the (two) different influences which make up the problem
- A **solution** describing the actions to be taken for solving the problem
- An illustration of the pattern's **structure**
- A set of **instructions** and guidelines for using the pattern
- **Application areas** and **constraints** for using the pattern
- Direct **consequences** for the project from using the pattern
- **Known uses** and **experiences** gained from those projects
- References to **related patterns** and **further reading**

[1] Refer to Section 2.6 for a description of how to obtain a pattern vector from a narrative case study.

2.5 The Pattern Mining Procedure

This section describes the procedure which has been developed for extracting patterns from a set of case studies. It consists of four major activities:

1. In the first step, **case studies** are collected from real world projects. Case studies contain team members' accounts of important events and experiences from projects.
2. Next, the case studies are analyzed and reorganized into a set of **observations**. Observations describe events in the format of the pattern vector.
3. To identify patterns, the entire set of case studies is searched for identical observations from different projects. Those observations are marked as **pattern candidates**.
4. Then, the pattern candidates are elaborated into the **pattern description**, which is then made available to practitioners in a central pattern repository.

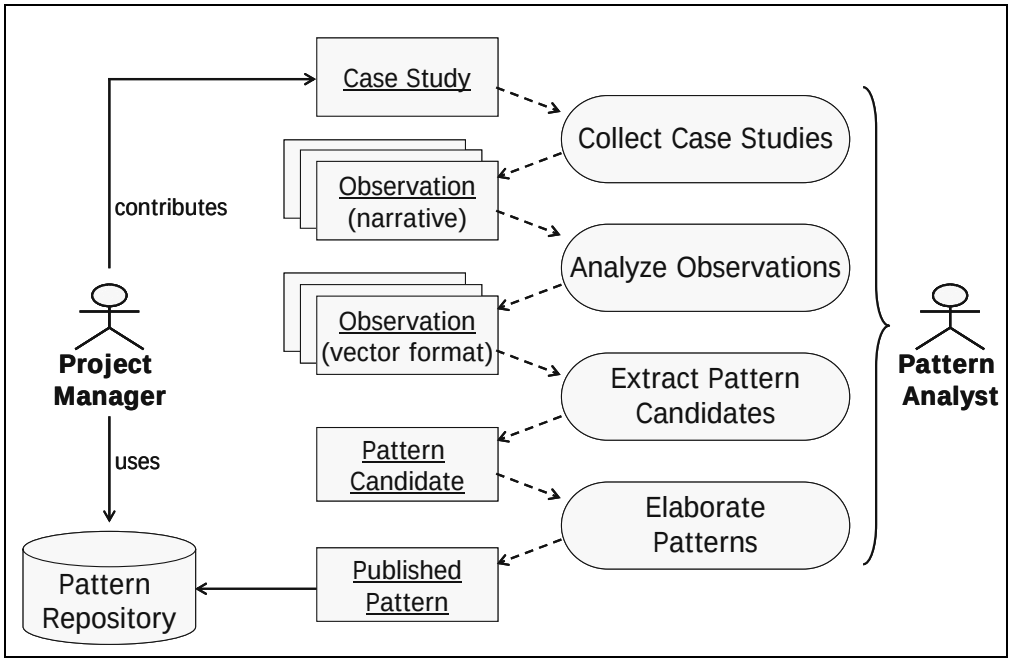

Figure 4: Overview of the pattern mining procedure.

Figure 4 illustrates the pattern mining procedure. The following sections contain brief explanations of each activity, and Section 2.6 provides an example of how to use the procedure.

2.5.1 Activity "Collect Case Studies"

Case studies should report noteworthy actions or decisions from projects. Ideally, they are reported by project managers or people from the project team who experienced the reported achievements personally. Case studies can be acquired either by interviews or by written contribution. Each case study should give an account of one project only.

A case study should typically contain a general project description and a set of observations from this project. The general part should contain characteristics of the project size and type together with the project goals, tasks, risks and constraints, as well as boundary conditions e.g. from the project organization, the communication infrastructure and the employed tools and methods. The observations should refer to typical RE tasks and contain a problem statement for the initial situation, a description of the measures taken together with some reasoning, an account of the observed results and a final evaluation. Figure 5 illustrates the structure of a case study.

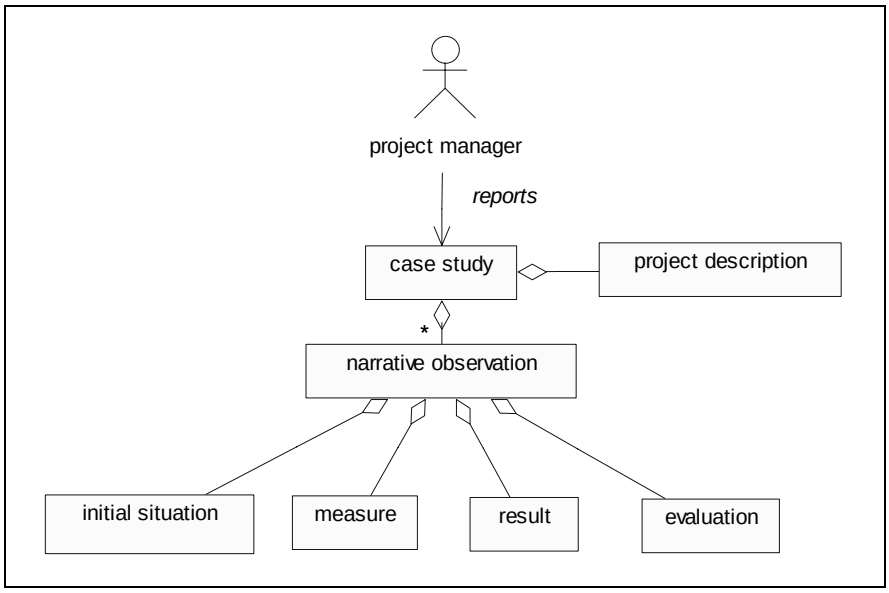

Figure 5: Structure of a case study.

Case studies are best started with the general project information, followed by a set of observations. The decision of which observations should be included in a case study should be left to the authors. For pattern extraction observations are useful which

- Describe activities or decisions which have been relevant to the course of the project
- Explain (repeated) changes of original plans or strategies
- Report unusual successes or failures
- Deal with strong external influences
- Are considered important by the observer or for a reader

2.5.2 Activity "Analyze Observations"

In preparation for pattern extraction the observations need to be rephrased in the pattern vector format. At first glance the process is relatively straightforward – the task description and the action are usually obvious, and the forces can be derived from the problem statement. However, pattern extraction searches for comparable observations from different projects, i.e. for identical vectors from different case studies. Such vectors will only occur if the vector components are properly classified, hence it has to be made sure that controlled sets of values are used for *T*, *F* and *A*. Figure 6 illustrates the relation between observations and pattern candidates.

The tasks, forces, and actions chosen for the RE patterns in this report are introduced in Section 3 preceding the pattern collection.

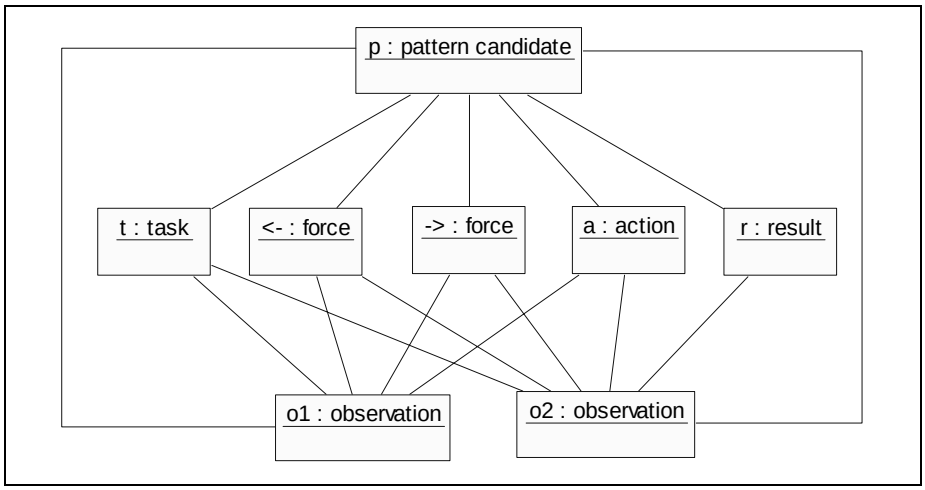

Figure 6: Patterns use the same basic elements as observations.

2.5.3 Activity "Extract Pattern Candidates"

Once the available set of observations has grown, a search for recurring entries can be started, and vectors occurring at least twice in the set should be marked as pattern candidates. The number of occurrences of a pattern candidate vector can be seen as a measure for the pattern's quality in terms of reliability and relevance.

It is essential that the pattern candidates are verified with the authors of the corresponding observations to make sure that the essence of the observation has not involuntarily been changed in its meaning during the analysis, when reformatting the observation into a vector.

2.5.4 Activity "Elaborate Patterns"

Last but not least, the pattern candidates have to be elaborated into detailed and instructive descriptions which are suitable for the intended target group. The list at the end of Section 2.4 provides an overview of the elements which should be included in the pattern's version for publication. Writing patterns turns out to be an iterative procedure that requires feedback from the authors of the observations, other RE experts, and the intended target group.

2.6 An Example

This section contains a walk-through of the proposed pattern analysis procedure. It demonstrates how observations are extracted from case studies and how they are written in the pattern vector format, and it shows how a pattern candidate – the use of prototypes for specifying innovative functionality – is identified from a set of observations.

2.6.1 Collecting and Analyzing the First Case Study

In this example, a project manager who is interested in exchanging RE experience has agreed to an informal interview about a recently completed system development project. After some general information, the interview turns to specific observations from the project. The interview relies on the project manager to select decisions or actions which were relevant to the project:

> *"I'm talking about the development of an intranet company information system for 1000+ infrequent users. In this project, user acceptance was a critical success factor …*
>
> *One of the important things in the project was the introduction of an exploratory user interface prototype in a very early project phase, while we were still envisioning and defining the system.*

For the tech people, it has been obvious from the beginning that the system would benefit from advanced graphical user interfaces, including interactive maps and virtual reality mock-ups. The target user groups on the other hand had less experience with Web technologies. They were used to native systems with forms and folder trees and found it hard to imagine other front ends.

Once we had the prototype, we were able to demonstrate the technological capabilities to different stakeholders. The stakeholders could understand the potential benefits of the new technology, and they could imagine how they would use the information system in their daily work. As a consequence, they started to request further innovative functionality from the system which they did not expect to be feasible before, and in general they specified with better imagination and motivation.

Another important thing was …"

The case study is now restructured for further analysis. It is assumed that the case study consists of a set of observations, where each observation is an episode from the project. The observations should include details of the initial situation, an explanation of the selected approach and why it was chosen, and a description of the obtained result. Table 1 collects these elements in a compact format:

Initial Situation	A specification for an information system using innovative user interface technology had to be created. The stakeholders had only limited technological experience and thus difficulties in envisioning new functionality.
Approach	An exploratory user interface prototype of the system was created and introduced to the stakeholders.
Result	Evaluating the prototype helped the stakeholders to understand the benefits they can expect from the new technology. They specified with better imagination and motivation.

Table 1: Observation *O1* from case study "information system".

The pattern vector can now be extracted to make the observation comparable to other case studies. The RE task, *T*, is usually part of the description of the initial situation, which for practical reasons should be supplemented with a project identifier, *Id*.

Initial Situation	A **specification** for an **information system** using innovative user interface technology had to be **created**. The stakeholders had only limited technological experience and thus difficulties in envisioning **new functionality**.

Table 2: Retrieving the RE task from observation *O1*.

Table 2 repeats the description of the initial situation and highlights the relevant keywords, yielding

Id = "information system"

T = "create a specification for innovative functionality"

In the same way, the forces can be retrieved. The forces are describing the conflict which has been resolved by the described action, so they should also be part of the initial situation. Highlighting again the relevant passages of the text (Table 3), one finds

$F^{\rightarrow}$ = **"**a new technology is available"

$F^{\leftarrow}$ = "stakeholders have limited technological experience"

Initial Situation	A specification for an information system using **innovative** user interface **technology** had to be created. The **stakeholders** had only **limited technological experience** and thus difficulties in envisioning new functionality.

Table 3: Extracting the forces $F^{\rightarrow}$ and $F^{\leftarrow}$ from observation *O1*.

$F^{\rightarrow}$, the availability of a new technology, is strictly speaking not a force, but rather a fact, which leads to the force that the project team tends to use the new technology for achieving optimum results. In the same way, $F^{\leftarrow}$ is the fact that stakeholders have only limited technological experience (or to put it nicer: stakeholders are experienced and trusting in conservative technology), which leads to the force that the project team tends to use conservative technology to achieve better involvement and acceptance of the stakeholders. The forces could be rephrased into

$F^{\rightarrow\prime}$ = "use leading-edge technology for optimum results"

$F^{\leftarrow\prime}$ = "use conservative technology for optimum involvement of inexperienced user"

Both forms of expression are suitable for pattern analysis: $F^{\rightarrow\prime}$ and $F^{\leftarrow\prime}$ are emphasizing the conflict which has been resolved by the action, while $F^{\rightarrow}$ and $F^{\leftarrow}$ are describing the situation in which it has been decided to take the action. For this analysis, the fact-oriented description is used in the pattern vector, as it seemed to be the practitioner's natural choice, while the pattern description contains both viewpoints.

Approach	An exploratory user interface **prototype** of the system was **created** and introduced to the stakeholders.

Table 4: Extracting the action A from observation *O1*.

To complete the pattern vector, the action has to be stated. It can usually be taken directly from the approach described in the observation (cf. Table 4),

A = **"**create a prototype"

Result	Evaluating the prototype **helped** the stakeholders to understand the benefits they can expect from the new technology. They specified with **better** imagination and motivation.

Table 5: Record the achieved result R of observation *O1*.

To ensure that only actions with the same outcome are considered when searching for pattern candidates, the pattern vector could be extended to include the observed result (Table 5), yielding

R = **"**👍"

The vector is now complete, and the observation can be written as,

O1 = **(***Id, T***,** $F^{\rightarrow}$, $F^{\leftarrow}$, *A***,** *R***)**

= (information system, create a specification for innovative functionality,
a new technology is available,
stakeholders have limited technological experience,
introduce a prototype, 👍).

Conversely, the observation can be regenerated from the pattern vector by e.g. building a sentence along "In an *Id* project it has been observed, that when $F^{\rightarrow}$, but $F^{\leftarrow}$, *A R* to *T*.":

"In an information system project it has been observed, that when a new technology was available, but the stakeholders had limited technological experience, introducing a prototype helped to create a specification for innovative functionality."

2.6.2 More Case Studies

Subsequently, more case studies were collected from different projects. As the case studies were analysed, more observation vectors became available. Two observations which address comparable tasks as ***O1*** in the previous section are briefly introduced below in tabular and vector format:

Initial Situation	A legacy system had to be migrated to enable business process optimization. The domain experts which had to specify the new system were mentally tied to the legacy system. Instead of envisioning new processes and functionality, they were describing the old system.
Approach	A metaphor from a related domain was created to discuss essential business processes independent of the application.
Result	The metaphor helped to break up habitual ways of thinking and shifted the focus from the system to the process. Useful new ideas and terminologies were created.

Table 6: Observation *O2* from case study "legacy system migration".

The first example refers to a legacy system migration where the key users were mentally too tied to their old system to be able to envision new ways of working (cf. Table 6). The forces refer to the same basic conflict as in ***O1***, if the term "technology" is understood in a way that it includes "processes": The project team needs to adopt a new technology to achieve the intended optimization, but at the same time it should stay with conservative technology according to the stakeholders' experience. The pattern vector reads[1]

O2 = (legacy system migration, create a specification for innovative functionality,
a new technology is available,
stakeholders have limited technological experience,
use a metaphor for envisioning, 👍)

The third observation (Table 7) was made in a small enterprise, which was deploying a new production line.

Initial Situation	A specification for the process control system of a new production line based on innovative technology had to be created. The staff had only limited experience with this technology and thus difficulties in detailing the specification.
Approach	A prototype environment for the intended production process was created to learn how the technology works.
Result	Working with the prototype helped to understand the technology and its opportunities and threats. With this knowledge the specification improved.

Table 7: Observation *O3* from case study "process control system".

[1] The definition of the forces is not unambiguous but rather soft, for discussion refer to Section 4.2.

Specifying the process control system required handling an available new technology, thus the staff had to be trained in a prototype environment prior to specification. The observation vector reads

O3 = (process control system, create a specification for innovative functionality,
a new technology is available,
stakeholders have limited technological experience,
introduce a prototype, 👍)

2.6.3 Extracting a Pattern Candidate

The observations can be combined into a common database to enable pattern extraction. Collecting all the elements of the previous sections yields

O1 = **(***Id1***,** *T1***,** *F1***,** *F2***,** *A1***,** 👍**),**

O2 = **(***Id2***,** *T1***,** *F1***,** *F2***,** *A2***,** 👍**),**

O3 = **(***Id3***,** *T1***,** *F1***,** *F2***,** *A1***,** 👍**),**

where

Id1 = **Project** information system,

Id2 = **Project** legacy system migration,

Id3 = **Project** process control system,

T1 = **Task** create a specification for innovative functionality,

F1 = **Force** a new technology is available,

F2 = **Force** stakeholders have limited technological experience,

A1 = **Action** introduce a prototype, and

A2 = **Action** use a metaphor for envisioning.

Identifying patterns requires discovering recurring observations from different projects, i.e. vectors containing identical task, force and action components. In this example, **O1** and **O3** relieved a similar conflict with the same measure in two different projects, which is expressed in the pattern candidate ***P1***,

P1 **= (***T1***,** *F1***,** *F2***,** *A1***,** 👍**)**.

The resulting pattern candidate recommends letting the technology leaders build prototypes (e.g. mock-ups, test environments, simulations…) when new technologies become available as an enabler for product innovation. User requirements are needed to assess the potential benefits for the end user, but without prototypes stakeholders are lacking the ability to envision new product capabilities and potential applications in their environment. In short:

IF	a new technology is available
BUT	stakeholders have limited technological experience
THEN	introduce a prototype
TO	create a specification for innovative functionality.

Observation ***O2*** has been addressing the same conflict as ***P1***, and it may now be speculated whether a prototype could have been used in project *Id2* as well, or if using a metaphor instead of a prototype would have had the same effect in projects *Id1* and *Id3*; but as patterns are intended to capture reliable experience and are therefore required to be based on at least

two independent observations, only the recommendation to introduce a prototype can be extracted as a pattern candidate from the data.

Figure 7 summarizes the discussion of the prototype pattern, showing the observable facts, the conflicting forces because of the different skill levels of the technology leader and the project team, and how the action of introducing a prototype can increase the project team's level of technological experience and thus resolve the conflict.

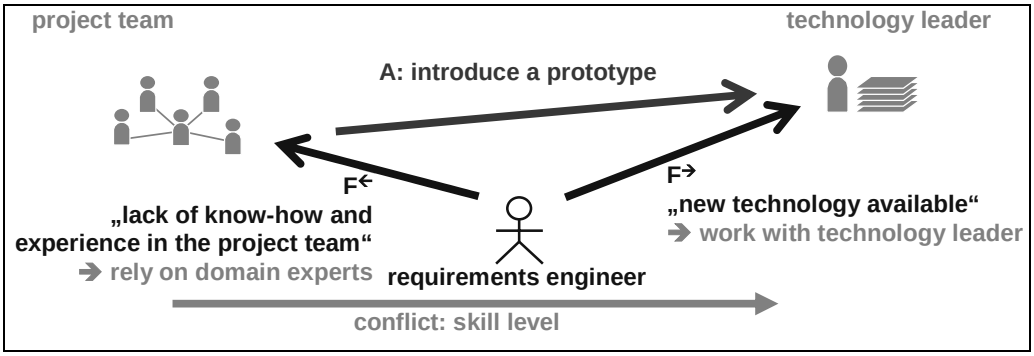

Figure 7: Pattern "Use Prototypes for Specifying Innovative Products" drawn in the conflict space.

2.6.4 Elaborating the Pattern Description

For publication, the different pieces of information from the previous sections have to be collected, elaborated and illustrated according to the guideline in Section 2.5.4. A complete description of the resulting pattern can be found in chapter 3.2.

3 A Collection of Requirements Engineering Patterns

Using the method described in the previous section, the WGREP collected fourteen case studies containing more than eighty observations. Fourteen pattern candidates have been identified, half of which have already been elaborated into patterns. This section first introduces the pattern candidates. After that the full versions of the finalized RE Patterns and summaries of the remaining pattern candidates are presented.

3.1 Introduction and Overview

This section presents an overview of the RE patterns, their relations and their connections to roles and concepts in the RE domain.

3.1.1 Glossary of Terms

Table 8 presents the main roles and concepts with a short explanation of how they were used by the working group.

Term	Description
Analyst	RE role which is responsible for understanding the requirements, their overlaps and their conflicts.
Author	RE role which is responsible for writing →specification documents
Client	person or organization which requests and pays for (parts of) the →product that is delivered by a →supplier on a contractual basis
Developer	engineering role which is responsible for creating the →product (i.e. the implementation)
Domain Expert	person with knowledge about the application environment of a →product, i.e. a special kind of →stakeholder
Facilitator	RE role which is responsible for initiating, maintaining, monitoring, and concluding structured group activities in the RE process
Management	superordinate organization to the →project which acts as decision maker, sponsor and provider of resources for the project
Product	deliverable good or service that is developed in a →project, in the software engineering domain usually a system
Product Specification	set of →requirements that describe the →product, frequently formatted in a specification document
Project	a planned undertaking or organized set of services designed to deliver a →product using engineering methods
Project Manager	engineering role which is responsible for managing a →project, i.e. planning and organizing activities required to develop →products requested by →clients
Project Team	set of persons assigned as resources to the →project who take different engineering roles to fulfil the project goals
Reader	person who uses the →specification documents and accesses →requirements in order to get an understanding of the intended →product, usually affiliated with a →stakeholder group
Requirement	describes characteristics that the →product must possess in order to meet the →stakeholders' needs and to be acceptable for the →client

Term	Description
Requirements Engineer	RE role which is responsible for coordinating all the RE activities within a →project.
Requirements Manager	RE role which is responsible for managing changes to the requirements [KS98]
Stakeholder	person or group or organization which will be affected by the →product and which has a direct or indirect influence on the →requirements. [SS97]
Supplier	person or organization which delivers parts of the →product to a →client on a contractual basis (synonym: contractor, provider)
Technology Leader	Person or organization with knowledge on leading edge technologies which contributes to the →specification, i.e. a special kind of →stakeholder
Tester	engineering role which is responsible for checking and certifying →product conformity with the →product specification
User	person from the →client organization who will work with the →product in its business processes

Table 8: Project roles used in the pattern descriptions.

The relations between the key terms are illustrated in Figure 8, while Figure 9 adds an overview of those roles that actively perform project tasks and therefore are concerned with engineering patterns.

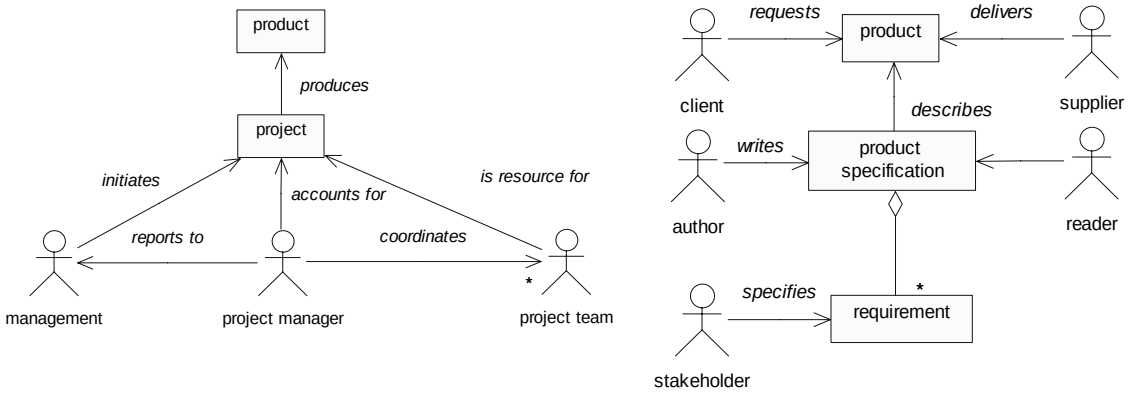

Figure 8: Roles related to project management (left) and product specification (right).

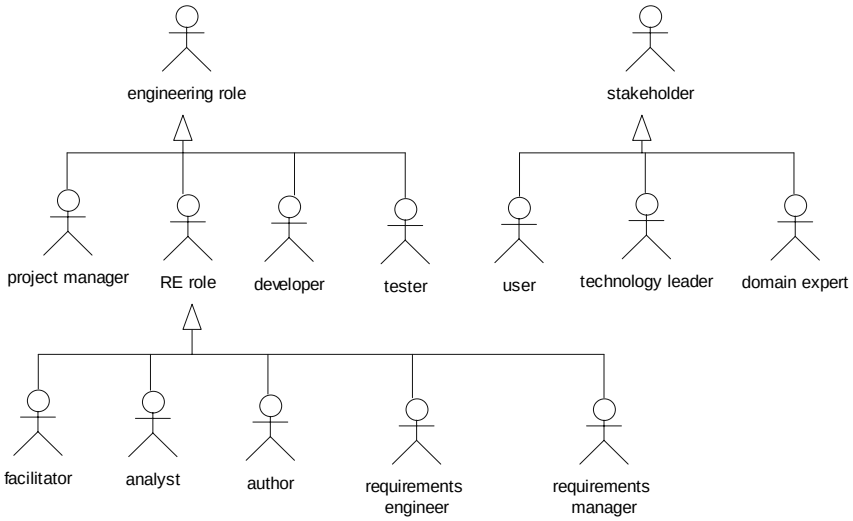

Figure 9: Roles which perform tasks within a project.

3.1.2 Patterns Summaries

The RE patterns of this collection address methodological, technical and organizational issues at different project stages. The following summary offers some orientation in the pattern collection by structuring the patterns around some key RE activities [Som05]: eliciting, analyzing, validating and verifying, negotiating, managing, and documenting requirements. In practice, the assignment of patterns to these activities is not exclusive, but in this overview patterns are only once summarized in the most appropriate category.

Patterns marked with an asterisk* are pattern candidates which are not yet fully elaborated; for those patterns, a partial version consisting at least of the pattern vector and the known uses is provided.

Requirements Elicitation covers acquiring the appropriate stakeholder requirements.

- **Use Prototypes for Specifying Innovative Products** (Section 3.2) deals with the problem that users cannot envision the benefits of new technologies for their problem at hand.
- **Evaluate Existing Documentation*** (Section 3.3) recommends to capture the knowledge, experience and requirements of stakeholder groups which are no longer directly accessible from documents they have been using – valuable e.g. for migration from legacy systems.
- **Bundle Requirements to Features** (Section 3.4) describes how to structure a set of requirements in a way that it is usable by different target groups, e.g. management who is interested in high-level descriptions, and developers needing technical details for implementation.

Requirements Analysis covers understanding the requirements, their overlaps and their conflicts.

- **Use Requirement Index Cards** (Section 3.4) proposes writing and classifying requirements as if they were kept on individual index cards. Index cards are natural to many persons in the project team, and the metaphor of index cards eases stakeholder involvement.
- **Write Reusable Common Requirements** (Section 3.6) suggests relating generic requirements to concrete products instead of managing multiple requirements versions for different products.

- **Build an Abstract Model to Integrate Fragmented Specifications*** (Section 3.7) states that if several different sources of information are available, their essence should be analyzed by creating a common (UML) model.
- **Provide Statements of Objective*** (Section 3.8) recommends capturing the reason and objective why a requirement has been raised, which is a helpful ingredient for structuring a requirements specification.

Validating and Verifying Requirements ensures that the stakeholders really get what they wanted.

- **Generate Approval Checklists** (Section 3.9) describes a way to document fulfillment of reused requirements.
- **Detail the Specification by Writing Test Cases** (Section 3.10) recommends that if a specification turns out to need improvement once a project is under way, an alternative to revising the specification is to provide test cases in addition to the requirements.

Requirements Negotiation ensures that conflicting views are reconciled and a consistent set of requirements is created.

- **Organize Specification along Project Structure*** (Section 3.11) recommends taking advantage of an existing project organization for defining specification responsibilities.

Requirements Management controls changes in the requirements that will inevitably arise.

- **Employ a Requirements Engineer as a Care Taker*** (Section 3.12) describes a measure to ensure that RE tasks are not neglected in daily project business.
- **Use a Central Issue List*** (Section 3.13) suggests tracking changes very early in a defined location.
- **Synchronize Change Requests** (Section 3.14) proposes a simple change process to stay in control of efforts that arise from changes.

Last but not least, **Documentation** concentrates on writing down the requirements in a way that stakeholders and developers can understand.

- **Create a Specification Guideline by Tracking How an Analyst Works** (Section 3.15) proposes a method to transfer modeling skills from method experts to domain experts for improving their specification skills.

3.1.3 Pattern Relations

This section gives an overview of the relations between the presented patterns. Figure 10 shows the connections between patterns and project roles – the assignments are kept to a minimum in order to maintain the diagram's readability – while Figure 11 shows the major relations between the patterns. Both figures are in UML notation [OMG03].

Figure 10 illustrates for the different project roles from which patterns they would benefit most. Project roles are represented as actors, patterns are shown as use cases, and relations define which actor should invoke the use cases, i.e. make use of the patterns.

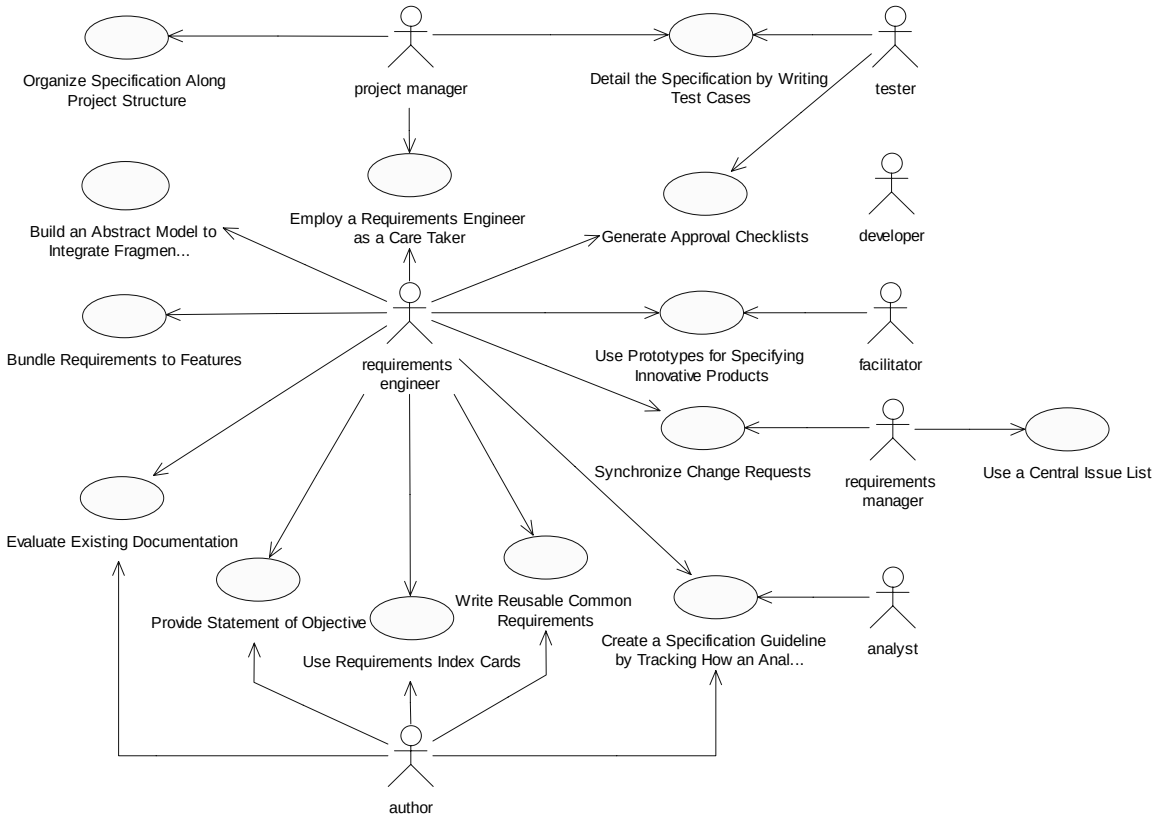

Figure 10: RE project roles and associated patterns.

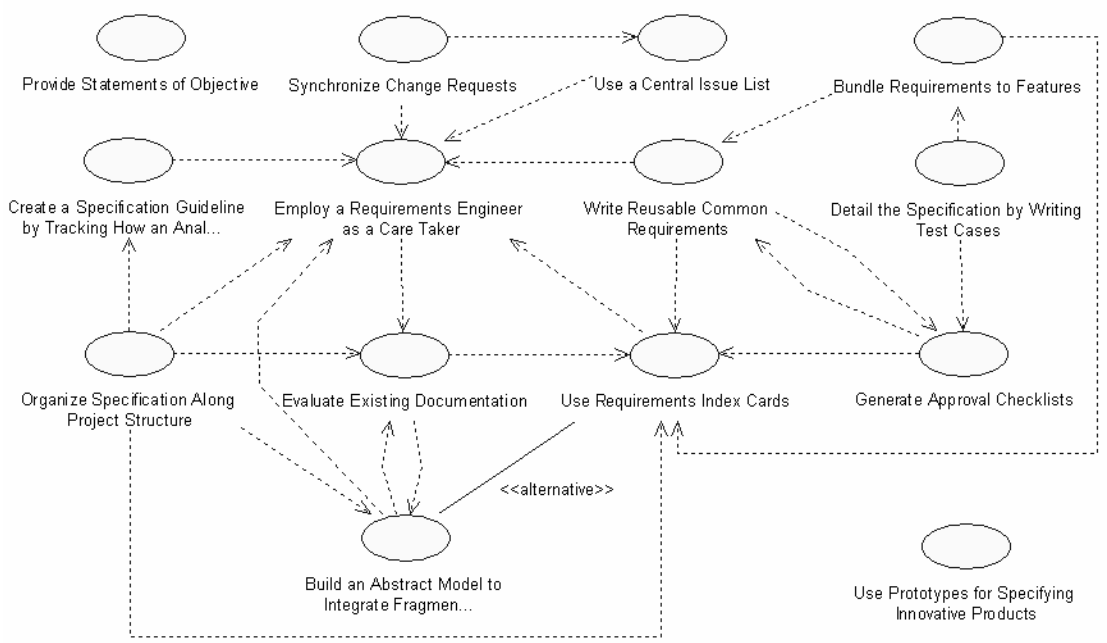

Figure 11: Associations between the patterns presented in this report.

Figure 11 shows the relations between patterns which are explained in the elaborated pattern descriptions. Pattern relations are represented as dependencies, implying that one pattern depends on another pattern being implemented as a basis. Two patterns are not related to other patterns - this is expected to change as the pattern collection grows. Further investigation in pattern relations might then yield a pattern language [AIS77] which enables a variety of pattern compositions to suit the individual project needs.

3.2 Use Prototypes for Specifying Innovative Products

REQUIREMENTS ENGINEERING PATTERN

Use Prototypes for Specifying Innovative Products

Designing innovative products requires stakeholders with strong imagination and good technical understanding. This pattern recommends using prototypes for improving the project team's knowledge and experience in state-of-the-art technologies.

Figure 12: Prototype used for specification (left) and final product (right).

Objective. User requirements for innovative functionality shall be elicited.

Context. The availability of new technologies is an enabler for product innovation, and user requirements are needed to assess the potential benefits for end users and to understand the underlying business case.

Problem. Stakeholders lack the ability to envision new product capabilities and potential applications in their environment. In case of technology-driven innovation, this is frequently caused by the fact that technological know-how and experience has not yet reached the stakeholders at the time of requirements elicitation.

Forces. The forces in this pattern deal with the tendency of people to be driven by and rely on personal experience:

- A new technology has become available for product development.
- Target users and key stakeholders have limited know-how and experience of the technology

Solution. Let the technology leaders build prototypes (e.g. mock-ups, test environments, simulations, animations) in parallel with envisioning and requirements elicitation. Use the prototypes to let stakeholders build up know-how, gather experience and grow confidence in the capabilities of new technologies.

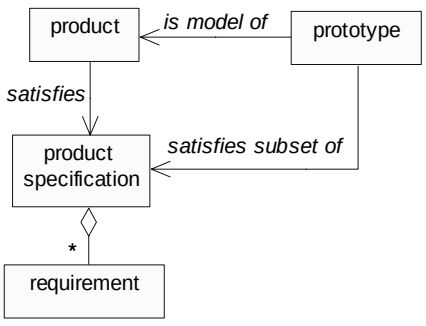

Figure 13: A prototype satisfies a subset of the product specification.

Structure. Specifying innovative products requires both an understanding of technological possibilities and users' necessities. A prototype acts as intermediary between the two worlds.

Instructions.

A prototype can be defined as a partial implementation of a system which allows investigating and deciding between design alternatives, addressing for example questions of acceptance, technical feasibility or performance. Prototypes follow different goals, e.g. exploratory prototypes focus on vision sharing and requirements elicitation, while evolutionary prototypes address primarily architecture design. Prototypes can be of different types, including mock-ups, functional prototypes and pilot systems. Most important, prototypes are to be thrown away.

Prototype development is specific for the type of prototype which is to be constructed and cannot be generalized in a brief instruction, but an example for building and using an exploratory user interface prototype can help to describe the general idea. It can be summarized in five major steps:

1. Create a UML overview business model which captures the essential (major) business processes and workflows.

2. Create a UML overview system model which defines the system components which are necessary to support the business, and which contains some scenarios for exemplifying the system usage. The scenarios should contain elements which are corresponding to forms in the user interface.

3. Create prototypes for the user interface elements using HTML or PowerPoint to simulate the behavior and the look and feel of the GUI. Establish links between the forms so that their order of appearance corresponds to the scenarios of the previous step.

4. Demonstrate the prototype to several stakeholders, obtain their feedback, and adapt the UML model and the prototype iteratively.

5. Evolve the UML model to provide further documentation like e.g. the design specification, test cases and material for tutorials. Establish agreement with the stakeholders on the prototype, then use it as a basis for implementation and quality assurance.

The method for building a concrete prototyping has to be chosen according to the current project conditions. It is essential to establish the goal, the target group and intended application of the prototype first. In case of functional prototypes it has to be ensured that the prototype is not going to be used for any productive work.

Application Areas. This pattern has so far been observed in product development projects.

Constraints.

- Prototypes require financial and time resources.

Consequences.

↗ Project decisions are supported by stakeholder experience and feedback.

↗ Potential product acceptance can be estimated before investments are made.

↗ Prototypes enable improved cost estimates of later design and implementation.

- ↘ Prototypes might focus the attention of stakeholders to the demonstrated capabilities only and thus inhibit the development of further visions and alternative solutions.
- ↘ A working prototype might also suggest capabilities which later cannot be achieved (e.g. if scaling rules were ignored when creating the prototype).

Experiences.

- Efforts for building a prototype are hard to estimate a-priori. Customers might be reluctant to finance prototypes as they require resources already during project initiation. It should therefore be attempted to acquire resources for prototyping outside of the project, e.g. from R&D departments.
- On the other hand, building early prototypes is not a big additional load to the overall project budget: prototypes would become necessary at later stages in any case for feasibility studies, cost estimates etc.
- Using prototypes becomes more efficient if they are based on scenarios from the stakeholders' environments.
- Functional prototypes are bearing the risk of being taken into and remain in production, this way evolving into long-lived systems without any proper previous architecture design.

Known Uses.

- **Introduction of COTS-based Information Systems**

  A technology-driven group developed a vision for an innovative user interface to an information system. The vision was difficult to communicate and share among system users and providers as they were unfamiliar with the technology. Partially, the vision was not understood as its description was on a high level of abstraction, other parts were not trusted to work as they seemed too far-fetched. On this basis it was impossible to elicit user requirements. Instead, a diploma student was employed to create a toy version of the envisioned user interface. The prototype interface could be demonstrated to expert users and system providers, who then joined the discussions of feasibility and potential benefits of the initial vision.

- **Control System for Innovative Production Process**

  A tool manufacturer for mechanical workshops intended to develop a new machine which was based on an innovative manufacturing technology. A control system for the envisioned machine had to be developed. The project team succeeded in creating a functional specification for the control system, but was unable to define the system details due to lack of experience with the technology and the corresponding manufacturing process. A test environment was created in which the project team could manually perform and investigate the envisioned manufacturing process. After this, the control system specification could be detailed.

Related Patterns. none.

3.3 Evaluate Existing Documentation

REQUIREMENTS ENGINEERING PATTERN *CANDIDATE*

Evaluate Existing Documentation

Requirements should be written by their stakeholders, but the stakeholders are sometimes hard to access. This pattern offers an alternative way of requirements elicitation from users' viewpoints without contacting stakeholders.

Objective. Acquire complete requirements from the stakeholders' viewpoints.

Context. A specification should be created for a project with several different groups of stakeholders. Some stakeholders are unavailable for requirements elicitation, but their demands and experience should be captured in the specification according to their viewpoint.

Problem. Requirements should be phrased by the stakeholders, but for some questions there are no stakeholders available in the project team. Finding and involving stakeholders could delay the project.

Forces. The forces address the availability of stakeholders and this way the authenticity and reliability of the requirements:

- Requirements should be elicited only directly from stakeholders to capture their viewpoint and to avoid misunderstandings and interpretations by third parties.
- Requirements should be elicited according to the envisioned system structure and use cases to ensure complete coverage in the specification.

Solution. Evaluate existing documentation and derive requirements from the described features and experiences, especially for topics of interest which are not sufficiently discussed by stakeholders. User manuals and tutorials of previous products are especially valuable sources as they provide insights from a user's perspective. Other sources include legacy systems, design descriptions and output produced from legacy products.

Known Uses. The pattern has been observed in projects which can build on legacy systems. In a development project of a new multi-purpose printer, scanner and fax machine, product descriptions and user manuals from preceding separate products were analyzed to capture their usability and design experience. In a project which introduced a new integrated business system, manuals and materials of earlier systems were evaluated to identify the key use cases.

Related Patterns.

- "Build Abstract Model to Integrate Fragmented Specifications" or "Use Requirement Index Cards" to integrate the information content of different documentations.
- Use this pattern to represent stakeholder groups which are not accessible for specification tasks when you "Organize Specification Along Project Structure".

Credits. The pattern was discovered during discussion at the REP04 workshop [HHP04].

3.4 Bundle Requirements to Features

REQUIREMENTS ENGINEERING PATTERN

Bundle Requirements to Features

If a specification needs to contain high-level information for managers as well as technical details for developers, then the specification should be decomposed using features, i.e. essential characteristics of the intended project.

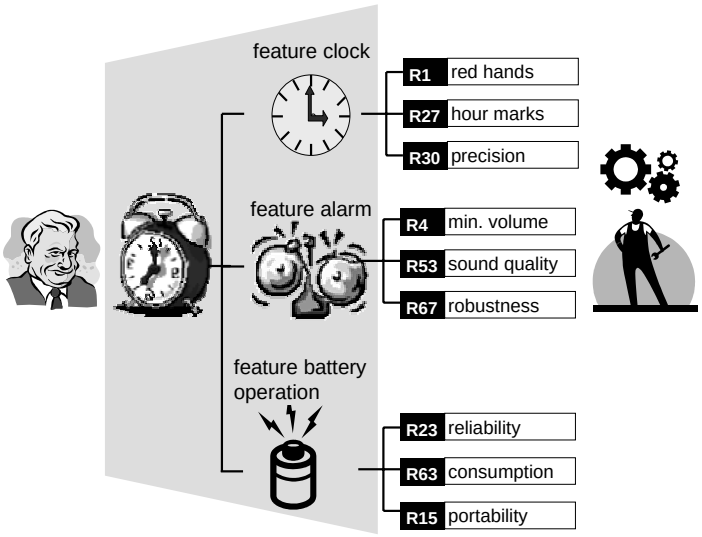

Figure 14: Shielding the detailed requirements from stakeholders by bundling them to features.

Objective. A specification shall be created and maintained that suites different reader groups who need access to information at different levels of detail, e.g. readers from management and readers from development.

Context. Requirements have been collected and documented and now have to be structured and organized according to the readers' needs.

Problem. Management and newly involved stakeholders need high-level elaborate requirements to understand the vision in a first step while developers need detailed technical requirements. The specification has to be structured in a way that it can satisfy all its readers.

Forces. The pattern addresses the different levels of abstraction in requirements specification.

- A general high-level specification with elaborate requirements has to be provided because it helps to build a common vision, eases communication and enables efficient decision making.
- A more specific detailed specification has to be elaborated to provide a reliable basis for contracting and to guide the developers.

Solution. Identify the essential characteristics of the product under development as product features and summarize them in a feature list. Bundle existing and new requirements to the according features. The result is a hierarchical structure that decomposes the product into features that act as a bridge head to more detailed requirements.

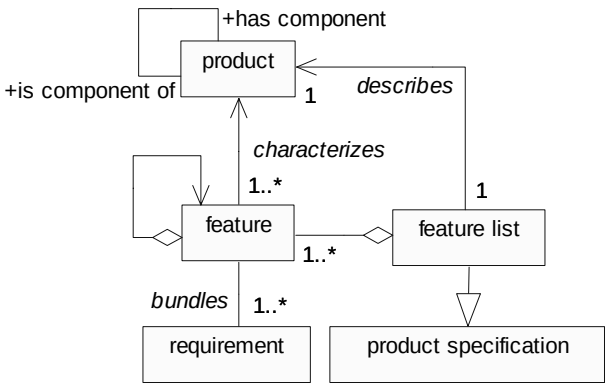

Figure 15: Using features to bundle requirements.

Specification readers can thus access different requirements levels at their command. A complete feature list provides an appropriate project description for communication.

Structure. A product is broken down into a number of features. Every feature bunches together a number of more detailed requirements. Thus the feature represents an intermediate level of detail for organizing the specification.

Instructions. The notion of a feature is not commonly defined, but in general a feature is characterized as:

- A product property which can be experienced by the users
- A static product element which can be tested for presence
- A negotiation topic that impacts the system and its costs

A (functional) feature provides added value to the system, i.e. it makes a considerable difference whether the feature is present or not. In this sense it is significantly more than, for example, a condition.

If properly introduced, features provide a vocabulary for describing a product. It could be used within the project team for negotiation and communication, as well as outside the team, for example for marketing purposes. For this reason, one of the first steps of introducing features is to create a list of meaningful, self-explaining keywords which are suitable for describing the product. Associating requirements with these keywords will then explain their meaning and establish a common understanding of the product properties.

The list of features with associated requirements can be hierarchically structured and developed: requirements which are detailing high-level features can in turn be detailed by further lower-level requirements, thus treating requirements like features.

If an existing requirements specification is available, identify features by grouping the requirements and modify the specification structure accordingly.

Ideally, feature lists are based on existing lists from previous comparable projects (e.g. in product lines) and are made available to follow-up projects.

Application Areas. Feature lists are beneficial to

- Projects with many stakeholders that have to read and understand the specification and therefore need access points to it; or
- Projects which deliver product families and thus can reuse features, which were identified once, in follow-up projects.

Constraints.

- Handling feature lists is facilitated by a requirements database that stores the relation between features and requirements.
- Some experience with the domain is required within the project team to identify the correct features.

Consequences.

↗ Feature lists give an early idea of the system under development.

↗ A feature is a term with some agreed co-notation, which can be used as a controlled vocabulary (or an "index") to more detailed specifications. This way, features serve as a short hand for communication purposes.

↘ Features may omit important details and thus diverge from their original meaning.

↘ Maintaining features and their relations requires configuration management efforts.

Experiences.

- A feature list can be used to define the product scope by adding or deleting new features.
- Feature lists are a helpful tool for release planning. Features can be assigned to product samples or milestones.
- Feature lists can be used for comparison of bids.

Known Uses.

- **Automotive multi-supplier environment**

  A feature list was created based on experience from previous projects. The list was used to define the project scope internally in the first place and then to ask for external supplier bids. Both negotiation processes were tracked by assigning open issues to the impacted features. More detailed requirements were bundled to features that were used to plan implementation and tests. The feature list was reused in subsequent projects to speed up the process.

- **Introduction of COTS-based Information Systems**

  To suit contractual needs for the introduction of a COTS-based Facility Management System (FMS), the specification had initially been organized to reflect the different system components. When the implementation team worked according to the specification, it became apparent that the specification put too much emphasis on technical aspects. It could not transport the vision of the intended system usage to the development team, and system tests according to the specification were not capturing usability and user acceptance. The specification was therefore re-structured: features corresponding to portions of system functionality were introduced, and requirements were assigned to (one or more) features. The features were successfully used for release planning and provided the guidance which was needed for system design and test.

Related Patterns.

- "Use Requirements Index Cards" may help to classify requirements in order to relate them to features.
- "Write Reusable Common Requirements" as a basis for making feature lists reusable.
- Use features for recording the relation of requirements and test cases to "Detail the Specification by Writing Test Cases".

Further Reading.

[KCH+90] K. Kang, S. Cohen, J. Hess, W. Novak, A. Peterson,, *Feature-Oriented Domain Analysis (FODA) Feasibility Study,* Technical report CMU/SEI-90-TR-021, Software Engineering Institute, Carnegie Mellon University, 2000.

3.5 Use Requirements Index Cards

REQUIREMENTS ENGINEERING PATTERN
Use Requirements Index Cards

If requirements contributions from different stakeholders need to be unified in style and format, this pattern recommends using index cards for recording and classifying requirements.

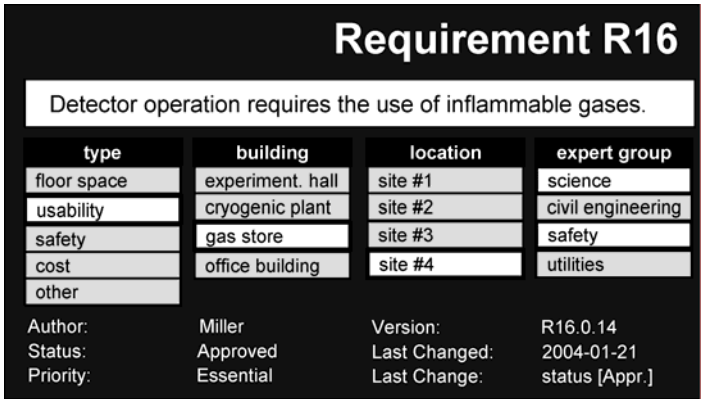

Figure 16: Example of a requirements index card.

Objective. Requirements contributions to a common product specification from numerous different stakeholders shall become compatible in format and level of detail as a basis for further analysis.

Context. Specifications have to be analyzed for completeness, contradiction and dependencies as a basis for negotiation.

Problem. For efficient comparison and correlation, project teams have to specify requirements in a unique, preferably text-based format, yet different stakeholders prefer their familiar individual formats.

Forces. This pattern addresses the diversity of specification formats:

- Communication across organizations and disciplines is facilitated by using a common format that is easily accessible to all stakeholders. For efficient comparison and correlation, requirements should be specified in a unique, preferably text-based format.
- Individual stakeholders have preferred formats for expressing requirements, which could include tables, flow charts, sketches, drawings or other graphics.

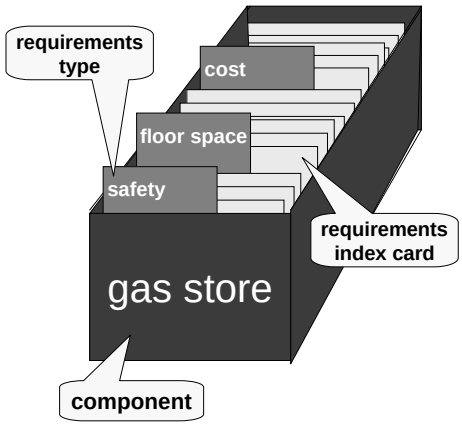

Figure 17: Card box principle to store requirements index cards.

Solution. Use the metaphor of index cards for requirements to introduce a common text-based format. Index cards are a simple and well-known tool and hence usually accepted by the stakeholder groups. Requirements on index cards are easily accessible. Sort the index cards to introduce subject based filter criteria for requirements access at the same time.

Structure. Each requirement index card contains a specification statement plus additional attributes for classification (see Figure 18). Include general attributes for requirements management and project progress tracking, and customized attributes like e.g. the affected system component and the relevant expert group(s) for filtering and correlating requirements.

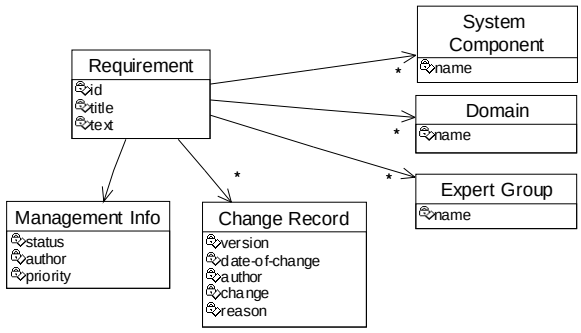

Figure 18: Requirement plus example attributes for project and requirements management.

Instructions. Select requirements attributes that suit the current project's needs from reference collections [MFM+96] [RR03] or other projects to design a requirements index card. Frequent attributes concern

- Requirements management (like id, history, author, source, ...)
- Project controlling (like priority, status, approval criterion, ...)
- Project specific information (like system components or disciplines for which the requirement is valid, requesting stakeholder, ...)
- Quality assurance (like stability, ambiguity, testability, ...)

Use the index card format to record all the existing requirements and introduce it as the standard format for writing down new requirements. Collect them centrally and file the index cards according to the project's needs, e.g. classify them by the affected component (see Figure 17).

Requirements index cards could be kept as "traditional" sheets of paper which are organized in card boxes at a central office, but a central electronic classification system like filtered spreadsheets or databases should be used to handle larger collections of requirements. Electronic filing enables requirements organization according to several criteria at a time.

Application Areas. Benefits have been observed in

- Interdisciplinary projects with different established "cultures" for specification (e.g. civil engineers are used to technical drawings, safety engineers to schematics, IT staff to models), which had to agree on a common format, and
- Projects with different interest groups, who want to filter and access the requirements according to different criteria which are of relevance to them.

Constraints. None observed so far.

Consequences.

↗ Requirements on index cards are easy to access. Classification enables filtering the specification and creating domain-specific views.

↗ Once they are made available in an accessible format communication about requirements among the stakeholders improves.

↘ Vague attributes or attributes with overlapping semantics might lead to useless classification.

Experiences.

- Index cards are easy to use and flexible in design.
- The required attributes are hard to guess in the beginning, but they are emerging as different users start accessing the specification. Index card design has to be foreseen as an incremental procedure.
- A set of index cards can be represented as a table and hence be implemented based on lightweight office products such as word processors and spreadsheets, as well as databases or sophisticated requirements management systems.
- Stakeholders might find it difficult to express their thoughts in an unused format and hence deliver specifications of lesser quality.
- If stakeholders have difficulties in using a fixed format, requirements expression can be carried out in two steps: first, stakeholders create requirements in their preferred format, then the requirements are adapted to index cards.

Known Uses.

- **Interdisciplinary Plant Construction**

  In a plant construction project, index cards have been used for capturing requirements along with several domain-specific attributes for classification. The attributes included for example the affected plant components, the authoring expert group(s), the disciplines and domains affected by the requirement, and the requirement type. The requirements were kept in a requirements management system, where they could be filtered by their attribute values. This way, different viewpoints could be created easily, e.g. by creating lists of requirements which were issued by a specific stakeholder group, or which are relevant to a certain discipline or plant component. The requirements lists were sent to the responsible experts for review and this way helped to maintain a good stakeholder involvement.

- **Introduction of COTS-based Information Systems**

  At a COTS-based system introduction, requirements were recorded with additional attributes for contracting purposes (e.g. vendor estimates of the maximum cost for realizing requirements), for relating functional requirements to work packages in which they should be realized, and for monitoring the development progress. The requirements were kept in a requirements management system, but with external companies requirements including their attributes were exchanged through spreadsheets. The tabular format enabled easy requirements reviews by the different involved stakeholder groups.

Related Patterns.

- “Employ a Requirements Engineer as a Care Taker” for handling the requirements index cards.
- Electronic index cards may help to identify candidate requirements if you want to “Write Reusable Common Requirements” or “Generate Approval Checklists”.

Further Reading.

[MFM+96] C. Mazza, J. Fairclough, B. Melton, D. de Pablo, A. Scheffer, R. Stevens, M. Jones, G. Alvisi, *Software Engineering Guides*, Prentice Hall Europe, 1996.

[RR03] S. Robertson, J. Robertson, Volere Requirements Specification Template, Edition 9, 2003, http://www.systemsguild.com/GuildSite/Robs/Template.html

3.6 Write Reusable Common Requirements

REQUIREMENTS ENGINEERING PATTERN

Write Reusable Common Requirements

This pattern recommends writing requirements in a general and reusable style to make an evolving specification robust against modifications of the project scope or structure.

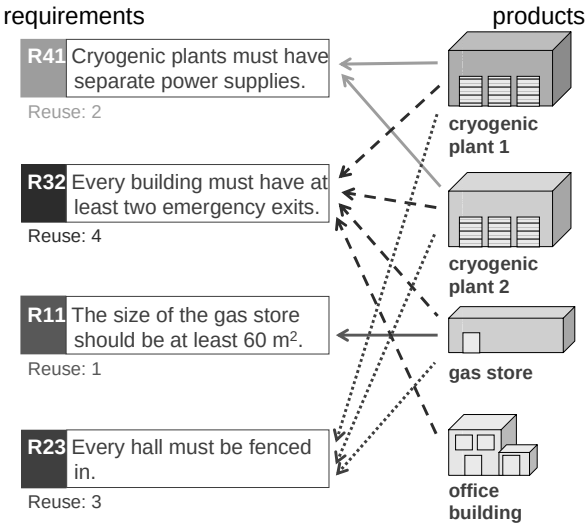

Figure 19: Example for reused common requirements in a plant construction pro-

Objective. The evolving specification shall be stable against changes when the planned product structure has to be extended or the configuration has to be changed. By reusing proven specifications double work shall be avoided and quality improved.

Context. The specification is based on the product structure or configuration. The same requirements apply to different products because the project forms part of a higher-level framework, e.g. a product line or program.

Problem. If requirements or the product structure change, updates have to be performed in many places. Inconsistencies arise from incomplete updates.

Forces. The forces in this pattern relate to the impact of requirements reuse in case of changes in the specification:

- Proven specifications describe the individual products and need only little adaptation in case of changes.
- Requirements are reused in several products so that changes in the requirements or in the product structure have to be updated at many places.

Solution. Write general requirements and relate them to the concrete product components which they affect (see Figure 19). Reuse the common requirements and manage the component relations instead of creating multiple requirements versions for the different products or components.

Structure. Figure 20 shows the schema for relating general requirements to product components or configurations in a database. Trace objects are used to implement the relation between requirements and specification and can carry additional information like e.g. explanation or fulfilment status.

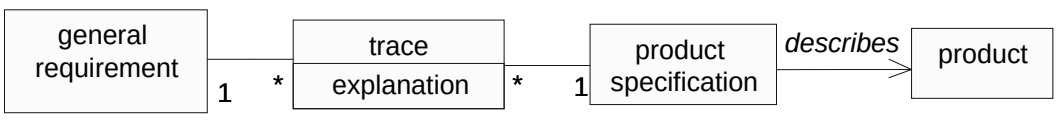

Figure 20: Relating common requirements to products using traces.

Instructions. There are several options for relating requirements with product components, including (static) traces in a requirements database, or requirements attributes containing specific values, which are used for (dynamically) querying the requirements database. The following example shows a combination of both:

- Provide attributes for relating requirements for example to product components, events or user groups.
- Check for each requirement if it explicitly contains a term which occurs as a possible value of one of the requirements attributes (for example a specific product component). Examine whether the requirement can be rephrased with general expressions and the reference be re-established by entering the specific value into the requirement's attribute.
- Obtain the requirements on a product component by filtering the database for example for those requirements which contain the component's id in their "component" attribute. Use the result set to establish static traces from each requirement in the set to the component.

Application Areas. Requirements reuse has been observed in projects which are part of a higher-level framework such as e.g. programs or product lines.

Constraints.

- Reusing common requirements for several products or components demands a requirements database that stores the relation between products and requirements.

Consequences.

↗ Requirements and specifications are adaptable for future extensions and follow-ups.

↗ Existing quality assured specifications can be extended to new products.

↗ The proposed implementation of requirements reuse enables product approvals by component based checklists and is also convenient for managing different product versions..

↘ The inner complexity of the specification is increased by implementing the relations.

↘ Change management has to cover requirements and their relations.

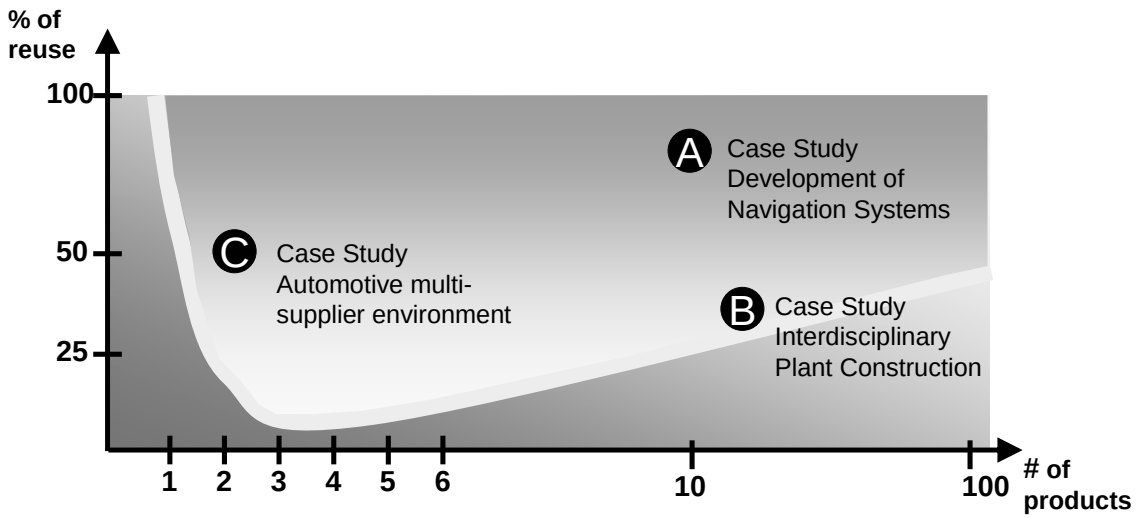

Figure 21: Experience from using the pattern in different settings. The area above the curve is supposed to result in a positive return on investment (ROI) for this pattern.

Experiences.

- Applications of this pattern have been observed in different settings. Benefits depend on the number of products and the degree of requirements reuse (cf. Figure 21; A and B are reported in the Known Uses section of this pattern).
- The growing complexity from tracking dependencies suggests the institution of a dedicated requirements engineer.

Known Uses.

- **Development of Navigation Systems (see A in Figure 21)**

  In the development of navigation systems for different car manufacturers, reuse was an issue both for single requirements and for complete component specifications:
  - Certain functionality turned up repeatedly, but was specified only once. A navigation system offers for example avoidance options for tunnels, freeways and other route characteristics.
  - New functionality was initially implemented only in a lead project. Only after the functionality has proven successful in the delivered product it was included to follow-up projects..

  Reuse was implemented by dynamic allocation of existing requirements or specifications to the individual product component.

- **Interdisciplinary Plant Construction (see B in Figure 21)**

  In a plant construction project, legal constraints and technological progress during the project were expected to change the number, composition, and configuration of the plant components compared to the initial plans. Requirements therefore were specified in a general style which made them independent of specific plant components. For example, security requirements read “Every hall must have two emergency exits“, instead of referring them to particular buildings, and similarly, “In every laboratory the temperature has to be kept constant within 0.5 degrees Celsius over the day”, rather than naming the labs explicitly. Reuse was implemented through requirements classification: attributes were used to assign the requirements to one or more plant components. If new components were added, the relevant requirements could be retrieved from the database through queries, and additional attribute settings could be added as necessary.

Related Patterns.

- “Use Requirements Index Cards” as a technical basis for implementing reuse by requirement classification.
- “Employ a Requirements Engineer as a Care Taker”, who should trace the relations between requirements and products.
- “Generate Approval Checklists” to conduct approvals with reused requirements.
- If you “Bundle Requirements to Features” this pattern serves as a basis for making feature lists reusable.

3.7 Build an Abstract Model to Integrate Fragmented Specifications

REQUIREMENTS ENGINEERING PATTERN *CANDIDATE*

Build an Abstract Model to Integrate Fragmented Specifications

Requirements from different sources have often been recorded in a variety of formats. This pattern recommends building an abstract model (for example in UML) to integrate their information content.

Objective. An integrated view on the information content of a set of specification documents shall be created.

Context. A set of separate requirements documents, which have been created independently from different sources, have to be integrated and consolidated into a central product specification.

Problem. Contributions to a specification are delivered in various formats; they contain for example requirements in different styles and terminologies, figures, sketches and flowcharts. For negotiating and defining the project scope, an integrated view on the specification needs to be created, but the different contributions are hard to relate due to their diversity.

Forces. This pattern addresses the difficulty of establishing a central specification in distributed projects:

- Stakeholders should be enabled to specify their requirements with minimum effort. This could include letting them perform their specification work locally, reusing existing material.
- A central specification is needed e.g. for requirements analysis and negotiation.

Solution. Build an abstract model – for example in UML – to capture and analyze the information content of the different specification fragments, and update or re-write the specification based on this model.

Experiences. Re-writing a specification based on a model is more efficient than trying to produce a text-based specification from the beginning. The model helps to establish consistent terminologies and effective project structures in a natural way prior to writing the specification.

Known Uses. The pattern has been observed in projects which further developed and replaced existing products. For the design of a new multi-purpose I/O device, user manuals and design descriptions of preceding products have been expressed and were related in terms of UML. In the introduction of a business system, process descriptions and system outputs of earlier systems were analyzed and integrated into a new process model.

Related Patterns.

- “Use Requirement Index Cards” as an alternative approach to unifying requirements style and format with a focus on building and querying a requirements database.
- “Evaluate Existing Documentation” as an additional contribution to the specification.
- “Employ a Requirements Engineer as a Care Taker” for performing the central modelling tasks.
- Use this pattern for creating a central specification if you “Organize Specification Along Project Structure”.

Credits. The pattern was discovered during discussion at the REP04 workshop [HHP04].

3.8 Provide Statements of Objective With Each Requirement

REQUIREMENTS ENGINEERING PATTERN *CANDIDATE*

Provide Statements of Objective With Each Requirement

Sometimes customers require technical features which seem convincing at first glance, but turn out to be expensive or even impossible to realize later on. This pattern provides the ground for finding reasonable alternatives.

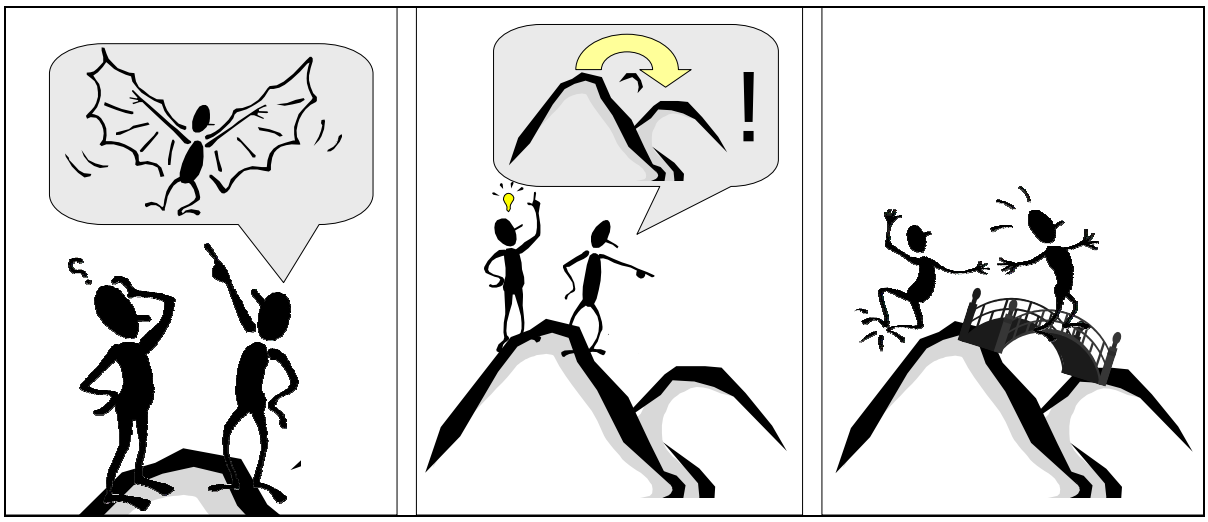

Figure 22: The statement of objective opens design and implementation alternatives: Wings are not required if bridges satisfy the same need.

Objective. Design alternatives for non-optimum technical approaches have to be developed.

Context. A customer orders the development of a new product and provides a vision which contains unacceptable technical details.

Problem. Design efforts are often biased by a technical solution which is proposed even before design alternatives have been developed and evaluated. Such proposals may be inspired for example by scaling available solutions, or by transferring technologies from other domains to the problem at hand. Although such proposals have their justification, the often describe unnecessary, expensive or even detrimental features.

Forces. The forces in this pattern relate to guidance in the design process:

- Novel product development should exploit existing successful technical solutions to benefit from their reliability and acceptance.
- Novel products should be inspired and invented independent of past or current technological solutions.

Solution. Treat proposed technological solutions as if they were system requirements. Relate them to user requirements and provide a statement of objective for each user requirement. If system requirements (i.e. the proposed solutions) refer to impossible solutions, use the statement of objective to initiate new design activities.

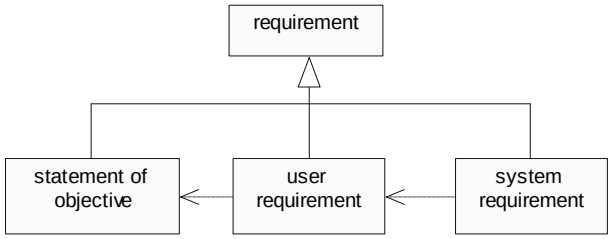

Figure 23: Statement of objective as the primordial requirement.

Known Uses.

- **Defence Project**

  In a defence project a long-range warship was envisioned to sail at 30 knots and more. As the necessary engines would have consumed most of the available payload, the reasons for the high speed requirement were investigated. The objective was to escape torpedoes – which used to run at 30 knots, but can nowadays reach more than three times the speed. The objective could be better reached using fake targets, and the high-speed requirement was dropped.

- **Spare Part Management**

  In a spare parts management project, mobile devices were to be introduced for maintenance work. The mobile devices should offer the same GUI as an available terminal client. Investigating the reasons for the technically difficult to realize GUI requirement revealed the intention behind: Maintenance workers should be enabled to register the spare parts they were using to enable immediate re-stocking of the parts – a functionality which was included in the terminal client. This essential requirement was satisfied by providing mobile scanners for part recognition with acoustic feedback for successful registration transactions. The GUI requirement on the mobile client was dropped.

3.9 Generate Approval Checklists

REQUIREMENTS ENGINEERING PATTERN
Generate Approval Checklists

If the project team expects checklists for tests and approvals, but requirements cannot be checked directly because they are reused for several project components, then checklists should be generated dynamically from the requirements database.

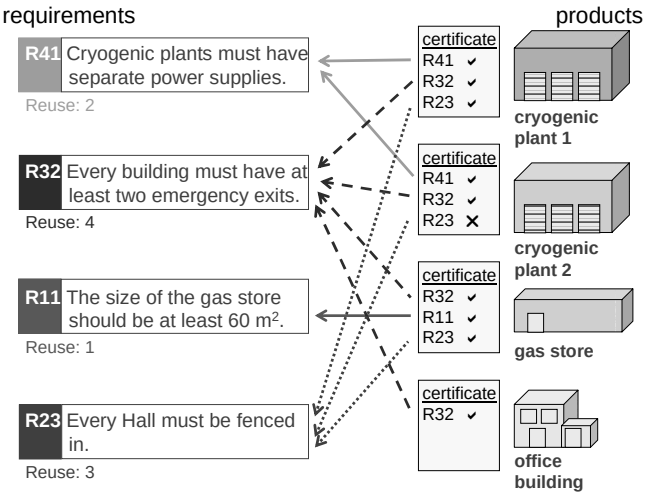

Figure 24: Example for generated approval checklists.

Objective. It shall be ensured that a given product has been created according to an agreed specification.

Context. A product version has been delivered and needs to be tested for compliance with the requirements specification. The test result has to be certified.

Problem. Requirements fulfilment cannot be implemented using check marks in the requirements document because the requirements are reused for several products (see Figure 24).

Forces. The pattern addresses the complexity of products and common (shared) requirements.

- An approval is an isolated activity which verifies step by step whether a specified requirement has been adequately implemented in a product. People therefore like to work with requirement checklists as a simple tool for conducting approval procedures.
- A requirement can never be marked “approved” because it might have been reused in different components or in approvals of other product versions.

Solution. For every product to be approved, use generated checklists that contain all the relevant requirements. Generate the checklists by querying the requirements database. Once completed, use the checklist as an approval certificate.

Structure. A checklist consists of a number of line items which relate a given requirement with a given product. Additional approval information like the fulfilment status is stored in the checklist line item (see Figure 25).

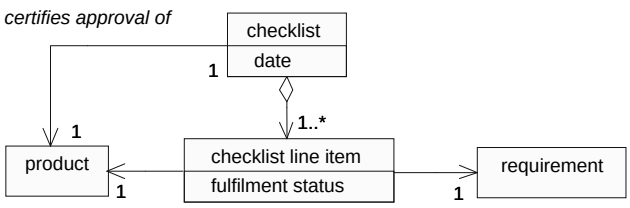

Figure 25: Storing approval information in checklists.

Instructions.

- Provide requirements attributes which store for each requirement to which products this requirement is related
- For every product approval to be carried out, filter the requirements database for the requirements which are related to the product. Export the result set to a spreadsheet or an office document for creating a checklist.
- During the approval procedure, go through the requirement list step by step and add fulfilment information to each requirement
- When completed, store the checklist as an approval certificate

Application Areas. The checklist approach is beneficial for any project that reuses the same requirements on several occasions, e.g.

- Projects that follow an incremental process model, where products are delivered in growing versions, and approvals are re-iterated;
- Projects which deliver product families;
- Projects which are still vague in their scope.

Constraints.

- Generating approval checklists demands a requirements database that stores the relation between products and requirements.
- In some cases, requirements fulfilment cannot be recognized directly from the requirement, but demands further instructions, e.g. test cases. In such cases, the pattern has to be extended to anticipate the need for test cases.

Consequences.

↗ Approval checklists enable product and requirement versioning and allow to track the history of approvals.

↗ Checklists can be formally treated as independent documents, i.e. reviewed, released, published, etc. They decouple approvals from the specification.

↘ Tracking checklists involves configuration management efforts.

Experiences.

- Generated approval checklists are a powerful tool for involving stakeholders into the RE activities. The checklists can be created e.g. as easy-to-use spread sheets, distributed through e-mail, and filtered according to the individual responsibilities of each stakeholder.

Known Uses.

- **Interdisciplinary Plant Construction**

  In a plant construction project, the requirements were kept in a central requirements database. Attributes were used to assign the requirements to plant components, engineering disciplines and responsible stakeholder groups. The stakeholders included mechanical engineers, civil engineers and future users of the planned facilities. For each plant component, approval checklists were generated by querying the requirements database for all requirements which were assigned to that component. The checklists were used for reviewing the CAD design models of the plant components to ensure they would include all the

intended installations without collisions. The checklists proved to be an essential vehicle for communication and documentation and for improving stakeholder involvement.

- **Introduction of COTS-based Information Systems**

  A project for introducing a COTS-based information system was structured in a way that project deliverables corresponded to work packages (WPs) in the project plan. Each requirements was linked with all the WPs which needed to observe the requirement. When approving a work package, checklists were generated by querying for requirements which were linked with that WP. This way, the implementation team could check the relevant functional requirements for their actual WP, and the project leader could follow implementation progress by observing requirements fulfilment.

Related Patterns.

- "Use Requirements Index Cards" as a means to record the requirements.
- "Write Reusable Common Requirements" to build your requirements database in an adequate information structure.
- Use approval checklists to record the test results if you "Detail the Specification by Writing Test Cases".

3.10 Detail the Specification by Writing Test Cases

REQUIREMENTS ENGINEERING PATTERN

Detail the Specification by Writing Test Cases

If a specification turns out to be ambiguous or incomplete while a project is already well on its way, this pattern describes a way of clarifying the specification without halting the implementation work.

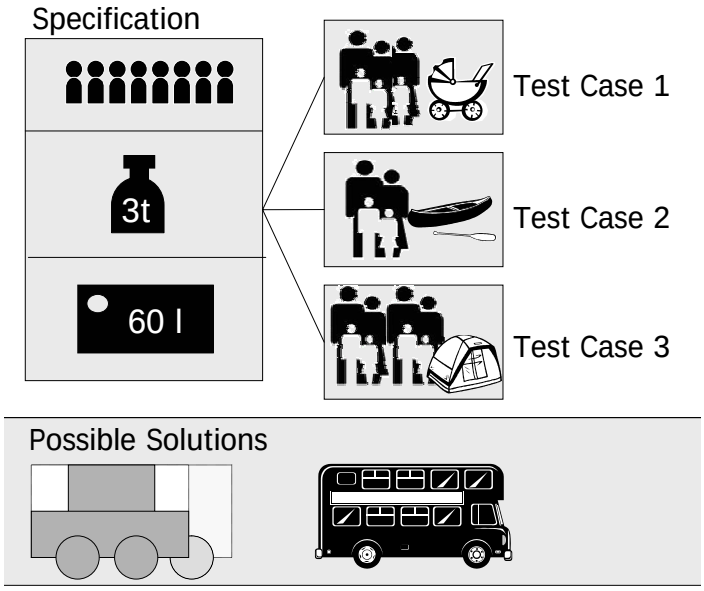

Figure 26: Test cases provide more concrete ideas of the desired solution.

Objective. Usability (e.g. clarity, testability, coverage) of a specification shall be improved after it has already been frozen.

Context. Client and supplier have agreed on a specification which has been frozen because it became for example part of a contract, or the project progressed to the next phase. When using the specification for implementation and test it turns out that it is incomplete and ambiguous.

Problem. The specification is in a shape which endangers the project success, but no more time and resources are available for improving the specification.

Forces. The pattern is related to pushing the progress in the project:

- The project has to step back to improve the specification.
- The project has to move ahead to meet the milestones.

Solution. Leave the specification as it is and create test cases instead. The test cases should describe the usage and the expected output from an end user perspective. Provide the test cases to the implementation team, and agree that the test cases will become the criteria for approval and in this sense an appendix to the specification.

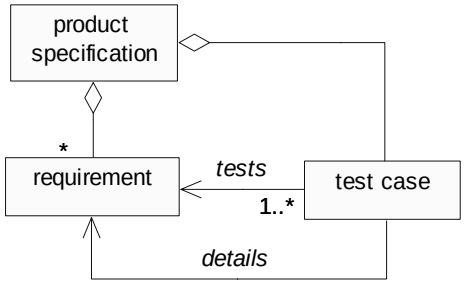

Figure 27: Test cases contribute to the specification by detailing requirements.

Structure. Test cases explain the context of requirements and provide examples for scenarios which rely on a requirement. They should contain an objective, preconditions, a course of events including exceptions and alternatives, and expected results for specific in-

puts. This way, they can improve e.g. the level of detail, the clarity, the coverage and the testability of requirements.

Instructions. The key to good test cases is the end user perspective. Test cases can be collected in different manners:

- Let key users describe use cases and concrete usage scenarios of the system and employ them as test cases.
- Let key users conduct tests and record their activities and their expectations.
- Let key users explain for each requirement in which situations the requirement is relevant to them, then create scenarios for each such situation.

Application Areas. This pattern has so far been observed in small and medium projects which are conducted by teams working to a large degree on a basis of understanding.

Constraints.

- Usually contracting is based on the initial specification, and the introduction of test cases later on can be seen as an attempt to extend or modify the project scope. It will therefore only work if both client and supplier follow the same intentions of improving the project quality.
- Ideally, the approach has to be realized without affecting the project resources, implying that it should be used only on small amounts of requirements at a time.

Consequences.

↗ The test cases are used as a new basis for implementation.

↗ Hidden assumptions and wishes are revealed and made explicit before tests and approvals.

↘ Test cases can lead to unsolicited updates of the requirements specification from the client side.

Experiences.

- Writing test cases cannot replace the requirements specification. Generally both are needed to completely capture the different views on a project.
- It is easier to acquire resources for tests and test case specification than for requirements analysis as the necessity of test is generally acknowledged.
- Writing test cases helps discovering weak points in the specification.
- Test cases are best written by domain experts or end users who ideally act as multipliers in the project team.
- Test cases can be written in parallel by several independent persons.
- “Better late than never” – the availability of test cases for requirements always pays off in reduced development cycles.

Known Uses. The pattern has been observed in the introduction of information systems.

- **Migration of an Information System (Logistics)**

  A logistics company’s liability management application should be replaced by a newly developed application. Management replaced the project leader after the specification was written. The new project leader found the specification to describe the new system’s functionality in an understandable and complete way. He found a “joint understanding” of what the new system should do having been established among all stakeholders. But he

also found the specification not to be suitable as contractual basis as the formulations were interpretable in too many places. The specification's authors resisted to doing substantial rework because if they did rework, they had to admit failures in their former work. Thus the new project leader convinced end users to specify and conduct tests. Clarifications took place when end users and developers discussed the test cases. As a consequence, the system was introduced with the correct functionality.

- **Introduction of a Facility Management System**

  To suit the purchasing of COTS components for a Facility Management System (FMS), user requirements were specified in an abstract, product independent style which was suitable for software selection and contracting. When customization started, more detailed questions arose and holes in the specification became apparent. As a consequence, the external developers provided features as specified, but were unable to envision the way the users intended to work with the FMS. The resulting system was functionally acceptable, but usability and user acceptance were low. As the specification could not be modified after contracting, both parties agreed to add test cases for better explaining how the clients expected to work with the system. The developers used the test cases already for early module tests and thus were implicitly guided by the test cases, obtaining an overall improvement of the system's ergonomics and acceptance.

Related Patterns.

- "Bundle Requirements to Features" for recording the relation of requirements and test cases.
- "Generate Approval Checklists" to record the test results.

3.11 Organize Specification along Project Structure

REQUIREMENTS ENGINEERING PATTERN

Organize Specification along Project Structure

This pattern recommends using the same structures for project management and requirements elicitation to minimize coordination efforts.

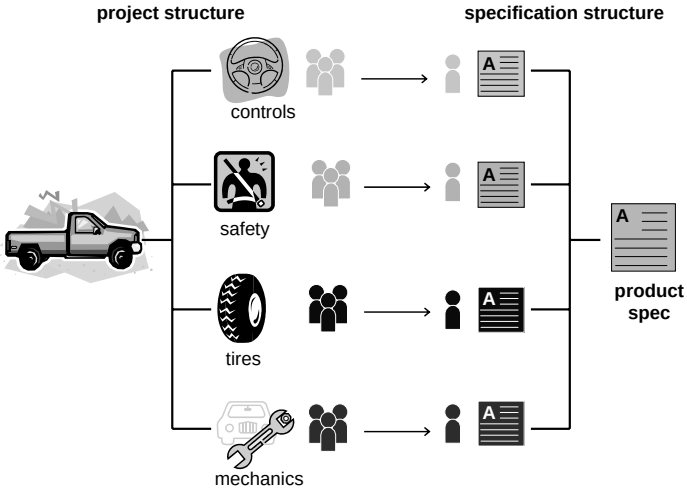

Figure 28: Specification mirrors project structure.

Objective. The different views on the project have to be coordinated and shall be represented in an agreed specification.

Context. A project is organized in a work breakdown structure (WBS). Distributed project teams work on the individual WBS elements. A specification that covers the whole product has to be worked out and agreed upon by the teams. Responsibilities for creating the specification have to be assigned in the beginning of the project.

Problem. The project teams are expected to spend their time primarily on working on the solution, but they also need to be involved in the specification process to record and negotiate their needs and constraints.

Forces. The forces in this pattern address the intensity of communication and information exchange in collaborative teams:

- On behalf of the mission, requirements have to be negotiated until mutual agreement has been reached. This requires a collaborative working environment.
- Groups of specialists tend to concentrate on the topics of their own immediate concern and may thus become too self-sufficient and difficult to access for negotiation.

Solution. Organize the specification procedure according to the project organization. Let each WBS team appoint an author who writes a specification from this team's view point on the product. Figure 28 shows a structure of specification documents and their authors that reflects the WBS elements and project teams. Introduce a requirements engineer who creates an overall product specification from the partial ones. Thus the communication overhead to write specifications is reduced to a minimum for the busy project teams. The requirements engineer takes responsibility for the specification progress and for ensuring requirements conflict resolution.

Structure. Figure 29 shows a class diagram for organizing the specification procedure. Every project team or stakeholder group who is responsible for a WBS element has to appoint one person as an author for specification purposes. Authors contribute to the project specification according to their team's viewpoint. A central requirements engineer acts as an architect who creates the overall project specification by coordinating the authors in a similar way as the project manager coordinates the project teams. The requirements engineer takes responsibility for the specification progress and for ensuring requirements conflict resolution.

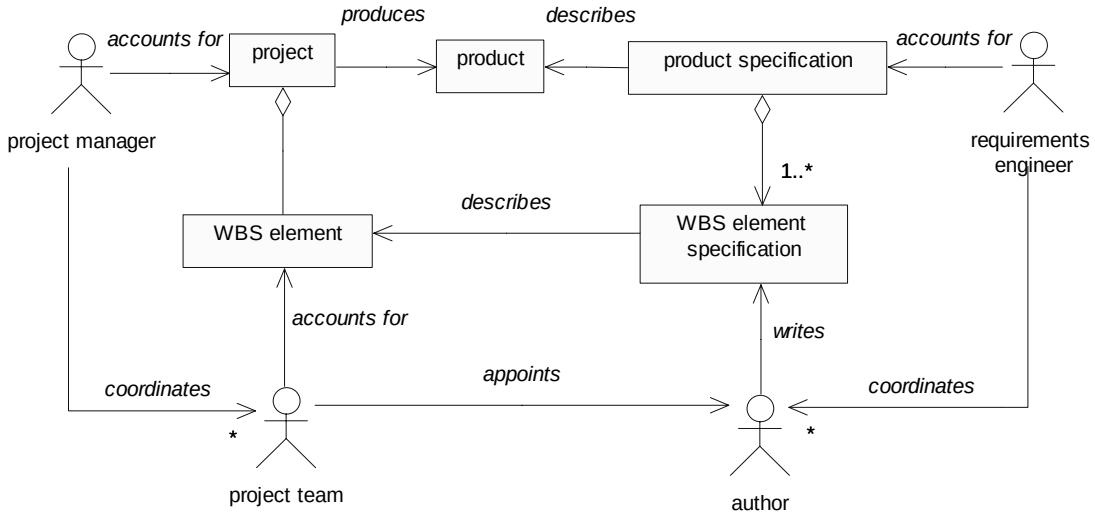

Figure 29: Responsibilities for WBS elements and specifications.

Instructions.

- Convince the client to support the measure by explaining the forces from this pattern and the perspective to compensate the cost for the requirements engineer by reduced development and test efforts.
- Let every project team appoint an author for the WBS element specifications. Try to make sure a suitable person with sufficient standing in the team is selected. Stakeholders which are not accessible to the project should be represented by a team member who is playing their role.
- Appoint a requirements engineer who instructs and supports the authors on how to write specifications.
- Let the requirements engineer consolidate an overall specification from the teams' contributions. He should analyze the requirements to detect omissions and conflicts that have to be solved, and facilitate negotiation and conflict resolution.

Application Areas. The proposed organization has proven to be useful in distributed projects that require formalized communication channels, e.g.

- Locally distributed projects, in which the project teams are spread across different locations;
- Projects with diverse stakeholders or expert groups, e.g. originating from different disciplines and having different "cultural" backgrounds;
- Projects with large fluctuations in the team over time.

Constraints.

- This pattern requires the project to be structured into teams according to WBS elements, i.e. each WBS element is assigned to one project team. It further assumes that the main stakeholders and their viewpoints are represented in the project teams, as otherwise the specification will not be complete.
- Influence on the assignment of responsibilities in the project is another prerequisite for using this pattern.
- Organizing the specification along the existing project structure requires a central requirements engineer as a caretaker for the overall specification.

Consequences.

↗ Authors are deeply involved in the negotiation procedure and are thus able to maintain the essential interests of their teams and at the same time support decisions and necessary compromises in their group.

↗ The role of the requirements engineer ensures a moderated and formally documented negotiation procedure.

↘ Requirements negotiation requires specification contributions with compatible formats and styles. Stakeholders might be forced to use other than their preferred formats.

↘ The communication overhead that is taken from the productively working teams has to be partially taken over by the authors and the requirements engineer.

Experiences.

- Structuring RE along the project organization can draw benefits from established teams and communication channels, but success depends on the experience and social skills of the requirements engineer and the project manager.
- Requirements negotiation should be instantiated as a regular meeting. If groups do not contribute according to their responsibilities, the specification remains incomplete. Thus, unwilling groups can endanger the whole project.
- The resulting specification describes a joint vision and is balanced in granularity and terminology.
- To ensure compatibility of the specifications a definition of terms and a set of templates for specification are advisable.
- Requirements should be management using an RMS, which helps generate acceptance if it makes specifications available and researchable for the whole project team.

Known Uses.

- **Interdisciplinary Plant Construction**

  In a plant construction project, the teams were organized according to plant components and different engineering disciplines, e.g. there were groups for the technical equipment, cryogenic components or halls and buildings, as well as for survey or electric supplies. Each team nominated one requirements author who contributed to a central specification. The authors were trained by a central requirements engineer to use a requirements management system (RMS), which facilitated access to contributions from other groups. The requirements engineer used the RMS to monitor and support the contributions to the specification.

- **Introduction of COTS-based Information Systems**

  For a couple of COTS-based system introduction projects, the project teams were structured according to business processes. One additional team was responsible for the overall system architecture and integration. In each business process team, one person took responsibility for the process and provided a specification for this process as a part of the overall specification. The contributions were prepared and updated by the teams using office tools. They were maintained by a central requirements engineer using an RMS with export/import capabilities for office tools.

Related Patterns.

- „Employ a Requirements Engineer as Care Taker" for creating and coordinating the overall specification.
- "Create a Specification Guideline by Tracking How an Analyst Works", and distribute the specification guideline to the project teams.
- "Use Requirements Index Cards" as a unique specification format in all the stakeholder groups as a basis for later integration of the specifications.
- "Build an Abstract Model to Integrate Fragmented Specification" provides a guideline for the requirements engineer for creating a central specification.
- "Evaluate Existing Documentation" to represent stakeholder groups which are not accessible for specification tasks.

3.12 Employ a Requirements Engineer as a Care Taker

REQUIREMENTS ENGINEERING PATTERN *CANDIDATE*

Employ a Requirements Engineer as a Care Taker

If requirements engineering efforts are growing so large that project teams are no longer able to accommodate them in their daily business with satisfactory quality, a dedicated requirements engineer should be established as a central person for performing RE activities.

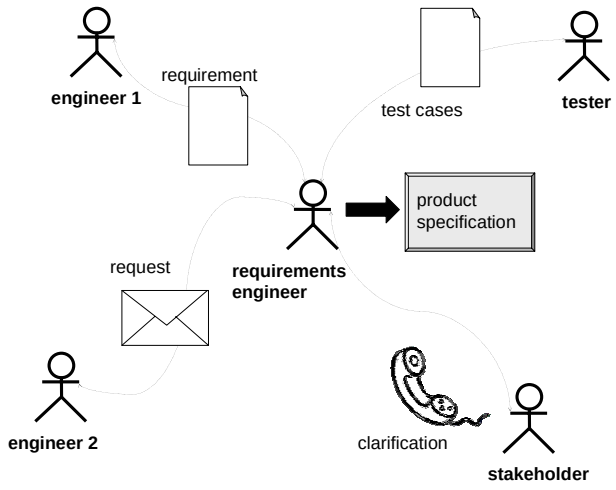

Figure 30: A central requirements engineer supports RE activities in a project.

Objective. A specification and a joint understanding of its meaning shall be created without taking the domain experts off their daily business.

Context. A project team of experts from several domains has to create a mutually agreed specification.

Problem. The domain experts need to be involved in the requirements tasks, as they have the technical domain knowledge and they are the persons who need to establish agreement.

On the other hand, their involvement is problematic as they tend to

- Have little knowledge of requirements engineering methods
- Be rarely available due to other obligations in the project
- Have little motivation or enthusiasm for requirements engineering
- Have no language talent.

Forces. For creating a quality specification,

- Authors should have profound know-how of the domain.
- Authors should have profound know-how of requirements engineering methods.

Solution. An author with the domain knowledge is supported by a person with know-how of the requirements engineering methods. This person is called a requirements engineer.

Structure. The requirements engineer ensures that the primary tasks of the domain expert do not suffer from workload due to RE activities by performing most of the RE tasks. These tasks include elicitation and negotiation (acquiring domain knowledge, specifying require-

ments and taking care that all the stakeholders develop the same understanding of the specification), coordination and requirements management.

Instructions.

- Select a person who is competent in requirements engineering methods and in communication, and which can gain acceptance in the project team. Acceptance can for example be gained by persons who are well-known from former activities, by persons which perform similar tasks in comparable settings, or by authority. Having domain knowledge can help, but is not obligatory and may in some cases even be counterproductive.
- If coordination and moderation are important in the project, consider selecting an external person.

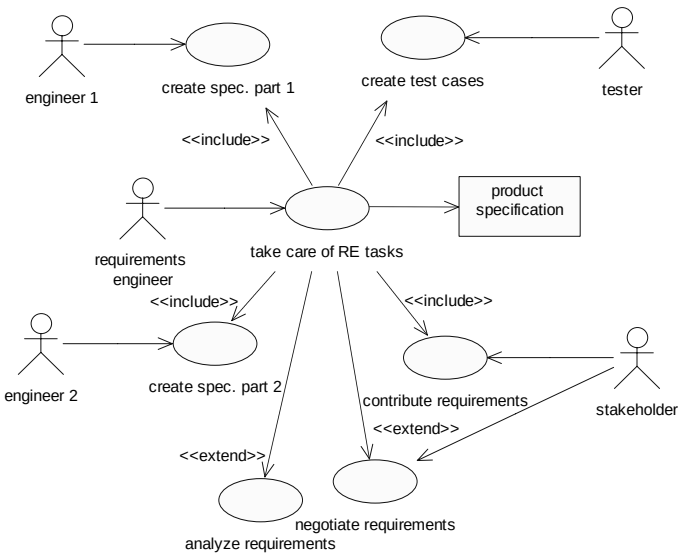

Figure 31: Product specification with a central requirements engineer as a care taker.

- Let the project manager introduce the person as “the requirements engineer” by pointing out the benefits. The requirements engineer will relieve all the stakeholders of most of their requirements engineering tasks (except providing their knowledge).
- Support the requirements engineer by letting the project manager affirm the role as often as possible towards all other stakeholders.

Application Areas. The pattern has been observed in various project types and sizes.

Constraints.

- Domain experts need to be at the requirements engineer’s disposal.

Consequences.

↗ The requirements engineer provides improved input for the project management (e.g. planning of releases).

↗ The requirements engineer replaces the domain expert as communication partner to the developer and vice-versa.

↗ The requirements engineer coordinates communication and improves its efficiency.

↗ The know-how of other stakeholders about requirements engineering methods increases over time.

↗ The domain know-how of the requirements engineer increases over time, enabling him to answer questions or make suggestions when working with domain experts.

↘ The requirements engineer is a dedicated position which has to be staffed accordingly.

↘ The requirements engineering resources have to be introduced timely; they cannot be increased in later project stages by simply adding more requirements engineers as the peo-

ple would need to catch up with the project teams learning curve before they become productive.

Experiences.

- In larger projects which make use of more elaborate requirements management tools the role of the requirements engineer as a care taker includes also technical support of the domain experts.

Known Uses.

- **Interdisciplinary Plant Construction**

  In a plant construction project, expert teams from different engineering disciplines had to develop and negotiate a common vision of the intended facilities. The experts were responsible for project coordination and for specification, design and installation work, and at the same time had to operate and maintain other installations. A central requirements engineer has been appointed for supporting the expert teams by organizing, performing and promoting the RE activities. Furthermore, the requirements engineer was responsible for introducing and operating a requirements management system (RMS). By coaching, distributing material and publishing specifications in the RMS, the requirements engineer successfully involved the different stakeholder groups into the specification procedure.

- **Migration of an Information System (Logistics)**

  A logistics company's liability management application had to be replaced by a newly developed application. Management changed the project manager after the specification was written. The new project manager found the specification to describe the new system's functionality in an understandable and complete way. He found a "joint understanding" of what the new system should do having been established among all stakeholders. But he also found the specification to not be suitable for contracts as the formulations were ambiguous in too many places. The specification's authors resisted to doing substantial rework because if they did rework, they had to admit failures in their former work. Thus the new project manager took over the role of a requirements engineer who cared for maintaining the "joint understanding" throughout the rest of the project.

Related Patterns.

- Define the requirements engineer's responsibilities through other RE patterns, for example let the requirements engineer „Create a Specification Guideline by Tracking How an Analyst Works" to help domain experts at authoring specification documents.
- "Build an Abstract Model" from "Analyzing Existing Documentation" to get the requirements engineer involved into the project.
- Employ a requirements engineer as a care taker if you want to "Organize the Specification Along Project Structure", "Write Reusable Common Requirements", "Use Requirements Index Cards" or "Synchronize Change Requests".
- "Use a Central Issue List" to coordinate the change procedures.

3.13 Use a Central Issue List

REQUIREMENTS ENGINEERING PATTERN *CANDIDATE*

Use a Central Issue List

This pattern recommends to immediately track any open issues in a central position of the project because they might otherwise be treated differently by different project team members - or even get completely lost.

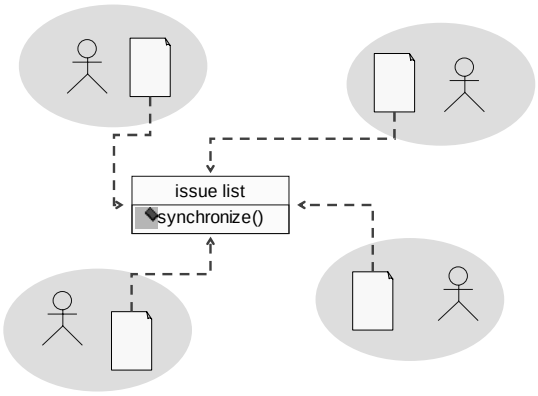

Figure 32: Distributed teams working on different tasks that are synchronized through a central issue

Objective. The goal of this pattern is to keep track of open issues in a client-supplier setting, i.e. in a setting with two or more different environments.

Context. Open issues occur due to changed requirements, identified (software) defects, unclear requirements that cause need for clarification, and many other reasons. Development activities have to be shared amongst at least two parties (e.g. client and supplier) that are not co-located.

Problem. Identified and accepted issues have to be communicated throughout the project team in a way that no individual issue can be "forgotten". The problem becomes the more severe the more issues have to be maintained simultaneously.

Forces. This pattern addresses the transparency of the change process:

- The involved teams intend to build the right system and thus are open for change requests. They are interested in getting their issues accepted and fixed by other parties, and in return they are also open to fix issues dedicated to them.
- The involved teams have only limited resources, so they need to be restrictive regarding issues they accept to fix.

Solution. Define, establish and maintain a central issue list. All follow-up activities of an issue are triggered and monitored by means of this list. New issues might only show up in the list if all the involved parties have agreed on that issue.

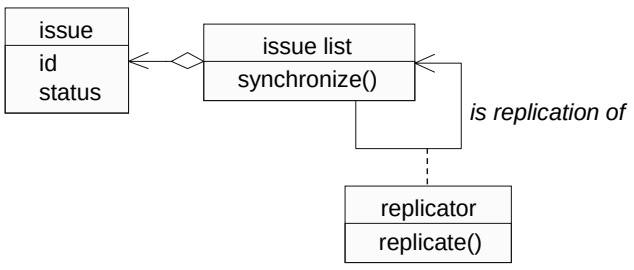

Figure 33: Structure of a central issue list that synchronizes list copies through issues IDs.

Structure.

Variant (a): Maintain one central issues list that is read-only accessible to all project members. Writing is limited to a single person during project meetings.

Variant (b): Maintain several lists that are kept in the different teams and synchronized on a regular basis.

Instructions. Introducing a central issue list involves two major activities:

- Establish an issue list with a commonly used state model for issues (e.g. new – under consideration – in work – closed).
- Establish a defined and agreed process how to (i) enter new issues and (ii) modify issue states.

In small projects, the issue list could be implemented using a spreadsheet on a network drive. A process could be emulated by appointing a single person only to maintaining the list, and granting read-only access to the rest of the team. Every change to the list would have to be communicated through the maintainer, who would automatically act as a filter for accepting and following issues. A simple database and / or generated Web front ends may be used as an alternative to the spreadsheet.

In larger projects, issues should be maintained and tracked in a dedicated database tool which records the history of issues and provides user access control. A variety of public domain tools are available for such purposes, among them for example the Request Tracker [BPS02] and bugzilla [Moz98]. Many of these tools also propose simple and easy-to-implement processes for accepting, coordinating and solving issues.

Application Areas. All kinds of development projects. No restrictions have been observed so far.

Constraints.

- Both the issue list structure and the process to handle the issues have to be commonly agreed and pursued.
- If two lists are used, tool support is highly recommended for synchronisation.

Consequences.

↗ Project status transparency increases as the issue list can be used for controlling and progress tracking purposes.

↗ Only agreed issues that have made it into the list should cause follow-up activities.

↘ Incorporation of new issues is bound to “synchronisation events” (e.g. regular project meetings).

Experiences.

- The earlier the issue list is introduced the easier it gets adopted. It is very hard to change established communication channels in later project stages.
- Stakeholders who have experienced the problem in former projects are more likely to understand the necessity of formal documentation.

Known Uses.

- **Introduction of COTS-based Information System**

  For a couple of COTS-based system introduction projects, the project consisted of both part-time internal participants and external developers. Direct communication was limited to dedicated meetings. During discussions at different places issues arose that escaped part of the project team and could not be conducted to a decision because they got lost. Introducing a central issue list helped to collect open points and make them available. Attributes were used to classify and filter items e.g. regarding priority or affected components. Thus issues could be followed over time and decided upon, e.g. to become a change request or a warranty issue. During the project, the issue lists were kept in spreadsheets which were maintained by the project managers. When the systems were released into production, the Request Tracker [BPS02] has been introduced for accepting and tracking issues: a hotline has been established for user consulting and accepting and dispatching issues, and the support staff was working on issues according to their assignments by the hotline.

- **Automotive multi-supplier environment**

  In an automotive development project a supplier was asked to develop an electronic control unit (ECU). This means, that the client had the duty to specify the system. The specification was than handed over to the supplier. At the beginning of a development, it is quite unrealistic to have a complete specification on the table. New functions and features or changes emerged over time. This means, that from time to time, the supplier got a new and more detailed specification. Every time he got an updated product specification, he evaluated the new or modified parts. Usually, this raised a number of questions and issues to be clarified. As a solution, a list that collects all these points both on the supplier's and the client's side had been established. After the evaluation of an SRS, the supplier handed over the list (filtered on those points that had to be solved by the client) to the client. The client included these points into his list, prepared answers to them and gave the answers back to the supplier.

Related Patterns.

- "Synchronize Change Requests" uses a central issue list to document all changes to the specification.
- "Employ a Requirements Engineer as a Care Taker" who is among many tasks also responsible for maintaining the central issue list.

Further Reading.

[BPS02] Best Practical Solutions LLC, RT Request Tracker, http://www.bestpractical.com/rt/

[Moz98] The Mozilla Organization, Bugzilla, http://www.bugzilla.org

3.14 Synchronize Change Requests

REQUIREMENTS ENGINEERING PATTERN

Synchronize Change Requests

If a project needs to be open for changes to ensure optimum customer acceptance, but at the same time has to meet its milestones, this pattern recommends establishing a change process for coordinating the interplay of project work and specification updates.

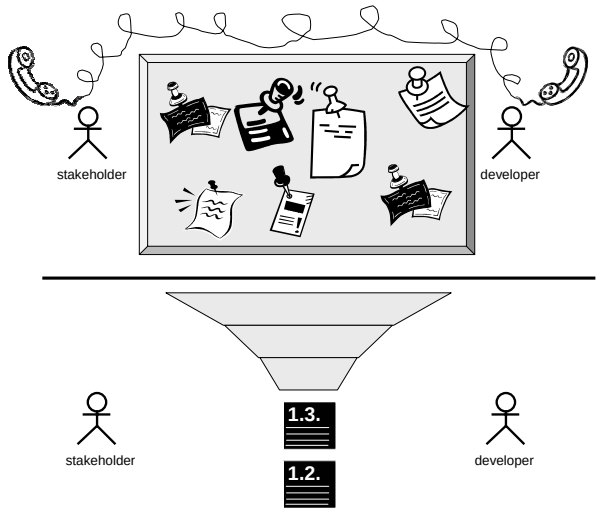

Figure 34: Ad-hoc communication on change requests (top) and exchange of synchronized change requests (bottom).

Objective. The goal of this pattern is to create and maintain a clear and consistent foundation of requirements for product planning and implementation. It shall be made clear to all stakeholders what the changed requirements are and what the impact on the development process will be.

Context. Requirements on products change during the development process for example because of new or changed stakeholder needs or results of prototype evaluations and tests.

Depending on the project phase, these change requests have different impact on the product development. In any case initial plans on time and budget have to be repeatedly updated.

Problem. Change requests emerge from diverse sources, but the priorities and the status of the change requests are not clear. As a result, the current requirements are scattered across requirements documents and change requests.

Forces. This pattern addresses the transparency and the intensity of change requests:

- The project team intends to build the right system and is thus open for necessary changes in the system specification.
- The project team fears the risk of losing the overview over impacts of changes and is thus restrictive regarding change requests.

Solution. Define, agree, establish and maintain a process for collecting, recording and tracking change requests. Link change requests to the existing requirements documentation in such a way that the latest agreed version of each requirement is immediately obvious to the reader.

Structure. Each change request follows the same defined life-cycle. The life-cycle activities include recording, deciding upon and implementing change requests. Responsibilities are assigned to each life-cycle activity (cf. Figure 35).

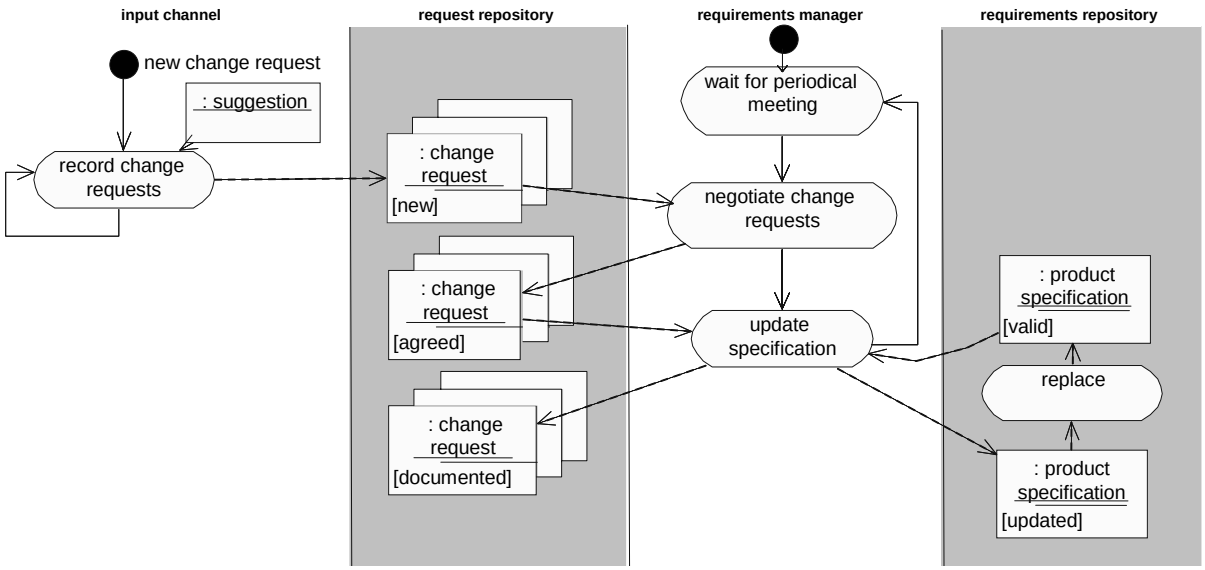

Figure 35: Example workflow for synchronizing change requests. The updated specification replaces the previously valid version.

Instructions.

- "Use a Central Issue List" to establish a single input channel (person and/or tool) for change requests. Refer to the pattern for more implementation details, especially on how to store and track the issues.
- Establish a defined and agreed process how to include changes into the existing specification. Again, refer to the "Use a Central Issue List" pattern for more details on how to implement the process.

Application Areas. The pattern has been observed in several kinds of development projects. So far, no restrictions are known. It is highly recommended for projects of ten or more persons.

Constraints.

- A dedicated person is necessary for managing the change requests.

Consequences.

↗ Specification documents are up to date, developers know what to develop, and testers know what to test.

↗ Risks are identified and can be addressed.

↗ It is possible to re-plan with reasonable data.

↘ Some stakeholders might have the impression that "flexibility" decreases, others that all this overhead it not necessary.

↘ There might be resistance against the "change in handling changes", which then would have to be treated like cultural changes.

Experiences.

- The earlier the change request workflow is introduced, the more natural it will become to the project team. It is very hard to change habits and culture in later project stages.
- If your requirements documentation is in a bad condition, try to fix this problem first..
- Project management support is needed as workflows induce cultural change.

- In project teams of fifteen or more persons, usually some tool support is needed. In smaller project teams, this problem may not arise due to an informal “common understanding”.

Known Uses.

- **Migration of an Information System (Logistics)**

  A logistics company’s liability management application should be replaced by a newly developed application. The development of the new application was started with implementing partial functionality which was specified in great detail. Some other functionality was not yet specified accurately at this stage of the project. Change requests emerged for example while gaining insights through tests of prototypes and sub-functionalities, or through refining requirements in the course of the project. The change requests emerged because of dependencies to requirements being already implemented. A change request management process was introduced to work out clearly the requested changes and the impacts on time and budget as the requested changes had to be commissioned and to be paid for to the software house explicitly.

- **Development of Infotainment system**

  An Infotainment-System for passenger cars should be developed. Such a system typically contains radio, compact disk, telephone, navigation, SMS and partially WAP and e-mail. The development concerns hardware, software and mechanics. The client initiated change requests through various channels (telephone, personally, e-mail, changed requirements specification documents) and sent them to various persons (developer, subproject manager, software project manager, project manager). It was not recognizable whether requirements were committed or not. Different client’s business departments requested contradictory requirements. The various documents were handled in a chaotic way within the project. An overview was missing. Documents were left in the project manager’s inbox. The situation improved when a requirements manager was introduced who managed all the existing documents and who became the channel through which changes to requirements entered the project in a defined way.

Related Patterns.

- “Employ a Requirements Engineer as a Care Taker” to ensure that responsibility for handling change requests is clearly defined.
- “Use a Central Issue List” to document all changes to the specification.

3.15 Create a Specification Guideline by Tracking How an Analyst Works

REQUIREMENTS ENGINEERING PATTERN

Create a Specification Guideline by Tracking How an Analyst Works

If requirements need to be written by domain experts personally, but the experts are lacking the methodological knowledge, this pattern recommends providing a specification guideline to the experts. The guideline could be obtained from observing and recording the way how an analyst creates specifications.

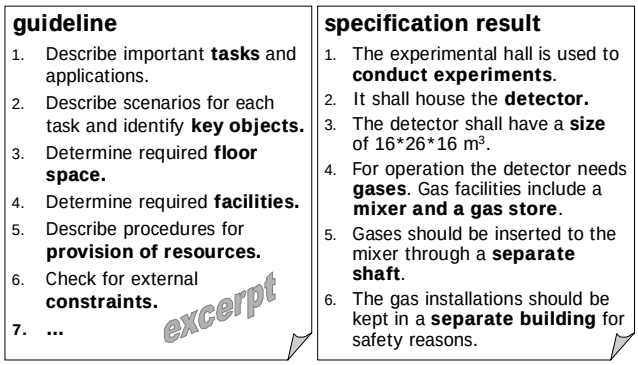

Figure 36: Example for a specification guideline and obtained results.

Objective. Distributed stakeholders shall be enabled to create complete, balanced and consistent contributions to the project specification.

Context. Envisioning and requirements elicitation have to be performed in a distributed project with independent teams. The stakeholders cannot delegate requirements specification, but have to write specifications personally, e.g. because of their specialist knowledge. They are not available for thorough method training.

Problem. The stakeholders create specifications according to their individual habits, background and experience and provide ad-hoc specifications. The resulting documents are of varying quality regarding for example completeness and precision.

Forces. This pattern addresses the different skills needed for specification.

- A specification needs to be complete, balanced in its level of detail, and consistent. Creating such specifications requires training in methods and notations.
- Writing and maintaining specifications requires expert knowledge in the stakeholder domains.

Solution. Create a specification guideline which tracks an analyst's modelling steps. Use the guideline to transfer the analyst's modelling skills to the stakeholders. It enables them to build a verbal specification similar to the quality of the visual model an analyst would normally build.

Structure. Analysts usually create business models using a modelling method. Such methods can be expressed in terms of modelling guidelines. Inversely, analysts could develop guidelines by keeping track of their steps when creating the model. The guidelines can be used by the stakeholders to create a verbal model (i.e. a specification document) that is comparable to a business model regarding completeness and balance.

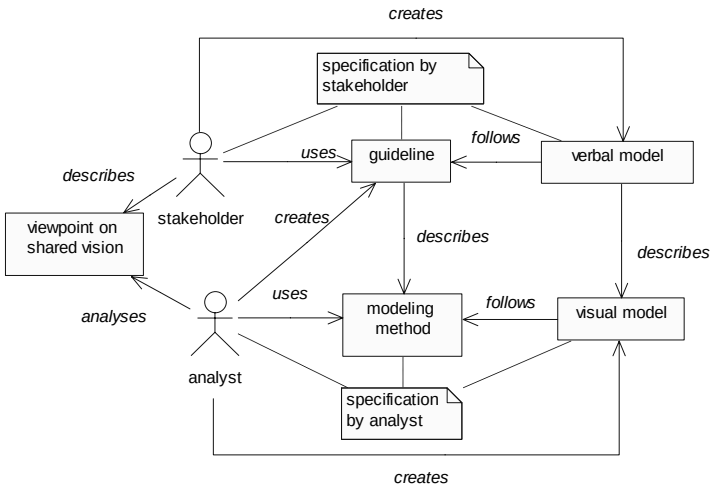

Figure 37: Specification by a stakeholder using a guideline provided by the analyst.

Instructions. The basic idea of this pattern is to have an analysts create a model and at the same time record every step in the modelling procedure.

- Let an analyst build a partial visual model from an example stakeholder specification in the target domain.
- Record all the modelling steps and transform them into a guideline. Illustrate the guideline with an example which shows how a resulting specification should look like.
- Hand the guideline to the stakeholders and let them write specifications along the guideline.
- To check the quality of the resulting specification and for continuous improvement of the guideline repeat the procedure, i.e. build and analyze a new model from the specification that was created using the guideline.

A simplified scheme for a guideline which is helpful for getting started can be summarized as:

1. define the main system use cases
2. create detailed scenarios for each use case, identifying the key objects
3. identify the relevant attributes of each object
4. create a class model and identify additional classes
5. for each class: identify the main use cases which operate on (or use) the class, continue from step 2
6. collect non-functional requirements

Application Areas. Creating specifications by guideline has been successfully used with

- Teams in long-term projects with frequent changes in the teams;
- Stakeholder groups that need highly specialized knowledge for specification, and
- Stakeholder groups of loosely-coupled individuals that are not accessible for other elicitation methods.

Constraints.

- Guidelines are better accepted by stakeholders that have not yet developed their own way of writing specifications.

Consequences.

- ↗ Guidelines enable remote specification work, thus supporting distributed projects.
- ↗ Specifications created according to a guideline have contents similar in scope.
- ↗ Guidelines offer flexibility to the authors regarding time and location of work, thus increasing the acceptance for specification tasks.
- ↘ Deficiencies in guidelines lead to according deficiencies in specifications.

Experiences.

- For quality control purposes and to improve the guidelines, some specifications were translated back into UML models. Meaningful models were achieved immediately, underlining the efficiency of the approach.
- The guidelines were used for envisioning and requirements negotiation in early project phases. At this stage, the granularity of the models were still coarse, hence efficient guidelines could be produced with reasonable effort.
- The method delivers fast results for the users and can thus be perceived as productive. It helps to improve the acceptance of RE.
- Guidelines are usually domain-specific and thus have to be re-created for every new domain. They should be improved in an iterative procedure.

Known Uses. The pattern has been observed in projects with distributed stakeholder groups who had to be able to work independently from the core project.

- **Interdisciplinary Plant Construction**

  In a plant construction project, buildings and facilities had to be jointly specified by experts from different disciplines who were distributed over several organizational units. A specification guideline was developed which followed a general scheme for creating object-oriented models. The guideline was translated into a specification template, which was then given to the different stakeholder groups. The groups nominated one responsible person each for contributing to the specification. They were using the templates, and the guidelines were used together with a central requirements engineer for quality control and improvement.

- **Introduction of COTS-based Information Systems**

  In an introduction of a COTS-based information system, stakeholder groups were given a guideline which asked them to describe their specific tasks, the information they required or produced when performing their tasks, and how the information should be entered or displayed in forms or reports. The guidelines were given to the stakeholders as “homework” assignments for ensuring progress in the specification procedure.

Related Patterns.

- “Employ a requirements engineer as caretaker” for coordinating development and distribution of the guideline, and for collecting the contributions from the stakeholder groups.
- Distribute the specification guideline to the project teams if you ”Organize Specification Along Project Structure”.

4 Experience

The following paragraphs summarize the experience gained in collecting case studies, analyzing observations and extracting and using the patterns.

4.1 Collecting Case Studies

Case studies have been collected in different ways: by interview, by telephone conference and by using templates. It was essential that they were reported by somebody from the project team who gained the experience personally. Good results have been achieved using a template in the beginning, followed by questions or an interview conducted by an analyst.

A successful approach for capturing case studies was letting the project managers decide what they wanted to report in the first place, i.e. any decision or action they took in the project which they considered noteworthy. After this initial report, an interviewer could ask for more details on specific topics which had also been raised in previous reports from other projects.

One difficulty was recording case studies in the "right" granularity, i.e. with enough details for understanding the observations, but yet focussed and short. The best results have been achieved when the observations were based on actions, and the reasons for the actions were stated afterwards. Starting with the conflict, on the other hand, often led to the same observation being included more than once in a case study, each time from slightly different viewpoints, but with the same essential information.

In summary, a recommended practice for acquiring case studies would first record the action ("what has been done") then ask for the intended objective ("what was the goal"), and finally investigate the conflict ("why was it done").

4.2 Analyzing Observations

Identifying conflicting forces is a tedious process. Frequently, a set of more than two forces seemed to apply to an observation. In such situations it was helpful to try to assign a "dimension" to each force and select those forces which act on the same dimension. For example, in the window place pattern the forces are pointing in opposite directions according to the location of chairs and windows, hence the pattern is acting on the "dimension" space (see Section 2.4). Similarly, the "prototype"-pattern (Section 2.4 and 3.2) addresses the technological knowledge or skill level of the project team. It could often be observed that during the analysis process the initial set of (sometimes more than two) forces changed to two forces which exactly pointed to the underlying conflict described by the observations.

Forces can be expressed in different styles: they can underline the facts which cause a force (e.g. presence of chairs, inexperienced users), or emphasize the effect of a force (e.g. drawn to chairs, drawn to using conservative technologies). Users seem to prefer the fact-oriented expressions for pattern access, thus it is recommended to choose this style for pattern vectors, while descriptions of the effects are better suited for understanding patterns and should thus be included in the elaborated patterns. In future, the pattern vector could be extended to include both facts and forces, but this requires further studies as there could be several facts related to the same force and vice versa.

Special attention needs to be paid to the level of detail in force specifications. Forces need to be specified at a somewhat general level to enable reusing forces in different observations as a basis for pattern discovery, while it has to be ensured that the generalization does not change the meaning or relevance of the reported observation. With a growing repository of observations and patterns the focus of the analysis shifts from extracting new forces to assigning known ones. Thus documenting the forces becomes easier.

Expressing RE experience in terms of pattern vectors turned out to be of great value to the participants in the process. For example, when the group analyzed an observation for the conflicting forces, it often revealed different underlying problems than the observer had originally assumed. In addition, the comparison with similar situations in other projects helped the participants to reflect and optimize their own way of work.

4.3 Discovering Patterns

The proposed method has been successfully used to conduct RE pattern workshops at the RE04 conference [HHP04] and at the annual meeting 2004 of the SIG RE in the GI. These events have shown that it is possible to teach the pattern mining procedure to a group of participants and discover meaningful pattern candidates in a reasonable period of time.

The workshops began with some of the attending RE experts reporting case studies from their experience. Then, one topic which had been touched by several experts was chosen for closer investigation. The experts provided additional details of their observations, and the pattern vector was used for summarizing the reports and for structuring the subsequent discussion.

Assigning the forces was a major issue. The initially reported observations were so broad that they referred to more than one conflict, which was recognized during the discussion when the group tried to understand the underlying conflict. After a few attempts, conflict statements emerged which were agreed by all the participants, and in both workshops the vector format enabled the participants to rapidly identify meaningful pattern candidates.

4.4 Using the Patterns

The WGREP members found the patterns beneficial for reflecting personal RE practices: The patterns were clarifying the reasons for using certain practices, resulting in the methods being used more consciously. The experience was confirmed at the REP04 workshop [HHP04]. Other intended future applications include:

- Training and coaching of new members in a project team after they have acquired initial text book knowledge: the patterns are expected to help getting newcomers faster to productive work
- Supporting project managers on the job: patterns offer guidance for decision making and provide arguments for acquiring RE-specific project resources
- RE process review: patterns provide a foundation for process evaluation by analyzing for example whether certain activities are established or certain conflicts are present in an organization.
- Teaching software engineering classes: patterns combine applicable engineering experience with criteria for when to use the practices

In example projects where the patterns were applied, they helped to discover potential specification conflicts, eliminate possible project risks, and clarify responsibilities. Explaining the patterns, especially the conflicting forces and their resolution, helped to increase the acceptance for RE activities. Promoting RE patterns close to project milestones, when RE becomes especially visible by providing the essential inputs for contracts, validation and approvals, ensured better management support.

It is too early to report profound experience in using the patterns – the patterns have just become available.

5 Conclusion and Outlook

After a one year period, WGREP has completed its intended program and achieved its goals. The group has successfully developed and demonstrated a format for describing RE patterns and an RE "pattern mining" procedure. Fourteen case studies have been collected which contain more than eighty observations, and an initial set of fourteen RE pattern candidates has been identified. About half of the candidates have been elaborated into patterns and are available for review by external readers.

The need for easy access to well-proven RE experience has been confirmed by the strong interest from different communities in the group's activities. This observation supports the initial motivation that many persons being responsible for RE in small and medium projects deal with the same challenges.

Follow-up activities are being prepared to improve and extend the RE pattern collection. Quarterly RE pattern workshops will be held with the aim of improving the existing and finding new RE patterns. The workshops will address RE experience for specific topics, like e.g. contracting requirements (client - supplier relationships) or requirements reuse. They will initially be held in Germany, but may also be conducted in conjunction with international conferences.

An online RE pattern repository, REPARE [REP05], is being set up which uses the pattern vector for navigating and filtering the patterns. It enables pattern retrieval e.g. for specific RE tasks or project roles and provides navigation between related patterns. The forces are used to enable project managers to retrieve patterns for a specific project situation. It is envisioned that project managers provide a set of characteristics of their current project conditions to retrieve in turn a list of applicable patterns as a recommendation, and it is intended to provide mechanisms for user contributions to the repository as an additional way of collecting feedback and further patterns.

References

[AIS77] C. Alexander, S. Ishikawa, M. Silverstein, *A Pattern Language*, Oxford University Press, 1977.

[Ale79] C. Alexander, *The Timeless Way of Building*, Oxford University Press, 1979.

[BE00] J. Bergin, J. Eckstein, et al., The Pedagogical Patterns Project, http://www.pedagogicalpatterns.org/

[BMR+96] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal, *Pattern-Oriented Software Architecture – A System of Patterns*, Wiley & Sons, 1996.

[BPS02] Best Practical Solutions LLC, RT Request Tracker, http://www.bestpractical.com/rt/

[CH01] J. Coplien, N. Harrison, et al., Organizational Patterns Page, http://www.bell-labs.com/cgi-user/OrgPatterns/OrgPatterns?OrganizationalPatterns

[EP00] H.-E. Eriksson, M. Penker, *Business Modeling with UML – Business Patterns at Work*, Wiley & Sons, 2000.

[GHJV95] E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns*, Addison-Wesley, 1995.

[HHP04] L. Hagge, F. Houdek, B. Paech, "International Workshop on Requirements Engineering Patterns (REP'04)", Sept. 6, in conjunction with 12th International IEEE Requirements Engineering Conference (RE'04), Sept. 6-10, 2004, Kyoto, Japan, http://rep04.desy.de/

[HL05] L. Hagge, K. Lappe, "Sharing Requirements Engineering Experience Using Patterns", IEEE Software 22, 1 (2005), 24-31.

[Kau04] M. Kauppinen et al. "Implementing Requirements Engineering Processes throughout Organizations: Success Factors and Challenges", Journal of Information and Software Technology 46, 14 (2004), 937-953.

[KCH+90] K. Kang, S. Cohen, J. Hess, W. Novak, A. Peterson, *Feature-Oriented Domain Analysis (FODA) Feasibility Study,* Technical report CMU/SEI-90-TR-021, Software Engineering Institute, Carnegie Mellon University, 2000.

[KS98] G. Kotonya, I. Sommerville, *Requirements Engineering – Processes and Techniques*, Wiley, 1998.

[MFM+96] C. Mazza, J. Fairclough, B. Melton, D. de Pablo, A. Scheffer, R. Stevens, M. Jones, G. Alvisi, *Software Engineering Guides*, Prentice Hall Europe, 1996.

[Moz98] The Mozilla Organization, Bugzilla, http://www.bugzilla.org

[OMG03] Object Management Group: UML 1.5 Specification. http://www.omg.org/-technology/documents/formal/uml.htm

[REP05] REPARE, *Requirements Engineering Patterns Repository* REPARE, http://repare.desy.de/

[RR03] S. Robertson, J. Robertson, *Volere Requirements Specification Template*, Edition 9, 2003, http://www.systemsguild.com/GuildSite/Robs/Template.html

[SIG04] FG RQ, *Homepage SIG Requirements Engineering of the German Informatics Society (GI)*, http://www.iese.fhg.de/gi_fgre/ (in German)

[Som05] I. Sommerville, "Integrated Requirements Engineering. A Tutorial", IEEE Software 22, 1 (2005), 16-23.

[SS97] I. Sommerville, P. Sawyer, *Requirements Engineering - A Good Practice Guide*, Wiley, 1997.

[STT03] Reports from and Contributions to the SIG RE Annual Meeting 2002 in Softwaretechnik-Trends, http://pi.informatik.uni-siegen.de/stt/23_1/index.html