Using Usability Experts to Improve Software Quality

Ilari Kajaste¹, Deepa Mathew², Suvi Peltomäki³, Timo Poranen¹

¹Department of Computer Sciences
University of Tampere
Kanslerinrinne 1
FI-33014 Tampere
Finland
ilari.kajaste@uta.fi, tp@cs.uta.fi

²Lionbridge
Hermiankatu 3 A
FIN-33720 Tampere
Finland
deepa.mailaparampil@lionbridge.com

³SYSOPENDIGIA Plc
Hämeentie 135 A
FIN-00560 Helsinki
Finland
suvi.peltomaki@sysopendigia.com

Abstract

Software project courses are an essential part of the university studies for Bachelor's and Master's degrees all around the world. When the results of projects are estimated, one of the key quality factors is the usability of the software. In this paper, we explain how usability issues are considered in a software project course at the University of Tampere. Our experience shows that using student usability experts in software projects improves software quality and project learning on a more general level.
1.0 Introduction

Software project courses belong to the body of knowledge in all computer science-related disciplines in every university. In many undergraduate programs, the software project course is studied on the third year, just before reaching the BSc degree. ACM and IEEE describe [1] the capstone project as follows:

“Provides students working in groups, with a significant project experience in which they can integrate much of the material they have learned in their program, including matters relating to requirements, design, human factors, professionalism, and project management.”

The software project course is a compulsory course for all computer science students at the University of Tampere. The course starts in October and ends at the end of May. Project stories [2] are published in the report series of the department. The stories consist in short descriptions of the projects, statistical data, experiences, and user interface screenshots. For other software project course descriptions, see Dalcher and Woodman [3] and Mahmood and Rashid [4].

The Department of Computer Sciences at the University of Tampere has two major subjects: Computer Science and Interactive Technology.

Computer Science students can study traditional subjects for their Master’s programmes: Algorithmics, User Interface Development, Software Development, Data Management and Information Retrieval, and Information Systems.

Interactive Technology students concentrate on their studies on issues related to human-computer interaction (HCI). The content of this major subject is described in the Curricula Guide [5] as follows:

“Interactive technology aims at training all-round IT professionals who have friendly approach to their work. Making the quality of interaction between man and technology is a core element in teaching of interactive technology. Due to a multidisciplinary basis, students can utilise their backgrounds and interests and specialising in making software and hardware usability evaluations, or concentrate on developing new and better ways of interaction from a human perspective. Students can find jobs in a wide variety of different fields; they can become product development professionals in the software and telecommunication industry, usability experts in Internet and multimedia companies, and researchers in the field.”

About one fourth of all our students are majoring in Interactive Technology. These students need experience on working in software projects, and they should be familiar with the different phases of software development life cycles. Before 2005, Interactive Technology students studied the software project course together with other Computer Science students. They were expected to do similar tasks to
those of the other students, although in practice they usually concentrated on user interface-related issues.

At our university, only basic programming courses are required as prerequisites for the Interactive Technology students. Instead of software development studies, these students acquire strong skills in user interface design and evaluation.

Over ten years ago, ACM SIGCHI’s Curriculum Development Group gave a recommendation [6] that important HCI topics and issues should be incorporated in technology-oriented courses.

Two years ago, we decided to develop further our concept of software project teaching and started the so-called usability team experiment. In our software project courses there are roughly 100 third-year students with a number of older students, who work as project managers for the projects. On the first time we selected seven students to study the software project course in the usability team, and on the second time 12 students with two project managers. The role of a usability team student differs from the other project course students in that the usability team members only concentrate on the usability-related issues in the projects.

There are several advantages in the Usability Team (UT) arrangement. Firstly, the team allows students majoring in Interactive Technology to concentrate in project work studies in a way that is meaningful for their studies. They get practical experience of collaboration in connection with software projects, a situation which they may very well encounter in their future work.

The team also makes sure that the usability matters of every project are being considered. Since there are not enough usability students for each project, and not each project needs the full attendance of a usability expert, the team provides a way to divide the work in the best way.

Finally, the team provides peer support for the members of the usability team working in different projects. Many usability tasks (e.g. usability testing) require collaboration with several usability-aware people, and the team arrangement makes that collaboration frictionless.

Further, as an indirect result, the more complex organizational structure of the project work course provides experience in various collaboration skills for all the participants of the course, not just for those in the usability team.

In this paper, we introduce our usability team concept and our experience during the academic years 2005–6 and 2006–7. During these years, we have gained experience from 37 projects in total and on two slightly different usability team structures.

The rest of this paper is organized as follows: First, we explain how usability
issues are related to software projects and what kinds of deliverables are useful for software projects from the usability point of view. In Section 3 we describe our experience of the usability team experiments. Section 4 concludes our work.

2.0 Usability in software projects

Usability has been defined in many different ways, one of which is the International Organization for Standardization’s [7] definition of usability as “the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.” Nielsen [8] divides usability into five measurable properties, which are commonly referred to when talking about the usability of a product:

- quick and easy to learn,
- efficient to use,
- allows rapid recovery from errors,
- pleasant to use, and
- easy to remember.

Usability can also be used as a means for measuring the quality of a user's experience when interacting with a product or a system. Usability is often a crucial factor for the success or failure of a product.

It is important to notice first that usability in a software project is not a one-off issue that can be covered with a few documents. It resembles good code structure - the software cannot simply be created and made usable only afterwards. To achieve software with good usability, it needs to be taken into account throughout the entire software project [9]. Next, we shortly describe standard usability-related deliverables in waterfall and incremental software development models [10]. Other development models, such as iterative and agile models, were also available, for the course, though they were not used as much. Moreover, as they are not as centered on the deliverables, they are not taken into account here.

2.1 Deliverables in waterfall development model

Waterfall software development model [10] is a traditional way to construct software. The workflow proceeds sequentially: first the requirements of the software are specified with the client, then architecture and software modules are designed to meet the requirements, and, finally, after implementing the system, the software is tested to verify that all the requirements are met. The waterfall model is a detailed description of the basic development premise of “first design, then implement”. As such, concepts of the waterfall model can be adapted for other development models that share the same basic development premise.

With the usability team experiment, we have used two mandatory usability-centered deliverables: usability analysis and user interface specification. These documents are written parallel to other project documents, and have a strong relation to them. The documents we describe have been considered suitable for the
software project course and the usability team, but, of course, they may need to be adapted to suit actual (e.g. corporate) projects.

In essence, the same two factors define both usability and software development: the users and what they do. As with requirements specification, it is important to learn first about the users and about the context of use [11]. This information can be gathered at the same time the project team starts learning about the context of the project (e.g. by observing and interviewing the users). On the basis of contextual information, the \textit{usability analysis} document is produced. It contains a definition of the general user types of the final product, the context of the product, the environment of use, and, based on these, the usability requirements for the software. This defines the premise on which the usability aspects of the project are designed. Even before the project plan is written, taking into account the user experience or usability perspective may also provide a valuable insight into what the software should be about.

Since the requirements specification defines what the software should be, it cannot contain definitions that are contradictory to the usability requirements of the software. The requirements specification of the software is thus based in part on the \textit{usability analysis} document. This also provides a good foundation for considering the usability later on in the project.

Some aspects of the requirements specification may depend on the user interface (UI) of the software, so the preliminary \textit{draft of the user interface plan} should be created together with the requirements specification. This should contain the general user interface model and the intended workflow for the users of the product. If UI is not designed at this stage, those writing the requirements specification will assume some user interface from the technical perspective and it may not be in line with the usability requirements. Because of this, the people designing the user interface plan also need to collaborate with the people writing the requirements specification.

The most visible aspect of software usability is the user interface itself. This is defined in the \textit{user interface plan}. As the plan is tied together with the implementation, it should be done in cooperation with those writing the implementation plan. This way the user interface designers get valuable feedback from the people doing the implementation. This prevents creating an UI design that will be discarded as impossible to implement. However, when compromises must be made for the sake of implementation, the better UI design concepts should, nevertheless, be written into the user interface plan to help further development.

The usability analysis and user interface plan documents are the basic building blocks in binding usability strongly in the software project. In addition to these basic documents, there can also be other documents concerning usability issues, such as \textit{usability test reports}, \textit{heuristic evaluation reports}, etc. These documents are usually written on the basis of various usability evaluations.
Usability design and analysis methods can be used to improve the usability of the software being designed. The most common analysis method, at least in the software project course, is probably usability testing and paper prototype testing, which provide valuable insight into how the actual users interpret the system through the user interface. This is important because one of the key issues in usability is that people tend to think within their own context. While user interface designers and IT professionals may understand a feature as self-evident, the actual users might not. Especially testing with paper prototypes may help discover significant unexpected user behavior in a very early stage of the design.

A closely related issue is that user interface designers easily become accustomed to their own design and blind to any mistake in their design. They may understand how something functions simply because they designed it themselves, and this is often hard to notice. (A similar effect occurs with programmers as well: it often takes a long time to understand how the code written by someone else works.) To overcome this, the easiest method is probably to hold peer reviews on the user interface. Peer review is conducted by holding a meeting with other user interface designers unrelated to the project. The interface is informally discussed together, with the designer presenting the interface to the others. The meeting requires an informal brainstorming atmosphere so that it does not focus on what the designer has done wrong but rather on how the design can be improved.

A significantly heavier method in comparison to peer reviews is heuristic evaluation, where the user interface is separately analyzed by usability professionals using a set of usability heuristics (often Nielsen’s [12] ten usability heuristics) and assessing whether the user interface follows these guidelines. Other methods include for example design, usability and cognitive walkthroughs, where usability experts analyze a prototype carefully by using a set of user tasks.

Figure 1 illustrates how different basic software project documents relate to each other and to usability documents in the waterfall development model. Strong arrow lines indicate that a document is based on information from another document. Dashed arrow lines represent a weaker relation. Lines with arrows at both ends indicate that the documents are done in collaboration and thus are based on information from both sides. Documents bound with a box with a strong arrow signify completing a draft into a final document. On the right in the Figure, recommended deadline months for reviewing the documents are shown.
2.2 Deliverables in the incremental development model

The incremental development model differs from the waterfall model in that the software is designed and programmed in smaller units — increments. Quite often, logical sets of functionalities that can be designed and implemented separately in an incremental fashion [10] can be recognized from the requirements.

Usability tasks are slightly different in the incremental development model. The definition of general user types, as well as the context of the product and the environment of use can be defined for the entire software in the usability analysis. However, since each increment may concern a different part of the software, the usability requirements should be written independently for each increment. These can be separate documents, or just revisions of the original usability analysis. Obviously, the user interface plan also needs to be written independently for each increment. The same idea applies in agile development models [10], though with a more flexible approach to documentation as most of the documents affect one
another in a more dynamic and open way. The basic development ideas conveyed by the waterfall model document relations to each other should still apply.

Figure 2 illustrates how different basic software project documents relate to each other and the usability documents in the incremental development model. The notation is similar to that used in Figure 1. The documents inside the large area will have to be written for each increment.

Figure 2: Usability-related deliverables in projects applying incremental software development model.

### 3.0 Usability team experiments

#### 3.1 Usability team 2005-2006

The Usability Team (UT) [13] pilot experiment took place during the academic year 2005-6. This first UT consisted of seven students, all in their third or later year in their studies, and each had Interactive Technology as their major subject. The students were selected for the group by using the amount of their completed Interactive Technology studies as a criterion. The following courses, considered indispensable for working as a usability expert, had to be passed with good grades
by the usability team members: Introduction to Interactive Technology, Introduction to User Interfaces, Usability Evaluation Methods, Human Factors in Interactive Technology, and Graphical User Interface Design [5].

The UT did not have a project manager, so everyone in the group was equally responsible for management. The reason was that there were not many project managers that year, only nine students took the Software Project Management course, and there were 18 projects to manage, so it was decided to try whether usability team could find a way to manage itself. This was seen by the UT as a serious shortcoming, since a consensus was harder to reach without someone setting and managing the agenda, distributing tasks and keeping the larger view in mind. The weekly group meetings were used to keep in touch and make sure each project was being consulted in fairly the same manner. The meetings also offered a chance to discuss design decisions, and many of the students also took part in heuristic evaluations or cognitive walkthroughs for the designs of other students. These were early forms of the peer reviews of the next year.

At the beginning of the course, the manager of each project was told that they could freely employ an outside usability expert student for their project for an estimated 50 hours. The managers could then decide for which phases the expert was needed. Each student in the UT chose two or three projects to consult, but most students also consulted other projects, when needed. At the end of the course, the students reported how many hours the experts had in fact been employed in the project. It was found that the time was anything between 20 to 161 hours, which in most cases was far more hours than expected. A few projects did not employ a usability specialist at all, since they programmed a command-line application, or they had enough usability skills inside the team.

The projects ranged from ones where the customer was a researcher from the same department to projects where the customer was a large international corporation. The platform also varied from mobile to web, so the usability experts had a good opportunity to work on several different kinds of projects.

The nature of the project determined how the experts worked and what role they played. In some cases, the usability expert worked as a complete outsider to the project, only consulting from a “distance”, and sometimes the expert became a solid member of the group. Consulting from the distance could also lessen the effect of the advice given, as the consultant did not always stay aware of the implementation decisions made by the project members. The consultant also needed to consider the programming skills of the project team members, so as not to design a user interface that the project could not implement. In some projects a good user interface design was created collaborating with other project members, though it was never implemented, as the time ran out.

Projects that required very good usability sometimes had the expert become a member of the group all the way from the beginning. This way there was a chance to have a stronger effect on the outcome of the product. This was the case in the
Safety at Work project, demonstrated in Section 3.1.1. The project demanding the most usability work was a project creating the support system for heuristic analyses. In this project, the customer was very demanding in terms of usability, while the project members were more technically oriented. Thus the consultant did all the usability work. In another project, the customer was so pleased with the performance of the UT member who had made extensive prototypes for the project that they even offered a job for the UT member. These are examples of projects where there was a clear need for a usability consultant, and thus the UT concept proved useful.

The average time the usability experts spent on the course was 250 hours, and the amount of written pages varied between 67 and 243 per expert. The usability team produced 17 usability plans, 9 user-interface plans, 3 usability test plans, 10 usability tests, 6 usability test reports, 5 user interface walkthroughs, 6 heuristic evaluations for the user interface, and 16 other documents (paper prototypes, a user’s guide, and various reports). The UT also wrote their own project plan, where all the usability tasks were listed for each project, including their scheduled starting and completion times with an estimated amount of work. The UT also wrote weekly reports to the course supervisor, and when the projects were finished, the team wrote their final report, which was checked together with the supervisor.

When the course ended, the UT members reported their experience and possible suggestions for improving the usability team concept for the following year. The UT members felt that their workload was heavier than that of normal project members, as they had to keep up with several projects and deliver documents to each of them. Similarly, this meant that an equal number of project managers instructed them in their work. Altogether, the students reported satisfaction in the idea of consulting the projects. This gave them the opportunity to have an extensive overview of the range of projects in the course and the design challenges in different types of platforms. The projects that were consulted also gave positive feedback for the help they had received from the usability experts, and thus the department’s decision to continue with the usability team concept was easy to make. Given the circumstances of introducing a new concept to the Project Work course, the usability team pilot experiment was considered a success.

3.1.1 Case study: Safety at Work project

In autumn 2005, Salpaus Further Education [14] from Lahti, Finland, approached the Department of Computer Sciences and offered a student project for the project work course. The idea was to develop a web-based learning environment for studying and getting a work safety card.

Salpaus Further Education had decided to move to web-based training instead of a normal one-day (8h) course arranged by a certified trainer. The goal of the project was to move the course material to the web to allow the students an opportunity to study the material and to exercise and practice their exams independently, so that the training would become easy and motivating.
The biggest challenge of the project was to ensure the good usability of the learning environment, as it would potentially have tens of thousands of users, whose levels of technical skills vary from a novice to an IT professional. The project group saw that the easiest way of development would be to use the already existing Moodle platform [15] and to develop additional modules for it. Moodle as such was insufficient in its usability for the work safety course, so the usability expert worked together with the project group to plan enhancements to the interface. These enhancements were needed to make the course material easily navigable, the exercises pleasing and educational to do with good feedback, and to make it easy to keep track of the learning process.

The usability expert had, incidentally, been involved in making the preliminary requirements specifications for the project. Because of this, it was natural to involve her in the project meetings from the very beginning. The project group also included a student keen on usability, so the expert worked together with her to ensure a good user interface design. The usability documents produced for the project included usability plan, user interface plan, usability test plan and usability test report. Additionally, another expert from the UT was consulted for an expert usability evaluation after the user interface plan had been created.

The culmination point of the project was when the nearly finished product was tested with several end users in a usability laboratory. The testing was done to see how real users would react to the concept of studying the material on the internet, and to spot any remaining usability flaws in the user interface. Normally, the users are expected to study the material and work on the exercises for several hours before taking an exam. Therefore, in the test it was only possible to note briefly how the users felt about the learning environment and whether they could navigate around in it and find all the necessary items.

The test resulted in a list of findings, positive and negative alike. The positive findings included flexible support for different navigation styles, easiness in learning the structure of the site, usefulness of working on exercises in studying for the exam, and the idea of being able to stay aware of one’s progress during the studying. The negative findings included lack of instructions, inadequate feedback in the exercises, excessive notifications and unnecessary steps in beginning an exercise, and insufficient display of the user’s location on the site. Many of these shortcomings were due to already existing Moodle features that the project group had not modified.

Most participants found the learning environment a good way to study for their exam, but one participant remarked that he would still rather have someone explain the material verbally to him. It seems that, for some students, a combination of lectures and working on exercises in the learning environment would be the best solution. One of the test participants was an immigrant, and he noted that for many immigrants taking the course it might be easier to study the material independently, as they could then study at their own pace and easily go through the material several times, if they did not understand every detail because of the language
barrier.

Most of the usability problems found in the tests were solved, and for the rest, ideas for further development were proposed. Many other projects in the software project course in 2005-2006 decided to skip the formal usability testing in a laboratory, but in the Safety at Work project it was seen as a meaningful phase in delivering a quality product. The customer’s representative observed the test and was able to form a very clear idea of how users work with the learning environment and the course material. Thus, in addition to performing the tests to find and eliminate errors, it was a worthwhile way to share the accomplishment with the customer before the final release.

3.2 Usability team 2006-2007

The main difference between UT 2005-6 and UT 2006-7 was that the number of team members increased to 12, and that two project managers were assigned to the team. The team was split into an English and a Finnish group. The English group consulted eight English projects and the Finnish group consulted 10 Finnish projects. Each UT member was assigned to one or two projects. Some of them also helped the other projects.

UT used the distributed organizational structure where usability experts were assigned to work with separate project teams and they reported to the usability project manager. This structure allowed the usability expert to work with the project development team. Staying with the project team from the beginning to the end of the project increased the possibility for usability recommendations to be implemented.

This year (2006-7) the usability project manager assigned tasks to the usability team members and held team meetings to discuss the progress and issues related to working with their assigned project teams. In addition, an online Moodle forum was created, where the usability team members could discuss and exchange ideas.

Last year the usability team members worked for more hours than expected. This year each member was assigned to work for a maximum of 255 hours. Each member filled in a timesheet every week, and the usability project manager approved the hours and made sure that the members did not cross the limit. If a UT member had too much work to do, the work was split with another member with less work load, in order to balance the working hours. Since each project had different requirements, and some usability experts worked on more than one project, it was important to maintain a balance in the amount of working hours.

At the end of the second usability team experiment, the workload reported was contrary to the first year - some students even reported that they would have been ready to do even more work. A lighter workload than that of the previous year was to be expected, since the document requirements were tuned down, based on the feedback from the previous year. However, the depth of the work the UT members were doing was agreed to be satisfactory, and the cases where there was scarcity of
work were attributed to coordination problems with the projects, an issue that needs to be looked into next year. Altogether, the students were satisfied with the concept of the usability team, and they were glad they were able to do the project work in a way meaningful for their studies. While effort is still needed to tune the usability team concept to perfection, it was agreed that the team should indeed be continued and even expanded, if suitable for the course.

3.2.1 Case Study: SysMLL Project (2006-2007)

The School of Modern Languages and Translation Studies at the University of Tampere offered a project SysMLL (Multiple Language Learning System), a web-based multilanguage learning software [16], for the students in the project work course in 2005-6. This year, 2006-7, they offered a project to develop further the existing working version of SysMLL built during the previous year.

The pilot UT had worked on this project in 2005-6, and all the usability designs, documents and reports were available for reference for the new UT. A new set of usability documents, such as usability analysis, usability requirement, user interface plan, usability test plan and usability test report were written by a usability team member. One of the English usability experts was assigned to work closely with this project team and to report to the UT project manager.

The main goal of the SysMLL project was to add more functionality to the previous year version. The incremental development method was used for this project, and thus each phase had a new goal. In each phase, the usability expert had to monitor the usability of the product, to suggest improvements if needed, to conduct usability tests, and to create usability documents and reports. The project group added increments to SysMLL until the end of the course period. The final increment, documentation and presentations were ready by May.

New usability tests were planned after each increment, and the first test was arranged in the Department of the Computer Sciences usability laboratory in February 2007. In addition to the usability expert assigned for this project, two other UT members took part in the usability tests as a moderator and an observer throughout the tests for monitoring and recording the findings. A technical support assistant was also available during the test period to take care of the technical problems of the laboratory and of the application prototype.

The focus of the usability test in February was on evaluating the major functions in the SysMLL application. However, the test tasks indirectly touched on other functions of the system such as settings, user management and user help. The test was planned and designed for students and teachers, since they are the target users of this system. The tests showed that the application still needed many improvements as for usability. The project team then worked closely with the usability expert to further improve the usability of the software.

As expected, in the final incremental phase of the project, more usability planning, evaluations and tests were carried out to improve the final version of the software.
4.0 Conclusions and future work

The usability team experiment has proved to be an effective and satisfying way for the students of Interactive Technology to study project work. Since the concept is still young, there are no reports from graduate students on how well the concept fits in with the actual work environment. Judging on the current feedback, we expect the results to be good.

Coordinating the team members with the projects appears to still need refining. The organizational structure of the project work course is more complex because of the usability team. This is, of course, a challenge in terms of arrangements, but, nonetheless, it provides a convenient opportunity for all project work students to learn the much needed skills of working within complex organizational structures. For the usability team members, dealing with this complexity is imperative during the course, and if they continue working as consultants, the situation will be the same in working life later on. While the freedom of adapting the role of an individual UT member to the requirements of each project does provide efficient flexibility, the possible roles that a UT member can take should perhaps be more clearly defined, to avoid losing efficiency in the organizational confusion.

The work of preceding usability teams will provide a good starting point for refining the structure and practices of the usability teams to follow.

5.0 References

1. SE 2004 - Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering, sites.computer.org/ccse/ (visited April 2007)
7. ISO 9241-11: Ergonomic requirements for office work with visual display terminals (VDTs) - Guidance on usability, 1998
11 ISO-13407: Human centred design processes for interactive systems, 1999
13 Usability team homepage. www.cs.uta.fi/~uteam (visited April 2007)
15 Moodle course management system website. www.moodle.org (visited April 2007)
16 Mihailov M, System for Multilanguage Learning. hamppu.uta.fi/~lomimih/sysmll_project/ (visited April 2007)