In: Avouris, N. and Fakotakis, N. (Eds.) Advances in Human-Computer Interaction I (Proceedings of the Panhellenic Conference with International Participation in Human-Computer Interaction PC-HCI 2001), Typorama Publications, pp. 45-50.

Bridging the Gap between User Needs and User Requirements

Sari Kujala, Marjo Kauppinen

Helsinki University of Technology
Software Business and Engineering Institute
P.O.B. 9600 FIN-02015 HUT, FINLAND
sari.kujala@hut.fi, hutmhk@cs.hut.fi

Sanna Rekola

KONE Corporation
P.O. Box 677
FIN-05801 Hyvinkää, FINLAND

SUMMARY

Developing a usable product involves more than considering the user interface. The product should also fit into a context of use and meet user requirements. It remains a challenge to represent use contexts and user needs in a way that designers with technical backgrounds are able to make direct use of them. This paper proposes an approach to representing user needs and translating them into user requirements in industrial product development cases.

KEYWORDS : user needs, user study, user requirements, use cases, user centred design, requirements engineering

INTRODUCTION

John Karat [13] describes in his article how the notions of system usability have been expanded. Now, the acceptability of any software product is not seen anymore as being dependent solely on user interface features, but on the way a system fits into its context of use. Furthermore, we have to understand the needs of the users as a way of informing the design process. The human-computer interaction community has generally adopted the term 'user-centred design' for this broader view of developing systems.

In order to develop useful and usable systems we have to understand user needs and contexts, represent them in user requirements and maintain a focus on user requirements throughout the development [6]. In a way, the challenge is how to bridge the gap between the technical world of the designers and the professional world of the users [16]. Rumbaugh [21] suggests that for designers the requirements must be expressed in some structured way, but for users the requirements must still be understandable, as they must verify them. The question remains: How can we describe requirements for each of these groups? In addition, as Imaz and Benyon [9] and many others have suggested, the question is how we can move confidently and consistently between informal descriptions of user data and more formal design representations.

We consider user needs to be the first and usually most informal data to be used in requirements definition. Imaz and Benyon [9] refer to "user stories". By user, we mean the individual interacting with the future product [10]. In product development, the user is also a customer or the customer is sometimes a different person. By customer we mean the individual who pays for the product. Both groups are important stakeholders in product development, but as Coble et al. [6] point out, the customer's primary goal is usually to provide a system, which supports users in their tasks.

User needs refer both to the difference between users' goals and the present condition, which is manifested by user problems and possibilities, and the context of use, which includes the characteristics of the intended users, users' present tasks and environment. This context of use needs to be considered and respected.

Requirements are the more formal descriptions. We have divided the requirements into user requirements and technical requirements [20]. User requirements are written from the user point of view. User requirements describe any function, constraint, or other property that must be provided to satisfy the user needs [1]. Therefore user requirements describe how a future product can hel users achieve their goals effectively, efficiently, and wit satisfaction in their context of use (cf. the definition of usability in ISO 13407 [10]). Technical requirements describe how the product will be implemented to meet user requirements.

The experience we will draw on here comes from our work with the four industrial partners as a part of the QURE project. The QURE project (Quality throug Requirements) is a three-year research project at t Helsinki University of Technology. The aim of t project is both to help three industrial partners in Finlan to improve their requirements engineering practices and to do research work on HCI and requirements engineering. The industrial partners are product development units of medium-size or large companies. The participating companies were the KONE Corporation, Tekla, and Vaisala. Their products are embedded and/or interactive systems, and they are either new versions of older products, or totally new products. The first two authors are researchers of the Qure project with psychology and software engineering backgrounds. The third author is a usability specialist with the KONE Corporation.

In the early phases of the project we explored by interviewing the state of user and customer practices more detailed in two of the companies. Three designers representing different projects were selected from each of the companies. We found out that the projects were technology driven. The companies have a long experience of developing products and the practitioners seemed to rely on this earlier experience, using indirect links to users and customers. Four out of six projects had no direct contact with users or contact was not until the late prototype phase. This means that in some projects the user and customer requirements were not actually gathered or documented. Gathering user information was sometimes passive, methods were informal, and the gathered information was not documented at all.

As a response to the interview results, the companies wanted to develop more user centred requirements processes, as direct contact with users and customers was seen as a success factor in projects. User-centred field studies were piloted to gather user needs. After user needs and requirements were gathered successfully, we faced the challenge of how to introduce the results into projects and maintain the focus on user needs. This paper describes a set of tasks we developed, and our experiences using it in companies.

Difficulty of Representing User Requirements

The user requirements documents we saw in the companies of the study described technical details and their implementation. This was the easiest way of describing user requirements for technical designers. Thus, the fundamental problem was how to introduce the user point of view and user needs for designers, and help designers transform them into relevant user requirements.

First, user requirements should be understandable for users to enable them to give feedback. As both Kyng [16] and Nielsen [19] state, it is not possible to gather user feedback by showing them traditional representations such as requirements specifications. They believe in prototypes and mock-ups, which are more concrete and ready for hands-on explorations. However, the focus of prototypes is in interface issues, which present difficulties to the users. For example, Beyer and Holzblatt [3] describe how users have to understand from the interface what the product does, and how it is structured, and at the same time evaluate whether the system supports their work practices. For that reason, most users tend to complain about interface details, but not the basic structure or functionality of the system.

Beyer and Holzblatt [3] and Kyng [16] consider that the crucial aspect of prototyping is cooperation between users and designers in a process that allows users to live out their own work in the new system and articulate the issues they identify. In practice, we have found that this is not so easy. Users and usability experts face an impossible task in trying to get an overview of the system functionality from a collection of paper prototypes. In addition, we found that designers had difficulties in remembering and conveying user requirements in prototypes. We need something more than prototypes to define the necessary functionality and communicate it to users and other stakeholders.

Use Case Driven Approach

The use case driven approach is a popular solution, which software engineering is providing to help with the problem of gathering and representing user requirements. Jacobson [11, 12] introduced use cases as a part of object-oriented methodology. A use case describes the possible sequences of interactions between the system and one or more actors in response to some initial stimulus by one of the actors [21]. Jacobson [11] employs a graphical use case model, which includes the system as bounded by a box, each actor represented by a person outside the box, and use cases represented as ellipses inside the box. Rumbaugh [21] complemented the model by proposing a written description of use case including name, summary, actors, preconditions, description, exceptions, and post conditions.

Ever since, Jacobson [11] introduced use cases, they have been considered to be a good way of capturing the users' needs and requirements [21, 17] and of modelling functional requirements [5]. Consequently, use cases are very interesting from the HCI-point of view. They are also widely accepted among designers, providing an opportunity to transmit the user point of view to requirements engineering, and resembling 'scenarios of use', the popular technique in HCI [4].

Jacobson's [11] original idea was that by use cases the whole system development starts from what the users wish to be able to do with the system. In this way, the system is built from the users' point of view. However, in practice it is different. In our experience, use cases are written without any knowledge of user needs and the documents are not read by users as Jacobson recommended. Therefore, use cases are written from a technical or interface point of view, but not from the user point of view (see also [18]). The resulting use cases are nearly impossible to understand by users. Either does the use case literature support the idea that user needs should be elicited from users [17].

In this paper, we defined a representation way by which user needs and the user point of view could be implemented in a software engineering process using use cases.

CASE STUDIES

Our first step in this project was to develop new user requirements elicitation methods for our industrial partners. The idea was to help designers gather user and customer needs by direct contact, and use this user point of view in developing usable products. In two of the companies, we worked with product development projects, and with one of the companies, we first worked with a process improvement group, which was focused

on user requirements gathering, and then with a product development project. Table 1 presents the tasks, which were completed in the case companies.

	Vaisala • new version	KONE • new version	Tekla • new product
Identifying stakeholders	X	X	X
Gathering user needs	X	X	X
Describing user needs for use cases	X	X	
Documenting use cases		X	X
Gathering user feedback			X

Table 1: Tasks completed in the three case companies.

Stakeholder identification was seen necessary in order to consider all stakeholders and find representative users, from whom to gather user needs. User needs were gathered by user-centred field studies or user studies using interviewing and observation [3, 8, 15, 22]. In Tekla, user need descriptions were not used in developing use cases, as the user needs gathering, and documenting use cases, were done in parallel. Next, we will focus on how the identified user needs were described and transformed to use cases.

Describing user needs for use cases

First, in Kone the results of the user studies were reported in written documents. Also photographs and video recordings were used to represent user needs. Designers found the reports, photographs and videos useful, but the user study teams found the analysis and reporting of user needs the most challenging part of the process. In two of the user studies 46% of the total time (112 hours) and 43% of the total time (277 hours) were spent in respectively analysing and reporting the results.

In another company, we found that it was not so easy for a technically oriented designer to use user needs in product development. The designer was able to write user study reports as he was advised, but he couldn't see how to use the documents in user requirements definition.

We found that a slightly more formal way of representing user needs was needed, so designers could use the information in analysing and rationally selecting a good combination of user needs for inclusion into the future system, and transfer the descriptions to use cases.

We thereby represented users' current task sequences in a box-diagram. Figure 1 describes a simplified example of the diagram for handling an elevator emergency call in Kone. The aim of this case was to develop a new and better system, which could offer both voice and data connections from an elevator to its related call centre.

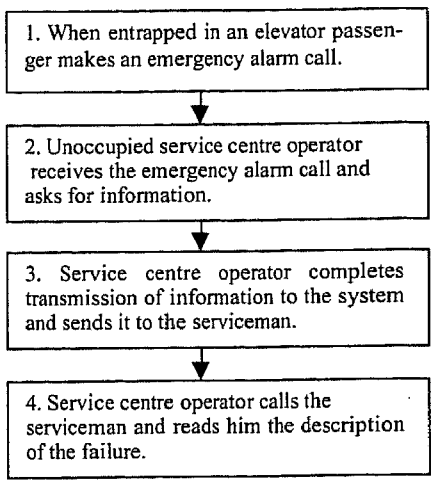

Figure 1: Users' task sequence diagram on elevator emergency call handling process.

Our hypothesis was that the task sequence is an initial description of user tasks which should be redesigned and translated into use case descriptions. Written in the users' own language, and from the users' point of view, the task sequences would convey the user point of view to use cases and requirements. Then we linked the task sequence steps to users' problems and possibilities which we had identified in the user studies (Table 2) Our hypothesis was that it is easier to use user needs data in design when the findings are connected to the task sequence, that forms the basis for use case descriptions.

We have initial evidence that these user need tables were useful for designers. The technically oriented designer got insight from one example table created for him. He got enthusiastic and wanted to make these kinds of tables from all of his findings. His project manager assessed that he could describe 70% of the requirements in this way. Also in another project, a designer who had not participated in user needs gathering found the user need tables useful.

We found in the third company that writing use cases without user needs tables was difficult. In a pilot project the use cases were written before user studies and thus user needs tables were not available to provide user points of view in users' language. The designer did not know all the details of the user tasks to be completed or the natural order of the tasks, and describing the use cases by users' language was difficult. We had to spend time reminding designers about the user point of view.

Task sequence:	Problems and possibilities:
Step 1: When trapped in an elevator, passenger makes an emergency alarm.	• Passengers want to get out of the elevator as soon as possible • All kinds of passengers must be able to make an alarm call (blind, foreigners etc.) • Sometimes passengers may make false alarms unintentionally. • Passengers may be in panic. • Passengers need instant confirmation that they have created a connection to the service centre operator and that they are going to get help.
Step 2: Unoccupied service centre operator receives the emergency alarm call and asks for information.	• Different versions and types of remote monitoring systems. • Passenger is the only information source. • Service centre operator does not notice the emergency alarm call.
Step 3: Service centre operator completes transmission of information to the system and sends it to the area serviceman.	• Laborious phase for the service centre operator. • Simultaneous calls must be differentiated. • Serviceman cannot see all information. • Inadequate information from a site system. • Possibility: Instructions as to how to operate the system. • Possibility: Possibility to open phone line from Call Centre to the elevator.
Step 4: Service centre operator calls the serviceman and reads the description of the failure.	• Extra work for the service centre operator.

Table 2: User need table.

USE CASE:	Making An Emergency Alarm Call
Summary:	An entrapped passenger pushes the emergency alarm button in order to get help. A service centre operator receives the emergency alarm call and informs the passenger that a serviceman will come and let the passenger out of the elevator.
Actors:	Passenger and service centre operator
Preconditions:	An elevator has stopped between floors and there is a passenger in the elevator. The goal of the passenger is to get out of the elevator safely and as quickly as possible.
Basic sequence:	Step 1: The passenger presses the emergency alarm button. Step 2: The service centre operator gets a visible notification about the emergency alarm call on the screen with an optional audio signal. Step 3: The service centre operator accepts the emergency alarm call. Step 4: The system opens a voice connection between the service centre operator and the passenger. Step 5: The system indicates to both the passenger and the service centre operator that the voice connection is open. Step 6: The system guides the service centre operator as to what information to ask of the passenger. Step 7: The service centre operator informs the system that the emergency alarm call is correct.
Exceptions:	Step 1: If an entrapped passenger does not push the alarm button long enough (less than 3 seconds), the system alerts the passenger with a voice announcement. Step 7: If the passenger has pressed the emergency alarm button by accident, the service centre operator informs the system that the emergency alarm call is false. The system resets the emergency alarm call.
Post conditions:	The entrapped passenger knows that the service centre operator will contact a serviceman who will help the passenger out of the elevator safely as soon as possible.

Table 3: A use case for handling an elevator emergency call.

Documenting use cases

In Kone, we have proceeded so far as use cases were written using user need tables and list-based requirements documents. Table 3 presents a simplified use case from the same case as in the user need table (Table 2). We used Rumbaugh's [21] way of describing use cases, except that we organised the written description of the use case in steps with numbers and connected the exceptions to steps identified by numbers. We also described the goal of the user in Precondition-part: what users are trying to accomplish and why (cf. [7]).

A total of seven use cases was developed for the system. About three days were spent in documenting this particular use case. Actually, it included 14 steps and 13 exceptions. Designers found the use case descriptions useful. Use cases helped designers to gain a higher-level view of product requirements and identifying missing undefined details. As this company had a long experience of developing these systems, a lot of information was undocumented up to the present.

Gathering user feedback

In Tekla, the user studies were conducted parallel to documenting use cases, and drawing user interface pictures, although, due to the shortage of time, the use cases were not based on these user studies. It was nevertheless possible to gather feedback from users with paper prototypes and use cases at the time when user needs were gathered. First, the feedback was gathered only with paper prototypes. We found that although important feedback was gathered about the usability of prototypes and interface details, the users nevertheless wanted more information about the functionality of the system. Also the usability expert had difficulties in understanding the whole functionality from few paper prototypes.

When use cases were shown to a user before the prototypes, it was easier to communicate the functionality to the user. The entirety of functionality was clearly described by Jacobson's [11] use case diagram, but we formed the opinion that the idea of the product should have been described shortly before presenting the diagram, as the user's first interpretation of the product was based on his current way of conducting the same tasks. Anyway, we got useful feedback from use case descriptions, as the user commented on the order of the steps, and pointed out details missing from the functionality. However, the user misinterpreted one aspect of the use case descriptions because he assumed that the system would support his current way of working. He thought that while he was operating with the system, he could immediately see the changes made in a figure on the screen. According to the use case description, a user has to first make all the changes and only at the end of the task he see the changes in a figure. The user interpreted the functioning of the paper prototypes as well on the basis of his current way of performing the tasks and this resulted in usability problems. Our view is that utilising the users' present processes would have made the product both more intuitive for them, and more usable.

CONCLUSIONS

As Armour [2] argued the hard part of building systems is not building them, but knowing what to build—it is in acquiring the necessary knowledge and identifying the areas of ignorance. In our experience, many designers think they know everything necessary about users and their needs, whereas actually they don't know, but they don't know that they don't know. Therefore, we have presented a representation way, by which user needs and the user point of view can be stated explicitly in requirements definition, and by which the gap between user needs and user requirements can be decreased. The following lessons have been learned:

- **User need tables are a useful way to represent user needs to make them understandable and useful for designers.** In user need tables we linked users' problems and possibilities to their task sequences. We found in two companies that this way of representing user needs is useful. The designers do not merely receive information about what users need, but they also see how to utilise the users' present processes, how the present processes should be redesigned and the context of use considered. As the users' problems and possibilities are linked to task sequences, the problems are not too vague and can be analysed systematically.

- **User need tables help the designers translate user needs to user requirements and write use cases from the user point of view.** User need tables provided an overview of user tasks and task sequences, users' needs and language. When the user need tables were missing, we had to spend time reminding designers about the user point of view. Furthermore, the use case descriptions missed th necessary level of details.

- **Use cases help designers to gain a coherent vie of the product.** In this way, undefined missin details were identified and definition work did no proceed too quickly to technical details. The designers said that they could use the use cases as checklists to guide the definition work and writing instructions. They also noticed that use cases could be used as test cases.

- **User needs gathering is a necessary step to develop a usable product.** We need real data on users and their needs, gathering user feedback with use cases is not sufficient. Users are still interpreting use cases on the basis of their present way of performing the tasks, and these implicit assumptions risk the mutual understanding between users and designers. However, the user need-gathering seems to require special training.

All in all, when we consider product development, we do not have any specific customer to work with or who would deliver user requirements. We believe that user and customer data have to be actively elicited from users and customers by direct contacts [14] and described in a way that designers can easily utilise. Connecting user need tables and use cases are an initial step towards building a process from user needs to user requirements.

BIBLIOGRAPHY

1. Abbott, R. J. An Integrated Approach to Software Development. Wiley, New York, 1986.
2. Armour, P. G. The five orders of ignorance. Communications of the ACM, 43, 10, 2000, 17-20.
3. Beyer, H. and Holtzblatt, K. Contextual Design: Defining Customer-Centered Systems. Morgan Kaufmann Publishers, California, 1998.
4. Carroll, J. M. (ed.) Scenario-Based Design: Envisioning Work and Technology in System Development. John Wiley & Sons, New York, 1995.
5. Chandrasekaran, P. How use case modeling policies have affected the success of various projects. In Addendum to the 1997 ACM SIGPLAN Conference on object-oriented programming, systems, langauages, and applications. pp. 6-9.
6. Coble, J. M., Karat, J. & Kahn, M. G. Maintaining a focus on user requirements throughout the development of clinical workstation software. In proceedings of the CHI'97, pp. 170-177.
7. Constantine L. L. Essential modeling, use cases for user interfaces. Interactions, 2, 2, 1995, pp. 35-46.
8. Hackos, J. T. and Redish, J. C. User and Task Analysis for Interface Design. Wiley, New York, 1998.
9. Imaz, M. & Benyon, D. How stories capture interactions. In A. Sasse & C. Johnson (eds.), Human-Computer Interaction—INTERACT '99: Proceedings of the Seventh IFIP Conference on Human-Computer Interaction, Vol. 1, IOS Press, pp. 321-328.
10. ISO 13407. Human-centred design processes for interactive systems. ISO/TC159/SC4. International Standard, 1999.
11. Jacobson, I. Object-Oriented Software Engineering, Addison-Wesley, Readin, MA, 1992.
12. Jacobson, I. The use-case construct in object-oriented software engineering. In Carroll, J. M. (ed.), Scenario-Based Design, Envisioning Work and Technology in System Development. John Wiley & Sons, New York, 1995, pp. 309-336.
13. Karat, J. Evolving the scope of user-centered design. Communications of the ACM, 40, 7, 1997, 33-38.
14. Keil, M. & Carmel, E. Customer-developer links in software development. Communications of the ACM, 38, 5, 1995, 33-44.
15. Kujala, S. & Mäntylä, M. How effective are user studies? In McDonald, S., Waern, Y., & Cockton, G. (eds.), People and Computers XIV (Proceedings of HCI'2000), Springer-Verlag, pp. 61-71.
16. Kyng, M. Making representations work. Communications of the ACM, 38, 9, 1995, 46-55.
17. Lee, J. & Xue, N.-L. Analyzing user requirements by use cases: A goal-driven approach, IEEE Software, 16, 4, 1999, 92-101.
18. Lilly, S. Use case pitfalls: Top 10 Problems from real projects using use cases. Proceedings of Technology of Object-Oriented Languages and Systems, TOOLS 30, 1999, pp. 174-183.
19. Nielsen, J. Usability Engineering. Academic Press, London, 1993.
20. Rombach, H. D. Software Specification: A Framework. SEI Curriculum Module SEI-CM-11-1.2, Software Engineering Institute, Carnegie Mellon University, January, 1990.
21. Rumbaugh, J. Getting started - Using use cases to capture requirements. Journal of Object Oriented Programming, September 1994, 8-23.
22. Wood, L. E. Semi-structured interviewing for user-centered design. Interactions, IV.2, 1997, 48-61.