

Business Process-based Requirements Modeling and Management

Mario Pichler

*Software Competence Center Hagenberg
Hauptstrasse 99, A-4232 Hagenberg, Austria
mario.pichler@scch.at*

Hildegard Rumetshofer

*FAW Software Engineering gGmbH
Hauptstrasse 119, A-4232 Hagenberg, Austria
hrumetshofer@faw.uni-linz.ac.at*

Abstract

In this paper, we are emphasizing our perfect appreciation of using visualization means in requirements engineering as key success factor in any complex (software) development project. We are discussing our experiences with requirements engineering visualization, in the large part gained in the course of a three years multidisciplinary software development project in the insurance domain. A business process-based requirements modeling and management approach that has been developed and employed within this project is described. In order to overcome some process-related and tool integration issues to refine this approach, we investigated a recently released requirements engineering software of a prominent vendor. Experiences using this tool, tool integration for requirements engineering visualization, and suggestions for future research are discussed extensively.

1 Introduction

The goal of any product development project is to build a product that is useful and useable for the users [12]. To make this happen, the stakeholders' desires and needs have to be carefully elicited, analyzed, specified, validated—above all, understood—and, last but not least, realized by a development team. It is common that the task of transforming the stakeholders' wishes into a format (or notation) that is understood by the development team is done by requirements engineers and (business) analysts. It is also known quite well that in normal case, customers (and even users) are 'speaking another language' than the development team does. For the objective of translating the customers' and users' language into the development teams' language, the translators (i.e. requirements engineers and analysts) are using different notations. Thereby, the usage of various visual modeling notations is quite useful [13]. However, traditionally, the primary (or even only) notation used is text (often unstructured and informal, in better cases formal and

template-based).

In [11], a report on the challenges and experiences gained during a three years multidisciplinary software development project in the insurance domain, focusing on the employed requirements process is given. Beyond using visual requirements modeling techniques for translation purposes, in the cited paper, the application of requirements modeling [3, 7] as powerful quality assurance technique is recommended. Applied rightly in early project phases, it helps to reduce requirements errors and, therefore, contributes to project success.

In the paper at hand, we report on what we learned throughout the above-mentioned three years project with regard to requirements engineering visualization. We present a business process-based requirements visualization approach to improve communication and negotiation processes among customers, users, and development teams (Section 3 and 4). In Section 5 and 6, we reflect on the experiences we gained by supporting our visual requirements engineering process with a novel requirements definition tool of a prominent vendor. Thereby, focus is laid on the experienced shortcomings, in order to identify and stimulate ideas for further research. A conclusion and an outlook on our future research concerns in requirements visualization are given in Section 7. In the following Section 2, we shortly describe the project and especially the reasons which the approach presented later in the paper emanated from.

2 Background

We begin this section by briefly describing the project context from which we derived the ideas presented later in the paper. For more details, we refer to [11]. In the second part of this section, we describe how we used models in early project phases as communication means with future users of the system to be developed and justify our appreciation of using visualization techniques for requirements engineering.

2.1 Project Context

In this paper, we refer to a project that has been carried out together by the nation-wide operating Austrian Social Insurance for Occupational Risks (AUVA) [1]—the customer—and two research and development institutions, represented by the authors.

The AUVA treats approximately 3 million employed persons and 1.3 million school children and students. A central motivation of the AUVA is the prevention of occupational accidents and diseases. One major element of the prevention activities are the visits of prevention workers (experts in a specific domain, e.g. noise control, and medicine) in companies, to suggest them possibilities to improve the employees' safety and health conditions at their workplaces. The goal of the AUVA for the project referred to herein is to provide software support for—up to now, partly manually handled—work processes in the prevention domain.

One of AUVA's major requirements is that the developed software represents a significant assistance for the users, rather than to require additional workload from them. Therefore, an iterative, evolutionary, user-centered development process, allowing strong user participation and short feedback cycles has been considered as optimal for reaching the project goal. This approach has been negotiated among the contractors and the IT department of the customer company.

2.2 Early Project Stages and Tools

In order to understand the users' working domain, we conducted several site visits, whereby we primarily observed the users doing their work, asked questions if something was unclear, and took paper notes [4]. However, most requirements have been elicited during workshops with users in meeting rooms at the customer's site. Different sites were visited, as the customer—as already mentioned—is operating nation-wide.

Supporting Argument for using Visual Modeling (Tools)

In one of the first meetings with a selected user group, the goal was to elaborate their current work processes. We—as analysts—decided in advance to lively visualize the findings in the workshop to let the users know that we really relied on their support. At the beginning of the workshop we suggested this approach to the users, who agreed. At this point, it is important to mention that the computing skills of the participants were on average, some of them were in a leading position (e.g. head of department), and none of them were familiar with UML modeling. However, they found it very positive to visualize their working processes. During the visualization work, we introduced them the used notation. The users were very positive, because they understood

the meaning of the used elements very fast. We assume that was because of the simplicity of the used notation. As Select Component Architect (SCA) by Select Business Solutions was predetermined by the customer, we used two diagram types offered by the mentioned tool for business process modeling. These are i) process hierarchy diagram and 2) process thread diagram, the first one for visualizing mainly static information, and the second one for (dynamic) process flows.

As users' feedback of using visual models to capture our understanding of their working domain and elicitation of requirements was quite positive, the same approach had been used to describe the future working processes as intended by the management and to discuss them with future users. Furthermore, we applied them in subsequent workshops, while we continuously tried to improve the used processes.

Need: Requirements Management (RM)

In early project phases, concept and requirements documents prepared by the customer and the already described visual models acted as primary sources for requirements elicitation and analysis. At that time, documents were copied to the project folder in the file system and the visual models resided in SCA. However, we needed a single access point—a single source [11]—to all the requirements-related information. It was clear from the very beginning of the project that requirements will come from different sites of the customer and there will be a high number of requirements. For these reasons, usage of Microsoft Word and Excel—while still the most used tools for requirements management—seemed not to be suitable for the project. SCA—the modeling tool predetermined by the customer—provided some basic functionality for requirements management. However, the performance of SCA as requirements management tool did not meet our needs (e.g. no support for requirements hierarchies).

We decided to use a professional requirements management tool. The contractors had previous knowledge about Requisite Pro by IBM and DOORS by Telelogic. At the time, when suitable tools were investigated, only CaliberRM by Borland offered integration with SCA. Following, after initial investigations of CaliberRM's functionalities, it was chosen as requirements management tool for the project.

Challenge: Visual Models and RM

One major observation in using visual models was that they have always been stimulating lively discussions among different stakeholders. Early examples for that were group discussions of business process models, mind maps for needs and features investigation, use case diagrams, and discus-

sions of graphical user interface (GUI) mock-ups and prototypes.

As result of these discussions, numbers of new as well as changes to already documented requirements appeared. We wanted to keep track of the sources of requirements. For this reason, we needed the possibilities to i) derive requirements directly from elements in our visual models and ii) keep track of such model element-requirement connections. In the early project phase, the CaliberRM-SCA integration was very restrictive. It was in one direction only—from CaliberRM to SCA and not vice versa. Therefore, the former of above mentioned needs was not satisfied. Hence, the additional benefit of an integration of the modeling tool and the requirements management tool was rather minimal.

2.3 Summary and Implications

At this point, we want to stick to the importance for the project referred to in this paper to i) model requirements and ii) manage the requirements in an adequate tool. By the way, we believe this represents a beneficial approach in general. The integration level of requirements modeling and management—as mentioned above—was rather suboptimal in early project phases.

Therefore, throughout the project—as kind of action research—we have continuously tried to improve our requirements (visualization) process on the one hand, and tool support for integrating our requirements models with the management of requirements on the other hand.

In the following section, we describe our requirements engineering process in detail.

3 Business Process-based Requirements Engineering

One of the customer's major requirements was to develop software that represents significant assistance and supports daily working processes of the intended users. Subsequently, we pushed working in close cooperation with them and based our process on techniques such as 'contextual inquiry', 'contextual design' [4] (see also Section 2.2) as well as 'usability engineering' methods and processes according to Mayhew [8] and Nielsen [10]. Thereby, we considered techniques as proposed by the usability engineering lifecycle [8] to come from the client's vision to the gateway of software design and implementation: user and user task analysis, construction of working model or design of first mock-ups. These concepts build the basis for our proposed business process-based requirements engineering methodology.

The software to deliver was planned to support daily workflows in the customer's organization. That these processes are better illustrated visually than textually has

been attested by the existence and widespread usage of visual design tools for business process modeling and goes along with statements given by E.-M. Melchior "*People do not really know what they want until they try it*" [9] and S. and J. Robertson ([12], p. 217) "*I do not know what I want but I will know it when I see it*". We experienced that users feel uncomfortable and burdened when asked to review textual use case documents. However, their reactions are totally different when asked to review use case diagrams. Insightful discussions were generated and feedback was prompt, valuable, and reusable for us analysts. These visual models and, furthermore, the usage of prototypes seem to be a perfect means of gathering feedback and collecting (eliciting) new requirements. Subsequently, they allowed us to continuously improve our knowledge model about the required product. Furthermore, we identified and improved a methodology for business process-based requirements engineering as illustrated in this section.

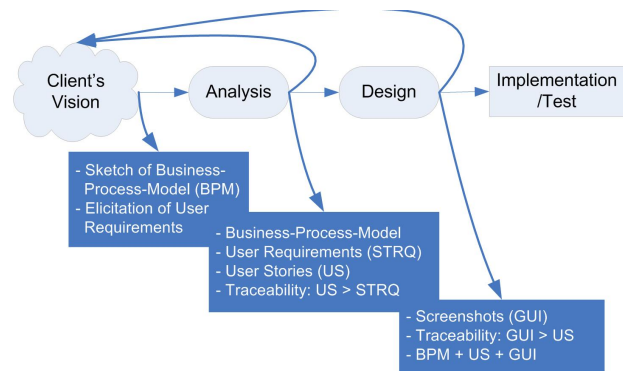


Figure 1. Business process-based requirements engineering.

The proposed requirements engineering process aims at transforming the client's vision (or parts of it) to artificial models. It is two-fold beneficial: first, these models illustrate to the client what he actually wants and, second, they serve as basic input for the chief architect and his development team who are responsible for its implementation. The intended process encompasses four main steps, which are client's vision, analysis, design, and stop or go for implementation/test (see Figure 1). Besides, each workshop and discussion either with customer, users or members of the development team was dedicated to deliver a particular result, e.g. visual representation of the business process model, common understanding of this business process or decision for stop or go. Although, this goal-orientation provoked heavy preparations for the analysts, it was worth the effort as it allowed to deliver valuable results and to elicit 'real', 'stable' user requirements.

Each of these process steps bases on visual models and

prototypical screenshots to foster communication and discussion with users. In the following, these steps are described in more detail. As example, we provide one selected part of the client's visions: the audiometry (in German: 'Audiometrie'). The audiometry summarizes activities, which are performed by AUVa engineers by means of examining auditory skills of industry workers in case they are noise exposed.

3.1 Client's Vision

It has to be stated that the selected part of client's vision had originally been described and documented textually by the customer. These requirements for the audiometry business process—telling us how it should be—were collected in the requirements management tool Borland CaliberRM. However, it was still nebulous and unclear for us analysts because these lines gave much room for interpretations. Besides, these descriptions were provided at the very beginning of the project but relevant for a later iteration (more precise about two years later). Consequently, after two years we faced two challenges: first, available requirements were incomplete, and, second, it was unclear if they were still valid or obsolete. Subsequently, our first mission was to consolidate these available requirements.

The first workshop was done in form of discussions with the representative of the customer's management. We wanted to understand how it should work from the customer's point of view and to update our knowledge about the appreciated business process. The result was documented on a whiteboard: a first visual draft of the business process model representing the audiometry by means of actors, process steps, and artifacts.

Having these images in mind, we entered into the second workshop with a group of user representatives aiming at understanding how it should work from the user's point of view (see Figure 2). Fortunately, imaginations and desires of both management and users were similar. Nevertheless, we identified and elicited further user requirements such as exceptions in the workflow, requested reports or additional actors, while discussing the evolution of the visual model of the audiometry business process on a flip chart. Furthermore, first mock-up-ideas were sketched as well.

By the way, we realized that being prepared for such workshops demands for at least two persons (better three: one customer representative) as a combination of analyst, mediator, and 'notes taker'. Being back in our offices, elicited user requirements were documented in CaliberRM and old, invalid ones were deleted or updated as well. Consequently, input for the 'analysis step' was available in form of the visual business process model represented on the flip chart and textual descriptions of user requirements in CaliberRM.



Figure 2. Sketch of business process model.

3.2 Analysis

Although, the business process model was developed in detail in these workshops, we considered work re-engineering as illustrated in Mayhew [8] for optimizing workflows as well as improving the model's quality and consistency. Thereby, the visual business process model drawn on the flip chart built the basis for the visualization in Borland Together Architect—we used this tool for research purposes. We modeled the business process as activity diagram.

The transfer from paper to digital media was a valuable quality check. This activity diagram provided the big picture for relevant process steps, details, and exceptions—an ideal basis for discussions and further tasks: it was used to identify 'user stories', to prepare ideas for mock-ups, and to derive more detailed requirements from it (see Figure 3). We were using stakeholder requests that had been collected in preceding workshops to break down the single process steps into more fine-grained 'user stories'. We could also apply 'use cases', but we recently chose to align our descriptions more on stories than on use cases. The 'stories' in most cases are short—single sentence—descriptions of required system functionality. As we appreciated to manage all requirements in a single source (tool), we decided to use CaliberRM as leading tool for management.

The Borland Suite allows to elicit requirements from diagrams visualized in Together Architect and to integrate them automatically in CaliberRM. Unfortunately, this automatic integration is only available when designing use case diagrams. This means that requirements in CaliberRM can only be elicited from a use case in Together Architect. We were using an activity diagram for the business process model. This fact demanded for a work-around: on the one hand, we documented the user stories manually—as requirements—in CaliberRM and on the other hand, we attached notes, which represented these user stories, to the particular step in the business process model in Together Architect. Besides, in CaliberRM a user story was traced

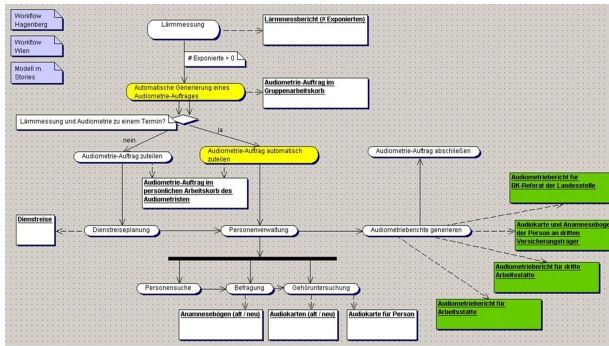


Figure 3. Business process model.

to the stakeholder requests, which triggered this story. Consequently, input for the 'design step' was available in form of the visual business process model represented as activity diagram enhanced with user stories (as notes) in Together Architect as well as textual descriptions of user stories in CaliberRM.

3.3 Design

Being prepared with the business process model and a set of user stories, which described the requested system functionalities, we entered the design phase. Our first designs were inspired by ideas, which had already been developed in the preceding workshops. By the way, paper is a quite suitable tool for visualizing screen ideas. It took us many sheets to develop convenient paper-mock-ups that based on the determined business process steps and defined user stories.

However, as the designer was familiar with the development tool (Microsoft Visual Studio 2005), these paper mock-ups were easily transferred to digital screens. This transfer from paper to digital media was once again a valuable quality check and an improvement of the paper design ideas.

The developed screens were integrated in CaliberRM as graphical user interfaces artifacts (short: GUI)—for documentation and traceability purposes. Additionally, these screens were attached as notes to the particular step in the business process model in Together Architect (similar to the user stories in the analysis step). We provided traces from a GUI element to the requested user stories in CaliberRM. Consequently, input for the 'stop or go step' was available in form of the visual business process model represented as activity diagram enhanced with user stories (as notes) and screens (as notes) in Together Architect as well as images and textual descriptions of screens in CaliberRM (see Figure 4).

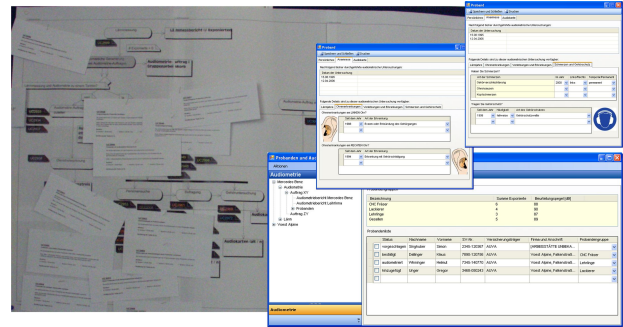


Figure 4. Business process model, user stories and screens.

3.4 Stop or Go for Implementation

The aim of the fourth step was to present the findings so far to both the customer and the development team. We appreciated to get the customer's decision for stop or go and to prepare the members of the development team for the estimation workshop, focusing on identifying and estimating system requirements (tasks) for implementation. Primarily, we used the Together Architect HTML Export as preparation for the planned presentations. This export is quite convenient to extract visual diagrams and links for documentation purposes. However, we realized that this media is uncomfortable by means of presenting huge visual business process models in interactive workshops. Besides, the usage of different tools for analysis and design provoked a further workaround with respect to illustrating our findings to both groups. Consequently, we used following artifacts in these presentations:

- Business Process Model (BPM) as activity diagram in Together Architect (prepared as A1 paper poster)
- User stories as textual descriptions (prepared as post-it papers to stick them on the BPM poster)
- Screens as images (prepared as A4 paper posters to stick them on the BPM poster)

The BPM poster was hung up on the wall and user stories and screens were stuck on it. Although, we followed the business process model step-by-step by attaching the corresponding user stories to the particular step, it was interesting to notice that most questions, discussion points, and answers were stimulated by watching the visual screens on the A4 paper posters. These presentations were extremely valuable for us analysts as we consolidated common understanding for the audiometry business process of all participants, elicited some further user requirements, and cleared misunderstandings at an early stage. In this step of our business process-based requirements engineering approach, we

were rewarded by both the customer who stated *"I cannot find any mistake"* and the members of the development team who declared the results as *"very useful for further development"*.

4 Identified Issues

As positive feedback about the sketched process was given by the customer, users, and members of the development team, we are convinced that this approach is worth considering in further projects. However, while identifying, changing, and improving activities for the here presented business process-based requirements engineering method, we discovered a list of issues, which are illustrated in this section and motivate what comes later in this paper.

Most important to mention are difficulties because of changing media: documents, whiteboard, flip chart, papers, and tools such as CaliberRM, Together Architect or Microsoft Visual Studio for screen design. It is evident that the usage of different media is necessary, especially with respect to paper, flip charts or whiteboards for sketching first ideas. However, when starting to integrate user requirements, scenarios, business process models into a single tool, it would be appreciated to work with these artifacts in any other tool—and vice versa—as well.

Consequently, we considered Together Architect on a research basis for integration with CaliberRM. Unfortunately, the combination did not live up to its promises with respect to modeling business processes, scenarios, and deriving requirements from the visual models and providing traces to CaliberRM. Not to mention the weak connection of GUI mock-ups, requirements (user stories), and the models in Together Architect. This changing media provoked our workaround for the presentations of the business process model, the screens, and the user stories as described in the previous section.

Subsequently, the usage of several tools provoked that (requirements) models did not live any longer once requirements were derived. Therefore, we faced the risk of inconsistencies between requirements managed in CaliberRM and visual models designed in Together Architect (Designer). Nevertheless, for the sake of consistency, reusability, and traceability, single sourcing of both requirements and visual models would have been appreciated.

Just when being aware of these issues, Borland has announced the availability of a new, challenging tool. This announcement seemed to solve our problems as we assumed support through a sophisticated suite of tools. Subsequently, the next sections illustrate our findings and experiences with Caliber DefineIT as 'workaround' for our challenges.

5 Experiences regarding Tool Support for Sketched Process

In the previous section, we gave an overview of the business process-based requirements engineering approach that has been developed and applied in the course of a three years software project. While we are optimistic with the process as such—sure, there is room for improvements—, tool integration to optimally support the described process is still an issue for further work.

Especially, the weak connection of requirements models, GUI mock-ups and screens, and additional information handled in a requirements management tool presents a challenge. In our case, remaining workarounds (see previous section) are concerned with i) export of requirements from models into a requirements management tool and ii) presentation of GUI mock-ups, screens, and design ideas to selected customers and users to collect feedback. The annotation of single process steps with mock-ups, screens, and design ideas and a suitable presentation and feedback gathering approach represent issues of further research.

For this reason, it made us prick up our ears when Borland released the first version of its requirements engineering software Caliber DefineIT (we will use DefineIT as abbreviation further on) in May 2006, designed to assist business analysts and requirements engineers to elicit, analyze, and validate system requirements [6]. Furthermore, DefineIT provides storyboarding capabilities that allow interactive scenario simulation in early development stages.

In the following, we present our initial experiences of applying DefineIT to support the above mentioned (tool) issues of our requirements engineering process. The business process for demonstration means, again, is the audiometry process, as already introduced and used in Section 3.

5.1 Usage of Borland Caliber DefineIT

At first, it is necessary to mention that the targeted requirements type of DefineIT are *system* requirements—an issue, we seemingly did not pay enough attention to initially. Our intention was to use DefineIT to model business processes as scenarios and, further, derive (user) requirements from them. Therefore, we were somewhat confused when prompted to first create a requirement to be able to model a scenario rather than to create a scenario beforehand. DefineIT is meant to use a scenario to model (or to visualize) how a particular feature of your system (defined by the corresponding requirement) works in real time ([5], p. 19, Scenario). Therefore, it is assumed that a (user) requirement is already there, and is to be analyzed through a scenario model to derive system requirements.

Client's Vision: Scenario Modeling

Our approach, as described in the previous section is different. We intend to use scenarios to visualize business processes and, through discussions with customers and users, derive user and later system requirements from the models.

For this reason, we created a requirements elicitation project, named 'Audiometrie', and a requirement called 'Audiometrie Prozess' within DefineIT. As result, beneath the requirement 'Audiometrie Prozess' that acts as placeholder for the corresponding business process, a scenario with two steps 'Start' and 'End' is created (see Figure 5). Then, we were prepared to model our audiometry process as scenario within DefineIT.

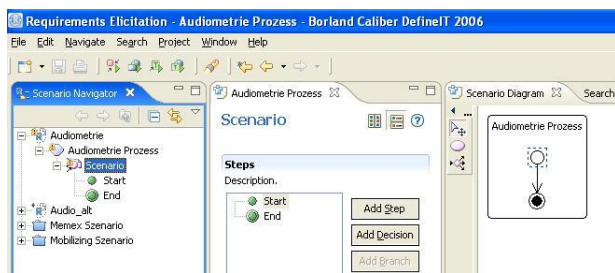


Figure 5. Newly created requirements elicitation project 'Audiometrie'.

Three views are shown in the perspective in this figure: Scenario Navigator, Scenario Editor, and Scenario Diagram. Scenario Navigator provides navigation aid within or across scenarios and requirements. Scenario Editor and Scenario Diagram (a graphical representation) are for scenario modeling. Provided model elements are scenario step, decision, and branch elements (in the editor and in the diagram). In the Scenario Editor, further properties for a chosen scenario element are managed, e.g. for a scenario step element, a (text) description field and two comboboxes, one to choose an actor and another one to invoke other scenarios. A fourth view, the Resources View, is shown on the lower left part in Figure 6. It is to attach images, URLs or files to single scenario steps.

See also the result of our modeling work in Figure 6.

Analysis: Requirements Export to CaliberRM

An analysis of the modeled (business process) scenario to derive more fine-grained 'user stories' represent the next step of our process. Together with the single scenario steps, user stories should be managed within CaliberRM as well.

Interesting for us, DefineIT provides functionality to extract a new scenario out of a scenario step. Therefore, it is possible to break down a scenario (a business process) step

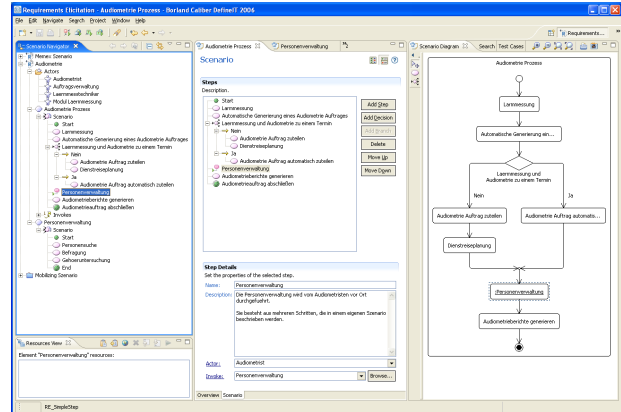


Figure 6. Audiometry process modeled as scenario in Caliber DefineIT.

into fine-grained user stories. In more detail, for each step that is extracted into a separate scenario, a new requirement (called as the step it was extracted from) with a subsequent scenario for modeling the details is created.

Requirements, further on, can be exported to CaliberRM. However, as we use requirements in DefineIT as placeholders for whole business processes (modeled as scenarios), this is not a satisfying solution for us. Export works as follows: the requirement name is used to create a like-named requirement in the CaliberRM requirements tree (desired position can be chosen) and the scenario diagram is pasted as image into the requirement description in CaliberRM. Single scenario steps are only referred to in the image further on. If the steps should also be exported as requirements, they have to be extracted into single requirements beforehand. This represents a situation we are not fully satisfied with. Another hindering issue, with respect to our process, is that in this way we cannot distinguish between business process steps and user stories, as the only available model element is 'scenario step'.

Concluding, the expected support for eliciting requirements (in our case 'user stories') out of scenario models and exporting and linking them to CaliberRM meets our needs only to a limited degree.

Design, Stop or Go: Storyboarding of Scenarios

We are addressing the process steps 'Design' and 'Stop or Go' together in the following. This is because the issue of weak connection of GUI mock-ups and screens, process models, and requirements in CaliberRM influences both steps. We are investigating how our workarounds for presentation of business process models, GUI mock-ups, and screens (see Section 3 and 4) can be supported by DefineIT.

After analyzing the scenario that we have acquired to-

gether with the customer and users, we are creating first design ideas. Especially important is the development of GUI screens. DefineIT offers the possibility to attach resources (files, URLs, images) to single process steps. Therefore, we are attaching created screens, within DefineIT, to the corresponding scenario steps.

The result is a business process, modeled as scenario, refined by child scenarios (user stories), and enhanced by GUI screens.

Using the storyboarding functionality of DefineIT, we are interactively presenting the scenario step-by-step to the rest of the development team. Without the need to print out the whole process model and corresponding screens, we are able to discuss the process flow and associated screens and dialogs (see Figure 7).

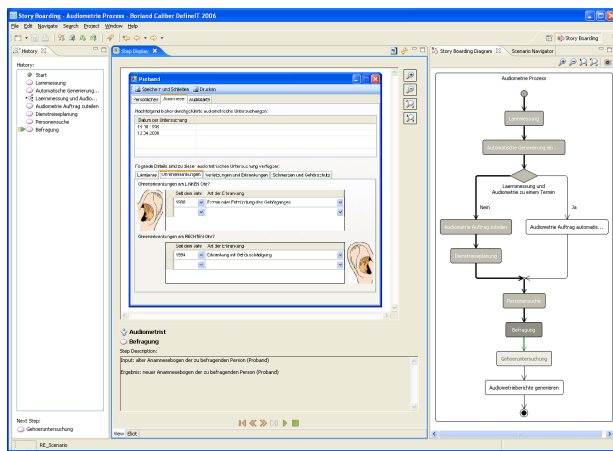


Figure 7. Storyboard execution of audiometry process in Caliber DefineIT.

Also the feedback process with customers and users benefits from storyboarding. The whole scenario model including the storyboarding functionality can be exported to an HTML site. As our users are distributed over the whole country, shorter feedback cycles including the requirements (and changes to them) of all stakeholders are possible.

From a requirements engineer's and analyst's perspective, DefineIT's storyboarding functionality adds value to our requirements engineering process. At the time of writing, we are investigating the stakeholders' satisfaction with this discussion and feedback mechanism.

6 Discussion

Some issues addressed in the previous sections need to be discussed on a more detailed basis. We provide further experiences of applying Borland Caliber DefineIT to support the open issues of the requirements engineering process

described in Section 3. Thereby, we are structuring the discussion in advantages and shortcomings we experienced. Furthermore, we reflect on tool support for requirements visualization and the integration with requirements management more generally. However, as we gained experiences, we also highlight this issue in the context of a tool vendor, and discuss our considerations regarding the provided tool chain for requirements visualization and management.

6.1 Advantages of Caliber DefineIT

Scenario modeling gets our perfect appreciation as an approach supporting requirements definition (elicitation, analysis, specification, and validation) and as means supporting communication among customers, users, and development teams in early project stages.

In its current version, we experienced DefineIT as a simple tool for this purpose, because it is not overloaded with questionable functionality.

Derivation of requirements from the model (scenario) and the integration with CaliberRM for requirements management is a valuable feature. Especially, the possibility to export requirements from the model into CaliberRM is valuable. However, as already mentioned, the implemented approach is different to our perception of requirements elicitation.

The feature which we most appreciate is scenario storyboarding. This feature adds most value to our requirements engineering approach as sketched in Section 3. Exporting a scenario together with the storyboarding functionality into an HTML representation especially supports our discussion and feedback cycles with customer and users, as they are spread across the whole country. Therefore, it addresses one of our process issues to be improved, namely 'weak connection of GUI mockups, Designer (Architect) models, and CaliberRM requirements' (see Section 4).

For describing another experience, we shortly leave the project context as introduced in Section 2.1. Thereby, we are addressing the issue of requirements visualization (modeling) as quality assurance technique [11]. In another project, a storyboard has been created using Microsoft PowerPoint. When trying to transfer this storyboard into DefineIT (see *Memex Scenario* in Figure 5), we realized that some of the Powerpoint slides included topics for different actors. Through the constraint of DefineIT to allow only one actor per step, we had to revise these steps—leading to two scenario steps in DefineIT. At the outset, we perceived this issue as a 'tool-caused' restriction of our modeling processes. However, by considering this 'restriction' in more detail, we came to the conclusion that it might be helpful to restrict each scenario step to just one actor. Reducing each scenario step to only one actor leads to more atomic steps and, therefore, might lead to a better separa-

tion of different requirements, eventually.

Another issue we experienced was that reengineering the original PowerPoint storyboard in DefineIT helped us to identify similar functionalities which originally were spread across the whole scenario. In DefineIT, through scenario hierarchies, it is possible to group functionalities that belong together.

At this point, we re-enter our primary project context.

6.2 Shortcomings of Caliber DefineIT

While we appreciate the simplicity of DefineIT in its current version, it might be helpful to be provided with more modeling elements. Currently, only scenario steps and decisions (branches) are supported. In our case, as we try to model business processes as scenarios, model elements for business objects, data objects etc. would be helpful.

A further minor, but powerful improvement would be support to use different colors for single model elements (e.g. scenario steps). As can be seen in Figure 2, for instance, we used different colors in our process sketches to distinguish between business process steps, intermediate results, and final process results. Colors can easily be used to bring further meaning into a graphical representation.

Other minor, but worth mentioning, issues regard 'inlining' and extracting of scenarios and display of requirements numbers in DefineIT. Inlining a scenario in another one works, but an inlined scenario cannot be extracted to the scenario it was inlined from; instead, a new one is created. Such 'inline/extract cycles' would be beneficial for scenario presentations in the Scenario Diagram view. CaliberRM Navigator within DefineIT does not show hierarchical and serial numbers as used in CaliberRM. Therefore, it is difficult to identify CaliberRM requirements in DefineIT.

6.3 Tool Integration for Requirements Visualization

One major challenge we experienced by using a requirements management tool (e.g. CaliberRM) and a requirements visualization tool (e.g. Together Designer, Caliber DefineIT), is to identify the leading tool. While there still are other integration problems, an important question to answer is which tool provides the primary access to requirements information. While different tools provide different viewpoints on requirements, a 'single source of information principle' turned out to be a major success factor in our (multidisciplinary and distributed) software project. An interesting question for instance is, in which tool links to further requirements (or generally spoken, project information) are managed, in the requirements management tool or in the visualization tool? Or should it depend on the contents of linked targets? One can argue that these questions

suggest that a better integration of those tools should stand on top of a research agenda.

Position of Caliber DefineIT in Borland's Tool Chain

Our original perception was that Caliber DefineIT is intended to be used to elicit or to derive user requirements from scenarios. According to the sketched requirements engineering process in Section 3, we started by modeling business processes, discussed them with customer and users, and elicited user requirements in this way. Usage of DefineIT for modeling and derivation of user requirements, therefore, would be beneficial for us. We repeatedly experienced that requirements emanate from discussions of business process models (or scenarios).

However, according to Borland's descriptions, DefineIT is intended to use scenarios to model requirements (we assume, user requirements) in more detail and to derive system requirements from the scenario model. *"Create scenarios for the requirements in your project and export them to CaliberRM with the Scenario Diagram attached to the requirement description"* is written in DefineIT's help system ([5], p. 26). Therefore, it is assumed that user requirements are already there (and managed within CaliberRM). The question for us now is, where did the user requirements come from? If we used DefineIT in the specified way, it would mean that we create a scenario for an already existing requirement in CaliberRM. Afterwards, we are modeling the requirement as scenario using DefineIT. And at last, we export the already in CaliberRM existing requirement (as it came from there) to CaliberRM again!? Sure, the scenario diagram is now attached to the requirement as image. But is this the only enhancement?

Let us take another perspective. Let us assume, we are using Together Architect (or Designer) for business process modeling. Then, we can use the models to derive requirements and export them to CaliberRM. At this point, we have user requirements in CaliberRM, which, in turn, can be modeled in more detail in the modeling tool to elicit system requirements. The system requirements are then exported and managed in CaliberRM as well. The question now is, what is DefineIT in this scenario good for?

In fact, DefineIT offers possibilities to export requirements elicitation projects to Together Architect, either as business process modeling notation (BPMN) project or as UML2.0 activity diagram. It is unclear—at least for us—for which reason a requirements elicitation project should be exported to the given destination formats. We consider BPMN to be used for requirements engineering in the problem domain. Thus, for the elicitation of business and user requirements. If now, a DefineIT requirements elicitation project—with the goal to derive system requirements—is to be exported into a BPMN project, we would consider

this as step backwards, from elicitation of system requirements to elicitation of user requirements in the V-Model [2], for instance. The question now is, why not integrating selected DefineIT functionalities into Together Architect (Designer)? Scenario modeling and, especially, storyboarding would be valuable features in Together Architect as well.

We want to mention a last issue concerning tool usage for requirements engineering visualization. Tools are without any question necessary to support a requirements engineering process as described in Section 3 and also given the basic conditions (project context) in Section 2.1. However, it is often useful to use paper, pen, whiteboard, flipchart etc. in a first meeting with customers and users. A software tool often restricts the degree of freedom one needs in early project meetings (taking notes, visualize them; usage of different colors etc.). If information captured in such trivial form is later entered into any software, this step already represents a quality assurance measure.

7 Conclusion and Further Work

In this paper, we argued that from our point of view—at least, our experiences suggest that—visualization is a meaningful way to support a mutual understanding among stakeholders in a project (cf. Section 2). Furthermore, as visualization encompasses a number of different artifacts throughout the whole project lifetime, we see requirements visualization as key success factor (and communication means) in any development project (note that most what we perceive in the world through our sensory system comes from our eyes—a *picture says more than 1000 words*).

In Section 3, we presented a requirements engineering approach that bases on this attitude and utilizes business processes as root for requirements visualization activities and artifacts. However, there are some issues for further improvement, especially with respect to end-to-end tool support for the sketched process. In Section 4, these issues experienced in a three years multidisciplinary software development project in the insurance domain are listed and described. We analyzed a recently released requirements engineering software (Caliber DefineIT by Borland), as preliminary information about this software suggested that it could be a reasonable choice to provide help for these open issues. Our experiences and analysis results are described in Section 5. Caliber DefineIT offers interesting functionality to experience the system to be developed during the early development stages through interactive scenario simulation (*storyboarding*). This feature adds value to our process, as it supports the requirements elicitation, discussion, and feedback cycles with customers, users and the development team. However, it is not clear for us, where and how Caliber DefineIT fits into Borland's tool chain, especially considering the fact that CaliberRM and Together Designer are

already there. A discussion on this and further topics with regard to Caliber DefineIT can be read in Section 6.

Our research agenda for the future suggests placing emphasis on reusing requirements models throughout the whole project (development) lifecycle. Especially, we are thinking about exploiting requirements visualization to support project management [12].

Acknowledgements

The authors gratefully acknowledge support by the Austrian Government, the State of Upper Austria, and the Johannes Kepler University Linz in the framework of the *Kplus* Competence Center Program.

References

- [1] AUVA: Austrian social insurance for occupational risks. <http://www.auva.at/>. Last visited: June 2006.
- [2] V model: Development standard for IT-systems of the Federal Republic of Germany. <http://www.v-modell.iabg.de/>. Last visited: June 2006.
- [3] S. W. Ambler. Agile requirements modeling. <http://www.agilemodeling.com/essays/agilerequirements.htm>. Last visited: June 2006.
- [4] H. Beyer and K. Holtzblatt. *Contextual Design: Defining Customer-Centered Systems*. Morgan Kaufmann, San Francisco, CA, 1997.
- [5] Borland Software Corporation. *Borland Caliber DefineIT 2006: User Guide*, April 2006.
- [6] B. S. Corporation. Borland Caliber DefineIT. <http://www.borland.com/>. Last visited: June 2006.
- [7] J. Dick and J. Chard. The systems engineering sandwich: Combining requirements, models and design. White paper, Telelogic, February 2004.
- [8] D. J. Mayhew. *The Usability Engineering Lifecycle: A Practitioner's Handbook for User Interface Design*. Morgan Kaufmann, San Francisco, CA, 1999.
- [9] E.-M. Melchior. User needs analysis: A precondition for validation planning. In *Inform Workshop*, Athens, Greece, 2003. [online, last visited: June 2006] www.vnet5.org/pub/inform-workshop-athens.html.
- [10] J. Nielsen. *Usability Engineering*. Morgan Kaufmann, San Francisco, CA, 1993.
- [11] M. Pichler, H. Rumetshofer, and W. Wahler. Agile requirements engineering for a social insurance for occupational risks organization: A case study. In *Proc. 14th IEEE International Conference on Requirements Engineering*, 2006.
- [12] S. Robertson and J. Robertson. *Requirements-Led Project Management: Discovering David's Slingshot*. Addison-Wesley, Pearson Education, Boston, 2005.
- [13] H. Solheim, F. Lillehagen, S. A. Petersen, H. Jorgensen, and M. Anastasiou. Model-driven visual requirements engineering. In *Proc. 13th IEEE International Conference on Requirements Engineering*, 2005.