4 Rationale as a By-Product

K. Schneider

Abstract: Rationale is an asset in software engineering. Rationale is communicated during several project activities, like design or prototyping. Nevertheless, very little rationale is captured today. There seems to be an inherent tension between creating or externalizing rationale, and capturing it successfully. In this chapter, the "Rationale as a By-Product Approach" is defined through seven principles. Those principles were identified while building two applications. In both, tools were tailor-made to support capturing design rationale on the side while working on software project tasks as usual. The approach is best applied to project tasks that create or elicit a lot of rationale.

Keywords: capturing rationale, by-product, task-specific path, FOCUS

4.1 Introduction

Rationale is among the most important information a software project produces. It is important to know why a decision has been made and why one design or solution has been preferred over another. Later decisions are facilitated by knowing why earlier decisions have been made. During maintenance, documented rationale can save a large percentage of effort. Chapter 1 introduced many important aspects of when and how to use rationale.

However, capturing rationale is not straightforward. *The most productive* project phases in terms of decisions and concepts are the *least likely* to accommodate opportunities for documenting rationale. Exactly at the point in projects where most design decisions are made, documentation is often not a high priority. All available resources and time slots are devoted to the product but none is devoted to (or "wasted on") documentation or rationale.

Capturing "rationale as a by-product" takes those constraints into account. A number of principles describe the core of this approach. Selected human interactions are recorded in several modes in parallel: In addition to audio or video recording, specific "paths" are recorded and reused to index the large amount of audio/video data. A path is a time-indexed sequence of elements (e.g. code modules) visited during the human interaction. For example, the sequence of all modules explained by an expert is logged as one such path.

In Sect. 4.2, some situations are described in which rationale is built and communicated. The Rationale Paradox describes the phenomenon that usually none of the surfaced rationale gets captured where it occurs. Section 4.3 defines what is meant by the "Rationale as a By-Product Approach." There is a general definition and explanations of the principles that make up the approach. Related work is mentioned in this context. Two instantiations of the approach are introduced as examples: Sect. 4.4 addresses software prototypes, and Sect. 4.5 is devoted to risk management. The approach is discussed in Sect. 4.6. Section 4.7 concludes.

4.2 Origins of Rationale in Software Projects

As described in Chapt. 1, there are many uses for rationale in software projects. But where and when do different kinds of rationale surface? Where could they be captured?

4.2.1 Rationale Occurs when Decisions are Made

Visions, requirements, and reasons for them first appear in the earliest project phase. Communication in this phase is typically based on informal meetings, slide presentations, and oral discussions. After a while, more formal requirements engineering takes over.

Design decisions are mainly discussed by technical experts and architects during the design phase. Decisions are made by groups and by individuals. They are typically communicated through overview charts, architecture sketches, and oral explanations.

Prototypes are often used to decide between design alternatives. Different kinds of prototypes were differentiated by Lichter et al. [12]. The types of information provided by those prototypes have also been analyzed [16]. Prototypes spark insights that add to the rationale for technical decisions. Demonstration prototypes elicit customer requirements and rationale.

During the entire project, *requirements* are further negotiated, prioritized, and rearranged [1]. Some of these activities will require initial design proposals or prototypes. Reducing *project risks* is a constant task in project management. Identified risks may cause design decisions. Different stakeholders may disagree on requirements or risks – probably disagreeing on deeper assumptions and rationale as well. Compromises must be found that will be accompanied by rationale. Much of the above-mentioned rationale *resides in the heads of project participants.* Rationale is seldom documented.

Documenting rationale in a systematic way has long been an issue in software engineering. The Potts and Bruns model [15] was used by Lee [11] to describe rationale. The result resembles a specific kind of "semantic web" with qualified relationships, similar to ontologies [20]. However, a sophisticated (and maybe "intrusive") representation calls for effort, time, and resources to build and to maintain. This chapter advocates a very different approach. In analogy to agile methods in software engineering [2,3], light-weight approaches to rationale capturing were studied and adopted. *Light-weight* indicates a clear priority to save time and effort *from the perspective of bearers of rationale*. This reduction of effort is afforded by sophisticated preparation and tools: tailor-made recording software is used, and masses of data are recorded just to capture some of the above-mentioned valuable rationale. The trick is to pick the right occasion and the right indexing-mechanism ("paths") for each specific activity observed.

4.2.2 The Rationale Paradox

Since rationale is so essential for project success, one would expect it to be highly regarded and captured carefully. However, that is seldom the case, as Chap. 1 states in some detail and with reference to the literature. Due to its perplexing nature, I call this observation the "Rationale Paradox":

> ***The Rationale Paradox:***
> When most rationale is created, chances to capture it are lowest.

This paradox is supported by a number of observations:

- Rationale is created when key decisions are made.
- During decision-making, participants are very attentive.
- Rationale is considered important and "evident" at the time when it is created. At that time, no one can imagine how it could ever be forgotten.
- Usually, further decisions are based on earlier ones, so there is pressure to continue fast in the project. New decisions overlay old rationale.
- Csikszentmihalyi [5] talks about the *flow state* in which knowledge and experience come together easily and knowledge workers seem to "flow" through their highly demanding work. During the flow state, knowledge workers are typically not willing to switch tasks and take care of rationale.

- Schön [19] and Fischer [7] discuss how practitioners need to be interrupted in their professional activities in order to become aware of the tacit (*internalized* [14]) expertise they currently apply, including experience and rationale. However, interrupting their flow (as in [19]) might endanger motivation and will slow down work. To avoid this, the team tends to focus on "essential constructive tasks" in the project – while capturing rationale is deferred.

When the project gets into a slower phase, rationale will already be partially forgotten (see above), so there is again little motivation to document it. Most software developers prefer doing what they consider "productive work" like designing or programming over documentation. They will rather make new decisions and continue designing or implementing than capturing rationale. As a consequence, rationale is least likely to be captured when it would be easiest to grab.

4.3 Rationale as a By-Product

Chapter 1 described many situations in which rationale can be beneficial in software projects. Section 4.2 indicated different situations in which rationale is communicated within a project. In many cases, there are only talks, telephone calls, or a few sketches in which the rationale is ever being made explicit. Usually, very little rationale is captured and documented. According to the above-mentioned Rationale Paradox, this is not an accident but an inevitability.

The approach presented in this paper is a generalization from several attempts we made at two different universities and a company to face the above-mentioned challenges of capturing rationale. The two applications stated below (FOCUS and Risk Analysis) are the most advanced implementations that incorporate the idea of "Rationale as a By-Product."

4.3.1 Definition of the By-Product Approach

The term *approach* refers to a set of guiding principles for someone to follow in order to achieve a certain goal.

The *By-Product Approach* is defined by two *goals* and *seven principles:*

Goals

(1) Capture rationale during specific tasks within software projects
(2) Be as little intrusive as possible to the bearer of the rational

Principles

(1) Focus on a project task in which rationale is surfacing
(2) Capture rationale during that task (not as a separate activity)
(3) Put as little extra burden as possible on the bearer of the rationale (but maybe on other people)
(4) Focus on recording during the original activity, defer indexing, structuring etc. to a follow-up activity carried out by others.
(5) Use a computer for recording and for capturing additional task-specific information for structuring
(6) Analyze recordings, search for patterns
(7) Encourage, but do not insist on further rationale management

All together, the principles shift effort away (1) from the time when project tasks are being carried out and (2) from experts and bearers of design rationale. Therefore, it may look from their perspective like the rationale is really "captured as a by-product of doing normal work." This is what counts. It justifies the name "By-Product Approach".

The style of describing a "method" or "approach" by a list of interconnected principles was successfully used by Beck in his widely known description of eXtreme Programming [2].

The principles respond to the challenges mentioned in Sects. 4.1 and 4.2. They were inferred from observations and hypotheses in the above-mentioned attempts to capture rationale. Like in Beck's description of eXtreme Programming, principles are not fully comprehensible by reading their titles only. In the remainder of this section, each principle will be explained with respect to the entire approach. Neither goals nor principles may sound extremely new or innovative. The difference is in their details and their combination.

For the purpose of the following discussion, a *learner role* is introduced. A learner in this context is a person who will need to use a certain kind of rationale in the future. Without support, a learner might simply talk to the bearer of the rationale and search additional material to read. This is a tedious task, as experts are often busy or not available. There may be a larger group of learners sharing similar interests and information needs. Instead of asking the same questions again and again, capturing rationale and keeping it persistent will assist in distributing it. Moreover, by focusing and supporting the teaching process the By-Product Approach is intended to pay off even with only one learner involved.

The By-Product Approach can be applied to different situations and activities in software engineering. It helps to identify rewarding activities and to design specific computer support. To build those software features,

substantial technological preparation is required. It reduces effort during the capturing step so that rationale seems to be captured "as a by-product."

4.3.2 Principles and Related Work

Each of the principle is now explained. Reasons for each principle are provided. By referring to related work, the principles are further clarified.

Focus on a Project Task in which Rationale Surfaces

The approach uses an existing task to capture rationale, called the "focus task". This refers to a selected task or activity that is part of the usual software process – not one inserted for the favor or rationale capturing or rationale management. Many experiences in real projects support the Rationale Paradox: during interesting project phases, even the slightest additional "task" will not be accepted. Therefore, no additional task is inserted.

Section 4.2 mentions different kinds of decisions made during different project activities. Principle 1 requires focusing on *one* such focus task in which the desired type of rationale is created or discussed. Rationale is said to *surface* when it is discussed, documented or communicated, either in phone calls, meetings, or prototype demonstrations.

In the terminology of Chap. 1, the approach is concerned with descriptive rationale, and there is a clear commitment to avoid intrusions during that selected project task. Approaches like gIBIS [4] impose argumentation structures and extra tasks on project personnel. That is carefully avoided in the By-Product Approach.

Capture Rationale During that Task

It is important to capture rationale where it surfaces. Waiting to capture it later will probably fail: Much will be forgotten, and project pressure will force people to prefer project tasks over rationale management duties.

This principle may seem to contradict the previous one: what is the difference between inserting an extra rationale task (above) and capturing rationale *during* an existing task?

An important psychological issue is the need to schedule and carry out an additional task in the first case, while in the second case there may only be a small percentage of extra effort during the existing task, with no additional time slot needed. Of course, this principle by itself does not reduce the effort of capture, it just increases acceptance. However, the next principle calls for reduction of this extra effort as the *main optimization goal* – even at the cost of sophisticated preparation and lengthy follow-up

work. Once again, this may not reduce overall effort, but it does reduce effort during the rationale-prone project task.

It has been argued (see Chap. 1) that IBIS captures rationale "on the fly" [13], just capturing the history of rationale as it occurs. While the By-Product Approach is descriptive in that same respect, it puts far more emphasis on low effort.

Put as Little Extra Burden as Possible on the Bearer of the Rationale

This principle makes a clear statement about the distribution of effort within the team. It is especially important that those who are the sources of rationale be spared the extra work of capturing it. This is contrary to many rationale capturing approaches that assume the experts will have to do most of the work (several mentioned in Chap. 1).

Grudin's seminal work [9] on "who is the beneficiary and who does the work?" reminds us to design work processes with the benefits and efforts of all stakeholders in mind. Grudin claims (originally in the field of CSCW) an approach will not be successful if some people are charged with extra work, while others receive all the benefits. Projected on capturing design rationale, the bearers of rationale will see little personal benefit in sharing or even documenting what they know. It is an old lesson from knowledge management that there may be incentives beyond money to create benefit. Demonstrating to experts the appreciation for their knowledge and help has often been a valuable benefit to them [6]. During our work with the two applications described later, the bearers of rationale recognized and appreciated our obvious attempts to save them time. Nevertheless, some effort needs to be invested for capturing and structuring. As a consequence, someone else has to do it, and at a later time. This differentiates the By-Product Approach from others that attempt to distribute the effort more "equally". However, benefits and potential contributions are not distributed equally, so why should efforts be? It is a conscious decision of this approach to let those people do most of rationale management work who benefit most from a well-structured base of rationale. Those who need the rationale are the ideal people to do that job.

Of course, there is a limit to all principles. When a learner has made an attempt to organize material, there should be the option for a feedback session. The expert could meet the learner and look through the results, as long as the expert is still available. The By-Product Approach and the tools developed to support each of its instantiations will support this feedback and provide a good basis for structuring and indexing (as explained later). When there is no time or opportunity for such a session, the By-Product

Approach will try to continue without: “raw” material in the form of paths can often be used since paths follow a well-known structure of products or work-processes. It is, in fact, not so raw. Skipping feedback will decrease the learning value, but not to zero (Principle 7).

Most of the extra work load for capturing rationale is shifted away from the focus project task, and most of the remaining rationale-related duties are assigned to learners or observers rather than bearers of rationale. Here is the capturing bottleneck.

As Chap. 1 points out, many approaches have shifted from intrusive to less-intrusive variants. We consider it important to distinguish the roles and balance effort, duties, and (potential) benefit, with a clear focus on relieving experts from any extra work.

Focus on Recording Rationale First

This principle is rather concrete compared to the first three principles. It resembles more a practice than a principle in Beck´s terminology [2]. It describes one contribution to fulfill the first three principles: The main rationale-related activity is supposed to be recording, but recording of many different kinds (e.g. audio, video, event traces, paths, and structures used, see example cases).

According to Principles 2 and 3, recording devices and environment need to be set-up in a nonintrusive way. Recording must either be trivial, or the recording devices should be operated by a learner (beneficiary of rationale transfer, see Principle 3).

Use a Computer for Recording and for Capturing Additional Task-Specific Information for Structuring

This principle differentiates the approach from simple recording. While audio or video recording would not necessarily require a computer, this principle demands the recordings to be computerized (at least in the end). But there is more to this principle: the more one knows about the focus task at hand, the more additional information can be recorded *on the side*. There is often an internal structure associated with a task or a discussion. For example, the table of content of a document under discussion, the agenda of a meeting, or the file structure of a software project provide hooks and opportunities to refer to.

Since we know that the focus task is related to software engineering, and due to focusing on only *one* task, typical structures can be identified and used. Assume a meeting in which requirements are discussed with a customer. At some point, participants point to a requirement in a DOORS

database, at another point they execute and comment a prototype. Time-stamped paths facilitate cross-referencing between DOORS document structures, code execution traces, and oral comments by the participants. What happened at the same time may be related. Analyses built upon the combined recordings will add value beyond simple replay, but will also require substantial up-front implementation work.

Natural language understanding, or any sophisticated form of artificial intelligent, is not the purpose. Time-stamped recordings are used as time-indexed paths through the discussion space. Given structures and paths provide an additional perspective on recorded rationale (e.g., code structure or DOORS-links).

There is a lot of similarity to approaches like domain-oriented design environments [8], but this principle is less general and more specific than DODEs. By stressing path and structures typical for the focus task at hand, the principle helps the user of the approach to narrow down on an issue.

Analyze Recordings, Search for Patterns

Additional paths and structures are captured while audio or video sources are recorded. For example, simple recordings can be replayed. In the end, there are several different recordings from *one* recorded session, e.g. a sequence of DOORS requirements discussed (sequence of Req.-IDs), specification structure (requirements within table of contents), and audio recording of discussion (time-indexed stream). All those parallel recordings are related through time stamps.

It is straightforward to link all recordings together for browsing, with an option to jump from one track of the record (audio, video, paths) to the other at common time-stamps. Looking at different perspectives (at the same recorded time) or following any of the paths creates an extended exploration space for learners. At the same time, the network of paths and structures is always associated with plain audio or video records that contextualize and explain things. One of the main values comes from guiding learners within a complex structure, such as a document or program.

We have also explored the opportunity to let a program search for suspicious or interesting patterns. In the FOCUS example, only a few trivial patterns were used, concerning hot spots (frequently executed or explained elements) and path deviations (when a method is executed but never explained) (issue explained but never executed).

Encourage, but do not Insist on Further Rationale Management

Most approaches about design rationale include capturing as well as indexing and structuring. Raw data is considered unreadable and unsuited for learner use by some [13]. The task of abstracting and structuring raw data into more manageable rationale is indispensable.

According to this principle, the By-Product Approach is different. The recordings and paths and structural information usually add up to a large amount of data. A single Camtasia (screen video and audio) record of a one-hour meeting may easily be 100 MB large. However, there will be only a few of those essential meetings, and only a few recordings. This principle again represents a conscious and rather extreme decision: Do not care about a few Gigabytes of storage space, when they fit easily on a 2$-DVD. If no one takes the initiative to further extend or modify or transcribe recordings, the web of recorded "raw" data from one session might just be burnt on a DVD and represent a snapshot of the project history. If desired, it can always be loaded back into the computer and updated. It was our initial intention to keep the rationale alive over an extended period of time. During the experiments with the two case examples, we had to accept that this rarely happens. In most cases, the effort required for creating a snapshot is much less than the effort needed for continuous rationale management. This approach was shaped by observations in software projects and optimized from a pragmatic point of view. Continuing rationale management is certainly desirable from a methodological perspective.

From Principles to Practices

As with agile methods [2,3], principles are guidelines to follow. For each concrete instantiation of the approach, principles need to be turned into concrete, operational practices, techniques, or rules. In that sense, each of the principles explained earlier can be implemented quite differently.

The following two applications show different instantiations of the approach. First of all, the focus tasks are different (prototypes and risk management). Consequently, relevant rationale looks different and needs to be captured in a different way. The principles help to approach both cases.

4.4 Case 1: Capturing Rationale in Software Prototypes

FOCUS is a strategy and a family of tools to capture knowledge sparked by prototypes [16]. According to Lichter et al. [12], there are different kinds of prototypes that are built with different goals in mind.

The definitions of various types of prototypes are listed below, with respective rationale mentioned in parentheses.

- *Demonstrators* elicit requirements (and rationale for raising those requirements) from customers.
- *Prototypes proper* try out implementation ideas (soliciting design rationale) implementing the core functionality only.
- *Breadboard prototypes* try out single technical solutions in isolation. They produce insights in how to fulfill a requirement (and why!).
- *Pilot systems* start out as prototypes and slowly turn into product software (all kinds of rationale play a role during this full development that shares all aspects of other prototypes).

FOCUS was initially created to solve the specific problem of capturing knowledge from prototypes in a light-weight way [16]. Experience elicitation in software projects may follow a similar approach [17]. All those findings were compiled and generalized to the "Rationale as By-Product Approach," weaving in related other approaches like LIDs [17] or Collaborative Risk Management [18]. The different kinds of prototypes elicit different aspects of rationale (as differentiated in Sect. 4.2). In the terminology of Chap. 1, it is basically "supporting knowledge transfer" (Sect. 1.4.4) that is supported by FOCUS.

4.4.1 FOCUS Implementation of the Principles

Where Does Rationale Occur?
When one of the above types of prototypes is selected, a certain kind of information and rationale is sought. During prototype development, further rationale is created (*why to do it that way?*). During development, the flow state [5] may be reached, and an interruption will hamper the creation process. However, as soon as the prototype is presented to other people, developers will use this opportunity to talk about their findings and successes. Observers have a chance to ask questions. This is a good opportunity to capture and record rationale. We have experimented with separate tape recorders and with computer-based audio and on-screen video recording. FOCUS now uses the Camtasia commercial tool to record both a screen video and audio of the explanations given

(http://www.techsmith.com/). Demos often convey highly condensed information, far beyond "raw rationale".

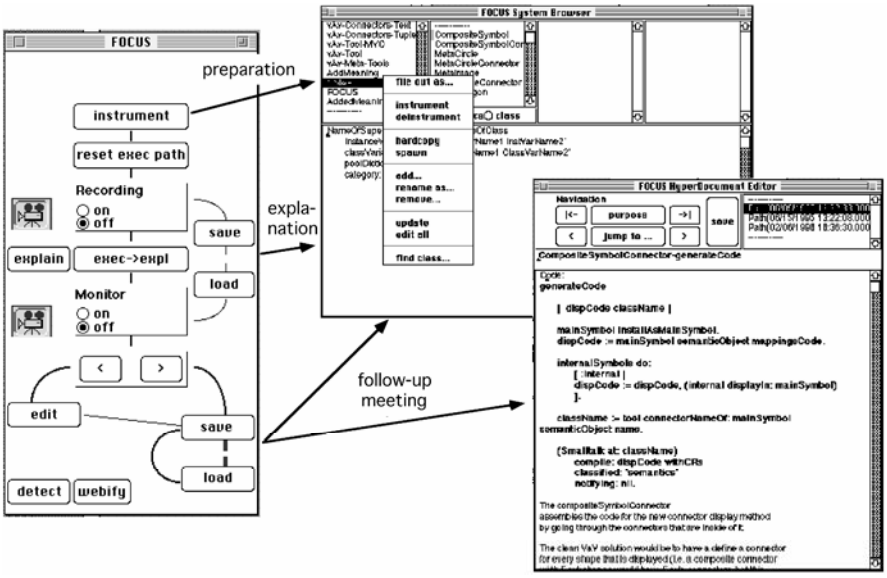

Fig. 4.1. From the FOCUS panel (*left*) commands are issued to control a code browser (*top*), code execution (not visible) and the rationale reader (*right*)

How to Shift Extra Effort Away from Experts?

There is very little extra effort from their perspective: The recording software is integrated in the FOCUS panel (see Fig. 4.1). Experts giving the demonstrations just follow the lines from top to bottom and press a few buttons, in order to start or stop recording.

A typical FOCUS use case follows the buttons in the FOCUS panel: (1) a learner or expert has marked the code to be discussed. After pushing the "instrument" button, all methods within that piece of code will be traced when executed. (2) A demo is started and recorded as both a Camtasia (video) file and as a sequence of executed methods (specific path). (3) The path may remote-control the code browser to guide follow-up explanations. In that case, method by method is displayed that was previously executed. During this second part of the demo, experts explain how the demoed features were implemented. (4) This explanation is again recorded via Camtasia into an "explanation path," which is a sequence of methods visited.

This original implementation was carried out in Smalltalk (Fig. 4.1). It is important to use an integrated environment that is used for writing,

running, and explaining code. Smalltalk was such an environment. However, FOCUS as an instance of the By-Product Approach is not restricted to any single language. To demonstrate this, we recently completed an Eclipse/Java version of FOCUS (see Fig. 4.2). Eclipse is a widely used Java development platform. Four Eclipse plug-ins (integrated platform extensions) were implemented to allow instrumentation and tracing of selected methods, linking and replay of different recorded paths and videos in an integrated way.

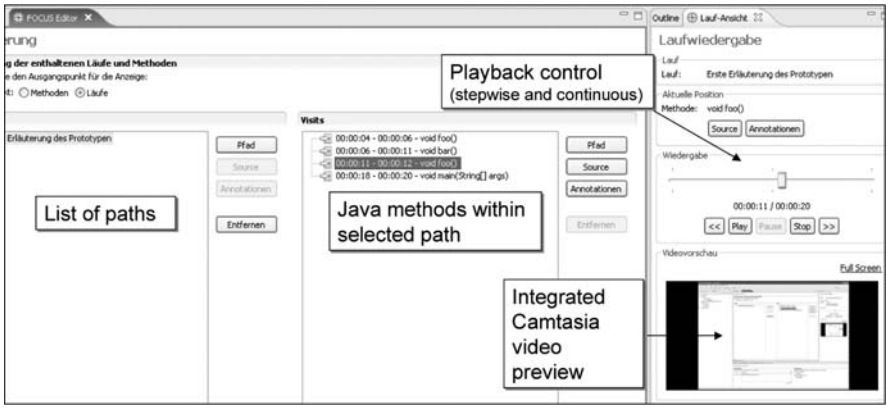

Fig. 4.2. Eclipse/Java version of FOCUS: On the *right* of the FOCUS window, there is a small Camtasia preview window (can be enlarged), buttons for stepwise replay of paths (method by method), and a slider for fast navigation. Of course, Camtasia videos with attached paths can also be replayed in continuous mode. Users may browse the entire web of paths and recordings at any point

In Fig. 4.2, a list of previously recorded paths provides access to the methods they consist of. From each recorded method, a learner can explore all paths that include that method.

Who will Benefit?

A prototype is created to answer questions about customer requirements or about technical options. Usually, only a small subset of developers is involved with prototyping, but their findings are used in a much larger team. Any person in that larger team who needs to learn about the prototype is an ideal candidate for transcribing or summarizing the audio record – if it is ever done.

What can be Captured During What Task?

That same computer that runs the demo, also executes Camtasia and path recording (see above). A main advantage is a perfect synchronization between explanations (audio), what is explained (video), and what part of the code is actually affected (path).

Additional Computer Recording or Analysis?

Prototype code is instrumented with tracing information. By running the code, a trace of executed (Smalltalk or Java) methods is created *as a by-product*. The sequence of executed methods is called an "execution path".

Recordings of explanations and screen video were synchronized with the execution paths and explanation paths through time stamps. The specific strength of FOCUS comes from its integration with the development platform, and code base. By being fully integrated, the code structure (inheritance, packages) is related to the execution and explanation paths through the methods executed or explained.

In addition, simple patterns can be detected by FOCUS. It may ask for rationale on "all methods that were explained but never executed." There is no "artificial intelligence" involved, just pattern matching. No attempt was made to have the computer explain a pattern. Explaining remains a task for human experts, as would be the case in a normal demonstration.

4.5 Case 2: Risk Analysis

In the next example, a very different task in software project management is supported by the "Rationale as a By-Product" approach. Risk management is a crucial project task to avoid running into foreseeable problems. For example, a subcontractor may have been unreliable in the past. Relying on this same contractor in a new project is a risk: it could cause a delay, and maybe contractual fees. Risk analysis is at the core of risk management. It deals with reasons and probabilities and consequences of risks. Since there is uncertainty involved, different stakeholders may use different reasoning (rationale) in assessing those risk parameters. In this phase, the Risk Analysis Tool comes into the picture.

Where does Rationale Occur?

During the discussion by stakeholders (project leader, experienced project staff), a lot of the previous experience made with the subcontractor or with other risks surfaces. Discussions elicit risk mitigation options.

How to Shift Extra Effort Away from Experts?

Usually, risk analysis is carried out in a regular project meeting as a separate topic. However, risks should be discussed frequently. When the project is running, risk analysis may only focus on the changes since the last meeting. Risk meetings are short, but they require that many stakeholders participate. A larger project might be distributed over different locations or buildings, causing traveling expenses.

We developed a Risk Analysis Tool that enables the team to save time by holding risk analysis meetings online. The tool offers a user interface displayed in Fig. 4.3. At the core, there is a chat facility (left) and

a portfolio (right) on which risks are placed with respect to their probability and their impact. Numbered circles represent risks. They are described after the portfolio. Relative positions imply different priorities for mitigation. Portfolios are the typical tool for discussing risks during risk analysis [10].

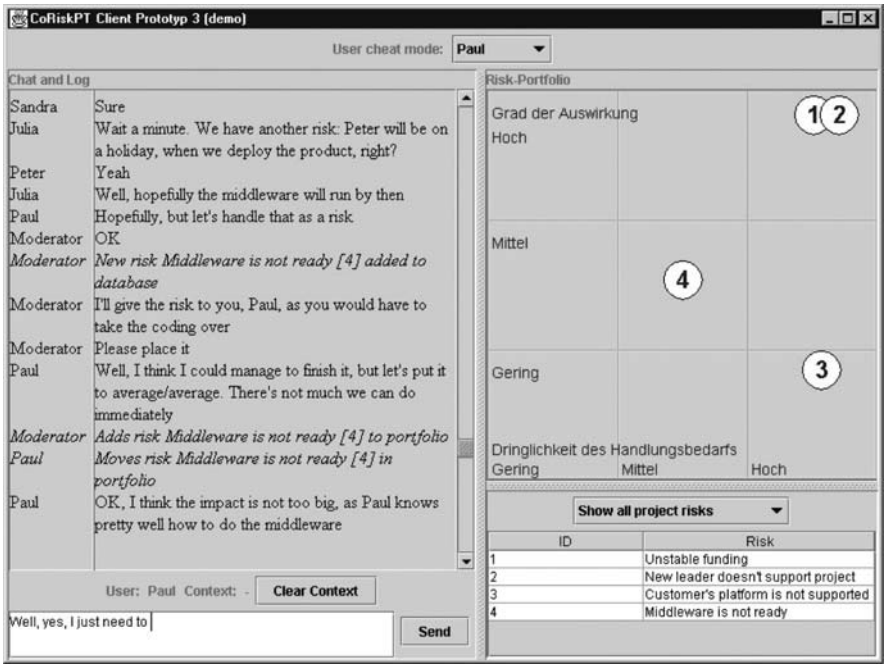

Fig. 4.3. Risk Analysis Tool. Integrates chat (*left*), portfolio (*right*), and recording control (*hidden*). Risks are represented by circles and listed below portfolio [18]

What can be Captured During What Task?

In the tool, an interactive risk portfolio is combined with a chat component. Both components are time-stamped and recorded. Since many comments refer to the same set of risks (on the portfolio), all chat contributions that mention a certain risk can be identified and cross-referenced. Of course, there could be a NetMeeting or Voice over IP component instead of the chat component. Nothing would change in principle as long as all relevant activities are recorded, time-stamped, and related. Again, the By-Product Approach can be implemented in several different ways. We used chat as the easiest option and because it can be easily demonstrated on paper and slides.

Who will Benefit?

Participants who would like to remember the meeting and the course of the discussion without taking notes. Future project members who would

like to learn about earlier concerns and discussions on risks. New team members trying to understand the project better, are good candidates to summarize the recordings, if at all necessary. Project leader assistants could carry out those things as well, as they usually have to keep track of the project status.

Additional Computer Recording or Analysis?

Recorded sequence of events (chat contributions, risk movements) can be filtered and presented in different forms. Besides a simple replay, filtering for participants or for individual risks is afforded. Also, the path of a risk circle on the portfolio during discussion can be visualized. For example, a risk may have entered as low-probability, low-impact in the lower left corner, and now follow an increasing curve to higher probability and impact: participants move it up as they agree on increased risk exposure. Such a pattern might be detected by either the tool or a human user when looking at the path. Without the tool, such a path may escape attention, especially when it consists of several smaller shifts. Once detected, a pattern like this will trigger high-priority risk mitigation actions.

4.6 Discussion

The concept of capturing rationale "as a by-product" was stimulated by the FOCUS project, the Risk Analysis Tool, and some smaller projects. Similar challenges and similar opportunities lead to similar solutions. In this paper, the "Rationale as a By-Product Approach" was factored out and explicitly described. This approach provides guidance in setting-up tools to capture rationale by recording project tasks. The approach is pragmatic in taking constraints and observations from practice seriously – and accepting that some nice features will probably not be used a lot.

The two applications were developed in university and industry environments, respectively. They have been applied to different projects, and have demonstrated the *feasibility* of the approach: there were no fundamental breakdowns or objections, and some participants were delighted. However, we consider this anecdotal evidence and recommend empirical validation of future tools built according to the Rationale as a By-Product Approach. With our new Eclipse implementation of FOCUS, we plan to validate the approach with a series of increasingly rigid experiments. We are aware, however, that a full validation will take months or years: the tools will only unfold their full potential when the bearers of rationale are no longer available or cannot remember what was recorded. So far we have seen only short-term effects.

From developing and applying techniques like FOCUS [16], Risk Analysis Tool [18] or LIDs [17], the merit of the by-product approach becomes obvious: in all cases, there is almost no interruption of work. There is a low threshold for rationale bearers to use the techniques, as they require little or no extra work. There is always a form of recording "raw information" that contains valuable rationale. Mere recording and replay is already an important support for capturing rationale, but hardly deserves being called an "approach". But when computer tools are enriched with further recording features, and when the resulting web of paths is offered as guidance, a new level of support is reached. Indexing by time-stamps and task-specific paths are crucial means for fast retrieval. Pattern matching and advanced analysis and presentation facilities can add yet another step. The two cases give some concrete examples.

4.7 Conclusions

The approach presented in this paper focuses on the capturing and "saving" aspects of rationale management. By focusing on one essential project task, it is highly restricted and very specific. Due to that small focus, powerful recording mechanisms (e.g. executed program methods, risk movement events) can be identified and supported.

Care needs to be taken to reintroduce the rationale when it is needed later. It would go beyond the scope of this paper to discuss this aspect in depth, but the approach obviously facilitates presentation of rationale, too. Paths could be visualized, and they always should be used to browse the space of recorded material. In that respect, the approach typically leads to "Rationale Capturing" components that are tightly integrated with path-oriented "Rationale Retrieval" components (as defined in Chap. 1).

Capturing rationale "as a by-product" sounds easy, but requires sophisticated technological preparations. Rebuilding FOCUS within Eclipse, for example, consumed more than four persons–months of a highly skilled software developer. Obviously, there is no way to get rationale for free. This approach simply shifts all the effort into building a computer tool like FOCUS or the Risk Analysis Tool and away from the actual project task in which rationale surfaces.

"Rationale as a By-Product" is an approach for building tools and techniques that have a realistic chance of being accepted and successful in real projects.

References

[1] Alexander IF, Stevens R (2002) Writing Better Requirements. Addison-Wesley, Reading, MA
[2] Beck K (2000) Extreme Programming Explained. Addison-Wesley, Reading, MA
[3] Cockburn A (2002) Agile software development. Addison-Wesley Reading, MA
[4] Conklin J, Begeman ML (1988) gIBIS: A hypertext tool for exploratory policy discussion. ACM Trans. Off. Autom. Syst. 6(4): 303–331
[5] Csikszentmihalyi M (1990) Flow: The Psychology of Optimal Experience. HarperPerennial, New York
[6] Davenport TGP (2000) Knowledge management case book – Best practices. Publicis MCD, Wiley, München Berlin Heidelebrg New York, Germany, p. 260
[7] Fischer G (1994) Turning breakdowns into opportunities for creativity. Knowledge-Based Syst.. 7(4): 221–232
[8] Fischer G (1994) Domain-oriented design environments, Autom. Softw. Eng. 1(2): 177–203
[9] Grudin J (1987) Social evaluation of the user interface: Who does the work and who gets the benefit. In: INTERACT'87. IFIP Conference on Human Computer Interaction. Stuttgart, Germany, pp. 805-811
[10] Kontio J (2001) Software engineering risk management – A method, improvement framework, and empirical evaluation. In: Department of Computer Science and Engineering. Helsinki University of Technology, p. 248
[11] Lee J (1991) Extending the Potts and Bruns model for recording design rationale. In: International Conference on Software Engineering, ICSE-13
[12] Lichter H, Schneider-Hufschmidt M, Züllighoven H (1993) Prototyping in industrial software projects – bridging the gap between theory and practice. In: International Conference on Software Engineering (ICSE-15), IEEE Computer Society Press, pp. 221–229
[13] MacLean A, Young RM, Bellotti VME, Moran T (1996) Questions, options and criteria. In: Moran TP, Carroll JM (eds.) Design Rationale: Concepts, Techniques and Use, Lawrence Erlbaum Associates, Mahwah, NJ, pp. 21–52 Moran T, Carroll J (1996) Design Rationale: Concepts, Techniques, and Use. Lawrence Erlbaum Associates, Mahwah, NJ
[14] Nonaka I, Hirotaka T (1995) The Knowledge-Creating Company. 17 ed. Oxford University Press, Oxford
[15] Potts C, Bruns G (1988) Recording the reasons for design decisions. In: International Conference on Software Engineering, ICSE-10
[16] Schneider K (1996) Prototypes as assets, not toys. Why and how to extract knowledge from prototypes. In: 18th International Conference on Software Engineering (ICSE-18) 1996. Berlin, Germany, pp. 522–531

[17] Schneider K (2000) LIDs: A light-weight approach to experience elicitation and reuse. In: Product Focused Software Process Improvement (PROFES 2000). Oulo, Finland, Springer, Berlin Heidelberg New York, pp. 407–424
[18] Schneider K (2001) Experience magnets – Attracting experiences, not just storing them. In: Conference on Product Focused Software Process Improvement PROFES 2001. Kaiserslautern, September 2001, pp. 126–140
[19] Schön DA (1983) The Reflective Practitioner: How Professionals Think in action. Basic Books, New York
[20] Van Heijst G, Schreiber AT, Wielinga BJ (1997) Using explicit ontologies in KBS development. Int. J. Hum.–Comput. Stud. 46 (2–3): 183–292