ISSN 1744-1986

Technical Report N° 2004/18

Managing Requirements for Mobile Learning

Debra T. Haley,
Bashar Nuseibeh,
Helen C. Sharp,
Josie T. Taylor

26th May 2004

Department of Computing
Faculty of Mathematics and Computing
The Open University
Walton Hall,
Milton Keynes
MK7 6AA
United Kingdom

http://computing.open.ac.uk

Managing Requirements for Mobile Learning

*+D. T. Haley, *B. Nuseibeh, *H. C. Sharp, +J. T. Taylor
+Institute of Educational Technology, The Open University
**Department of Computing, The Open University*
Walton Hall, Milton Keynes, UK MK7 7AA
Email: D.T.Haley [at] open.ac.uk

Abstract

This paper reports on the experience of eliciting and managing requirements on a large European-based multinational project, whose purpose is to create a system to support learning using mobile technology. The project used the socio-cognitive engineering methodology for human-centered design [1] and the Volere shell and template [2] to document requirements.

We provide details about the project below, describe the Volere tools, and explain how and why we used a flexible categorization scheme to manage the requirements. Finally, we discuss three lessons learned: (1) provide a flexible mechanism for organizing requirements, (2) plan ahead for the RE process, and (3) do not forget the waiting room.

1. Introduction

Managing multiple stakeholders in a project is a well known problem. A project with many stakeholders who are also generating new requirements, commenting on existing requirements, and accessing requirements for design and implementation activities cause requirements engineers further difficulties. These include ensuring a consistent capture mechanism for requirements, reconciling differences of opinion over what constitutes a requirement, and providing suitable access mechanisms to support the multitude of goals, backgrounds, and pre-existing biases of the many partners. We describe here MOBIlearn, a large European research project for mobile learning that faced these problems.

This paper reports on how MOBIlearn managed requirements and focuses on the surprisingly difficult problem of how to organize and categorize the requirements. The conundrum was how to design a scheme for organizing requirements that would satisfy the needs of most of the diverse stakeholders when it seemed impossible to do so. We found that while it was easy to "file" requirements using one of the 27 categories defined by the Volere template [2], it was difficult to retrieve them later. A search of the literature failed to reveal any directly relevant ideas. In fact, the literature supported our experience that classifying and categorizing are non-trivial tasks.

It was necessary to find a suitable classification scheme to manage the hundreds of MOBIlearn requirements. As the number of requirements became larger, it became harder to find a particular one. The effort expended eliciting and documenting requirements is wasted if the requirements cannot be located when needed.

The fact that different team members have different needs complicates the process of establishing useful categories. For example, Anne, a researcher who wanted to compare requirements gathered by different techniques would need a category called *elicitation method*. Bob, another researcher, wanting to examine all of the requirements he elicited on a certain day would need a category called *date elicited*. A category called *hardware platform* would be useful for Carol, a developer charged with implementing all requirements for a laptop.

A few questions should make the problem clear. Assume Requirement A for a laptop was elicited on 15 July using a questionnaire. Should it be filed under questionnaire in a drawer marked elicitation method? Should it be filed under 15 July in a drawer marked

date elicited? Or should it be filed under laptop in a drawer marked hardware platform? Since one copy of a requirement cannot be filed in three places, should we make copies? Does one categorization criterion exist that would allow Anne, Bob, and Carol to locate the requirement? If we implement a particular categorization scheme in January, what happens in February when Dennis requests a completely different view of the data?

We found a workable solution to the problem of determining appropriate categories by deciding *not* to decide. The rest of the paper explains this paradoxical choice.

Section 2 presents the MOBIlearn project. Section 3 explains how the project elicited requirements. Section 4 gives an overview of the Volere tools MOBIlearn used to document requirements and how they were adapted to suit the needs of the project. Section 5 discusses the difficulties in categorizing and describes the database we designed to deal with the categorization problem. Section 6 offers lessons learned. Although this project involves mobile learning, the lessons are applicable to any project that has more than a few requirements. Sections 7 and 8 conclude with a summary and a discussion of further work required.

2. Background

This section discusses MOBIlearn, the three learning domains covered by the project, and the development methodology used by the project. Further information is available on the MOBIlearn website [3].

2.1. Details of the project

MOBIlearn is a large, multinational, European-funded research project involving more than 15 organizations from seven European countries plus one Middle Eastern country, whose purpose is to provide a framework for improved learning using mobile technology. Pedagogical research about the effectiveness and usefulness of mobile learning is an important part of the work; as final deliverables, MOBIlearn will produce a set of requirements, pedagogical guidelines, best practices, and an architectural framework to support mobile learning. The system produced will be a prototype, or instantiation, of a state-of-the-art mobile learning environment validated by the research.

MOBIlearn has several complications, not least of which is the large number of team members who are geographically, linguistically, and professionally diverse. These international, multilingual partners from industry and academia contribute different perspectives and expertise, which contributes to varying preferences for looking at requirements.

In addition to the diverse team, another complication of MOBIlearn is the tension inherent in the two types of project deliverables: reports that present research results and a working prototype. A desirable and necessary output of the research includes requirements that cannot be implemented during MOBIlearn due either to unavailable technology or insufficient resources allocated to the project. Requirements engineers must resist the pressure from the developers to discard requirements that will not be implemented in *this* project but could serve as a roadmap for *future* projects.

The scope of the project adds additional challenges. MOBIlearn is concentrating on three types of mobile devices and three learning domains. The system delivered must work on laptops, smart phones, and PDAs. This multiplicity of hardware platforms provides issues of varying screen size, processing power, and existing software infrastructure. The three learning domains provide the challenges of differing characteristics, needs, and types of learners.

2.2. Learning domains

MOBIlearn has chosen to concentrate on three learning domains, or strands. Each of the strands correlates to a type of learning. The museum strand typifies informal learning, the MBA strand concerns formal learning, and the health strand addresses learning on the job.

2.2.1 Museum strand. The first strand concerns museum visitors, the most varied types of learners of our three strands. They range from children on school outings to adults with a passion for art. They may have been forced to go by a teacher or spouse, or they may be eager to experience as much as possible. Their motivations might range from learning just enough to pass the test to seeking enrichment and enlightenment.

2.2.2 Business strand. The second strand serves business students, both beginning first year students and MBA students. The MBA students are usually highly motivated, extremely busy, and want value for their time and money. Beginning students may be less

demanding and need more support in their first days including help getting around the campus and meeting other students. MOBIlearn supports both of these student types by providing an Orientation Game for the beginners and a collaborative support system for the MBAs.

2.2.3 Health strand. The health strand supports the need for periodic training and the updating of skills of first aid workers. First aiders need occasional reinforcement because a period of time may elapse between their initial training and an event requiring a specific piece of knowledge. One way MOBIlearn supports the first aiders' training needs is to run a competitive quiz involving an informal re-enactment of a potential emergency situation. The quiz is a fun and safe way for first aiders to practice and update their skills.

2.3. Socio-cognitive engineering

MOBIlearn used the socio-cognitive engineering design methodology [1], which strives to design computer systems that are both usable and useful. A key component of the method is consultation with users from the earliest design stages. As in the twin peaks model [4] and the spiral methodology [5], it emphasizes repeated cycles of requirements analysis, development, and testing.

Socio-cognitive engineering asserts that by analyzing the interaction between computers and people, it can produce human-centered technology. The first step is to state the purpose of the system being developed. Next, analysts conduct field studies to learn how users perform their tasks. Concurrently, researchers approach the tasks from a more theoretical perspective, considering both social and cognitive issues. From this work, analysts create a task model detailing the interactions among people, tools, technologies, and contexts.

3. Requirements elicitation in MOBIlearn

MOBIlearn, using the precepts of socio-cognitive engineering [1], started its design process by stating the overall purpose of the system – to support learners using mobile technology. It then conducted research (future technology workshops, questionnaires, observation, and interviews) to learn more about the tasks, types of learners, and interactions between learners and technology. From these studies came scenarios that encapsulated the knowledge gained in a format that is easy to read and understand.

Our scenarios serve as the task model created in the socio-cognitive engineering methodology. We extracted requirements from these detailed scenarios. We fed back the experience of developers and testers to refine our requirements and enhance our scenarios.

Sections 3.1, 3.2, and 3.3 describe our elicitation techniques; section 3.4 discusses some problems we had with them.

3.1. Future technology workshops

The members of the health strand conducted Future Technology Workshops (FTW) with first aiders to elicit requirements for the training and updating of their skills. Researchers originally developed the FTW technique for use with children although it can be adapted for adults [6]. FTWs aim to explore the relationships between current and future technology, now and in the future.

Participants attend several separate sessions; each one focuses on one of the four interactions shown in Table 1 (copied with permission from [6]) that FTW was designed to study. Participants start with interaction 4 to encourage complete creativity. They then cycle through interactions 3, 1, 2, and back to 4. Each session refines and solidifies the ideas elicited from the participants.

Table 1. Interactions between activities and technology, now and in the future.

	CURRENT TECHNO-LOGY	**FUTURE TECHNO-LOGY**
CURRENT ACTIVITY	1. everyday technology-mediated activity	2. Familiar activities supported by new technology
FUTURE ACTIVITY	3. New activities that current technology might support	4. New activities with new technologies

This elicitation technique produced many requirements, both relating to the goal of MOBIlearn (support mobile learners) and beyond its scope. For example, some first-aiders wanted a feature that would immobilize an injured person. Inventing such a

technology is not part of the mandate of the project. Even so, we did not want to lose track of any requirement and needed to use a category for documenting even those requirements that we knew would not be implemented.

3.2. Questionnaires

The museum strand used questionnaires to gather data from prospective visitors to the Uffizi Gallery in Florence. One interesting result from the questionnaire, although having nothing to do with learning, indicates a desire to have the ability to make and pay for reservations from a mobile device rather than the current system of waiting in line for hours. This requirement is an example of what unstructured questions on a questionnaire (or other unstructured methods) can produce: requirements that have nothing to do with the intended goal or purpose of the proposed system. We needed a way to categorize these unexpected requirements.

3.3. Observation and interviews

The MBA strand observed and interviewed students and educators to discover requirements. The requirements ranged from making and sharing annotations of PowerPoint slides to remote control of a classroom projector. Many of the requirements are implemented on widely available PDAs. We needed a way to separate requirements that implemented such familiar functionality from requirements that documented more innovative, MOBIlearn-specific features.

3.4. Problems with elicitation techniques

Maiden and Rugg [7] claim that requirements engineers should use a range of elicitation techniques. They suggest that scenario analysis, prototyping and RAD are the best techniques for new systems such as MOBIlearn. MOBIlearn used two of these techniques: scenario analysis and prototyping, in addition to the methods listed above. However, contrary to the advice in [7], we did not use a range of techniques for each strand because of time constraints. Our results might have been more complete if we had followed Maiden and Rugg's advice [7].

Possibly the reason that our questionnaire for the museum strand did not elicit particularly novel requirements is that, although collecting and analyzing large amounts of data is easy, the structured format of a questionnaire inhibited creative thinking. Running a FTW might have been more productive.

Perhaps because we used a different technique for each strand, we obtained a large number of overlapping requirements. For example, users in each strand wanted to access a database. The first aiders wanted their mobile devices to be able to access a database of health information, the MBA students wanted to be able to access a database of class notes, and the museum visitors wanted to access a database containing facts about the art they were viewing. Some questions arose: should we keep three versions of a single requirement to document that it came from three sources? But since we are developing one product (not three), would it make more sense to keep one version with accompanying information giving the sources?

Another problem arose because each of the three techniques was used by a different set of requirements engineers, not all of whom used the Volere shell. They had differing views of what requirements should look like. One team delivered requirements that sounded like goals, e.g., "support the learner in everyday situations" but did not specify what the mobile system should do to fulfill this requirement. What was needed was a hierarchy of requirements; the top level would specify the general goal and lower levels would provide specific functionality.

These are just two examples from our MOBIlearn experience that point out the need for a good classification system.

4. Requirements management in MOBIlearn

MOBIlearn adopted part of Robertson and Robertson's Volere process [8] for requirements elicitation and management: the Volere shell and template. (Volere is the Italian verb for to wish or to want.) These tools are part of a method to ensure that requirements engineers collect accurate and comprehensive requirements from all relevant stakeholders of a project. Although the Robertsons incorporate the shell and template as part of their method, each can stand alone as an independent requirements documentation tool, which is how MOBIlearn used them.

The terms shell and template are confusing to some people. The Volere template is meant to be a guide for

writing a complete requirements specification including all of the individual requirements. The Volere shell is also a template, but for a single requirement. Perhaps in response to this confusion, the Robertsons sometimes use the term *atomic requirement template* to refer to the shell. [9].

4.1. Volere shell

The Volere shell is simply a form containing fields relevant to a single requirement. See Figure 1 for the complete shell. The fields need no explanation except perhaps fit criterion, which is a measurement that allows a tester to verify if the system satisfies the requirement.

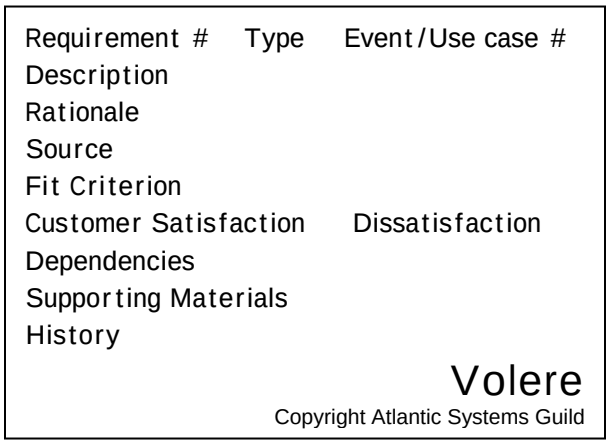

Figure 1. Volere requirement shell

Nothing in the Volere process implies the medium for storing shells. In fact, some people call them "snow cards" to reflect their origin as 5 x 7 inch white paper index cards [2]. The paper form is convenient when brainstorming requirements with stakeholders or when gathering requirements in a busy or mobile environment. The informality of the paper cards encourages stakeholders to jot down their ideas. Their portability allows requirements engineers to use them whenever and wherever they are needed. Paper index cards quickly become unwieldy as the number of requirements grew. We decided early in the project to use a computerized database of Volere shells to document the MOBIlearn requirements.

4.2. Volere template

The Volere template is like a filing cabinet for storing requirements written on Volere shells. It comprises 27 categories covering most, if not all, of the types of requirements. Each of the categories is like a drawer in the filing cabinet. Table 2 shows the complete template.

The purpose of the template is twofold: it is a template, or guide, for writing the final requirements documents and it serves as a checklist for the project [2]. If the requirements engineers carefully consider each of the 27 categories of the template, they can be reasonably sure of not omitting critical requirements.

Table 2. Volere template

PROJECT DRIVERS
The Purpose of the Product
Client, Customer, and other Stakeholders
Users of the Product
PROJECT CONSTRAINTS
Mandated Constraints
Naming Conventions and Definitions
Relevant Facts and Assumptions
FUNCTIONAL REQUIREMENTS
The Scope of the Work
The Scope of the Product
Functional and Data Requirements
NON-FUNCTIONAL REQUIREMENTS
Look and Feel Requirements
Usability Requirements
Performance Requirements
Operational Requirements
Maintainability and Portability Requirements
Security Requirements
Cultural and Political Requirements
Legal Requirements
PROJECT ISSUES
Open Issues
Off-the Shelf Solutions
New Problems
Tasks
Cutover
Risks
Costs
User Documentation and Training
Waiting room
Ideas for Solutions

We now describe our experience using the Volere tools to manage requirements, why it became necessary to categorize requirements and the problems

we had in doing so, and finally, how we dealt with the conundrum of how to categorize our requirements.

4.3. Advantages of the Volere shell

The Volere shell provides a form for documenting requirements. It ensures consistency and compatibility in a clear and simple format. It affords traceability, both in where a requirement originates and where it appears in later documentation such as use cases. When used correctly and filled out completely, it encourages the originator of a requirement to study the detail of the requirement, justify the requirement, consider how it relates to other requirements, and how a tester can evaluate or test the requirement.

Two fields on the shell serve to organize information. The field labeled Supporting Materials provides a place to record pointers to other documents that correlate to a particular requirement. As a project progresses, this field is a handy way to avoid losing information. The field labeled Event/Use case # is another useful place to record information. In fact, S. Robertson, in a private email message to the first author, suggested that the Event/Use case # is the primary way to organize a set of requirements. Unfortunately, we were unable to correlate use case numbers with requirements until well into the project because the requirements were created months before any use cases were written.

4.4. Adapting the Volere shell

Because we found that the "out of the box" Volere shell did not completely satisfy our needs, we added two fields: status and title. The status field provides an easy search key while researchers and developers are creating and polishing the requirements. The title gives a short description that is useful for quick review of all the requirements. These simple additions helped enormously to locate particular requirements. Indeed, we could have added many more fields to the shell in an attempt to enable easier retrieval. Instead, we tried to adapt the Volere template for reasons discussed in the following sections.

The next improvement was more substantial. The Volere shells on 3x5 cards were not sufficient in themselves because the number of requirements quickly grew too large to manage without some kind of computerized requirements management system. Some of the many existing products that are available are: Doors (http://www.telelogic.com/), Calibre (http://www.calibresys.com/index.cfm), and the Rational product suite (http://www.rational.com/). Comparative reviews of these and other products can be found on [9], [10], and [11].

Because we did not have the budget to purchase a suitable existing tool, we created an in-house database system. The MOBIlearn database was designed to offer online access with easy search and retrieval by our international team members. The initial version was based on the Volere template's 27 types of requirements. We found that although it was easy to store a requirement and assign one of the 27 Volere categories, it was not easy to retrieve a particular requirement. The next section explains why.

4.5. Adapting the Volere template

The Volere template is similar to a filing cabinet with 27 very useful and relevant categories of requirements providing an organizing principle for a requirements database. We "filed" each MOBIlearn requirement in one of the 27 "drawers". However, the drawer for functional and data requirements became overstuffed.

We were troubled to discover that even 27 categories were not enough to provide useful search keys. We found that about 66% of our requirements were in one category – functional and data requirements. We needed a better way to organize functional and data requirements than lumping them in one category.

The first proposed change to the Volere template to make it more useful as a model for organizing requirements in the database was to split functional and data requirements into separate categories. This change was not sufficient because there were still about 64% of the requirements in the single category of functional requirements. We needed a very large "drawer" in our "filing cabinet" for functional requirements necessitating tedious one-by-one searching to locate the one we wanted. This discovery led to an attempt to sub-categorize functional requirements as a means to improve the organizational structure of the requirements database. It was this attempt that revealed how difficult categorization is.

5. Sub-categorizing functional requirements

We tried several techniques to sub-categorize requirements. First, we used the "armchair" method [12], that is, we sat and thought about what made the

most sense to us. After realizing that many possible organizing criteria exist, we reviewed the literature but identified no one who had solved the problem of deciding on just one method of classification. Next, we tried the card-sorting method. Finally, we decided not to decide, which resulted in our redesigned database. The following sections discuss these points in greater detail.

5.1. The difficulty of choosing categories

In order to provide easier access to the individual requirements in the database, we needed to break down the category of functional requirements to sub-categories, which at first glance, seemed an easy task. On second and further glances, it became clear that it was not a trivial problem. In fact, the literature contains numerous accounts of the difficulty of categorization. This section presents relevant research from four sources.

Dumais and Landauer [13] describe two kinds of problems in categorization for menu-based information retrieval systems. Their research applies to the MOBIlearn database because it is a menu-driven, information storage and retrieval system. First, they identify the difficulty of selecting a category name that can be agreed upon and understood by most users. Secondly, the categories that most people use are blurry and not mutually exclusive. Our experience supports their findings.

Bowker and Star [14] make the point that designers of classification systems from different disciplines will have different needs; this results in classification systems where just one point of view is legitimized and others are ignored [14:5]. They claim that it is impossible for designers to know in advance what information will be relevant in the future [14:116]. They further claim that categorizing for information systems is particularly difficult because designers must consider both "the hard technical problems of storage and retrieval with the hard interactional problems of querying and organizing" [14:7]. Again, our experience supports their findings.

Rugg and McGeorge (implicitly acknowledging the difficulty of categorizing) provide a comprehensive tutorial [15] on card-sorts as a knowledge elicitation technique. Card-sorts lead to the discovery of categories and criteria for categorization. Rugg and McGeorge describe the results of giving people eleven pictures of computers and asking them to categorize the pictures using different categorizing criteria. Among the five criteria chosen were type, which produced the categories PCs, laptops, and small portables; storage space, which produced hard disks, floppies, and not sure; and color, which produced black/dark gray and beige/creamy colors. The fact that the subjects came up with five separate categorization criteria for the pictures of computers, a reasonably familiar object, supports our difficulty in devising a single criterion-based system to sub-categorize functional requirements.

Prieto-Diaz [16] has been investigating the problem of categorizing for about 20 years [17] (as cited in [18]). He uses faceted lists in his classification scheme [18]. His work involves software component reuse but many of his findings apply to requirements engineering as well. For example, he states that any classification scheme must allow for continuing growth in the items being classified [18], a condition that certainly applies to requirements engineering.

5.2. Attempting to use the card-sorting technique

This section describes our attempt to create sub-categories for functional requirements using card-sorts and why it failed. (See [15] for a comprehensive tutorial on card-sorting.)

We chose card-sorting because, while we were trying to select the appropriate system to categorize our functional requirements, serendipitously, three of our Open University colleagues reported on their successful use of the card-sorting technique to categorize critical incidents [19]. This technique seemed a promising method to pursue since its main output is the categories and criteria identified by the sorting participants [20], which was exactly what we were trying to do with our functional requirements.

Unfortunately, the card-sorting technique failed because of the large number of functional requirements we had identified. According to [21], card sorts involving more than 20 items become increasingly hard to manage. Not only would sorting the approximately 250 MOBIlearn functional requirements be unwieldy, it would be very difficult to recruit subjects willing to undertake the huge sorting activity involved.

Even if we had been able to perform card-sorts, the results would be difficult to use in the requirements database. Recall that the main reason for establishing sub-categories for functional requirements was to allow easier searching and organizing of the database. Card-sorting requirements by experienced computer

professionals could result in many different criteria for categorizing [21]. Even the simple card-sort described in [15] came up with five ways of categorization. Rugg and McGeorge [15] claim that "Experts are expert largely because they have a more extensive and sophisticated categorization than non-experts." As users gain more experience working with requirements, it seems clear they will refine and expand any existing categories.

5.3. Addressing the problem of too many categories

Thinking back to our filing cabinet analogy, we see that we would need a filing cabinet for each categorization criterion and copies of each requirement to file in each cabinet. For example, one day a requirements engineer might want to see all of the requirements that arose from a particular elicitation workshop. Another day, she might want to see all of the requirements relating to the museum strand. To support her varied points of view, we would need one filing cabinet with requirements filed according to source and another filing cabinet with requirements filed according to whether they concern the museum, MBA, or health strand. Even if we decided to use such a cumbersome storage system, we would need to install a new filing cabinet and re-file each requirement every time a stakeholder requested a new organization of the requirements.

These problems led to the conclusion that any database system used to manage requirements must provide a flexible view of the data. The next section describes the improved MOBIlearn database.

5.4. The MOBIlearn requirements database

The main innovation in the requirements database is a feature that allows flexible, non-static, and ad-hoc categorizations. Any user (with appropriate permission) can add a new categorization criterion at any time. Thus, the database can be modified easily to reflect new decisions, new information, and experience gained.

For example, at the beginning of the project we might have set up categories using the criteria of Strands (health, MBA, and museum) and of Work Packages (e.g., learning content, mobile media delivery, context awareness). These two criteria would reflect our belief that users would wish to query the database based on which requirements pertained to a particular strand or work package. Strand and Work Package information would be added to all of the requirements in the database. After more experience, we might discover a need to find all requirements pertaining to a particular service. Then, service information would be added to all of the requirements. Later on, managers may find that looking at requirements based on Work Package does not suit their needs because a particular Work Package may have members from different countries as well as different organizations. Managers could add a new criterion, location, to the database.

The disadvantage to this method is the need to add an additional piece of data to every requirement when a new categorization scheme is adopted. The advantage is that, with a well designed database, the process of adding data is straightforward, if time consuming.

This simple idea of providing the ability to modify the categorization criteria of the requirements database enabled it to meet the needs of the many different project members throughout the life of the project.

6. Lessons learned

This section presents the lessons learned on the MOBIlearn project. Although MOBIlearn focused on support for learners on the move, the lessons are applicable to a wide range of projects.

6.1. Lesson 1: Provide a flexible mechanism for organizing requirements

We found various problems that prevented us from using a static, predefined categorization system for retrieving our requirements:

- People have varying points of view and want to examine the requirements from different aspects.
- These desires change over time and during different stages of the project.
- People don't always know what they want until they are deeply involved in the project.

The negative impact from these problems can be lessened by using a storage system for requirements that allows ad hoc updating of categorizing criteria. As the number of requirements grows, it becomes necessary to give project members a more personalized view of the requirements.

6.2. Lesson 2: Plan ahead for the RE process

Our experience of managing requirements on the MOBIlearn project suggests that the RE process has requirements itself. If a requirements tool is not in place early in the project, practitioners will create their own, lightweight tools. The MOBIlearn database was not available until midway through the project resulting in duplication of effort. The first version of the database provided a view of the data organized according to the Volere template. One team member developed a spreadsheet to categorize requirements according to his needs, which were not satisfied by Volere. Another member created a sophisticated word-processing tool to organize information in yet a different format. These three tools, while not equivalent, contained a substantial overlap of information. This duplication of data results in duplicated effort in keeping the information up-to-date as well as increases the chance for errors, omissions, and conflicting data. By reducing wasteful effort, early selection or development of a requirements tool and commitment to using it can help keep projects on schedule.

6.3. Lesson 3: Do not forget the waiting room

Various tensions become apparent on a project like MOBIlearn with widely diverse stakeholders comprising both research-oriented academics and product-oriented industry representatives. In particular, we have noted an increasing tension between the researchers' desire to create an architectural framework and list of requirements for *future* implementations and the practitioners' desire to produce a functioning product *now*. The tension is natural because MOBIlearn's sponsor expects both types of results.

Practitioners want to limit the requirements to what they are able to deliver. The requirements engineers want to deliver a set of requirements for mobile learning regardless of whether they can be implemented with current technology and within the MOBIlearn time and budget constraints.

Requirements engineers can lessen the tension while preventing the loss of requirements that are not able to be implemented by using the category in the Volere template called *waiting room*. If circumstances change, either technological advances or budget constraints, any requirements stored in the waiting room are candidates for implementation.

7. Conclusions

We handled the conundrum of how to categorize the functional requirements after realizing the impossibility of a system that would meet everyone's needs. The MOBIlearn database provides a feature that allows ad hoc creation of new categorization criteria. By deciding not to decide and allowing users to customize the database, we offered users a balance between flexibility and uniformity.

Our experience has reinforced the following points:

- Do not impose a static, predefined scheme for categorizing requirements.
- Take time early in a project to address the requirements of the RE process.
- Use the idea of the waiting room to avoid losing track of interesting, but unable to be implemented, requirements.

8. Further Work

We are continuing to refine requirements. Part of this effort involves recording links to other documents (e.g. UML use cases) as they are written. We collect more requirements during each cycle of testing.

Further validation of our approach and testing of the database for both usability and functionality remains to be done. We need comments from a larger group to ensure the database meets everybody's needs.

Although not in the scope of the MOBIlearn project, a potentially fruitful research effort is to adapt Prieto-Diaz' work in software component reuse to the domain of requirements engineering. He has built a semi-automatic tool [16] that uses faceted lists to create and maintain categories for software component reuse. A tool that accomplishes for requirements engineering what he claims his tool does for component reuse would simplify the task of manually updating each requirement every time a new category is adopted.

9. Acknowledgements

We would like to acknowledge the European Union for financial support through the MOBIlearn project IST-2001-37440 .

10. References

[1] Sharples, M., Jeffery, N., du Boulay, J., Teather, D., Teather, B., & du Boulay, G. "Socio-Cognitive Engineering: A Methodology for the Design of Human-Centred Technology," *European Journal of Operational Research (Elsevier)* 136(2), 2002: pp. 310-323.

[2] Robertson, J., & Robertson, S. Volere Requirement Resources. http://www.volere.co.uk, 2003. 15 September 2003.

[3] MOBIlearn. http://www.mobilearn.org/. 23 January 2003.

[4] Nuseibeh, B. "Weaving Together Requirements and Architectures," *Computer (IEEE Computer Society Press)* 34, March, 2001.

[5] Boehm, B. "A Spiral Model of Software Development and Enhancement," *Computer (IEEE Computer Society Press)* 21(5), May, 1988: pp. 61-72.

[6] Vavoula, G., Sharples, M., & Rudman, P. D. "Developing the 'Future Technology Workshop' Method," In M. Bekker, P. Markopoulos, & M. Kersten-Tsikalkina (eds.), *Proceedings of Interaction Design and Children*. Eindhoven, The Netherlands, 2002.

[7] Maiden, N., & Rugg, G. "ACRE: A Framework for Acquisition of Requirements," *IEEE Software Engineering Journal*, May, 1996: pp. 183-192.

[8] Robertson, S., & Robertson, J. *Mastering the Requirements Process*. Harlow, England: Addison-Wesley, 1999.

[9] The Atlantic Systems Guild. http://www.systemsguild.com/. 23 January 2004.

[10] Robinson, M. S. Requirements Management Tools: A Trade Study. http://www.ess.stsci.edu/fset/projects/Sabbatical/, 2001, 28 March. 23 January 2004.

[11] Young, R. Requirements Tools Trade Study. www.ralphyoung.net, 2002, 11 January. 23 January 2004.

[12] Furnas, G., Landauer, T. K., Gomez, L., & Dumais, S. T. "The Vocabulary Problem in Human-System Communication," *Communications of the ACM* 30(11), 1987: pp. 964-971.

[13] Dumais, S. T., & Landauer, T. K. "Using Examples to Describe Categories," In *Proceedings of CHI'83*. ACM, 1983, pp. 112-115.

[14] Bowker, G. C., & Star, S. L. *Sorting Things Out: Classification and Its Consequences*. Cambridge, Massachusetts: The MIT Press, 1999.

[15] Rugg, G., & McGeorge, P. "The Sorting Techniques: A Tutorial Paper on Card Sorts, Picture Sorts and Item Sorts," *Expert Systems* 14(2), May, 1997: pp. 80-93.

[16] Prieto-Diaz, R. "A Faceted Approach to Building Ontologies," In *IEEE International Conference on Information Reuse and Integration*. IEEE Computer Society Press, 2003, pp. 458-465.

[17] Prieto-Diaz, R. *A Software Classification Scheme*. Ph. D. Dissertation, Department of Information and Computer Science, University of California, Irvine, 1985.

[18] Prieto-Diaz, R. "Implementing Faceted Classification for Software Reuse," In *12th International Conference on Software Engineering*. IEEE Computer Society Press, 1990.

[19] Dawson, L., Minocha, S., & Petre, M. "Exploring the Total Customer Experience in e-Commerce Environments," In *Proceedings of IADIS International Conference E-Society*. Lisbon, Portugal: IADIS 2003, 2003, pp. 945-948.

[20] Bitner, M. J., Booms, B. H., & Tetreault, M. S. "The Service Encounter: Diagnosing Favorable and Unfavorable Incidents," *Journal of Marketing* 54, January, 1990: pp. 71-84.

[21] Upchurch, L., Rugg, G., & Kitchenham, B. "Using Card Sorts to Elicit Web Page Quality Attributes," *IEEE Software*, July/August, 2001: pp. 84-89.