Requirements Elicitation Using a Combination of Prototypes and Scenarios

Markus Mannio, Uolevi Nikula
Department of Information Technology, Lappeenranta University of Technology, Lappeenranta, Finland
{Markus.Mannio|Uolevi.Nikula}@lut.fi

Abstract. A survey of different types of prototypes and current prototyping methods is presented. Especially the usage of prototypes with use cases and scenarios is investigated, and an approach to requirements engineering combining the systematic use of prototyping and use cases is described. The approach consists of repeated prototype demonstration sessions with stakeholders in which use cases represented by multimedia prototypes are demonstrated. The prototypes are presented in a context familiar to the stakeholders and the use cases and prototypes are refined based on stakeholder feedback for subsequent iterations of prototyping. The purpose of the method is to aid in eliciting requirements from a diverse group of stakeholders. A small case study demonstrating the approach is described where the method is used to elicit the initial requirements for a mobile payment system for car parking fees. The results of the study are discussed and recommendations for future work are presented.

Keywords: requirements engineering, requirements elicitation, prototyping, use cases, scenarios

1. Introduction

A modern software engineering project involves a diverse group of different stakeholders. The needs and expectations of the stakeholders in addition to, for example, laws and industry standards form the basis for the requirements of a software system. To successfully complete a software engineering project an agreement has to be reached at the early stages of the project on what capabilities must be delivered and what else is required of the system [1].

One of the most difficult problems in deciding on the system requirements is the communication gap between different stakeholders. The system analysts tend to speak with technical terms while the customer and end-users prefer to use the language used in their daily business. As a result the analysts may not have a clear idea what kind of system they are building and the customer and end-users are often surprised when they notice that the final product does not fulfil their expectations and solve the problems they needed the system for in the first place. A well-known solution for narrowing the

communication gap is the usage of prototypes, which present something concrete the stakeholders can react to [2]. Another, more recent solution for improving the quality of communication in requirements engineering is the introduction of use cases and scenarios, which allow the different stakeholders describe and review the problem in their own language instead of some abstract model [3].
Use cases and scenarios are common in requirements engineering and various methods for their use exist (e.g. [4,5,6]) and a number of methods for the usage of prototypes also exist (e.g. [7,8,9]). However, even though use cases and prototypes are frequently used together in software development, practical approaches combining requirements specification, scenarios, prototypes, and evolutionary system development do not exist [3]. To make the end user participation in requirements development more natural we wanted to have an approach that helps the user to image herself in the real use context. Thus we developed a systematic method combining prototypes, use cases, and scenarios for specifying initial stakeholder requirements by representing textual use cases with multimedia prototypes of the new system.
In Section 2 survey of different types of prototypes, prototyping methods, and their usage with use cases and scenarios in requirements engineering is presented. Section 3 describes the proposed method itself and Section 4 presents a small case study demonstrating it. Finally Section 5 includes the conclusions and proposals for future work based on the case study results.

2. State-of-the-art study

In the following an overview of different types of prototypes, prototyping methods, and use cases and scenarios is presented.

2.1 Classification of prototypes

Different types of prototypes for different purposes in software engineering exist. Prototypes can be categorised, for example, as throwaway versus evolutionary, horizontal versus vertical, architectural versus requirements, textual versus visual, and executable versus non-executable prototypes [7,8,10,11].
Architectural prototypes are concerned with the performance and the feasibility of the technology used, whereas requirements prototypes are suggested to be used in requirements acquisition and user interface design [8]. Wiegers [11] presents horizontal prototypes as behavioral prototypes that do not necessarily contain any real functionality, whereas a vertical prototype implements a specific part of system functionality in a quality manner. The goal of a horizontal prototype is refining unclear requirements while a vertical prototype is used, for example, in algorithm optimisation. According to Kotonya and Sommerville [10], throwaway prototypes are discarded when the final system has been developed, whereas evolutionary prototypes are made available to users early in the development process and then extended to produce the final system. Sommerville and Sawyer [9] describe executable prototypes as software constructed using a high level programming language and non-executable prototypes as paper prototypes and other mock-ups of the system.

Visual prototyping methods include such techniques as diagrams, state charts, storyboards, and scenarios, whereas textual methods consist of formal prototyping languages [7]. Sutcliffe [12], and McGraw and Harbison [13] propose that the use of prototypes, mock-ups, examples, scenes, and narrative descriptions of contexts could all be called scenario-based approaches.

2.2 Use cases and scenarios

Many different definitions for use cases and scenarios exist. Constantine and Lockwood [14] define use cases as a collection of user actions and system responses. Similarly, Leffingwell and Widrig [8] describe use cases as interactions between the user and the system. Wiegers [11] gives a more general definition presenting use cases as descriptions of the tasks the system is required to accomplish. Although use cases were introduced as a part of object oriented approach, in practice their use is not limited to object oriented methods [11,14].

Scenarios are defined as narrative descriptions or stories in a specific context bound in time [14] or as specific instances containing descriptions of the environment, the context, the actors, and the actions with definite beginning and end points [13]. They are also presented as specific instances of use cases where a scenario describes a path of actions through a use case [5].

Thus, both scenarios and use cases describe the interactions between the system and the users without considering the internal structure of the system. Both describe the user activities and system responses while scenarios describe the context and the environment of the events in more detail. Use cases can be presented with different levels of abstractions [14], and use cases themselves can be thought as abstract scenarios.

Use cases and scenarios can be used throughout the whole software system development process from requirements elicitation to testing. In requirements elicitation and analysis use cases can be used in an iterative fashion. First, the actors and their high-level use case descriptions are acquired based on interviews and application domain knowledge. After that the use cases are refined by gradually adding more detailed scenarios and descriptions. The refined use cases are then analysed and the essential use cases are chosen for implementation [5,6]. Leffingwell and Widrig [8] suggest use cases to be used mainly for gathering functional requirements, but Kulak and Guiney [5] propose also methods for non-functional requirements gathering with the aid of use cases.

2.3 Prototype usage and methods

Asur and Hufnagel [7] suggest prototypes to be used for example in finding and specifying requirements, studying feasibility of development strategies, defining user interfaces, helping in the communication between stakeholders and increasing their participation in the system development process, and visualising future applications. Non-executable and visual prototypes are considered as cheap, fast, and low-risk techniques of prototyping that should be used when hypothetical system behaviour is to be displayed or the problem is not well understood [7,9,11].

Sommerville and Sawyer [9] and Wiegers [11] suggest non-executable prototypes to be used together with narrative evaluation scenarios to elicit requirements from stakeholders. Scenarios are proposed to be used in interaction sessions between the user and the system represented by a prototype. Kotonya and Sommerville [10] also discuss a 'Wizard of Oz' prototyping technique where a person simulates the responses of the system in response to some user inputs and thus acts as the prototype. Leffingwell and Widrig [8] discuss a storyboarding requirements elicitation technique where a storyboard is an early representation of the system. Three different classes of storyboards are presented: passive, active, and interactive.
Asur and Hufangel [7] present a general process for prototyping, which constitutes of the following steps:

1. Preliminary analysis of users' requirements.
2. Prototype design.
3. Rapid prototype implementation.
4. Prototype evaluation with different stakeholders.
5. Iterative refinement of the prototype.
6. Refinement of requirements and specifications.
7. Design and implementation of the production system.

Sutcliffe [15] and Sutcliffe and Ryan [16] present a scenario-based method that uses early prototypes in combination with scenarios. A prototype is built based on initial requirements elicited by conventional requirements elicitation techniques. The prototype is then presented as a script in a requirements analysis-validation session with the users. Different options are visualised with design rationales and probe questions are asked at key points of the prototype presentation. The sessions are recorded and the data collected is analysed afterwards, and the prototype is revised based on user feedback. In the presented example cases the method was effective in gathering functional and usability requirements. The noticeable differences between different analysts, and the bias towards the option implemented in the prototype were some of the problems discovered.
Studies in the commercial usage of prototypes [3,17] show that prototypes are widely used in software development together with other methods. The main advantages of prototyping perceived by software developers were better user involvement and improved communication with the users. The biggest disadvantages were reduced management control of project, false user expectations, and time required for user participation [2,17].
A study of scenario usage in industrial projects [3] shows that scenarios are used for diverse purposes in software development. According to the study scenarios are used mostly in making abstract models concrete, interdisciplinary development, and facilitating agreement between different stakeholders. Scenarios presented in text, images, diagrammatic notations, or animations and simulations were used with prototypes of the new system in two-thirds of the projects studied. It was noted, however, that no practical approach exists in combining requirements specification, scenarios, prototypes, and evolutionary system development [3].

3. Method

A method using a combination of prototypes, use cases, and scenarios for eliciting initial software requirements is described. The process is based on the general prototyping process presented by Asur and Hufnagel [7] and the scenario-based method described by Sutcliffe [15]. It combines elements from the use case approach first presented by Jacobson et al. [4] and further developed by, for example, Schneider and Winters [6] with prototyping and scenarios [18].

In the method presented the use cases begin as abstract high-level descriptions of the tasks the system needs to accomplish for each user class and gradually move towards more concrete descriptions of how these tasks could be accomplished. Thus, the first versions of the use cases resemble the "essential use cases" presented by Constantine and Lockwood [14] evolving towards the more concrete "filled use cases" presented by Kulak and Guiney [5]. Scenarios are used as more concrete instances of use cases presented in user language within the context of use. The use cases and scenarios are visualised with the prototypes in demonstration sessions with system stakeholders in order to elicit software requirements.

The method is an iterating process in which the number of iterations through the phases depends on the objectives of the process, the size and complexity of the system, and the level of detail needed for the requirements. The outputs of the whole process are the detailed use case descriptions with scenarios, other requirements, and the prototype versions.

The steps of the process are shown in Figure 1 and explained in the following subsections.

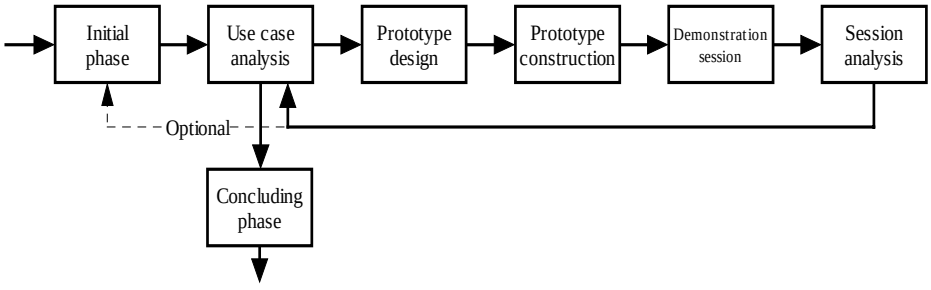

Figure 1. The prototyping process.

3.1 Initial phase

The main purpose of first phase in the process is acquiring the basic understanding of the system and familiarising the system analysts to the problem domain. This phase roughly corresponds to the first steps of the processes and techniques presented by Sutcliffe [15], and Schneider and Winters [6].

The system description, context, and initial boundaries are identified with traditional requirements elicitation techniques such as structured and unstructured interviews, and brainstorming sessions with the system stakeholders. Existing documentation and

developer domain knowledge are also studied. Different user classes of the system are identified and the primary tasks that each class needs to accomplish are documented to use case templates [6].
The templates for recording use cases and scenarios can be based, for example, on the ones presented by Kulak and Guiney [5]. At least the name of the use case, date, a short description, and user classes associated to the task are recorded at this stage. For traceability reasons a unique identifier, the source of the use case, and change history can also be documented [19,20]. The use case description is presented at a high level of abstraction to describe only the task, not the implementation. General system requirements that are not connected to any specific use cases are recorded in a separate document. Also, the goals of the prototyping process are recorded to aid in evaluating the progress and guiding the process.
This initial phase needs to be executed only in the beginning of the process. However, it can be executed also on subsequent rounds through the process if major changes occur in the system definition.

3.2 Use case analysis

In the use case analysis phase the decision is made about continuing the prototyping process. The use cases, other documentation, and system analyst knowledge are combined in document inspections to decide if the goals set in the initial phase of the prototyping process are achieved and whether these goals are reasonably achievable. If the goals are achieved or they are deemed unreasonable to prototype, the process iterations are aborted and the concluding phase is executed.
If the prototyping process is continued, the use cases to prototype have to be selected. The previously recorded use cases are analysed and prioritised according to the their significance so that critical, poorly understood, ambiguous, or conflicting use cases should have high priority and therefore they should be likely candidates for prototyping. Prototyping well-understood requirements is unnecessary and should not be done [8].

3.3 Prototype design

One prototype is designed for each use case chosen for prototyping. A context, which is a description of the prototype environment presented in the user's language, is devised for the prototype. User classes are presented as specific people and storylines are generated to justify and illustrate user actions. Scenarios are generated based on the prototype context and can be documented as scenario descriptions.
Prototype structure is developed by first designing a prototype skeleton where every use case event is represented by at least one prototype screen. The purpose of the skeleton is to present the use case and the problem; proposing any solutions to open questions should be avoided. The solutions and design options for the use case are presented in the scenarios. Multiple alternative scenarios may be directly implemented in the prototype, and separate documents such as design rationales [15,16] are not needed. The starting and end points of each scenario are determined in the prototype

skeleton, so that each scenario can focus on a specific problem area and the common, overlapping parts in scenarios need not to be designed multiple times. Also, the relationships between the scenarios themselves are determined; a scenario selected at any given point of the prototype skeleton may propose further refined alternatives for solutions. An example of a tree-like scenario structure is presented in Figure 2, which also illustrates the difference between the use case events and scenarios.

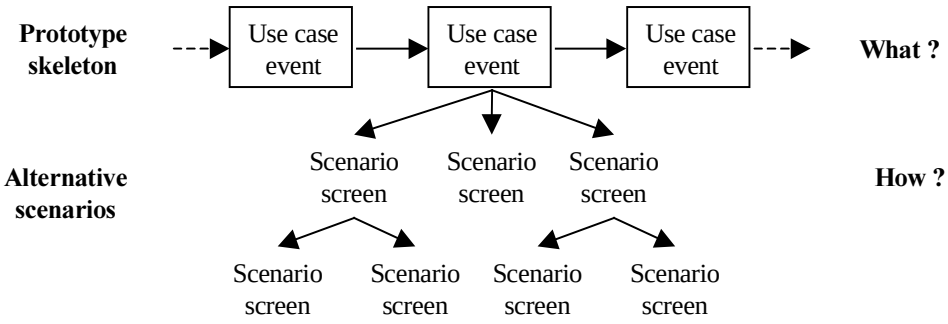

Figure 2. An example scenario structure.

Finally, screens are designed for each scenario and the skeleton of the prototype by describing what kind of multimedia objects each screen should contain.

3.4 Prototype construction

In construction phase the designed prototype structure and screens are implemented. The prototypes are constructed from multimedia objects, which may consist of pictures, animations, sounds, or video clips. These objects may be combined to form more complex objects that may also contain user interactivity. The primary requirements for the prototyping software are the ability to display multiple multimedia objects on the screen, ability to interact with the objects, and the ability to link the screens together so that an alternative scenario can be selected from any screen of the prototype. Each prototype is named according the use case it represents and given a unique version number to allow requirements traceability to a specific prototype version.

First step in implementing a scenario is to construct the screens by searching suitable multimedia objects from an object database. If suitable objects do not exist, they are constructed, documented, classified, and inserted to the object database for future use. Objects can be modified on screen to fit better in the context of the prototype and scenario. Second, the screens are linked together to form the designed structure of the prototype as shown in the example in Figure 2. Finally, the prototype is tested by traversing the screens and links of the prototype, and by checking the functionality of possible interactive objects.

3.5 Prototype demonstration session

The prototype is used in a demonstration session with the system stakeholders to record stakeholder views and elicit requirements. The prototype is used to invoke reactions from the stakeholders and create discussion about the proposed system and design options. The session set-up consists of the computer running the prototype, a system analyst operating the prototype, a small group of system stakeholders, and a projector displaying the prototype. The session set-up is similar to the one presented by Sutcliffe [15]. Optionally, the session audio or video can be recorded, but this may present some problems, such as stakeholder distraction [13]. If the prototype contains interactive objects, recording the prototype screen display can also be used.

The course of the session is adapted from the principles described by McGraw and Harbison [13]. The session begins with an introduction, where the system analyst introduces himself, states the purpose of the session, explains the context of the prototype and a short description of the use case. Then the system analyst runs the prototype through the skeleton of the use case events and concurrently recites the storyline of the use case, thus presenting the use case and the problem. The main purpose of this first round through the prototype is to give an overview of the use case and the context.

The prototype is then restarted from the beginning and the system analyst tries to stimulate discussion about each screen by asking questions and presenting the scenarios as possible solutions. At each screen stakeholder suggestions, comments and ideas are recorded in the prototype or on paper. If multiple optional scenarios are discussed for a use case event, a preferred order of the scenarios should be decided and recorded. General requirements that are not directly linked to the use case at hand are recorded in the other requirements document. The system analyst concludes the session by presenting a summary of the use case and the alternative scenarios discussed and the stakeholder preferences documented during the prototype presentation.

Multiple prototypes and use cases may be presented in a session; preferably so that the use cases are interconnected and have the same context to ensure continuity of the storyline. That is, the post conditions of the previous prototype fulfil the preconditions of the following prototype.

3.6 Session analysis

Each round of the process in concluded by a session analysis phase, where the prototyping session recordings are analysed. The session recordings consist of the textual information recorded in the prototype screens and paper, the other requirements document, and the possible audio, video, and screen captures.

Recorded text that is connected to the use case events skeleton should propose new use cases, changes to the use case events or new scenarios. On the other hand, recorded text connected to a specific scenario screen describes the suitability of the solution presented by the scenario to the problem presented by the use case. The use case events should be filled and made more concrete by modifying the use case description based on the most preferred related scenario.

Thus, the use cases become more detailed in each round, as it is also proposed in the processes presented by Kulak and Guiney [5], and Schneider and Winters [6]. After the prototype version is documented on the use case and scenario templates, and the prototype version and possible other recordings are stored, the prototype may be used to serve as the basis for modifications for the next round of the process iteration beginning from the use case analysis phase.

3.7 Concluding phase

This is the final phase in the process, which is executed after the decision about stopping the iterations through the prototyping process is made. The purpose of the phase is to combine all essential documentation generated during the process to a single document to be available for the following stages of the system development. This document should contain at least a description of the system and the problem prototyped, and an explanation of the goals of the prototyping process and their fulfilment. Also, the latest versions of the use case descriptions and the other requirements document should be attached. In addition, the constructed prototypes and session recordings should be stored and made available for future reference.

3.8 Summary of the proposed method

The proposed method is summarised in the following Figure 3. Each phase is described by its inputs and outputs and tasks. The inputs with bold letters indicate new inputs to the process; other inputs are outputs from previous phases.

4. Case study

The described method was experimented using a small fictive test case in which the first round through the process was executed. As the subject for the example case a system handling the mobile payment of car parking fees was chosen. The phases of the process executed during the case are described below in more detail.

In the initial phase the main idea is to familiarise the system analysts to the problem. In this case the methods used for acquiring the basic understanding of the problem were two stakeholder interview sessions and studying topic related literature. The goal of the prototyping was set to be the elicitation of the initial system requirements. Based on the interviews three top-level user classes of the system were identified: the vehicle driver, traffic warden, and the system administrator. Also, the most important high-level tasks for each of the main user classes were uncovered.

In the use case analysis phase it was determined that the objectives could be achieved through prototyping and thus the process was continued. The information gathered during the initial phase was analysed and three essential high-level use cases were chosen for prototyping. The chosen use cases were the car driver parking a car, the car driver ending parking, and the traffic warden checking the parking area.

Inputs	Phase	Outputs
Domain knowledge **Stakeholder vision** **Documents**	*Initial phase*	Prototyping goals System context User classes Use case descriptions Scenarios Other requirements
Prototyping goals System context User classes Use case descriptions Other requirements **Analyst knowledge**	*Use case analysis*	Go / no-go decision Description of the selected use case
Description of the selected use case System context	*Prototype design* *1. Design context* *2. Select scenarios* *3. Design prototype structure* *4. Design screens*	Prototype context Scenarios to implement Screen descriptions Scenario storylines Prototype structure
Prototype context Description of the selected use case Scenarios to implement Screen descriptions **Possible previous version of prototype** **Multimedia objects**	*Prototype construction* *1. Construct screens* *2. Link screens* *3. Test prototype*	Use case prototype New multimedia objects
Use case prototype Scenario storylines **Stakeholders views** **Facilitator skill**	*Prototype demonstration session* *1. Introduction* *2. Prototype presentation* *3. Conclusion*	Session recordings
Session recordings	*Session analysis*	New use cases More detailed use cases New scenarios Other requirements
Prototyping goals System context Use case descriptions Other requirements	*Concluding phase*	Process report

Figure 3. The phases of the prototyping process.

The prototype design phase was executed for each of the three use cases chosen for prototyping. A common context was devised for all the prototypes forming a continuous story from the beginning of the first use case to the end of the last. Scenarios were generated and corresponding screens designed for each use case to present possible solutions to the use case events. The screens contained mainly images but also some

animations were used. Finally, the structure linking the scenarios to a use case was designed.

In the prototype construction phase the designs of the previous step were implemented. In the lack of any domain specific multimedia object database or previous prototype versions the primary source for the screens was Microsoft Digital Gallery [21]. The tool used to construct the prototypes was Microsoft PowerPoint, which loosely fulfils all the requirements set for the prototyping tool. After the prototype screens were constructed and assembled to a PowerPoint presentation, the prototypes were tested by executing all the scenarios by displaying all their respective screens in correct order

The prototypes were demonstrated in one session with two customer representatives and a system analyst. One of the customer participants had extensive technical knowledge on the example case, while the other had a business-oriented background. The purpose of the session was to increase the understanding between the customer and the system analyst about the system properties. The session set-up consisted of a laptop computer connected to a projector operated by the system analyst. The session was begun by a short introduction from the system analyst describing the method and the purpose for the session. The participants were encouraged to express their ideas and opinions at any time during the prototype demonstration. The participants were especially encouraged to think what the main actor in the use case would do in the situation presented by use case and the scenarios.

The three prototypes had the same context and they formed a continuous story. First, the car driver parked his car to attend a lunch meeting, and while he was having his lunch a traffic warden checked the area where the driver's car was parked. Finally, the car driver left the restaurant to get back to his car and drove back to his office. At the beginning of each prototype the analyst presented the use case and recited the storyline. Then the individual scenarios were discussed and the analyst recorded the discussions about the use case event or scenario in question on paper. The session was concluded with the analyst stepping through the prototype and explaining the preferred solution for each event.

After the demonstration session the written recordings were analysed. Unambiguous customer preferences were obtained to most of the open questions presented by the use cases. Details were added to the use cases based on the preferred scenarios of the session participants. For example, in the parking use case the car driver should use a mobile phone to call to a number to signal the beginning of parking. In some cases the customers considered multiple options possible. These resulted in alternative paths in the use cases, which could possibly be converted to totally new use cases. Completely new use cases, such as: "Vehicle driver reserving parking space beforehand" or "Vehicle driver registering to the parking service" were also discovered.

A summary of the time used by the system analyst for different phases in the case study is presented in Table 1. The initial phase of the process took 17 man-hours and other phases took a total of 40 man-hours. The time used on the design and implementation would probably have been notably reduced if previous versions of prototypes and domain specific multimedia object database had existed.

Phase	Time (hours)
Initial phase	17
Use case analysis	5
Design of prototypes	13
Prototype construction	14
Demonstration session	2
Session analysis	6
Total	**57**

Table 1. Summary of the time usage in the case study.

5. Discussion and conclusions

The method presented was successful in eliciting requirements from the customers in the example case. An initial reaction was acquired from the customers about the different design options and in most open questions a preference was also established between different options. In addition to the design options presented in the prototypes, the session participants also presented other solutions to the questions, which was probably due to the good domain knowledge of the customers. The main advantage of the method was the increased mutual understanding developed during the prototype demonstration session. Both the session participants and the system analyst considered the session more focused and systematic than a session without the prototypes would have been. Presenting the use cases and design options with the prototypes helped the participants to concentrate on specific problems. Also, a better general view of the system and the different design options was achieved.

Since the open questions and their possible solutions were separated in the prototypes and the multiple design options were presented in the prototype itself, there was no need for separate documents illustrating the different options. As a consequence some of the possible disadvantages presented by Sutcliffe [15] and Sutcliffe and Ryan [16], such as the difficulties in stakeholder understanding the options presented in diagrams or stakeholder bias towards some solutions, were not so apparent in the example case. One perceived problem was the lack of visualisation of the design options that were presented by the session participants and had not been implemented in the prototypes. These options were discussed but a better understanding would probably have been achieved if the options could have been visualised somehow. Another problem was the structural complexity of some of the prototypes presenting a lot of different scenarios, which might cause stakeholder confusion in some cases. This could probably be avoided by better planning and structuring of the use cases. Also, as suggested by Sutcliffe [15], the system analyst style may have a considerable impact on the number and quality of the requirements elicited.

Methods suggested by Schneider and Winters [6], and Kulak and Guiney [5] present the use case approach, but offer little guidance how use cases could be combined with the use of prototypes in practice. Sutcliffe [15], and Sutcliffe and Ryan [16] present a method combining scenarios with prototypes without including all the design options in the prototype itself. In the method presented in this paper the design options are

visualised inside the multimedia prototype itself to facilitate the communication with the stakeholders. Also, the scenarios are presented in a use case context, thus making it possible to incorporate the method in an existing use case based software process.
Naturally a realistic assessment of a requirements elicitation method in software engineering is impossible unless a complete project is executed and evaluated. The quality and sufficiency of the captured requirements is difficult to determine based only on an incomplete example case. Although a substantial number of new requirements were gathered in this case, the acquired information still represents the view of only one group of stakeholders on some aspects of the future system.
The developed method focuses currently only to combine prototyping and scenario approaches to improve the customer understanding and participation to the requirements elicitation in software development. Since issues like management and selection of use cases, prototypes, scenarios, and traceability are addressed in other literature we chose to ignore them in this phase of the method development (see e.g. [5,19,20]). Of course, before the method is ready for industrial application these questions must be studied closer and approaches to them must be integrated in the method.
In the future further studies are needed to assess the real value of the proposed method for requirements elicitation. In addition to a more extensive case study a more comprehensive comparison of the method with existing methods should be conducted. The most urgent development needs in the method itself are the ways to document session discussions and presenting new and modified scenarios during the session.
For example being able to draw sketches of the proposed scenarios during the session could reduce the need for follow-up sessions since new scenarios could be validated immediately. Despite the shortcomings of the current method both the initial literature and practical studies suggest that there is still space for requirements elicitation methods that provide the user a more realistic view on the planned software system.

Acknowledgements

This work was carried out in the Telecom Business Research Center (TBRC) of Lappeenranta University of Technology in co-operation with Sonera Service Software. The authors would like to thank TBRC and Sonera for making this work possible and especially Laura Nihti and Kari Leino from Sonera for their valuable advice and support. The project responsible director Prof. Heikki Kälviäinen and academic instructor Prof. Jan Voracek are also acknowledged for their support for this work.

References

[1] Hruschka, P., "Detailing and deriving system requirements", *Proceedings from the International Conference and Workshop on Engineering of Computer-Based Systems*, March 1997, pp. 25-32.
[2] Kimmond, R.M., **"**Survey into the acceptance of prototyping in software development", *Proceedings from the IEEE Sixth International Workshop on Rapid System Prototyping*, June 1995, pp. 147-152.

[3] Weidenhaupt, K., Pohl, K., Jarke, M., Haumer, P., "Scenarios in system development: current practice", *IEEE Software*, IEEE, 15(2) 1998, pp. 34-45.
[4] Jacobson, I., Christenson M., Jonsson P., Overgaard, G., *Object-oriented software engineering: a use case driven approach*, ACM Press, 1992.
[5] Kulak, D., Guiney E., *Use cases – requirements in context*, ACM Press, 2000.
[6] Schneider, G., Winters, J.P., *Applying use cases: a practical guide*, Addison Wesley Longman Inc., 1998.
[7] Asur, S., Hufnagel, S., "Taxonomy of rapid-prototyping methods and tools", *Proceedings from the IEEE Fourth International Workshop on Rapid System Prototyping*, June 1993, pp. 42-56.
[8] Leffingwell, D., Widrig, D., *Managing software requirements: A unified approach*, Addison-Wesley, 2000.
[9] Sommerville, I., Sawyer, P., *Requirements engineering: A good practise guide*, John Wiley & Sons Inc., 1997.
[10] Kotonya, G., Sommerville, I., *Requirements engineering: processes and techniques*, John Wiley & Sons Inc., 1997.
[11] Wiegers, K. E., *Software requirements*, Microsoft Press, 1999.
[12] Sutcliffe, A., "Workshop exploring scenarios in requirements engineering", *Proceedings from the IEEE Third International Symposium on Requirements Engineering*, Jan. 1997, pp. 180-182.
[13] McGraw, K., Harbison K., User-centred requirements: *The scenario-based engineering process*, Lawrence Erlbaum Associates, Inc., 1997.
[14] Constantine, L.L., Lockwood, L.A.D., *Software for use: a practical guide to the models and methods of usage-centered design*, Addison Wesley Longman Inc., 1999.
[15] Sutcliffe, A., "A technique combination approach to requirements engineering", *Proceedings from the IEEE Third International Symposium on Requirements Engineering*, Jan. 1997, pp. 65-74.
[16] Sutcliffe, A., Ryan, M., "Experience with SCRAM: a scenario requirements analysis method", *Proceedings from the IEEE Third International Conference on Requirements Engineering*, April 1998, pp. 164-171.
[17] Khalifa, M., Verner, J.M., "Drivers for software development method usage", *IEEE Transactions on Engineering Management*, IEEE, 47(3) 2000, pp. 360-369.
[18] Chance, B.D., Melhart, B.E., "A taxonomy for scenario use in requirements elicitation and analysis of software systems", *Proceedings from the IEEE Conference and Workshop on Engineering of Computer-Based Systems ECBS '99*, March 1999, pp. 232–238.
[19] Gotel, O.C.Z., Finkelstein A.C.W., "An Analysis of the Requirements Traceability Problem", *Proceedings of the First International Conference on Requirements Engineering*, April 1994, pp. 94-101.
[20] Robertson, S., Robertson J., *Mastering the Requirements Process*, Addison-Wesley, 1999.
[21] Microsoft Digital Gallery, http://dgl.microsoft.com, Microsoft Corporation, 2001.