How use cases are defined from scenarios in development of mobile services

Henrik J. Asplund, Mervi Ranta
PM&RG, Laboratory of Information Processing Science
Department of Computer Science and Engineering
Helsinki University of Technology
P.O.B 5400, FIN-02015 HUT, Finland
fax +358-9-451 3293
tel. +358 50 383 8918, Henrik.Asplund@hut.fi
tel. +358 50 551 4441, Mervi.Ranta@hut.fi

Abstract

Technological development is very rapid in mobile and ubiquitous computing. Almost daily there are some new innovations and the business advantage is enormous for the one who is able to bring out new innovations before the competitors.

Perhaps not so strangely the mobile and ubiquitous computing are not well defined areas. When compared to older applications of computer systems, there is much confusion in terminology and even the actual scope of the concepts mobile and ubicomp.

The rapid development leads to the need for reusing the old designs as much as possible. Building and designing everything from scratch leads to an impossible dead end – having to cope with the rapid development and the confusion of methodologies and terms and still striving for business advantages.

Pre-product development of the mobile and ubiquitous services requires very special methodologies. Two methods, use cases and scenarios, are discussed in context of two cases, Ämppäri and VHO Ämppäri. The methods are found to complement each other and it can be seen that using both is actually necessary in the pre-product development phase. These two methods allow also the valuable possibility to reuse information and the analysis of old designs into reusable components.

Introduction

Pre-product development

Pre-product development refers to the process that is before the actual product development process, and which generates the service ideas as an input to the product development process. The pre-product development must be seen as an independent entity and an enabler of, rather than as a part of the product development process.

Mobile and ubiquitous computing services

Mobile and ubiquitous computing are very rapidly developing and changing fields. Mobile computing is somewhat older, and can intuitionally be thought to be interested in devices that are easy to move around. Ubiquitous computing focuses on devices becoming “transparent”, i.e. it is not obvious to the user that the user is using some device. Also context sensitive services can be thought to be ubiquitous computing, as the gadgets start to react to the user’s environment or context and vary their behavior according to this additional information.

Scenarios, services, use cases and realizations

Figure 1 show the basic concepts that are used in pre-product development

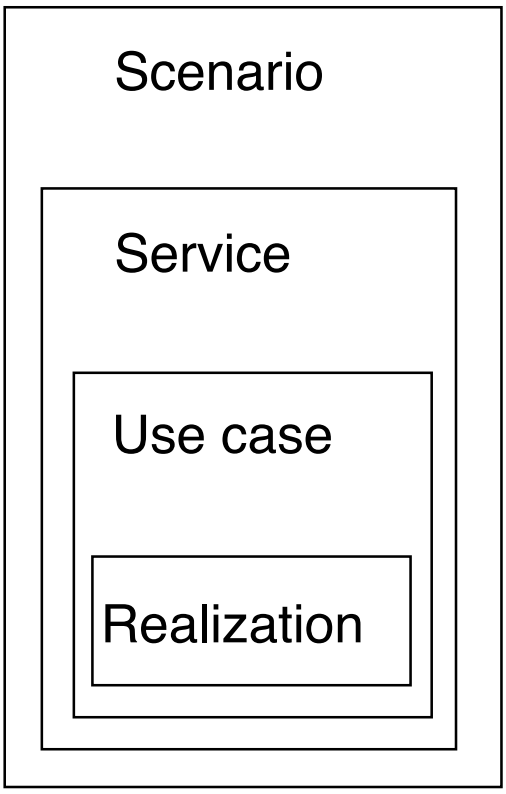

Figure 1. Scenario, service, use case and realization

Scenario

A definition for scenario could be: "The defining property of a scenario is that it projects a concrete description of activity that the user engages when performing a specific task, a description sufficiently detailed so that design implications can be inferred and reasoned about.

Using scenarios in system development helps keep the future use of the envisioned system in view as the system is designed and implemented; it makes use concrete – which makes it easier to discuss use and to design use."[1]

It is convenient to describe actions of the users of a service as scenarios, i.e., as short stories that do not contain description of technology or gadgets. Scenarios can be written in movie script format; leaving the gadgets out of the script is crucial. Otherwise the service will be constrained by the gadgets, what is not the purpose when doing innovative work.

A scenario is a narrative that tells the actions of users but does not contain any description of technology or widgets. A scenario is a specific sequence of actions of the users'. It is divided to multiple scenes according to the contents. Narration is a suitable form, since it can capture almost anything related to mobile services development.

Scenarios are expressed in colloquial language. Therefore they allow communication between the experts and the users. Since the context is the services of the future, interviewing the users is not, of course, sufficient but still necessary.

The scenarios are also very concrete. This prevents losing valuable information by generalizing too heavily during the design process.

Finally, the services may be used as a reference for checking and testing the realization in the context of the original idea. Of course scenarios do not render using the other techniques like requirements engineering unnecessary. The future technologies have to be simulated using

the existing ones. Therefore losing the design context would be fatal, as the actual realization would not in fact be linked to the original ideas.

It must be remembered, that a scenario is never a specification of an innovation prototype or the experiments and tests that must be done with the innovation prototypes. Rather, a scenario is a description of the problem area giving the context to the innovation prototype and the experiments it should allow.

Service

In the pre-product development stage structuring the scenarios into service ideas is much more convenient than using the conventional concept of product prototypes. The reason for this is twofold:

1) The product prototype as a concept is a very gadget- or complete application oriented. It is not uncommon and can actually be desirable that there are many gadgets and applications that together realize some service, instead of all-in-one solutions that are more familiar from the desktop computers.
2) The pre-product development stage is a wrong place for thinking in terms of final products. Following the software or hardware through to the very final product that can be sold to consumers is much too time consuming to be even implied here.

As is, the service ideas are found from the scenario. An example of a service idea would be “music playing service”, a service that provides music for listening to mobile users.

Use case

In object oriented software design the use cases are commonly defined as typical interaction between a user and a computer system[2]. Use cases are described to capture user-visible function and to achieve well defined, discrete goals.

In pre-product development this kind of definition can be used to give an idea. Rather, another definition is more suitable. Use cases are thought to have three important properties:

Use cases define some interaction between two interfaces, e.g. Interaction between application and network, link layer and physical layer and so on. This means that the distinction between an user and an interfae in computer system is not actually as important as it intuitively seems to be.

Use cases capture some functionality but are not “functions” or method calls. For example, a use case could define that application wants to change quality of service parameters or make a request to policy control module.

The use cases are, indeed, discrete and have well defined goals. The size of a use case has not therefore an upper or a lower boundary.

Use cases are often confused, even in terminology, with scenarios. Use cases may contain information about gadgets, networking and other very technology specific things, which scenarios are not allowed to have. Therefore the most important difference is that the scenarios are written from the users' point of view, and use cases from the systems' point of view.

The use cases must be very focused and generic, where the scenarios are detailed and concrete. It is possible that same use case can be found in multiple scenarios and therefore including very scenario specific information leads to a situation where there is unnecessary separation of similar use cases. The aim is to find use cases that are as generic as possible, to allow one use case apply in several places of the system.

Realization

Realizations are the implementations of some protocol stack, the actual gadgets, the software modules and so. The realization also has the design rationale and documentation as a part of it; without this the realization would be quite pointless, as the design rationale is crucial for reuse and re-evaluation of the design.

There is a distinction that must be made: realizations in sense of innovation prototypes are not the same thing as the final products. They may lack certain features and have extra functionality that is important only for testing, and is even undesirable in final products.

Relationship of scenarios, services, use cases and realizations

The relationship between scenarios, services, use cases and realizations is a many-to-many relationship, to use the terminology of software engineering. This means that from a set of scenarios it is possible to find one or several scenarios, from a set of services it is possible to find one or many use cases and so on.

Of course, if there are no services to be found from the scenarios, new scenarios must be found. It is also possible that the use cases are unrealizable. This means that new technologies must become available before the actual realizations can be made.

The distinction made between realizations and products must be remembered. It means that it is possible to actually choose the scenarios, services and use cases that are to be realized – there is no automaton to do that, nor is there a need to realize all use cases at once.

Chronological freedom

The four elements are not chronologically bound to each other. Sometimes it is quite well to first create use cases and then the scenarios, or begin defining the use cases by analyzing a realization. The order in which the designers actually do their can be very different than the cute models in the project management handbooks claim them to be.

Development of the Ämppäri

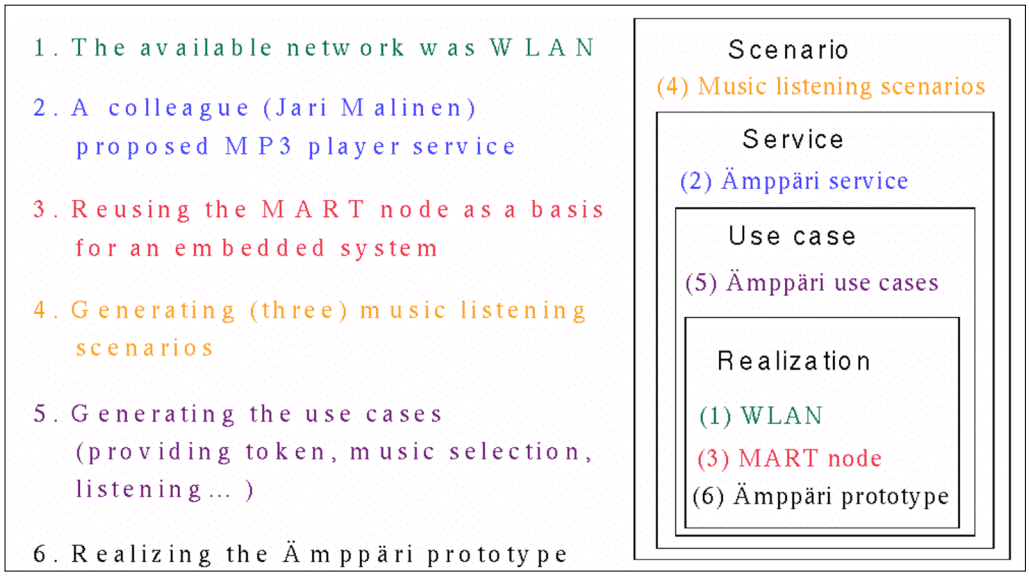

Figure 2. Chronological freedom in designing the Ämppäri prototype

The process that lead to the creation of mobile MP3 player can be analysed to have six phases that also demonstrate the chronological independence of scenarios, services, use cases and realizations. Figure 2 shows these phases.

1) The first phase was that the research group was available to use WLAN (IEEE

802.11b) network for creating new services in a project called GO. At the time WLAN was quite a new technology and there were no self evident uses other than the traditional use as a substitute for an ethernet cable.

2) A colleague (Jari Malinen) proposed the research group that creating a mobile MP3 player service would be a novel idea. At the time (and even currently) the MP3 players were off-line gadgets, meaning that the music must be selected using some desktop computer and then uploaded to the MP3 player gadget. The mobile player would make it possible to select the music and listen to it from the network, without the uploading process. Also the selection could be as wide as a service provider wanted it to be.
3) Originally it was planned to implement the MP3 player device on a Linux laptop. Since an embedded platform MART node[3] was found, it was chosen as the basic technology for the mobile MP3 player. The smaller size and customizability of the MART node were seen as an advantage, as was also the ready-made Mobile IP implementation Dynamics[4].
4) Three music listening scenarios were created. The scenarios describe few common usage situations like a research Leena going to a conference trip to Tampere, during which she wants to listen to music; there are also scenarios for listening to samples in a ticket booking office and before agreeing to go to a concert.
5) Use cases from the scenarios were found. The use cases considered things like token provision (how the access right can be transmitted), how the music is selected, how the listening is done etc.
6) The actual MP3 player prototype later called as Ämppäri was realized. This sounds like a very straightforward process, but actually it included hardware failures and lots of reconsidering of the original plans. Also a design rationale was made so that the design information was captured.

Development of the VHO Ämppäri

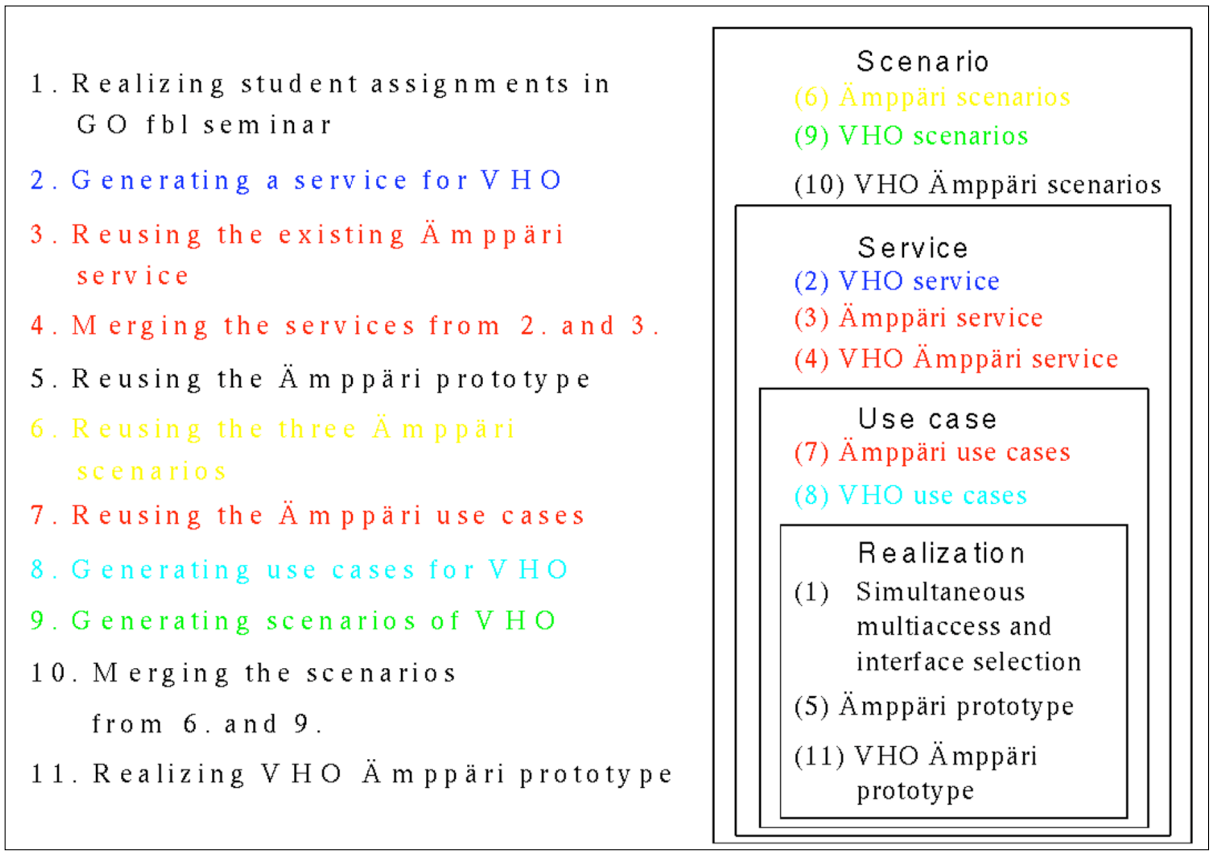

Figure 3. Chronological freedom in design of the VHO Ämppäri

There are considerably more phases before the actual Ämppäri MP3 player in VHO project was realized. The VHO project aims for developing services in a vertical handover environment (vertical handover happens when a connection is transferred from a network type to another, as opposed to horizontal handovers, i.e. connection moving between access points of the same network). The phases can be seen in Figure 3.

1) There was a seminar called GO for better life, where two student groups made realizations of Interface Selection in Simultaneous Multiaccess and Multiaccess. These realizations form a basis for creating prototypes in an environment containing possibility to do vertical handovers.
2) Relevant services for the first VHO prototype were created. Some service that would contain an application jumping from one device to another was thought to be interesting.
3) The existing Ämppäri mobile music playing service idea was reused. The idea of a mobile music service, where the network connection may change from a network to another according to how network connections are available and what gadgets are in use.
4) Service ideas created in phases 2 and 3 were merged to one ultimate VHO mobile music listening service.
5) The Ämppäri prototype design rationale was revisited and analysed, which parts of the original design could be reused and what kind of improvements and implementation ideas could be found.
6) The three Ämppäri scenarios were reused to bring the original music playing service context to VHO Ämppäri.

7) The Ämppäri mobile music playing service use cases were revisited. The use cases were created for mobile music player, and therefore it was deemed that they probably would be applicable in the VHO context also.
8) Network experts among other specialists gave some possible use cases, e.g. ccommunicatiog with the policy control (that decides when to do the handover), moving around in VHO environment (bit more complex than moving around in one single network) etc.
9) Suitable scenarios for VHO were generated so that they matched the service ideas created earlier.
10) Scenarios from phases 6 and 9 were merged – a packet of scenarios that contain information about a user moving around in VHO environment a listening to music.
11) The VHO Ämppäri was realized – this means, that a new version of the Ämppäri prototype created before was implemented. The basic platform changed from an embedded system to a PDA and the playing software was reconsidered – basically the whole realization was built from scratch, but some parts of the rationale was still applicable.

Relationship of use cases and scenarios

From the scenarios to the use cases

The first case, the original Ämppäri player, happened in the more conventional order, i.e. the use cases were extracted from the scenarios by reviewing and analysing what kind of situations and interaction there is between the interfaces of the system or between the user and the system.

There were found to be use cases like “listening to the music”, “selecting the music” and “acquiring the access right token”. As can be seen, these are not functions but rather look quite a lot like usage situations. It is possible, though, that this means the gadget using a network interface, not necessarily the user using a gadget.

From the use cases to the scenarios

When designing the VHO Ämppäri, the first to be discovered were the use cases. There were some existing use cases from the original Ämppäri service and some new use cases were generated by the network experts.

From these use cases, the problem was to find some scenarios that would match the use cases. Of course, some scenarios were gained by reviewing the original Ämppäri scenarios. This would not have been enough, though – there would not have been any justification for actually bringing the VHO environment to the design, it would have looked like a mere whim. Therefore some scenarios were created so that they would actually match the use case – the use cases acted as constraints for possible scenarios, so to speak.

The user viewpoint and the technological viewpoint

The scenarios are expression of users' point of view – the users' actions and interaction with environment is important, not the specific gadgets and network technologies. The use cases, on their part, are the expression of the system viewpoint and the collaboration of any interfaces that there is in the system, not only the usual human-computer interaction. These two exist as methodologies that complement each other. Neither can express all information needed to successfully follow through design process in the pre-product development stage.

Separating the idea and the realization

The use cases can be used as a tool for separating the realization from the service and

scenario ideas. This is very important, since being able to make abstractions on system levels enables easier generation of ideas by freeing the designers from the need of committing to very specific technologies and technology restrictions. After all, mobile technologies develop very rapidly, and the ideas should be valid for longer time than one or two months for real reuse purposes.

Use cases as a tool for structuring the service for realization

A can be seen, the use cases can be used to structure the service before the actual realization. This gives two important advantages:

- The service idea stops being a monolithic entity that is very hard to modularize
- The service must be implemented as a whole, without being able to drop uninteresting parts out of the design

These two properties give the possibility for the experts to pick and choose the interesting features that must be implemented without having to put too much effort and time on parts that are well known and usually not interesting. This also makes it possible to actually take few use cases from some other design, as was done when developing the VHO Ämppäri.

Conclusions

Use cases, scenarios and communication

The use cases and scenarios are a means for communication between the experts, and between the experts and users. A requirement that follows from the need for the communication to be meaningful and sensible is that there is no place or possibility for transgressions or transdisciplinarity. Every expert must be an expert on their familiar, own field of study. An expert who is only an amateur can cause disaster, since there is a risk of misinformation or a risk that some of the crucial expertise is not available.

Chronological freedom vs. polished stories

The important consequence of making the distinction between scenarios, service ideas, use cases and realization and giving them chronological freedom is that the true rationale of the design process can be captured. Unfortunately a requirement that we should first create scenarios, then service ideas, then use cases and last the realizations leads to polished stories that do not tell the actual order of the process, and neither tell the true reasons and true consequences of the design decisions. These polished stories make true reuse impossible, since the most important requirement for reuse is that the whole, true rationale must be well known.

Reuse

In the pre-product development stage the reuse of scenarios, service ideas, use cases and realizations is an objective as such. Since the mobile and ubiquitous computing are fields of very rapid development, the reuse makes it possible to test and experiment with the ideas much faster than if there would be need to do everything from scratch over and over again.

The life-cycles of scenarios, services, use cases and realizations vary and are not bound to each other. A service idea may live long after some of its' realizations (e.g. prototype gadgets) have been dumped to trash bin. Experts may generate new use cases every day, but some scenarios become old-fashioned indeed as the technology develops. This independency makes it possible to focus on the components of the design that are currently interesting and reuse old information when needed.

The related scenarios can be used to find appropriate use cases. This means that by finding some analogy in a new scenario with the old scenarios, it is possible to gain a clue, what or

what kind of use cases could be suitable. This works other way around, also. By seeing the use cases it is possible to create new scenarios or parts of them. An old scenario can help to find novel scenarios by similarity of the related use cases – a way to really reuse information.

References

1. Carroll, Scenario-Based Design - Envisioning Work and Technology in System Development, 1995
2. Booch G., Jacobson I., Rumbaugh J., UML distilled - applying the standard object modeling language, 1997
3. MART project staff, Mobile Ad-Hoc Routing Testbed, 1999, http://www.cs.hut.fi/Research/Mart
4. Dynamics group, Dynamics - HUT Mobile IP, 1999, http://www.cs.hut.fi/Research/Dynamics/