2007

From Rich User Requirements to System Requirements

Michael Przybilski
University of Helsinki, Michael.przybilski@cs.helsinki.fi

Tuure Tuunanen
University of Auckland, tuure@tuunanen.fi

42. From Rich User Requirements to System Requirements

Michael Przybilski
Helsinki Institute for Information Technology (HIIT), Basic Research Unit
Department of Computer Science,
University of Helsinki
Michael.przybilski@cs.helsinki.fi

Tuure Tuunanen
Department of Information Systems & Information Management,
University of Auckland
tuure@tuunanen.fi

Abstract
In recent years the usage of information systems has changed dramatically. Today many information systems are developed for non-organizational users. These wide-area end-users are often socially, as well as geographically very widely dispersed, which makes it for organizations that develop information systems extremely difficult to know who their users are, or what they expect. Previous research has claimed that rich user requirements information is necessary, in order to understand how to serve this audience right. However, at the same time current requirements engineering methods, capable of providing this rich information, do not serve the needs of designers and developers, who actually implement the services and who need precise knowledge of system requirements. It appears that there is a severe gap in the communication of requirements between end-user, analyst, and designer. We have the design science research agenda to develop a method for extending one advanced requirements engineering method, WARE, to provide support for the full spectrum of communication. Our study presents results of ongoing research program, studying the innovation possibilities of Mobile Presence technology. Our method enables analysts to make the transition from rich user requirements to system requirements, which designers and developers can use in their implementation work.

Keywords: Requirements Engineering, Rich User Requirements, System Requirements, Wide Audience End-Users, Design Science Research

Introduction

We live in a time when we no longer use information systems (IS) primarily for work related tasks, but more often these systems are used by a new type of end-user, consumers.
This changes how we see the end-user of these systems, as previously most information systems and software have been developed for organizational users. Instead, we are now facing an era of wide-audience end-users (Tuunanen 2003), who are often more consumers of IS services. This change from organizational needs to consuming IS services is affecting how the requirements engineering (RE) work needs to be done. It is necessary to give even greater emphasis to communicating the needs of wide audience end-users to analysts and, more importantly, to the designers and developers, who implement these IS services. The need to understand user requirements has been a driving force behind a very extensive culture of requirements engineering method development (Nuseibeh et al. 2000). Practitioners and academics alike have been trying to find ways to elicit analyze and model requirements in order to find most efficient ways.

Lately, researchers have presented that by using rich requirements information, it is possible to efficiently discover the user requirements (Peffers et al. 2005), and communicate them from the end-users to analysts. Peffers and Tuunanen (2005) have applied media richness to user requirements. This refers to the capability of the media to carry complex, multidimensional information and cues that help message recipients to better understand the intended message (Daft et al. 1986). Rich user requirements, in turn, thus help stakeholders to overcome equivocality of user requirement as it provides sufficient multidimensional clues to help them understand the intended meaning of complex and ambiguous messages (Peffers et al. 2005). Tuunanen et al. (2006; 2004a) have reported a priori results of success with the Wide Audience Requirements Engineering method that applies the concept of rich user requirements via the use of laddering interview techniques and theme clustering of requirements, which enables the provision of a meaningful link between end-users and analysts.

However, present literature does not provide suggestions on how to strengthen the analyst-designer link. Several studies have shown successful ways of using rich requirements information to convey the needs of end-users to analysts (Browne et al. 2002; Browne et al. 2001; den Hengst et al. 2004; Tuunanen et al. 2006), but the current literature offers no straightforward solutions on how to extend the communication to designers and developers, when using advanced requirements engineering methods, like WARE (Tuunanen et al. 2004a). Moreover, the current, more designer-oriented requirement engineering methods, such as scenarios (Haumer et al. 1998), often start with a different agenda for requirements elicitation. Instead of trying to discover requirements they frequently rely on contextual factors of a use-situation (Holtzblatt et al. 1993), or a scenario of the probable use-situation. Similarly, another well-used method, prototyping, usually assumes that it is already possible to present something to the potential users, such as a mock-up, or a prototype of the application.

Our study aims to provide a feasible approach to the described problem: **How can we transform rich user requirements to system requirements that designers and developers can better understand and use?** We see that one feasible approach in a stronger integration of existing modeling techniques, such as the unified approach, using the Unified Modeling Language (Bjorkander et al. 2003) and the well established data modeling approach, using entity-relationship diagrams (Chen 1976). This way we consider it possible to bridge from advanced RE methods to current design practices, and to provide a stronger link between the analyst and the designer, without limiting the possibilities of developing radical, innovative solutions. This way it is also possible to retain the already gained link between end-users and analysts, and to integrate designers and developers, with their knowledge of the target domain, and their expertise and experience, into the information cycle.
Our study is a part of an on-going mobile presence research program (Tuunanen et al. 2006), which uses the collected wide-audience end-user requirements of 80 interviewees from Helsinki, Hong Kong and Las Vegas, to understand the innovation possibilities of presence, a new emerging mobile technology.

In following, we provide a brief review of current requirements engineering methods from the perspective of requirements elicitation and the connection of information for end-users, analysts and designers. Then we take a look at current modeling practices and their demands with regard to their integration with advanced RE methods. After, we present the methodology of the study, our mobile presence case and the design research effort to develop a method. Finally, we evaluate the developed method and conclude.

Requirements Engineering Methods

The literature offers many methods for handling the problems associated with requirements elicitation. Textbooks primarily refer to interviews, use-cases, soft systems methods, scenario analysis, observation and social analysis, ethnographic analysis, requirements reuse and prototyping. A well-cited recent review by Nuseibeh and Easterbrook (Nuseibeh et al. 2000) provides one way of classification: Methods are divided into six meta-groups: 1) traditional methods, 2) prototyping, 3) group elicitation, 4) contextual methods, 5) cognitive methods, and 6) model-driven methods. In following we review these meta-groups from the perspective of the end-user-analyst-designer-link, and their integration of techniques that are used to gather requirements. This analysis is summarized in Table 1.

Traditional Methods: Nuseibeh and Easterbrook (Nuseibeh et al. 2000) list as traditional methods a broad range of generic data-gathering methods, such as questionnaires and surveys, interviews etc.. The linking emphasis of traditional methods can be characterized as being between end-user and analyst. The analyst's role is to collect the requirements either from end-users or from existing documentation. Therefore, the documentation is usually requirement-content-driven and modeling is not emphasized, but can be supported via entity-relationship diagrams or other modeling techniques.

Prototyping has been referred to by many researchers as a good way of getting feedback from end-users (Mathiassen et al. 1995). Davis (1982) has promoted using prototyping when end-users are not able to express their requirements and when they need help in visualizing the new possibilities of a system. Therefore, prototyping can provide throughout two-way linking possibilities from end-user to analyst, and finally to designer and developer (Tuunanen 2003). The limiting factor in prototyping is in its main strength: it is necessary to have a (visual) prototype, or mock-up, to demonstrate the features to end-users. Modeling support for prototyping has so far been very limited. However, the recent Model-Driven Engineering (MDE) developments from the Object Management Group (OMG) are promising. Namely, domain specific modeling is seen as a promising way to capture the best sides of connecting stakeholder groups and still retain strong modeling support (Atkinson et al. 2004; Selic 2003).

Table 1: Categories of RE methods, end-user, analyst, designer links, and their innovation capabilities

Technique Category	***Linking Emphasis***	***Modeling Support***
1.Traditional methods	End-User and Analyst	Limited
2.Prototyping	End-User, Analyst and Designer	Good
3.Group elicitation methods	End-User and Analyst	Limited
4.Contextual methods	End-User, Analyst and Designer	Good
5.Cognitive methods	End-User and Analyst	Limited
6.Model-driven methods	Analyst-Designer	Good

Group elicitation methods contain a wide range of methods, the purpose of all of which is to elicit requirements from groups of end-users. Group elicitation practices aim to foster stakeholder agreement and buy-in, while exploiting team dynamics to bring out a better understanding of the needs. Group procedures include, for example, brainstorming and focus groups, or group support system (GSS) workshops. Group Support Systems can provide a very rich set of user requirements and are thus efficient in linking end-users and analysts (Bragge et al. 2005a; Bragge et al. 2005b). However, so far GSS systems have been designed to provide reports, based on the information modeling needs of analysts and managers. The modeling information, needed by designers and developers has been mostly lacking.

Contextual methods include the use of ethnographic methods, and ethnomethodology and conversation analysis, both of which apply fine-grained analysis to identify patterns in conversation and interaction. Contextual design (Holtzblatt et al. 1993) includes strong modeling and elicitation components, to provide the needed link between end-users, analysts and designers. Contextual design is considered to offer best of both sides from the selected perspectives. The only limitation the method is its reliance on contextual input of visual prototypes.

Cognitive methods have been originally developed for knowledge acquisition purposes (Shaw et al. 1996). These include protocol analysis (in which an expert thinks aloud while performing a task to provide the observer with insights into the cognitive processes used to perform the task), laddering (using probes to elicit the structure and content of stakeholder knowledge), card sorting (asking stakeholders to sort cards into groups, each of which has a name of some domain entity etc. Browne et al. (Browne et al. 2002; Browne et al. 2001) have stated that by using laddering, analysts are enabled to produce a richer set of requirements compared to other methods. Our experiences with WARE method concur with this observation (Tuunanen et al. 2006; Tuunanen et al. 2004a) and have provided good a priori evidence of an efficient connection between end-users and analyst, supporting the understanding and discovery of requirements. However, cognitive methods do usually not provide linkage between analysts and designers through a strong modeling component as is the case also with the WARE method. The methods can also provide very in-depth information about the end-users needs and wants (Browne et al. 2002; Browne et al. 2001).

Model-driven methods usually provide a specific model of the type of information to be gathered, and use this model to drive the elicitation process. Examples of Model-driven approaches are goal-based methods (van Lamsweerde et al. 1998; van Lamsweerde et al. 2000), scenario-based methods (Haumer et al. 1998; Sutcliffe et al. 1998). Problematic is also the fact that these methods usually require a thorough knowledge of the system domain area and a high level of knowledge of related work practices. Users without basic skills in modeling languages, like E-R diagrams, or UML, find it difficult to grasp the meaning of the diagrams and the requirements information behind them. Furthermore, these methods rely on visualizing probable use situations of existing work practices or services. However, these methods naturally provide very strong modeling support for the designers and analysts.
Above we have analyzed requirements engineering methods at a categorical level in order to recognize their capabilities to support end-user-analyst-designer communication linkage (Tuunanen 2003). Furthermore, we have analyzed the method categories, to understand whether they support information modeling integrating the requirements knowledge with design. Our analysis reveals that even though both, end-user-analyst and analyst-designer linkage are supported, only two of the method categories at the moment provide means for enabling cross-stakeholder communication: contextual methods and prototyping. While this is an encouraging result, the two method categories, cognitive and group elicitation methods, which potentially can provide an in-depth understanding of end-users' requirements (Bragge et al. 2005a; Bragge et al. 2005b; Browne et al. 2002; Browne et al. 2001; den Hengst et al. 2004), lack the strong modeling capabilities that would link analysts and designer.

In the following section, we review what requirements modeling needs today's designers and developers face, in order to understand how requirements engineering methods should be extended, in order to facilitate the full spectrum of communication, i.e., enabling end-users, analysts, designers and developers to share requirements information.

Requirements Modeling Needs

One of the biggest problems in software development is still to find the correct level of abstraction for a particular problem domain (Mylopoulos et al. 1999). Programming languages have evolved from assembly language, to first attempts in structured programming in the late 1960's. Then came the advent of object-oriented programming and more recently component-based software development. New developments always attempted to provide a higher level of abstraction, in order to enable humans to develop increasingly complex applications and foster the re-use of software artifacts. Current developments try further, to find even higher levels of abstraction, in the form of aspect-oriented programming, model-driven development, and the like. At the same time, special languages were developed, for specific purposes, such as mathematic descriptions, or logical expression.
This evolution was accompanied by an evolution in the field of software systems. While initially systems were build in a monolithic way, in order to adhere to the limited hardware resources available, they evolved, together with the programming languages, into object-oriented systems and from there further, into component-based software systems, with a clear organization of large-grained functionality and advantages, such as an increased level of re-use, more flexibility, and so forth.

Design methodologies have developed at the same time, in a similar way. The initial, functional analysis, used to combine groups of functionalities, was replaced by object-oriented analysis, grouping functionalities and data-types. Component-based modeling goes further beyond, describing the distribution of services over components, their interconnections, quality attributes, and so on.

Software Engineering has provided many techniques, and tools, to assist software developers, as well as designers and help the derivation of system requirements, and the optimal integration of their results in the further development process. Amongst those are structured language specifications, form-based specifications, and graphical models. Perhaps one of the best known and most widely used approaches is the unified approach, which relies heavily on the Unified Modeling Language (UML).

UML depicts a modeling language that enables the specification, visualization, construction and documentation of artifacts of system-intensive processes (Bjorkander et al. 2003). UML defines nine different types of diagrams, class diagrams, object diagrams, use case diagrams, sequence (chart) diagrams, collaboration diagrams, state chart diagrams, activity diagrams, component diagrams and deployment diagrams, used to describe different aspects of the system under development. UML specifies further extensions, used in the software design process, such us the Object Constraint Language (OCL), thus providing the means for functional constraint specification (Richters et al. 2000).

Current developments extend the language for other, often very specific purposes, such as the description of real-time criteria (Selic 2003) and agent technologies. An alternative can be found in the Catalysis process (Brown 2000; Jürjens 2002), which takes into account aspects of the component-based development. It uses a business- or domain model, to describe the client's world, separately from any notion of the target software. System requirements are elicited from the client's needs and thorough the specification of system functionality. The process does however suggest several iterations of elicitation of customer needs and requirements revision. In further steps, these are reflected in the system architecture, the component design, which provides a high level description of the major building blocks (components) of the system and their collaborations, the implementation and object design,

which describe how major building blocks are implemented, and the platform description, which specifies how components interact with each other, how they work together and how to build and assemble interoperable components.
The requirements in either of these processes are however different from those provided by the WARE method. Sommerville (2001) defines two essential types of requirements:

- User requirements: Statements in natural language plus diagrams of the services, which the system provides and its operational constraints.System requirements: A structured document, setting out detailed descriptions of the system's functions, services and operational constraints.

In contrast to user requirements, which are the result of RE methods, such as WARE, and which focus on the services that the customer requires from the system, system requirements, also take into account constraints under which the system operates and is developed.

System requirements can be divided further into functional, non-functional, and domain requirements. Functional requirements are hereby statements of services the system should provide, how the system should react to particular inputs, and how the system should behave in particular situations. Non-functional requirements specify constraints on these services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc. Domain requirements eventually come from the application domain of the system and reflect characteristics of the domain itself.

Designers and developers are working primarily with these lower-level system requirements, which must be obtained from the high-level user-requirements, and be modeled accordingly.
For this it is necessary to be familiar with the target domain area, as it will have a major impact on those requirements, as well as the technologies and techniques that will be used in the development process.

Domain knowledge reflects what kind of constraints have to be observed, but also which kind of readily usable components can be reused, or which legacy components need to be integrated. The term components here refers not necessarily only to the software components in the sense of Component-based Software (CBD), but in a more general way to existing software assets that can be reused. Examples for such assets can be software libraries, frameworks, or middleware mechanisms. Constraints that need to be taken into account are on the other issues such as the targeted hardware, the operating system, or legacy software that needs to be used or integrated.

The modeling of the system requirements requires must take into account the high-level user requirements, the target domain, available, or legacy components that can be used, or should be integrated, and, as well as any further restrictions that will influence the system design, the further implementation, the system deployment, or potentially even its later maintenance. We found some RE methods, which do support creating rich end-user requirements information and therefore providing the basis for also creating radical innovations. However, as we have seen these methods do usually not support very well designers' everyday needs today. For example, with WARE (Tuunanen et al. 2006; Tuunanen et al. 2004a) this gap has been handled by offering roadmap documents (Lehtola et al. 2005). Some efforts has been done in order to link the method to modern design practices with Meta modeling approach (Tuunanen et al. 2003; Tuunanen et al. 2004b). The results have been promising. But still we have been faced with the dilemma of understanding how end-user requirements could be easily transformed to usable system requirements, and finally to functional requirements of the system.

In following we present a case study we were develop a way to infer system requirements from user requirements, thus bridging the gap between analysts and designers, with the help of a domain expert (Grudin 1991).

Design Science Research Study: Mobile Presence Services Case

We use the design science research agenda (Hevner et al. 2004) as philosophical basis for conducting the study. Design research is said to be yet another "lens" or set of analytical techniques and perspectives for performing IS research. Moreover, the researchers claim that design research addresses important unsolved problems (Hevner et al. 2004). When compared to traditional ways of conducting research, design research can be considered to complement the positivist and interpretive perspectives. It involves analyzing the use and performance of designed artifacts, in order to understand, to explain, and very frequently to improve the behavior of the various IS aspects (Orlikowski et al. 2001). Five general outputs have been proposed for design research: constructs, models, methods, instantiations, and better theories (March et al. 1995; Purao 2002; Rossi et al. 2003).

Our research objective is to *develop a method* for integrating an advanced requirements engineering method with the modeling needs of designers. We used the rich user requirements derived from a case study conducted within DiVia project commissioned by LTT research, Inc. In total 13 researchers participated to the whole research effort (Tuunanen et al. 2006). A total of 80 interviews were done in Helsinki, Hong Kong and Las Vegas during summer 2004 to discover rich user requirements. For requirements elicitation and analysis we used the previously developed WARE method (Tuunanen et al. 2006; Tuunanen et al. 2004a). This study uses the analysis results of the second phase of mobile presence study as basis. The gathered requirements have been analyzed with theme clustering (Tuunanen et al. 2006; Tuunanen et al. 2004a). Via this process we have generated a general roadmap for mobile presence services (Tuunanen et al. 2006). These roadmap features are the basis for the presented study.

The context area of our study is a novel mobile service: presence. Presence is a powerful tool for bonding mobile users together and empowering all types of communities. Presence is said to be a mass-market service with a high growth potential targeting all mobile users, because it enables easier, richer, and more discreet communication[1]. It can be an enabler of new services as well as enriching existing ones, which leads to a richer communication experience. A basic presence service could allow users to publish their information and share it with others in order to make mobile communication and services more sensitive and personal. This information may include the availability of the subscriber, the preferred means of communication, the subscriber's whereabouts, as well as visual content for self expression of one's emotion, in order to guide other users' communication decisions while controlling their own information. In following section we present the proposed method.

Designing a Method: From User Requirements to System Requirements

In the following, we describe how the WARE method (Tuunanen et al. 2006; Tuunanen et al. 2004a) was extended, to improve the communication between end-users and developers with

[1] http://www.3g.co.uk

the help of the analyst who posses the domain knowledge, a high level of expertise in the according area. For this study we had the aid of a principal investigator of the mobile presence study, to provide an understanding of the used data gathering and analysis method. As analyst we had a practitioner who was domain expert of the mobile technology field and nowadays pursues a doctoral degree in computer science. The objective was to transform a very rich set of end-user requirements (Tuunanen et al. 2006) into an ordered set of system requirements. The initial starting point were more than 2500 individual requirements, gathered with laddering interviewing technique and clustered by using theme clustering (Tuunanen et al. 2006; Tuunanen et al. 2004a). The requirements engineering process provided us with the data structures for the end-user requirements that are described in the top part of Figure 1. In the following, we focus on the designer's perspective of the analysis process and the different analysis stages. This process is summarized in Table 1.

The WARE method provides us with a set of user requirements, on a rather high level of abstraction. They are derived from interviews with potential users of the system-to-be, as well as "lead users" and typically consist of a mixture of functional requirements, related to the services provided, as well as non-functional requirements, related to the qualitative attributes of the system, such as timeliness, security, and the like, and sometimes also abstract ideas that the potential user might have of the system. These user requirements are contained in a structure, identifying different chains and sub-chains, the idea contained in the chain, an application cluster, a ranking, attributes, consequences, and most importantly, the ladders, describing the actual requirement.

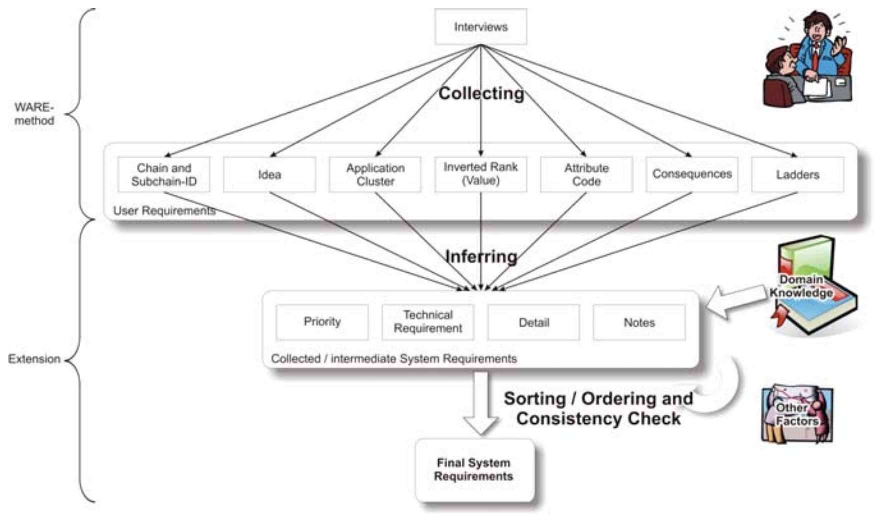

Figure 1: Information Flow and participating Entities in the proposed Process

For a designer, or even more so for a developer, it is still very difficult to find all necessary requirements, and guidelines towards their design and implementation. In order to implement these high-level user requirements, it is necessary to infer low-level, system requirements, which take into account knowledge of the actual domain, as well as experience in the implementation of such systems. How we extended this process, as well as the involved sets of information is depicted in Figure 1.

In a first step, the ladders in the different chains of the user requirements are analyzed for required functionalities, which are noted separately. They describe the basic system requirements and the resulting functionality. At the same time it is necessary to maintain the connection to the related details, and notes, resulting in the data-structure depicted in
Figure 1.

In this step it is furthermore important to try to keep an overview of the functionalities, as similar functionalities should be grouped together. With an increasing number of chains, this problem however increases exponentially. Often it is so that more than one functionality can be derived per chain, which means that additional data has to be added to that chain. It is thus also necessary to maintain several references to the same chains, in order to enable a later clustering of the derived functionalities.

In the following, second step it is possible to order the derived system functionalities and infer furthermore, which user-requirements result in the same, or very similar system requirements. For this process, very specific knowledge of the target system is necessary, in order to know which functionalities will be implemented in which way, or which components can be used to provide a specific functionality or a set thereof.

In the third step, the found functionality clusters are analyzed, and based on the information contained in the linked user requirements, the system requirements are obtained. For this, the initial functionalities are replaced with system requirements, which are annotated by according details and notes. These details and notes can serve as additional sorting criteria later on. As in the previous step, it is still necessary to maintain a link to the chains from which these system requirements originate, which often results in several references to the same ladder.

In a fourth step it is now possible to prioritize the derived system requirements. For this the values (inverted ranks) of the originating ladders, which result in the same system requirement, are summed up. The value initially indicated how important a particular user requirement was to the end-user. By summing up the values of those user-requirements that result in the same system requirement, we can achieve a prioritization of the system requirements, according the importance, assigned by the end-user. It needs to be observed that occasionally the values of some of these functionalities may be zero. While it may be possible to neglect those functionalities entirely, as they do not appear to be relevant, it might still be useful to keep this information, as it will in any case not have any negative influence on the resulting set of functionalities. In future reiterations of the development process, they may however be re-evaluated, and may thus receive a higher importance.

Alternatively, it is possible to offset all values by one, thus assigning all functionalities a non-zero value, which would also not have a negative influence on the sorting process.

The result of the process so far is a prioritized set of system requirements, which reflect the user requirements that have been provided through the initial WARE method. At the same time they take into account in which domain, e.g., on which hardware, in which environment, and for what kinds of users, the system will be developed. The system requirements can also contain alternatives, which have been obtained during the process. These can be the results of varying interpretations of the user requirements, or different options in the implementation. The level of detail provided in these system requirements enables the designers of the system and the developers to implement the system for the specified environment, the chosen language, using existing components and taking into account any further requirements that come from the deployment domain itself.

In a fifth step we can now modify the priorities of the derived user requirements, according to other goals, such as a company's strategic agenda, or the availability of other system components. Thus other factors are able to influence in which order system functionalities are developed, or which alternative in the development process will be taken. This provides additional flexibility and can make the resulting product considerably easier to maintain, or to extend and develop further. At the same time, the system functionalities can be implemented in the form of components that can be easily re-used for future products.
Finally, in the sixth step the list of system requirements is combined with additional information and references, into a final report.

Table 2: The process of transforming rich end-user requirements to functional system requirements

Process Step	***Description***
0. Familiarization with the requirements data and the WARE method	Learning basic components of the WARE method; laddering interviewing principles and how theme clustering is conducted. Understanding what each Meta feature set (idea, application cluster, ladders, etc.) includes.
1. Analyzing the data and initial separation of functionalities	Creating a list of technical functionalities, together with details and notes and with references to their originating chain.
2. Clustering of functionalities	The found functionalities are clustered together. For this it is necessary to keep an overview of previously discovered functionalities. Additional details and notes are made.
3. Grouping into system requirements	Similar functionalities are grouped together. More general system requirements are derived based on domain knowledge and technical expertise. Minor differences in the originating functionalities are kept as details and notes.
4. Ranking	The values of the chains from which each system requirement originates are summed up. The ranking information is used as a priority of the importance of a particular system requirement.
5. Integration of other factors	Further factor that influence the prioritization or the choice of alternatives are integrated and the ranking is modified accordingly.
6. Report	The resulting prioritized list is refined, and complemented with additional information that might be necessary, as well as references to further details for the implementation.

Evaluation

We were successful in extending the WARE method and our preliminary findings indicate that we might have been able to resolve one of the conceptual difficulties faced by advanced requirements engineering methods. The method process description shows that with a reasonable effort a very rich set of end-user requirements can be further distilled to usable information for designers. Furthermore, the needed extension will not require any changes to the original WARE method, but instead the current data structure of the laddering interviews (Tuunanen et al. 2006; Tuunanen et al. 2004a) can be kept as is. This can be considered as a breakthrough since changing the data structure would definitely question the current way of using theme clustering for analyzing the laddering interview data. In addition, in cases where it is not necessary to actually proceed to design phase this means that the analyst is not burdened with an additional modeling effort. Instead, if a decision to continue is given, the proposed method can be seen as the next step in the workflow.

Although, our findings show a high potential for the developed method we can also see some limitations. Our analyst was successful in deriving the system requirements from the selected set of end-user requirements. However, we found that the cognitive limitations of human brain gives limitations to the size used data sets. The largest of the chosen data sets consisted of 128 ladder chains each consisting of 4-8 data points. The data structures included all the items needed for data collections, such as notes on the idea that was used as stimuli for the particular interview part, the actual requirements data (ladder chains of interviews and their IDs (Tuunanen et al. 2004b). This was found to be difficult to handle by one person. Without further assistance, we believe that this may be close to the maximum number of chains that can efficiently be handled, without the risk of major mistakes, or loss of information

Furthermore, by incorporating domain knowledge to establish functionalities, such as categories of functional system requirements, the analyst was provided with the possibility of using further, more complex modeling languages to support design work, such as UML (Bjorkander et al. 2003), which in turn provided the means for using modern component based development (Brown 2000)

Finally, when considering the field of requirements engineering methods research in more general sense, we can see anecdotal evidence that support our literature review and design research science study findings. First of all, it appears plausible that cognitive requirements engineering methods, like WARE, can be extended so that end-user, analyst and designer communication linkage is not disrupted.

Conclusions

In this paper we examined how end-user requirements can be turned into functional system requirements. This is an especially demanding task when we are facing a new set of end-users, wide audience end-users (Tuunanen 2003), who are usually outside of the developing organization and they do not have strong ties with it. This makes enabling the essential communication linkage between end-user, analyst and designer more difficult. We cannot no longer assume that our stakeholders in the project are culturally similar like previously. What is more, the importance of domain knowledge becomes even more central (Mylopoulos et al. 1999). Furthermore, the analyst's role as a catalyst between the end-user and the designer becomes more critical (Grudin 1991). Finally, we face the old problem of end-users not knowing what they actually want (Peffers et al. 2005). Instead of harvesting the requirements from the organization, we actually need to discover them by using advanced requirements engineering methods, such as WARE (Tuunanen et al. 2006; Tuunanen et al. 2004a).

Secondly, we see that the nature of information systems use is rapidly changing. We are in fact starting to be consumers of information systems. Wide audience end-users consume and seek gaining pleasure from information systems in stead of just getting the work done. The used requirements engineering methods development should reflect to this change. How we can support innovation work that is not related to the customs and ways of work environment? In our literature review we concluded that two of the reviewed requirements engineering method categories can support this, namely cognitive and group methods. We used the design science research agenda (Hevner et al. 2004), to conduct a study, which aimed at examining the possibility of using cognitive RE methods (Nuseibeh et al. 2000), and more specifically the WARE method (Tuunanen et al. 2006; Tuunanen et al. 2004a), to enable a requirements communication linkage between end-users, analysts and designers. We used the data of 80 interviews conducted earlier in the mobile presence study (Tuunanen et al. 2006) which consists of a very rich set of end-users requirements and resulted in a data set of more than 2500 individual end-user requirements.

Our study contributes by developing a method for transforming end-user requirements to functional system requirements. Our method extends a current cognitive requirements engineering method, WARE (Tuunanen et al. 2006; Tuunanen et al. 2004a). The previous work within the field (Tuunanen 2003; Tuunanen et al. 2006; Tuunanen et al. 2004a) has told us that the link between end-users and analysts can be handled by available means. However, the two RE method categories supporting the development of wide audience end-user targeted information systems do not at present provide ways to extend the information linkage to the essential character in the projects, designers and developers, who actually need to architect and implement the services. As a solution we have presented that these methods should provide a stronger connection to modeling mechanisms (Tuunanen et al. 2004b), such as UML (Bjorkander et al. 2003), in order to serve the needs of modern design practices.

Our conceptual method of extending the WARE method is a feasible beginning of enabling the requirements information linkage between end-users, analysts, designers and developers. Our straightforward six step method gives both practitioners and academics the knowledge of how to transform a rich set of end-user requirements into functional systems requirements that analysts and designers can use as basis for applying more advanced modeling languages, like UML, or Meta modeling (Tuunanen et al. 2004b). Additionally, we found that the WARE method supported this transformation as is. We did not need to change any of the underlying data structures etc. Instead it was entirely possible to extend the current process. This is a positive argumentation for the use of laddering, interviewing and theme-based clustering analysis techniques in practical requirements engineering work.

Naturally, our research is not without limitations. First of all, the conducted study was limited in the size of used data set. So far we only used few of the Meta feature sets to conduct the study. However, we feel that using a limited data set gave us the opportunity to seek first if proof of concept level evaluation could be gained, and which was concluded successfully. Secondly, it should be noted that we have not yet used the obtained functional systems requirements specification to either model the system with UML or with Meta modeling. Therefore, our findings remain at theoretical level for the moment, even though the anecdotal evidence is there for support of the modeling effort. Furthermore, our previous good experiences with the WARE data structure and Meta modeling (Tuunanen et al. 2004b) provides us with optimism.

In future research, we see important that we, first of all, use the obtained functional systems requirements specification document to model the proposed mobile presence service system. We initially consider using Meta modeling (Tuunanen et al. 2004b) for the task, but on the other hand we find it appealing also to extend this view. Why don’t we try also to use the de facto modeling language, UML, for this purpose? This might lead to interesting research findings between the efficiency of using Meta modeling and UML in this field of research. Finally, we intend to use the gained models to create working prototypes of the mobile presence services. The design work itself will evaluate the usefulness of the obtained information and give indications if more research should be done in the area.

References

Atkinson, C., and Kuhne, T. "Model-driven development: a metamodeling foundation," *IEEE Software* (20:5), September-October 2004, pp 36-41.

Bjorkander, M., and Kobryn, C. "Architecting Systems with UML 2.0," *IEEE Software* (20:4), July-August 2003, pp 57-61.

Bragge, J., Marttiin, P., and Tuunanen, T. "Developing Innovative Information Systems Services Together with Wide Audience End-Users," Hawaii International Conference on System Sciences HICSS38, IEEE, Big Island, Hawaii, 2005a.

Bragge, J., Merisalo-Rantanen, H., and Hallikainen, P. "Gathering innovative end-user feedback for continuous development of information systems: a repeatable and transferable e-collaboration process," *IEEE Transactions on Professional Communication* (48:1), June 2005b, pp 55-67.

Brown, A.W. *Large-Scale Component-Based Development* Prentice-Hall, 2000, p. 300.

Browne, G.J., and Ramesh, V. "Improving information requirements determination: a cognitive perspective," *Information & Management* (39:8), Sep 2002, pp 625-645.

Browne, G.J., and Rogich, M.B. "An empirical investigation of user requirements elicitation: Comparing the effectiveness of prompting techniques," *Journal of Management Information Systems* (17:4), Spr 2001, pp 223-249.

Chen, P.P.-S. "The entity-relationship model - toward a unified view of data," *ACM Transactions on Database Systems* (1:1), March 1976, pp 9-36

Daft, R., and Lengel, R.H. "Organizational Information Requirements, Media Richness and Structural Design.," *Management Science* (33:5) 1986, pp 554-569.

Davis, G. "Strategies for information requirements determination," *IBM Systems Journal* (21:1) 1982, pp 4-31.

den Hengst, M., van de Kar, E., and Appelman, J. "Designing mobile information services: user requirements elicitation with GSS design and application of a repeatable process," The 37th Annual Hawaii International Conference on System Sciences, IEEE, Big Island, Hawaii, 2004, p. 10.

Grudin, J. "Interactive Systems - Bridging the Gaps between Developers and Users," *Computer* (24:4), Apr 1991, pp 59-69.

Haumer, P., Pohl, K., and Weidenhaupt, K. "Requirements elicitation and validation with real world scenes," *IEEE Transactions on Software Engineering* (24:12), Dec 1998, pp 1036-1054.

Hevner, A.R., March, S.T., and Park, J. "Design Research in Information Systems Research," *MIS Quarterly* (28:1) 2004, pp 75-105.

Holtzblatt, K., and Beyer, H. "Making Customer-Centered Design Work for Teams," *Communications of the ACM* (36:10), October 1993, pp 93-103.

Jürjens, J. "UMLsec: Extending UML for Secure Systems Development," UML 2002 - The Unified Modeling Language: Fifth International Conference, Dresden, Germany, 2002, pp. 412-425.

Lehtola, L., Kauppinen, M., and Kujala, S. "Linking the business view to requirements engineering: long-term planning by roadmapping," 13th IEEE International Conference on Requirements Engineering, IEEE, Paris, France, 2005, pp. 439-446.

March, S., and Smith, G. "Design and Natural Science Research on Information Technology," *Decision Support Systems* (15) 1995, pp 251-266.

Mathiassen, L., Seewaldt, T., and Stage, J. "Prototyping and Specifying: Principles and Practices of a Mixed Approach," *Scandinavian Journal of Information Systems* (7:1) 1995, pp 55-72.

Mylopoulos, J., Chung, L., and Yu, E. "From Object-Oriented to Goal-Oriented Requirements Analysis," *Communications of the ACM* (42:1), January 1999, pp 31-37.

Nuseibeh, B., and Easterbrook, S. "Requirements engineering: a roadmap," Future of Software Engineering, ICSE 2000, ACM Press, Limerick, Ireland, 2000, pp. 35-46.

Orlikowski, W., and Iacono, C. "Desperately Seeking the "IT" in IT Research - A Call to Theorizing the IT Artifact.," *Information Systems Research* (12:2) 2001, pp 121-134.

Peffers, K., and Tuunanen, T. "Planning for IS Applications: a Practical, Information Theoretical Method and Case Study In Mobile Financial Services.," *Information & Management* (42:3), March 2005, pp 483-501.

Purao, S. "Design Research in the Technology of Information Systems: Truth or Dare," Georgia State University, Atlanta, USA.

Richters, M., and Gogolla, M. "Validating UML Models and OCL Constraints," UML 2000 - The Unified Modeling Language, Advancing the Standard: Third International Conference, Springer, York, 2000, pp. 265-277.

Rossi, M., and Sein, M. "Design Research Workshop: A Proactive Research Approach," IRIS 26, Helsinki School of Economics, Haikko, Finland, 2003.

Selic, B. "The pragmatics of model-driven development," *IEEE Software* (20:5), September-October 2003, pp 19-25.

Shaw, M.L.G., and Gaines, B.R. "Requirements acquisition," *Software Engineering Journal* (11:3), May 1996, pp 149-165.

Sommerville, I. *Software Engineering*, (6th ed.) Addison-Wesley, 2001.

Sutcliffe, A.G., Maiden, N.A.M., Minocha, S., and Manuel, D. "Supporting Scenario-Based Requirements Engineering," *IEEE Transactions on Software Engineering* (24:12), December 1998, pp 1072-1088.

Tuunanen, T. "A New Perspective on Requirements Elicitation Methods," *JITTA : Journal of Information Technology Theory & Application* (5:3) 2003, pp 45-62.

Tuunanen, T., Peffers, K., Gengler, C., Hui, W., and Virtanen, V. "Developing Feature Sets for Geographically Diverse External End Users: A Call for Value-based Preference Modeling," *JITTA: Journal of Information Technology Theory & Application* (8:2) 2006, p in press.

Tuunanen, T., Peffers, K., and Gengler, C.E. "Wide Audience Requirements Engineering (WARE): a Practical Method and Case Study," Helsinki School of Economics, Helsinki, p. 52.

Tuunanen, T., and Rossi, M. "An Advanced Requirements Elicitation Method and Tool," WITS 2003, Seattle, USA, 2003.

Tuunanen, T., and Rossi, M. "Engineering a Method for Wide Audience Requirements Elicitation and Integrating It to Software Development," 37th Hawaii Int. Conference on System Sciences, IEEE, Big Island, Hawaii, USA, 2004b, p. 10.

van Lamsweerde, A., Darimont, R., and Letier, E. "Managing conflicts in goal-driven requirements engineering," *IEEE Transactions on Software Engineering* (24:11), Nov 1998, pp 908-926.

van Lamsweerde, A., and Letier, E. "Handling obstacles in goal-oriented requirements engineering," *IEEE Transactions on Software Engineering* (26:10), Oct 2000, pp 978-1005.