Abstract

In March 2001, two student teams, one in Melbourne, Australia, and one in Dortmund, Germany, started a common software engineering project to develop an electronic management tool for the support of scientific conferences. Both teams had to take part in all key activities of this project. The different deliverables had to be matched periodically in order to ensure the development of a common tool. In doing so, the teams had to deal with typical communication problems caused by different time zones, geographical distribution, language issues, and different approaches to requirements analysis. This paper presents the experiences and the lessons learned during requirements elicitation in the context of a distributed software engineering education project.

Keywords: Distribution, Requirements Analysis, Collaboration, Communication.

1. Introduction

As the development of component-based software systems becomes more and more distributed, software engineering education means not just teaching the concept of components and the related technologies for requirements analysis, design, implementation, and integration into a software system, but also tackling issues such as the interaction between geographically distributed teams. This requires appropriate communication tools and easy data access for all participating teams. A further issue is the need for a common vision of what has to be developed and how it can be collaboratively undertaken. This demands special consideration, in particular for requirements analysis. Suitable requirements engineering methods have to be identified and/or adapted for such a context. The distributed participants need to coordinate their activities and communicate their decisions.

This paper describes experiences relating to requirements elicitation for a current distributed and collaborative software development project, called Chairware. Two geographically distributed student teams in two different time zones, one placed in Melbourne, Australia, the other in Dortmund, Germany, are currently participating in this project. The common task is to develop an electronic conference management tool which assists program committee chairs during the preparatory phases of a scientific conference. This includes activities such as paper submission, assigning of papers to reviewers, and obtaining camera ready copies of all accepted papers. The final system should be easily customisable, making management of conference organization more efficient and streamlined, and allow for an easy submission of scientific papers by authors.¹

This paper is organized as follows: Section 2 illustrates the academic context in which the Chairware project is run, outlines some of the problems we faced in organizing a distributed team project, and gives a brief introduction into the problem area. Section 3 presents the requirements analysis of both teams in further detail, starting with the identification of requirements and ending with the results of the first

¹For further information about the Chairware project, refer to the web-site at http://www.chairware.org.
face-to-face student meeting, where the requirements were merged. The two approaches are discussed and compared in Section 4. Section 5 offers a description of the lessons learned during this project and Section 6 suggests issues for further consideration in distributed development and in education programs.

2. Background of the Chairware project

Two of the authors of this paper had previous experience in organizing international software engineering conferences. Their experience was that there were severe deficiencies in existing electronic conference management tools.

This common background initiated the idea of a joint project and the development of a configurable and adaptable system for conference management support using state-of-the-art component technology. After evaluating various alternatives and performing a brief feasibility study, it was decided to run this project as a joint one-year student project. In particular, it seemed to be an ideal topic for students to enhance their skills in component-based development and to experiment with various implementation technologies. Such a project is difficult enough with a team located together. Developing a tool in such a context would also show them the relevance of engineering approaches in software development and expose them to the increasingly popular approach of developing software in a distributed environment.

2.1 Overview of electronic conference management systems

The purpose of an electronic conference management tool is to assist the program committee chair (PC chair) during the preparatory activities (or phases) of a scientific conference. These phases can briefly be described using the following schema:

- establishing a highly regarded program committee by the PC chair,
- doing the “call for papers”,
- submission of abstracts,
- submission of papers,
- bidding for papers (i.e. giving program committee members the possibility to make a preliminary selection of the papers they would like to review),
- assigning papers to reviewers,
- submission of review reports,
- preparation of the program committee meeting,
- notifying authors of acceptance/rejection of their submissions,
- submission of camera ready copies, and
- setting up a conference web-site with the conference program.

Although these phases are not strictly ordered, there are time dependencies between them, as it is illustrated in Figure 1. For example, $B_3 < R_1$ means that the review phase cannot start before the bidding phase is terminated.

The schema given above is generally followed by conferences in the information technology field, although some of the phases need to be configured for a particular event. However, conferences in other disciplines have a different submission and review
process and, therefore, require a modification of this schema. Unfortunately, most existing systems are not flexible enough and major changes in the process they support cannot be done. The idea of the Chairware project was to develop a system which can cope with a variety of submission processes, and could be adjusted according to the needs of a given conference.

A second aspect of an electronic conference management tool is the distinction between the different roles within the lifecycle of a conference. The roles supervisors considered to be of importance for such a system are (i) program committee chair, (ii) member of the program committee, (iii) reviewer or referee (generally a member of the program committee), (iv) author, (v) system administrator, and (vi) general conference attendee. Most existing systems do not cover these different roles.

2.2 Academic background

For both institutions involved in the Chairware project, a one-year team project is part of the corresponding curriculum in software engineering. This allows students to gain hands-on experience in running software projects and to apply the knowledge acquired throughout their studies [6]. One of the main issues of these projects is that participating students experience all phases of the development process.

For the Melbourne team (nine undergraduates), this project is part of their final year. Prior to this, it is usual for them to undertake practical work in industry for one year. Most of this group of students had a minimum of two years experience in industry. For the Dortmund team (nine students) the project is part of their Diplom studies. While prior industrial experience was no precondition for participation, most of the students work either in industry or as tutors at the university, in parallel to their studies. Altogether, the students’ skills and experiences were not greatly dissimilar at both project sites. It should be noted that the students in Dortmund were able to express their preferences and choose from various proposed projects whereas the students in Melbourne were assigned the topic.

2.3 Project set-up and organization

Problems were anticipated in managing team dynamics and integration, collaboration of distributed team members, and language barriers—all of which have already been discussed elsewhere [1,2,3]. Therefore a face-to-face supervisor meeting was held during the setup period. Further issues covered during this meeting included

- a discussion of the different educational preconditions for the two student teams based on different underlying study programs and individual student skills,
- the preparation of a first coarse-grained project plan, and
- the coordination of different approaches for the requirements analysis.

One of the main problems organizing the Chairware project, in particular the requirements analysis phase, was to coordinate different starting dates in Melbourne and Dortmund. Whereas the Melbourne semester started early in March 2001, the official start for the Dortmund team was not until the third week of April. Nevertheless, it was considered to be essential that both teams undertake some requirements activities because (i) it is one of the most important phases during software development and (ii) this seemed most likely to reach a common and consolidated vision of the project’s purpose. A two week face-to-face meeting at the end of the requirements analysis or at the beginning of the design phase was planned in order to consolidate all identified requirements and to subdivide both design and implementation tasks, respectively. For this purpose, a Melbourne sub-team, consisting of two students and one supervisor, travelled to Dortmund in June 2001. A return visit took place during the integration phase in September 2001.

In addition to different starting dates and preparations concerning communication issues, it was also decided to follow a role-based approach where each student team has its own project manager, quality manager, requirements manager etc. The people in corresponding roles in both sub-teams were responsible for coordinating their different tasks to ensure retention of the common vision. The students were considered to ‘own’ the project, with guidance being offered by the supervisors when needed. Coordination between the two teams was especially important for the requirements phase as the two student teams were intended to identify the requirements in different ways. This issue is addressed in the next section.

3. Requirements analysis

Although all supervisors involved in the Chairware project had previous experience in various activities performed during the organization of an academic conference, none of us had an in-depth understanding of the requirements of all of these activities or knew the problems and limitations of existing tools. Therefore, the supervisors could only help with very specific problems during the requirements phase, and were not able to provide enough input for an in-depth requirements analysis.

In order to circumvent these restrictions and to get in-depth information about the desired functionality for an electronic conference management system, the two project sites used different approaches for requirements collection. The first approach, conducted by a sub-team in Melbourne, focused on interviewing people with experience in organizing conferences and on evaluating questionnaires sent to other interested people. The second approach, used by a sub-team in Dortmund,
focused on the study of existing tools for conference preparation support, in order to get an overview of the features integrated into these tools. The following sections describe these two approaches in further detail and discuss how the resulting requirements have been integrated into a single requirements specification.

3.1 Collecting requirements in Melbourne

As the official start of the project in Melbourne was a few weeks earlier than the start in Dortmund, it was agreed at the set-up meeting that the students in Melbourne should focus on collecting requirements from interviews and questionnaires, as such activities would probably require more time than evaluating tools. The main idea behind this approach was to get as complete an overview as possible of the required features for a conference management tool and to decide after the requirements collection which subset of the requirements actually had to be implemented. We hoped that collecting such broad requirements would give the students a better understanding of the topic area. We also hoped to enhance their motivation, particularly as they were not aware of the specific topic beforehand.

At the start of the project, the students were given a two week period where they could investigate various project-related issues and familiarise themselves with the topic. Based on the previous experience of these students, we expected that this would give them a sufficient background so they could work independently and require only minimal guidance. After this initial period, the students distributed the various project roles amongst themselves and formed a requirements team consisting of six students, which performed the tasks discussed in this section. The students not directly involved in the requirements analysis were in charge of other project-related issues. Due to the lack of a concrete vision about the system to be built, the students did not focus on the (from our point of view) central issues, but gave the interviewee the possibility to talk about their vision and ideas of a conference management tool. This allowed the Melbourne sub-team to get a broad picture of what many people considered to be key features in such a system, but, on the other hand, many important issues related to the submission and review process were not covered. Nevertheless, important requirements for the system were collected during this period.

In parallel with the interviews, the requirements team defined a questionnaire which they intended to send to other interested people. Again, the students set up the questionnaire mainly by themselves, but heavily underestimated the required time as several reviews of the questionnaire were necessary. It was finally sent out early April. Unfortunately, only very few people responded, and not many further insights were gained. However, a lot of information collected during the interviews substantially influenced the structure of the questionnaire: it still covered a broad range of questions, but was already much more focused around the central issues. Therefore, the effort invested in the questionnaire was not in vain as it assisted the students in structuring the requirements.

Early in the project, we suggested to the students to apply a use-case driven approach to collect and structure the requirements, and to use task analysis to assist them. We assumed that using a task analysis tool would help them not only in collecting requirements, but also in designing appropriate user interfaces at a later stage. However, the students decided to dispense with task analysis, mainly because the tools they looked at were either unstable or unsuitable for their needs.

The initial requirements specification document (SRS) was based on an adapted version of a template proposed by Robertson and Robertson [5]. The structuring of the functional requirements was centered around the different types of users of a conference management tool, a description of their associated roles and usage scenarios for each of these roles (refer to Figure 2). In order to enhance readability and give a logical view of the system, the SRS document contained the definition of an initial set of subsystems corresponding to phases in the submission process (similar to the ones described in the next section) and assigned each use case to one of these phases. Furthermore, the SRS document contained various data requirements as well as non-functional requirements (e.g., security, usability) which were not assigned to a specific phase. Various drafts of the documents were produced and thoroughly reviewed on two occasions by both students and, to a lesser extent, supervisors.
3.2.2.3 Submit Abstract

<table>
<thead>
<tr>
<th>User/Actor Name</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>An author attempts to submit an abstract for consideration to a conference. At the time when an author submits an abstract they will also be prompted to choose the topic of their paper.</td>
</tr>
<tr>
<td>Fit Criterion</td>
<td>The system will accept and store the abstract and topic for later retrieval by referees or programme committee members.</td>
</tr>
<tr>
<td>Use Case Scenarios</td>
<td>The description for this scenario describes the normal way that it operates. Scenario model 3.2.2.3.1 illustrates the exception case for this use case.</td>
</tr>
</tbody>
</table>

As a summary we can say that the initial SRS document covered a wide range of precisely specified requirements. However, it only considered the results of the local requirements elicitation process and did not specifically include any of the requirements collected by the Dortmund sub-team.

3.2 Collecting requirements in Dortmund

Although the official start of the project in Dortmund was the middle of April, 2001, the students had started to prepare for the project two months earlier. Every student worked on a presentation which was given in a kick-off meeting early April. These presentations covered a wide range of topics, including UML, Enterprise JavaBeans, application servers, security, role-based software development, and the electronic conference management tool CyberChair [7]. The inspection of CyberChair was the basis of further tool evaluation, as it structured the features by the main phases of a conference preparation and by the participating roles within each phase.

The idea behind structuring features of an electronic conference management tool was to assign every feature the tool offers to a phase of the conference submission process. For applications of this nature, process phases seem to be a very obvious way of structuring tool features because many features belong to exactly one phase. Although at the beginning this was only an assumption, the results of inspecting a first tool made the students confident that this structure would work for other tools as well.

The tool evaluation was based on the phases given in Section 2.3. Although these phases are not strictly ordered, there are time dependencies between some of them, which makes it easy to assign most features to phases. Nonetheless, some features cannot be assigned to one specific phase, especially communication features like email or confidentiality aspects. These features must be listed in separate categories.

An alternative way of classifying features of an electronic conference management tool is to distinguish functionality for the different roles within the submission process. The roles considered for the tool analysis were (i) the program committee chair, (ii) the reviewer, (iii) the author, and (iv) the system administrator. With these two orthogonal possibilities of classifying the results of tool evaluation, students had the opportunity to distribute the work of tool evaluation and to compare the resulting tables.

Three students assembled all information gathered during the tool analysis into a homogeneous requirements list document. The structure defined for tool evaluation could mainly be reused, and only some minor changes in the phases were done, especially after getting input from the team in Melbourne. The following sections were added:

- a global section for requirements that belong to many other phases,
- a setup section for collecting requirements that belong to customisation,
- a discussion section for online discussions before the program committee meeting, and
- a security section which mainly covers confidentiality and integrity issues.

The time dependencies between different phases are illustrated in Figure 1. Three of the new sections, the global section, the setup section, and the security section do not fit into this diagram because the requirements of these sections are not time-dependent. Adding these new sections to the existing structure caused a substantial overhead in preparing the requirements list document.

The reader should note that in order to collect the requirements related to security issues, the students who edited the requirements list were assisted by two other students. This approach was used to make sure that non-functional security requirements were not skipped.
As a summary it can be said that a given structure supported the students in collecting requirements from different tools and merging them into a requirements list. The main difficulties occurred with parts of the given structure that had to be revised, because the initial view, based on evaluating one tool, was too limited.

### 3.3 Merging requirements

As described in the previous two sections, the distributed collection of requirements resulted in two documents with considerable differences in structure, precision, and completeness (refer to Figure 2 and 3). Although many common requirements were identified, the essential contents of the two documents diverged considerably. It was thus necessary to merge these two documents into a single document for the successful completion of the project.

It was originally planned to merge the two requirements documents before the visit of a Melbourne sub-team to Dortmund and only spend a short time on this topic during the visit. Unfortunately, this was wishful thinking as the requirements sub-teams of both project sites considered their document to be the “right” one and no real progress in merging was made. To prevent putting the project at a too great a risk, it was decided to postpone the finalization of the requirements to the start of the visit and to base initial design and prototyping activities on the requirements list document elaborated by the Dortmund sub-team.

Prior to the visit to Dortmund, both project sites tried to identify the main reasons behind the problems encountered in the merging. Ultimately, we found that the real problem was not the different document structures and contents—that was merely the symptom of the problem. The lack of adequate communication between the two project sites and the language barriers were identified as the two main factors. Hence, we paid particular attention to these issues at the beginning of the visit. As a result, the merging of the requirements made considerable progress and a common working document was elaborated within the first day of the visit.

The situation the project ran into is probably best described by the following quote, extracted from the report the two Melbourne students wrote after the visit in Dortmund:

> “Along with the members of the Dortmund requirements team, we spent a long session working on the requirements. Eventually, we came to a common agreement on those requirements that would need to be implemented. Interestingly, language barriers and communication issues had stopped us from coming to an agreement before traveling to Dortmund. As an example, one of the major requirements we had identified in Melbourne was the generation of reports while the Dortmund team had identified the requirement of generating overviews. They felt that our requirement of generating reports was unnecessary. After some

---

**Figure 3: Extract from Dortmund Requirements List.**
discussion, it was understood that we were talking about the same thing. We were all very relieved to realise that this important requirement was still to be included in the system and it showed the advantage of working together as a group rather than from distant locations."

The common working document was further improved during the next few days of the visit, remaining ambiguities were clarified, and a few missing requirements were added. All other activities during the visit were then based on this common document.

Subsequently, the joint version of the requirements specification document (again based on the template proposed by Robertson and Robertson [5]) was completed by a Melbourne sub-team and was subjected to a final review.

4. Comparison of approaches

In Section 3, two completely different approaches of collecting requirements are discussed. The students in Melbourne did a wide range analysis of requirements based on questionnaires and interviews whereas students in Dortmund analysed existing tools with a focus on requirements actually to be implemented. In both cases, important input for a final requirements specification was collected. In order to understand why the results of these approaches are different and how these results contributed to the final requirements specification, we further compare the two approaches in this section.

4.1 Focused analysis vs. wide range analysis

Right from the beginning of the project, students in Dortmund collected requirements with a strict focus on the scope of the project. Requirements that seemed to go beyond this scope were not discussed at all. This approach allowed the students to describe many details of their view of the required product in a reasonable amount of time.

The students in Melbourne took benefit from their earlier start and worked on a more complete requirements specification document with a broader view of the scope of the product. The intention was to collect all kinds of requirements that are important for organizing conferences and to specify them using a variety of techniques. Of course, the intention was not to implement all these requirements in the resulting system, but to decide after the requirements collection which subset to implement. However, this detailed work helped in better understanding how an electronic conference management tool could be integrated with other tools. For example, requirements concerning internationalisation and customisation could not be understood without this broader view.

4.2 Tool evaluation vs. questionnaires and interviews

Evaluating existing tools enabled the students in Dortmund to elicit known requirements. Owing to the imposed structure of the tool evaluation, all details within the bounds of this structure were included. This activity was important as it allowed the students to cover functionality of existing tools within a defined scope of usage. This approach of collecting requirements has the advantage that the functionality required for a conference management tool can be specified very precisely. On the other hand, no real innovation can occur as the point of view is very limited.

The students in Melbourne tried to overcome this limitation by interviewing people with experience in organizing conferences and by evaluating questionnaires sent to people working outside information technology. This approach was motivated by the assumption that an evaluation of questionnaires and interviews would mainly result in functionality that is missing from existing tools, but not reveal features already incorporated into these tools. The results of these activities revealed that this assumption was correct and that people have a tendency to focus on features they are missing in the products they have been using. Therefore, collecting requirements using this approach allowed the students to be innovative in the development of their product. On the other hand, no questionnaire or interview gave an input that was as structured or as complete as the results of the tool evaluation.

We come to the conclusion that the two approaches complemented each other quite well and that both sub-teams contributed to the requirements specification equally and in an important way, even if they did not start working at the same time. However, poor communication seemed to cause problems with incompatible requirements. We further elaborate the main lessons we have learned during this project in the following section.

5. Lessons learned

The Chairware project was resourced by two teams of nine students half a world apart, but with the same basic objective—to work together to produce one quality system. The geographic separation had implications in terms of project commencement times, project completion times, timezone, language, and culture. The additional academic objective, where both sub-teams were to experience all phases of the software development, placed an additional constraint on the way in which the project was run.

The idea of having the two groups focus on requirements in different ways suited our purposes. It meant that each group could concentrate on one perspective without overlap, and their ideas could then
be shared. However, this required the two groups to have set up good communication channels, and this took some time to happen.

By far the greatest influence on the progress of the project during the requirements phase was communication. Part of the problem could be said to have been cultural, with some students reporting they “found it difficult to know how to write email messages”. That is, they were aware of small cultural differences, and were uncertain of the level of formality needed for a response. They therefore delayed replying to requests instead of responding promptly. Each delay raised doubts as to the reception of their email, and consequently reinforced a feeling of “us versus them”, which was not helpful. After the visit of the Melbourne team to Dortmund, the attitude dramatically improved, and responses were much more prompt. Similar improvements in progress have been noted in large industrial projects in Europe when members of distributed teams meet face-to-face [4].

Having the Melbourne team use surveys and interviews to elicit requirements meant that they were exposed to a very wide range of different perspectives on what was required for a tailorable product. This led to their version of the Software Requirements Specification supporting quite diverse functionality, but also missing some of the key ones. In retrospect, it is apparent that these methods of elicitation are likely to draw comments about the deficiencies of existing systems, at least as much as statements about desired functionality. The Dortmund version, on the other hand, was relatively tightly focused for project outcome reasons, and ignored much of the flexibility the Melbourne students had tried to build in. Thus considerable work was necessary to merge the two versions. In the interests of time, the project was scoped to a relatively narrow range of requirements. The other requirements are being considered as possible extensions to the existing system and, therefore, are being taken into account in the design.

From the academic perspective, we had hoped that the students would have learned that the two different methods of requirements elicitation could complement each other in a useful way. However, the need to restrict the implemented system to a subset of the requirements has sent a message to some students that it was a waste of time to consider widely what a conference management system should do.

The merging of the requirements provided another valuable lesson for the students. The styles in which the two versions had been written were vastly different. The Melbourne contingent had focused on use cases, and many of their requirements were couched in this manner. The Dortmund students had written theirs into a requirements list using natural language. It also became clear some difficulties were caused by the fact that task analysis had been neglected. Consequently it was a difficult task to combine the two, and caused further delays in producing a suitable working document. The situation definitely underlined the need for a common document standard to be available very early in the project.

Would a dual approach to gathering requirements be useful in a distributed industrial setting? We would probably not recommend the process as it was carried out in this project. From the Dortmund perspective, they had little idea of the variety of ways in which such a conference management system might work. The Melbourne students, on the other hand, were exposed to relatively little of the basic requirements for organizing a conference and, therefore, missed important requirements. User task analysis was necessary, but was only considered in an ineffectual way. The parallel activities did not speed up the elicitation process, as there was a lack of knowledge on either side, and no real context to assist in seeing the total picture. Reconciling the two versions added another overhead.

In the requirements process used here, effectively the Melbourne students made their enquiries, and then received the information from the Dortmund students later. Therefore, it is reasonable to ask whether the requirements process could have been shortened by doing these activities in the reverse order. We believe that this may have sped up the process, in that the Melbourne students did not have a firm foundation on which to build their understanding of the role of systems like Chairware. This is likely to have implications for industry if the requirements process is to be distributed. We suggest that a task-analysis focus should come first, along with a study of relevant extant systems. The focus could then shift to deficiencies in existing systems.

This paper reports our experiences in dealing with two groups of students trying to elicit and specify system requirements in a distributed manner. The experience clearly indicated areas for development of curriculum, especially in seeing that task analysis is accorded its rightful place in eliciting requirements. While the Melbourne students had learned about task analysis in their Human Computer Interaction subject, they did not seem to remember its relevance after a gap of two years. Further, there is room for teaching about other methods for requirements elicitation and analysis, such as goal-oriented approaches [8].

Our experience in this project has been instructive in highlighting the need for an early focus on communications and an early commitment to product standards. There is also a need to minimise any uncertainties in the cultural contexts. Despite the difficulties encountered during the requirements phase, the students produced a quality software requirements specification, an achievement neither group was likely to have accomplished alone.
6. Conclusions and future work

In summary, we observed in the student environment that:

- When distributed teams undertake to split the process of requirements elicitation, both teams need to start with the same perspective of the scope of the product, or merging of the requirements becomes an opportunity for dispute.
- Where distributed elicitation is used, the ordering of the requirements activities could possibly affect the duration of the requirements phase. We propose that an appropriate ordering of activities is task analysis, then extant systems analysis, and then interviews.
- Working with people you have never met raises cultural issues about how email and chat should be phrased, and deters prompt communication. This is exacerbated by actual, language-based, cultural differences.
- It is imperative to have an early definition of the standards for the document, or merging requirements from differing styles is unnecessarily difficult, and delays progress.
- User task analysis proved not to be a disposable activity, as there was no clear idea of how a conference chair would approach using such a system, with a resultant poor effect on the user interface.

There are clear lessons for curriculum development, especially in ensuring that the learning and teaching do not become compartmentalised. Task analysis is an important part of requirements engineering.

The key factor in the difficulties encountered, and underlying all others, was that of communication. Many of the other issues would be resolved if suitable communication channels were used. We are proposing to study this issue further.

It is likely that the above factors would also apply to the situations where there is distributed requirements engineering in industry. This is also an avenue for further work.

Acknowledgements

The Chairware project was supported financially by both institutions as well as industry partners. We are grateful for this assistance and for that of many people, especially the participating students. We also thank the reviewers, known and unknown, for helpful comments.

References