User experience design in open source development: Approaches to usability work in the Moodle community

Olli Savolainen

University of Tampere
Department of Computer Sciences
Interactive Technology
Master's thesis
Supervisors: Saila Ovaska and Eleni Berki
September 2010

University of Tampere
Department of Computer Sciences
Interactive Technology
Olli Savolainen

User experience design in open source development: Approaches to usability work in the Moodle community

Master's thesis, 82 pages, 6 appendix pages
September 2010

Usability work is problematic in open source software development on many levels. There is often no documentation or explicit research of user goals to drive design, no user experience vision to direct development, no usability testing, and typically no resources allocated for usability work. No usability practitioners are usually employed, and doing usability work is hard due to the community value system being focused on concrete results. An attitude of “code is cheap” leads to a tendency to skip user research, design and validation before implementation. However, developers in open source communities do get feedback from actual users. Also, software development work is iterative, as recommended in user-centered design, so there is some foundation to base usability efforts on.

This thesis builds on a review of the observations made about usability in open source literature. It is followed by presentation of the Quiz user interface project in the Moodle virtual learning environment project. I introduced needs of a user group of the already existing Quiz module, which had not been considered previously, into the design of the user interface. I carried out interviews of users and stakeholders, and iteratively designed the user interface using prototypes, usability testing, and community feedback, in each phase of the project. The results of this project are integrated in Moodle 2.0.

As main results, I present observations of the current state of the usability engineering of the Moodle project, followed by proposals for directions to take in the project, as well as in open source projects in general. The aim of these proposals is to help make the usability body of knowledge an active and permanent source of insight in development, while keeping open source projects accessible and attractive to volunteer developers. I suggest ways to engage the community in usability work and propose increasing usage of low-fidelity usability testing in development. I then discuss how working circumstances of usability practitioners could be improved and propose various tactics to learn about users in an open source context.

Keywords: Usability, software development, user experience, user-centered design, user research, open source software, community, virtual learning environment

Contents

1. Introduction

Open source software (OSS, or Free/Open Source Software, FOSS) is becoming increasingly visible in current software landscape. Ideas central to open source development include transparency of development processes (by making the source code of software openly available) and decentralized peer review. Open source projects typically have a core team, possibly paid for their work, and a community of volunteers contributing according to their interests. In the business models of open source enterprises, the revenue typically comes from providing support and other services, instead of from selling software.

Usability work in open source software projects is challenging. Pemberton (2004) states the problem being as follows: Either open source developers need to become interested in ordinary users, or the users' needs have to be brought to developers by mediators. Indeed, it seems that not only open source communities have a lack of systematic thinking of users, but it is also challenging for usability practitioners to approach open source communities.

In OSS projects, usability professionals' participation is rare (Nichols & Twidale, 2003). Typically usability work is more challenging in OSS projects than in the commercial world due to usability practitioners' world view being foreign to a functionality centric developer community (Nichols & Twidale, 2005). Iterative development and releasing often is appreciated and there is often no clear product development cycle in OSS projects. Design of new features, fixing bugs and maintenance tend to be highly integrated (Massey, 2003). Alas, there is often no specific phase in which usability practitioners could provide input, and they can not take any influence on the application for granted: they are easily viewed in a consultative role instead of a participative one, where they could more effectively represent users (Iivari, 2008). Developers go straight from user requirements to implementation and more or less improvise the user interface (UI). Usability, then, tends to get “glued on” to applications when the data structures and software architecture are already finished from the developers' point of view. (Nichols & Twidale, 2003; Thomas, 2008; Trudelle, 2002)

During the years 2007 and 2008, I implemented a user interface change in the Quiz module of the popular open source virtual learning environment (VLE) software Moodle. The goal was to improve the usability of the UI of the module for editing quizzes, and to increase its viability to one of its potential user groups. Throughout the project, I invited teachers, support staff and other potential users to provide input to the project in interviews and usability tests, in order to gain further understanding to guide the design work. Of the time I used on the project (originally called Quiz UI redesign and referred to as the *Quiz UI project* in this thesis; see Figure 1), I also spent a considerable amount of time interacting with the community and convincing them that the needs unmet by the application existed, and that the issue cannot be remedied just by making simple fixes here and there. During the project I also learned a lot about the general attitudes of the members of the community toward usability work, as well as about their skill levels. This work presents that project.

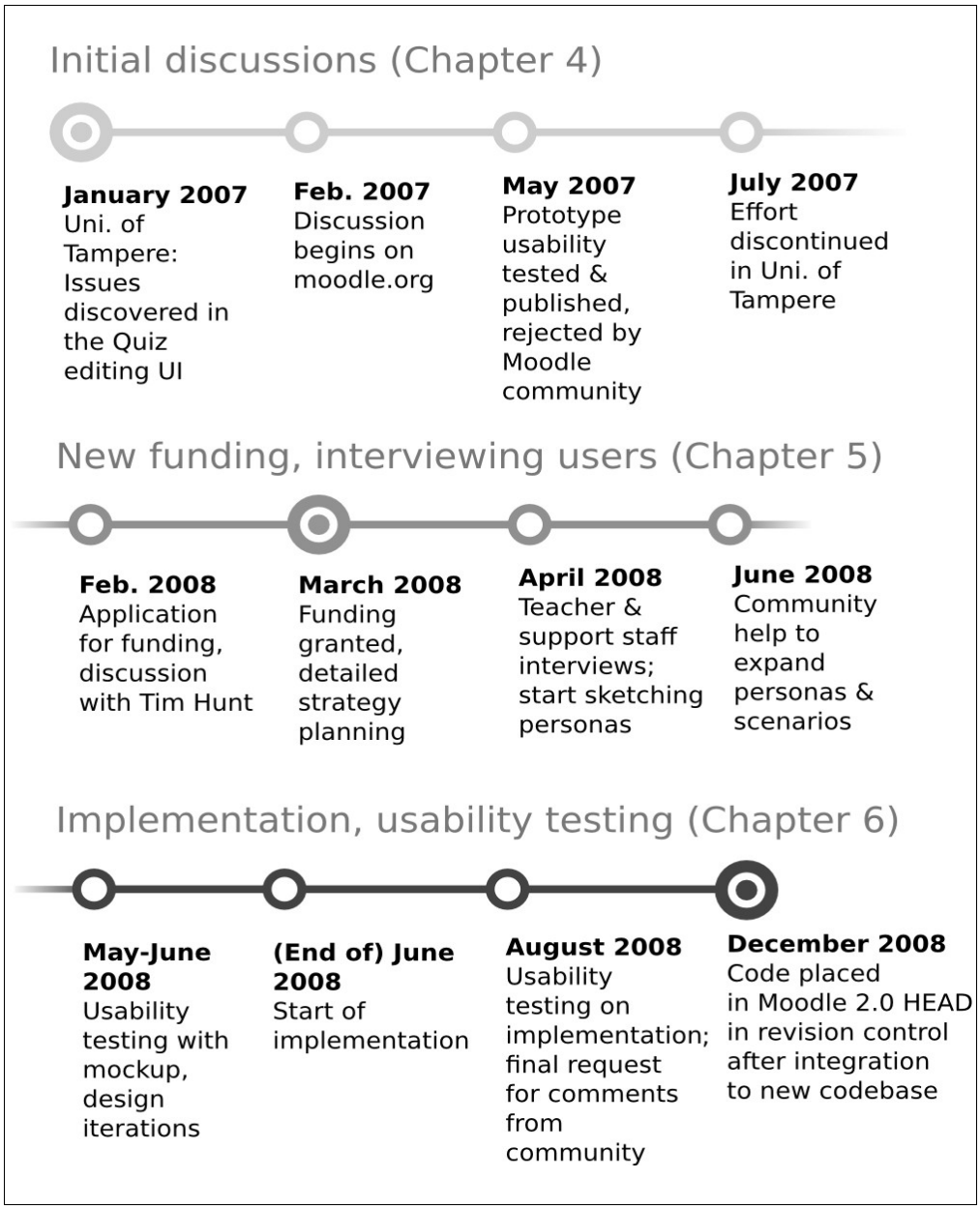


Figure 1: The time line of the Quiz UI project

My personal motivation to take up the Quiz UI project was that I was working as a programmer, but was tired of programming. I could master it, to a degree, but could never be as excited by it as by interaction design and by the personal experiences of people. I saw usability work and the field of Human-Computer Interaction (HCI) as a challenge, but one that was very meaningful for me, being interested in understanding people also outside of the context of computer science, in the context of psychology, social psychology and even social work. Open source fit my views of how software development should be done: Maybe less work would be wasted reinventing the wheel when development was in the open. Maybe open source could empower also those users who could not afford proprietary solutions. Faced with a UI level challenge in the open source VLE Moodle, finding the solution to which would take me on a journey to the worlds of open source development, usability work, and learning, I then took the opportunity.

User-Centered Design (UCD) is an approach, in which information about the people using the product are at the core, and usability of the product is the sought after outcome (ISO, 1999). It is defined in the ISO 13407 standard as four stages that form an iterative process (see Figure 2):

- Specifying the context of use (user characteristics, user goals and tasks, and characteristics of the working environment),
- specifying user and organization requirements,
- producing design solutions, based on the understanding gathered, and
- evaluating the design (often by means of usability testing) against requirements.

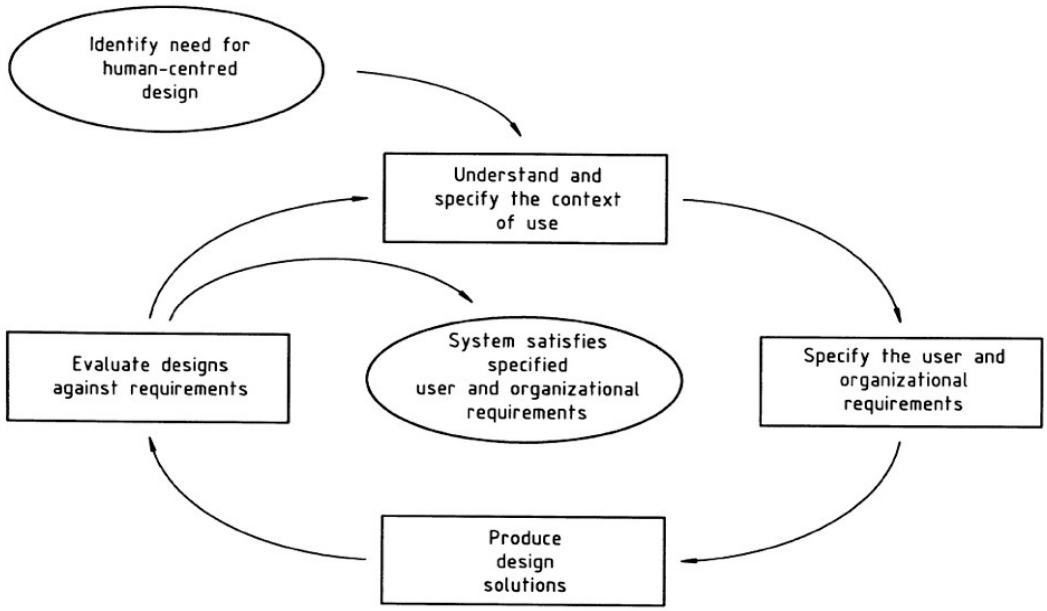


Figure 2: The ISO 13407 standard: Human-centred design processes for interactive systems (ISO, 1999)

UCD is defined as a specific process that is to be followed in a software project to make usable software and user experiences in more general. The HCI community has

developed an array of methods and tools to gain understanding about users: inspection methods, usability testing methods, participatory design. The importance of multidisciplinary design teams, drawing from a variety of different intellectual cultures including but not limited to psychology, sociology, graphic design and even theatre studies, is emphasized (Nichols & Twidale, 2003).

During the Quiz UI project I came to realize that people in the Moodle community are interested in serving the users. However, the level of usability is not applauded (or measured), user goals are not explicitly taken into account in the design, and no processes such as UCD are used. Common UI patterns seem to be mostly ignored. None of the phases of the UCD process described above are carried out before the development already begins. Still, some of the mechanisms in use do support gaining some understanding of the users and their goals: community member feedback is encouraged, actively discussed and used in development.

In any case, it seems that aspects of the decades of research that UCD is based on should somehow be incorporated into OSS development practice, as well. This is challenging, however: in particular, specifying context and organization requirements is non-trivial in the distributed development environment of OSS, where several organizations may be represented and user requirements specification is done very informally. Key questions for open source usability work then seem to include: In order to have premium user experiences (UX) in open source software, which elements in standard processes like UCD, should be taken into account more profoundly in open source? Which parts of the iterative process emphasized in UCD should be developed, in order to make sure usability quality assurance (QA) is carried out to a sufficient degree in OSS? These questions are bigger than this work, but some angles of them will be covered. In this thesis, I discuss the questions:

- How does learning about users happen in an open source project? What are the challenges and possible pitfalls?
- How to encourage an open source community to embrace usability practice and to base the design of software on a deep understanding about users, like in user-centered design?

Chapter 2 reviews the literature, observing what others have found tricky about open source quality assurance and usability work. Chapter 3 introduces Moodle, the Quiz module I worked on, and the design I created to improve a particular user interface of Moodle. Chapters 4 to 6 present the interactions I had with the community and other stakeholders, when introducing different iterations of the design to them, and finally when implementing the result design. In Chapter 7, I take a broader look at Moodle's situation with regard to usability work, and in Chapter 8, I examine proposed solutions to the challenges related to usability, as encountered in Moodle and in other open source projects.

2. Achieving good usability in an open source project is difficult

> "[…] the challenges of designing hardware and software must come back to designing for human beings--*all* human beings. This path will be long, and it won't be easy. But we owe it to ourselves and each other to do it right. May the Open Source be with you!" (Raymond 1999; *emphases* are the author's)

Regardless of this declaration by Raymond, an iconic open source pioneer, several authors recognize that usability often continues to be a weak point in open source software. The situation and its solutions have been well looked into by Nichols and Twidale (2003) among others (Hedberg, Iivari, Rajanen, & Harjumaa, 2007; Iivari, 2008; Nickell, 2002; Raymond, 1999; Thomas, 2008; Trudelle, 2002). The claim that usability (or a lack thereof) constitutes an inhibitor for OSS adoption is not without merit (Nichols & Twidale, 2003; Viorres et al., 2007). On the other hand, open source projects do produce GUI applications which are of value to users. Examples like mozilla.org, who engaged UI designers in the community (Trudelle, 2002), in addition to OpenOffice.org, GNOME, and NetBeans, backed by Sun's StarOffice HCI experts (Benson, 2004), seem to show that open source projects, with dedicated usability practitioners, are indeed capable of producing quite usable software (Çetin & Göktürk, 2007). Nichols and Twidale (2003, 2005) perceive a historic parallel with the development of the usability of commercial software: Systems initially developed for experts gradually got adapted into contexts where non-technically oriented users required UIs suitable for them. In fact, in examples like OpenOffice.org, OSS seems to imitate closed source software and thus to be forced to play catch-up with it, to make transfer learning easier (Nichols & Twidale, 2003).

This chapter examines usability in open source projects, and in particular the different challenges OSS projects tend to meet with respect to usability work.

2.1. The traditional open source project

Many successful open source projects work on infrastructural software for horizontal domains. In projects producing operating systems, web servers, browsers, compilers or databases, domain knowledge may already be in the software development community (Fitzgerald, 2006). Systems are built by hundreds or even thousands of volunteers, though some are supported by companies, and some participants are not volunteers (Mockus, Fielding, & Herbsleb, 2002). Work is often not assigned; people undertake the work they choose to (Mockus et al., 2002).

The user experience is often not designed before implementation. Instead, developers rely on iteration and feedback from community members – who hopefully represent the user base – for making sure the UI corresponds to user needs (see Section 2.2.).

The Unix Philosophy (Raymond, 2003), often summarized as *do one thing, and do it well*, is a design philosophy. It expresses interaction design ideals: a flexible framework is built on modular independent parts, which users can combine whichever way they wish. Application programming interfaces (API) and command-line options are no less interfaces for human beings than graphical user interfaces (GUI). This approach to interaction design is very different than UCD, though.

OSS projects have a leader, who, at the end of the day, can decide whose contribution goes into the official software. The leader does not, however, manage the effort to a similar degree than in traditional, commercial software development. Overall, open source projects lack many of the traditional mechanisms for coordinating development (Mockus et al., 2002). Projects are often oriented in a *straight-to-the-action* manner: the user research and design phases are largely skipped (Trudelle, 2002). Defining what the software should be like happens iteratively, as development progresses, and replaces the initial planning, requirements analysis and design phases of conventional development. As such, there may not be an explicit system design, a project plan, a schedule, a list of deliverables, or defined processes (Warsta & Abrahamsson, 2003). The idea of a user interface getting explicitly designed is thus in a sense antithetical to the idea of open source. “Code is cheap” is a general attitude, and the real action happens in programming code. Satisfaction comes from practical problem solving, not from applying abstract usability concepts (such as scenarios or personas). Users cannot be reduced to points in a to do list, so they, too remain abstract (Nielsen & Bødker, 2007) and distant.

It is often claimed that OSS software development is typically started to “scratch a personal itch” of the developer (Pemberton, 2004). Especially with large packages with many modules, governed by different people, the design can be fragmented. Many people may claim to have the right to determine the future of the product, and reaching consensus may be difficult with a lack of common goals. (Frishberg, Dirks, Benson, Nickell, & Smith, 2002)

To facilitate distributed collaboration between open source developers (and to reduce the learning curve for new developers), applications are highly modularized (Fitzgerald, 2006), and the OSS approach to building software is bottom-up (Nichols & Twidale, 2003). This, without global oversight of the UI as a whole, makes designing consistency into a UI difficult (Trudelle, 2002). On the other hand, making sure that for each component there is a maintainer, who can handle maintaining the development of that component, makes it more probable that each component gets at least minimal attention (Trudelle, 2002).

It is said that developers care about the quality of their work more, because they only work on things for which they have a real passion (Raymond, 2001). An open source project is typically a meritocracy (Trudelle, 2002) where status (and trust) in the community is gained by tackling problems considered challenging. There is a bias

for improvements in functionality instead of usability: Nichols et al. (2001) state that “hard” algorithmic problems may have a greater perceived value in the “reputation market” than “soft” usability related issues, being also easier to specify, evaluate and modularise than usability issues.

2.2. User feedback and co-development

Open source projects take pride in getting input from the users in the open source community, by means of “elaborating requirements, testing, writing documentation, reporting bugs, requesting new features, etc.” (Nichols & Twidale, 2003). If the developers are similar to the users, to whom the software is targeted, and conventions for how interaction with users and other applications is to be handled are strong enough within the community, it seems that a commonly accepted level of usability (within that given community) may be reachable just by means of iterative community feedback.

Still, developer communities can appear inward-looking to users and the community hacker culture closed and idiosyncratic (Nichols et al., 2001). Depending on the application domain, there can be a language barrier between developers and end users. Current systems for reporting software issues problems are reported difficult to use and biased towards bugs that can easily be expressed textually or with single screenshots. Usability problems and their interrelations are often hard to express in such issue trackers. In addition, analysis of complex usability problems is difficult with these tools and meaning is often lost (Nichols & Twidale, 2003; Viorres et al., 2007). Users can also be seen as co-developers (Raymond, 2001) – if a user has a usability issue, they may be encouraged to fix the application themselves. Alternatively, developers may find it more convenient to encourage the user to read the documentation in order to work around an issue, perceived to be a problem of the user instead of one of the software.

Software quality is perceived to be controlled by factors such as the Linus' Law (Raymond, 1999), which asserts that “given enough eyeballs, all bugs are shallow”: the more people work on the code, the easier and quicker the bugs in software will be found. However, due to the subjective nature of usability issues, often this rule can not be applied to them.

Twidale and Nichols (2003, 2005) differentiate between objective and subjective usability bugs. It is easy to agree on the severity of objective usability bugs, such as a button not fitting a window in a given situation (rendering the application dysfunctional). On the other hand, developers may be tempted to tag subjective usability bugs as “works for me” in the bug tracking system. There is typically no actual research about the severity of such issues, so people reporting such bugs may have a hard time justifying them. Also, if the developers doing the “debugging” hold common knowledge not obvious to users, the lack of such information in the UI of the application could be missed by all such developer eyeballs (Nichols et al., 2001).

According to Fitzgerald (2006) the potential for open discussion between users and developers is still there though. It be a clear benefit compared to proprietary software practice, where users and developers are often located in separate departments and may have little mutual respect or voluntary interaction. In commercial software enterprises users are often being only listened to by support staff (Cooper, 1999). They might or might not mediate users' issues to the developers, to add to the understanding of user goals in the development process. In the OSS project Iivari (2009) studied, she perceives that often, “the developers have to rely on very few user comments when making decisions”.

2.3. Professional usability practitioners in open source projects

The field of HCI (Hedberg et al., 2007) stresses that great usability requires trained usability specialists taking part in the development effort. Usability needs to be acknowledged early in the process by learning about the users: usability is always related to specific people using the system to achieve specific goals in a specific context of use (ISO, 1999). In open source projects, the interaction gets defined by sometimes getting the opinion about the system from some users of the system. Listening to users, however, does not give realistic insight into what users actually do, and a confusion of the target user audience reigns (Nielsen, 2001). Open source projects tend to risk not taking into account any of the stated usability factors: the people, their goals, or the context of use. Hedberg et al. (2007) predict that as even more non-technical users are entering into OSS world, the cornerstones of OSS will break down, unless proven usability methods and processes are introduced and adapted to OSS.

Getting usability professionals to join open source projects is thus critical in order to gain realistic insight about the users, but rare (Nichols & Twidale, 2003). Usability practitioners may not feel valued in open source communities. In addition to integrating their analysis and design into ongoing functional development, they must show extra effort to convince the community about the value of their work, which is not commonly understood. They must manage discussion between the two very different world views of functionality-centric expert developers and end user advocates (Adams, 2003; Nichols & Twidale, 2005). Being isolated and alienated from the community's efforts makes it difficult for usability practitioners to participate.

That burden of proof of the value of the entire field is demonstrated to a degree also by the Quiz UI project presented in this thesis. Developers may not see the point in discussing abstract usability issues and discussions easily derail to implementation details or to “bike shed colour” (Kamp, 1999) type discussions, where disproportionate amount of attention is paid to details. Nielsen and Bødker (2007) describe an OSS community to have a “culture of doing” where results are valued over abstract questions (with which usability work often deals).

Also, even in OSS projects where designing the user experience is key to the success of the application, it may be hard for a usability practitioner to provide his/her input. Design documentation, which a usability practitioner could use as an input for design, is often scarce. It can be very hard to get funding for professionally done research. This was an issue also in the Quiz UI project (see Section 4.1), where in practice the funding I was granted were intended for implementing the design. I had to do most of the research required on my own time.

As many developers are volunteers, they choose whether to participate and which problems to take on. Without an additional incentive, reported usability bugs may get crowded out amongst other bugs if they are not interesting enough for the developers to fix (Nichols & Twidale, 2003). Not getting design work implemented will likely additionally lower the motivation of usability practitioners to participate.

3. A user interface with issues: Editing a quiz in Moodle

To provide an example of usability work done in an open source project, this chapter presents a user interface of the Quiz module in the web-based virtual learning environment (VLE) Moodle[1]. This chapter also gives background to the presentation of the progress of the project itself in the following chapters. The Quiz UI project related to this user interface took place during the years 2007–2008. In the project, I utilised user interviews, discussions with the community, and usability testing, to guide design.

All in all, the effort to get the Quiz UI changed lasted two years from the beginning of 2007 to nearly the end of 2008. The work was begun while I was working in the Tenttis project of the University of Tampere (UTA), but eventually I needed to get separate funding for it, as the scope of the changes required was larger than originally expected.

In this chapter, I will present the issues found in the user interface, followed by the eventual solutions: the redesigned and implemented user interface. Chapter 4 continues to describe how the effort to redesign the UI got initiated and the initial reaction of the Moodle community to the ideas in their then-current form. Chapter 5 describes the expansion of the effort from its original existence inside the Tenttis project to a separate project with funding and user research of its own. Chapter 6 wraps up the description of the Quiz UI project by discussing the iterative usability testing and implementation that happened during it, and the discussions leading to its inclusion in the Moodle core code.

3.1. Moodle basic concepts

To help understand the issues of designing the Quiz UI, I will proceed to present some of the core concepts related to Moodle and the Quiz module. Moodle's features and its learning philosophy, are further presented, though succinctly, by Zsolt and István (2008).

Moodle is a web application for producing courses and websites online. It is written in the PHP language popular for web applications. It has been translated to over 70 languages. Moodle development is principally ran by the Moodle Trust. The Moodle Trust is funded by Moodle Partners, who get their revenue by providing Moodle services to companies and educational institutions, and in turn forward a percentage of their revenue to the Moodle Trust. Dougiamas, the leader of the Moodle Trust, describes his way of running Moodle as a 'benevolent dictatorship'. (Dougiamas, 2010)

On a site managed by Moodle, all content is divided in courses. A course is the basic structure, within which practically all content and activities exist (Figure 3). The bulk of all users are typically either students, who participate in courses, or teachers,

[1] http://moodle.org/

who manage and run those courses, though also more intricate roles exist. Roles can be renamed, and additional ones can be defined.

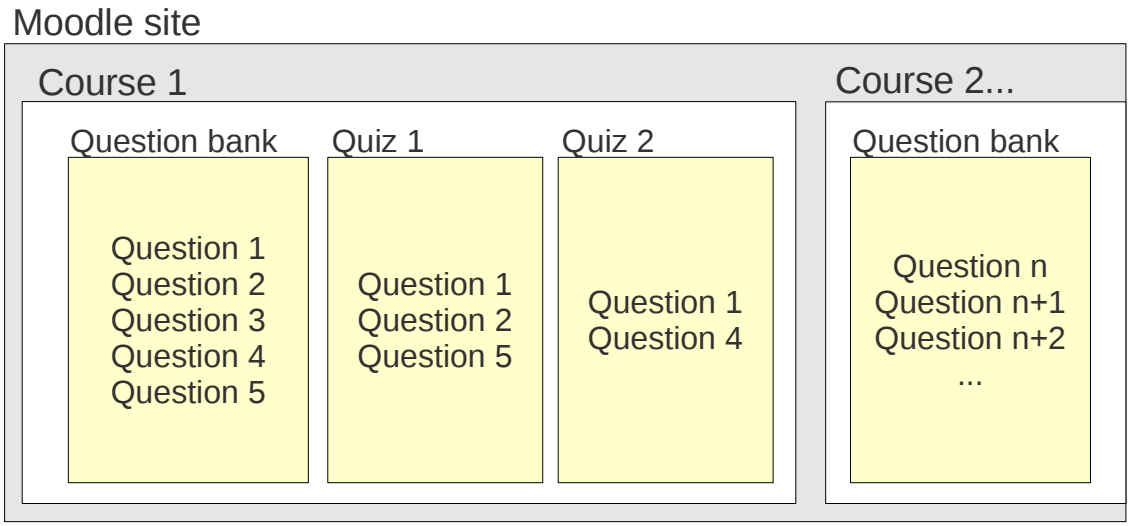


Figure 3: Simplified illustration of a Moodle site, its course hierarchy, and the relationship between the Question bank and Quiz modules.

Within a course, questions are created and stored in the Question Bank module, possibly organized into categories and subcategories. From the Question Bank, they can be added into any number of modules that make use of questions, such as quizzes (Figure 4). Questions in Moodle can be of various types such as Essay, Multiple Choice, Matching, Short answer, and True-false. Also custom or third-party question types can be added[2].

Like other activity modules, each quiz exists in a given course. Quizzes can be taken by students (Figure 5), and quiz attempts graded by teachers. There is also functionality for random questions in a quiz. A random question is always connected to a given category in the Question Bank. When a student attempts the quiz, a question from that category is randomly picked to the student's quiz.

[2] http://docs.moodle.org/en/Question_types

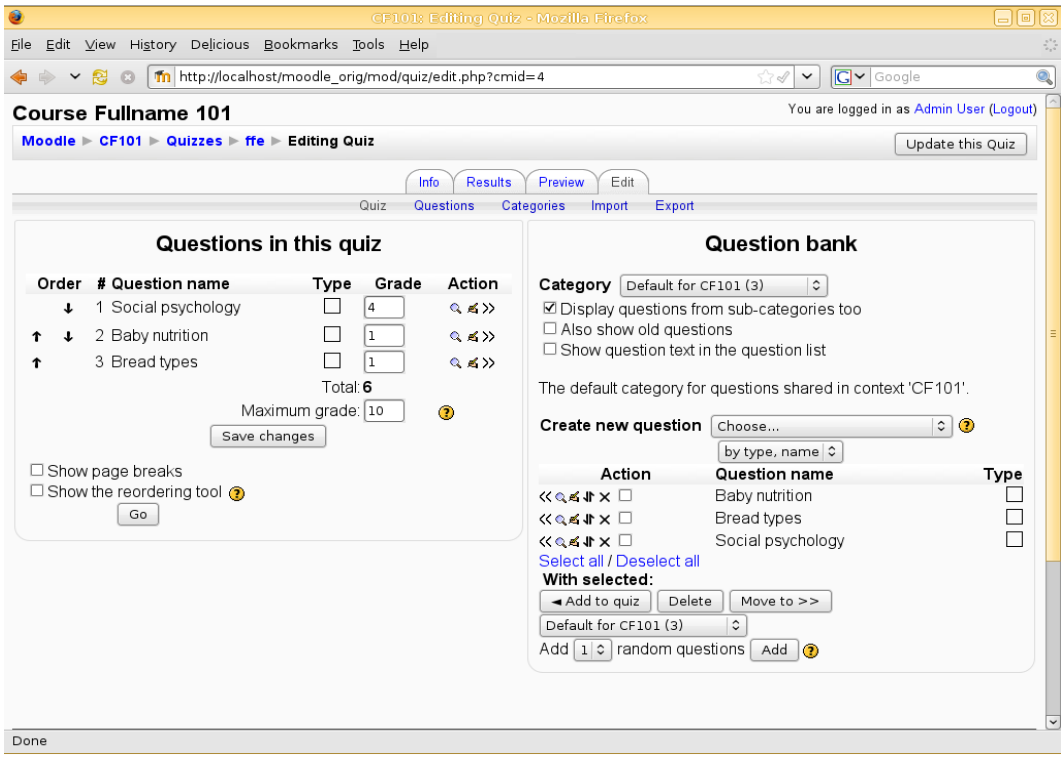


Figure 4: The main screen of the Quiz editing UI before the Quiz UI project, in Moodle 1.9

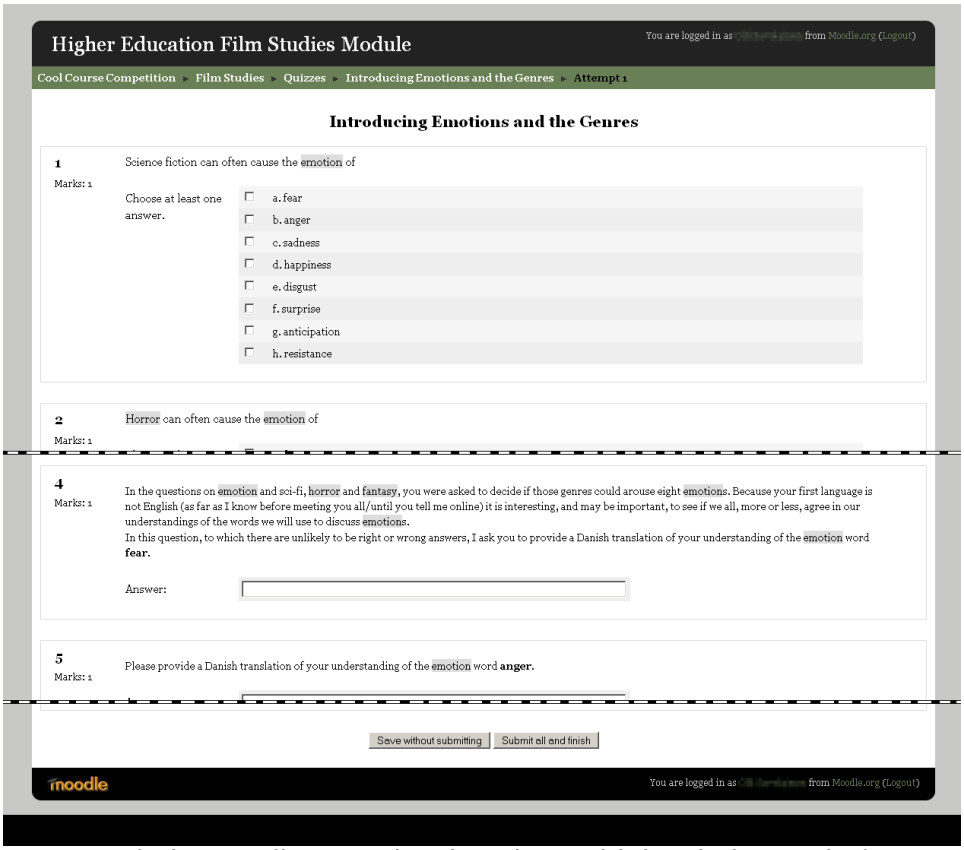


Figure 5: A quiz in Moodle 1.9, showing the Multiple Choice and Short Answer question types (screen shot altered; alterations marked up with horizontal dashed lines)

3.2. Moodle development

Moodle 1.0 was released in August 2002[3]. It is released under the General Public Licence (GPL), along with its documentation wiki content[4]. For following development plans of future versions, Moodle has a roadmap[5]. There is extensive documentation for developers[6], and an issue tracker[7]. Since summer 2009, Moodle also has preliminary Human Interface Guidelines (HIG) documentation, which I started to develop with the community in summer 2009[8].

In the Moodle community, six discussion forums are dedicated to offering Moodle support, 33 are dedicated to different parts of Moodle, and 10 to coordinating development efforts distinct from individual components[9]. In addition, there are more user oriented forums for discussing pedagogy. The forums of different components of Moodle contain both development and usage discussion. Also, there are forums for discussing different contributed plugins that can be added to a Moodle installation. Links to Moodle forum threads are marked in this thesis as footnotes. The forums are open to guest access by clicking the "Login as a guest" button resulting from opening any of the URLs.

Among the development forums, there is a forum for usability issues. Usability bugs can be reported in a separate 'usability' component in the bug tracker (as opposed to the advice of Trudelle (2002)). Issues in the tracker can belong to several components at the same time, so a usability issue in the Quiz module can be assigned to both the component "Quiz" and the component "Usability". Apparently many of the usability-related bugs reported are not perceived as usability bugs: the number of usability bugs is low, about 1,4% of all reported bugs according to the tracker[10], as opposed to KDE, where the rate has been as high as 27% (Çetin & Göktürk, 2007)

Members of the Moodle community gather in Moodle Moot conferences[11] worldwide to discuss pedagogy, development and usage of Moodle. In Spring 2010, the first online Moodle conference, iMoot[12], was arranged.

3.3. Quiz module and its development in the community

The Quiz UI project presented in this work aimed to improve the user experience of constructing and editing quizzes, exams and the like, in the Quiz module.

---

3 http://docs.moodle.org/en/Background
4 http://docs.moodle.org/en/License
5 http://docs.moodle.org/en/Roadmap
6 http://docs.moodle.org/en/Development:Developer_documentation
7 http://tracker.moodle.org/
8 http://docs.moodle.org/en/Development:Moodle_User_Interface_Guidelines
9 http://moodle.org/forums/
10 Situation on September 9th, 2009: 284 issues in the usability component, out of a total of 19850 issues
11 http://moodlemoot.org/
12 http://imoot.org/

The Quiz module is sometimes not regarded highly in the Moodle community. The module's usage tends to not be in the spirit of social constructivism, which Moodle aims to support (Zsolt & István, 2008). It is a commonly stated challenge to Moodle that though Moodle facilitates learning where everybody is a potential teacher as well as a learner, the majority of courses created on Moodle are based on "traditional" learning and pedagogical methods (Zsolt & István, 2008).

Regardless of this, the Quiz module was included in Moodle quite from the start of the development of Moodle. Dougiamas, the Moodle project lead, pushed the first version of the Quiz module into the revision control in October 2002[13], shortly after the release of Moodle 1.0., and requested for a dedicated maintainer for Quiz in October 2003[14], while Moodle was at version 1.1.

In November 2004, Dougiamas started discussions about getting funding to pay for the living of the developer who had been developing the module since December 2003[15]. After this, the maintainer of the module changed once more, before Tim Hunt of Open University, the current maintainer, took over the maintenance of the module, on May 23, 2006.

To give a rough idea about the volume and popularity of the discussions on the Quiz module, some statistics follow. I first inquired about the need to work on the Quiz module on January 31 2007[16], when Moodle 1.7 was the last released version. Hunt committed my work into Moodle 2.0 CORE in the revision control system in November 2008[17]. During this time, in the forum of the Quiz module, 1795 discussion threads (possibly even more[18]) have been active. Threads have 4,3 messages on average, with a median of 2 messages per thread. The threads were or had been started by 1085 different user names[19]; forty threads by Hunt, and five of them by me. Moodle 2.0 is expected to be released in autumn 2010.

3.4. The eventual improved design and its design goals

The initial need for development of the Quiz editing UI of Moodle 1.9 was to make the UI more approachable. The sheer amount of functionality exposed by the UI was an obstacle to novice users. Many refused using the UI and required support staff to create their quizzes for them. It also seemed to force a workflow suitable to only some of the

---

13 http://cvs.moodle.org/moodle/mod/quiz/index.php?view=log&pathrev=Moodle_109

14 http://moodle.org/mod/forum/discuss.php?d=3074

15 http://moodle.org/mod/forum/discuss.php?d=15235

16 http://moodle.org/mod/forum/discuss.php?d=63612

17 http://moodle.org/mod/forum/discuss.php?d=103869#p486607

18 This count includes threads that had messages added to them during the period of time, that is, also threads started before the start of the period have been included. Threads, which had activity after this time period (and before the data extraction date, February 25th, 2010), even if they were started within the period, were not included in this count, though. This is due to the fashion in which the Moodle forum module displays discussion threads.

19 Of course, as in any internet forum, it is possible that several real life people used the same user name, as well as a single person using many different user names.

users. Figure 6 presents the final, redesigned UI as it was after it was added to Moodle 2.0 in November 2008.

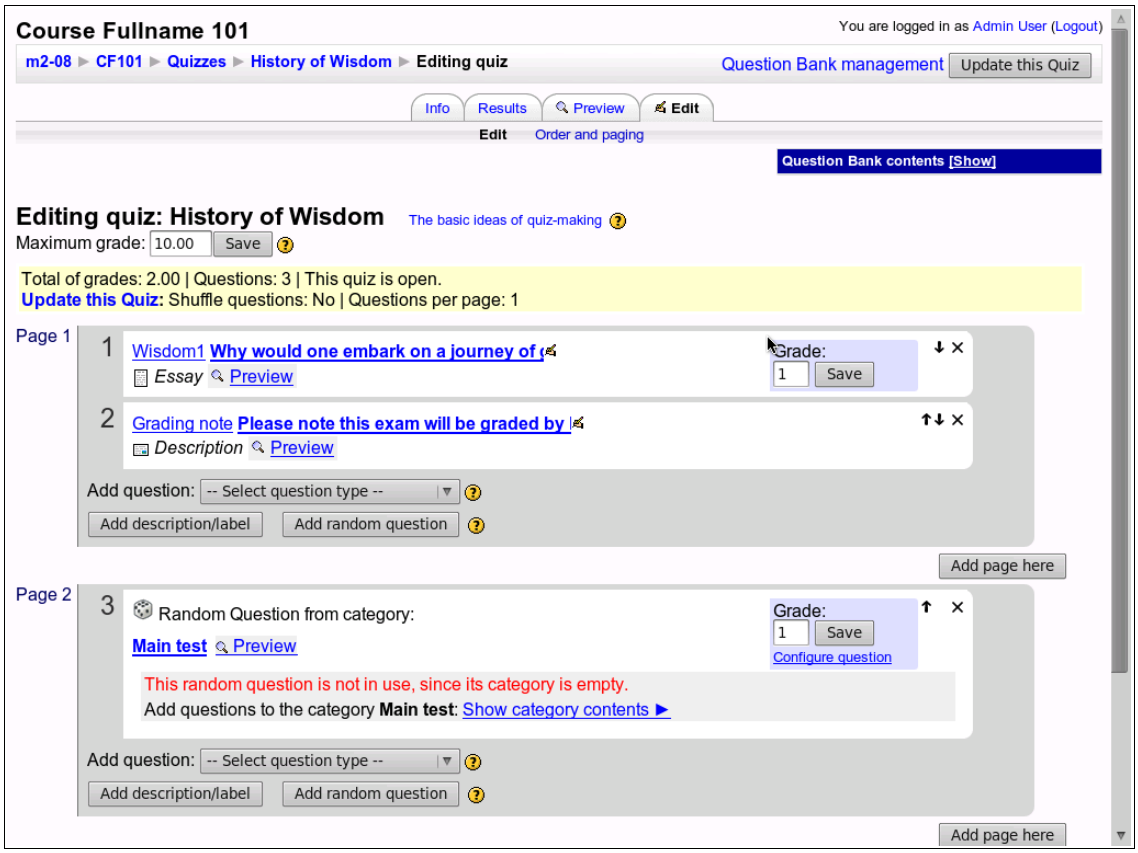


Figure 6: The final user interface after the project (with the Question Bank "window" closed; the default view) at the end of 2008

I wanted to make the new UI support more of a direct manipulation (Shneiderman, 1997) style interaction than it did: users should see the target of their actions, the contents of the quiz and necessary attributes of questions. They should be able to manipulate what they see without unnecessary obstruction. Also, the main operations available and the workflow supported by the application should be obvious when looking at the UI. Implementing undo functionality, considered a part of direct manipulation, was and is out of scope at this point, though.

No new functionality was designed into the new UI: this was a redesign of the existing functionality in a new UI. This was with the exception of minor features such as the ability to select and delete multiple questions from a quiz at once. I will now briefly introduce the features of the new design, along with their justifications.

Add questions directly to a quiz

In the workflow implied by the old UI and its original conceptual model, questions were managed and organized in categories of the Question Bank of a given course (Figure 7). Actual quizzes were only the end results of the workflow: once you had your Question Bank organized appropriately, you would be able to generate different kinds of quizzes. The UI, with its rather abstract conceptual model, had apparently been

built with the primary goal of rigour in mind: the learning curve of the application was relatively high, but it was powerful for managing large numbers of questions. In the new design, this rigour needed to be preserved to satisfy existing users.

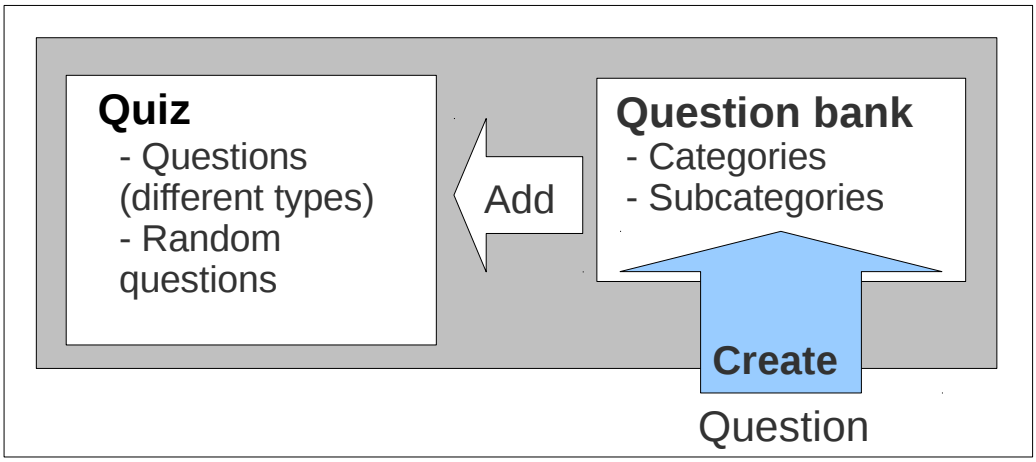


Figure 7: The conceptual model the old UI is based on. Questions are created and stored in the Question Bank, and added into quizzes as needed.

Quizzes (or tests, or exams) have a paper-based equivalent in the traditional educational world. My assumption was that teachers are invariably familiar with that model. My goal was to lower the learning curve of the UI by taking the UI closer to that model, facilitating building concrete quizzes directly. I made this possible by

- supporting creating questions directly into quizzes and
- allowing immediately seeing the results of doing that in a somewhat WYSIWYG – what you see is what you get – manner.

To this end, also actual question text was made visible in the Quiz editing UI, whereas the old UI showed only the teacher-viewable question name (label), required by Moodle for each question. Also the question type is expressed as text in the new UI, in addition to an icon representing the type.

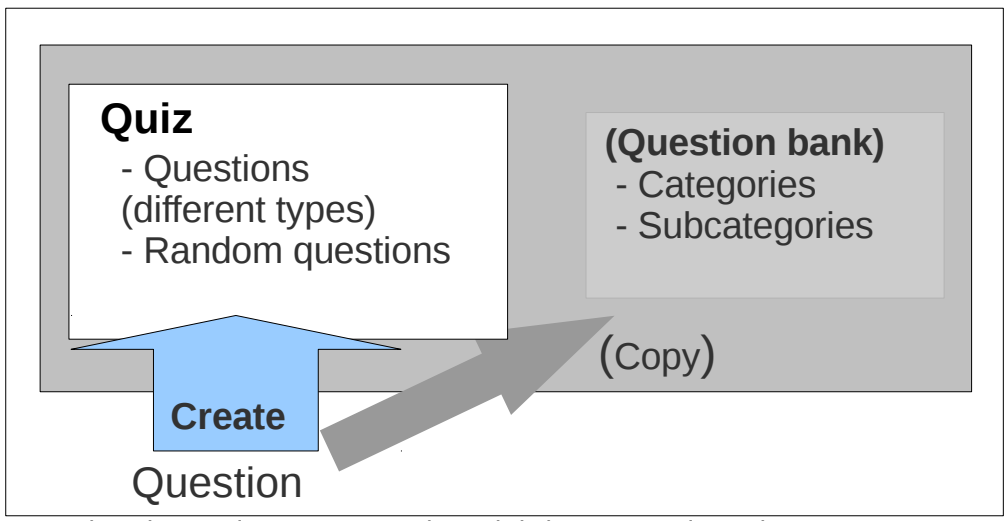


Figure 8: The alternative conceptual model the new UI introduces

Also, instead of adding questions to the Question bank first and then selecting them to be included in a quiz, in the new UI users can do both at once: enter questions directly to the actual quiz, while the software would add copies of them to the Question Bank, as well, for later use (Figure 8).

Effectively communicate the conceptual model

One aspect of effective UI design is usage of visual hierarchy to make the mutual relationships of different elements clear. In the old UI (Figure 4), the quiz being edited and the course Question Bank were expressed as visually equal elements. The quiz editing screen was focused on facilitating assigning questions, which were in the Question Bank, to quizzes.

As the new UI aims to support direct manipulation of quizzes, enabling users to add content directly to a quiz, the Question Bank's role is one of a necessary tool. What is in the question bank is no longer the main focus of the UI – there is a separate UI for working on the content of the Question Bank already in Moodle 1.9. Because of this, the Question Bank is also less prominent in the visual hierarchy of the new UI: it is displayed as a toolbox window (Figure 9).

Also, the question bank is hidden by default, and available by means of progressive disclosure (Nielsen, 2006). This decreases the choices initially available to users learning the application. Progressive disclosure helps alleviate the fact that the old UI

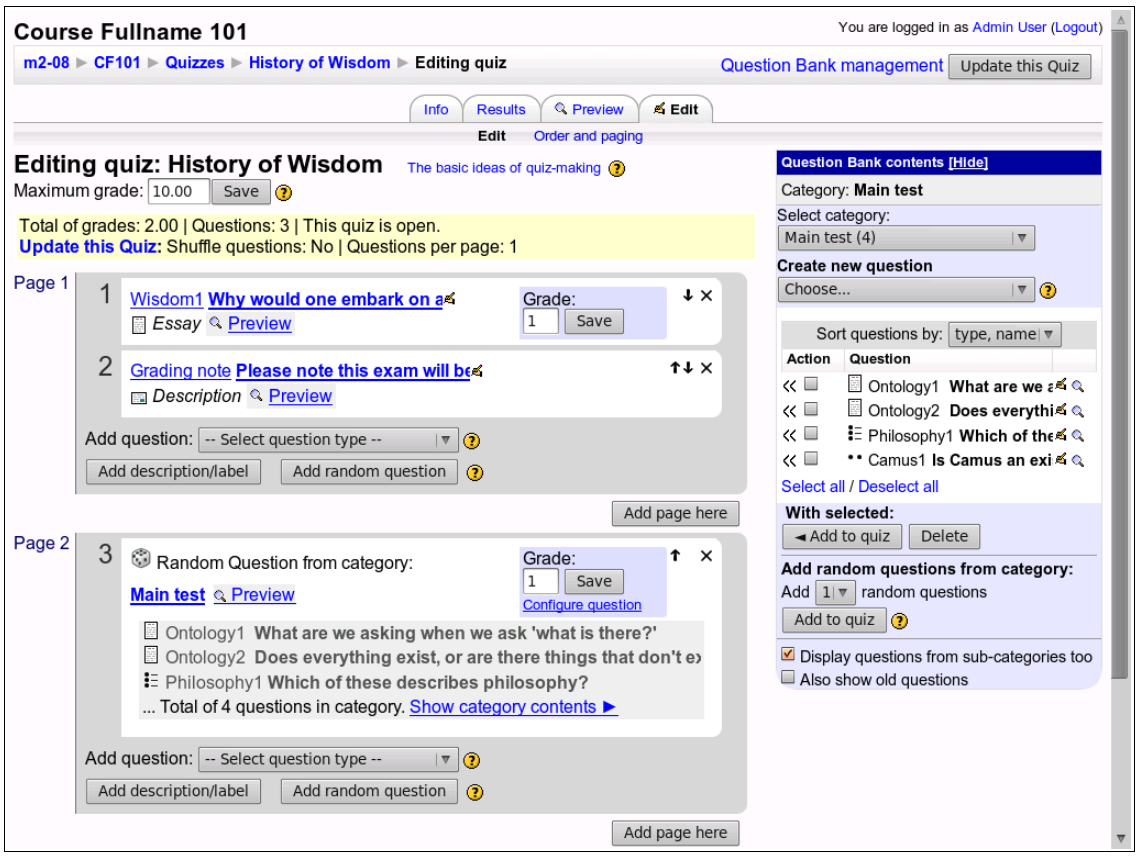


Figure 9: The final user interface after the project (with the Question Bank "window" open)

had lots of functionality, none of which could be removed as such. As the initial view is simplified, this results in enhanced guidance (*guidage*) defined as “the means available to advice, orient, inform, instruct, and guide the users throughout their interactions with a computer [...]” (Bastien & Scapin, 1993, p. 9). This helps users more fluently understand the workflow the application supports when seeing the UI. The eventual goal of the progressive disclosure here, as is typical, is to allow inexperienced users to move from the simple just-the-quiz model to the full model at their own pace, perhaps after first having gotten done what they wanted to do on their own terms.

Make modes more apparent

The old UI also had two additional modes, triggered by checkboxes “Show page breaks” and “Show the reordering tool”. These checkboxes were placed inside the quiz in the UI, implying that they are meant for manipulating quiz content. In reality, these both altered the view by adding additional controls for manipulating aspects of the quiz being edited. They had apparently been added since the UI was too cluttered if those features were always visible.

I determined that the need, which the checkboxes were meant to meet, was "rapid switching between alternative views of the same information object", which Nielsen (1999) calls the original and primary purpose of **tabs**. Thus, I moved some of the reordering functionality to a separate tab (see Figures 10 and 11).

The compromise made here was that not all functionality could any longer be visible at the same time: the functionality placed in each of the tabs would need to be carefully considered in order to not make users continuously switch back and forth between tabs.

As an unplanned side effect, some of the more experienced users have apparently ended up using the new order and paging tab for managing their quizzes, since it is less verbose than the main tab. Also, as the new tabs are actually not just views to the data, but also alternative sets of functionality related to that data, the point Nielsen made over a decade ago may be moot.

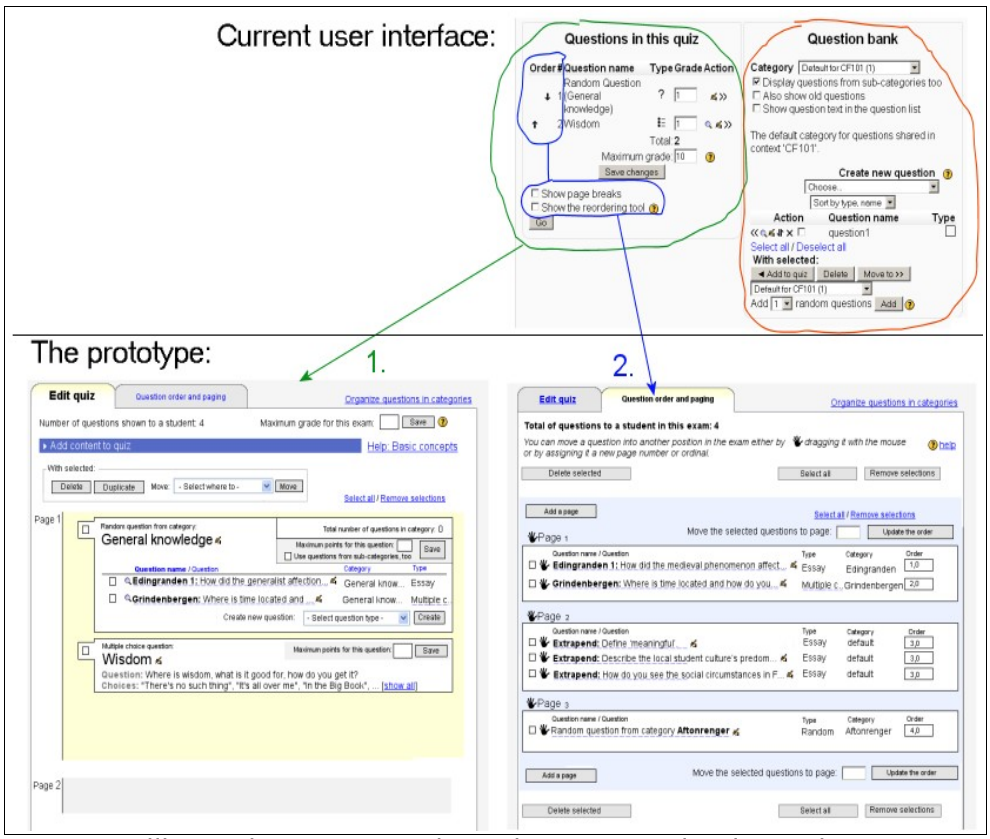


Figure 10: An illustration presented to the community in Spring 2008, of the change I was planning to make

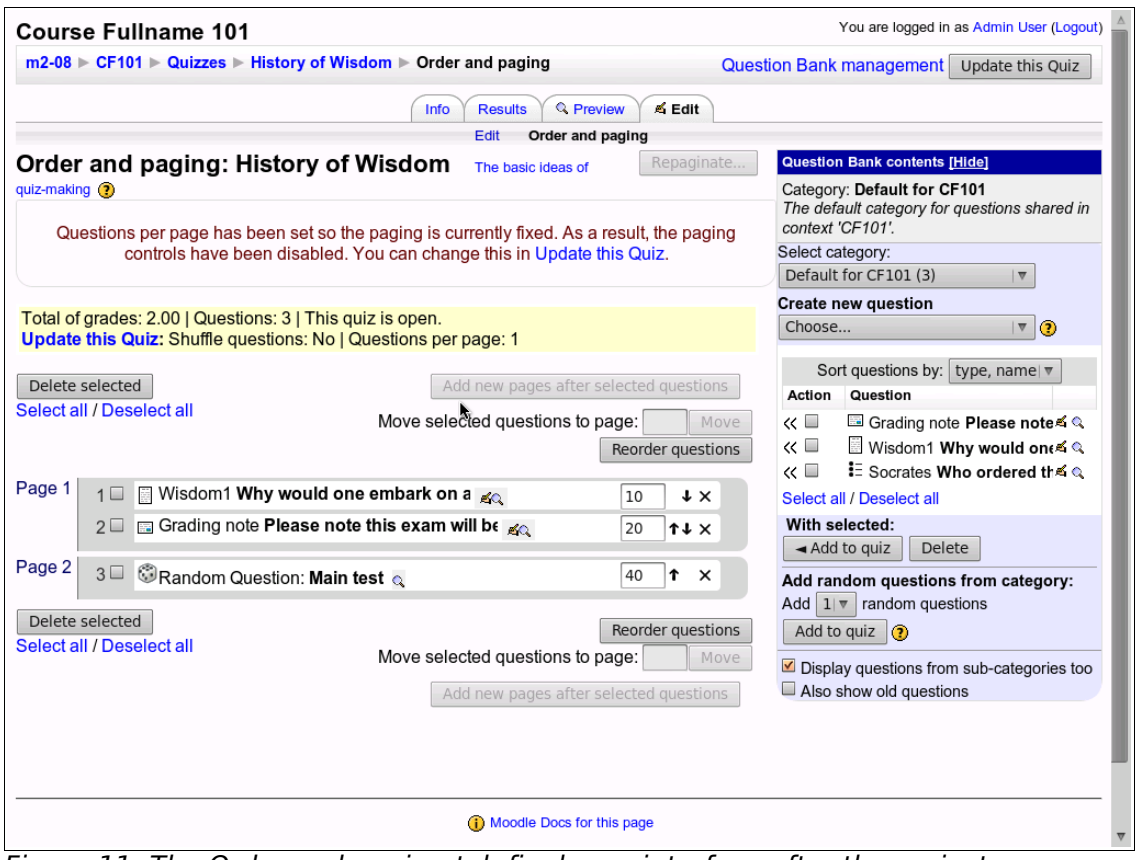


Figure 11: The Order and paging tab final user interface after the project

Concept switch: Pages versus Page Breaks

In preliminary paper prototype tests of an early iteration of the UI, a test participant noted something along the lines that they did not understand the concept of "page breaks". Discussion with the test participant after the usability test led me to understand that the UI violated the usability heuristic *Match the system with the real world* (Nielsen, 2005) or *Speak the user's language* (Nielsen & Molich, 1990). As in real world exams, teachers think of placing items on pages, instead of placing page breaks between items. It seemed the design decision of using the concept of page breaks had been made to reflect the fact that the underlying data structure of a quiz is a comma-separated list, where a zero in the list means a page break. I thus changed the UI by adding buttons between pages for adding new pages, as well as by changing the question reordering functionality to convey the terminology of pages instead of page breaks.

In practice, this change also required several other aspects of the UI to be altered. Most importantly, it allowed introducing an aspect of more direct editing into the user interface: rectangles representing questions were now placed visually "inside" rectangles representing pages. Once pages had a clearer visual representation of their own, it was possible to make the controls for adding elements to a quiz more contextual. Each page in a quiz got its own buttons for adding questions and description elements (i.e. text labels; see Figure 6).

Indicate the status of the quiz being edited

Few changes in the UI were made based on the scenarios interviews (Section 5.6.). Adding a status display in the quiz editing UI is one of such changes. In the interviews, I learned that when a teacher creates exams/quizzes, for many it is a task that requires rather intense concentration. On the other hand, it also became clear that today's teaching world does not make it very easy for teachers to isolate long periods of concentrating just on quiz/exam creating. A teacher may return to a quiz after a long time of not having used it. In order to continue working from where he or she left off, it is crucial to be able to see where it was that he or she left off.

Thus, the UI should support the teacher's memory: when a teacher gets interrupted, it should be as easy as possible to continue where they left off. That is, the current status of the quiz it needs to be visible at a glance: whether the exam is open for student to take, and if not, and when it will be (and possibly, when it will close). The Quiz module's settings have quite a few other modes[20] that can affect the behaviour of a quiz in unexpected ways, but have not yet been included in the UI design.

[20] http://docs.moodle.org/en/Adding/updating_a_quiz

Summary

All in all, although the new UI is visually quite similar to the old one, it introduces significant conceptual changes and takes the UI one step further from a system-centric model to a model where the user's data, in this case the quiz and its questions, are on the main stage. Coming to the design presented above did not happen in a day, however, but through several iterations and through quite an amount of discussion with the Moodle community. The following chapters present the phases of that work.

4. Initial discussions

In this chapter, I will describe the interactions with the staff in the University of Tampere (UTA), and with the Moodle community to set the stage for the Quiz UI project. This is to illustrate the context and circumstances of usability work in an open source context, demonstrating the various phases the work went through and the challenges met along the way. Another focus here is to take a look at how UCD methods affected the design.

4.1. Discovering the mess: the old user interface

Starting 2006, I worked as the developer of a project that eventually created Tenttis, formally the Electronic Exam service, in UTA. The project was funded by UTA as well. The aim was to integrate Moodle (by forking its source code into a customized system), and the functionality of the Quiz module, in a custom exam-taking system. The reservation side of the system also uses source code from MRBS, which is an OSS for booking meeting rooms. Tenttis[21] has been in use in UTA since 2007, and it is being piloted in at least one other Finnish university[22]. In the system, students can reserve a PC via the service's website, and then take their exams in a dedicated, camera surveilled classroom on the PC at the reserved time (The Learning Technology Centre of the University of Tampere, 2009).

The instance responsible for providing Moodle support and documentation (among other things) to teachers in the university, is called the Learning Technology Centre (LTC). They participated as stakeholders in the development of Tenttis. Because of their position, they were also informants about Moodle usage in UTA. They reported during the Tenttis project that with the existing Moodle installations, teachers often delegate creating and managing their exams to LTC. It appeared that many of the teachers in UTA who were potential users of Tenttis, were novices with Quiz (and many of them, also with computers in general). They were reported to find the user interface for quiz editing too difficult to understand, and to require too high a level of commitment. The then-current version of Moodle was 1.6, but the UI had been roughly similar also in earlier versions (and remained so up until version 1.9).

In January 2007, I approached the Moodle community in the module's forum[23] with the Quiz module usability problems we had discovered. I learned that no work was being planned on the UI. I requested for requirements documentation for the module, and was told none exists. Already this first discussion revealed the fact that some community members were approaching the UI from a different point of view. They were managing their questions in the separate Question Bank module, and using Quiz

[21] https://tenttis.uta.fi/
[22] University of Turku: https://tenttis.utu.fi/
[23] http://moodle.org/mod/forum/discuss.php?d=63612

in a sense as a secondary tool to present the questions to students. However, the need in the Tenttis project was to make the barrier to entry as low as possible: teachers would need to be able to *just create* their quizzes.

So, in addition to my other tasks in the Tenttis project, I was given the task of determining and solving the issues with the UI, and to make changes to the variation of Moodle 1.6 used in the project. I could perhaps then try to offer the changes to the Moodle community, as a patch.

4.2. Conception of the first prototype

The initial approach I took with the UI was to examine it with the Heuristics for User Interface Design (Nielsen & Molich, 1990), mostly used from the back of my mind. While working to create a paper prototype, I tried to eliminate the issues I had identified in the UI and which (I assumed) had to do with the problems the teachers were having. In the old UI, I found issues not only with how the editing UI itself was presented, but also with the surrounding navigational model. I included suggestions also for solving these issues, making this initial prototype (Figure 12) largely inconsistent with the navigation model of the rest of Moodle. Another very visible difference of this prototype to the old UI was that the Question Bank side of the UI was removed completely.

At this point I thought it best to design an entirely new UI, which would try fix various levels of issues of the UI in a single holistic solution. I assumed that the issues of the old UI, and my proposed solutions could, then, be discussed with the community.

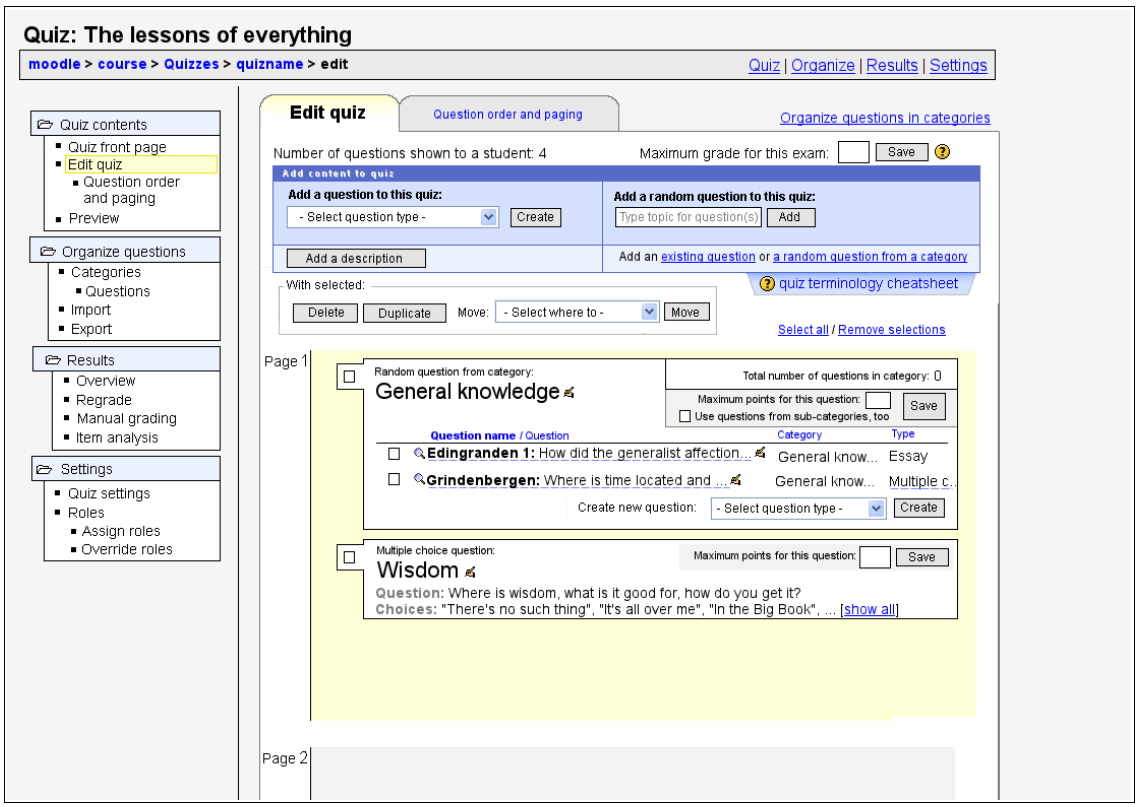


Figure 12: The mock-up presented to the community in Spring 2007

Before presenting the UI to the community, it went through several iterations, and through one brief, informal paper prototype testing with three participants (a teacher from the University of Tampere, another teacher from a Finnish polytechnic, and a relative of mine)[24]. One of these early tests led me to realize the artificial nature of the concept of a page break, which led to the usage of the concept of a page instead (Section 3.4. and Figure 12). The full list of changes brought about by this initial testing were not very formally recorded and have not been kept to be reported here. See Appendix 2 for the tasks used in these tests to develop the initial prototype.

Reactions of the community

In May 2007, I proceeded to present my prototype to the community in the forum of the Quiz module[25]. I explained the problems with the then-current UI and offered justifications for the new solutions.

The response I got in the forum thread was mostly negative. A considerable number of community members were not even convinced there were any problems with the UI. Instead of changing the UI, many community members saw improving the documentation a more viable way to approach the difficulties users were facing. Tim Hunt, the maintainer of the Quiz module, was worried about the consistency of my proposition with the rest of the Moodle UI.

I was dumbfounded. I spent days trying to find an appropriate reaction to what they were saying. I had spent some weeks creating a prototype and testing it, and I felt was being treated like I had attacked the community. Where not ignored, I thought I had received little relevant feedback on the actual work presented. From the first responses I could read virtually no understanding or seemingly not even an attempt to understand what I was trying to express. There was no counter-argumentation to the actual design, just brief comments bypassing my effort as uncalled for. I used quite a bit of personal support from a colleague, friends and family before going back and continuing the discussion to see just how bad the misunderstandings were – either on my side, or theirs.

I suspected that I was confronting users who had originally been involved in making the UI like it was, as they stated to have none or few issues with the then-current UI. Although at first I found it hard to understand their point of view, the discussion did lead me to see their way of using the UI. In the following section, I will briefly present the discussion, before continuing with the story about the interaction with the community.

Then, during summer 2007 it was decided that development of the design would not be continued in the Tenttis project. The amount of work required by my design was larger than the resources available. Instead, some extra labels in the standard Moodle

[24] http://docs.moodle.org/en/index.php?title=Development:Quiz_UI_redesign_prototype&oldid=23892

[25] http://moodle.org/mod/forum/discuss.php?d=72828

1.6 Quiz editing UI were added in the local Tenttis revision control, to help teachers understand the workflow required by the module.

The conceptual conflict behind the design

As the discussion thread I had started with my proposition of a new UI continued, also more helpful comments appeared. They explained what the existing users believed was good about the old UI, and what the new UI lacked. Not all of what is discussed in this section became this clear to me during that first spring of the effort though – the understanding kept forming throughout the project.

There was a mismatch of the user group the module had been designed for, and of the group whose needs I was exposed to in my work in the Tenttis project (as discussed in Section 3.4.). This was at the core of the initial issue I perceived in the UI.

The old UI did not allow focusing on just the quiz being worked on, adding questions directly to it. This was for a reason: the model assumed in Moodle for quizzes was that a quiz is always dependent on the Question Bank of the course in question. The thinking behind this model is that users *should* always organize their questions, so even if you want an one-time simple quiz, the software makes you create them in the Question Bank anyway. As such, the benefits of this model are clear: keeping things organized for reusability.

The problem with this existing model was that it made the interaction quite complicated. To its developers, the UI apparently facilitated working with quizzes in the only way that made sense, first organizing questions into the Question Bank and then being able to create quizzes in batches.

My perception of the users was one of teachers moving from traditional paper based exams to electronic ones. Some of them possibly see Moodle as something imposed on them by the organization. As such, their motivations vary. Many may not be immediately willing to learn the usage of an intricate UI for creating questions to Question Bank categories, and then moving the questions into quizzes. Maybe they often store their questions in word processor documents that were either old exams, or just lists of questions acting as Question Banks, and saw no reason to move these collections of questions to any new system. In any case, the users I represented just wanted their questions into quizzes. According to the usability testing done, this is exactly what the new UI facilitates.

The discussions about the new UI seemed to stem from the basic question of whether it was really a good idea to change the model of the old UI, introducing a new workflow that eliminated the Question Bank and allowed users to just concentrate on quizzes if they wanted. Does it make sense to allow users to add questions without organizing them too, or should they be required to do it “right” from the start? Should the existing workflow just have been made more easily accessible? This could perhaps have been done by enhancing the old UI that violated numerous heuristics of good

design, and perhaps by adding guided functionality such as a wizard. But I doubt this would have solved the original user need the effort started with: working with quizzes directly, with as little initial investment of time as possible. The design I proposed provides a learning path from basic to advanced usage, using progressive disclosure. But using a wizard at first would contrast too much with the more advanced usage of question banks. The learning curve of moving from using a wizard to the more advanced usage of the Question bank would be almost as steep as the learning curve of the original UI was.

Both models were designed to coexist in the new UI. But the usage scenarios envisioned for these two models were quite different and separate, and in design decisions of the details of the UI, even conflicting: To a degree, the two mental models available are mutually exclusive. If a novice user simply creates questions in a quiz directly without ever using the Question Bank, they might in principle assume that the Question Bank has no necessary role related to quizzes. In reality all the questions also end up in the Question Bank. So if such a user wanted to start organizing their questions or using random questions, they would need to unlearn this assumption they may have made.

5. New funding, interviewing users

I still saw an interesting design challenge in the Quiz editing UI, but saw that I could not continue the work while being funded by UTA. This chapter describes the course of actions of Spring 2008, when I worked to expand the effort to a separate project with funding and user research of its own.

5.1. Acquiring funding

In spring 2008, I applied for and was accepted in Kesäkoodi[26], a Finnish open source effort much like Google Summer of Code, to work on the Quiz UI during summer 2008. I had introduced the need for the effort the previous Spring, so the questions I had been posing about the existing UI were known to the community and to Hunt, the Quiz module maintainer. I thought the initial push had been done, so perhaps I could continue discussion to persuade the community about the need for some UI level work.

The Kesäkoodi organisation required that in order to apply, I would need to have found a someone in the open source project who would agree to be my mentor. I contacted Hunt with my application, asking him to act as the contact person for the project on behalf of Moodle. To make it easier to discuss the core problems, this time I suggested a smaller set of changes to the UI than in the first prototype presented to the community. I now focused on the main issue (the lack of support for direct manipulation of quizzes) and on the usability problems of the main editing screen, resisting the temptation to redesign the navigation of the application. I also tried to take into account what I had learned from the comments of the community so far.

Hunt refused, but seemed to have started to see benefits in the proposal. He still pointed out that the way I approached the usability issues was not how things get done in an open source community. My proposition, though no new features were introduced in it, seemed too big to him. Instead, he proposed that I would start with smaller, clearly beneficial changes in varying parts the UI, some of which were out of scope of my proposition. The issue, from his viewpoint, was that I lacked credibility in the community to do something as big as changing a whole UI (Hunt, private e-mail, February 20, 2008). I did not have any code commits in the revision control, after all, from which people could verify that my further contributions would likely be of high quality. To a degree, this also explains the reaction of the community to my initial prototype the year before (Section 4.2.). What Hunt expressed is also in line with the view of Nielsen and Bødker (2007) that in open source projects, decisions are based on the meritocratic nature of the community.

I responded to his e-mail by going through his argumentation. I responded to his thoughts about working on smaller pieces of the UI one by one. I explained him the degree of the design flaws in the then-current UI, and that these flaws did not warrant

[26] http://www.coss.fi/en/kesakoodi

just making small changes, but called for reconsideration of the basis of the design. Ignoring these issues would mean ignoring the needs of novice users (effectively forming an obstacle to gaining new users to the user base of the application). I also told him that making fixes to code here and there (in order to have an impressing commit history) would prove my talent in coding, not in usability efforts, and would take time from the actual usability engineering. I continued to express my willingness to implement the designed UI, but agreed I would work on any terms Hunt saw reasonable.

I now also proposed including the Question Bank in the UI by means of progressive disclosure, though closed by default to not distract inexperienced users. This was to accommodate for the needs of existing users primarily using the Question Bank to manage quiz content. Until then, the Question Bank had been left out and the prototype had focused exclusively on quiz contents (and thus, rather exclusively on the needs of first-time users). I also expressed my willingness to include the community in the process, as well as the fact that I did not expect to have my work included in Moodle core before it was well tested and approved. After this, Hunt agreed to be my contact person for the project, supposedly after also having discussed with Dougiamas, the project lead developer.

The Kesäkoodi organisation required me to provide a definition of usability, justification for the project, a project schedule, plans for risk management, and a list of the most important use cases, along with explanation why these use cases are served better by the new UI. After providing them with a document with this information, I was eventually invited to the finals to present the project to the board of the competition. The five projects consequently accepted for funding, including mine, were presented in news articles on the sites of two Finnish digital media news websites[27]. I started a blog[28] to communicate the progress of the project, as also requested by the Kesäkoodi organization. At this point, I also invited a usability oriented fellow computer science student for coffee, in order to discuss my prototype, and gained plenty of fruitful critique and ideas for iteration. Within the Moodle project, I knew no usability oriented people to really brainstorm with.

5.2. Second introduction of the project to the community

Already in Spring 2007, I had started to see that many of the community members thought that my approach of allowing users bypass usage of the Question Bank and thus changing the conceptual system of the UI, was wrong. Still, I saw that a fruitful

---

[27] http://www.digitoday.fi/data/2008/04/04/Avoimen+koodin+k%E4ytt%E4j%E4kokemus+kohenee+kes%E4ll%E4/20089589/66
http://sektori.com/uutinen/cosskes%C3%A4koodarit-valittu/8186/

[28] The blog posts of the project are available at http://www.pilpi.net/software/moodle/tag/kesakoodi/ at the time of writing this.

compromise could be reached between the seemingly conflicting goals. Also, some community members had so far seen the same needs I did:

> "Most of all, I disagree with comments that we simply need better documentation for the quiz as a solution. Documentation is fine, but the interface should not require any reading in order for a new teacher to make a quiz. I have taught numerous teachers how to make a quiz in Moodle 1.3–1.7, and they are invariably confused unless I walk them step by step for over an hour. That should not be necessary." (Hinkelman, 2007)

I went on with a bias for the users I had started designing for. I adopted this bias partially as a result of also the stand I felt I needed to take in the discussion in the Moodle community. That is, I needed to defend the novice users to community members with the opposing bias. I saw it as my challenge to convince the community that the issues in question indeed exist, to get a discussion going about solutions. Many community members, apparently also rather experienced users of the Quiz module, seemed to not have so far wanted to admit the existence of the kinds of users, the needs of whom I wanted taken into account.

After having gotten Hunt's approval of the project in mid-March 2008, I presented to the community[29] the new iteration of the design (Figure 10), still resembling the version of Spring 2007 (Figure 12) quite closely. I included a link to a document[30] in the documentation wiki, giving responses to the most relevant arguments against the previous proposition. I also presented the change I planned of turning modes explicit by means of adding tabs (Figure 8 in Section 3.4.).

On my request, Hunt (2008b) commented on the discussion thread. He confirmed our previous discussions, and called the project a “worthwhile experiment”. He also emphasised the importance of consistency with the rest of Moodle, and noted that the project would only affect the UI: database structures or other “under-the-hood” details would not change. He also stated that when the implementation would be done, the decision whether to include it in Moodle core could then be made together.

All of these statements can be seen as factors helpful in persuading the rest of the community to support the effort. At this time, most of the responses were positive. There was discussion, though, about how random questions were to be understood, stemming from the fact that I was still proposing to change the way the conceptual system is used and communicated in the UI. Also some opposing voices remained. (An author of such a voice commented the following in another Moodle discussion forum, apparently not knowing the status of my project at the time. The underlining in the quotation was originally a link to my original redesign proposition of 2007[31].)

---

[29] http://moodle.org/mod/forum/discuss.php?d=92797

[30] http://docs.moodle.org/en/index.php?title=Development:Quiz_UI_redesign_prototype_presumptions&oldid=33722

[31] http://moodle.org/mod/forum/discuss.php?d=72828

> “Lots of teachers just want to make quizzes and don't want to be troubled with the question bank. If we cater to them, we create an interface that hides the question bank. Unfortunately, you can't hide it, as this failed experiment in quiz UI redesign amply illustrates.” (Isner, 2008)

Despite these differences, I believed that the new UI could cater to both the users who stress the importance of the Question Bank, and those, to whom simple quizzes are the main resource. I believed that no matter how good practice efficient usage of the question bank was, imposing the entire conceptual model on novice users would result in Moodle losing many users.

5.3. Finding teachers and support staff for interviews & usability testing

Having secured funding at the end of March 2008, I decided that I need to learn more about the users I am designing for.

I knew about the Finnish Moodle interest group, Moodle-rinki, from my work in the Tenttis project. I decided to go in one of their meetings to present myself and to tell them I would like to find teachers and support staff for interviews or later on, for usability tests, and that this would facilitate Moodle user interface design.

While active development of the Tenttis project was finished at this point, I was still maintaining it and thus took part in related meetings in the university. Representatives of Finnish universities visited UTA to learn about the solution, which provided me with yet more opportunities to recruit test participants. As my mother is a teacher in a Finnish higher education institution, I also used her connections to recruit some teachers. There was no incentive to participate involved except for the opportunity to influence Moodle development.

5.4. Content of the interviews and results

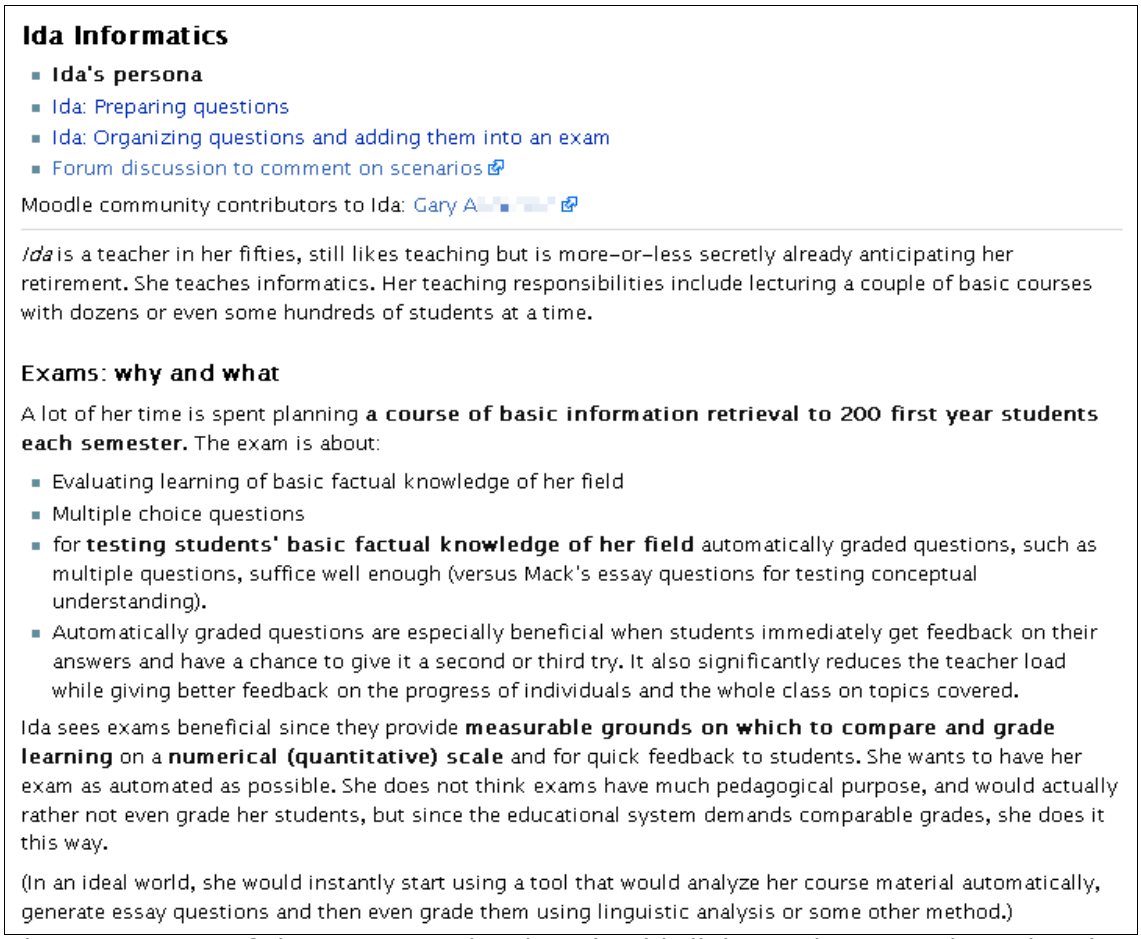
I interviewed four Finnish teachers in May 2008 in a Finnish institution of higher education, as well as two members of the support staff in the LTC in UTA.

I started the interviews by trying to find out the context in which creating exams/quizzes takes place, such as a course in a given faculty, followed by questions related to the actual circumstances of quiz building (see Appendix 1). My goal was to create at least preliminary personas and scenarios based on these interviews, despite the small number of interviewees and the limited one hour time frame allocated for each interview.

I recorded the audio of these interviews and transcribed the recordings very informally – noting roughly all the things the interviewees said, but without aiming to be precise about exactly what words they used. Afterwards, I used these notes and my previous understanding about the usage of the Quiz module to construct personas and scenarios. These described different needs and goals teachers have related to exams and quizzes. The personas (Figure 13) were composites of different interviewees, bundling

similar usage styles into one persona. They did not try to model any particular interviewee's characteristics precisely, but rather served as design tool to guide design. I gained a fair amount of high level insight into how teachers see the usage of exams/tests/quizzes, some of which was new. I formulated this understanding into two bits of documentation, published in the Moodle Docs documentation wiki:

- A document listing the goals of teachers that seemed to not have been addressed, along with some suggestions on what could be done to address them, both in the then current project and in the future[32]
- A series of documents describing personas I had created, and related scenarios[33]



Ida Informatics

- **Ida's persona**
- Ida: Preparing questions
- Ida: Organizing questions and adding them into an exam
- Forum discussion to comment on scenarios

Moodle community contributors to Ida: Gary A[redacted]

Ida is a teacher in her fifties, still likes teaching but is more-or-less secretly already anticipating her retirement. She teaches informatics. Her teaching responsibilities include lecturing a couple of basic courses with dozens or even some hundreds of students at a time.

Exams: why and what

A lot of her time is spent planning **a course of basic information retrieval to 200 first year students each semester.** The exam is about:

- Evaluating learning of basic factual knowledge of her field
- Multiple choice questions
- for **testing students' basic factual knowledge of her field** automatically graded questions, such as multiple questions, suffice well enough (versus Mack's essay questions for testing conceptual understanding).
- Automatically graded questions are especially beneficial when students immediately get feedback on their answers and have a chance to give it a second or third try. It also significantly reduces the teacher load while giving better feedback on the progress of individuals and the whole class on topics covered.

Ida sees exams beneficial since they provide **measurable grounds on which to compare and grade learning** on a **numerical (quantitative) scale** and for quick feedback to students. She wants to have her exam as automated as possible. She does not think exams have much pedagogical purpose, and would actually rather not even grade her students, but since the educational system demands comparable grades, she does it this way.

(In an ideal world, she would instantly start using a tool that would analyze her course material automatically, generate essay questions and then even grade them using linguistic analysis or some other method.)

Figure 13: One of the personas developed, with links to the scenarios related to the persona

Most of the new things I learned in the interviews were clearly out of the scope of this project – the resources of the project were divided between user research and implementation, so creating new functionality was out of the question. The different levels of additional metadata for questions is a good example of such a requirement I could not implement. I was mostly limited to using the existing data structures, most changes being on the level of the user interface. The addition of a status bar into the

[32] http://docs.moodle.org/en/index.php?title=Development:Quiz_UI_redesign_scenarios_-_conclusions&oldid=38953

[33] http://docs.moodle.org/en/index.php?title=Development:Quiz_UI_redesign_scenarios_-_Scenarios_index&oldid=38959

quiz editing screen was the only change inspired and supported by the interviews (see Section 3.4.).

Also the notion of creating quizzes, with the focus of not so much testing, but of just helping students to practice, changes the pedagogical approach to creating a quiz a lot. In the context designing an application aimed for editing quizzes, however, the difference of working with a test or with a a self-evaluation quiz is apparent in the selection of question types and their parameters, and in determining a grading scheme. Both of these were out of scope for my project.

5.5. Expanding personas and scenarios with the community

There was also a secondary goal of communicating the lessons learned in the interviews back to the community. I wanted to try using personas and scenarios as a communications tool to raise awareness of the variety of needs related to the Quiz module in the community. Also, I wanted to gain feedback about the personas in order to make them even more realistic, and to fill in what I had missed in the interviews.

In the forum of the Quiz module[34], I proceeded to ask the community for feedback. To further engage people to tell about how they work with quizzes, I attached to the forum message a video[35], which I had recorded and placed on YouTube. I asked people

- to find a persona that matches how they work with exams,
- to study the scenarios related to that persona, and
- to comment in the forums about differences between their scenarios and the persona's, or if the persona lacked something.

They could do this by responding in the forum, or by directly editing the personas in the Moodle documentation wiki.

Between Spring 2008 and March 2009, six people from the community contributed to the scenarios I gathered originally, describing how they use exams/tests/quizzes in the persona(s) they thought best matched their usage. Two community members edited only the wiki pages directly (changes available in the wiki history logs), three commented only in the forum thread, and one community member did both. However, the consecutive questions, which I asked to get further details relevant to understanding the usage, were not answered. In this respect, the face-to-face interviews I had done first were more effective in getting relevant data.

Much of the feedback in the forum would have required more analysis and aggregation than I had resources for in this project. Also, some of the personas in the wiki were added to so prolifically that their readability suffered, and restructuring the understanding in those additions to be more useful was, again, not allowed by the time available. In terms of understanding the users, the discussions we had with the developer community during the project affected design decisions a lot more than these

[34] http://moodle.org/mod/forum/discuss.php?d=99351
[35] http://www.youtube.com/watch?v=hyfLOSadJW4

efforts related to personas and scenarios. Still, the additional comments I received helped complete the personas to a degree. Some new personas were created as a result of community feedback, too.

Similarly to what is reported by Nielsen and Bødker about their usability efforts in the TYPO3 project (Nielsen & Bødker, 2007), my attempt of creating personas did not provide very promising results. I did learn about some important aspects about the process in which teachers build quizzes. Most insights that actually changed the design during the development process did not result from the data I gathered in interviews though. Rather, they came from usability tests and from inside the community, who provided feedback for the different iterations of the design. So in a sense, one can say I was being the most user-centered when I though I wasn't: when I was not following any particular methodology, but just watching people use the design, and discussing it with the community.

The understanding formulated in the personas and scenarios could still be used in future projects. Thus far, interest in using them has not been documented in the forums by anyone actively involved with Quiz development.

6. Implementation and usability testing

Active work with the scenarios and community discussions about them lasted until the end of June 2008, simultaneously with usability testing with a mockup click-through. After that, it was time for implementation and usability testing with the implemented UI, and finally for discussion about if and how the work could be incorporated in the Moodle 2.0 codebase.

6.1. Usability testing with the mockup click-through

In May and June usability tests, I used a mockup click-through, created with OpenOffice.org Impress 2.4, which provides click target areas for moving between slides[36] (Figure 14).

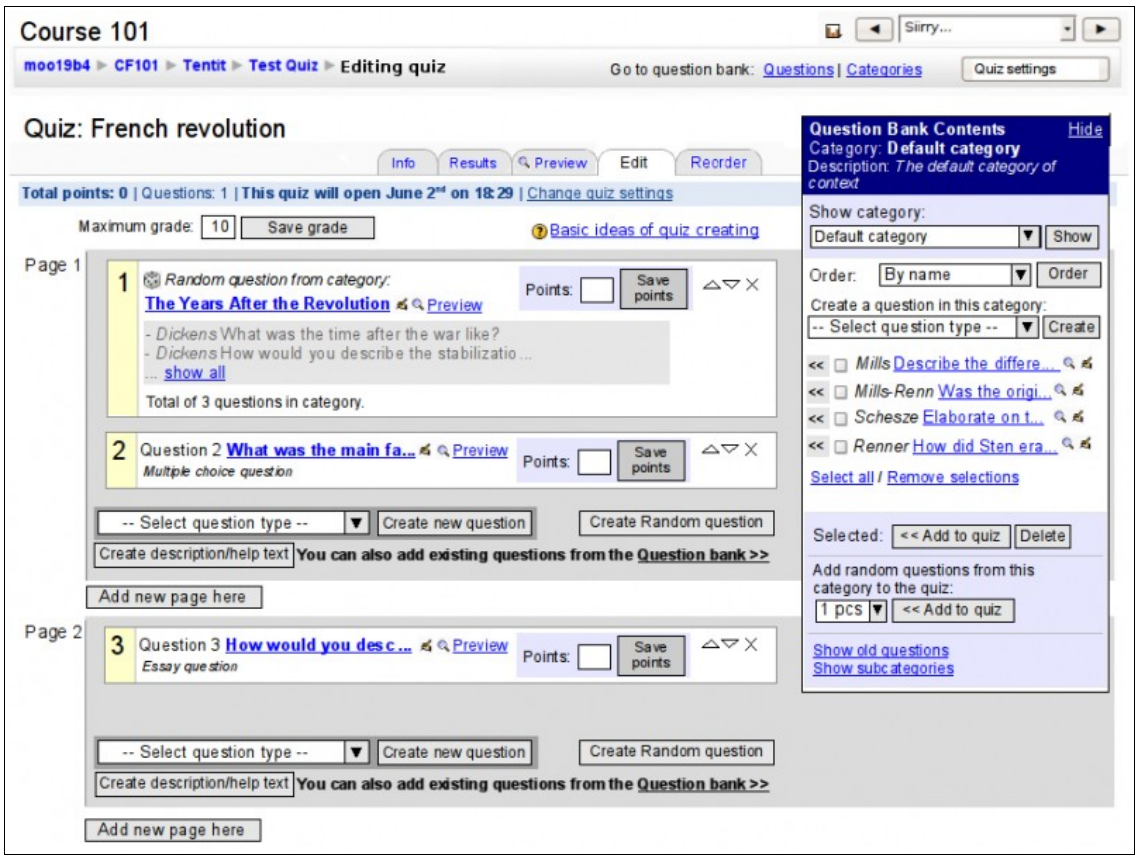


Figure 14: The prototype of the click-through usability test (with Question Bank visible).

Functionality was mostly limited to navigating through a predefined route, and the test participants (5 in May, 3 in June after another iteration) could not use the keyboard during the tests. Still, the tests provided plenty of usability bugs and other observations to iterate the design with. The test tasks dealt with adding questions to quizzes, previewing questions the way they look to students, and editing questions. The test

[36] It could work better for the purpose though. More information available at http://www.pilpi.net/software/moodle/2008/05/27/having-openofficeorg-impress-not-change-slides-when-clicked/

tasks employed are listed in Appendix 3, and recruitment of participants also for this test is described in Section 5.3.

I used the easy-to-approach usability test protocol presented by Krug (2000) as a reference, to make sure I would not forget any important details about doing ”guerilla style” testing. I had some experience of paper prototype testing and of laboratory usability testing, so the purpose of this was to mainly provide a checklist.

I went physically to the educational institutions that had agreed to provide participants for the tests, and they provided a room or a classroom for doing the tests. I took notes and recorded the audio of the usability tests with the voice recorder of an MP3 player. I had no assistant while conducting the test: while observing the test participants' behaviour, I also ran the test. It is, thus, apparent that during the tests I likely missed some cues, which might have helped interpreting the results. Each test (the test tasks and the UI) was in the native language of the participant: most in Finnish, one in English.

A useful exercise, had there been resources, would have been to test also the old UI, comparing the advantages and disadvantages of the new design to the old one. Also, the 'Order and Paging' (see Figure 11 in Section 3.4.) screen was left with very little usability testing due to lack of time.

With some of the participants I failed to remember to mark the participants name on the page I wrote the issues down on; alas, with some of the issues, I was not sure who it was related to. I considered it not worth it to listen to the audio recordings just for this (at least not at this point). It was the most important to know whether something has been confirmed an issue, not whose issue it was.

This testing revealed 32 separate issues with the prototype[37]. Examples:

- A user did not know which was the current category the Question Bank window was showing, so the Question Bank layout was changed such that the category name was displayed more prominently.
- A user had difficulty figuring out how to add a “Description” (i.e. text label) element to the quiz using the “Select question type” menu, so a separate “Add description/label” button was added. (This change was discarded at a later point, as Hunt thought that having too many buttons is also harmful. After my project had finished, he implemented[38] an alternative solution in February 2009, which was never usability tested, to alleviate the discoverability issue.)
- If a user wanted to change the number of points a question was worth in a quiz, they had to click the ”Save” button next to the field for entering the number of points. Some users assumed that this button actually saved the entire quiz, so the

[37] http://docs.moodle.org/en/index.php?title=Development:Quiz_UI_redesign_-_prototype_testing_report&oldid=38689

[38] http://tracker.moodle.org/browse/MDL-18355

button label was changed to ”Save points”. This change is already visible in Figure 14.

6.2. Community discussion and implementation

After the usability tests with the mockup click-through, I presented the last iteration to the community for comments[39], and started the actual implementation phase. This was the first time I got comments about individual design choices of the UI, albeit from only two community members.

When the implementation was nearly completed, I presented the functional UI to the community[40]. I ran Moodle 1.9 with the UI modifications on my personal website and asked for comments to it in the Quiz forum. I got comments from five people, regarding things like making the UI more fluent with JavaScript, as well as suggestions for changes to various about how new the UI works. I asked people who commented to provide for scenarios, in which their suggestions would be useful, and I gained further understanding. Still, many of the requests were either technically too time consuming to implement, or seemingly corner cases that would compromise the UI for the assumed majority of users.

6.3. Usability test with the implemented application

In mid-August 2008, it was time for final usability testing (with 8 teachers as participants) with the actual application (implemented into Moodle 1.9 codebase). This resulted in further minor changes in the UI[41].

To further reflect my test tasks against real uses of the application, I asked participants to optionally bring their own content. I excluded actual user-defined tasks as discussed by Cordes (2001) for practical reasons though – running the test on my own seemed challenging enough without additional variables. Ultimately, only one of the test participants expected to have her own exam done in the test situation, the questions for which she brought with her. In this case, I decided that the actual usability test I had designed needed to be cancelled and the participant created her own exam with the application. The data she provided was not compatible with the test tasks I had constructed, and since the participant wanted to get the quiz done and walk out with the results, that took the entire time of the test session. I continued to observe the participant with minimal involvement on my part. The user succeeded well with these independently defined goals, but obviously this test did not provide results directly comparable to the other tests.

After this last usability test, I wrote an executive summary of the results and suggestions for improvements, also reporting the issues that had been solved since the

---

[39] http://moodle.org/mod/forum/discuss.php?d=100367

[40] http://moodle.org/mod/forum/discuss.php?d=102062

[41] The test tasks employed are listed in Appendix 4.

previous round of usability testing, and those that still remained[42]. The issues found[43] were related to users not noticing the question bank tool and other elements of the UI, the users not learning the conceptual model of Quiz during the test sufficiently to carry out the more complicated tasks, and so on.

6.4. Final round of comments

At the end of August, Hunt requested the community for final comments about whether the new Quiz UI should be included in Moodle 2.0[44]. There was still some discussion about whether the UI is ready, but overall many users agreed that the change was for the good. Some users were still questioning the overall conceptual structure of the new UI, defending the structure of the old version of the UI. I responded to their argumentation, but they did not seem to indicate having understood the reasons that lead to the new UI. Finally, Hunt decided that the work to incorporate the new code into Moodle HEAD, the main branch revision control system of Moodle, should begin.

The project funding ended at the end of August. Alas, I finished the actual implementation with the support of a Finnish study grant. After a period of polishing the code and converting it from the codebase of Moodle 1.9 to the codebase of Moodle 2.0, the new UI made it into Moodle's official version in November 2008[45], almost two years after the first discussion with the community.

Visiting the Moodle Moot '09 conference in April 2009 in Loughborough, UK gave me a chance to meet members of the community face-to-face. Of particular interest was a discussion with a member, who had given some feedback on-line. Being face-to-face enabled us to go through his feedback thoroughly – the discussion had been a challenge online and the misunderstandings numerous. I also discussed the new UI with Moodle's original author and the leader of the project, Martin Dougiamas, who also thought that a change in new UI may be required: The main change suggested reflects a conflict between the persona the old version was designed for, and the new one that compromised existing experienced users' needs.

So a need to change an aspect of the UI was identified, but since the project was finished there were no resources to do actual user testing that I saw required before making non-trivial changes to the interaction. I integrated further responses to the critique I had received, along with remaining open questions, in a document[46] I had

---

42 http://docs.moodle.org/en/index.php?title=Development:Quiz_UI_redesign/usability_testing_of_August_2008/Executive_summary&oldid=42564

43 http://docs.moodle.org/en/index.php?title=Development:Quiz_UI_redesign/usability_testing_of_August_2008/Issues&oldid=42533

44 http://moodle.org/mod/forum/discuss.php?d=103869

45 http://moodle.org/mod/forum/discuss.php?d=103869#p486607

46 http://docs.moodle.org/en/index.php?title=Development:Quiz_UI_redesign_prototype_presumptions&oldid=54508

earlier created in the documentation wiki, and reported the situation to Hunt and Dougiamas.

The Kesäkoodi organization had few requirements in addition to asking projects to report progress in development blogs roughly weekly. After the summer, they did require projects to present their results in the Openmind 2008 open source conference. As I was at that time doing an Erasmus student exchange in Metz, France, I could not be present, but instead created a Youtube video[47] explaining the project's phases and results so far.

6.5. Changes in Moodle affecting the UI after the end of the project

The release of Moodle 2.0 is being planned in autumn 2010[48]. That makes over 2 years since last major release: the maximum time between releases until version 1.9 has been one year [49]. After my development efforts ended on Quiz, there have been quite a few changes (Figure 15). Community members have also reported further issues with the UI[50], and some of them have been addressed[51].

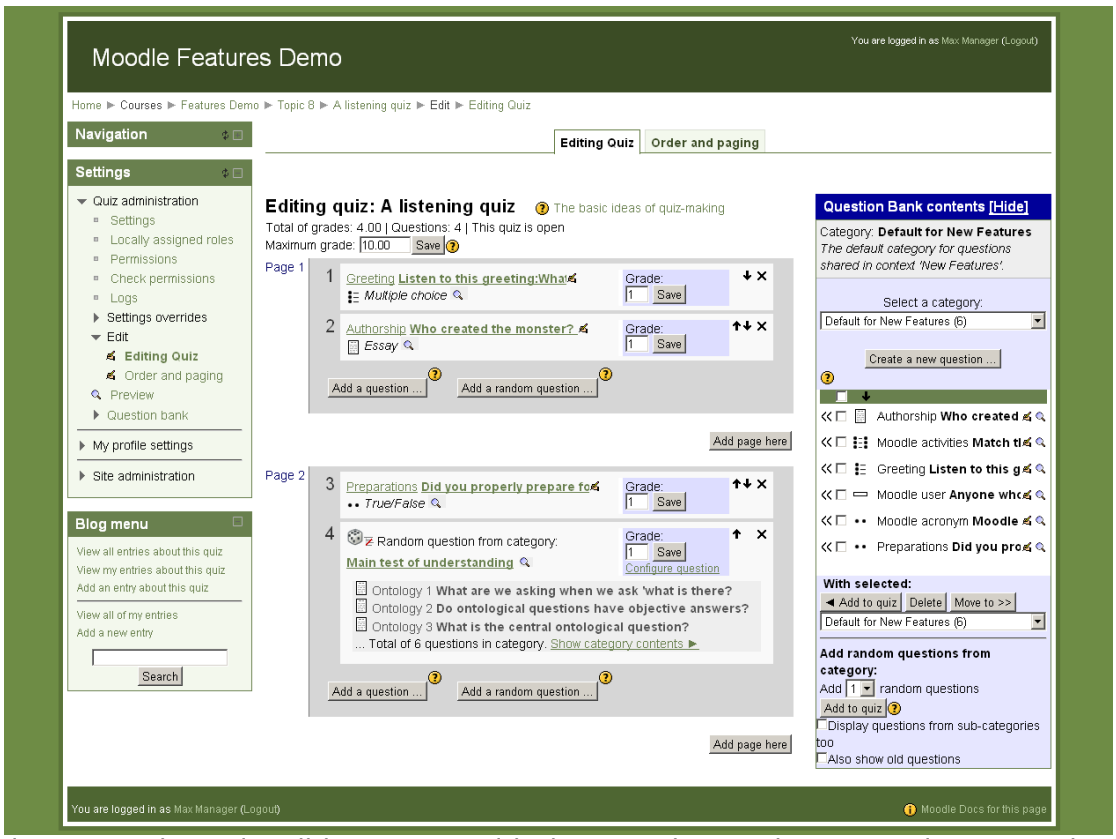


Figure 15: The Quiz editing screen with the Question Bank open, as it appears in a Moodle 2.0 preview release on May 22, 2010. (Screenshot taken on a relatively large 1280×1024 resolution monitor.)

---

47 http://www.youtube.com/watch?v=YCV_MRbulhA

48 http://docs.moodle.org/en/Moodle_2.0_release_notes

49 http://docs.moodle.org/en/Moodle_version_history

50 For an example, see: http://tracker.moodle.org/browse/MDL-21358

51 An example of a fixed issue: http://tracker.moodle.org/browse/MDL-18173

After my work got finished, changes have been introduced also to the overall navigation scheme of Moodle[52]. The sidebars of the new navigation scheme – quite similar to the navigation scheme I proposed in my original prototype (Figure 12), apparently by chance – take up so much horizontal space that the space left for the actual quiz content is quite small.

On the other hand, Moodle 2.0 has a feature at least with some themes to minimize the blocks visible on the left to a dock (not visible in Figure 12), so users may be able to restore a wide quiz view at will. It remains to be seen what the final form of the UI will be like in Moodle 2.0. The final test for the UI will be the reaction of the users in the Moodle forums, when Moodle 2.0 finally gets released.

[52] Related planning documentation:
http://docs.moodle.org/en/Development:Navigation_2.0#3._Use_blocks_for_most_navigation
http://docs.moodle.org/en/Development:How_the_quiz_navigation_should_work_in_Moodle_2.0#Clean_up_the_quiz_edit_page

7. Usability in the Moodle community of developers and users

The Moodle OSS project is interesting in that it was started with a clear focus on the target domain in mind. The software is built to support social constructivist learning, while remaining flexible enough to support other conceptions of learning. Regardless of this focus on the users' world, the user experience has major shortcomings. According to my experiences in actively interacting with the community up until today, developers of the community often lack a sufficiently fine-grained understanding of the goals and the usage context of Moodle users, to make informed design decisions. Also, there is an apparent lack of understanding about usability or about heuristics, such as the well-known 10 Heuristics for User Interface Design by Nielsen (2005).

In the Quiz UI project, the community was caught off guard. Looking at the initial discussions, it seems that there simply was no one in the community who would have been capable of taking in the feedback I was giving of the issues in the UI. There likely were also communication problems related to the high-level issue I was presenting. Regardless, the initial responses were simply rejective to the ideas that were eventually accepted.

In this chapter, I will discuss the project and its end result in Section 7.1., after which I go through factors of the Moodle OSS project that relate to usability work, and to understanding the users, their goals and their context of use.

7.1. Lessons learned in the Quiz UI project about user research

The Quiz UI project also meant a lot of learning for me in terms of doing interviews, creating personas and scenarios, and communicating that work with an open source community. There were quite a few challenges, as well.

As the schedule of the project was tight, I was impatient to lock down design decisions, and in some cases this happened without sufficient justification for these decisions in the user research I had done. I wanted to have some sort of an idea about whether or not my fix would help, so I made small changes to prototypes even between participants. So in some cases, a specific detail in an iteration got even fewer test participants than originally allocated. Mostly the UI stayed the same within an iteration, though.

Though I gathered some relevant information during the interviews, gaining a deeper level understanding would have required more time: As a critical difference between user research advocated by UCD and my research, I interviewed users instead of observing them working in their realistic working environments. At the end of the day, it seemed that I learned more about actual user needs in usability tests, in informal discussions with the support staff in my university, as well as in the discussions in the Moodle community, than in the actual interviews. It seems in a sense that I was doing the most user-centered design when I didn't think I was, and not doing it when I was

trying to do it more formally. Different ways of applying user-centeredness in an OSS context are discussed in Section 8.1. in more detail.

Although I asked the community for feedback about the personas, more active participation could probably have been expected if it had been made easier for community members, especially for teachers. For example, a questionnaire asking explicit questions about different aspects of quiz making could have helped. In future usability efforts, it is important to make understanding about users more explicit also in the online community, as I propose in Section 8.6.

The Quiz UI project might have never gotten initiated to the extent it did, without the input coming from the real world teachers, who supposedly are not active members of the Moodle community. There was evidence of teachers having issues with the UI also online. However, the users having the most serious issues – those who did not want to become users of the old UI – likely also never report their difficulties in the forum.

Usability testing can also be used for measuring the usability of the application. To determine success or failure of a usability effort, usability goals need to be set: measurable targets of user performance with the user interface. According to Cordes (2001), these goals can then serve as acceptability criteria for the software. Such usage of usability tests is in a sense similar to how unit tests are used for determining the acceptability of the functional operationality of software (see Section 8.4.). Due to the nature of my project and the status of usability practice in the Moodle project, measuring usability against such criteria was not possible. Such goals have not been determined in Moodle's development process. I could have measured the time it took for the test participants to perform the usability test tasks, and as I recorded the audio during the test, approximate measurements could still be made to support future work. For now though, the main purpose of the tests was to evaluate the functionality of the UI subjectively, and to fix any barriers or "usability bugs" found.

At the end of the project, I made recommendations for further developing the Quiz module outside of the scope of the Quiz UI project[53], based on the interviews and community discussions in the project. These suggestions included, among others:

- taking the direct manipulation in the UI yet further
- making quizzes editable even when students have made attempts at them (currently not possible, but planned)
- introducing additional metadata for quizzes and questions, mainly to support teachers' workflow of iteratively developing quiz questions
- fixing the novel behaviour of the tabs in the Quiz module (being addressed in Moodle 2.0 since tabs are being removed).

[53] http://docs.moodle.org/en/index.php?title=Development:Quiz_UI_redesign_scenarios_-_conclusions&oldid=65560
http://docs.moodle.org/en/index.php?title=Development:Quiz_Usability_portal/Where_to_go_next_with_Quiz_UI%3F_Autumn_2008&oldid=42569

The old Quiz UI had been designed in an apparent system-focused manner (Section 8.2.): rather than primarily bringing to the forefront the data users are interested in, it focused on the data structures (that supposedly reflected the data central to users). Still, after the project I have come to the conclusion that the issue with the usability of the Quiz module was not that the developers were developing for themselves or that the software was not designed for *any* of its users' needs. Rather, the software had been made to support a wholly different user group than the one I introduced in the new design. The main merits of my effort were:

- advocating for the users who required a more directly approachable UI,
- the usability testing I did and made known in the community, and
- the usage of several design heuristics first intuitively used based on my previous experience, and then explicitly to justify design decisions.

As a summary, the choices I made have been disputed and I have learned a lot about open source usability work and about the user needs of this particular UI during the course of the project. I would not create the exact same design if I were to start designing it now. The principal doubt regarding my design in hindsight is exactly what I originally believed to be its strength: it tries to serve two different primary "personas" simultaneously, thus compromising it to a degree for both of them. On the other hand, no one in the community has so far questioned my approach as a whole. The user needs met by the new UI were not addressed before.

7.2. Observations about OSS usability work in the Quiz UI project

The beginning of the Quiz UI project was the most challenging part of the project – getting the work started in the first place. It was in part also due to the fact that I was not very aware myself how well software engineers in general understand usability thinking. Also, looking back at my writing, I was myself overwhelmed by the complexity of the issues, so the initial proposal itself was lengthy and not as easy to approach as it could have been[54].

It was clear that most my argumentation was being ignored in many of the responses received. It also seemed that the people with whom I was discussing approached the entire Quiz module from different premises than I did. I will discuss conceptual differences between usability practitioners and developers in Section 8.2.

It is widely recognized that usability work in open source is demanding. “Vicious circles” seem to form easily in an OSS project that does not already have usability expertise from the conception of the project: Usability contributors often have difficulties in gaining credibility and becoming contributors in existing OSS projects. Section 8.3. introduces different approaches for making it easier for usability practitioners to work in an OSS project. The effectiveness of the work of usability

[54] http://docs.moodle.org/en/index.php?title=Development:Quiz_UI_redesign_prototype&oldid=37642

practitioners is highly dependent on the existing culture of the project, and on whether and to what degree usability work and user goals are embedded to the project vision. Only if a sufficient number of usability-aware people first build momentum for usability in an OSS project (see Section 8.2.), will it become easier for further usability practitioners to approach the project.

During the work it became clear that whenever someone did not understand or agree with my design plans, I needed to repeatedly explain the design decisions I had made, from the point of view of the other party. There appeared to not be many people in the community who had first-hand experience on usability testing, for instance. So although my testing efforts were welcomed, nobody in the community actually asked for them, yet alone for any inquiries for background information such as the interviews I did. Also, there was no feedback on the conclusions I had made from the interviews or the usability testing itself. Instead, the criticism I got, though invaluable, originated in the domain knowledge and personal experiences of the community members who participated in the discussion.

Especially during the initial discussions about the need for the work in the Moodle community, there was discussion about whether my UI proposal fit the existing UI conventions of Moodle. However, such conventions were not actually documented. Even before that, I had asked for documentation on requirements for the Quiz module, and found none existed. Informality of the development process is a part of the nature of OSS (Scacchi, 2002). Still, it is worth exploring collaboration tools that organize information required by design, and are integrated to the informal development processes of the OSS community. This is further discussed in Section 8.4.

The Quiz UI project had quite a few other background factors that made it possible for me to complete the work. I had no pressure to generate income stream before getting the project funding. Before and during the initiation of the effort, I was employed in UTA, the university I also studied in. The Tenttis project that I worked on there provided me with the initial understanding that there was an unmet need for an easier to approach UI in Moodle. Unlike the journey of becoming open source developer, usability work requires gaining understanding of existing user needs, so simply playing around with the software with one's own computer won't get one far. At the time, I had participated on one Moodle course myself as a student, and was not a teacher, so without the Tenttis project it seems unlikely I would have ever seen the Moodle Quiz editing UI. The Tenttis project also provided me with the opportunity to start creating mockups for the design, and to paper prototype test them with staff. Without this initial phase I would not have had a prototype to present to the Kesäkoodi board in Spring 2008. Without the initial discussion of Spring 2007, Hunt, the Quiz module maintainer, would probably not have seen the benefits of the project and collaborated as my mentor in Kesäkoodi.

Hunt's initial response to me in Spring 2008 was that I did not have sufficient experience in the Moodle project to carry out such big changes. In other words, he referred to the fact that as an OSS project, Moodle is a meritocracy where previous evidence of quality code determines whether one is trusted to work further. I sidestepped the meritocracy by stating that the contribution I had to make was not one of programming, but one of usability work. Even if I had had a history of quality code, that would not have proven my skills in usability work. Hunt also proposed making a group of smaller fixes at first, instead of the complete redesign that I was proposing, which exemplifies the fact that in OSS iteration appears in actual small steps in source code, whereas in UCD it is usually seen that iteration is best done before any implementation. Suggesting small, incremental changes might also be seen connected with thinking that usability can be added to a UI after development.

I not only did usability work in the Quiz project, but also implemented my design: the Quiz UI project could not have been completed if I had only stuck to the profession of usability practitioner, which I wanted to develop in. All in all, I had quite weak ties to the company developing Moodle, the Moodle Trust itself, most my resources came from elsewhere. Also, changes to the UI were suggested after the project that caused me to rethink a part of the design, but I no longer had any time nor funding to do further user research, usability testing, or implementation. Eventually, usability practitioners and user researchers will want to concentrate on their work full-time and have a long-term position in the development team such that they can strategically think about the development of the UX. I will discuss issues of usability practitioners work in OSS in more detail in Section 8.3.

7.3. The additional (personal) gains from the Quiz UI project

The Quiz UI project was a potent platform to discuss usability questions with the Moodle community. It also allowed me to promote usability understanding in the community in general and to gain trust in the community, and I have since continued to work on other Moodle efforts from a usability point of view.

Probably due to then having become a trusted member of the Moodle community, in summer 2009 I got to work in the Google Summer of Code program on Moodle, without a specific programming task. The goal was to work on Moodle UI consistency. To this end, I initiated UI guidelines documentation for Moodle, but perhaps more importantly, engaged with the community in various active development efforts and discussed current usability issues[55], while also working to understand the various UIs and their relationships in Moodle.

[55] http://www.pilpi.net/software/moodle/2009/08/17/the-fruits-of-the-summer/

7.4. The education research Moodle is based on

The bright side of Moodle and usability work is that the Moodle community does already live for the users of the software and there is a strong vision for Moodle overall. That vision is on one hand pedagogical, and on another, thrives in making use of new technologies in ambitious ways. Compared to the "typical" open source project and the problems it can have (discussed in Chapter 2), the starting point in the development of Moodle was supporting users other than the lead developer himself. Its entire development process was born from scratching itches of *other people* – from a frustration towards existing learning platforms which did not support what teachers wanted to do (Cole & Foster, 2007). Being a vertical solution, it is a necessity for Moodle that development must take the needs of users into account, and there are teachers involved in the community. The goals of teachers are somewhat shared knowledge in the community.

Dougiamas & Taylor (2003) formulate the research question of a project that contributed towards the development of Moodle, as follows: "How can internet software successfully support social constructionist epistemologies of teaching and learning? More specifically, what web structures and interfaces encourage or hinder participants engagement in reflective dialogue within a community of learners […]". Moodle is designed explicitly to serve pedagogical principles, and focused on social constructionism, relating directly to teachers. A systematic approach to learning about the needs of users appears to have been taken in constructing the core vision of Moodle: "participatory action research, including techniques such as case studies, ethnography, learning environment surveys and design methodologies" are reported to have been used.

Still, the thinking behind Moodle does not relate to Human-Computer Interaction. Though user needs are perceived and are used to drive design, usability has not been an explicit focus of the project, as noted also by Dougiamas, the project lead (Ruffini, 2008). Nichols and Twidale (2003) state that the challenge in OSS usability "is how to enable and encourage far greater participation of non-technical end users and HCI experts who do not conform to the traditional OSS-hacker stereotype". No systematic framework such as Human-Centered Design or Goal-Directed Design (Cooper, 1999) is in use in the Moodle community, in order to ensure that understanding about users has a clear and systematic effect on the user experience of Moodle. According to my experience with the community, usability sometimes tends to to be seen only in terms of "ruthlessly reducing the user-visible complexity of the default environment to an absolute minimum" (Raymond, 1999). That is, usability is seen as a aiming for minimalism, rather than in terms of creating applications that actually *fit the needs* of distinct user groups, the characteristics of which we need to be intimately familiar with. As the application's scope has already been defined, trying to after-the-fact limit the number of primary personas to design for, would likely prove challenging. The fact that

some users are there to give feedback makes the temptation larger, of not doing user research and usability testing – after all, we've already got the users involved. The status quo is that the Moodle community does not know how usable Moodle is, let alone doing user research to understand usage. User behaviour and goals need to be intimately and systematically known to actually improve usability.

Concepts such as personas, perceived as esoteric by most of the community, are in danger of being ignored in discussions about whether “my favourite” features are threatened to be made less prominent. In the Quiz UI project, there were a few people participating in the discussion with strong opinions about how *everybody* should be using quiz: responsibly, organizing their questions into the categories of the Question Bank before using them in quizzes. Effective usage of personas could eventually help clarify such discussions, if they were used as tools for agreeing what kinds of users the software is intended to serve in the first place.

7.5. Determining the level of usability work in Moodle

Nielsen (1994) outlines steps from step one, complete ignorance (”Usability does not matter”) to step eight, usability permeating the development lifecycle. Although the steps assume a commercial enterprise, it can be assumed that something similar might happen in open source projects. If usability advocates succeed to communicate their case, resources slowly start to get directed towards integrating usability practice in the development process. On the other hand, it is hard to define the borders for an open source community. Perhaps even more than in a commercial enterprise, different people in a given project may simultaneously exhibit behaviour of several different steps.

In step two, it is assumed that regular development staff can produce good user interfaces by themselves – Nielsen compares this to the attitude expressed by King Frederik VI of Denmark on February 26, 1835: "We alone know what serves the true welfare and benefit of the State and People." Most development in Moodle is done entirely without rudimentary usability testing or user observation. Alas, much of the Moodle community is on this step.

In recent discussion, an “artisan-developer” model for usability was discussed in the community, proposing that developers should have special skills in UI design and development. However, as this notion focuses on developers' talent, it bypasses the fact that actual research and testing is required to “know your users”. Anyone developing user interfaces without checking their suitability for their audience, is designing to suit the needs of a very narrow group of users, ultimately only for themselves. We can not take what users say in the forums at face value: people can usually not predict their own behaviour to any precision sufficient for informing design. It seems however that open source developers typically believe that research and usability testing is something they do not have the resources for, or do not want to do.

Step three is characterised by the desire to ”have the interface blessed by the magic wand of a usability engineer”, typically after it is already too late to do user research or major design changes. In the Moodle community, I have several times served in such role of a usability engineer that can only help by recommending in the most cosmetic changes. Other changes are often too hard to make for two reasons:

- decisions have already been made in the underlying architecture, such that required changes would require too much work, and
- there are no resources to study the implications of different design ideas.

Step four is the highest one I can place Moodle in with any amount of good will. The step is called ”GUI/Web panic strikes”: There is a sudden urge to bring in usability practitioners to give their advice from the start of a project. There is still no systematic approach of learning about the users. In Moodle, I know of no development efforts that have happened within the Moodle project that have had even as much user research as the Quiz UI project. User performance or other usability metrics are not used. Steps from five to eight are as follows: “Discount usability engineering sporadically used”, “Discount usability engineering **systematically** used” (emphasis mine), “Usability group and/or usability lab founded” and “Usability permeates lifecycle”.

The applicability of these steps to open source projects can of course be questioned. However, Moodle core is developed by Moodle Pty Ltd, a commercial enterprise. They do have influence over much of Moodle development that takes place, and could incorporate usability processes in their development. Still, strategies for empowering also the actual Moodle community to work on usability are critical for making OSS applications such as Moodle truly usable. The number of contributors can be vast, so it does not seem likely that there will any time soon be a large enough professional team of usability practitioners, to serve all the development efforts that occur simultaneously.

Some factors in UI design are modularized with relative ease, such as skinning/themeing the user interface, and internationalization (Nichols & Twidale, 2003). These have been paid considerable amount of attention in Moodle.

7.6. Documentation of development efforts in Moodle

Essentially Moodle is still closed source in terms of design (like its proprietary rivals). None of the higher-level thinking about the design of the system is readily available in a manner that would make it possible for anyone to further develop the UI. Such documentation would be critical so that one could know that when doing a redesign, the user needs that were taken into account in while creating the original UI, are still supported. Documenting any of user goals, conceptual models, workflows, personas and scenarios could provide such constraints to base design on.

Instead, any planned work is documented in a wiki[56] and in a bug tracker[57] and discussed in forums[58]. Typically, to initiate the development of a desired feature, a developer describes the use cases in the community forum of the project, and other users and developers may take part in the discussion, presenting other use cases that might need to be taken into account. So some of the information to support design is still available in the forums, in an informal form. It is not particularly organized in any given form, making it very difficult to discern which features were added to a UI as a result of which discussions, or in which order. Knowledge about such developments would be required to avoid making the same mistakes again when further developing functionality. It is hard for a user experience designer to plug in if he/she has not been a part of the specific sub-project from the start. On the other hand, documentation for users, such as the Using Moodle book (Cole & Foster, 2007) illustrates to a limited degree how different parts of Moodle are expected to be used. Collaboration tools for also facilitating documentation of user goals are returned to in Section 8.6.

Much of the appeal of open source projects for volunteers though, is in the fact that things can be done somewhat spontaneously. People can be invited to just play with the software and to make changes in the code to scratch their own itches. The challenge is to keep the spirit of playing alive in the community, while also having sufficient documentation to facilitate taking into account actual user needs and other software quality properties.

The development roadmap[59] can be useful for finding active development efforts where a usability practitioner could plug in and provide his/her work. I did this on few occasions during summer 2009 and my work was in general welcomed. I further discuss providing usability services to developers in Section 8.2.

7.7. Current development needs for user research data

The place of user-centered design in Moodle seems to be between the two factors prominent in the community: bridging the technology and the pedagogy visions together, and making the link between user goals and what technology facilitates, more visible. As teachers are on some level active participants in the development process, the overall selection of tools likely tends to be sufficient to them, and is growing. Indeed, the use cases of Moodle are very varied, and keeping them varied is a goal to the community. To illustrate, groups represented in the presentations of the 2010 Moodle online conference iMoot included universities, Africans, small businesses, enterprises, charities, vocational educators, and so on.

However, the challenge Moodle faces is that although the community wants to design inclusively for everyone, the priorities of whom to design for are not clear. As

[56] http://docs.moodle.org/
[57] http://tracker.moodle.org/
[58] http://moodle.org/
[59] http://docs.moodle.org/en/Roadmap

developers perceive that there are indeed people who want a given feature, they are easily tempted to try to satisfy every feature and every user equally. According to Hunt (2009b), every time he suggests to remove something, someone (assumably in the Moodle community) will offer him a “compelling educational situation where that feature really makes a difference, and [he] will realise why that feature was added in the first place, and why it would be a mistake to remove it”. Moodle is modular in its structure, but keeping unnecessary functionality out, to keep the UI simple, is still a constant challenge, like discussed by Nichols and Twidale (2003). There is, however, a CONTRIB code repository for contributions that are seen as useful but are not wanted in the core Moodle code. This helps in keeping some of the less commonly used features out.

The issue is the most apparent in the long configuration forms, where marginal users and usage scenarios tend to get represented equally to central ones. Figure 16 presents a configuration form of a Moodle core module, which burdens users with a plethora of options. This form is shown always when an activity based on the module is taken into use, and it can later be accessed to reconfigure the activity. Many options in

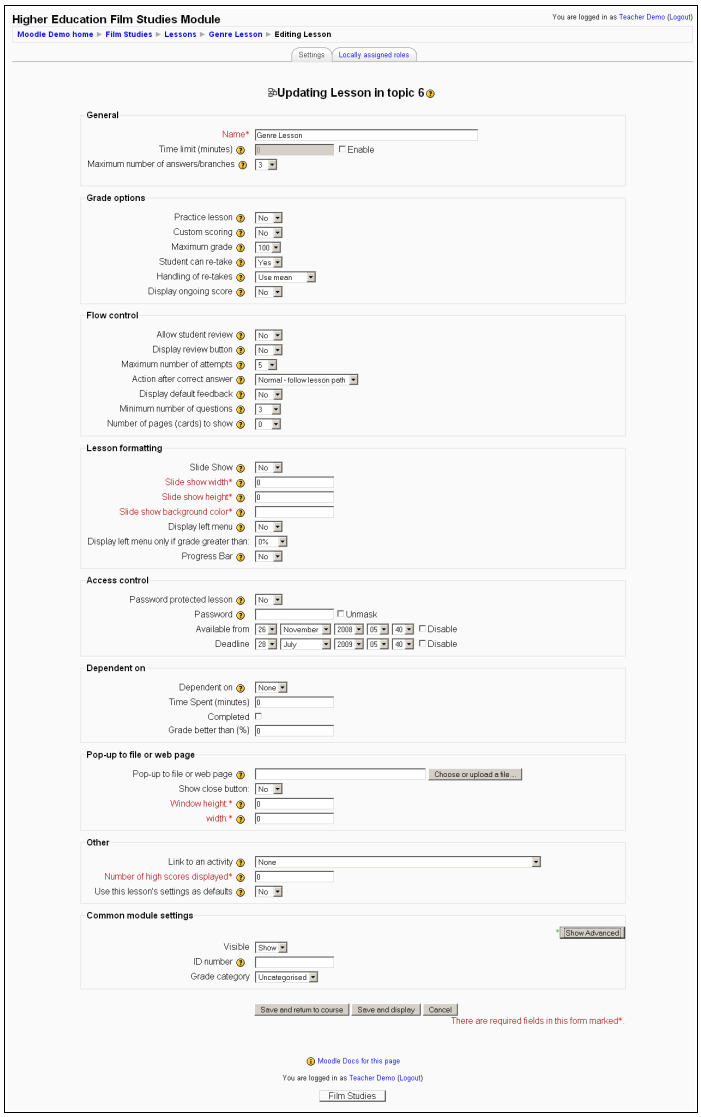


Figure 16: The configuration form of the Lesson module of Moodle 1.9

such configuration forms have reasonable defaults and they may not require changing. However, there is no way for users to know this, so they may feel that in order to use the module, they need to understand all the questions the form is posing them. Some such configuration forms in Moodle do employ progressive disclosure by means of ”Show advanced” buttons, although the benefit of these is unclear. By selecting to hide options based on a criteria often seemingly arbitrary to users, those options may be hidden from novice users, but also from the experienced users the features were designed for. Using progressive disclosure (Nielsen, 2006) requires knowing which tasks are the fundamental, basic ones, in the first place. The problem, then seems to be that progressive disclosure has been used simply to hide some options from novice users, without thinking about the user goals as a whole.

The old Quiz editing UI is another example of the lack of actual user research in the Moodle project. The selection of functionality was close to right, but the user interface was designed exclusively for one type of user, coincidentally not only an experienced one, but a user of a feature set required only by a rather specific user group also in other regards. The fact that other user groups should have been designed for would have been apparent, should potential user groups have been identified before implementation, involving those users who are not in the community. The users who had the need had not come to the community to complain, since it does take a professional to say what exactly is wrong in the UI in the first place – and even they are not listened to easily, as illustrated by the Quiz UI project.

8. Discussion: Usability practice in an open source community

Chapter 2 presented common issues related to usability work in open source projects, and introduced discussion about how they might be approached. In this chapter, I will expand that discussion by examining different solutions suggested in literature, to improve the usability practice of open source projects.

To this end, it is useful to state the goals of the endeavour. Frishberg et al. (2002) list three key areas, where the Human-Computer Interaction (HCI, the scientific field where user-centered design is rooted) community can contribute to the success of open source software (OSS). These areas are:

- usability (of software) to ordinary people/non-programmers,
- consistency in the behaviour of the software regardless of the variety of contributors, and
- accessibility to disabled users (such as Section 508 of the US Federal Rehabilitation Act and the guidelines put forth by the W3 consortium's Web Accessibility Initiative).

Scacchi (2002) states that the requirements developing process in OSS is a process of building a community. Members of the Moodle community, and quite possibly members of also other open source projects, see the community and its learning process as an end in itself. The way open source software gets made, as illustrated by the below quotation, relies so much on the community participating in the construction that also building awareness of usability work should be a core goal for usability practitioners.

> "Martin [Dougiamas] often says that Moodle is not really a development community building a better Moodle, but a social-constructionist learning community, learning, through peer interaction what a better Moodle should be like. He is absolutely right." (Hunt, 2008a)

In addition, private benefits such as learning of community members, are an important factor in motivating the members to contribute and in fostering a community where usability work is viewed favourably.[60]

I will now examine the user-centered design process and other approaches for involving users in design of software, and then go on to discuss the feasibility of implementing them in the context of an open source project. I will discuss existing research aiming to understand open source development processes and possibilities of incorporating usability work into them. Then, in Sections 8.3. through 8.5., the question of fostering a culture to facilitate learning about users and usability work is discussed, both from the point of view of all community members, and from the point

[60] Another benefit this quotation seems to reveal is that the purpose of the Moodle community, at least in the mind of some core developers, is to really work for the benefit of the users. This is something the closed source software development model can make difficult, as concerns about profit often drive enterprises creating proprietary software.

of view of usability practitioners working in an open source community. Finally, I will present some efforts, the undertaking of which may facilitate change in practice.

8.1. UCD and other approaches to understanding users

If a process is introduced for developing usability practice in an open source project, it needs to acknowledge the existing body of usability knowledge. User-centered design (ISO, 1999) provides a candidate model for such a process. In UCD, a clear vision of defining archetype users (personas) and then designing for these archetype users rather exclusively is key. The challenge addressed by user-centered design methods is knowing the users, and communicating their needs in a meaningful way to support making design decisions. In order to create internally consistent designs, this is typically done in in a centralized fashion.

The open source approach is fundamentally different. Knowing the users is mainly based on having domain experts and enthusiastic users in the community, and on them giving their feedback. Barcellini, Détienne, and Burkhardt (2008) discuss people acting as *boundary spanners*, mediators of information between developers and users, in the open source project of the Python programming language. The people in this emergent role are cross-participants of both the mailing list of developers and that of the users. According to Barcellini et al. (2008) this is important since users discussing their usage and personal experience on the users' list alone does not guarantee described needs getting taken into account in design of the OSS. Also Terry, Kay and Lafreniere (2010) discuss core users that provide input into a project, and call them *reference users*, valued by their domain expertise and experience in using the software. Another group of core users in a OSS project is *bleeding edge users*, who use the latest development version of the software and provide feedback. (Terry et al., 2010).

The risk here, from the point of view of UCD is that open source developers rely too much on the users available in the community. There is no guarantee of this being a representative sample of the target users. In other words, the selection of the target users of an open source application seems to happen inadvertently: target users are those who happen to have domain experts to represent them (i.e. whose usage of the application has similar characteristics) in the community. Also, as users typically can not predict their behaviour (Nielsen, 2001). Thus, basing design decisions solely on user opinion guarantees nothing about suitability to actual users. Further exploration is in place: How does taking direct feedback from users as the main source of data to fuel design affect the usability of open source applications?

There are downsides to the user-centered methodology, as well. UCD can be perceived to be too burdensome in an open source community. Doing ethnographies to gather necessary understanding about users can be a lot of work (Cunningham & Jones, 2005). Also, the research methods of UCD are not very collaborative by nature – unlike code commits in a revision control system, the real action of user research is hard to

transfer online into a form in which the open source community might feel they can follow the process as it happens, and participate in it. Trying to introduce usability only by researching users in ways that UCD encourages, and ignoring the informal communication modes an open source community itself fosters, is problematic. Also Terry et al. (2010) suggest that motivations of contributors for working are different in an open source project than in what HCI methods were originally developed for. The methods need to be reconceptualized to fit the OSS culture of practice and the corresponding value system, fostering and taking advantage of the OSS social relationships, since these ”motivate attention to usability issues in day-to-day development” (Terry et al., 2010). UCD being unfit as a process for OSS seems a likely reason why usability practitioners often have hard time gaining trust in OSS communities.

Scacchi (2002) presents the concept of an informalism as both the media, and the subject, of software requirements engineering work in an open source project. Informalisms include communications of the community using computer-based tools, usage scenarios as linked web pages, how-to guides, external publications about the software, OSS websites, bug reports, documentation for users and developers, and software extension mechanisms or architectures. Instead of *formal* and explicit requirements documentation, he states that open source requirements are post-hoc, informal, and implicit, in all phases of the lifetime of a requirement. He states that OSS does not need and probably won't benefit from classic software requirements engineering. Instead of consistency, completeness, and correctness, OSS concentrates on community building, freedom of expression, ease of informalism navigation, and implicit structures of informalisms.

UCD presents a similar formal process for usability work as requirements engineering does for understanding the requirements of software. Requirements engineering also has considerable overlap with UCD in that both aim to define software, albeit from different perspectives. So does open source defy all formal methodologies, or will they start to form within OSS as its practice gets more mature than now?

Some promote approaches to usability practice alternative to UCD. Spool (2008, 2009) calls the attention of the usability community to the risk of accepting methodologies blindly, of following them as dogma. Instead of measuring the results, he says, we focus on convincing people: we are following the right methodology, so the product *must* be good. Instead of UCD, which he argues never worked but lead in people following the process blindly, he encourages what he calls *informed design*, a reward system for any design activity such that the effectiveness of the activity can be measured. Spool states having researched successful and unsuccessful interaction design projects. He claims having found three quality attributes successful projects had in common: a long-term user experience **vision**, a working **feedback** loop, and a

culture of encouraging design failure. Such simple principles might be useful also in the context of an open source project, where methodologies such as UCD appear harder to implement.

> "Vision: Can everyone on the team describe what the experience of using your design will be like five years from now?" (Spool, 2008, 2009)

The point here is that you have a larger goal, and when you take daily baby steps in your design, you can tell whether those steps are towards what you're aiming, or not. This seems to also facilitate consistency, as everyone in the team are supposedly aiming for the same, shared goal.

> "Feedback: In the last six weeks, have you spent more than two hours watching someone use yours or a competitor's design?" (Spool, 2008, 2009)

The usefulness of feedback from actual users is rather obvious. Spool seems to stress the importance of the *experience* of seeing real users use your design – supposedly, in order to remember just how differently real users perceived your design than you did. The reason Spool argues for six weeks as the maximum amount of time one should be allowed to not sympathize with users, is that according to him, after that time memories fade. In an online, distributed development environment, we want to regularly expose developers to perceptions about how users experience their designs.

> "Culture: In the last six weeks, have you rewarded a team member for creating a major design failure?" (Spool, 2008, 2009)

Here, Spool argues that encouraging failure results in a culture of learning from that failure, instead of making contributors ashamed or afraid of mistakes. This also seems to facilitate creativity and trying also experimental design ideas.

Regardless of the validity of Spool's criticism against UCD, his propositions for things to include when designing for our actual users look useful, because they can be adopted seemingly without burdening the team with a heavyweight process. Further examination of the actual research behind this would likely be a useful exercise.

Open source has been considered loosely a form of distributed **participatory design** (Barcellini et al., 2008; Terry et al., 2010). Participatory design is concerned about democracy at work and about democracy in design. In its roots then, its approach to design is quite similar to the ideals of open source (Nichols & Twidale, 2003), where transparency of the development process is a key value. Contrasting it with UCD where the focus is on researching the users so that designers can make better design decisions, participatory design seeks to empower users as decision makers in the design process itself. Researcher-designers are thus expected to facilitate user-designer cooperation instead of representing the users (Iivari, 2009). Ehn (1993) approaches the question of communicating understanding about the intricacies of work, between designers and users, in terms of wittgensteinian language-games:

> “If designers and users share the same form of life, it should be possible to overcome the gap between the different language-games [of users and designers]. It should, at least in principle, be possible to develop the practice of design to the point where there is enough family resemblance between a specific language-game of the users and the language-games in which the designers of the computer application are intervening. A mediation should be possible.” (Ehn, 1993, p. 70)

Within the domain of participatory design there are techniques, supposedly helping both users and designers see “other person as partner” instead of “other person as problem”, recognizing the differences in perspective of different stakeholders (Allen et al., 1993).

Also in the context of an open source project, users and developers have different language-games. Participatory design explores the conception that making mediation possible between those language-games facilitates design and user participation through enhanced communication. Erickson (1996) discusses telling stories, considered perhaps an initial, less formal alternative to scenarios, as a similar "equalizer" of different participants of discussion, since storytelling requires no special skills. This comes very close to the challenge of involving users in development that seems to have a culmination point in open source development. An open source community indeed engages users at a level of discussion. Observation of users in their real working environment is difficult – and sharing those observations in a way that engages the community (such as a video narrative illustrating users' environments) raises privacy issues – and again, is very time-consuming. Thus, the notion of a solution encouraging users to further *participate* in the design instead of being observed is a very tempting one. However, Iivari (2009) states that in the OSS project she studied, users do not have actual decision making power regarding the OSS but are left in a consultative role. This seems to be true of most OSS projects. Ultimate decision making is left to developers only, to whom a very limited amount of understanding about the users of the software is available in the forums of the community (Iivari, 2009). In OSS projects, user involvement is thin from the perspectives of both UCD (no usability practitioners to represent users) and participatory design (the few users who do participate have no true power over the eventual software).

Warsta & Abrahamsson (2003) have studied the similarities of agile methodologies and open source development. Sy (2007) builds a case for agile as an efficient framework in which to employ user-centered design. Further research could reveal whether these approaches might also be applicable to open source development.

So central questions, yet ones without clear answers, seem to include: Is there a way UCD understanding could enrich the natural way OSS projects learn about their users? Could usability practitioners plug in to the community's existing means of communication, finding fruitful points of contact with the community while using

methods favoured by UCD, such as ethnographies, to fuel design? Or should we perhaps concentrate on what seems to come naturally to an OSS community, and only refine user research methods to be based on the open source community's feedback?

One approach is to simply add usability work to an OSS project at the initialization phase of the project. If usability practitioners are integrated with the birth of the project from the start and are key influencers in the project, they may find it easier to integrate their work to the project. However, what is outlined in this chapter starts from the assumption that usability practice is introduced to a more typical OSS project – that is, to one without usability expertise of their own.

The challenge of usability practice in open source projects seems to inevitably deal with two problem areas, the solutions to which are related but separate. I will discuss these two areas in the two following sections:

- Creating circumstances for the open source community to engage with usability matters (Section 8.2.), and
- creating circumstances for dedicated usability practitioners to work with the community (Section 8.3.).

8.2. Creating conditions in an OSS community to engage in usability thinking

In order to make usability thinking and practice pervasive in an open source project, all opportunities must be used for giving community members opportunities to engage with usability issues. Usability practitioners need to take part in the open source community as peers. Understanding the users of the application in question can be explored with them.

As mentioned in Section 8.1., Spool (2008, 2009) argues that encouraging failure results in a culture of learning from that failure. Most open source communities are far from such an ideal in terms of user experience vision, though. Before being able to positively react to failure in design, the failure must be recognized consistently, and there has to already be a culture of design, which recognizes user experience in all its complexity (and even artfulness), as a factor in the vision about the software.

In the following sections, I outline and discuss different tactics usability practitioners and other stakeholders in an OSS community can employ to promote usability work and understanding of factors of user experience.

Make developers aware of what is already being done

Terry, Kay and Lafreniere (2010) argue that OSS developers “often discount their own ad-hoc practices”, not recognizing their own contributions to software usability. Their examples include “obtaining high quality, targeted feedback from a relatively small set of trusted, knowledgeable users early on”, and “graduated testing”, that is, gradually opening up new designs to groups of users increasing in size, catching the most serious errors early. Sometimes the problem is simply that users and developers in an OSS

community are unaware of what others are already doing. Terry et al. (2010) suggest a catalogue of techniques for improving usability, with instructions for how to get the most benefit of them.

Drive change: Motivate developer community to learn about usability work

To make longer term usability work possible in an open source community, it is critical to nurture an environment where members can work on the aspects of usability they find relevant, and to provide opportunities for further learning about how to gain understanding about users. Motivations of different groups of community members need to be discussed within the community: what is their inclination towards usability. As usability is often a relatively obvious core factor in determining the success of an application, the existing beliefs of community members should be understood, about how usability-related issues are best addressed, and the discussion started from there. Terry, Kay and Lafreniere (2010) suggest that the traditional motivations for HCI cannot be assumed in open source projects. They present growth of the user base as an example of an assumed motivation that may not apply in open source projects: a large user base increases the number of bug reports, feature requests, and complaints. If this growth happens uncontrollably, it may not be desired. Their research indicates social rewards, such as praise and positive feedback from users whose views are valued, being more important a motivator than an increase in user base.

In the Moodle community, an artisan developer model of developers taking care of software usability, as a part of their profession, has its advocates (Hunt, 2009a; Marshall, 2009), and open source developers often claim to have an interest for usability. Such willingness to participate needs to be addressed. As long as developers consider usability work a responsibility, it should be possible to challenge them to actively engage in the questions: How does one go about effectively knowing if a UI is good? If it is not, then what are the things we can do about it?

Increasing the attention to users and user-centered design in open source software projects is stated as a goal for OSS usability by Frishberg et al. (2002). Getting the OSS community members and developers in contact with real users is encouraged and beneficial to usability. Still, if getting developers to do simple guerilla-style usability testing is in practice a luxury, engaging with users in more time consuming activities such as contextual inquiries likely needs to be done by someone whose primary focus is not programming or engineering. So, the question an open source community seems to need to eventually address is: Are there parts of the work that developers are not willing to do on a regular basis, but would like to have, say, trained usability practitioners doing them?

Heath and Heath (2010) discuss how to bring about change in organizations. Their claim is that people can be engaged by showing them the problem in a manner that appeals to them emotionally. In the sphere of usability work, demonstrating user

frustration in usability testing sessions is a tried technique for this (see Section 8.2.). In addition, clear, practical steps need to be provided for people to follow so that they can direct their behaviour (Heath & Heath, 2010), empowering them with the knowledge that they are truly serving the users and their peers in the community, by means of creating tools for others beside themselves. Also, Heath and Heath (2010) recommend focusing on the bright spots, the little things that already go right, which can be analysed and used for finding workable approaches that already work (another justification for researching what makes open source tick, see Section 8.7.). According to social constructionism, learning happens best when the learners are actively engaged in constructing solutions (and meanwhile, their own mental models) in groups, ending up creating culture[61]. An OSS project can act as fertile ground for such learning together while constructing the software. Finally, a clear goal has to be set, making concrete the benefits being aimed at (Heath & Heath, 2010). As usability practitioners engage with the community to help the community realize their vision related to usability-related questions, usability practitioners also have a chance to gain trust in the community.

In cases where a community member is asking a specific usability-related question in a forum, giving answers or challenging the participants of the conversation to find answers can be a fruitful way to engage with the community. Recently in the Moodle General developer forum, there was discussion about the usage of dropdown menus/list boxes[62] for binary (yes/no) selection[63]. I took a clear stance about what UI elements should be used in different situations, believing the subject is something in my domain of expertise. However, community members raised several points that lead me to reconsider and expand my original response. If for anyone, this was a useful learning exercise for me, likely creating additional links in my mental model about form elements. In any case, an attitude of giving answers "from usability people in their ivory towers", needs to be carefully avoided as it likely stifles communication (and thus, learning). In the case of Moodle, the expertise of the social constructionists (teachers) in the community would be useful indeed for engaging the community to learn about usability work together.

Creating opportunities for the community to engage in usability practice is important also for creating common ground for discussion. Usability thinking is generally foreign to developers of an OSS community, and for cooperation to be possible, usability thinking needs to pervade the software vision of the open source community. To a degree, developers and other community members need to understand the work of usability practitioners, and vice versa.

---

[61] http://docs.moodle.org/en/Philosophy

[62] Drop down menus/list boxes being implemented using the HTML <select> element

[63] http://moodle.org/mod/forum/discuss.php?d=126481

Usability work as a service: Support active development efforts

Nickell (2002) approaches usability work in OSS communities from a model of usability practitioners offering services to developers. For instance, doing testing and user research in the OSS project, and discussing findings with the developers may help developers see the value usability work can bring in, without burdening them with extra work. This can happen by plugging into efforts in the OSS community, where development is already happening. If a design phase can be included in a current effort, the developers of the effort can see the effectiveness usability practitioners can have in making user interfaces work for users. Potentially they start to see benefits in including usability efforts also in future projects. The project roadmap can be useful for finding such opportunities. Developers stay behind the wheel but get data from the usability practitioners to fuel design.

Terry, Kay and Lafreniere (2010) discuss an OSS project that holds monthly meetings where a usability practitioner "makes herself available to provide expert reviews" on existing or proposed designs. The benefits of the practice are twofold: developers get immediate benefit, and they are reminded about the need to keep usability practice as a day-to-day consideration.

Providing services to existing processes is probably good for getting developers introduced to the idea of cooperating with usability practitioners, but it still leaves developers potentially changing designs at their whim. Ultimately, architecture decisions should be made by developers while user experience decisions should likely be left to usability practitioners, or anyone more actively engaged with both usability practice and the users of the application in question.

Usability testing in a culture of doing

> "Intervening in a community that values results over abstractions demands that usability experts make themselves meritable in a way that resonates with the community. If usability wants to have an impact on the future of OSS development, it must participate in the 'culture of doing' and not just communicate understanding of problems." (Nielsen & Bødker, 2007)

Nielsen and Bødker (2007) see that the TYPO3 open source development community focuses on functional problem solving in a culture of doing, and see this in disagreement with the current usability methods' approach of trying to understand users and their problems. This seems to be the case in other open source projects as well, so it seems apparent that open source should take full advantage of practical usability engineering, which approaches usability practice with a similar mindset as developers are used to: finding issues (typically with usability testing), and then getting them fixed.

The primary way of promoting usability in a culture of doing is usability testing. Demonstrating effectiveness of usability testing has often been suggested for selling usability practice in an organization since its effectiveness is usually obvious to anyone

watching a testing session (Marty & Twidale, 2005). Usability testing typically makes it obvious how dramatically differently various users perceive their designs (Nichols & Twidale, 2005). As mentioned in Section 8.1., Spool (2008, 2009) suggests that **all** members of a software development team should spend at least two hours per six weeks watching users use their own or a competitor's product. Usability testing is a good fit for doing this. Usability testing can show people the existence of usability issues and the simplicity of solving them, and if done well, seems likely to also lower the barriers of communication between the different cultures of user-centered design practitioners and software developers.

In the open source context, an online video edited to show the most “dramatic” moments of the test seems like the most effective way to reach the community's attention. Videos need to be sufficiently short and captivating, so resources and skill are required of the usability practitioner (presumably) doing the editing. The video recording would likely need to be edited to be captivating enough, so that it would actually be watched by a considerable number of developers.

On a practical note, the user's face should be displayed on the video, to make the videos also emotionally captivating. Getting a permission from test participants to publish recognizable videos of them doing usability tests may be difficult. When usability testing demonstrations are started, they should be done at a regular pace, in order to remind people of the usability considerations they need to keep in mind, and keep as much of the community engaged with usability as possible.

In Chapter 2, I listed reasons why usability work is difficult in an open source project. However, unlike corporate environments where people may be “just doing their job” and thus are not interested in doing anything extra, in open source projects people will do things they consider important and interesting. If the ease of simple, informal usability testing can be shown to the community, there is a lot of potential for crowdsourcing usability work. A way to ensure reliability of the test data needs to be established before this, though.

Though usability testing demonstrations are impressive to anyone, it is likely the most eye opening to developers seeing their own designs tested. Of course, it also has the benefit of the developer walking away with a discovered collection of usability bugs waiting to be fixed.

Another opportunity to reach a part of the community (not only developers) would be to arrange a demonstrative usability testing session in a community happening of the open source community, such as Moodle Moot[64] in the case of Moodle (discussed in sections 3.2. and 6.4.).

[64] http://moodlemoot.org/

Thrive for a shared goal despite of conceptual differences

According to Cooper (1999), the goals of software engineers and of user experience designers are often in conflict. By virtue of software engineers' position and what is expected of them, they tend to optimize for short implementation time, elegant software architecture and system performance instead of context-sensitive suitability to user needs (Cooper, 1999). This difference in goals can result in stark disagreement between developers while discussing implementation priorities. Acknowledging such the difference can help alleviate misunderstandings.

Spillers (2008) has presented a dichotomy of technology and user-centered design driven approaches to software design (Table 1). Though as a dichotomy it necessarily simplifies reality by dividing it in two approaches, it may shed light on situations where usability practitioners' and developers views are in conflict. In the design of the underlying architecture of the software, questions about the division of the system to components, and about how bug free the system is, are important, but whether such questions should drive the overall software vision is another consideration entirely. The starting point of the Quiz UI project does demonstrate this question, as well: the design of the original Quiz editing UI seemed to be directly derived from the data structures. The data structures were probably guided by actual user goals, but the user experience seemed still to be restricted by the focus on the system.

Technology driven	**User-centered design driven**
Component focus	Solutions focus
System driven (use cases)	(real-world) Scenario driven
Product defect view of quality	Task success view of quality
Focus on system robustness	Focus on User Interface robustness

Table 1: Dichotomy of approaches of technology versus user-centered design according to Spillers (2008)

A similarly helpful dichotomy is that of Product vs. Framework (adapted from Reichelt (2009)). The so-called product thinkers may be inclined to provide highly integrated products to users. Product thinking – which may often be the inclination for usability practitioners – proposes that if one does not aim to make finished, holistic experiences, one is providing bad service to users. Software engineers are perhaps more inclined towards framework thinking. It asserts that flexibility and re-use of software is a key priority, because this results in users gaining freedom to combine the tools they want.

Product thinkers' criticism towards framework thinkers may be that for example, framework thinkers tend to overload users with options (making the user experience worse), while thinking that they are increasing freedom for users.

Framework thinkers may think that product thinkers assume users to be more stupid than they actually are, that integrating products too tightly results in users losing freedom and thus results in degraded UX, and that documentation is the key to help users learn usage of complex software.

Both product and framework thinkers aim to provide good service to users – their approaches to this are simply different. Product thinkers may want to decrease need for documentation by means of UI design – but complex software still often needs documentation, too. Framework thinkers may want to create complex, powerful software, and think that documentation is the natural price users must pay if they want the power of the software – but complex software does require integration and design of the UI level, too, if it hopes to gain popularity among *any* group of users. Whether product or framework thinking needs to be emphasized also depends on the kind of software being made: Professionals, who are committed to using the software, might use the software if its usage can be learned from documentation, even if it has a poor UI. Applications for general use, however, usually require more careful thinking of users' goals.

What is important here is to realize differences in thinking when discussing development and design, so that both product thinkers and framework thinkers can understand the motivations that drive the actions of the other party.

8.3. Creating work circumstances for usability practitioners

It seems important that usability practitioners take part in OSS projects. Professionals who understand the field are needed, whether doing usability work such as user research and usability testing, or cooperating with the community to give design advice and keeping usability questions on the table. Usability practitioners are needed regardless of whether a given usability methodology is applicable to OSS projects. In any case, basic usability testing is critical just for setting a baseline for the usability of *any* application. Iivari (2009) states that attracting some kinds of intermediaries (such as HCI specialists) to do empirical user data gathering and analysis in open source would help in the current situation of OSS usability practice a lot.

Section 2.3. outlined issues that usability practitioners often have in OSS projects. In this section, I discuss different factors that may positively affect getting usability practitioners into OSS communities and making sure they can do their work.

In need of project vision and management

In order to guide development, a clear vision to define features as core functionality of the OSS, and other features as being out of its scope, is crucial, to avoid losing focus of the project on its core vision. Basic activities such as usability testing can help to validate design, but can not drive it.

A clear vision is also crucial to usability practice. Introducing personas and scenarios to direct and narrow down the scope of development to an OSS project after the fact is difficult, as shown in the Quiz UI project. Ideally, user centric constructs (such as personas, user goals, or scenarios) can be embedded in the software vision so that they direct the community and help UI designers make individual design decisions that are consistently aligned with those constructs.

Eventually, project lead needs to be aware of usability as a design goal and a vision of the future user experience needs to be a an active ingredient in the project vision. Management can, to a degree, set standards and give directions about what the attitude towards usability practice is in the project. Also keeping the vision alive in the everyday work of the community is important. Unless the actual open source community buys in to the usability perspective, the management's endeavours can be perceived as authoritarian. In any case, usability practitioners need to enjoy the trust of the entire community.

In need of mutual appreciation

The true power of open source development is in the community’s joint effort. One or two usability practitioners working on one module at a time might even just slow the overall development down: If the user goals have not been clearly defined, it is all up for debate. Usability practitioners spend an unnecessary amount of their time in trying to justify their case, which would often be obvious if there were an agreement of a common goal. As discussed in Sections 8.1 and 8.2, the community needs to be engaged.

Ashley and Desmond (2010) discuss usability practitioners doing research within a corporation that is itself doing software development, in order to understand the developers and other stakeholders in projects in terms of their expectations. It seems likely that people responsible for the user experience in a project need to create new processes in the development team. These processes do need to take into account existing processes and conventions, explicitly expressed or not, in order to be able to plug in to such processes. While pioneering to introduce user experience practice in an OSS organization, usability practitioners need to eventually not only gain understanding of the users, but also of the development community itself. Hence, a suitable job title might not be only “user-centered designer” or “UX researcher”, but also “community researcher”.

Also, if there is nobody in the community with understanding for usability issues, it can be hard for a open source community to even receive usability advice. This was discovered also in the Quiz UI project (see Section 4.3.). Frishberg et al. (2002) state recruiting students and practitioners as one of the goals of the HCI community to contribute to OSS projects.

Von Krogh, Spaeth, and Lakhani (2003) have studied the processes of software developers joining open source projects. Based on their observations, they have come up with a set of propositions about what are the typical steps of becoming an active contributor of code. As the nature of usability work is different than development, joining an open source project may be difficult in itself. Usability practitioners may only be willing to contribute in activities not in accordance with the "joining script" of the OSS project (whether such conditions are stated explicitly or not), that is: "the level and type of activity a joiner goes through to become a member of the developer community" (Krogh, G. von et al., 2003). Alas, they may not be able to join the project, which they attempt to.

An example of this challenge seems to have been my discussion with Hunt in the Quiz UI project, as presented in Section 5.1., where my earlier contributions to the project were stated as a reason for not allowing me to contribute in the manner I wanted to. It seems likely that had I wanted to not implement the design I provided myself (thus not complying with the "joining script"), I would have not been able to contribute at all. As usability practice can be seen as a separate profession and it is likely that not all usability practitioners want to contribute by programming, means to get respect for usability practice in open source projects, on its own merits, should be looked into, in order to enable more usability practitioners to join projects.

The lowest common denominator between usability practitioners and everybody else in the community is that everybody, at least publicly, wants to serve the users. At the end of the day, everybody wants to make the OSS as accessible and usable as possible. Different people choose different means for doing that. Ultimately this is where usability folks need so start: illustrate their aspect to the work to those who want to hear it, probably using usability testing intensively both to demonstrate the effectiveness of and the need for usability work (Section 8.2.) and to fix bugs (Section 8.4.).

In need of new ways to work together

Whether or not a usability practitioners feel they are a part of a larger community depends on how well their work is understood and respected. Also because of this, creating circumstances for the community to engage with usability practice (Section 8.2.) is critical to getting further usability practitioners to join an OSS project.

If usability practitioners are to focus on their work full-time, they have to rely on others to implement designs. If usability practitioners work from inside a bigger organisation that contributes improvements to the open source project, the programmers to implement design work may be inside that separate organisation. Otherwise, usability practitioners need to find programmers motivated to implement their suggestions inside the open source community.

Von Hippel and von Krogh (2003) and von Krogh et al. (2003) have examined the factors that motivate developers of open source software. Contributions to open source software are not pure public goods, but have significant private elements: learning, freedom to work on what one pleases, status in the development community, enjoyment, and sense of ownership over the resulting code. As a result of this, project members may not have to be separately encouraged to work by a project-assigned person. Also, releasing innovation to the public can result in further innovation in the community, increasing the profits of the original innovator. For example, if the company of an OSS developer is selling support to the product and that developer's code is released to the community, the community may build upon the developer's work. That is, as a result of releasing that developer's code to the community, the company may get better software than if the code was only being worked on in-house. Increased client satisfaction from such development of the software may then result in increased income to the company, as well. (Hippel, E. von & Krogh, G. von, 2003)

The benefits listed by von Hippel and von Krogh (2003) are not tied to only writing code, but can supposedly also apply to doing usability work. Challenges in delivering these benefits to usability practitioners are related to gaining credibility in the community. Usability practitioners can only gain the private benefit of working on what one is interested in, if developers and the community as a whole have trust in them, and are thus eventually willing to implement their designs. Releasing usability innovation to the community can lead to accelerating UI level innovation in the community, just like released functional improvements can be built upon.

The Quiz UI project shows that an open source project can act as a platform for learning about the world of usability and UCD, on the practitioner's own terms, like with software development. Before the Quiz UI project, I had taken several usability/human computer interaction courses in my university. I doubt becoming a usability practitioner by learning while working is possible in open source communities, at least not until there are usability professionals working in those communities who are able to act in some kind of a mentor role. In contrast, hobbyist programmers without formal education, having learnt to program to a degree sufficient to participate in open source projects, are not unheard of.

Also, in the Quiz UI project, I did quite a few concessions towards the open source community's terms, without which my contribution would not have gotten implemented, let alone included in the core code repository. That is, I implemented the new design myself. As development efforts in open source projects are often driven by the developers' motivation to implement a given part of the software, a usability project with not much interesting programming challenge other than UI level programming designed by a non-developer may be unlikely to find developers.

8.4. Knowing the users: Activities to employ

Various relatively low-effort techniques to learning about users have been suggested in literature. These include using log data for analyzing user behaviour, usability testing for measuring success of design, and collaboration tools such as design blogs for making design work more participatory. I will look at each of them in turn. Actual UCD methods are discussed in depth, for instance, in Cooper (1999) and in his other books.

Web analytics

As users describing their own behaviour is unreliable (Nielsen, 2001), getting log data about what users are actually doing with an application is a tempting option to gain understanding for driving design. While using log data, it must be acknowledged that log data only gives the "what" and not the "why" of user behaviour (Spool, 2008, 2009). Nonetheless, it is possible to find out in the logs what actually happens, and then verify the probable reasons of why it happens by taking in a smaller sample of users and gathering qualitative data from them with methods like usability testing.

Like in many other OSS projects, in Moodle the quantitative data available about individual sites is limited and hardly helps with design decisions. Individual sites can opt in to be in the list of registered Moodle sites[65], sending application version data and statistics about the number of courses and students on the site, and even the number of Quiz questions[66], etc. to Moodle.org. This allows Moodle.org administrators to present statistical data about different sites using Moodle and the popularity of Moodle overall.

Getting statistical data or logs about actual usage of an installation is tricky, as it can raise serious privacy issues. Allowing each user to opt in and to view the data that is being sent before accepting it, as well as making sure the group of people with access to the data is trustworthy, may alleviate these issues.

Usability testing and measuring success

Besides being effective for demonstrating usability issues to developers and other stakeholders, usability tests can be used for measuring the level of usability of software, and thus also for determining the effectiveness of usability practitioners' work. When the usability of a UI has been measured, it is also possible to set usability goals on how well we would like users (of different experience levels) to be able to perform.

User goals and tasks are considered important artefacts for UI design in UCD. However, they may seem abstract to OSS community members. Usability test tasks reflect user goals and tasks, but they are more connected than user goal documents to "real usability action" – that is, to usability testing. So making the discussion public about which usability test tasks are be the most important ones for a given UI, can

[65] http://docs.moodle.org/en/Moodle_registration
[66] http://moodle.org/stats/

perhaps also be used for taking OSS community discussion from the usual feature centeredness towards user goals. After all, a precondition for successful usability testing is that the test tasks match what users would really do if they weren't in a usability testing session. That is, the usability test tasks need to match actual real-life goals of users. Making test tasks public and asking community members to add their own and to vote on which tasks are the most critical ones would also make data about user goals visible, to be used in design. Usability test tasks could then also be used similarly to unit tests in test-driven development: When a change in the UI is made, usability tests with standardized test tasks can be used to verify if, and how well, the UI still serves its original purpose.

To measure usability with usability testing, it is necessary to determine aspects of usability to measure, agreed upon widely in industry standards (Sauro & Kindlund, 2005). Questions to ask include: How long does it take for the user to successfully carry out the task? How many of the test participants can do it successfully? How many wrong routes do users try before finding the right one? How satisfied do users report to be after having used the UI?

As the OSS community often does a large part of the development in an OSS project, it is tempting to get the community engaged also in doing usability testing. However, validity of the results of such testing can be questionable while community members are only learning to facilitate usability testing. Asking community members doing testing to provide video of the testing sessions may alleviate this issue, but may be problematic if participants do not want to appear on video publicly on the internet. Arranging introductory online courses on the topic of usability, similarly to *Introduction to Moodle Programming*[67] (that is provided for Moodle community members so they can acquaint themselves with programming functionality for Moodle) may be useful not only for engaging the community. They can also help taking the very first testing tries of community members to an area where it is clear that the testing results may not be reliable.

When compared with other UCD methods, the downside with usability testing is that, besides validity concerns of the results, it does not give a very deep understanding of users. A usability test can usually only measure the success of given predefined aspects of the user experience (UX). If the test contains unrealistic tasks, you may only learn how well users perform tasks they would not otherwise perform (Cordes, 2001). In this case, at best, you may get hints of a task being unrealistic, knowledge which you can explore using other methods such as contextual inquiries.

Getting feedback from users and responding to it

Regardless of how lively a user community of an OSS project may be, especially OSS purposed for general use will typically always have much more users who will never

[67] http://dev.moodle.org/course/index.php

interact with the actual community. In the context of OSS where user participation is already a part of the development effort, it is a challenge to bridge the gap to those users who do not care about the OSS community, but may still have valuable feedback to give the developers of the software about the issues they experience or the new ways of using the software they can envision. Getting the feedback into a form that can be used while designing software is a challenge, but a meaningful one, when aiming to further learn about the users. The aim here is also to engage users, to make them experience they are actually a part of the process, with however small a contribution they can make.

Nichols, McKay, and Twidale (2003) discuss post-deployment usability, which well suits the iterative nature of OSS. According to them, in addition to capturing incident data that would otherwise be lost, allowing users to report incidents as they happen may also enhance the user experience by giving users a forum where to express their frustration. Their prototype featured an “I'd like to complain” button, which would capture the current program state to be sent along with the issue report, and additionally allowed the user to state how they perceived the problem.

Online services such as Get Satisfaction[68] have raised the bar for customer feedback channels, making them interactive forums where users and enterprises behind products – software and otherwise – engage in an open dialogue about the product and its features. Instead of hiding the contact information of support personnel somewhere deep into a corporate website to avoid support expenses, product websites have prominent ”Feedback” badges that take ordinary users to a web application that helps users classify the nature and severity of their issues or the perceived importance of their ideas, as well as helping them determine if the issue has already been reported. Such services are intended to have a considerably lower barrier to entry of data than a typical OSS project issue tracker. Of course, lowering the barrier for feedback can increase the feedback and thus the workload for project personnel, so the possibility of doing this depends on the project's resources.

Also, it may be feasible to allow users in an open source community to provide actual usability feedback. Zhao and Deek (2006) discuss exploratory inspection, based on social constructivist learning, while critiqueing user interfaces. They created a prototype for a pilot study and suggest that users not trained in usability work can spot even in-depth and diversified usability issues if they have a software environment that provides them with a HCI pattern reference while doing the task. They discuss a two-phase “fading-out/phasing in” method, where only the core of an HCI pattern is presented to inspectors at first, and then during actual inspection the inspector of the UI can get further information of the pattern. Zhao and Deek (2006) report encouraging preliminary results.

[68] http://getsatisfaction.com/

Research users both in community forums and in the field

In an open source context, making users active participants and getting feedback from them is an effective way to learn about them. Though Iivari (2009) states that the users in the community forums are probably not representative of the overall user population, she continues that attempts can be made to contact further non-developer users too through the same communication channels. One can attempt to invite them more permanently into iteratively defining how users are seen. She discusses forum lurkers as a potential resource – people who no longer need to be invited to the project, but whom it might be possible to invite to participate more actively.

Nonetheless, to approach user-centeredness in open source UI design, users and their goals need to be researched. It is commonly stated that users do not know what they need and can not predict how they behave (Nielsen, 2001) and what they say cannot be relied on as the only source of data. Observing users work on tasks related to the application being designed is the only credible way to start seeing things from the users' point of view.

Also, users are often ashamed of the issues they experience with a program, thinking they are stupid. (This is the very reason that before a usability test you need to tell users that we are testing the software, not the test participant, so they can relax; see Appendix 5). Those who already know the application well enough to know that the fault is not theirs alone seem the most likely users to give feedback. Further research on this is required, but the issues of the novice may likely be under-represented in the community forums of OSS. Before the Quiz UI project, novice users' struggle with the Quiz editing UI had definitely not been addressed. The question is unexplored though, how much evidence of these struggles there were in the forums over the years before the Quiz UI project.

The amount of information in a forum of an open source community is often overwhelming and information relevant to design typically scarce – or at least so scattered and unstructured that holistic design decisions can not be supported. One potential approach might be having a usability practitioner following the forums, and any time something new is envisioned or something is changed as a result of the requirements for a component developing, he/she would have a systematic approach to documenting what happened and how it related to user experience. Adding another level of documentation for what has already been said seems counterproductive though, so the practice developed for this would need to be very lightweight.

Collaboration tools for learning about users

A usability practitioner may find it overwhelming to take in all the design vision inherent an OSS community, due to the informal and distributed nature in which it is being communicated. On the other hand, the information about users gained in OSS community forums may not provide deep enough insight about users and their goals to

inform design. Ways to integrate informal understanding about users of an OSS community with qualitative user research need to be explored. This way, we can perhaps better incorporate informed design in open source development processes.

The solution seems to involve engaging people (developers and other community members) in activities, in which they can begin to better understand other people (users): their goals, their cognitive capabilities and restrictions, their external conditions in a given context, and so on. Means to creating social reality where user goals are acknowledged as a focal ingredient in development, include creating guidance to developers, such as Human Interface Guidelines (HIG) and guidelines on what processes to follow in a given OSS community. Engaging the community, as described in Section 8.2., may also help in creating constraints for development work by means of establishing social norms. These may help developers see the degree to which they should concentrate on usability practice in the first place, in addition to other non-functional properties of software such as security and performance.

The goal here is to meta-design: to create the technical and social conditions for broad participation to the design, supporting users as designers, rather than making them passive consumers (Fischer, 2003; Hagen & Robertson, 2009). Hagen and Robertson (2009) propose that boundaries of design are being restructured. The roles and responsibilities of designers vis-a-vis users get closer to each other. User experience in information technology becomes more participatory in terms of not only socially engaging in using the technology, but also in defining it.

Çetin & Göktürk (2007) mention the lack of tools for online collaboration of usability practitioners as a major obstacle.

Using blogs for specifying design collaboratively is discussed by Nichols and Twidale (2006). The GIMP open source bitmap graphics editor project has a UI brainstorming blog since 2007. This is one way to encourage community member participation in forming a vision for user experience, and to give usability practice more visibility in the community. Anyone in the community is free to post images describing a particular aspect of the application UI, and in order to encourage people to contribute in a true brainstorm style, commenting is disabled on the posts in the blog to avoid criticism. Though functionality and user experience can never be really separate, such a communications channel can help shift attention from the usual feature centric nature of development discussions to thinking about the overall UX in a more holistic manner.

The challenge of involving an OSS community in usability efforts is inherently a social one. It seems straightforward to assume that also the solution would involve social efforts. However, many of the constraints that guide community members' behaviour are formed by not only the interactions with other community members, but also by the tools the community uses, be it websites, chat rooms, discussion forums, issue trackers or revision control systems. Current typical OSS community tools can be

divided into two classes: generic communications tools that facilitate discussion, and functionality centric tools. The latter have been designed for a focus on requirements for new functionality, or on deficiencies in existing functionality. Open source projects have successfully made programming a popular excercise also by having tools to support manipulation of source code, like integrated development environments or IDEs, bug trackers, and revision control systems.

Notably missing are tools that would explicitly direct focus from a functionality or requirements point of view to the software to the goals of users. Nichols and Twidale (2005) describe a process of solving a usability bug in an open source project. They call for a tool that would better support the process of describing and analyzing subjectively experienced “usability bugs”, which can be a much more complicated task than in the case of functional bugs. Such a tool, according to them, would need to support “explicit representations of design arguments, trade-offs and rationales” to clarify “the multi-aspect nature of these more complex discussions”. Though to a degree, generic communications tools, such as the forums of an OSS community, can and do facilitate user-developer mediation, the communication is still largely unstructured. There is no guarantee that a developer will be able to focus on matching the user experience with general heuristics, yet alone with user goals beyond what is obvious from the feedback they get.

The world of usability practice, namely human cognition and the basic laws of usability change relatively slowly (Nielsen, 2007). When code is changed, the design considerations that were taken into account when first designing the system risk getting lost – unless they have been made explicit somewhere. The actual artifacts of development and design need to be made visible in the community in order to be taken into account.

In the Moodle community the development is considered a joint effort about learning how to create a good learning platform (Hunt, 2008a). However, what has so far been learned is not very accessibly available to new members of the community or to outsiders. Those goals, which a good learning platform is to serve, need to be explicitly made visible and linked with the community current understanding of how a given user goal is best served in a UI.

The problem with documentation that is meant to be permanent is that it ages quickly. The “code is cheap” attitude of many open source communities is a response to the documentation-intensive conventions of traditional software engineering (SE) (Massey, 2003). Too much documentation is seen as an extra burden on developers, and it is thought that in practice, developers do not see a need to use the documentation often enough to justify the effort.

Still, it seems that a place to document understanding about users could be useful to avoid redoing the same user research and requirements engineering over and over again. User needs addressed by a UI can not be derived from the code or existing UI

alone. In order to remain useful though, documented needs have to stay up-to-date. Currently, many of the discussions that are carried out in the discussion boards of an OSS project likely touch questions of the correspondence of a given functionality to the goals of different users. Making those expected goals of users explicitly represented in a dedicated system as entities actively used in the development of the application, might help direct the discussion into that system and keep the documentation up-to-date.

When user goals and circumstances are promoted to visible entities that are appreciated in the project, it seems that this might facilitate also user participation in actual design, in form of comments such as “it seems that user interface X does not really correspond well to user goal Y, but users have to use a workaround to achieve this, so I drew this mockup proposal that I think better supports achieving this goal”.

Personas and user goals can be too abstract and difficult to make as artifacts of an OSS development community. As discussed in Section 8.2. and earlier in Section 8.4., usability testing can be used both for convincing the user community about the need for usability work, and for measuring the level of usability of the software. Usability tests and usability test tasks are also related to the existing UI in a concrete manner, and also connected with more abstract user goals. Usability test tasks as units of information that can be manipulated socially in an OSS community may prove to be a great artifact for eliciting discussion about users.

Test tasks are supposed to reflect what users (are expected to) do with the application. Perhaps an OSS community could thus be engaged to **generate usability test tasks** and to even vote which test tasks are the most important ones for a given screen, module or functionality. This could essentially serve as a simplified mechanism for understanding what are the core user goals the application is meant to serve, and perhaps even for categorizing different goals into personas.

8.5. OSS 2.0: a new focus on users?

Fitzgerald (2006) presents a thinking framework, in which different approaches to also usability and user experience can be considered. According to him, a transformation has taken place, in which open source has adopted a more mainstream and commercially viable form. Open source conquers new fronts on vertical domains, from invisible infrastructure of the back-office, to highly visible deployment in the front-office.

Fitzgerald describes a shift in the open source software phenomenon using a framework of process and product factors. He presents the concept of OSS 2.0, a new form of open source that is characterized by an approach to software development more alike commercial software, with a requirement for more rigorous approach to project management, rendering the development process *less* bazaar-like. Alas, software developers are no longer domain experts by virtue of being themselves, and they lack

the experience in the domain to derive the requirements from the target users. A change is visible also from the process thinking, where developer-users take part in the continual creation of an always-developing application, to product thinking, where customers are using at least in principle a finalized product, with marketing, sales and support services. To replace “free” as the key word of free and open source software, “value” will be the key word for OSS 2.0; both in the sense of “value for money” and of “acceptable community values”. (Fitzgerald, 2006)

An OSS 2.0 scenario of projects moving to vertical domains may provide opportunities to usability practitioners approaching open source projects. In OSS 2.0, providing a user (customer) experience becomes a priority – one that is not only usable and consistent, but also a good fit to particular user goals and appealing overall. Scratching a personal itch is no longer enough. A degree of domain knowledge is a prerequisite to getting the requirements right and thus to the survival of any software project. In a vertical application, understanding about the core user needs – requirements – has to exist within the development community at least on a general level.

8.6. Research OSS communities to foster a culture for usability learning

Usability practitioners must have the trust of the community to be able to cooperate with them. Frishberg et al. (2002) state that ”continuing to uncover secrets of building trust and collaborative relationships in non-hierarchical systems” is one of the goals of the HCI community contributing to OSS.

In addition to building trust, Frishberg et al. (2002) state research about successful OSS practice in general as a key goal for open source usability:

> ‘Develop testable hypotheses of what makes a successful open source project, where "success" is determined both by the development of a released product and by the acceptance of that product beyond the developer community.’ (Frishberg et al., 2002)

In open source development, everything is done in the community – what is not done in the community has no credibility. Since UCD usually cannot be introduced to existing OSS projects as a separate process, it must **integrate to the existing processes** of the open source project, in which it wishes to have an influence. Thus, the relationship between how understanding about users is developed in the informal model of open source, and how that compares with the UCD approach, needs to be looked into. This may help usability practitioners make use of the existing processes in open source communities for learning about users. The visibility and the degree to which the community can participate in the work – or the degree to which it makes any sense to not involve them – is a key question. To what degree should the work happen “in” the community, placing all there is to know about the design decisions made online, or at least documented in forums, in the wiki, and in the bug tracker?

In the OSS project Iivari (2009) studied, she perceives that often, “the developers have to rely on very few user comments when making decisions”. Nonetheless, those comments are open for the world to see and research on them may also help in gaining insight in closed source projects (Nichols & Twidale, 2005). In Iivari's research, she analyses user participation using an approach where OSS writers are assumed to produce subject positions for readers (users) to occupy, 'configuring' them and defining them, “their competencies, motives, tastes and aspirations”. These subject positions can be adopted, or also negotiated or opposed (Iivari, 2009). She classifies the users of the OSS as newbies, guests, members and developers, and message content in OSS community forums as “feature request”, “problem”, “helping” and “thanking and praising”. Having analyzed all the messages included in the research, she then goes on to describe the different kinds of interactions users and developers have in the community: for example, sometimes users represent themselves and characterise the type of user they are, and sometimes they are in the forums representing other users. In some cases, users provide empirical evidence of users' behaviour, and in others, they seem to “offer only opinions or stereotypes”. She also describes a scenario in the forums where major features changes were discussed or done, and how users' reactions affected the developers' decisions. (Iivari, 2009)

Von Krogh et al. (2003) discuss contributor motivation, project organization and dynamics of key events such as joining or leaving a project. I believe further research, such as that of Iivari and von Krogh, about the social dynamics of OSS communities is required in order to find the ways in which usability practitioners could most fruitfully interact with an OSS community.

9. Conclusion

I presented the Quiz UI usability project to illustrate the open source usability work in the Moodle OSS project.

In the Quiz UI project, I used UCD methods in a lightweight manner with varying success. It was crucial in the Quiz UI project to get the the community to accept my work, and eventually they did, well in time before my project was finished. I implemented the design and it was added to Moodle 2.0 in November 2008.

I iteratively designed the Quiz editing UI, guided by teacher interviews, usability testing and feedback from the Moodle community. Having to both do usability work and implement the user interface resulted in me working during a period of nine months, three of which were funded. During the project I learned about the community centric fashion in which any development work is carried out in the Moodle community, and about the tremendous value community feedback can provide even to a usability project. Approaching the community that seemingly had little or no previous understanding of user research or user-centered design was difficult, and before the project it took me over a year to get the relevant members of the community convinced of the need for work and my ability to follow it through.

As my main results, I listed several factors that may help both open source communities develop their usability practices and usability practitioners to work in open source projects. I recommend different practical approaches (such as usability testing, UI design blogs and regular expert reviews) for getting started with usability practice in an OSS community. I suggest actively engaging the open source community, by helping them participate in usability testing efforts, for example. It is still necessary for usability practitioners to support community efforts though, by offering user research, usability testing and other work as a service to active development efforts. Online collaboration tools for discussing and documenting user goals (complementing feature and bug centric tools, like current issue tracking software) may be of great benefit in making usability work less an endeavour for academic usability practitioners, and more a general interest for an entire OSS community to engage with. Involving the community is necessary since the development done by the community is often a considerable part of the development of a given OSS. There are usually too few usability practitioners to provide user research services to all of them. This can be alleviated also by making OSS projects more accessible for usability practitioners to join. If usability practitioners actively participate in the community, this also helps them understand the software engineering perspective of development better, also enhancing communication.

The basic idea of UCD is that understanding about users should permeate the process of designing the user experience in a holistic manner. User experience can be perceived to relate to the ease of programming and extending the software, in addition

to the UI. A key to Moodle's success is that a high-level pedagogical vision does indeed determine a lot of the direction of the overall development. The functional requirements for Moodle do seem to gradually approach actual user goals (or at least the question of that is a topic for another thesis). Fulfilling the non-functional requirements of usability is another story: Understanding the user goals to the degree, where they can be used in the design of the UI on all levels of abstraction, requires user research. Making informed design decisions is impossible solely based on information that the community can provide as feedback. This is true both on a higher level of UI patterns and heuristics, and on the level of concrete UI screen design – creating visual hierarchy to guide users' attention to what is most important for reaching their goals.

At the moment, there is no readily applied process or methodology to follow for usability engineering in OSS communities. The one thing in common with UCD and OSS development is that work happens iteratively. Feedback and iterative development is what OSS relies on for much of its quality assurance, also in terms of usability. However, in UCD, the iterativeness of the design process is focused explicitly on the quality of the user experience. To not lock down design decisions too early, implementation needs to be avoided until user goals are confirmed to be met by the design. OSS thinking emphasizes that "code is cheap" and that iteration can happen even when implementation has already occurred. Still, it is clear that lengthy programming efforts such as the Quiz UI project would never be required, and developer effort not unnecessarily wasted, were user goals taken into account already at pre-implementation design phase.

Also, many UI issues can be avoided if user research is already present before implementation. An important example is that even Moodle 2.0 will not facilitate fully editing a quiz after a student has taken it[69] even though iteratively developing quizzes is central to teachers using quizzes. Implementing the feature requires a major overhaul of the underlying data structures and functionality, which may apparently happen in time for Moodle 2.1.

Based on the Quiz project, no generalizations can be made on how well UCD methods fit open source projects in general. Some methods such as usability testing and also remote testing are obviously powerful in an OSS context. Having to rethink UCD practice may be good for those usability practitioners willing to work in OSS communities. This will hopefully lead to creating practices that fit open source better. In time, practices will emerge, but it seems likely that whatever will be successful will be as informal and community centric as the current development workflows in OSS communities.

[69] http://docs.moodle.org/en/Editing_a_quiz#After_quiz_has_been_attempted

References

Adams, R. J. (2003, October 31). The open source and usability problem. *roBlog dot org*. Retrieved July 1, 2009, from http://loki.lokislabs.org/weblog/archives/2003/10/31/the_open_source_and_usability_problem.php

Allen, C. D., Ballman, D., Begg, V., Miller-Jacobs, H. H., Muller, M., Nielsen, J., & Spool, J. (1993). User involvement in the design process: why, when & how? In *Proceedings of the INTERACT'93 and CHI'93 Conference on Human Factors in Computing Systems* (p. 254). ACM Press.

Ashley, J., & Desmond, K. (2010). Enterprise Usability Maturity: Designing an enterprise application suite. *UPA User Experience Magazine*, *9*(1). Retrieved from http://www.upassoc.org/upa_publications/user_experience/past_issues/2010-1.html#ashley

Barcellini, F., Détienne, F., & Burkhardt, J. (2008). User and developer mediation in an Open Source Software community: Boundary spanning through cross participation in online discussions. *International Journal of Human-Computer Studies*, *66*(7), 558-570.

Bastien, J. M., & Scapin, D. L. (1993). *Ergonomic criteria for the evaluation of human-computer interfaces* (Research report No. 156). INRIA.

Benson, C. (2004). Meeting the challenge of open source usability. *Interfaces*, *2004*(60), 9–12.

Çetin, G., & Göktürk, M. (2007). Usability in open source: Community. *interactions*, *14*(6), 38-40. doi:10.1145/1300655.1300679

Cole, J., & Foster, H. (2007). *Using Moodle - Teaching with the Popular Open Source Course Management System* (2nd ed.). O'Reilly.

Cooper, A. (1999). *The Inmates Are Running the Asylum*. Macmillan Publishing Co., Inc.

Cordes, R. E. (2001). Task-selection bias: a case for user-defined tasks. *International Journal of Human-Computer Interaction*, *13*(4), 411-419.

Cunningham, S. J., & Jones, M. (2005). Autoethnography: a tool for practice and education. In *Proceedings of the 6th ACM SIGCHI New Zealand chapter's international conference on Computer-human interaction* (pp. 1-8). Auckland, New Zealand: ACM Press. doi:10.1145/1073943.1073944

Dougiamas, M. (2010, April 20). Moodle: a case study in sustainability. *OSS Watch*. Retrieved May 28, 2010, from http://www.oss-watch.ac.uk/resources/cs-moodle.xml

Dougiamas, M., & Taylor, P. (2003). Moodle: Using learning communities to create an open source course management system. In D. Lassner & C. McNaught (Eds.),

World Conference on Educational Multimedia, Hypermedia and Telecommunications 2003 (pp. 171-178). Honolulu, Hawaii, USA: AACE.

Ehn, P. (1993). Scandinavian Design: On participation and skill. In D. Schuler & A. Namioka (Eds.), *Participatory Design*. Routledge.

Erickson, T. (1996). Design as storytelling. *interactions*, *3*(4), 30-35. doi:10.1145/234813.234817

Fischer, G. (2003). Meta—Design: Beyond User-Centered and Participatory Design. In *Proceedings of HCI International* (pp. 88-92). Crete, Greece.

Fitzgerald, B. (2006). The transformation of open source software. *MIS Quarterly*, *30*(3), 587-598.

Frishberg, N., Dirks, A. M., Benson, C., Nickell, S., & Smith, S. (2002). Getting to know you: open source development meets usability. In *Conference on Human Factors in Computing Systems CHI '02* (pp. 932–933). ACM Press.

Hagen, P., & Robertson, T. (2009). Dissolving boundaries. In *Proceedings of the 21st Annual Conference of the Australian Computer-Human Interaction Special Interest Group - OZCHI '09* (p. 129). ACM Press.

Heath, C., & Heath, D. (2010). *Switch: How to Change Things When Change Is Hard* (1st ed.). Broadway Business.

Hedberg, H., Iivari, N., Rajanen, M., & Harjumaa, L. (2007). Assuring quality and usability in open source software development. In *Proceedings of the 29th International Conference on Software Engineering Workshops* (pp. 122-126). IEEE Computer Society. Retrieved from http://portal.acm.org/citation.cfm?id=1261326#

Hinkelman, D. (2007, June 3). Re: Quiz UI development. Retrieved June 9, 2009, from http://moodle.org/mod/forum/discuss.php?d=72828#p326140

Hippel, E. von, & Krogh, G. von. (2003). Open source software and the "private-collective" innovation model: Issues for organization science. *Organization Science*, *14*(2), 209–223.

Hunt, T. (2008a, March 12). Using Moodle: Re: Concern of Moodle Usability - Quiz module. *Moodle Quiz module forum*. Retrieved May 4, 2010, from http://moodle.org/mod/forum/discuss.php?d=92870#p408294

Hunt, T. (2008b, March 18). Re: Simple Quiz building user interface. Retrieved June 9, 2009, from http://moodle.org/mod/forum/discuss.php?d=92797#p410008

Hunt, T. (2009a, April 28). Usability workflow in open source. Retrieved October 5, 2009, from http://moodle.org/mod/forum/discuss.php?d=121612#p535631

Hunt, T. (2009b, June 22). The geeks don't really matter. *Tim's blog*. Retrieved April 15, 2010, from http://tjhunt.blogspot.com/2009/06/geeks-dont-really-matter.html

Iivari, N. (2008). Empowering the users? A critical textual analysis of the role of users in open source software development. *AI Soc.*, *23*(4), 511-528.

Iivari, N. (2009). User Participation in 'Configuring the User' in OSS Development. International Conference on Information Systems (ICIS) 2009 Proceedings, Paper 199. Retrieved March 19, 2009, from http://aisel.aisnet.org/icis2009/199

Isner, J. (2008, March 13). Re: Concern of Moodle Usability - Quiz module. Retrieved May 24, 2010, from http://moodle.org/mod/forum/discuss.php?d=92870#p408592

ISO (1999). 13407: Human-centred design processes for interactive systems. *Geneva: International Organization for Standardization*.

Kamp, P. (1999, October 2). A bike shed (any colour will do) on greener grass... Retrieved October 5, 2009, from http://www.freebsd.org/cgi/getmsg.cgi?fetch=506636+517178+/usr/local/www/db/text/1999/freebsd-hackers/19991003.freebsd-hackers

Krogh, G. von, Spaeth, S., & Lakhani, K. R. (2003). Community, joining, and specialization in open source software innovation: a case study. *Research Policy*, *32*(7), 1217–1241.

Krug, S. (2000). *Don't Make Me Think!: A Common Sense Approach to Web Usability*. Que Corp.

Marshall, S. (2009, June 30). Re: "Why open source software usability tends to suck" - how do we work around this? Retrieved October 5, 2009, from http://moodle.org/mod/forum/discuss.php?d=126856#p555967

Marty, P. F., & Twidale, M. B. (2005). Usability@ 90mph: Presenting and evaluating a new, high-speed method for demonstrating user testing in front of an audience. *First Monday*, *10*(7). Retrieved from http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/viewArticle/1260/1180

Massey, B. (2003). Why OSS folks think SE folks are clue-impaired. *Proc. Workshop on Open-Source Software Engineering*, 2003 International Conference on Software Engineering, 91–97. Retrieved July 1, 2009, from http://www.cs.pdx.edu/~bart/papers/icse-osse.pdf

Mockus, A., Fielding, R. T., & Herbsleb, J. D. (2002). Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, *11*(3), 309-346.

Nichols, D. M., McKay, D., & Twidale, M. B. (2003). Participatory Usability: supporting proactive users. In *Proceedings of the 4th Annual Conference of the ACM Special Interest Group on Computer Human Interaction New Zealand Chapter CHINZ'03* (pp. 63-68). ACM Press.

Nichols, D. M., Thomson, K., & Yeates, S. A. (2001). Usability and open-source software development. In E. Kemp,, C. Phillips, Kinshuck, & J. Haynes (Eds.), *Proceedings of the symposium on computer human interaction. SIGCHI New Zealand* (pp. 49-54). ACM Press. doi:10.1.1.2.1116

Nichols, D. M., & Twidale, M. B. (2003). The Usability of Open Source Software. *First Monday*, *8*(1). Retrieved from http://firstmonday.org/article/view/1018/939

Nichols, D. M., & Twidale, M. B. (2005). Exploring usability discussions in open source development. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences HICSS'05* (p. 198c).

Nichols, D. M., & Twidale, M. B. (2006). Usability processes in open source projects. *Software Process Improvement and Practice*, *11*(2), 149-162.

Nickell, S. (2002). Why GNOME hackers should care about usability. *GNOME Usability Project*. Retrieved July 1, 2009, from http://developer.gnome.org/projects/gup/articles/why_care/

Nielsen, J. (1994). Guerrilla HCI: using discount usability engineering to penetrate the intimidation barrier. In D. J. Mayhew & R. G. Bias (Eds.), *Cost-justifying Usability* (pp. 245-272). Academic Press, Inc.

Nielsen, J. (1999, November 14). When bad design elements become the standard. *Jakob Nielsen's Alertbox*. Retrieved March 31, 2009, from http://www.useit.com/alertbox/991114.html

Nielsen, J. (2001, August 5). First rule of usability? Don't listen to users. *Jakob Nielsen's Alertbox*. Retrieved May 4, 2010, from http://www.useit.com/alertbox/20010805.html

Nielsen, J. (2005). 10 heuristics for user interface design. Retrieved March 18, 2010, from http://www.useit.com/papers/heuristic/heuristic_list.html

Nielsen, J. (2006, December 4). Progressive disclosure. *Jakob Nielsen's Alertbox*. Retrieved November 12, 2009, from http://www.useit.com/alertbox/progressive-disclosure.html

Nielsen, J. (2007, June 11). Change vs. stability in web usability guidelines. *Jakob Nielsen's Alertbox*. Retrieved May 4, 2010, from http://www.useit.com/alertbox/guidelines-change.html

Nielsen, J., & Molich, R. (1990). Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems CHI '90* (pp. 249-256). ACM Press. doi:10.1145/97243.97281

Nielsen, L., & Bødker, M. (2007). To do or not to do : Usability in open source development. *Interfaces, 71*, 10-11.

Pemberton, S. (2004). Scratching someone else's itch: (why open source can't do usability). *interactions*, *11*(1), 72. doi:10.1145/962342.962366

Raymond, E. S. (1999). The Revenge of the Hackers. In *Open Sources: Voices from the Open Source Revolution*. O'Reilly Media. Retrieved from http://oreilly.com/catalog/opensources/book/raymond2.html

Raymond, E. S. (2001). *The Cathedral and the Bazaar*. O'Reilly Media, Inc.

Raymond, E. S. (2003). *The Art of Unix Programming*. Pearson Education.

Reichelt, L. (2009, October 12). Designing for the wrong target audience (or why Drupal should be a developer tool and not a consumer product). *disambiguity*. Retrieved May 27, 2010, from http://www.disambiguity.com/designing-for-the-wrong-target-audience/

Ruffini, L. (2008, November). Interview with Martin Dougiamas - Open University of Catalonia. Retrieved April 3, 2009, from http://www.uoc.edu/portal/english/la_universitat/sala_de_premsa/entrevistes/2008/martin_dougiamas.html

Sauro, J., & Kindlund, E. (2005). A method to standardize usability metrics into a single score. In *Proceedings of the SIGCHI conference on Human factors in computing systems CHI '05* (pp. 401-409). ACM Press. doi:10.1145/1054972.1055028

Scacchi, W. (2002). Understanding the requirements for developing open source software systems. *IEE Proceedings-Software*, *149*(1), 24–39.

Shneiderman, B. (1997). Direct manipulation for comprehensible, predictable and controllable user interfaces. In *Proceedings of the 2nd international conference on Intelligent user interfaces IUI '97* (pp. 33-39). ACM Press. doi:10.1145/238218.238281

Spillers, F. (2008). *User Centered Design 101*. Presented at the Web Seminar: Experience Dynamics Inc., Portland, Oregon. Retrieved from http://www.experiencedynamics.com/science-usability

Spool, J. (2008, April 12). *Journey to the center of design (Slideshow with audio)*. Retrieved from http://www.slideshare.net/jmspool/journey-to-the-center-of-design

Spool, J. (2009, March 19). *Journey to the center of design*. Presented at the SXSWi 2009. Retrieved from http://www.youtube.com/watch?v=WCLGnMdBeW8

Sy, D. (2007). Adapting usability investigations for agile user-centered design. *Journal of Usability Studies*, *2*(3), 112–132.

Terry, M., Kay, M., & Lafreniere, B. (2010). Perceptions and practices of usability in the free/open source software (FoSS) community. In *Proceedings of the 28th International Conference on Human Factors in Computing Systems CHI '10* (pp. 999-1008). ACM Press. doi:10.1145/1753326.1753476

The Learning Technology Centre of the University of Tampere. (2009). What is the electronic exam service? - Electronic Exam Service - University of Tampere. Retrieved May 21, 2009, from http://www.uta.fi/studies/webtools/electronic_exam/presentation.html

Thomas, M. P. (2008, August 1). Why Free Software has poor usability, and how to improve it. Retrieved June 30, 2009, from http://mpt.net.nz/archive/2008/08/01/free-software-usability

Trudelle, P. (2002). Shall we dance? Ten lessons learned from Netscape's flirtation with open source UI development. In CHI 2002 Getting to Know You: Open Source Meets Usability Workshop. Retrieved April 10, 2009, from http://www.iol.ie/~calum/chi2002/peter_trudelle.txt

Viorres, N., Xenofon, P., Stavrakis, M., Vlachogiannis, E., Koutsabasis, P., & Darzentas, J. (2007). Major HCI challenges for open source software adoption and development. In D. Schuler (Ed.), *Lecture Notes in Computer Science LNCS* (Vol. 4564, pp. 455-464). Presented at the Online Communities and Social Computing. 2nd International Conference, OCSC 2007, Springer.

Warsta, J., & Abrahamsson, P. (2003). Is open source software development essentially an agile method? In *Proceedings of the 3rd Workshop on Open Source Software Engineering* (pp. 143–147). Presented at the 25th International Conference on Software Engineering.

Zhao, L., & Deek, F. P. (2006). Exploratory inspection: a learning model for improving open source software usability. In *CHI '06 extended abstracts on Human factors in computing systems* (pp. 1589-1594). ACM Press. doi:10.1145/1125451.1125741

Zsolt, T., & István, B. (2008). *Moodle and social constructivism*. Network for Teaching Information Society. Retrieved from http://www.ittk.hu/netis/doc/textbook/Toth_Bessenyei_moodle_eng.pdf

Appendix 1: The materials for the scenarios interviews

All of the below has been translated from Finnish. Originally published in the Moodle Docs documentation wiki[70].

Background material

- Consent for recording audio (1 A4 sheet)
- Background information form (1 A4 sheet):
  - Frequency of website/browser usage
  - Do you experience with which learning environments?
  - How often do you use a learning environment as a teacher
  - Question types you use: multiple choice; essay; other, what
  - Categorisation basis of question categories
  - Do you use electronic exams, with which software?
  - If you use websites, which kinds of services do you mostly use?

Preparatory talk

> We are developing the Moodle Quiz module. The goal is to understand how actual users/teachers do their work, especially related to exams. The setting here, so to speak, is that you are the master and I am an apprentice - that is, I will mostly just ask questions, listen and take notes. I apologize in advance that I might seem very concentrated on my notes at some point, since I am not a very accustomed interviewer.

Actual interview questions

(The idea was not to follow this list strictly in the interviews, but to use it as a rough memory help of the questions to find out about. When a teacher's thought wandered, I let it do so and sometimes asked related questions previously unthought of, where it seemed relevant.)

Background

- Please describe the character of your work.
  - What kinds of responsibilities do you have?
  - What forms of teaching do you use; group work, lectures, online teaching, ...?
- Apart from reality, if you could imagine a tool that would optimally serve your way of making exams/quizzes, what would the experience of using it be like?

[70] http://docs.moodle.org/en/Development:Quiz_UI_redesign_-_what_makes_a_scenarios_interview%3F (first published June 13th, 2008)

What tasks related to making quizzes/exams could a machine help you with or do for you?

- What is usually the context of evaluating students' learning?
  - Within that context, how do you end up with the conclusion that your course requires an exam/quiz? (→ Curriculum, course?)
- What do you think about exams as a way to evaluate students' learning?

Exam making process

- What tools or accessories do you use while making exams/quizzes, physical or software?
- Please describe how you proceed from that to making an exam/quiz. What are the different stages?
- If you have made electronic exams, what do you do, how does the process differ from making pen and paper exams/quizzes?
- Chronologically (on a time scale), at which stage of a course do you think of the exam/quiz content for the first time?
  - How long a time is there between making the quiz/exam, actually having the students do the exam, and grading it? How often do students take your quizzes/exams, what determines this?
  - When do you create the actual, exam/quiz which you give to students to do?

Exam making environment/situation, content

- What is the situation like when you create an exam? Where do you do the exam physically? Are there distractions or can you tolerate distractions?
- What do you use as material or tools when creating an exam?
- Do you have to copy some information to many places?
- Do you cooperate with other parties when making exams or otherwise about exams? What are the ways/medias in which you communicate with them?
  - What determines the structure of an exam?
  - Do you use old exams as material?
- Is the order of questions predefined and determined already before you create the actual exam, or do you reorder questions when they are already in the exam?

Exam making tools

- What is problematic about the current tools for making quizzes/exams?
- What is good about current tools for making quizzes/exams?
- When you create an exam, where there are questions that are randomly selected from a larger selection of questions, to each exam attempt, what are the stages to doing this?

If they have used Moodle Quiz: Do you use names of questions? How, for what? What could you imagine them being used for?

Appendix 2: Test tasks for the paper prototype tests

These questions were used to test the original prototype of 2007. They were later translated from the original Finnish to English to be accessible to the Moodle online community in the Moodle documentation wiki[71].

The names of elements such as "random question" in the user interface are not mentioned, since the point is to see whether the name of the element means to the user what it is intended. The user interface in the test was in Finnish, too.

1. Create questions for the exam of General science and publish the exam.

Add the questions of Aftonrenger to the exam so that for each student's exam, one of the questions will be selected
2. Add the questions of Extrapend to the exam so that each of them will be used in any given exam
3. Change the exam so that all the questions show on the same page in a student's exam
4. Make it so that one of Extrapend's questions is alone on the first page.
5. On the first page of the exam, instruct the student taking the exam that they "must answer the questions with reckless abandon".
6. In the first task, you added a random question to the exam. Make it so that there are two random questions in the exam from that set of questions.
7. There is a ready-made set of questions available in the Organize-section. Add those questions to the exam.
8. There is a question by Mr. Dell in the category Good questions. Add it to the exam.
9. Make sure the reservation information concerning the exam is correct, and publish the exam.

Material provided to test participants, for use in the test:

Aftonrenger:
Present the main points of the theory of beings and unbeings.
Compare the argumentation style of Aftonrenger to that of Wiles.
Present your view about the current situation.

Extrapend:
Define 'meaningful'.
Describe the local student culture's predominant argumentation style.
How do you see the social circumstances in Finland in the following ten years?

[71] http://docs.moodle.org/en/Development:Quiz_UI_redesign_prototype_questions#Testing_questions_used_in_spring_2007

Appendix 3: Test tasks for the click-through mockup usability tests

These tasks were used in the usability testing of Spring-Summer 2008.

1. You are teaching a course about the French revolution and you are making an electronic exam for that course. You will use an exam building tool, which you can see in front of you, to build the exam. (In case you have your own material with you, you may assume during the test that the exam is about your own subject).

First, one warm-up question. If you would normally use a keyboard, please instead speak out loud what you would do with the keyboard.

Add an essay question to the exam: "Describe Napoleon's stages on his way to autocracy."

With each task, you can give the the card back to the test organizer any time you feel you have completed the task or if you wish to give up.
2. See how the question you just made looks to a student in an exam
3. You realize that the question you just made should have the full name, Napoleon the Beardless. Correct the question.
4. Read about the Basic ideas of the exam making tool. Especially familiarize yourself with the concept of a Random question. If something still seems unclear to you, please tell about this to the test organizer.
5. To make sure students can not copy questions to each other, you want a selection of questions to the exam, from which one is randomly chosen for each time a student takes the exam. The subject for the question is The Years After the Revolution.

Add the first question (name the new question "Dickens") to such a selection: "What was the time after war like?"
6. Add two more alternative questions by the subject The Years After the Revolution

How would you describe Napoleon's strategic strengths?
Analyze the damage, caused by the war, in the society
7. Add instructions to the exam to tell the student that especially the question about The Years After the Revolution must be answered according to the writing norms of scientific texts.
8. Edit the question "How would you describe Napoleon's strategic strengths?" in the category The Years After the Revolution. The question should have the full name of Napoleon Beardless.

Appendix 4: Tasks for the usability test with the implemented application

These tasks were used in the usability testing of August 2008.

1. Create a new quiz on the course “My Course”. (This task was left out after the first test as irrelevant to Quiz itself, as it took too much time from the start of the Quiz.)
2. Create a new essay question to the quiz

Question: Describe Napleon's journey on his way to autocracy.

3. See how the question you just created looks like to a student in the quiz.
4. You notice that the question you just made has a spelling error. Fix the error.
5. To make sure students can not copy questions to each other, you want a selection of questions to the quiz, from which one is randomly chosen for each time a student takes the quiz. Create such a selection.
6. The topic for the questions: The Life of Marie Antoinette
7. Add two questions to the selection you just created.

Questions:

-Gaining power: Describe Marie Antoinettes efforts to become powerful
-Death: Which events led to the death of Marie Antoinette

8. Read about the Basic ideas of making quizzes. Especially familiarize yourself with the concept of a Random question. If something still seems unclear to you, please tell about this to the test organizer.
9. To make sure students can not copy questions to each other, you want a set of questions to the quiz, from which one is randomly chosen for each time a student takes the quiz. The subject for the question is The Years After the Revolution.
10. Add two questions to the set you just created.

How would you describe Napoleon's strategic strengths?

Analyze the damage, caused by the war, in the society

11. Add a new page to the quiz. Ensure that only question number 1 is on the first page.
12. Move the question, currently third in the quiz, to the same page where the first question is.
13. Switch the order of the questions, which are on the same page.
14. Add instructions to the exam to tell the student that especially random question number 1 must be answered according to the writing norms of scientific texts.
15. Open the tab “Order and paging”.
16. Add a new page to the end of the exam.
17. Move all the questions in the quiz to the last page in the quiz.
18. Set grades for each of the questions, which you think are appropriate.
19. Remove one question from the quiz.
20. Notify the student in the exam about the fact that cheating will have grave consequences.

Appendix 5: Checklist of things to mention in usability testing

The following list was constructed to help remember what to say to participants in the usability tests. It is partially based on Krug (2000). It was used in the usability tests during 2008, though the basic concept was same in the paper prototype tests of Spring 2007. It is translated from the original Finnish, and originally published in the Moodle documentation wiki[72].

- We are developing Moodle's quiz tool; we need to find out how real users work while in interaction with our application.
- We are testing the application, not you. We hope that if anything puzzles or annoys you during this test, you say it out loud. You do not have to be afraid of hurting anybody's feelings here.
- This is probably the only place in the world where you don't have to be afraid of making mistakes, since that is what we are indeed looking for: we want to find about where users might have problems with the application.
- In usability tests a technique called thinking out loud is used. This is very simple, but it varies between different people how easy it is for them. It means simply that while doing the tasks you are going to receive, you say out loud whatever pops into your mind, whatever you are wondering about or perhaps if something is difficult, or whatever.
- If you have any questions during the test, please do not hesitate to ask. It may be that I cannot reply during the test, but we will try to remember everything until the end of the test, when we can talk through all the questions you may have unanswered.
- I have a couple of forms for you to fill out before you can continue, please fill them out - they are just to provide some background information for us.
- Thanks - before we start, I want to remind you of thinking out loud. But don't worry – if you should stay silent for too long during the test, I will remind you.

After the test:

- Now that we still have the last screen here, please give us feedback: do you see anything on the screen that you think is extra, or the meaning of which you don't understand?
- Finally, please fill out this feedback form
- Thanks!

[72] http://docs.moodle.org/en/Development:Quiz_UI_redesign_-_prototype_testing_background