Telecom Business Research Center Lappeenranta
Research Report 5

Markus Mannio, Uolevi Nikula

Requirements Elicitation Using a Combination of Prototypes and Scenarios

Telecom Business Research Center Lappeenranta
Lappeenranta University of Technology
P.O.BOX 20, FIN-53851 LAPPEENRANTA, FINLAND
http://www.lut.fi/TBRC/

Lappeenranta 2001

ISBN 951-764-527-9
ISSN 1456-9140

ISBN 951-764-528-7 (URL: http://www.lut.fi/TBRC/)

Requirements Elicitation Using a Combination of Prototypes and Scenarios

Markus Mannio, Uolevi Nikula
Department of Information Technology, Lappeenranta University of Technology, P.O. Box 20, FIN-53851 Lappeenranta, Finland,
{Markus.Mannio|Uolevi.Nikula}@lut.fi

ABSTRACT

A survey of different types of prototypes and current prototyping methods and tools 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 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 capturing quality requirements from a diverse group of stakeholders. A small case study demonstrating the approach is described. The method is used to capture the initial requirements for a system of mobile payment of 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 people with different backgrounds. A typical project includes at least the system developers and their management, the customer, and the end-users that often are a completely separate group from the customer. These people that are affected by the system and place different expectations for the system are called system stakeholders (Sommerville & Sawyer 1997). The needs and expectations of the different 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 (Hruschka 1997).

One of the most difficult problems in deciding on the system requirements is the fact that the software developer representatives, the system analysts, and other stakeholders often do not really understand each other. The analysts tend to speak with technical terms and 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 between the system analysts and other stakeholders is the usage of prototypes, which present something concrete the stakeholders can react to (Kimmond 1995). 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 the problem in their own language instead of some abstract model (Weidenhaupt et al. 1998).

Use cases and scenarios are common in requirements engineering and various methods for their use exist (e.g. Jacobson et al. 1992; Kulak & Guiney 2000; Schneider & Winters 1998) and a number of specific methods for the usage of prototypes also exist (e.g. Asur & Hufnagel 1993; Leffingwell & Widrig 2000; Sommerville & Sawyer 1997). However, methods describing the systematic usage of prototypes in combination with use cases in requirements elicitation are not common, even though use cases and prototypes are frequently used together in software development (Weidenhaupt et al. 1998). In this paper a survey of different types of prototypes, prototyping tools and methods, and their usage with use cases and scenarios in requirements engineering is presented. Also, a method combining the use of prototypes, use cases, and scenarios is described for capturing initial stakeholder requirements. Finally, a small case study demonstrating the method is described and analysed.

2. STATE-OF-THE-ART STUDY

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

2.1. Classification of Prototypes

Different types of prototypes for different purposes in software engineering exist. Leffingwell and Widrig (2000) present a classification tree for prototype selection that categorises prototypes according to the purpose the prototype is needed for. Prototypes are categorised as throwaway versus evolutionary, horizontal versus vertical, and architectural versus requirements prototypes. Asur and Hufnagel (1993) divide prototypes according their representation into textual and visual prototypes, and define rapid prototyping as the use of tools or environments for quick prototype construction. A division into executable and non-executable prototypes can also be made (Kotonya & Sommerville 1997; Wiegers 1999).

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 (Leffingwell & Widrig 2000). Wiegers (1999) presents horizontal prototypes as behavioural 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 (1997), 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 (1997) 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 entity-relationship –diagrams, state charts, storyboards, and scenarios, whereas textual methods consist of programming and formal languages. Storyboards are considered as visual prototyping tools and defined as interactive screen displays of system behaviour, which can be used as simulation tools for depicting man-machine interaction and mimicking applications (Asur & Hufnagel 1993).

Thus, a prototype can vary from a simple paper mock-up to a fully functional executable system. A classification of different types of prototypes is presented in Table 1. Sutcliffe (1997a) and McGraw and Harbison (1997) propose that the use of prototypes, mock-ups, examples, scenes, and narrative descriptions of contexts could all be called scenario-based approaches.

***Table 1**. Classification of prototypes.*

System development model	
Evolutionary ▪ Incremental development	**Throw-away** ▪ Specific usage
Operation	
Executable ▪ High-risk ▪ Complex	**Non-executable** ▪ Low-risk ▪ Simple
Representation	
Textual ▪ Formal languages	**Visual** ▪ Mock-ups ▪ Storyboards
Level of detail	
Vertical ▪ High detail ▪ Non-functional requirements	**Horizontal** ▪ Low detail ▪ Functional requirements
Objective	
Architectural ▪ Technical feasibility	**Requirements** ▪ Requirements elicitation

2.2. Use Cases and Scenarios

Many different definitions for use cases and scenarios exist. Constantine and Lockwood (1999) define use cases as a collection of user actions and system responses. Similarly, Leffingwell and Widrig (2000) describe use cases as interactions between the user and the system. Wiegers (1999) 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 (Wiegers; Constantine & Lockwood).

Scenarios are defined as narrative descriptions or stories in a specific context bound in time (Constantine & Lockwood 1999). A more specific definition is offered by McGraw and Harbison (1997) who define scenarios as specific instances containing descriptions of the environment, the context, the actors, and the actions with definite beginning and end points. Scenarios are also presented as specific instances of use cases, where a scenario describes a path of actions through a use case (Kulak & Guiney 2000).

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 also describe the context and the environment of the events in more detail. Use cases can be presented with different levels of abstractions (Constantine & Lockwood 1999) and use cases themselves can be thought as abstract scenarios.

While informal ad hoc processes for scenario generation and use are seen as the most prevalent, also systematic methods for scenario generation exist (Hsia et al. 1994). A more formal approach for use case representation can be achieved by using a notation such as Unified Modeling Language (Rational software corporation 2001), where use cases are presented as diagrams. In UML system users and other external entities are represented by actors that participate in interactions with the system in use cases and scenarios (Kulak & Guiney 2000).

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 (Kulak & Guiney 2000; Schneider & Winters 1998). Leffingwell and Widrig (2000) suggest use cases to be used mainly for gathering functional requirements, but Kulak and Guiney propose also methods for non-functional requirements gathering with the aid of use cases.

2.3. Prototype Usage

Asur and Hufnagel (1993) 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 (Asur & Hufnagel; Sommerville & Sawyer 1997; Wiegers 1999). Kulak and Guiney (2000), who define prototypes as mock-ups of the screens and windows of an application, argue that prototypes represent only the front end of the system and should not be used in early requirements elicitation. Instead, they suggest that prototyping should be used later, in the user interface specification phase.

Studies in the commercial usage of prototypes show that prototypes are widely used in software development together with other methods. The main advantages of prototyping perceived by software developers in the studies were better user involvement, and the improvement of communication with the users (Kimmond 1995; Khalifa & Verner 2000). The biggest disadvantages were reduced management control of project, false user expectations, and time required for user participation. In most cases in commercial projects prototypes were put to some subsequent use after its development (Kimmond 1995).

A study of scenario usage in industrial projects 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 were presented in text, images, diagrammatic notations, and animations and simulations. In two-thirds of the projects studied, scenarios were used with prototypical implementations of the new system. It was noted, however, that no practical approach exists in combining requirements specification, scenarios, prototypes, and evolutionary system development (Weidenhaupt et al. 1998).

2.4. Techniques and Processes

Sommerville and Sawyer (1997) and Wiegers (1999) 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 (1997) also present 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 (2000) 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. In passive storyboards the analyst plays the role of the system and walks the user through the storyboard with an explanation. Active storyboards provide an automated description of the way the system behaves in a typical usage or operational scenario. Interactive storyboards give a more realistic picture of the system and require the participation of the user in order to execute.

Asur and Hufangel (1993) 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. Iterating refinement of the prototype.
6. Refinement of requirements and specifications.
7. Design and implementation of the production system.

Ghajar-Dowlatshahi and Vernekar (1994) propose a similar process with the addition of a usability testing phase with end-users to the iterative development of a prototype, and suggest prototype demonstrations and brainstorming sessions as the means for prototype evaluation.

Sutcliffe (1997b) and Sutcliffe and Ryan (1998) present a scenario-based method (SCRAM) that uses early prototypes called concept demonstrators 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 where the users are encouraged to comment and critique it. 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 example cases presented 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.

2.5. Tools

There exist a number of commercial CASE tools for aiding in the process of requirements engineering. These tools focus mainly on requirements management: requirements tracing, classification, validation, configuration management, documentation, specification generation, etc. (McMullen 2000). Tools for business process modelling, engineering, and re-engineering also exist, which use diagrammatic notations to represent the system being modelled (Lamb 2000).

High-level programming languages for executable prototype construction, such as Visual Basic, are also common, and they provide the means for example rapid user interface generation. However, there is little guidance for choosing a tool for non-executable prototyping. Some proposed tools are multimedia authoring and presentation tools such as Apple Hypercard, Microsoft PowerPoint, and Macromedia Director (Sutcliffe 1997b; Constantine & Lockwood 1999:215). There exist also some virtual product demonstration tools, such as eSim's RapidPLUS, which are used to construct prototypes of human-machine interfaces in embedded systems. Some of the tools used in constructing different types of prototypes are presented in Table 2.

Table 2. *Tools for different types of prototypes.*

Prototype	Tools
Executable	High level programming languages (Visual Basic, Visual C++, Borland JBuilder/C++, PowerBuilder, etc.)
Non-executable	Paper sketches, post-its, transparencies Presentation tools (PowerPoint, Macromedia Director, etc.)
Textual	Formal specification languages (EPROL, LOOPS, etc.)
Visual	High level programming languages Presentation tools

3. METHOD

A method that uses a combination of prototypes, use cases, and scenarios for capturing initial software requirements is described here. The process is based on the general prototyping process presented by Asur and Hufnagel (1993) and the scenario-based method described by Sutcliffe (1997b). It combines elements from the use case approach first presented by Jacobson et al. (1992) and further developed by, for example, Schneider and Winters (1998) with prototyping and scenarios (Chance & Melhart 1999).

In the method presented here 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 (1999) evolving towards the more concrete "filled use cases" presented by Kulak and Guiney (2000). Scenarios are used as more concrete instances of use cases presented in user language within a context of use. The use cases and scenarios are visualised with the prototypes in demonstration sessions with system stakeholders where software requirements are elicited.

The process consists of seven phases containing a loop in which the process steps are executed in an iterative fashion: after finishing one round consisting of all the phases in the loop, the next round is begun from the beginning. The number of iterations through the phases depends of the objectives of the process, the size and complexity of the system, and of the level of detail needed of the requirements. The outputs of the whole process are the detailed use case descriptions with scenarios, other requirements, the prototype versions, and other recordings of the prototype demonstration sessions.

The steps of the process are shown in Figure 1 and described in more detail in Figure 2.

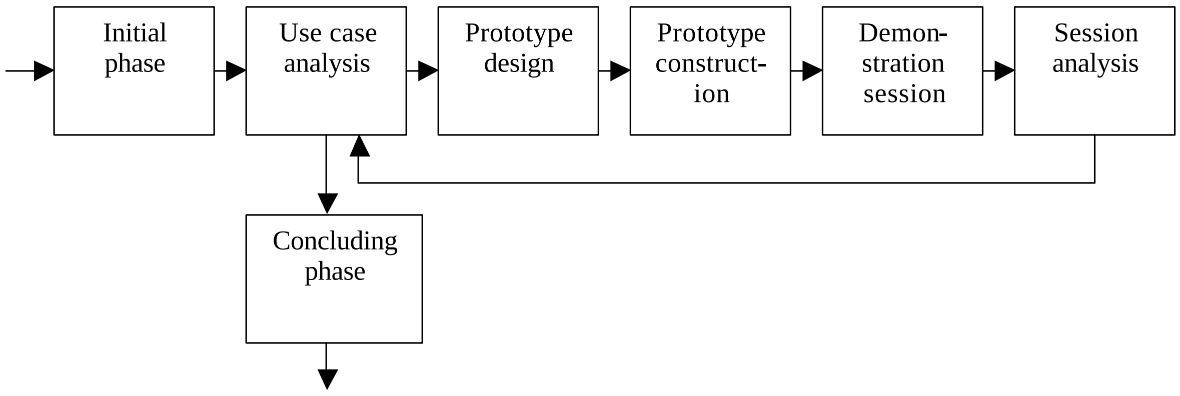


***Figure 1**. The prototyping process.*

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 2**. The phases of 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(1997b), and Schneider and Winters(1998).

The system description, context, and initial boundaries are identified by using 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 (Schneider & Winters 1997).

Examples of the templates for recording use cases and scenarios, which are modified versions of the templates presented by Kulak and Guiney (2000), are presented in Appendix 1. At least the name of the use case, date, a short description, and user classes associated to the task should be recorded at this stage. The use case description should be presented at a high level of abstraction to describe only the task, not the implementation; if some concrete scenarios are discussed, these can be recorded in the scenario templates. General system requirements that are not connected to any specific use cases should be recorded in a separate document. Also, the goals of the prototyping process should be 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 reviews or 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 selection of the use cases to prototype has to be made. The previously recorded use case templates are analysed and searched for conflicts or ambiguities. The use cases are prioritised according to the their importance and complexity so that critical, poorly understood, 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 (Leffingwell & Widrig 2000).

3.3. Prototype Design

One prototype is designed for each use case chosen for prototyping. A context, which should be 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 also documented to scenario templates.

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 (Sutcliffe 1997b; Sutcliffe & Ryan 1998) are not used. 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. Activity diagrams can be used to help illustrating the prototype structure, especially if the prototype contains loops or multiple levels of scenarios. 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 this 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 requirements for the prototype are the ability to display multiple multimedia objects in a screen, enable interaction with the objects, record textual information at each screen of the prototype, and the ability to link the screens together and allow the selection of an alternative scenarios at a given screen of the prototype to form a continuous path from scenario beginning to the end. It should be possible to traverse the links between screens in both directions.

Each prototype should be given the name of the use case it represents and a unique version identification to allow requirements traceability to a specific prototype version. The basis for the prototype construction can be a previous version of the use case prototype that can serve as a starting point for modifications. First, the screens are constructed by searching suitable objects from the 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 beginning from the prototype use case event skeleton. Finally, the prototype is tested by traversing each screen and link of the prototype, and by testing 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 capture software 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 developer representative 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(1997b); the main differences being the absence of the design rationales and the number of system analysts present. Optionally, the session audio or video can be recorded, but this may present some problems, such as stakeholder distraction (McGraw & Harbison 1997:83). 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 (1997) and Sutcliffe (1997b). The session begins with an introduction, where the system analyst introduces himself, states the purpose of the session, a short description of the use case, and the context of the prototype. 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. Even though the main purpose of this first round through the prototype is to give an overview of the use case and the context, any stakeholder comments can already be recorded in the relevant screens or written on paper. The comments that are recorded in the prototype are visible to the stakeholders through the projector, so that the stakeholders see what is recorded and can ensure that it corresponds to their meaning.

The prototype is started from the beginning and the system analyst tries to create discussion about each screen by asking questions and presenting the scenarios as possible solutions. At each screen stakeholder suggestions, comments and ideas are recorded. If multiple optional scenarios are discussed for a use case event, an order of preference of the scenarios should be achieved and recorded. General software 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 scenarios discussed with 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 postconditions of the previous prototype fulfil the preconditions of the following prototype. The overall length of a session, however, should not exceed two hours.

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, while recorded text connected to specific scenario screens 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 template 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 (2000), and Schneider and Winters (1998). User comments about the scenarios can be recorded in the scenario templates. Possible new use cases and scenarios should also be recorded in the use case templates for future use.

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. It is possible that not even a single prototype has been designed and constructed, if the goals set in the initial phase were deemed unreasonable to prototype in the first use case analysis phase. The purpose of this phase is to combine all essential documentation generated during the process to a single document to be available at 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.

4. CASE STUDY

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

4.1. Initial Phase

The main idea of the initial phase is to familiarise the system analysts to the problem. The methods used for acquiring the basic understanding of the problem were two stakeholder interview sessions and studying some articles of magazines related to the topic. The goal of the prototyping was set to be the capture of the main system requirements. Based on the interviews three top-level user classes of the system were identified: the car driver, traffic warden, and the system administrator. Also, the most important high-level tasks for each of the main user classes were uncovered. Other, non-functional requirements were also discovered for the system, such as its ease of use. The initial phase took 17 man-hours to complete.

4.2. Use Case Analysis

In this phase of the process it was determined that the objectives could be achieved through prototyping and thus the process should be continued. The information gathered during the initial phase was analysed and three use cases were chosen for prototyping. The chosen use cases were high-level descriptions of the most important tasks the different top-level user classes should achieve: the car driver parking a car, the car driver ending parking, and the traffic warden checking the parking area. An example of a use case description at this phase is presented in Appendix 2. The use case analysis phase took 5 man-hours to complete.

4.3. Prototype Design

This 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 still images, but also some animations were used. Also, the structure of the prototypes was designed. The design of the three prototypes was conducted in 13 man-hours.

4.4. Prototype Construction

In this phase the designs of the previous step were implemented. There was not any domain specific database of multimedia objects or previous versions of prototypes available that could have been used as the basis for modifications. The tool used to construct the prototypes was Microsoft PowerPoint, which loosely fulfils all the requirements set for the prototyping tool. The primary source for the multimedia objects used was Microsoft Digital Gallery, but also some other web resources were used. Some of the objects were constructed or modified by hand with Jasc Software Paint Shop Pro. After the screens of a prototype had been constructed according to the design, they were assembled to a PowerPoint presentation. Linking the screens together was achieved by jumping to a specific screen using the keyboard and screen number. The prototypes were tested by executing all the scenarios by displaying all their respective screens in correct order. An example of a prototype structure and screens is presented in Appendix 3.

Prototype construction took 14 man-hours, of which a major part was consumed in searching relevant multimedia objects.

4.5. Prototype Demonstration Session

The prototypes were demonstrated in one session with two customer representatives and a system analyst. One of the customer participants had an extensive technical knowledge about the topic of the example case, while the other had a more business-oriented background. The purpose of the session was to create a better understanding between the fictive 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 course as well as the purpose of 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 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 drivers car was parked. Finally, the car driver left the restaurant to get his car and drive back to his work. 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. Finally, the analyst executed the prototype once more while explaining the preferred solution at each event.

The demonstration session lasted for one hour and 45 minutes.

4.6. Session Analysis

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 altogether. An example of a use case after the session analysis phase is presented in Appendix 2. Completely new use cases, such as: “Car driver reserving parking space beforehand” or “Car driver registering to the parking service” were also discovered. The session analysis phase took 6 man-hours to complete.

5. DISCUSSIONS 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 background knowledge of the customers.

The main advantage of the method perceived by the session participants and the system analyst was the increased mutual understanding in the requirements elicitation session. The session was considered 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 in concentrating 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 (1997b) and Sutcliffe and Ryan (1998), 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 had been also 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 (1997b), the system analyst style could have a considerable impact on the number and quality of the requirements elicited.

However, to fully assess a method of requirements elicitation in software engineering is impossible unless a complete engineering 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 initial phase of the process took 17 man-hours and the use case selection, design, implementation, and session analysis phases took a total of 38 man-hours. The time consumed on the design and implementation would probably have been notably reduced if previous versions of prototypes and domain specific multimedia object database had existed.

In future, more empirical studies are needed to assess the usability of the method in requirements engineering. The future studies should be more extensive so that the system analyst could achieve a better understanding of the problem domain than it was possible in this case, and multiple demonstration sessions with different groups of stakeholders could be organised. The visualisation of the ideas that are not implemented in the prototypes should also be experimented. One possibility could be drawing and adding screens to a prototype during a demonstration session. This could help the session participants in better understanding the idea and its implications in the use case at hand. Experiments could also be conducted with different methods for recording the session information.

ACKNOWLEDGEMENTS

This work was carried out in Telecom Business Research Center (TBRC) 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. Also, thanks are extended to project responsible director Prof. Heikki Kälviäinen and academic instructor Prof. Jan Voracek.

REFERENCES

Asur, S., and Hufnagel, S. 1993. Taxonomy of rapid-prototyping methods and tools. Proceedings from the IEEE Fourth International Workshop on Rapid System Prototyping, 42-56.

Chance, B. D., and Melhart, B. E. 1999. A taxonomy for scenario use in requirements elicitation and analysis of software systems. 1999. Proceedings from the IEEE Conference and Workshop on Engineering of Computer-Based Systems ECBS '99, 232–238.

Constantine, L. L., and Lockwood, L. A. D. 1999. *Software for use: a practical guide to the models and methods of usage-centered design.* Addison Wesley Longman, Inc.

Ghajar-Dowlatshahi, J., and Vernekar A. 1994. Rapid Prototyping in Requirements Specification Phase of Software Systems. Proceedings from the 1994 NCOSE International Symposium.

Hruschka P. 1997. Detailing and deriving system requirements. Proceedings from the International Conference and Workshop on Engineering of Computer-Based Systems, 25-32.

Hsia, P., Samuel, P., Gao, J., Kung, D., Toyoshima, Y., and Chen, C. 1994. Formal approach to scenario analysis. *IEEE Software*, 11(2), 33-41.

Jacobson, I., Christenson M., Jonsson P., and Overgaard, G. 1992. *Object-oriented software engineering: a use case driven approach*. ACM Press.

Khalifa, M., and Verner, J. M. 2000**.** Drivers for software development method usage. *IEEE Transactions on Engineering Management*, 47(3), 360-369.

Kimmond, R. M. 1995**.** Survey into the acceptance of prototyping in software development. Proceedings from the IEEE Sixth International Workshop on Rapid System Prototyping, 147-152.

Kotonya, G., and Sommerville, I. 1997. *Requirements engineering: processes and techniques*. John Wiley & Sons, Inc.

Kulak, D., and Guiney E. 2000. *Use cases – requirements in context*. ACM Press.

Lamb, David A. 2000. CASE tools by category. [WWW-document]. <http://www.qucis.queensu.ca/Software-Engineering/toolcat.html>. (Accessed January 10, 2001).

Leffingwell, D., and Widrig, D. 2000. *Managing software requirements: A unified approach*. Addison-Wesley.

McGraw, K., and Harbison K. 1997. *User-centred requirements: The scenario-based engineering process*. Lawrence Erlbaum Associates, Inc.

McMullen, William L. 2000. Tools survey: Requirements Management (RM) tools. [WWW-document]. <http://www.incose.org/tools/tooltax.html>. (Accessed January 10, 2001).

Rational software corporation. 2001. The unified modelling language. [WWW-document]. <http://www.rational.com/uml/index.jsp>. (Accessed March 20, 2001).

Schneider G., and Winters, J. P. 1998. *Applying use cases: a practical guide*. Addison Wesley Longman, Inc.

Sommerville, I., and Sawyer, P. 1997. *Requirements engineering: A good practise guide*. John Wiley & Sons, Inc.

Sutcliffe, A. 1997a**.** Workshop exploring scenarios in requirements engineering. Proceedings from the IEEE Third International Symposium on Requirements Engineering, 180-182.

Sutcliffe, A. 1997b. A technique combination approach to requirements engineering. Proceedings from the IEEE Third International Symposium on Requirements Engineering, 65-74.

Sutcliffe, A., and Ryan, M. 1998. Experience with SCRAM: a scenario requirements analysis method. Proceedings from the IEEE Third International Conference on Requirements Engineering, 164-171.

Weidenhaupt, K., Pohl, K., Jarke, M., and Haumer, P. 1998. Scenarios in system development: current practice. *IEEE Software*, 15(2), 34-45.

Wiegers, K. E. 1999. *Software requirements*. Microsoft Press.

A use case template

Use case name		
Summary		
User classes		
Preconditions		
Postconditions		
Trigger		
Basic course of events	1	
	2	
Alternative courses of events		
Prototype		
Author		
Date		

A scenario template

Scenario name & description		
Course of events	1	
	2	
Stakeholder review		
Prototype		
Author		
Date		

A use case description before prototype demonstration session

Use case name	Car driver begins parking	
Summary	The car driver parks his car and begins parking	
User classes	Car driver	
Preconditions		
Postconditions	The car is parked and the beginning of parking has been signalled	
Trigger	The car driver has a need to park the car	
Basic course of events	1	The car driver drives a car and has a need to park it
	2	The car driver locates free parking space
	3	The car is parked
	4	The beginning of parking is signalled
Alternative courses of events		
Prototype		
Author	Markus	
Date	9.3.2001	

A use case description after session analysis

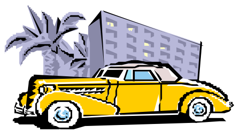
Use case name	Car driver begins parking	
Summary	The car driver parks his car and begins parking	
User classes	Car driver	
Preconditions		
Postconditions	The car is parked and the beginning of parking has been signalled	
Trigger	The car driver has a need to park the car	
Basic course of events	1	The car driver drives a car and has a need to park it
	2	The car driver locates free parking space by driving around and searching
	3	The car driver parks the car
	4	The car driver signals the beginning of parking and billing by calling a number stated in a sign and identifying his car
Alternative courses of events	2	The car driver locates free parking space by using a mobile phone to get information about free parking spaces
	4	The car driver signals the beginning of parking and reserves a certain amount of parking time by calling a number stated in a sign.
Prototype	CD_Park_Car_01	
Author	Markus	
Date	14.3.2001	

The prototype skeleton

At the office

1 / 1

Outside office

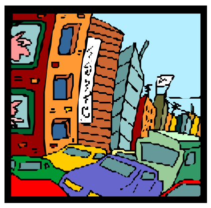


2 / 2

Driving through city

3 / 3

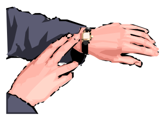
Locating free parking space



4 / 4

5 / 5

The car is parked



6 / 6

Signaling the beginning of parking

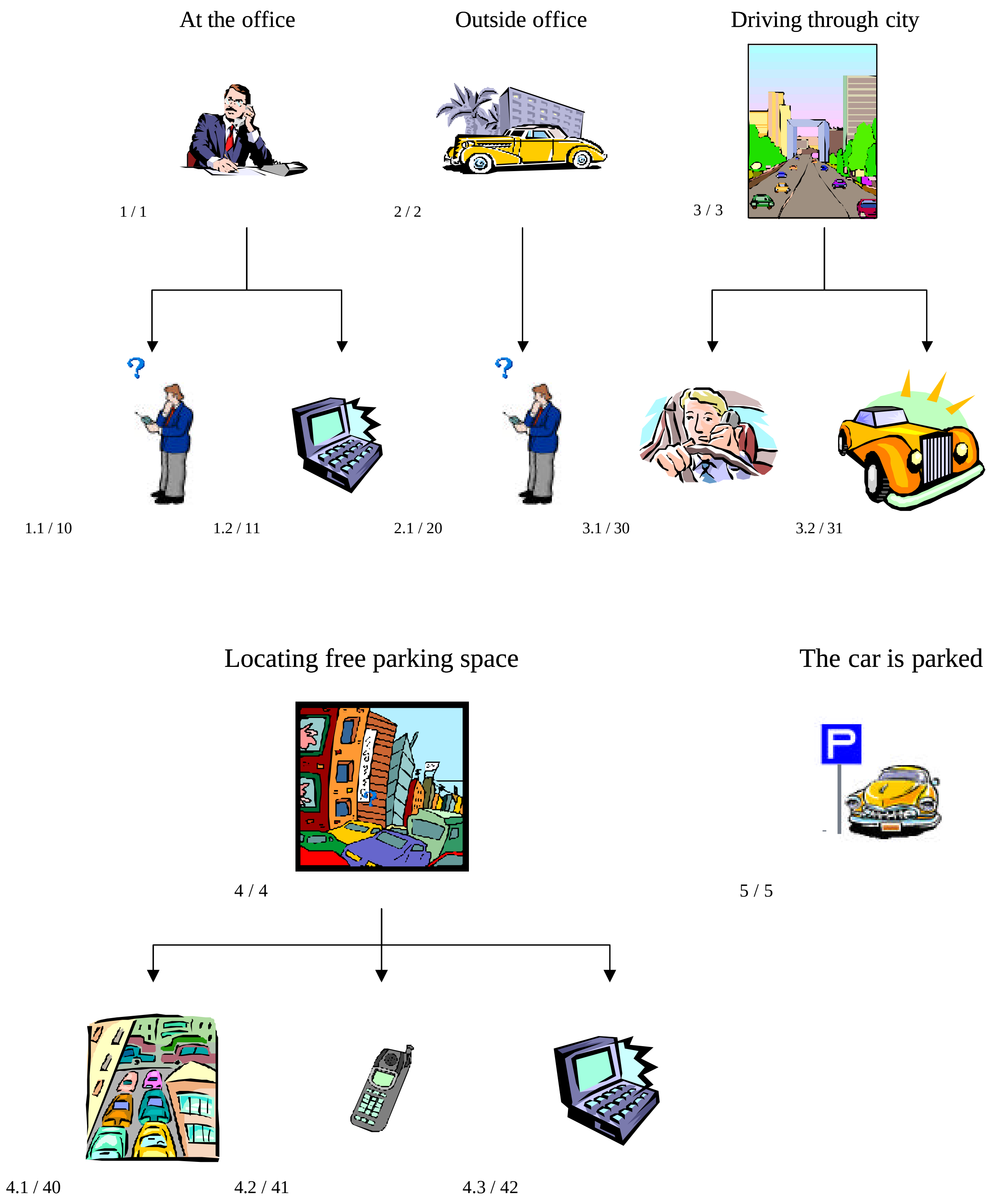
7 / 7

At the restaurant

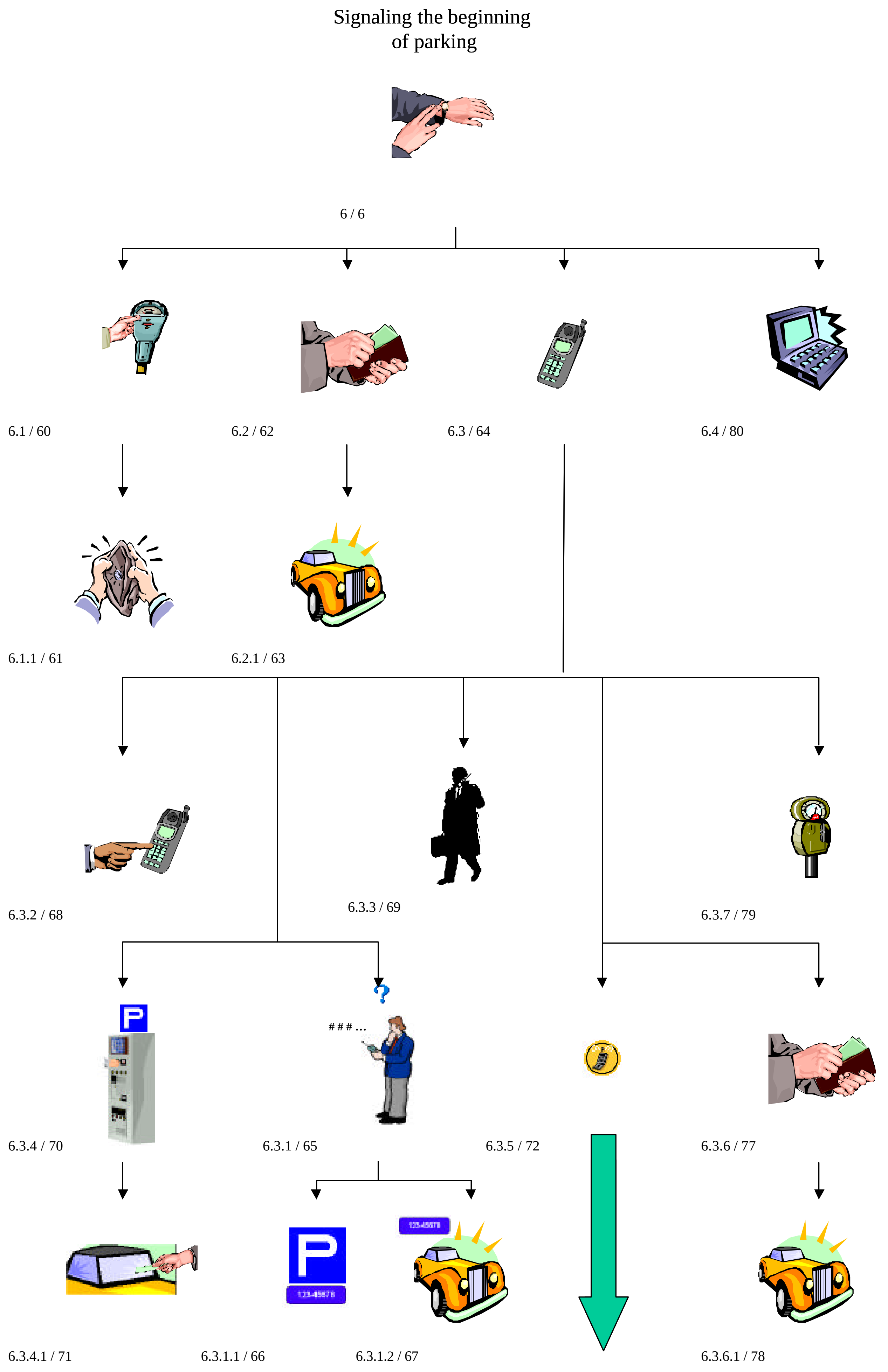
8 / 8

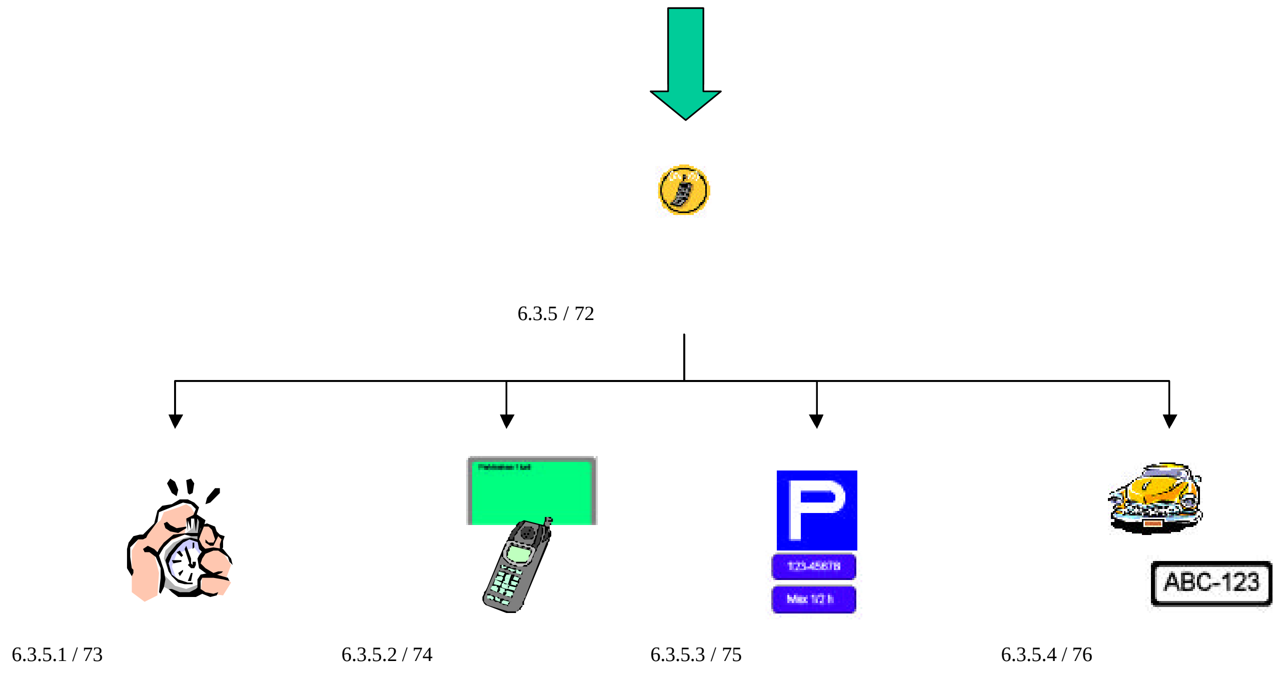
At the meeting

Prototype skeleton with scenario screens



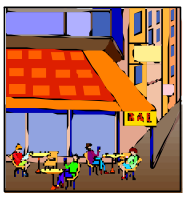
APPENDIX 3: Example of a Prototype Structure





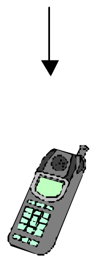
At the restaurant

At the meeting



7 / 7

8 / 8



7.1 / 81