Frontpage

Todo list

The department of Computer Science
Student report
Selma Lagerlöfs Vej 300
Telephone +45 9940 9940
Fax +45 9940 9798
http://www.cs.aau.dk/

Title:
MyMoodle

Theme:
Application Development

Project period:
SW6, spring semester 2012

Project group:
sw608f12

Participants:

Alex Bondo Andersen

Kim Ahlstrøm Jakobsen

Mikael Midtgaard

Rasmus Veiergang Prentow

Supervisor:
Kurt Nømark

Page count: 115

Appendix count: 4

Finished: 4/6 – 2012

Synopsis:

This project is a part of a multi-project consisting of four groups. The goal of the multi-project is to develop the system MyMoodle, which is an extension to the learning management system Moodle. MyMoodle is an attempt to incorporate the Aalborg Problem Based Learning model into an existing learning management system by creating a virtual meeting place for students and supervisors. This report deals with the sub-system of MyMoodle that handles the administration of the virtual meeting places. Additionally, it deals with the development of the virtual meeting place in which modules developed by other sub-groups are incorporated.

Preface

This report is the documentation of a bachelor project conducted in the period from February 1, 2012 to June 4, 2012 by the software student group sw608f12. The project is the outcome of the 6^{th} semester at Aalborg University. The overall theme of the multi-project is: "Application Development". The project is a part of a multi-project consisting of four groups. The first part of the report is written in cooperation between the four groups of the multi-project, while this preface and parts two and three are written entirely by our group, who are the participants noted in the title page.

In code snippets through out the report the notation of three dots (...) can be seen. This notation illustrates that of one or more lines of code are omitted. Code is only omitted in situations where those lines are not important to the description of the code.

The first few lines of every chapter (written in *italic*) in Parts II and III is a header, which very briefly describes what the given chapter contains and why it is important to the project.

When citing literature the notation [XX] is used where XX indicates the number of the specific literature, which can be seen in the back of the report on page 91. A copy of the code can be found on the attached CD. We refer to the source code as Appendix C.

The product are hosted on a computer by AAU University. Information regaring how to connect to the server can be found in appendix C.

The final system is hosted on a server at Aalborg University. Login information can be found on the CD. This information is referred to as Appendix D.

We, the authors, would like to thank our supervisor Kurt Nørmark for guidance, continuous support, and great enthusiasm throughout the course of the project. We would also like to thank the people who participated in out interviews and demo meetings, which was a tremendous help.

II

Contents

Part I

Inter Group Work

Chapter 1

Introduction

As computers are becoming a more integrated part of our lives, they are becoming more integrated with out educational system as well. Different educational methods are supported by different information communication systems called e-learning systems. This observation has lead us to investigate how the method of our university, the Aalborg Problem Based Learning (PBL) model, can be implemented into the e-learning system being used at our university, namely Moodle.

This project is conducted as a multi-project, which means that sub-groups will be working on sub-projects and these groups must work together to complete the whole project. This part of the report is shared among every sub-group and contains the exact same content in every report. In this part "we" refers to all 14 members of the multi-project.

1.1 E-Learning

The term e-learning covers all forms of electronically supported learning. E-learning is often associated with distanced learning and out-of-classroom teaching, but can also be used to support traditional teaching in class rooms. In short e-learning is defined as learning that is facilitated and supported via Information and Communications Technology (ICT). The cooperation between the teachers and students and among the students themselves can similarly be partially or completely conducted via ICT [2][3].

At Aalborg University e-learning is employed in a variety of forms. There are fully online courses taught in the Master of Problem Based Learning (MPBL) education [12], and regular courses that make use of online quizzes as a method of teaching.

linjen oven over denne todo skal vaere udkommenteret med mindre der er nogen der har en kilde paa det

1.1.1 Learning Management Systems

E-learning is often conducted through the use of a Learning Management System (LMS). An LMS is, loosely defined, as a software system, which administrates, tracks, and reports teaching. It is considered robust if it contains the following functionality [37]:

- Has centralized and automated administration.

- Uses self-service and self-guided service.
- Is able to assemble and deliver learning content rapidly.
- Consolidates teaching activities on a scalable web-based platform.
- Supports portability and standards.
- Has the ability to personalize content and reuse knowledge.

LMSs have many forms each catered to a specific target groups. The general characteristics of an LMS are [47]:

- Student registration and administration.
- Management of teaching events such as scheduling and tracking.
- Management of curricula and obtained qualifications.
- Management of skills and competencies (mostly for corporate use).
- Reporting of grades and approved assignments.
- Management of a teaching records.
- The ability to produce and share material relevant to courses.

The purpose of an LMS is to handle and administrate all study related activities at a learning institution.

Moodle (Modular Object-Oriented Dynamic Learning Environment) [10] is currently the primary e-learning platform at AAU. Its main purpose is to allow lecturers to share course-relevant material with students and to serve as a calendar service containing dates of lectures, meetings etc. The problem is that Moodle does not support PBL, which is the learning method used at AAU.

1.2 The Aalborg PBL Model

The Aalborg PBL model is a term coined to describe Aalborg University's problem based learning model. It originates from the philosophy of the staff of the university. They were interested in giving the students an active role in obtaining knowledge, as opposed to the lecture based learning method used at many universities. Furthermore, they wanted to give the faculties a more active role in the students' learning experience than what the lecture setting provided. From this the Aalborg PBL Model was developed. This section is based on [30] with supplementary literature from [49, pp. 9-16].

The Aalborg PBL model consists of six core principles, which outline the students' learning process. These are:

- **Problem orientation** - A problem relevant to the students' field of study that serves as the basis for their learning process.
- **Project organization** - The project is the medium through which the students address the problem and achieve the knowledge outlined by the curriculum.

- **Integration of theory and practice** - The curriculum and staff at the university are responsible for teaching the students to connect their project work to broader theoretical knowledge.

- **Participant direction** - Students define their problem and make decisions relevant to the completion of the project themselves.

- **Team-based approach** - Most of the students' project related work is conducted in groups of three or more students.

- **Collaboration and feedback** - Peer and supervisor critique is used continuously throughout a project to improve the students' work. The aim is for the students to gain the skills of collaboration, feedback, and reflection by following the Aalborg PBL model.

From this point on we will use the term "project group" to describe a team of students working on a project in cooperation.

1.3 Moodle

Moodle is an e-learning platform for producing dynamic web sites for courses. It is written in PHP and supports SQL databases for persistent storage. Moodle is originally developed by Martin Dougiamas in 2002 and is released under an open source license (GPLv3+) [7][58]. It is currently maintained by a community of developers. Due to its modular design, the functionality of Moodle can be extended with plugins developed by the Moodle community.

Moodle is built around the concept of courses, and most activities in Moodle are centered around them. Courses can be divided into categories. Topics, resources, activities, and blocks can be added to courses [36]. Topics are an integrated part of courses, and resources can be links to external sources or uploaded files. Both these elements are an integrated part of courses. Activities and blocks are plugins, which can be added to a course to provide additional functionality. An activity is added by placing a link on a page, where the functionality of the activity lies. A block can be shown visually on the course page. An example of a Moodle course page can be seen in Figure 1.1

As Moodle is built around courses it does not provide much functionality to support the Aalborg PBL model discussed in Section 1.2.

The individual groups will elaborate further on Moodle if needed. Based on the content described so far we define our problem in the following section.

1.4 Problem Definition

The problem that this project is concerned with is the following:

Moodle does not fully support the work method used at AAU. Moodle is built up strictly around courses, and does not contain the concept of project groups. To accommodate for this students must use different tools for project group work. This project deals with how support for the Aalborg PBL model can be implemented in Moodle. This involves researching other systems than

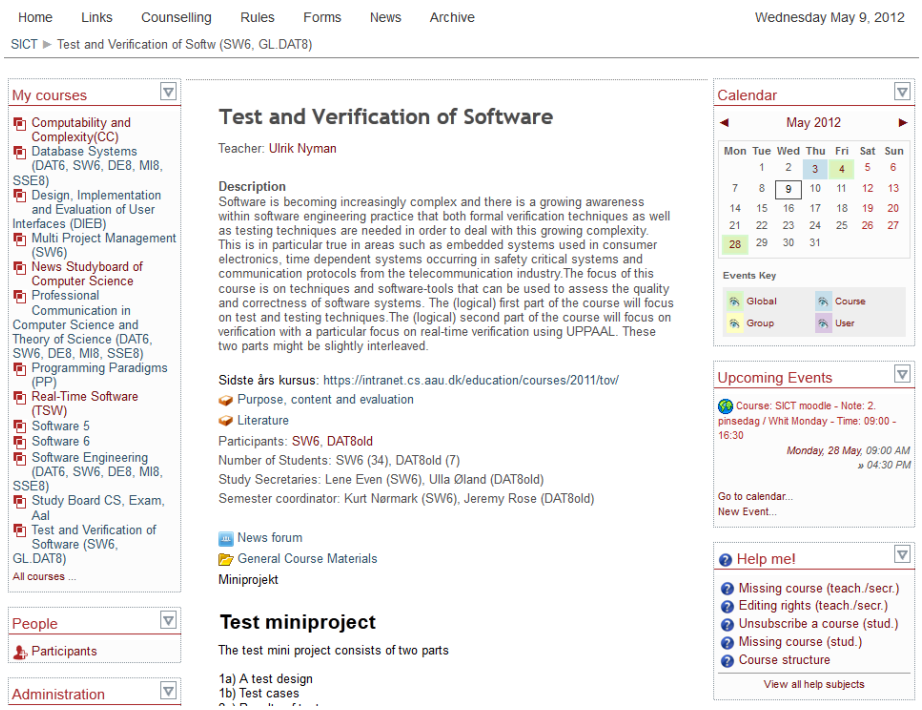


Figure 1.1: *A Moodle course page for the Test and Verification course.*

Moodle for information on how they accommodate project groups, and conducting interviews with students and administrative personnel of different faculties to gather requirements for such an extension to Moodle.

Chapter 2

Related Work

We are not the first to undertake the task of implementing support for the Aalborg PBL model in an LMS. As Moodle is not the only LMS, it is relevant to examine other LMSs to gain an understanding of how they are typically structured. These may provide inspiration for developing a new system. The project that the students from last year worked on is on the same topic, and it is equally relevant to examine their results and extract key points. There exist several plugins to Moodle that enhance it in respect to the Aalborg PBL model. We will compare the most promising of these plugins and conclude if they can be used.

2.1 Learning Management Systems

In this section we will examine a number of existing LMSs. The systems that will be examined are SharePointLMS, Litmos and Mahara. We will examine how they handle students, courses, and the concept of groups. The objective is to gain an understanding of how an extension to Moodle could be structured, and which tools might be relevant to include in the system.

2.1.1 SharePointLMS

SharePointLMS [13] is an LMS based on the Microsoft SharePoint platform. It offers the basic functionality of an LMS such as course management, student assessment tools such as quizzes and course certificates, conference tools, and document sharing.

In terms of PBL, SharePointLMS offers some relevant features, such as creation of groups. Within a group it is possible to enable features such as a chat, an internal mail, a calendar, and an online conference tool for meetings, which could be useful when working in a PBL context. The only drawback is that groups cannot be created as independent entities in the system, but have to be created in the context of a course. This does not fit well into the Aalborg PBL model, where there is a clear distinction between courses and projects.

2.1.2 Litmos

Litmos [8] is a lightweight LMS with a focus on being easy to use and set up. Its main features are creation of courses including multimedia content such as audio, assessment of students, and surveys to gain feedback on courses.

In contrast to SharePointLMS, Litmos supports creation of groups as independent entities and even creation of groups within a group. A group can be assigned to a course, which could be utilized to model the way courses are structured at AAU. This could be achieved by creating one group containing all students on a given semester, and then assigning this group to the relevant courses. Within this group a number of sub-groups would be created, representing project groups. Litmos has a built-in mailing system and is also integrated with Skype, which could be used to communicate within the groups.

2.1.3 Mahara

Mahara [9] is technically not an LMS, but a Personal Learning Environment (PLE), meaning that it is more learner-centered, as opposed to LMSs, which are typically more institution-centered. However, Mahara still has some features that are relevant to the Aalborg PBL model.

Mahara aims to be an online portal where students can share their work and be a member of communities within their area of interest. It does not, however, come with a built-in calendar, which makes planning of projects an issue when just using Mahara. Mahara compensates for this by providing a single sign-on bridge to Moodle, allowing users to access their Moodle accounts directly from Mahara without signing in again, and vice versa.

The aspect of Mahara relevant to the Aalborg PBL model is its social networking feature, which allows users to maintain a list of friends and create groups. Within a group it is possible to create a private forum as well as share files.

Overall Mahara provides a good platform for communicating within project groups, but it lacks support for planning and coordinating the projects.

2.1.4 Comparison

None of the examined LMSs provide complete support for the Aalborg PBL model. However, they do provide a variety of features, that if combined, would provide the functionality one would expect a PBL-oriented LMS to have.

The group structure found in Litmos could be combined with the features found in SharePoint to create a portal where students could organize their group work. The functionalities typically provided in LMSs appear to be mostly forums, internal communication and planning tools. None of the examined systems consider the aspect of communicating with a supervisor.

2.2 Relevant Student Reports

During the spring semester of 2011 a group of students at AAU were given the task of solving a problem similar to ours. They were to develop an LMS as a multi-project consisting of four groups, with the goal of fulfilling the needs of the students and faculty of AAU. However, the solution they developed was created

from scratch, unlike our solution, which is an extension of an existing system. Furthermore, the focus of the earlier project was on developing a solution which would fulfill the needs of teachers and students in relation to their courses, whereas our project focuses on developing a solution to better facilitate PBL.

Despite the differences between our project and theirs, we conclude that it would be beneficial for us to examine their development methods and try to learn from their experience, as the multi-project approach, target group analysis, and product testing aspects still remain highly relevant for us.

We examine the following reports: *E-LMS - Authentication, Database, and Educator* [33], *E-Learning Management System: Implementing Customizable Schedules* [31] and *E-LMS - Administration, Calendar, Model, Education, Courses* [56].

The following is a list of relevant and interesting observations that the groups made in their reports:

- Each group was allowed to pick their own development method, which lead to problems later in the development process as some groups used an agile development method, while others took a traditional approach. Integration testing became a problem as the groups were not synchronized and were always in differing stages of their development process. Agreeing on a shared development method would have been preferable.

- Each group sent a representative to meet with representatives from other groups at least once every fortnight, where each representative presented their progress since the previous meeting. As these meetings were primarily for status updates, very little coordination between the groups took place.

- Every month there would be a large meeting where all the students and supervisors were present and the current status of the system was presented.

Despite that we can not use the product that was developed by the students last semester, we can still use their experiences to improve this project and not make the same mistakes as they did.

Chapter 3

Initial Requirements

To determine how to improve Moodle to support the Aalborg PBL model we conduct two informal discussions with E-Læringssamarbejdet ved Aalborg Universitet (ELSA) [5] and the Master in Problem Based Learning (MPBL) department [12] at Aalborg University. This section accounts for the results of the discussion and based on that we decide how to solve the problem explained in conceptual terms.

3.1 Discussion with Expert Users

We conducted meetings with ELSA and MPBL. In both meetings we discussed which improvements we could make to Moodle in regards to the Aalborg PBL model.

3.1.1 ELSA

- Type: Informal meeting
- Participants: Lillian Buus [35], Marie Glasemann [40], Mads Peter Bach [29], and us
- Date: February 6, 2012

ELSA is responsible for technical, organizational, and pedagogical support of Moodle. [5] However, the actual development on Moodle is outsourced to the different IT departments on the university. ELSA receives all feature requests and is responsible for providing support to the departments of the university, therefore they have keen knowledge about the problems and shortcomings of Moodle. We are only interested in improving Moodle in relation to the Aalborg PBL model, therefore we have chosen five subjects of those ELSA proposed that we deem are relevant to this project:

3.1.1.1 Automation of input data

A course is automatically generated from a template. This empty course now needs to be populated with content; this requires manual work. The population of a course is primarily done by the administrative personnel and the lecturers.

ELSA would like that this could be generated automatically based on central data. ELSA would prefer that new entities introduced by us were also generated automatically.

3.1.1.2 Maintenance of data

Courses are, as mentioned above, maintained by the administrative personnel and the lecturers.ELSA mentions that we should consider who should maintain the data introduced by features implemented by us.

3.1.1.3 Overview of data

When a person is enrolled to several courses, the task of finding and entering the course page becomes difficult. ELSA would like a function in Moodle that renders a simpler overview, which eases the task of finding the wanted item. We need to ensure that this problem does not occur in any features we develop.

3.1.1.4 Sharing of data

The ability to share files and relevant material is a central concern when engaged in project group work. ELSA would like that features were more general such that they could be used in different contexts. This could be the ability to share quiz question between courses.

3.1.1.5 Archiving

ELSA currently archives all courses and the related material. The archiving is done by closing all courses for enrollment and then copying all data from the Moodle installation. This approach is not working very well, since students complain that they can not enroll or unenroll from courses during the backup process.

3.1.2 MPBL

- Type: Informal meeting
- Participants: Jette Egelund Holgaard [46], Morten Mathiasen Andersen [28], and us
- Date: 08-02-12

The MPBL department defines themselves as: “Master in Problem Based Learning is a fully online and highly interactive e-Learning programme for faculty staff at institutions who want to change to Problem Based and Project Based Learning (PBL)”. [12] At the department of MPBL they educate staff from other educational institutions in how to conduct PBL. Moodle and other communication technologies are used in the education process. The duration of the online education is two and a half years. To gain knowledge about how Moodle is used in professional and PBL context we conducted a meeting and discussed how Moodle can be improved to better support PBL. We choose to consider four of the subjects MPBL suggested:

3.1.2.1 Joining of tools

MPBL explains that they are using Moodle as a part of their education but it does not support any functionality that aids PBL. To facilitate the online education other technologies such as Skype [15] and Adobe Connect [1] are used. MPBL would like that the necessary tools are available in Moodle to avoid using several external tools.

3.1.2.2 Nearness

When working as a group it is an advantage to be in the same room. When that is impossible, which is often the case for MPBL, then tools must create the feeling of nearness. A virtual meeting place made available could create the wanted feeling of nearness and thus the foundation of working as a group is set.

3.1.2.3 Collaboration

Management and sharing of documents between the members of a group is a common issue. There is a need in Moodle to better handle sharing of documents. The ability to comment on documents are a common way to give feedback on other's work. Currently document sharing is conducted via emails, Dropbox [4], and Google docs [6].

3.1.2.4 Planning

When group members are geographically separated it is essential to coordinate and plan. MPBL would like that Moodle contains a tool to plan the collaboration process.

Chapter 4

System Definition

Based on our Problem Definition, Initial Requirements, and Related Work, we define the system that we develop in this project, MyMoodle, as follows:

> *MyMoodle is an extension to Moodle that allows Moodle to support the Aalborg PBL model.*

Since MyMoodle is an extension it adds functionality, but does not change existing functionality. We do, however, allow changes to the user interface of Moodle.

4.1 Decomposing MyMoodle

The project that we are undertaking is rather large. This leads us to decompose it into smaller sub-projects. This also makes sense with respect to the context that the project is being conducted, namely an educational context where we are to work in one large group divided into smaller sub-groups. Each sub-group will have its own sub-project, whose goal is to make a given sub-system, integrate it with the other sub-systems, and write a report to document the process and the results [19].

The final system, as defined above, should enable PBL in Moodle. We will divide the system into sub-systems to ensure that we cover the core principles of the Aalborg PBL model as defined in Section 1.2. The following core principles we find relevant to this project are:

- Project organization
- Participant direction
- Team-based approach
- Collaboration and feedback

These principles are important for the system, and they are all important for each sub-system. However, each sub-system has a different priority of the different core principles. Notice that two of the core principles are not considered. These two, problem orientation and integration of theory and practice,

are important concepts, but we do not believe that we can cover these in an LMS. These must be covered by the students themselves with guidance from their supervisor.

There are several ways to decompose MyMoodle to integrate the core principles of the Aalborg PBL model in Moodle. To support the principles of project organization and team-based approach we decide to implement the concept of project groups in Moodle. One sub-system, called Project Group Management, implements this concept. To implement collaboration and feedback we need a tool for planning, a tool for collaboration between the members, and a tool for communication with the supervisor(s). The principle of participant direction is implemented in general by all parts.

Each sub-system is described in the following sections.

4.1.1 Project Group Management

A project group is a team of students working on a project. It should be possible for students and their supervisors to work together on a project. It should be possible to create and manage project groups in a simple and intuitive manner. The nearness request made by the MPBL department in Section 3.1.2 should be satisfied in this sub-system by giving participants of a project a virtual meeting place.

4.1.2 Project Planning

The students can plan the way they want to organize their projects themselves. This sub-system is responsible for allowing students to make a schedule for a project and assign tasks to students. A student that is participating in several projects concurrently should be able to get a collective overview of his schedule. This also corresponds to the request made by ELSA to allow sharing of data described in Section 3.1.1, along with the planning request made by MPBL department in Section 3.1.2.

4.1.3 Intra-group Communication

During a project, the participants must be able to communicate and thereby direct the course of their project. The communication must be persistent, such that the participants can use their communication as documentation should it be needed. In Section 3.1.2 MPBL describes that they would like to have better collaboration between students in Moodle.

4.1.4 Supervisor Communication

The participants must be able to communicate and receive feedback from their supervisor. Students should be able to give feedback to projects, provided that the participants of the project allows it. As with the Intra-group Communication sub-system, this sub-system should cover the MBPL departments request for collaboration.

Chapter 5

Methods & Tools

In this chapter we describe the development methods and the tools, which are considered to be used in this project. A choice is given for tools and development methods along with a reasoning for the choices.

5.1 Development Method

In this project we are four project groups working together. For this collaboration to work it is important that we have a common understanding of the development method we are using. The groups that were developing an LMS last year used different development methods, which gave them problems during development of their product (see Section 2.2). Different development methods are presented here and a single one is chosen to be used in this project.

5.1.1 Considered Methods

We consider some traditional and agile development methods. In general, traditional methods follow an upfront planned schedule, where every task is handled in a single unit and the result is not changed afterwards [54, sec. 2.7]. This method is inspired by the construction industry from where it originates.

Projects using agile methods are developed in iterations, where some part of the product is developed in every iteration [50, p. 25]. This should help cope with the changes that the end users or customers might pose during the development.

Some methods can have characteristics from both categories. The development methods considered in this project are: Extreme Programming, Scrum, and Waterfall. These are presented shortly in the following section.

5.1.1.1 Extreme Programming

Extreme Programming (XP) is an agile development method that consists of a series of 12 recommended core practices [50, p. 137]. These include, but are not limited to: Frequent refactoring, pair programming, and the whole team working together in a single room. The core of XP is to find every practice that is considered good and taking it to the extreme, e.g. since code reviews are good, do them all the time through pair programming. Kent Beck, the creator

of XP, states that in general all the practices of XP should be applied because they compensate and support each other [50, p. 156-157].

There are roles assigned to different people involved in the project [50, p. 145]. These roles are: Customer, programmer, tester, coach, tracker, and consultant. All these roles are important, but Larman stresses that an on-site customer or at least an on-site customer proxy is needed [50, p. 152-156].

When using XP, the development is to progress in iterations of one to three weeks each. An iteration should not be planned until right before the start of it. At the start of the iteration the on-site customer should help prioritize which features should be implemented in the given iteration and the programmers estimate the time to implement them. This process is called "Iteration planning game".

5.1.1.2 Scrum

This section describes the agile development method Scrum. The knowledge of Scrum is from [50, chap. 7,pp. 109-136]. Scrum is, as XP, an agile development method that utilizes a number of iterations of development cycles. These development cycles are known as sprints. A sprint usually has a length of 30 calendar days. A sprint backlog is created prior to the sprint. This backlog consists of the features, ordered by priority, that should be implemented during the sprint. During the sprint the sprint backlog cannot be changed. If some of the features in the sprint backlog are not implemented during the sprint they are moved to the product backlog, which is a list of the features that should be implemented in the future.

In Scrum there are different roles. There should be a product owner, whose task is to meet the costumer's and end users' interests. These interests should be formalized and prioritized in the product backlog. There is also a Scrum master. A Scrum master serves as a link between the development team and any external individuals. It is the Scrum master's task to ensure that the development team is not disturbed. The development team is usually small, and their task is to design, analyze, and implement the features from the backlogs. Every day the Scrum team holds a short meeting where they say what they have done since the preceding meeting, what they plan on doing the present day, and any problems they are having.

A variant of Scrum for more teams is called Scrum of Scrums or Large Scrum [41, pp. 23-30][50, p. 111]. In such a project there is one complete team consisting of several smaller sub-teams that will hold ongoing Scrum of Scrums meetings during each sprint to synchronize their work. The Scrum of Scrums meeting is similar to the daily Scrum meeting with a representative from each sub-team, except that they talk about what the sub-teams have done and should do instead of what individuals have done and should do.

5.1.1.3 Waterfall

A project following the Waterfall method is divided into a number of phases. The number of phases in a Waterfall development process varies from implementation to implementation. One version of the waterfall model defined at [52] divides the waterfall model into six phases:

- **Requirement gathering and analysis** - The initial phase where all possible requirements are gathered from the end users of the system. Their validity and the possibility to implement them is analyzed. The result of the phase is a requirement specification document, which is used in the next phase.

- **System design** - Based on the requirements a design for the system is constructed. The result is a system design specification document.

- **Implementation and unit testing** - Based on the design document the system is divided into modules and units. The coding is started in this phase. Upon completion every unit is tested.

- **Integration and system testing** - All the developed units are combined into the final system. The integration of the units is tested. The complete system is tested to ensure it satisfies the requirement specification document. After a satisfiable number of tests have been conducted the system is delivered to the customer.

- **Deployment** - Depending on the system some initial setup and configuration of the software may be necessary before the end users can use the system. The result of this phase is a deployed system that is set up to the end users' needs.

- **Operations and maintenance** - A theoretically never-ending phase. After the software is delivered issues may arise from practical use of the software and bugs may be discovered. When issues arise they are dealt with as long as the software is in use.

Waterfall models are strictly traditional, since it in its pure form does not allow to move back to a phase once the next is started. This suggests a big and heavy up-front design plan that is to be followed until the project is finished.

Of the presented methods Waterfall is the method that requires the most documentation. To complete a phase some document must be created to be used in the following phases. These documents can vary from an architectural design document to source code.

5.1.2 Choosing a Development Method

The following list shows the characteristics of this project, which will be used to determine the development method for the project.

1. **Four groups (14 persons in total)**
   This semester is divided into two major groups, one of which is working with Moodle. This semester at AAU has a group limit of four members therefore the Moodle project has been divided into four smaller projects.

2. **Diverse target group**
   The relevant target group is: Students, supervisors, and administrative personnel

3. **No on-site costumer**
   Because this is a university project, there are no on-site customers. Instead there are several contact persons such as supervisors, students, and administrative personnel, which test the product and provide feedback.

4. **Hard deadline**
   Because this is a semester project the project must be completed when the semester ends.

5. **Pass on project**
   This project will be passed on to the students next year.

6. **Known framework and platform**
   The Moodle platform is open source, and some documentation is available on most relevant topics.

7. **Education environment**
   Since this is a university project, we are working in an educational environment.

8. **Not full-time development**
   Because this is a semester project there are also lectures beside the project, therefore the time for development is limited.

9. **No manager/Product owner**
   Because this is a semester project, there is no project manager or product owner like there would be in a corporate environment.

10. **No shared working room**
    We do not have a room available where all four groups can work simultaneously.

11. **Low criticality**
    If the system fails it will only affect the comfort of the users.

Barry Boehm and Richard Turn have identified five factors that can be used to determine whether to use a traditional or agile development method [32]. These factors are: Personnel, dynamism, size, culture, and criticality.

The personnel factor covers the composition of personnel based on the *extended Cockburn method skill rating scale*, where people are divided into five categories based on the methodological skills, namely levels -1, 1b, 1a, 2, and 3 [32, p. 34]. The levels of the extended Cockburn scale are defined as follows: A person with level -1 is unable or unwilling to follow a shared development method. With training a person with level 1b can perform procedural development method steps such as writing a function while conforming to coding standards. A trained level 1a person is able to perform discretionary development method steps such as using design patterns to solve a problem. A person with level 2 can alter a development method to handle a new but similar situation. A person with the last level, 3, is able to alter a development method or create a new one to handle a new unfamiliar situation.

Dynamism is the anticipated percentage of changes in requirements that will occur during the project. The size is simply the number of people in the development team. The culture factor is a scale of how much the team prefers

chaos over order. Criticality is a scale of how much a system failure will influence the real world. This is based on the Cockburn Scale used to differentiate between Crystal methods [50, pp. 36-37].

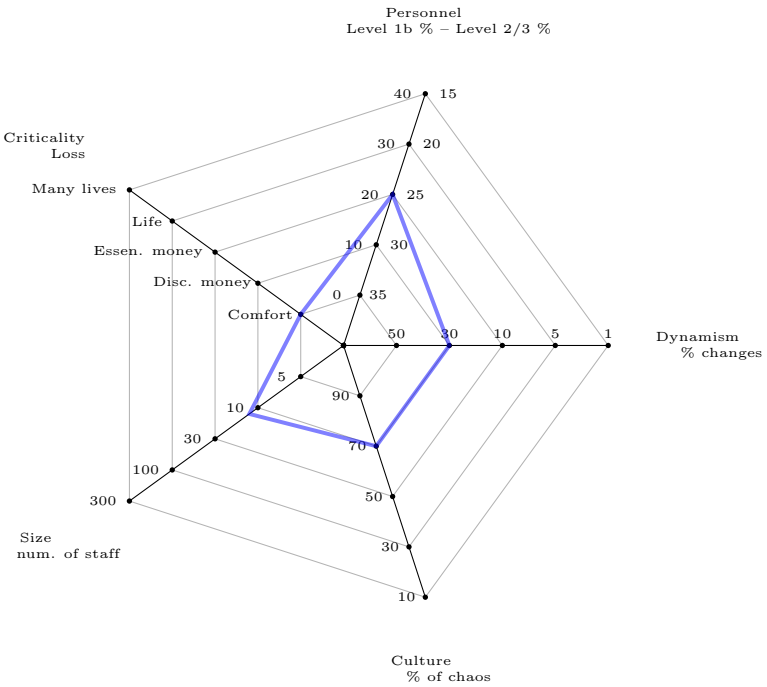


Figure 5.1: *Radar chart showing the deciding factors for choosing the development method of our project.*

A polar chart showing the factors for our project is seen in Figure 5.1. As seen, most of the points are closer to the center than the periphery, this indicates that an agile method is preferred over a traditional one.

The personnel score is positioned in the middle because we consider ourselves above level 1b, but none or very few of us qualify as level 2 or 3.

We will use a development approach similar to Scrum of Scrums. The reason for this is three-fold. First of all we have a diverse target group (item 2), which may make a big up front analysis and design difficult. This leads us to choose an agile method due to "I know it when I see it" (IKIWISI), that is the customer does not know what she wants before she sees it. Our choice is also supported by the polar chart showing the development factors in Figure 5.1.

Secondly we are 14 members divided into four groups (as item 1 states), which is not handled very well in other agile methods such as XP, which dictates that all the developers should be in the same room. This is not the case for us as item 10 states.

The third reason is that we have a hard deadline (item 4), which means that we have to hand in our project at a specific date. Scrum of Scrums suggests that iterations (sprints) are time-boxed, which is ideal for us since we can cut less important features instead of missing the deadline. This is also supported by item 5, because the end product is a working release although some features

may have been cut. The features cut may then be suggested to the group of students, who are to take over this project next year.

5.1.3 Refining Scrum

As items 9 and 3 state in the previous section, we have neither a project manager nor an available on-site customer. We will handle the missing on-site customer by having shorter iterations and contacting the customers whenever an iteration is over. Scrum of Scrums [17] dictates that there should be a Scrum master in each sub-group. Since none of us have used Scrum before, none of us are qualified to be a Scrum master. We are in an educational situation (see point 7) so we will strive to allow every member to try to be Scrum master for a shorter time period. This may not be ideal, but we consider it to be more important that every member of the sub-groups tries to have the responsibility of a Scrum master than having only one member trying it and learning it well.

5.2 Tools

A series of tools are used in the creation of this project. These tools are briefly described in this section. The tools that need to be chosen must complete some task. The tasks that need tools are: Version control, bug tracking, code documentation, and testing.

5.2.1 Version Control

All of us have been using subversion (SVN) in previous projects as the system for version control. SVN is a centralized solution [16] with a single repository that the group members can update from and commit changes to. This project does, however, differ from previous project with respect to the organization of groups; we are not one group of x individuals, but rather one group consisting of some smaller sub-groups. This has lead us to choose a distributed solution rather than a centralized one.

The solutions considered are the distributed systems Git and Mercurial (Hg). These systems are quite similar and the essential difference is that Hg is simpler than Git and Git is more flexible than Hg [18]. A few of us have been using Hg and none of us have used Git, which leads us to choose Hg, such that we have a little knowledge of the chosen system.

5.2.2 Bug Tracking

We need to have some way of communicating and tracking the defects or "bugs" that we will run into during our project. The most important requirement for the tool we will use is that it should be able to track the bugs and make them easily available to the team that will continue on this project next year. The tools that we are considering to use for bug tracking are Bugzilla [34] and Eventum [24]. These tools are very similar in their sets of features. When a bug is discovered a bug report must be added in the tool. In the tools the members of a team can see the bugs and their status, and mark them as fixed when they are handled.

Since we have used Bugzilla as part of the course Test and Verification, it will save us the overhead of having to learn a new tool by choosing that over

Eventum. Furthermore, development of Eventum seems to be discontinued since the start of 2009 [23], which means that any defects that may be in the system are very unlikely to be fixed. In conclusion we choose to use Bugzilla to track our bugs.

5.2.3 Code Documentation

We do not wish to use a tool that requires us to type in documentation externally from the code. We want to use a tool like PHPDocumentor [43] or PHPXref [57] to handle our documentation, since both of these read comments in the source code and use it as documentation. This gives us the ability to write code and documentation in the same file. The syntax that both of these tools use is the same, which means that as long as we adhere to this syntax we can use either tool to generate documentation.

The difference between the two tools is that the focus of PHPDocumentor is on giving a more diverse set of final documents (different HTML and PDF templates), where the focus of PHPXref is more on the references between classes, functions, etc. We use both tools such that we and the group that is to take over this project next year can choose the documentation they prefer.

5.2.4 Testing

We want the tests to remain after our work on the project ends, in particular the groups that are to take over this project next year should be able to use our test cases again. We will be using the built-in testing framework of Moodle [11], which is based on SimpleTest [14]. We do not consider other tools, since it reduces the number of tools we have to use by using one already part of Moodle. This should ensure that the test cases can be used next year and be part of the final product, should it be released to the public.

Part II

Development

Chapter 6

Introduction

Imagine an LMS that incorporates course management and the Aalborg PBL model along side each other. This is the main idea behind this project.

As of this part only the authors noted in the title page have been working actively on this report, with help from our supervisor. Hence "we" only refers to the four authors of this report, and not the 14 collaborators of the entire system.

During the development of this project we use an agile development practice. The development practice is explained further in Chapter 7. The report structure and content does not reflect the use of an agile approach and is written as it was developed using a traditional development approach. We structure the report in this manner to make it more readable and comprehensible.

It should be noted that the final system, MyMoodle, is supposed to be a complete system, not a set of independent systems. This means that we will be referring to the other sub-systems whenever there is a dependence on them or we are making services that the other sub-systems depends on.

This project is concerned with integrating the Aalborg PBL model, defined in Section 1.4, into Moodle. Recall Section 1.2 where the concept "Team" was presented, we will now use it to define the term "project group". Our sub-group has the responsibility to develop a sub-system that allows a team to work together through Moodle and some way to manage who is part of which teams with what roles. We use the term "Project Group" to describe a project with a belonging team. The relation between a team and a project will be define in Section 10.1.

To make MyMoodle usable at any educational institution we provide a tool to manage project groups – e.g. allowing creation, modification, and archiving of project groups. The members of the project group will have access to some shared activities through which they can conduct their group work. These activities include planning the progress of the project, communicating internally while the project is being conducted, and communicating with people not actively part of the project group, such as supervisors. These activities are created in our peer sub-projects (as mentioned in Section 4.1). Our responsibility is to make all these activities available to the members of the project groups.

Chapter 7

Development Practices

This chapter gives a description of how we are using the development method Scrum of Scrums. Where Section 5.1 is regarding the development method of the entire group, this chapter regards how we are implementing the development method in our sub-group. There is some repetition from Section 5.1, to recall the decisions made previously.

7.1 Utilizing the Development Method

Since we are not familiar with Moodle, from the beginning of the project we do not have a full understanding of how the entirety of the system should be structured and created. We solve this problem by using the development Scrum and further dividing the development process into a number of sprints.

Our first sprint will consist of information gathering; we study the Moodle platform as well as interview our costumers in order to get a few initial requirements. We plan this non-programming sprint to learn and experience the Scrum development method before we start on a programming sprint. The requirements we gather are used to create feature descriptions, which are used to fill the product backlog. At the beginning of each succeeding sprints we choose items from the product backlog and move them to the sprint backlog, which is a list of features we expect to implement in the particular sprint.

In order to choose which items to move to the sprint backlog we assign each item, currently in the product backlog, a number of story points. We do this by playing planning poker, which is a card game where we all give our estimates of the items on the backlog simultaneously, and then discuss the estimates if there is a significant difference between us. Based on the estimate we choose which items we are able to implement in this sprint.

As we progress in the sprint the number of remaining number story points starts to dwindle. We keep track of this by a burn-down chart, which is a physical chart with a line that shows the expected progress. The total number of remaining story points are plotted into the graph each day, so we can see if we are progressing at a satisfactory rate.

In our sub-group we start each day with a Scrum meeting. In this short meeting we all stand up and each tell three things: What we did since last Scrum meeting, what we are going to do today, and which – if any – impediments we

have. This gives the entire group and idea of what is being worked on, and by doing this everybody always have a task they have chosen themselves.

7.2 The Development Method Across Groups

Since all sub-groups are dependent on each other in the multi-project, we need to organize what each sub-group is doing.

At the beginning of each sprint all the sub-groups present their plan for what they are going to produce in the coming sprint. If other sub-groups have any dependencies, they communicate these and the sub-groups collaboratively decide the overall tasks each group should accomplish in the given sprint.

At the end of each sprint the groups meet and present what they have created during the sprint. End-users can be invited to try or be showcased specific aspects of the system by individual sub-groups in order to acquirer feedback between sprint.

During sprints the sub-groups work together to some extent. Since we all work close to each other we can always go to each others project group room to ask for help or request that some specific work should be done. Additionally we hold Scrum of Scrums meetings at an approximately weekly rate. In these meetings the Scrum masters of all the sub-groups meet and discuss the direction of the project and share information regarding how the sub-systems should be integrated with the different components that the sub-groups are developing.

In the next chapter we will present our requirements.

Chapter 8

Problem Scope

In this chapter the scope of our sub-project is defined. To define the scope of this project we start with defining the characteristics of the end users of our sub-system. We are defining the sub-system that we will be developing in this chapter. To define this sub-system we will be gathering requirements from our end users.

8.1 End Users

Since we are using Scrum as our primary development practice we prefer to have a product owner [50, p. 115] that can speak on behalf of the end users. An end user is a person that will be using the final system. As mentioned in Section 5.1.2 we do not have a product owner thus we facilitate the contact and communication with the end users our self. In this section the group of end users for our sub-system are defined and the people that we use as representatives for this group are presented shortly – we cannot speak with the entire group of potential end users, because it is simply too large.

For the full system the group of end users is very diverse. In the sub-system which we are developing we are interested in two different categories of end users.

- The first group is the members of the project group.
- The second group is the managers of the virtual group room.

The two groups of end users can overlap. When choosing representatives for our user group there are several properties that we want to cover. These properties are: Type, Faculty, Department, and Moodle experience. We divide the types in three: Administrative personnel, supervisors, and students. Moodle experience is a scale from low to high. Faculty and department are the end user's actual department and faculty. We want users of each type and within these types we consider Faculty and Moodle experience as important factors to diverse. We would like to have representatives that cover these two properties as much as possible. That is, we want our group of end users to contain both well experienced and inexperienced users of Moodle and other LMSs, and they should be from different faculties.

The end users we are using come from Aalborg University. The reason for this is two-fold. Firstly we are implementing the Aalborg PBL model into

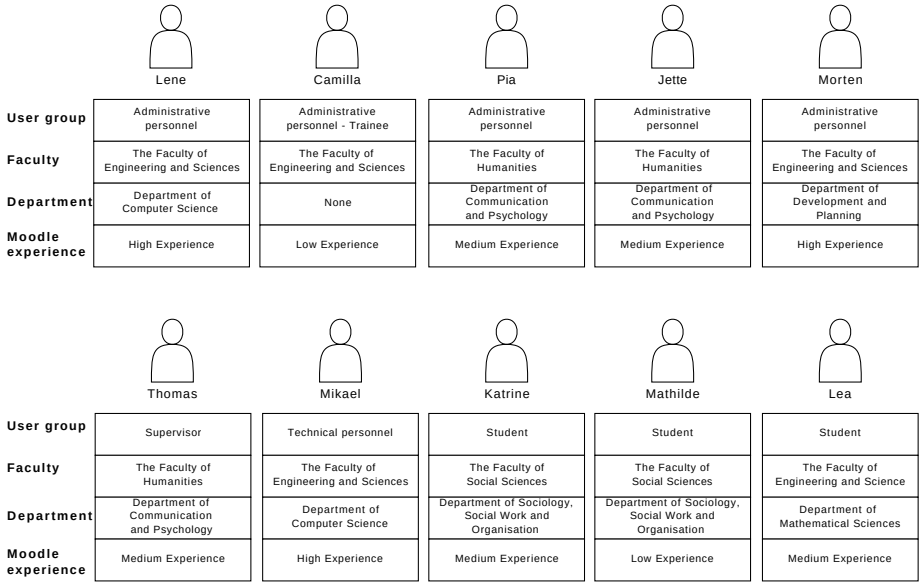


Figure 8.1

Moodle, which means that Aalborg University comes as a natural choice to look for end users. Secondly our project is being conducted at Aalborg University, consequently it is easy for us to find people to participate and the system is to be used at Aalborg University.

In Section 8.1.1 and Section 8.1.2 we explore the two different categories of user groups. An overview of the chosen end users can be seen in Figure 8.1.

8.1.1 Members of Project Groups

As discussed in Chapter 4 our responsibility is to create the concept of a project group and creating a tool to manage these groups. Since we are working together with three other peer-groups we will receive requests for functionality from them. Our peer-groups have their own end users, both students and supervisors, where they get their requests from. We can choose to regard students as a part of our end users and make our own field studies. Alternatively we can choose to rely on our peer-groups requests and effectively use our peer-groups as our target group. We choose a compromise and rely on our peer-groups requests and cooperate in the field studies related to the entire system or the concept of project groups.

The end users of the project group functionality are students and supervisors. We want to have student representatives of different types, in particular students from different faculties. We have two students from The Faculty of Social Sciences and one from The Faculty of Engineering and Sciences. These are respectively Katrine Holmgaard Dinitzen, Mathilde Gammelgaard, and Lea Gustafsson. Regarding supervisors, we choose to have a single representative; Thomas Ryberg Vibjerg Hansen [55], who is supervisor under The Faculty of Humanities. In Figure 8.2 can the users of the project group be seen. The figure also displays their properties.

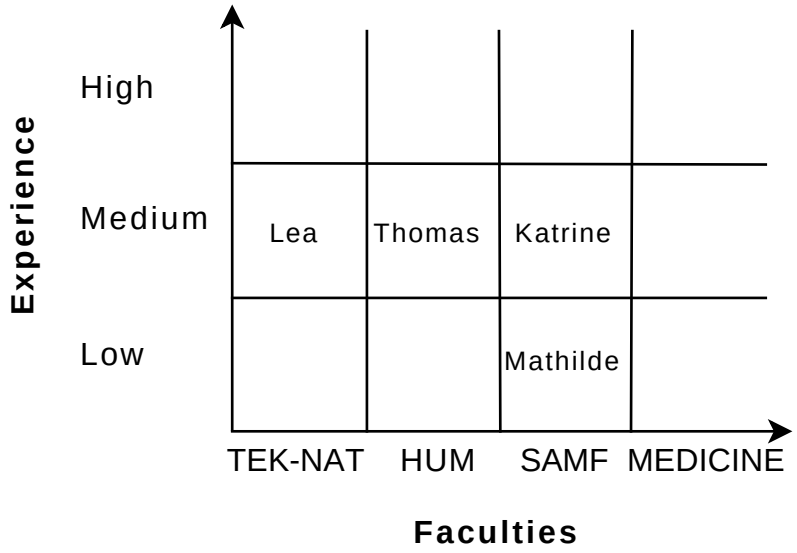


Figure 8.2: A graph showing how the end users representatives of project group members are spread between faculties and how much experienced they have in regards to LMSs.

8.1.2 Users of Administrative Tool

Regarding the administrative tool, we choose to use administrative personnel as our main target group. From our own faculty we have the representatives Lene Winther Even [38], Camilla Gæraa Larsen [51], and Mikael Møller Hansen [45]. Where the two former are a senior secretary and an office trainee respectively, and the latter an IT administrator From The Faculty of Social Sciences we have been in contact with Jette Due Nielsen [46] and Pia Knudsen [48] as our representatives. We have also been in contact with Morten Mathiasen Andersen [28] from MPBL in addition to the initial discussion discussed in Section 3.1.2. Morten is from the faculty of Enginerring and Sciences. Figure 8.3 shows the administrative personnel and in regards to their facility and experience.

8.2 Requirements

During this project we have had several meetings with people that we consider to be our end users. Some of the meetings were held before we coded the system and some of them during coding. In this section we present our the final requirements. The continues change in our requirements is evaluated in Section **??** in Part III. The interviews that have been conducted to gather these requirements are found in Appendix A.

Since we use Scrum as our main development practice we use a product backlog to capture our requirements [50, p. 114]. Different sources suggest different notations for what is in a backlog [41, p. 17][50, pp. 123-124]. Our backlog items can be are all use cases, features, or fixes. Since this project is to be proceeded by a new group of students next year, we allow our product

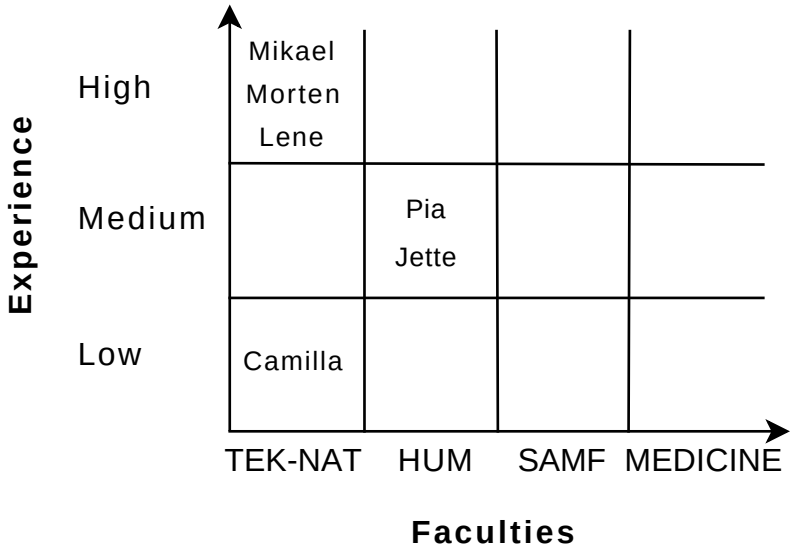


Figure 8.3: A graph showing how the end users representatives of the administrative tool are spread between faculties and how much experienced they have in regards to LMSs.

backlog to be larger than we are able to complete in this project. We will now focus on the items that are part of this semester project. These are said to be part of our release backlog. The backlog items that make up our product backlog for our sub-system are defined in Figure 8.4. We will now describe how we derived some of our backlog items.

Virtual Meeting Place The concept of a virtual meeting place comes from the meeting with ELSA described in Section 3.1.1. From the interview with Lene in Appendix A.1 we know that she manually types a list of groups and their members into Moodle. Moodle is not able to interpret this. In the interview with Jette and Pia (Appendix A.3) we learn that a member of the administrative personnel is responsible for the formed groups. This implies that there exist records of the composition of the groups. In contrast to the these two interviews Morten explains in an interview(Appendix A.2) that on the MPBL education the administrative personnel do not manage the project groups. Based on this information we decide that we will make a virtual meeting place that reflects the project groups that exist on Aalborg university. We decide this knowing that it will cause additional work for the administrative personnel on the MPBL education. We will elaborate on the virtual meeting place in Chapter 10.

Find Project Groups We deem that it is necessary to create a list with all the project groups that exist. This should be available to the part of our target group that manage the project groups. In our third sprint we held a demo meeting where Lene mentioned that it was difficult to find a project group when there were many project groups in the system(see Appendix A.6). We decide to

Name	**Description**
Manage Project Groups	A subset of our target group should be able to manage the project groups, this includes creating, editing, and deleting project groups.
Virtual Meeting Place	Create a virtual space that are compliant with the principals of the Aalborg PBL model and create a feeling of nearness.
Recursive Project Groups	*Allow project groups to contain groups of persons. This also include the administrative tools to manage the groups and sub-groups.*
Introduce Roles	Add the possibility to assign roles to the members of a project group, roles such as supervisor and student.
Manage Roles	*Create Functionality to define and manages roles. An example of a new role could be secondary supervisor.*
Find Project Groups	Produce functionality to find a given project group. This involves creating a list of project groups and making features to allow search or filtering.
Navigate to Project Groups	The members of the project group should have a way to navigate to it.
Project Group Members	It should be possible to see who the members of the group is, to create the feeling of nearness.
Project Group Synchronization	*Project groups should reflect the physical project groups at the university. Thus when a student joins an existing group, the student should automatically be added to the virtual group room.*
Virtual Meeting Place Template	*To account for the user groups diversity and different needs, a set of templates, that defines the visual appearance of the virtual meeting place, could be applied when creating a new virtual meeting space.*
Archiving Project Group	*ELSA*

Figure 8.4: *A list showing features from our product backlog. The italic items are not part of our release backlog.*

make functionality that will allow users to search in the list of project groups. The implementation of this concrete problem will be addressed in Section 12.3.

Manage Project Groups It should be possible to manage the project groups. We let us inspire by how courses are currently being created at Aalborg University. ELSA are responsible for creating new courses at the start of every semester. This is done by the administrative personnel from every institute informs ELSA of which courses that should be created. ELSA creates the courses based on a general templates. The administrative personnel and lectures then populates the courses with data. Information about this process can be read from Appendix A.1, Section 3.1.1, Appendix A.3, and Appendix A.5. Similar to the creation of courses it should be possible to create and in other ways to manage project groups. The design of project groups will be explained in Chapter 10.

We will not explain all the backlog items because we reckon that the derivation of them are of similar nature. We will now account for the choices for the release backlog.

8.2.1 Release Backlog

Recall the table seen in Figure 8.4. The release backlog is based on our product backlog. To explain the selection of backlog items we will focus on the items that we do not choose. These items are marked with italic on the figure. The four items are described below.

Recursive Project Groups In an interview with Lene (see Appendix A.6), we discussed the possibility of having a recursive group structure. Lene could not imagine any concrete cases where such a structure would be beneficial. An example of a place where there exist groups in groups is our current multi-project. The 6th semester project of the Software Engineering education anno 2012 is a multi-project, which means that the student are divided into two groups each with its own subject. The multi-groups are split into project groups of fitting size. Despite this we choose to prioritize other features that will bring more value to the project according to our target group and not implement recursive project groups. We will argue further for this in Section **??**

hvis det er beskrevet i analdesign så indsæt ref. ellers så slet denne sætning.

Project Group Synchronization In the computer science department of Aalborg university AdmDB is among other things used to manage project groups. During the interview with Mikael(see Appendix A.4) we learned that he is currently developing a new API for the AdmDB system, and it would not be optimal to rely on the old API. We also learned that there does not exist a university wide system that contains all project groups. We believe that if this feature should be implemented it should be after the new API is finished. We will elaborate further on this subject in Section 10.2.1.

brug fulde navn?

Virtual Meeting Place Template As previously mentioned ELSA is responsible for creating course pages in Moodle. They do this using a template. It is desirable to have the same functionality when creating the virtual meeting place, but this feature will not bring any immediate value to the product. We

postpone this feature until there arises a concrete need for such functionality. We further discuss this feature in Section **??**.

insert ref when the paragraph is written in the analdesign. If it will not be written delete this line.

Archiving Project Groups From the meeting with ELSA, see Section 3.1.1, we know they archive courses at the end of each semester. Lene Winther Even explains that it is to preserve the content for future reference, see Appendix A.1. Morten Mathiasen Andersen, see Appendix A.2, explains that teachers use the archived versions of a course to create a new course of the same subject. ELSA proposed that project groups can be archived for the same purpose so supervisors and students can look back at their previous work.

We believe that the archiving of project groups is a very beneficial feature for students, supervisors, and administrative personnel. However, since we have limited development time we suggest that future student projects could look into this feature. A further discussion on archiving can be found in Chapter 17.

The items in the release backlog are the following: Manage Project Groups, Virtual Meeting Place, Introduce Roles, Find Project Groups, Navigate to Project Groups, and Project Group Members. Based on our requirements we will now present our System Definition.

8.3 System Definition

Based on the System Definition and Requirements we define the subsystem we are implementing as:

> *A sub-system of MyMoodle that implements project groups in Moodle and allow for administration and usage thereof. The sub-system includes a virtual meeting place, which integrates the other sub-systems.*

The subsystem must conform to the overall system definition, defined in Chapter 4.

Chapter 9

Platform

In this chapter we present the constraints that Moodle set's for MyMoodle, the core concepts of Moodle, and the framework in which MyMoodle is integrated. The concepts that are presented here are: Courses, Groups, and Plugins.

9.1 Constraints

Moodle is written in the programming language PHP [59] and uses an SQL database. This restricts us to write MyMoodle in PHP as the server side scripting language, to allow usage of the Moodle functionalities – otherwise it would not be an extension to Moodle, but rather a new system. Since Moodle uses an SQL database we choose to restrict us to use that as well, to avoid adding unnecessary dependencies to third party systems.

9.2 Courses

Courses are a fundamental concept in Moodle, since Moodle is a course management system. Courses can use activity modules, which are described further in Section 9.4, and are often split into sections depending on the selected course format. One example of a course format is the weekly format which splits the course into a section for each week [64]. Every course in Moodle is part of a course category [63]. Course categories can be hidden such that students and lectures are not able to see the courses in these categories on the Moodle page, though administrators are always able to find courses even if they are hidden.

9.3 Groups

Moodle has a built-in concept of groups and groupings. Groups can contain users and groupings can contain groups. The database scheme of groups and grouping can be seen on Figure 9.1.

This structure supports that users can be in groups and that more groups can be grouped together in one grouping. One group can be a part of several groupings and a user can be a member of several groups. The foreign key from groups to course and the foreign key from grouping to course indicates a tight

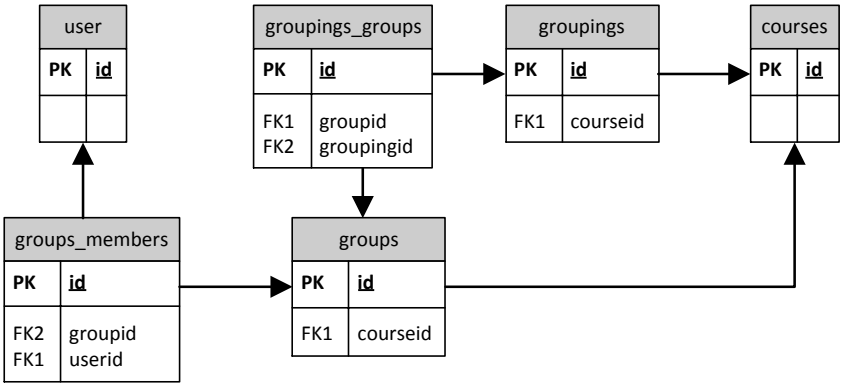


Figure 9.1: *The database scheme for groups and groupings. Non-primary key and non-foreign keys data fields are omitted for brevity [42].*

connection between groups and courses. In fact every group and grouping must refer to a course, to which the group or grouping is said to belong.

9.4 Plugins

As mentioned in Section 1.1.1 the M in Moodle stands for modular. Moodle currently supports 34 types of plugins [27]. The four most common is explained in the following:

Blocks Moodle use the term block to refer to a box that can contain a wide spectrum of information and functionality. A block can as default be placed in the left or right side of a page. Pages can allow placement of blocks in the center of a page. The navigation block [26] is a very common block which as default is displayed on all pages. It contain links such as: My Profile and Courses. Blocks are not limited to navigational purposes, but can also display data from the database.

Each block have settings regulating on which pages it can displayed.

Activity modules "An activity is a general name for a group of features in a Moodle course." [21] Activity modules are the main way to create new features in a course, and a new feature is called an activity. It is common that a course page contains an activity block [22], that contains available activities for the course.

Activity modules often contains a new page where an activity is available. An example of this could be a forum. On the course page there is a link, when it is pressed the forum page is displayed. An activity is saved with the course, this means that several individual forums can exist on different courses.

Activity modules are intended to be used when making a feature that interact with several users in the context of a course.

Admin tools Admin tools are, as the name implies, tools used for administrative purposes. The Moodle developer wiki define an admin tool as the following: "Provides utility scripts useful for admins to examine and modify a Moodle site". [27] The only users that can access the admin tools are administrators.

Admin tools are intended to be used when making tools for the administrative personal. These are made available through the Site administration menu point in the Settings block [61].

Local plugins Local plugins are the most general type of plugins. If the wanted feature does not fit the other types of plugins, it is recommended to use the local plugin. An example of this could be communication with an external system or for extending the navigation block.

9.5 Framework

Moodle includes various APIs [62], which aids the developer in the process of creating application. Some of the APIs are:

- Data Manipulation Language (DML) API [66]. Used for all persistent data access and is described further in Section 9.5.1
- Form API [68] for rendering HTML form objects and handle the validation of the data send back through the form.
- Output API [69] for general HTML rendering.
- Page API [70] for setup of a standalone page including methods for adding javascript and configuring display options.
- Access API [60] for controlling access levels of users throughout the Moodle.
- Unit test API [72] for testing components in Moodle.

ref evt til test(ven til test planlagt)

Moodle APIs are used in different ways. The DML, page, and output APIs are used through global variables, whilst others are used through class extensions. In Code Snippet 9.1 the page and form APIs usage is shown.

```
1 $PAGE->set_context($somecontext);
2
3 class some_form extends moodleform {
4     ...
5 }
```

Code Snippet 9.1: *Example of the Page and form APIs in Moodle.*

9.5.1 Database Layer

To access and manipulate data in Moodle the DML API is used. The API is accessed through a global variable named $DB. It has several method for update, insert, delete, and select. The basic method for select, insert, and update, namely get_record, update_record, and insert_record, take or retrieves an object of type stdClass. StdClass is a standard PHP class which is empty, but allows for arbitrary addition of properties at runtime. This gives a unified procedure for manipulation data. A simple update procedure is seen in Code Snippet 9.2. For simple operations no SQL is needed, but for more complex

```
1 function change_name($id, $new_name){
2     global $DB;
3     $user = $DB->get_record('user', array('id'=>$id));
4     $user->name = $new_name;
5     $DB->update_record('user', $user);
6 }
```

Code Snippet 9.2: *Example of how to change the name of an user.*

queries such as queries using joins SQL most be written directly.

9.5.2 Context System

The context system in Moodle is used to set the context of a given page to determine the users capabilities and which blocks and activity modules to present [71].

When a user is logged in to a Moodle system he can have various roles in various contexts. One person might be a student in one course and a teacher in others. Moodle use a hierarchical context system to manage users roles and their capabilities. Capabilities are further described in Section 9.5.3. The context systems hierarchical layout can be seen in Figure 9.2 [67].

The highest level of context is the system context which all inherit from.

Every page in Moodle which is loaded directly through HTTP(S) must set the page context. An example of how to set the context can be seen in code snippet 9.3 from /course/view.php, which loads the default page of a Moodle course.

```
1 ...
2 if (!$context = get_context_instance(CONTEXT_COURSE, $course->id))
      {
3   print_error('nocontext');
4 }
5 ...
6 if ($switchrole > 0 && confirm_sesskey() &&
7   has_capability('moodle/role:switchroles', $context)) {
8 ...
```

Code Snippet 9.3: *A snippet from /course/view.php.*

The function `get_context_instance` requires a constant and in this case a course id. The constant used for the function is a constant telling which context level that needs to be loaded. This is a numeric value. In this example an error

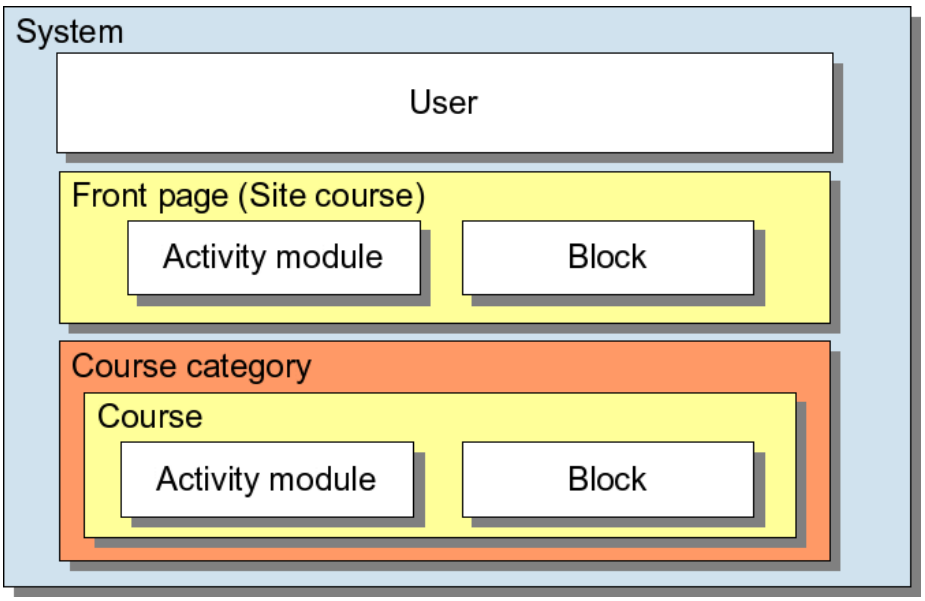


Figure 9.2: *The hierarchical context structure in Moodle.*

is outputted if the context fails to load. To determine whether or not a user has the capability to do certain actions the function `has_capability` is used [71]. It requires a string specifying which capability and the context for the current page. It is required to set the context for each page. In some pages Moodle sets the context itself [70].

Beside being responsible for capabilities contexts are used to determine the presented blocks on a page. If a user with the capability to edit a course page adds a block and the instance is set to the system context the block will be showed on all course pages. If the context is set to the course context the block will only be presented on that courses page.

9.5.3 Capabilities

Moodle integrates a capability system too manage permissions throughout the site [71]. Every capability is connected to the system context, which is global for the entire site. Capabilities can be changed in inherited contexts. For example a user who does not have a permission in the system context to create courses can have the permission to do so in a certain course category.

Creating new capabilities is possible when creating a new plugin. To check capabilities the function `has_capability`, which takes a capability, given as a string, a context instance and extra parameters depending on the capability and the context.

Chapter 10

Analysis & Design

In this chapter we analyze the requirements and design our sub-system for implementation. We split our sub-system into two sections, which are analyzed and designed separately in Section 10.1 and Section 10.2. The complete architecture of MyMoodle is presented in Section 10.3. This chapter makes a foundation for our sub-system and its implementation.

The requirements from Section 8.2 are used in this chapter to analyze and design our system. We divide our system into two sections. One section being the usage of project groups, we call this part "Virtual Meeting Place". The other being the administration of project groups, called "Project Group Management". This division is corresponding to our division of end users in Section 8.1. The analysis and design of each to part is presented in Section 10.1 and Section 10.2 respectively. Analysis and design is presented in succession, to be more comprehensible.

Jeg har ændret part til section. ændre tilbage hvis det ikke fungere. kim

10.1 Virtual Meeting Place

In this section we analyze the requirements related to the part of our subsystem called Virtual Meeting Place. The primary requirement analyzed is the one named the same as this section; Virtual Meeting Place. Related requirements are: Navigate to Project Groups and Project Group Members. These are analyzed in Section 10.1.5 and Section 10.1.6.

A virtual meeting place can take many forms. As our requirement of a virtual meeting place states, we want it to comply with the Aalborg PBL model. We will analyze our requirements based on our own experience, and the interviews and demo meetings conducted.

The aspects of the requirement Virtual Meeting Place that we have discovered and are presenting here are: Structure of virtual meeting place, assignment of virtual meeting place, division of projects and groups, and interface of virtual project tools. This are presented below.

10.1.1 Structure of Virtual Meeting Place

The members of a project group should have a place where they can meet and engage in project related virtual project tools. Recall from Section 8.3 that our responsibility is to construct a place where this can happen, not construct the

actual virtual project tools – that is the responsibility of our peer-groups. We have to decide the structure of the virtual meeting place. The two structures considered are described below.

Shared Group One idea is to have every project group share everything with other project groups in a semester. This corresponds to having every project group working in the same room in the real world.

Team Group Another idea is to give a virtual group room to every project group and ensuring that only the members of the project group and the supervisors can contribute to the work on the project. The corresponding situation in the real world is that every project group has their own group room where only they (and their supervisors) can do work on the project.

The Shared Group idea require less effort than Team Group to implement since no permissions need to be considered. However, the Team Group is closer to the way that Aalborg PBL is implemented.

We choose the Team Group structure. It is infeasible to have every project group share everything. A project group member could be forced to look through many functionalities to find the one relevant to the given project group. Furthermore, by allowing each project group to have their own virtual meeting place, the members can customize the place as they see fit by removing irrelevant functionalities and possibly adding new relevant functionalities. The virtual place where these functionalities can be found is called the "project group room". A virtual meeting place for a project group thereby consists of the project group room and the functionalities available to the project group from their project group room.

The project group room should have some of the same functionality as a physical group room. We as the Project Group Management implement the virtual group room, and the other three peer-groups implement the different functionalities.

10.1.2 Assignment of Virtual Meeting Place

At Aalborg University group rooms are assigned to project groups periodically. The period at which assignment of group rooms occur can differ. Some of the most common periods are presented here:

- **No Group Rooms** There are no group rooms available because the field of study does not follow the Aalborg PBL model or the students are studying from remote locations.
- **Daily** The students have to reserve the group rooms on a daily basis (described in Appendix A.3).
- **Half-yearly** Group rooms are assigned to each project group at the start of a semester.

The virtual meeting place should be a hub where tools central to group work are available. It should be similar to the Aalborg group room concept and be available on a half-yearly basis.

10.1.3 Division of Projects and Groups

Project and Groups can be regarded as two different entities. The question arises if project groups should be designed as groups that have a relation to a project or as an single entities where the group and project are one. Below is the properties of the two variants presented.

Groups and Projects Divided We now consider the properties of groups and projects as two different entities:

- Flexibility between projects and groups, allowing many-to-many relationship.
- Complex to implement, by comparison.

Groups and Projects United We now consider the properties of groups and projects as one single entity:

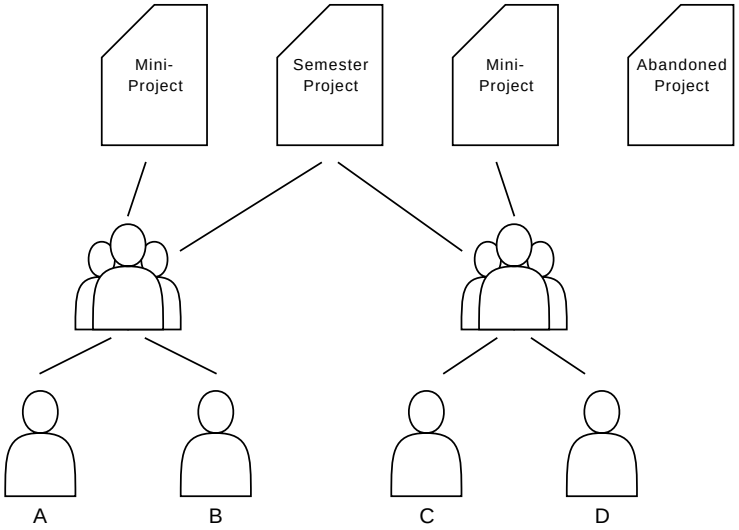
- A group is linked to a single project, they are inseparable.
- Simple to implement, by comparison.

The differences between these variants are illustrated with an example in Figure 10.1. The example is as follows: There are four students identified by the letter A to D, and four projects; two mini projects, one semester project, and a single abandoned project. Students A, B, C, and D collaborate on a semester project. Students A and B are working on a mini project. Students C and D are working on the other mini project. Finally no one is working on the abandoned project.

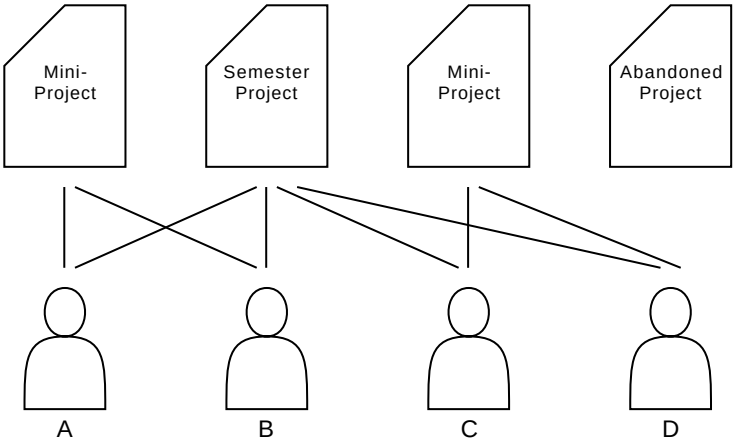
Intuitively the option to divide groups and projects (seen in Figure 10.1a) might seem more appropriate since it introduces a level of indirection. However our interview with Lene W. Even (seen in Appendix A.1) shows that the administrative personnel does not see any benefit from having any extra levels of indirection between students and projects. Since we are working agile we prefer to take the simplest viable approach and allow for extension of the system later. We choose to design groups and projects as one (seen in Figure 10.1b) in accordance with our end users' needs and to simplify our system. The database design for keeping project groups and members of the project group is found in Section 10.4.

10.1.4 Interface of Virtual project tool

As mentioned previously we are developing a virtual group room where different virtual project tools can take place. The virtual project tools are primarily developed by our peer-groups. For the virtual project tools to be available to the users through the virtual group room we have to have a common interface over which the virtual group room and the virtual project tools can communicate. A typical communication message from the virtual group room to the activity could be a request to make the activity render itself. Alternatively we could integrate every activity directly into the virtual group room. This, however, is neither flexible nor modular enough. Recall that we are four peer-groups of three to four students working together, which means that we would likely "step



(a) *Divided.*



(b) *United.*

Figure 10.1: *The difference between the two approaches to the division of project and group illustrated by an example.*

on each others toes" if we all were to work on the same component at the same time. Furthermore, there is a group of students next year that must take over this project. They might want to change the functionality or disable an activity. This should be much easier if each activity is encapsulated in a component with a common interface than if every activity was mingled together into a single component.

Now that we have established that a there is a need for an interface between virtual project tools and the virtual group room, we want to examine the possibilities that we have for doing so. The two broad categories that the final choice can fall in are: Use an existing interface or develop a new interface. These are not mutually exclusive. E.g. we may take an existing interface and modify it to fit our needs. Since we are making a plugin to a well established LMS, we examine the possibilities in it to find an existing interface. There are two plugin types in Moodle that we consider to be used as interface. The systems are: Moodle blocks and activity modules. Both of these are described in Section 9.4. By choosing any of these as the interface, we restrict our peer-groups to make some or all of their sub-system as the given plugin type, hence this has been a joint decision.

Intuitively the activity module might seem like the right choice, after all the name is *activity* modules. However, activity modules are not for general virtual project tools, but only for course virtual project tools. In fact an activity module is required to have a database relation containing the instances of the activity module, each of which must have a link to a course [25]. We abandon this idea because we do not want to have links between our project groups and courses – a project group is an entity in its own right.

We now consider whether to use Moodle blocks or develop a new interface. Developing a new interface would give us more flexibility because we may choose exactly how it is defined. On the other hand a Moodle block ensures that MyMoodle will have a similar look and feel as the rest of Moodle. Additionally, the students that take over next year (and possibly third parties later) will have accessible documentation by through Moodle-dev to modify virtual project tools or add new ones. The advantages of Moodle blocks outweighs the additional flexibility that we may gain from defining our own interface. This leads us to choose to use Moodle blocks plugin type that every activity must be implemented as. For the virtual group room to communicate with virtual project tools it can use calls to members methods of Moodle blocks. E.g. to render a Moodle block a call to `get_content` is used [20]. Moodle, however, provides functionality to render all blocks associated with a given page.

10.1.5 Navigate to Project Groups

For a user to navigate to a project group that he is a member of, there should be procedure like the one for navigating to courses in which the user is enrolled. To accomplish this we have looked at how navigation to courses work in Moodle. There is currently a problem in Moodle when navigating to courses. When a user is enrolled in a large number of courses his entire front page is filled with links to those. There is no built-in functionality to move or sort the links. ELSA mentioned this problem during the meeting that we conducted with them described in Section 3.1.1.

We want to avoid this problem when we design the navigation for project groups. Moodle has a navigation block with a list of important links. We want to add an item to this list that, when expanded, shows the project groups the user is a member of. Since a supervisor or an administrator might be a member of a many project groups, we want to limit the size of the list of project groups. We do this by showing a maximum number of project groups according to a preset value. If the user is a member of more groups we show a link to a page that has a list of all the users's project group.

10.1.6 Project Group Members

Recall that this requirement must ensure nearness for project group members. If a project group tend to work in the same physical room, they have nearness. To fulfill this requirement students conducting group work from remote sites should be able to experience nearness through our system. Supervisor Communication has requested a feature for supervisors to see the members of a project group to help them identify a particular group. We choose to design this feature to accommodate the requirement for project group members. A part of the project group room should show all the members of the project group belonging to the room. There should be a picture of the members to help identify them. There is to be a functionality to hide or remove the members since students working in a physical group room might not need this and therefore prefer to focus on the other virtual project tools available in the virtual group room.

10.1.7 Combining the Aspects

The virtual meeting place consists of many things. First and foremost there is a project group room where virtual project tools can be used directly or alternatively linked to. Secondly there is all the virtual project tools that are not located directly in the project group room. Only students and supervisor(s) associated with a project group are allowed to access the project group room of the given project group. An exception to this is the administrative personnel, who have privileges to access and change every project group room.

Users are linked directly to project groups. This reduces the complexity of the system at the cost of flexibility.

Each activity related to a virtual group room is implemented as Moodle block. This allows for a well defined interface which a virtual group room can use to communicate with the virtual project tools related to it.

To ensure familiar navigation for a user to his project groups, we extend the existing navigation bar of Moodle. To avoid expanding the navigation bar out of proportions we limit the number of project groups that will be shown and provide an additional page where every project group belonging to a user can be found.

To provide a sense of nearness and to help supervisors recognize project groups, a simply activity is added to every virtual group room. It can be hidden by users, if they do not need it.

10.2 Project Group Management

During the meetings with ELSA, see Section 3.1.1, and the administrative personnel we discovered three different approaches to administrate the project groups: Automatic management through external application, manual management by administrative personnel, and administration by students.

10.2.1 Automatic Management

In this section an automatic group management approach is explored.

Courses are created automatically on Moodle by ELSA. The courses are not populated automatically, but it still eases the work load for the administrative personnel. A similar approach can be used to create groups from an external database containing the student groups.

With automatic management through an external system it is necessary to synchronize Moodle with the external system in the case that the groups are changed. If the data is inputted only in the external system a one-directional synchronization is needed, but if groups can be created, added, and deleted in Moodle a bidirectional synchronization service must be created. This service must be able to handle conflicts if changes are done at the external application and Moodle at the same time.

After an interview with Michael Møller Hansen of IST (see Appendix A.4) we discovered that AAU does not have a single system containing the groups at the University. Several systems exist and some of the systems are not online. Each of these systems contain a fragment of the entire set of project groups. Furthermore, some departments does not keep official records of the groups (see Appendix A.4). Morten Mathiasen Andersen explained in an interview that the systems do not have an API from which the groups can be extracted and he himself is working on an API for one of the systems.

10.2.2 Student Management

In this section an approach which lets students create, edit, and delete groups is discussed.

In some cases the administrative personnel do not know which student groups are created (see Appendix A.4), and it is not guaranteed that all students want an online project group in Moodle. Therefore, a solution is to allow students to administrate, allowing the students to create and edit their project groups as needed without adding extra workload to the administrative personnel.

The problem of this approach is that students might create badly named groups or misuse the system by e.g. destroying groups for other students. This approach must include various security constraints to ensure proper use. Students should not be able to delete groups they are not members of or to delete groups by mistake. A mistaken deletion of a group late in the semester could lead to a large amount of lost information and work. The creator of the project group must be able to govern who can be a member or the members of the group must accept new members. Alternatively they should not be able to add or remove members from a group after creation since a student should not be able to join a group they do not belong to.

10.2.3 Administrative Personnel Management

In this section the approach of having the administrative personnel administrate project groups is discussed.

The administrative personnel plays a central role in the administration of the courses and student groups and is thereby capable of adding the necessary project groups to Moodle. Since the administrative personnel is trained professionals this approach ensures that only the needed project groups are created and that they are maintained properly. A drawback of this approach is that it adds more work to the administrative personnel.

10.2.4 Choosing an Approach

The automatic management approach using an external system can be ruled out due to the fact that there does not exist a central student group database at AAU and the decentralized system, AdmDB, in the process of getting a new API developed. Thus are the current API gonna be deprecated in a near future. If a centralized system with an API to extract group information existed this would be an the ideal approach, since it will not add workload to the administrative personnel and all groups will be able to have a project group in Moodle. From the two remaining approaches we choose to have the administrative personnel administrate the project groups. We choose this because the student administration approach has several constraints which must be overcome. With the student approach the administrative personnel must be able to clean up the project groups and therefore an administration panel for the administrative personnel is necessary. Without having to spend time figuring out how to constrain the system to prevent misuse we can implement a working system by choosing the administrative personnel management approach. The two approaches do not exclude each other and the functionality used in the chosen approach can be reused if the other approach is implemented later.

With the chosen approach administrators need to be able to add, remove, and edit project groups. Per default Moodle has a Moodle block called Settings. If a user is an administrator there is a list in this block called site administration. This list contains a lot of tools administrators use, and we choose to add project group administration to this list.

We want a page where administrators can add a new project group that also adheres to Moodle standards. It should be possible to name a group, add the desired members, and choose who the supervisors of the project groups are.

Additionally there needs to be a list of all project groups. From this list it should be possible to add, edit, and delete groups. This list can potentiality grow very large, so there needs a way of finding a specific group or a specific set of groups as specified by requirement Find Project Group. This requirement is fulfilled by developing a filter system where an administrative user can add filters on attributes to find a specific project group or a set of related project groups. The filter attributes should be those that administrative users are likely to use. During the demo meeting described in Appendix A.6 we experienced that our end user need to find project group based on its name. This has lead us to decide that the project group name should be a filter attribute.

The final requirement to consider is Introduce Roles. An administrative user must be able to set roles of the members of a project group. We choose to allow

an administrative user to change the role of a project group member in the page where he can add and remove members. The roles that we consider to be important in a project group are student and supervisor. An option we consider is to allow administrative personnel to create roles themselves, to adapt to a given situation. However, we have not found any need for this, hence we choose only to have the two roles student and supervisor.

10.3 Architecture

MyMoodle is an extension of Moodle and can be seen as a package of plug-ins. The architecture does not specify how each plugins should be created, but specify a general structuring of the components of the system. The complete architecture can be seen in Figure 10.2.

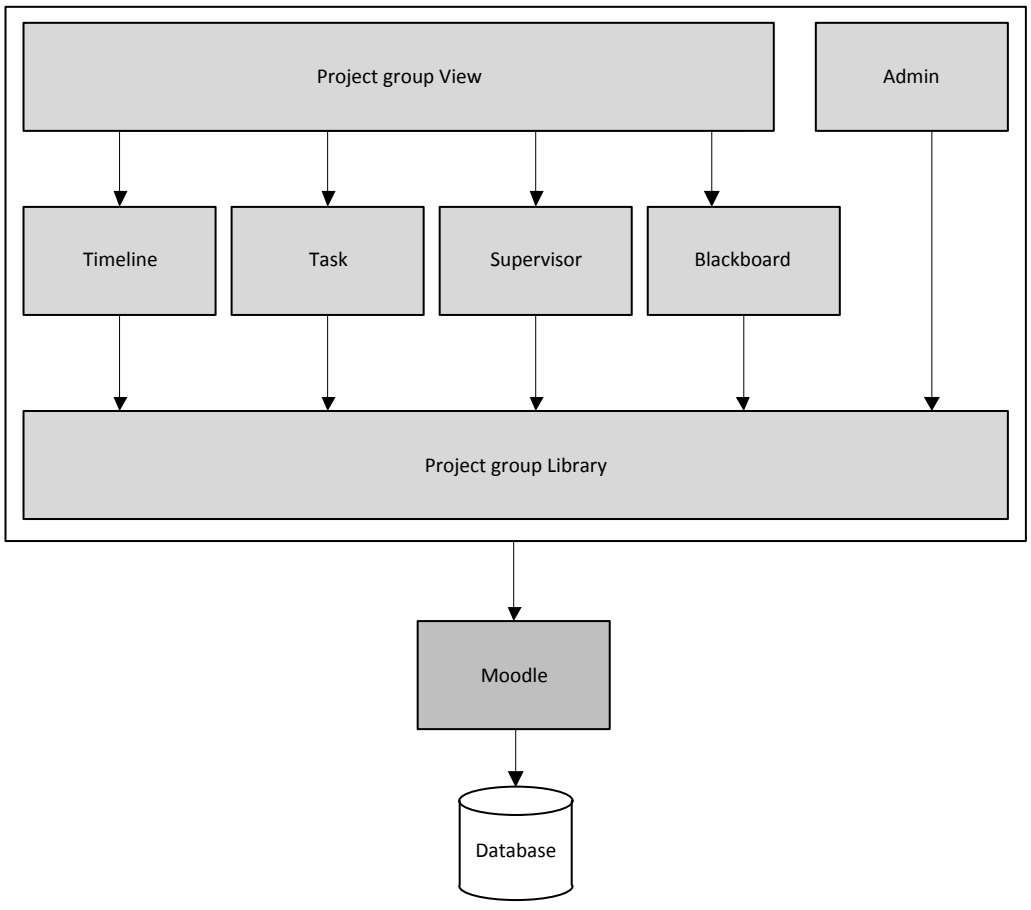


Figure 10.2: *The overall architecture of MyMoodle.*

The system architecture consists of a total of five layers. The three up-

permost constitute MyMoodle, they have a common dependency, namely the Moodle platform. Layers four and five are Moodle and the Database respectively.

We will now explain the three uppermost layers:

1. The upper most layer is the project group view and the administration tool. The project group view is used for presenting virtual meeting place for a project group. It is described in Section 10.1. The administrative tool is a tool used by administrative personnel for managing project groups. It is described in Section 10.2. This layer is called "View Layer".

2. Directly below the upper most layer is the "middle" layer which consist of the four parts: Timeline, Task, Supervisor, and Blackboard. These four parts are created by our peer-groups and are not explained further. We call this layer "Content Layer" since the components in this layer generate content for the Project Group View component.

3. Below the middle layer is the project group library which contains common functionality. This layer handles all intercommunication between the parts in the Content Layer.

There are two primary factors to be considered when planning an architecture. First, we are four sub-groups working together. This creates the need for a structured way of communicating between the different parts and it lets every sub-group know how their part is connected to the rest of the project. Second, the project should be passed on, which require an architecture that increases the comprehensibility of the project. It is not possible to make a strict layered architecture due to the Moodle dependency and the administrative tool which does not have to use the intermediate layer, but depends directly on the project group library. We do, however, prohibit ourselves from accessing the database directly, by using the Moodle Database Layer (note that it is not a layer in our architecture) described in Section 9.5.1. In Figure 10.2 the dependency from the box encircling MyMoodle indicates that every component in MyMoodle depends on the Moodle component. The Moodle component consists of the Database Layer as well as the Context System, Capabilities, etc.

Note that the components relative size does not imply anything, e.g. the Moodle component is much larger, code wise, than any of the components in MyMoodle, but is illustrated with a rectangle the same size as any of the Content Layer components.

10.4 Database

In this section we will present the design of our database. Our design will support the concepts that we have introduced, namely project groups and project group members.

Recall that we are designing to comply with a relational database, more specifically an SQL database as described in Section 9.1.

10.4.1 Project Group Table

A project group is an entity that consists of different elements. How these elements are represented in the database is described here. One core element

is the project group members, this is described in Section 10.4.2. The other elements will be represented as fields in the `projectgroup` relation.

A project group must be identifiable by both human and computers. To make a project group identifiable by humans we give it a short name that must be unique among other project groups. The field that holds the short name in the project group relation is called **shortname**. Initially it may seem that the short name could be used as a general identifier (both for humans and computers) since a short name is unique. However, this name might be changed during the lifespan of a project group. We much rather prefer an identifier that is constant for a project group to a avoid race conditions. An example of a race condition in this system could be a user trying to access a project group with an identifier (the short name in this case) while another user is changing the short name of the group. Should the short name be changed before the first user is trying to access the project group. The request will fail because the identifier that he is holding is deprecated.

We choose to have a numeric auto-incrementing **id** field to identify the project groups. This cannot be set or changed by the users of the system. It is a Moodle guideline to have an auto-incrementing field as primary key [65]. We clearly obey this guideline by making **id** the primary key of `projectgroup`.

One could argue that the numeric id field could be used as human identifier as well. However, we deem it much easier for a humans to remember and associate an alphanumeric name with a particular project group than an arbitrary number. In summary a project group has two identifiers; the numeric **id** and the alphanumeric **shortname**. The former being the primary key of the relation.

To give a better description of a project group, we have an optional field called **longname**. This field does not need to be unique, but only serves as a description for a project group.

To allow project groups to be ordered or filtered based on which semester (or another time-based constraint) they belong to we choose to have a timestamp field called **created**. The value of this field is the Unix-timestamp at the time of the creation of the project group.

The four fields of `projectgroup` are as follows.

- **id** to identify the group.
- **shortname** to easily identify the group for humans.
- **longname** of the group, for a more descriptive name.
- **created** timestamp for a given project group.

An instance of `projectgroup` with a single tuple (representing our own group) is shown Figure 10.3. Notice in the figure that **shortname** might seem somewhat cryptic. It translates to: "Software Engineering, 6^{th} semester, group number 8, spring (forår), 2012". This illustrates the need for **longname**, since supervisors might find difficulty distinguishing several groups of the same semester simply by their group number. **longname** gives a more descriptive name that should clearly give an idea of which it project group is.

10.4.2 Project Group Members Table

As opposed to a project group a project group member is not an entity in its own right. A project group member is a user that is a member of a project

id	shortname	longname	created
1	SW608F12	Project Group Team working on MyMoodle	1337723013

Figure 10.3: *Example of an instance of* `projectgroup`.

group. Moodle saves users in a relation named `user`. This relation contains information about the users of the system – a unique identifier in particular – and is part of the Moodle core database schema [**?**]. We want project group members to be represented in the database as a link between `projectgroup` and `user`. We want this link to be many-to-many, since a project group can contain many members and a user can be a member of many project groups.

We can save the link in either `user` or `projectgroup`, or create a new relation to save them in. We state in our system definition (seen in Section 4) that current functionality of Moodle should not be altered. The first alternative is therefore out of the question since we cannot guarantee that the other parts of Moodle will continue to work as before if we alter in the core database relations.

Should we choose the second alternative we would make n fields in `projectgroup`, each containing a member of the project group or a value indicating no members, e.g. as null. This would set a maximum of n members on every project group. If there is a maximum number of members that will ever be in a project group and this number is acceptably low this alternative might be viable. The number of users in the system could be a candidate for a maximum value. This number may, however, change as more users are added to the system. Furthermore, this number is likely to be larger than the actual maximum number of members in any group, assuming that no or few projects are conducted by every user of a Moodle system. Another option is to choose a value that we believe is high enough for most purposes. This would result in a many-to-n link between project groups and users, in that a project group can have n members and a user can be a member of any number of project groups. If this n value is chosen to high, space will be wasted, e.g. if n is set to 20 and no project group ever has more than 10 members, at least 10 fields in each `projectrgoup` tuple will be wasted. If n is too low the system will be unacceptable because some desired project groups cannot be created.

The final alternative requires a new relation which stores a combination of users and project groups. This allows for a true many-to-many link between users and project groups. By choosing this alternative we have great extensibility, e.g. we are able to save the role of project group member in this relation as well. A weakness to choosing this alternative is greater complexity by an increase in the number of relations. The strengths of this alternative outweighs its weaknesses and it is better than the previous alternative by allowing a true many-to-many link without wasting space or making the system unacceptable. Hence we choose this alternative as the way to save project group members in the database. The relation that is used to link `user` and `projectgroup` is called `projectgroup_members`.

For `projectgroup_members` to link `user` and `projectgroup` it will have a foreign key to both of these relations' primary keys. The fields that hold these foreign keys are called **user** and **projectgroup** respectively. The combination of **user** and **projectgroup** is a candidate key for `projectgroup_members`.

However, as with `projectgroup` we choose to have a numeric auto-incrementing single field called **id** to comply with Moodle guidelines [65]. We still make the combination of **user** and **projectgroup** unique, to ensure that no person is a member of the same group more than once.

Whenever a user is a member of a project group we want to have a role associated to that member. We can choose to give every user a role and whenever that user is a member of a project group he can act and have privileges as fit for the role. Another way to do it is to save the role along with the link between users and project groups. We prefer the latter for two reasons. Firstly a user may have different roles in different project groups, e.g. a Ph.D. student has a student role in his Ph.D. project, but may also be supervising projects being conducted by younger students. Secondly we would have to make a new relation to contain only this information and thus increase complexity if the former alternative is to be chosen. We will not alter the core relations such as `user`, as described in Section 10.4.1.

To allow for extensibility we save the creation and update time of each tuple in `projectgroup_members` in **created** and **updated** respectively. It may be useful to have these timestamps for administrative personnel, e.g. to see whether or not a user has joined a project group later than the other members.

The final set of fields in `projectgroup_members` are:

- **id** which is Moodle convention. [65].
- **projectgroup** references to the related project group.
- **user** references to the related user.
- **role** denoting the type of membership the user has to the group.
- **created** is the timestamp for its creation.
- **updated** is the timestamp for the last update to the membership.

Figure 10.4 shows an instance of `projectgroup_members` containing four students (indicated by role "0") and a supervisor (indicated by role "1"). Notice that all tuples have the same **projectgroup** value. This indicates that every tuple links user to the same project group, namely the one shown in Figure 10.3.

id	projectgroup	user	role	created	updated
1	1	1	0	1337723013	1337723013
2	1	2	0	1337723013	1337723013
3	1	3	0	1337723013	1337723013
4	1	4	0	1337723013	1337723013
5	1	17	1	1337723184	1337723184

Figure 10.4: *Example of an instance of* `projectgroup_members`.

The database scheme can be seen in Figure 10.5. The user entity is a built-in part of Moodle and is not modeled by us.

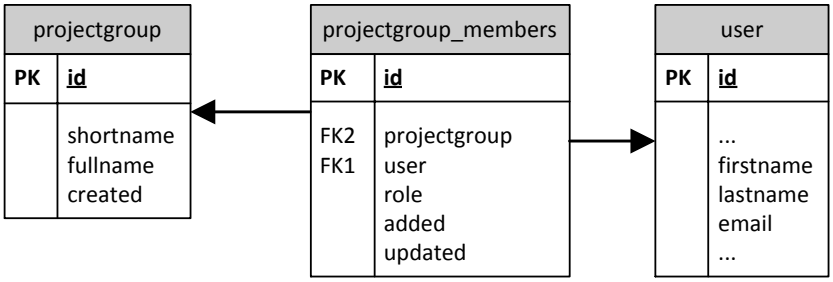


Figure 10.5: *The database scheme of project groups and memberships. Most of the data fields of the user table are omitted for brevity. The user table consist of more than 50 fields.*

Chapter 11

System Presentation

The chapter presents how the system works from the perspective of the user, which is key knowledge in order to comprehend the Implementation. The chapter is divided in two sections, one presents the project group view and the other presents the administrative part.

11.1 Project Group Room

A screenshot of a project group room can be seen in Figure 11.1. In this project group room the blocks of the standard layout are seen.

The project group room consist of three rows. The screenshot illustrates the area boundaries by dashed lines. The three rows on the page are: The header, the middle, and the footer. The header is standard Moodle and we affect it by adding a headline the rest is added automatically. The heading shows the name of the project group. The middle part is the actual project group room and it is divided in three columns. The left column is the standard navigation menu in Moodle and is not created by us, though it is extended by us. The center and right columns both contain Moodle blocks. The center area is much wider than the left and right and can therefore contain bigger Moodle blocks. The various Moodle blocks presented on the project groups page are described in Section 12.2.1. If a user wants to edit the layout for the project group room he can press the "Turn editing on" button. This will allow the user to remove, add, move, and edit Moodle blocks. A special "add new block"-block is added in editing mode to allow for adding new Moodle blocks. If a user edits the page, the change can be seen by all group members.

The figure an example of a Moodle block can be seen, it is denoted with the grayed out area. The grayed out area is the members block and is created by our group, we will elaborate on the implementation of this in Section 12.2.1. The block shows a photo and the name of each member of the group.

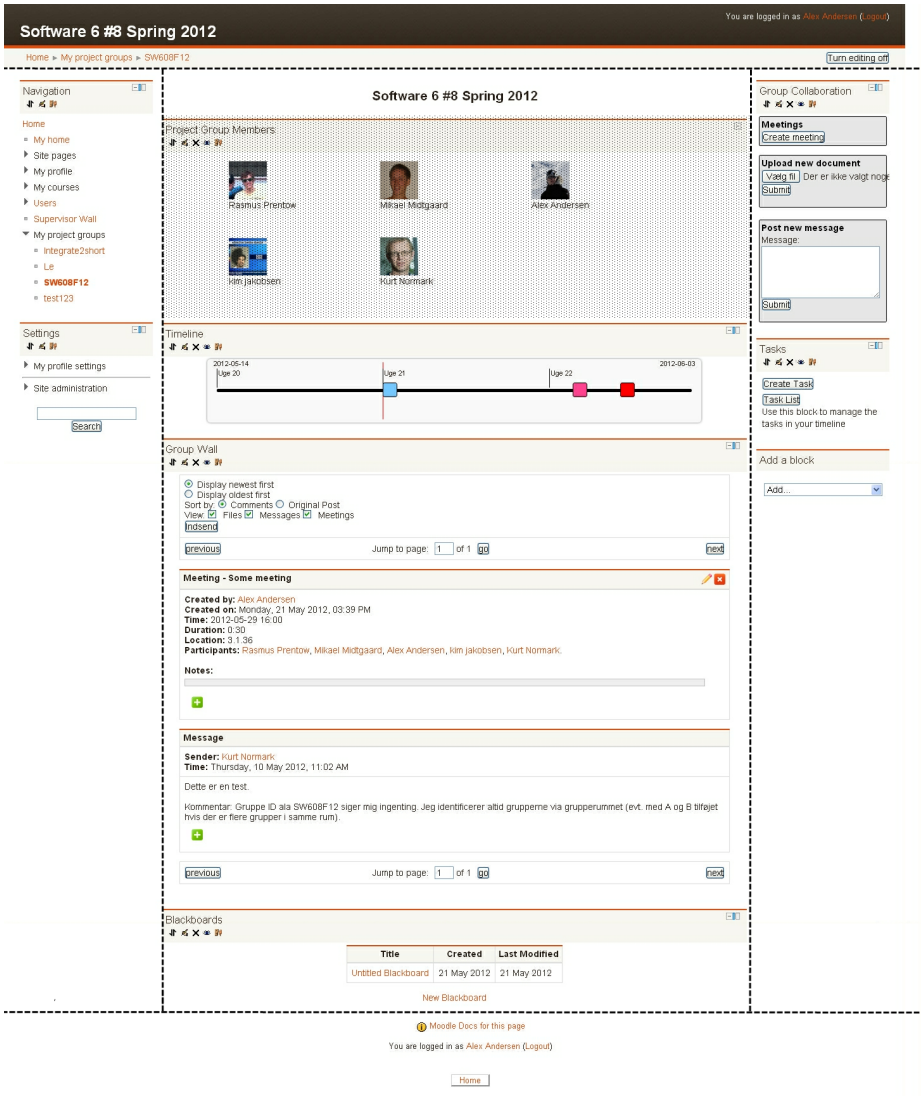


Figure 11.1: *The project group room. The greyed out area marks one of the block and the dashed lines shows the different areas on the page.*

11.2 Administration

This section presents the system as seen and used by the administrative personnel.

The administration tool can be accessed through the site administration menu as seen in Figure 11.2.

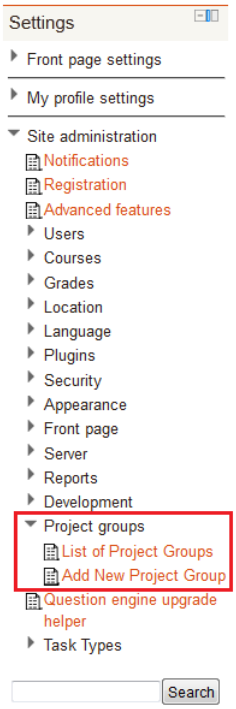


Figure 11.2: *The settings block, which contains the site administration menu.*

The menu provides two links: One link to a list of all project groups, which are further described in Section 11.2.1, and one link to a page where project groups can be edited. This page is described in Section 11.2.3.

11.2.1 List of Project Groups

The list of project groups, seen on Figure 11.3 provides an overview of all project groups. It contains a filter which can be used to reduce the amount of presented project groups and thereby making it easier to find one or a set of specific project groups. The page contains a link to a page where project groups can be edited. This page is further described in Section 11.2.3. For each project group there is a row in the list contain the following: The short name; which also works as a link to the page described in Section 11.2.2, the long name of a project group, an edit button, and a delete button. The edit button takes the user to the same page as clicking on the “add new project group”-link. The only difference is that the project group id is sent along, which forces the add & edit page into edit mode.

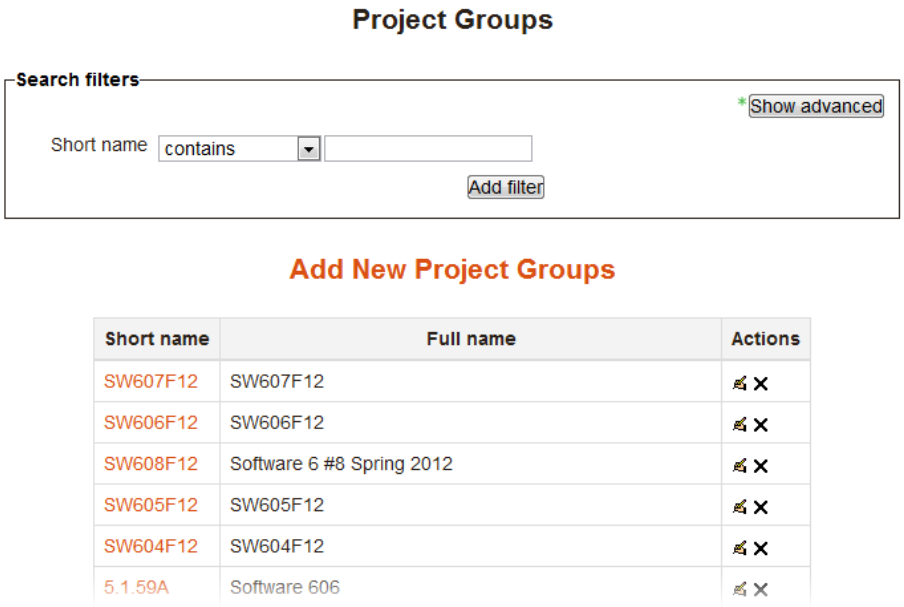


Figure 11.3: *A screen shot of a page listing all proejct groups in the system.*

11.2.2 Project Group Members

The project group members page, seen in Figure 11.4, contains a list of the members of the group. The page contains a link to the project group room (described in Section 11.1) and a link to the add & edit page. For each member there is a delete button, which removes the member from the group after a confirmation from the user.

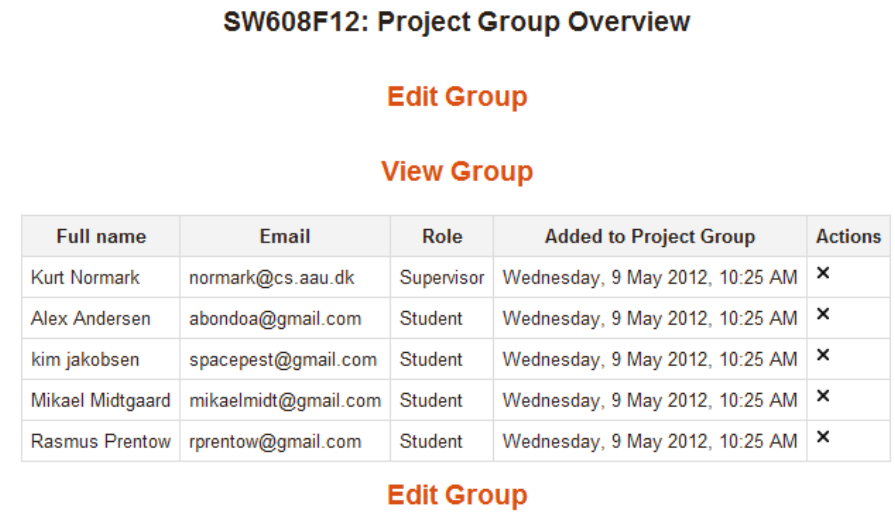


Figure 11.4: *A screen shot of the project group members page, which shows all current members of the project group.*

11.2.3 Add & Edit

The add & edit page serves two purposes; it allows editing of a project group and it allows the user to add a project group. The page is divided in four parts and can be seen on Figure 11.5. The first part contains fields allowing the user to enter basic info of the project group which is either edited or added. Beneath this there is a filter, which is based on the same as the filter used in the list of project groups (see Section 11.2.1). When a filter is applied the list of available users in the third part of the page is reduced according to the applied filter. The last part shows all added members and the user can highlight members to mark them as supervisors.

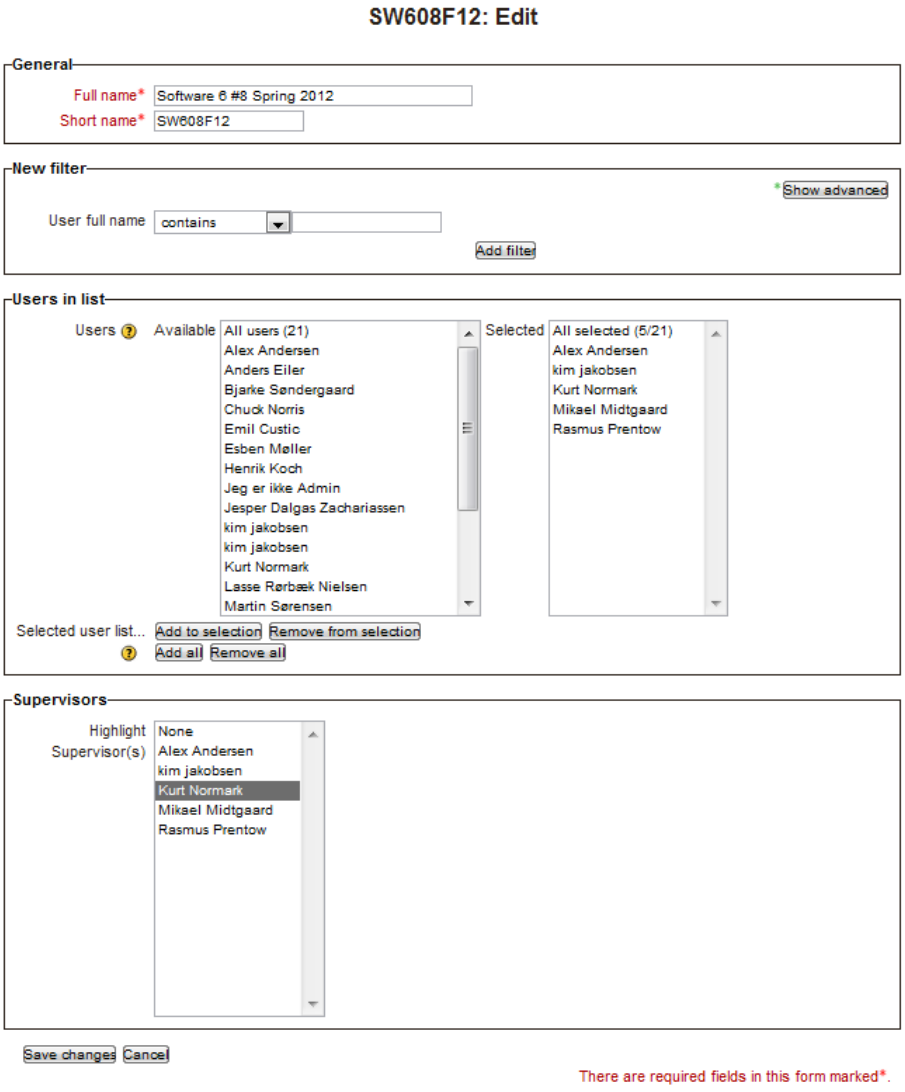


Figure 11.5: *A screen shot of the add & edit page.*

Chapter 12

Implementation

In this chapter we describe how we implement the concepts described in Section 10. This chapter is divided into three sections where the three main parts of our system are described, namely Project Group Library, Project Group View, and Managing Project Groups. The implementation details presented here are paramount for Chapter 13 and Chapter 15.

From section Section 10.3 we know there is three part(described as layers) which must be implemented by us. The administration tool to manage project group, the project group, and the project group view. The administration tool is implemented as an admin tool, which, as described in Section 9.4, is a special Moodle plugin type.

Recall Section 10.1.4 where we designed the concept of project groups. In the implementation of the project group concept we are faced with two choices. It can be implemented as courses (see Section 9.2) by creating a new view for the course page. Hereby solving the problem of both implementing project groups and the aforementioned project group view. This will make it possible to use activity modules, described in Section 9.4, in the project group. Another approach is to make a local plugin, which, as described in Section 9.4, gives us basic functionality, such as database installation and the possibility to extend the inbuilt navigationblock. With a local plugin we cannot use activity modules since they are too strongly connected to courses. Instead we can use Moodle blocks for the functionality. The later approach is chosen.

The local plugin includes both the project group library and the project group view.

12.1 Project Group Library

The project group library is implemented in the file /local/projectgroup/lib.php. The library includes several other files. One for each sub group, excluding ours, and several helper files. Our contribution to the library is located in the library file itself. The library consist of $\approx$ 300 lines of code, including comments and not including any included files. All the functionality in the library is helping functionality for the entire system.

The following sections explains how project groups are put into a moodle context and how we ensure that users has the correct permissions.

12.1.1 Context of Project Groups

In Section 9.5.2 the context system of Moodle is described. In this section the creation of a custom context is described.

To be able to define capabilities for the project groups and have different blocks for different project group we need our own context. We will create our own context level and class. We call the context **context_projectgroup**. The class is located in /local/projectgroup/context.php and is included by the project group library.

Moodle does not support extension of contexts through any of the 30 different plugins types available [27]. This fact poses a problem for us. There are two parts of the problem. The first is to create a new context and the second is to load it properly. We create a new context by making a new class which is very similar to **context_course** class and by defining the context level of project groups as a constant. The class header and the constant definition can be seen in Code Snippet 12.1. The constant is set to 55, as seen in line 1. It is chosen because that this context level is unused.

```
1 define ('CONTEXT_PROJECTGROUP',55);
2 class context_projectgroup extends context {
3 ...
```

Code Snippet 12.1: *The context_projectgroup class header and constant definition.*

When a context is loaded in a Moodle page it is instantiated by `get_context_instance`, which takes a context level and an instance id. The instance id can be a course id or similar depending on the context level – in our case this would be a project group id. This function calls a static method in the **context_helper** class which uses a private array to translate the context level into a class name. As stated in Chapter 4 we refrain our self from changing the core code of Moodle. If this constraint were not enforced the array could simply be extended directly in the code. Since the array used is private we can not extend the context system by overriding the **context_helper** class. The newly created context is only used in pages created in this sub-project and we can therefore create our own version of `get_context_instance`. The new function can be seen in Code Snippet 12.2.

```
1 function get_projectgroup_context_instance( $instance = 0,
      $strictness = IGNORE_MISSING)
2 {
3     return context_projectgroup::instance($instance, $strictness);
4 }
```

Code Snippet 12.2: *The function to get projectgroup context.*

The **instance** variable denotes a project group id. The **strictness** variable indicate its compatibility mode. There are three constants defined in Moodle DML that are supposed to be used as input for **strictness**. These are: **IGNORE_MISSING**, **IGNORE_MULTIPLE**, and **MUST_EXIST**. The value of **strictness** is passed on in the static method `instance` of **con-**

text_projectgroup to the Moodle database API call to `get_record` which implements functionality to throw exceptions depending on **strictness**.

With the new context and the function to instantiate it we can now make different permissions per project group and add blocks to specific project group rooms.

12.1.2 Ensuring Permissions

Permissions can generally be divided in two types; read and write. Read permissions gives you the ability to view the content of the project group room while write lets you change the content. If a user has write permission he must also have read permissions. Otherwise he cannot see the page he attempts to edit. To ensure the user attempting to enter a project group view has permission to enter the function `has_projectgroup_read_permission` is used. It checks if the user is an administrator or is a member of the group. The administrator check is necessary since administrators should be able to see the group even if they are not members of the group.

The function `has_projectgroup_write_permission` checks that the user has write permissions and uses the read permissions function to check that the user can read. If he cannot read he should not be able to edit. In the current implementation the write permissions function does not make extra checks to permissions since the permission levels for read and write are equivalent. Making both function gives the ability to later change this. An example could be if the potential users requires that their supervisor should only have read permissions. Then the change will be in one place only.

Both functions can be found in the project group library.

12.2 Project Group View

In this section the actual implementation of the project group room, its custom blocks, how the context system is used to implement our own context to allow better administration of blocks and capabilities, and the navigation to the users project groups is described.

12.2.1 Blocks

When a new project group is created a number of default blocks are added to the page. The default blocks are specified in a configuration file, /local/projectgroup/config.php, and the content can be seen in Code Snippet 12.3

The syntax for the format is $left : middle : right$. Left, middle, and right represent the three columns in the project group room. The blocks: Timeline, groupwall, blackboard, upload, and tasks is created by our peer-groups while the block named projectgroup_members is created by us.

As described in Section 8.2 the projectgroup_members block shows the names and pictures of the project group members. It is implemented as a normal Moodle block in the same manner as the Moodle blocks created by our peer-groups, though our Moodle block is less complicated. All it does is fetching the members of the project group using the function `get_projectgroup_members`

```
1 <?php
2     /**
3     * Example usage:
4     * "left1,left2:center1:right1"
5     * Will add two items to the left, one in the middle, and one to
          the right
6     */
7     $format['defaultprojectgroupblocks'] = ':projectgroup_members,
          timeline,groupwall,blackboard:upload,tasks';
```

Code Snippet 12.3: *The default block configuration.*

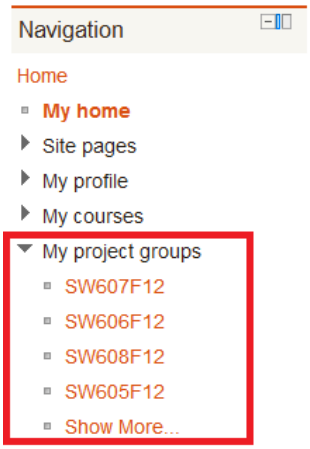


Figure 12.1: *Moodle navigation block. The highlighted box shows our extension with a list of project groups, which the currently logged in user is a member of.*

and presenting these along with a picture for each member, which is fetched through a Moodle API call.

The layout of a project group view can be seen on Figure 11.1 in Chapter 11.

12.2.2 Navigation

From Section 8.2 we have a requirement called Navigate to Project Groups which states that a user should be able to find the project group(s) that he is a member of. This implementation that satisfies this requirement is presented here. Since we implement the project group room as a local plugin we can use a build-in functionality of Moodle to extend the navigation block [39]. The navigation menu can be seen in Figure 12.1. The extended part is highlighted by the rectangle.

The code to add project groups to the navigation block can be seen in Code Snippet 12.4. The currently logged in user is accessed to get the project groups that he is a member of. The code checks if something should be displayed at all and then run a loop to print out the link to each group. If more than four project groups is associate with the given user a link to "Show More" is added and no more links are generated.

```
1 function projectgroup_extends_navigation(global_navigation
      $navigation)
2 {
3     global $USER;
4     $number_of_groups = 4;
5
6         ...
7
8     $groups = get_groups_of_user($USER->id);
9     if(sizeof($groups) > 0)
10    {
11        $type = navigation_node::TYPE_CUSTOM;
12        $my_projectgroup_navigation = $navigation->add(get_string('
              myprojectgroup','local_projectgroup'),null,$type,null,'
              myprojectgroup');
13        $count = 0;
14        foreach ($groups as $group_id)
15        {
16            $count++;
17            if($count > $number_of_groups ){
18                $my_projectgroup_navigation->add('Show More...',
19                    new moodle_url('/local/projectgroup/usergroups.
                          php',
20                    array('id'=>$USER->id, 'sesskey'=>sesskey())),
21                    $type,null,'show_more');
22                break;
23            }
24            $group = get_projectgroup($group_id);
25            $my_projectgroup_navigation->add($group->shortname, new
                   moodle_url('/local/projectgroup/index.php', array(
                  'id'=>$group_id, 'sesskey'=>sesskey())),$type,null,
                  $group->id);
26
27        }
28    }
29 }
```

Code Snippet 12.4: *The code for extending the navigation block.*

12.3 Managing Project Groups

Before any project group can be useful it has to be created first. Administrators need to have the ability to add, edit, and delete project groups. This section describes how we implement the administration tools needed to manage project groups.

The features we provide to manage project groups are known as admin tools in Moodle.

The main functionality is placed in the file /admin/tool/projectgroup/lib.php, which is a code library consisting of $\approx$ 500 lines of code. The library is know henceforth as the admin tool library. The library includes the project group library discussed in Section 12.1, and hereby allowing pages in the administrative part to use code in the project group library. This include is the implementation of the dependency between the admin tool and the project group library seen on figure Figure 10.2.

In the following section creation, editing, and listing are presented. Deletion of project groups are omitted since we consider the functionality used as

trivial. The reader can see the code for the deletion view in /admin/tool/projectgroup/delete.php and the function, `delete_projectgroup`, that does the actual deletion is located in the admin tool library.

12.3.1 Add & Edit

To add and edit project groups, the file /local/projectgroup/edit.php is loaded into the browser. The look and feel of this page is described in Section 11.2. The same file is used for both editing and creation of new projectgroups. The only difference between editing a group and adding a group is that when editing a group an HTTP parameter, *id*, is set. When *id* is set the form fields are filled with the relevant information from the database. Once the submit button is pressed the function `save_or_update_projectgroup` is called with a project group object as input. That function checks if the *id* of the group is set. If it is we know that a project group is being edited, and we can update a row in the database. If not we know that a new project group is being created and we insert a row in the database.

When `save_or_update_projectgroup` is used to update a project group the possible change in members is handled by removing every member from the project group and then adding the ones received as part of the arguments to the call. This means that a project group briefly contains no members whenever it is updated, even when there is no change to the members or their role. However, the communication with the database is handled in a transaction, which means that the database system, MySQL, should ensure that the data is constantly in a consistent state.

12.3.2 Lisiting

In the list of project groups, mentioned in Section 11.2, we provide a filtering functionality, which makes it easy to find a specific set of project groups. The project group filtering is a rework of the user filtering, which already exists in Moodle. The filter itself is placed in /admin/tool/projectgroup/projectgroup_filtering.php and is class named **projectgroup_filtering**. This functionality is requested by the administrative personnel (see Section 8.2). The project group list is rendered in /admin/tool/projectgroup/index.php and the filter is created here aswell. Code Snippet 12.5 shows the invocation of the filter. The filter is created and by the use of the function `get_projectgroup_selection_data` we get the data which we use in the creation of the projectgroup form, which handles the HTML rendering of the filter.

```
1 //create the user filter form
2 $filtering = new projectgroup_filtering();
3
4 //get the data from the form
5 $data = get_projectgroup_selection_data($filtering);
6 $data['fields'] = $filtering->get_fields();
7 $editform = new projectgroup_form(NULL,$data);
```

Code Snippet 12.5: *The invocation of the filtering mechanism.*

Chapter 13

Test

This chapter describes how we perform testing of the code written in this project. Additionally there is a discussion on how effective the testing is. Testing is important, since it helps find the bugs in the program and thereby making the end product better.

13.1 Strategy

Our development is to some extent test driven. This means that we are creating unit tests for our functionalities and integration tests to test whether or not our sub-system as a whole work as expected and work together with Moodle. As mentioned in Section 5.2.4 we are using SimpleTest to test our PHP code. When we know what a function is supposed to do prior to its implementation we write test cases for the function. If we do not write test cases prior to its implementation we write test cases for the function after its implementation. Since working with Moodle is new to us we do not have a full understanding of the Moodle core API. This means that even with the documentation of the Moodle core API we may misinterpret the behavior, input, or output of a function. Many bugs related to this are discovered by writing test cases.

There is code in this project which is not included in functions and are not included in any unit tests. This include code belonging to our user interfaces (UI). These UIs are tested using UI testing, which is conducted by the developer to ensure that the functionality is as expected, and through the demo meetings where the system is tried out.

The test cases we write do not catch all bugs in the system, but that does not mean that they do not serve a purpose. Apart from the bugs they actually catch, they also improve the robustness of the system. If someone decides that the implementation of a function should be altered the test cases can be used for regression testing and thereby reducing the risk of failures. This is especially important in our case since a new group of people will continue our work later.

A metric for determining how well some software is tested is code coverage. The specific type of coverage we measure our code with is line coverage. Determining the percentage of coverage we strive for is a cost/benefit situation. On one hand we want a stable system that is working as intended, but on the other hand we do not want to spend all our time writing test cases. The importance of testing is based on the criticality of the system. The radar chart in

Section 5.1.2 shows that if our system fails it will only affect the comfort of the users. Low criticality calls for a relatively low code coverage. We choose that 60% line coverage is sufficient for our core functionality. Our core functionality being the admin tool library and project group library. An evaluation of our testing can be seen in Section 13.3.

In addition to considering code coverage we choose to perform equivalence partition [53, pp. 67-69] and boundary value analysis (or data testing) [53, pp. 70-79] in the creation of test cases. Without going into too great details of these concepts, equivalence partitioning means that for every function we test, we partition input values into partitions. Within these partitions the outcome of each element is expected to be the same. Consequently we have to test at least one element in each partition, but not necessarily every element. A boundary value analysis is performed by identifying boundary values and selecting values around each boundary value for testing, e.g. if 0 is a boundary value for some integer input, -1, 0, and 1 is chosen for test values. Boundary values are identified as values that delimit a boundary, described by Patton to be "like the edge of a cliff" [53, p. 71]. These can be the value where elements change from being in one partition to being in another. Alternatively it can be special objects of the certain input type, such as null, empty string, or empty array.

13.2 Test Implementation

Implementing the testing is done by writing a number of test cases that use the functions in our sub-system and check if they behave as they should. There is built-in functionality in Moodle to see how many of the SimpleTest test cases pass. This tool is used every time a function is implemented, to see if the functions behavior corresponds to our expectation of it.

In Section 13.2.1 we will show some examples of our test cases. The entire set of test cases can be found in our source code folder. We will draw our examples from the tests of the function `remove_projectgroup_members`, which is responsible for removing a set of members from a project group. The documentation of `remove_projectgroup_members` states that the inputs are a project group id, **group_id**, and a non-empty set of user ids, **users**, belonging to users who are members of the group initially. We create partitions and perform boundary value analysis on both these inputs.

The **group_id** is partitioned into two partitions. The two partitions of **group_id** are; ids that belong to an existing project group, and ids not belonging to an existing project group. This yields two tests for **group_id**. The test case for an id belonging to an existing project group is tested with a value n where the id $n+1$ is not belonging to a project group, i.e. the id n is the newest project group. The test case for an id not belonging to an existing project group, we test it with value m where both ids $m-1$ and $m+1$ belongs to project groups and id m does not. As part of the boundary value analysis of this input we deem that we should try to use something that is not and integer as input, we choose to use a string. We deem that every invalid input value belongs to the same equivalence partition, therefore we do not test for input of objects, null values, arrays etc. This yields three test cases for this function. In each of these **users** is a valid value.

The partitioning of **users** is not as straight forward. Here we have cases

where the ids to be remove can be either be all, partly, or not at all valid, depending on whether or not the given id belongs to a user that is a part of the project group being tested. We choose to partition it coarsely; either all the ids of the set to be removed belongs to valid members or at least some id does not belong to a member. The reason being that if the function can handle a single invalid user id it is likely that it can handle that some or all is invalid. The boundary values for these partitions we are considering are as follows:

- Removing a single member from a project group
- Removing every member from a project group
- Removing a user that is not a member of the project group, where the project group in question has members.

Additionally we test the case where the set of ids to be removed is empty and is an illegal type. For the illegal type we choose two cases; one where the input is an array but one of the elements is a string instead of an integer and a case where the input is an object instead of an array. The final test case regarding **users** is where the call to the database API indicating the actual deletion of data fails. This test case is inserted to test how the function handles unexpected situation regarding function calls to trusted API or sudden database failure (though the database is responsible for recovery itself).

13.2.1 Test Case Examples

```
1  function testRemoveMembersFromGroupEveryone() {
2      global $DB;
3
4      $groupmembers = 6;
5
6      //The data
7      $projectgroup = new stdClass();
8      //Set data - shortname, fullname, and members
9      ...
10
11     //create the test group with members
12     $groupId = $this->createTestGroup($projectgroup);
13     //Read back users and ensure that the project group contains 6
           members
14     ...
15
16     remove_projectgroup_members($groupId, $membersID);
17
18     //array with the members in the group. Should be empty.
19     $delMembers = $DB->get_records($this->groupMemTableName, array(
           "projectgroup"=>$groupId));
20
21     $this->assertTrue(empty($delMembers));
22 }
```

Code Snippet 13.1: *A test case for the function* `remove_projectgroup_members`. *The test case tests if the function correctly removes all the members of the project group when instructed to..*

In Code Snippet 13.1 a test case for the function `remove_projectgroup_members` is shown. The test case in question checks if the function can correctly remove all members of a group. In line 12 a group of six users is created. The function `createTestGroup` utilizes the function that is used to create project groups in the Moodle application, but it also ensures removal of the project group after the tests are run, so the database does not get polluted with test project groups. In line 16 the function `remove_projectgroup_members` is used to remove all six members of the project group. Finally in line 21 we assert that the project group at hand is empty. If it is empty the test succeeds, if not it fails. This test case is an integration test between the function `remove_projectgroup_members` in admin tool library and Moodle DML (see section Section 9.5.1), since we use an actual database for data storage during the test.

```
1  function testRemoveNoUserFromGroup()
2  {
3      global $DB;
4      //Setup
5      $groupId = 1;
6      $data = array();
7
8      //Mocking up database
9      $DB = $this->DBMock;
10
11     //Set expectations
12     $DB->expectNever('start_delegated_transaction');
13     $DB->expectNever('delete_records_select');
14     $this->expectException('coding_exception');
15
16     //Run
17     remove_projectgroup_members($groupId,$data);
18 }
```

Code Snippet 13.2: *A test case for the function `remove_projectgroup_members`. The test case tests if the function correctly handles the erronous input of an empty set of users.*

To make most of the unit tests run we have to to mock up the database API. An example of this is the test case with an empty set of user ids as the argument **users** to `remove_projectgroup_members`. This is seen in Code Snippet 13.2. In lines 12-14 the expectations for the call is set up prior to the call in line 17. The expectations are that there should be no transaction started, no deletion should occur and an exception is thrown. This test case only pass if these expectations are met.

The choice between mocking up the database API and using the actual database API has several factors. Firstly the time it takes to run the tests is an important factor. If every test case connects to the database and performs reads and/or writes on the data the test suite will run slower than if some or all are using a mocked up database API. The time that tests take to run is crucial when using test driven development because a test suite will be run many times and the developer running the test suite needs the respond to continue his development. Secondly some integration testing is needed. We cannot simply choose to use a mocked up database in every test case because we may have misunderstood some of the specification of the Moodle DML. This could lead to calls with invalid input, that would not be caught by the tests because the

developer writing the function using the API invalidly is possibly mocking up the database API to accept the invalid input. Thirdly the database that is used for integration testing is also used to store data used for demo meetings. A test case that tests that a function behaves correctly on errors is better suited for a mocked up database API to avoid loosing data that may be lost if the error is not handled or handled incorrectly.

13.3 Results

In this section we evaluate the code coverage of the test cases. Moodle has a built-in functionality for line coverage of unit tests, which we use to measure the code coverage.

The core functionalities that we are testing are the project group library and the admin tool library. Reports are generated for these and can be seen in Appendix B.2 and Appendix B.1 respectively. We have created 53 test cases for the project group library, and 105 test cases for the admin tool library. The line coverage for the project group library is 68.09% and 78.24% for the admin tool library.

As stated in Section 13.1 the line coverage that we are striving for is 60%. Our line coverage is above this and we consider it acceptable for the kind of system we are building. Our tests are combined with several UI tests which helps find bugs and ensure better quality of the product.

In addition to our test cases presented in Section 13.2 we have conducted demo meetings. These demo meetings have served as generators of requirements as well as validation tests of our system. The demo meetings that we have conducted are found in Appendix A.6 and Appendix A.9. These have shown that our sub-system in general is usable by our end user representatives.

Part III

Evaluation

Chapter 14

Development Approach

In this chapter we discuss and evaluate our development approach. This is split into the two sections **??** *and Intra Group Development. The evaluation of our development approach in important since it forces us to learn what works well. Additionally the evaluation is a service to the students taking over the project. Hopefully they will learn from our mistakes and use what we have found to work well.*

We chose the development method Scrum, which is described in Section 5.1.1.2, refined in Section 5.1.3 and Chapter 7.

14.1 Intra Group Development

14.1.1 Scrum

We were unable to follow the scrum development method completely because some roles were not filled. While still a problem, fulfilling all roles is not paramount in the intra group development of this project. We did not have a constant Scrum master, instead we took turns taking the role of Scrum master. This worked sufficiently well, but it would be preferable to have the role filled by an experienced Scrum master. One could argue that it would have been preferable to have only one person take the role of Scrum master, since he would gain more experience and the development would be more uniform. However, since this is a learning project we all would benefit from being the Scrum master.

An issue in the implementation of Scrum of Scrums is the lack of a product owner, which is described as key concept. We compensated for this lack by doing several interviews with possible end users and to have them included in the demo meetings. In effect we have had the role of the product owner our selves, which is not an optimal solution. It is not optimal for the reason that it is recognized as bad Scrum practice to have many product owners to prioritize and select tasks for a sprint [50, p. 128]. Furthermore, we have conflicting interests – we want to implement something exciting and challenging, while the end users may in fact need something we consider to be simple and boring.

14.1.2 Task Management

When planning a sprint we used planning poker, which is a Scrum practice to estimate how much time a task takes. In the beginning we had very different ideas of how much time a story point was and we did not decide anything specific. After working on tasks our valuation of story points became increasingly similar over time.

We tried several different approaches for defining tasks and backlog items. In the first sprint we created a few backlog items and after estimating them using planing poker we decomposed the items into smaller tasks, which then got evaluated. This lead to a large amount of small tasks, but as progress went on we realized that some tasks were redundant and could be combined, while others were missing to fulfill the backlog items requirements.

In the second sprint we tried to estimate only the backlog items and not create smaller tasks. This made it difficult to hand out small tasks and evaluation the progress during the sprint.

In the third sprint we created one task for each backlog item and the person who took the tasks had the job of starting to program and create smaller tasks that other people can take. This proved to be the most efficient method of distributing tasks.

A burndown chart of remaining story points was a part of the tasks management. It was positioned in a visible position and served as a motivational tool.

14.1.3 End Users

During the development we relied on several end users in the gathering of our requirements. The end users selected was not as spread out as we initially had wished for. This is mainly because it is difficult to find end users who are willing to volunteer for interviews. For both target groups, the users of the project groups and the administrative personnel, we lack end users from the Faculty of Medicine. For the usage of the virtual group room we need users with high experience and more than one supervisor for interviewing. The fact that we did not fully cover all types of people for interviews does not render the requirements useless, but it gives room for further improvement.

Chapter 15

Product

In this chapter we discuss and evaluate our developed product. We focus on MyMoodle and our sub-system in Section 15.1 and Section 15.2 respectively. The arguments presented in this chapter are used to conclude whether or not the developed system live up to our system definitions.

15.1 MyMoodle

MyMoodle is the system we as the four groups collaboratively have developed. In this section we will evaluate on the the product, which we created.

In this project we created an extension to Moodle, which supports education through the Aalborg PBL model. The extension, consisting of several smaller extensions, gives the students and their supervisor a virtual representation of their project group. The virtual project tools allow students to communicate internally through a message board and a black board, and communicate with their supervisor through the supervisor message board. The students can make appointments with themselves and/or their supervisor through the timeline tool. The virtual group room is a place where all the virtual project tools are available. Each project group has a virtual group room associated to it.

15.1.1 Decomposition

We chose to split the entire multi-project into four parts based on the core principles of PBL. In retrospect the split could have been done more harmonic than it is. Our group got the task of implementing the project group administration and the virtual group room, which is two very separate tasks with different end user groups. The black board tool could be merged with the message board tool by integration an existing drawing application, such as google apps, which is already integrated in Moodle by a 3^{rd} party extension [44]. A new decomposition will then be:

- Management
- Planning
- Communication
- Virtual Group Room

This division would have give a more distinct group of end users to all components. The distribution of end users would be: administrative personnel to Management, students to the others, where Communication would have to consider supervisors as well.

15.1.2 Architecture

The architecture was not the result of an up front analysis but rather the result of a few iterations of development. We discovered that there was a need for a specification of where to put new functionality. In particular we wanted a shared code library where functionality that was used be some or all parts of MyMoodle could be placed. This resulted in our three-layered architecture (not counting the database and Moodle core) described in Section 10.3. It has allowed us to have a common understanding of how we wanted the final system to be. It has also given guidelines regarding where to put functionality and has encapsulated related functionalities. This should help the developers that will take over this project next year to add new features as well as changing existing features.

15.1.3 Database

The database schema used in this project is fairly simple. It only consists of two tables, namely `projectgroup` and `projectgroup_members`. These two tables represent the entity project group and the relation that exists between project groups and users in Moodle.

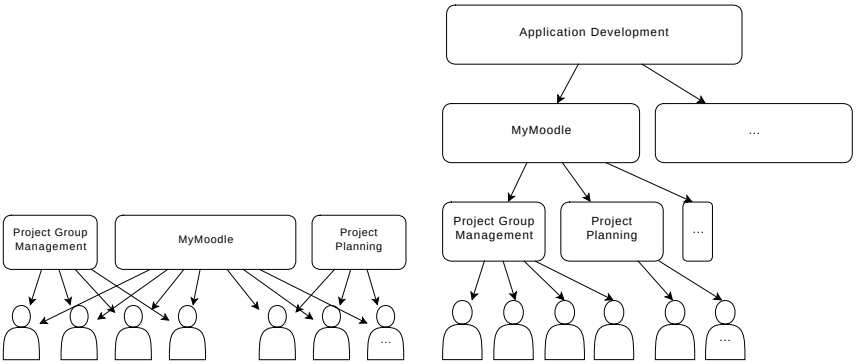
The **longname** field in `projectgroup` is by Lene W. Even considered useless, as described in Appendix A.9. The field is optional, but it should be removed if members of the administrative personnel from other departments confirm that there is no need for it. In the `projectgroup_members` there are two time fields – **created** and **updated**. These fields are unused in the current implementation. Since the members of a project group are all deleted whenever a project group is updated these fields are worthless. For these fields to have a purpose the implementation must be changed to update entries in `projectgroup_members` and set **updated** appropriately without modifying **created**.

15.2 Project Group Management

In this section the sub-system implemented by our group is evaluated. This section is divided into the different aspects of this sub-system. The focus is on both design and implementation choices.

15.2.1 Project Group Structure

We decided to implement the concept of project groups in Moodle to satisfy the system definition of MyMoodle. We did not make a distinction between projects and groups but combined the two into one entity, which made the system more simple. If the distinction should be made the concept of projects can be added to the system and a link between project groups and projects must be created. A project would encapsulate the over-arching theme of a semester, e.g. Application Development in our case. The project groups would be the



(a) *Flat group structure.* (b) *Recuresive tree group structure.*

Figure 15.1: *Recursive tree structure and our our flat structure.*

instantiations, so to speak, of the project – MyMoodle in our case. A project can have a page in the same way a project group has a page, which will allow for collaboration and communication between groups in a semester.

An alternative approach to consider is to make project groups nestable; that is to allow a project group to have a parent. This would give even further flexibility. This will allow for large groups of students to be split into several subgroups, which would make a student part of his subgroup and its parent group. It can be implemented using a recursive tree structure. Consider our case, we are four people work in a sub-group, this sub-group along with three other constitute the group developing MyMoodle. The MyMoodle group is part of a semester project with the theme Application Development, that has another group as well – the Android group. In our system this can be accomplished by making a project group for both sub-groups, groups, and the semester. A student is then part of the appropriate project groups. The difference between a recursive tree structure and our implementation is illustrated in Figure 15.1.

15.2.2 Virtual Group Room

In Section 10.1 we decide to implement the virtual group room to satisfy the need for a virtual meeting place. Each project group has its own room for collaboration.

The virtual group room is implemented as a container for blocks, which made the inter-group integration less time consuming by having a common interface between the virtual group room and the virtual project tools. At the same time it gives a great flexibility since the individual group can organize the blocks themselves and add arbitrary blocks.

In the implementation of the virtual group room we created our own context, which gives a better solution compared to linking project groups to courses. Such a link would create an unnatural relation, which does not reflect the problem domain.

15.2.3 User Interface

In the last demo meeting with Lene W. Even (seen in Appendix A.6) she expressed some dissatisfaction with how the filtering functionality of the project groups work. It would be easier for some users to use the filtering of project groups as a search query instead. Specifically this means that filters should as default be discarded when adding a new filter. It should still, however, be possible to have more than one filter.

Additionally, it should be possible to filter the project groups on more attributes. It should be possible to filter project groups by attributes of the members of the project group. These attributes could be the year the students started their education, their group room (if they have one), their email, and the city in which they study.

15.2.4 Administration

We decided that management of project groups should only be done by administrative personnel. This is not a final decision and we reckon that allowing users to create their own project groups is not necessarily a bad idea. With student management of project groups there are several constraints, which must be considered; these are discussed in Section 10.2.2. The constraints that should be considered are mainly security and integrity related. Examples of constraints that must be considered include users cannot delete project groups prematurely. Many of the constraints that apply to student management also apply to administrative personnel management, but we believe that administrative personnel are less likely to perform any destructive action than students. First of all it is possible for administrative personnel to receive training in the usage of the system, such training is much more expensive to give to students because there is simply more students than administrative personnel. Secondly it is the job of the administrative personnel to administrate entities in the university, hence they have a responsibility that students do not.

When the administrative personnel manages project groups the system has a possibility to become authoritative. That is, the system can be used as primary project group database. To become an authoritative project group database the structure of the project groups and projects, which are discussed in Section 10.1.3, must be changed to be more general to support all structures at the university (see Appendix A.4). With student management the authority of the system is removed and will only be a tool for students and their supervisors.

15.2.5 Testing

In the process of creating our sub-system of MyMoodle we used TDD. It worked well for us during the implementation of the core functionality, because it made any misconceptions we had of the Moodle core functions clear. We did not use TDD for the creation of user interfaces.

Testing of the developed system is done using the SimpleTest framework. A framework already included in Moodle and hence we did not have to spend time integrating a testing framework.

The result of the tests showed a good code coverage percentage and a large amount of test cases which combined with the demo meetings shows that the

sub-system is stable. The system does not have a high criticality and attempting to achieve a more thoroughly test suite is not necessary.

In the worst-case, e.g. the system is unavailable and all data is lost, the users does not lose money or lives, assuming that MyMoodle is not used as an authoritative system. Hopefully this will never occur, but cannot be ruled out since it is impossible to declare a program bug free using testing. To overcome this potential loss the database and user files must be backed up.

Chapter 16

Conclusion

In this chapter the project is concluded upon in two ways. Initially we will conclude upon the multi-project, and then we will conclude upon our sub-project.

16.1 Multi-Project

In the first part of this report the system definition for the multi-project, MyMoodle, is presented as the following:

> *MyMoodle is an extension to Moodle that allows Moodle to support the Aalborg PBL model.*

Through our analysis we decided that to support the Aalborg PBL model we make a virtual meeting place available for project groups in Moodle.

The services that the virtual group room provide are a virtual blackboard, a tool for communication between students and supervisors, and a time management tool. The virtual blackboard is a place where creative work and notes can be produced collaboratively by students in a project group and saved for later use. Communication between students and supervisors is done by writing on a message board and planning meetings. Each project group has an individual message board and each supervisor has a personal message board displaying messages from every project group that he supervises. The time management tool allows for arrangement of tasks, and shows these in a simple timeline display. This timeline also shows blackboard events and planned meetings.

We believe these tools allow students to work in a project group following the Aalborg PBL model through the existing platform Moodle. However, since we have only interviewed a small number of people we cannot be sure that the system we have developed is actually a useful tool on a large scale.

The virtual group room is implemented as a "local plugin" to Moodle. The content of the virtual group room is Moodle blocks. These contain the tools described above or have links to the functionality of the tools. Since every component of MyMoodle is implemented as a Moodle plugin we have not changed anything in the core code of Moodle. This ensures that we do not change other functionality in Moodle. Thereby MyMoodle is an extension to Moodle that supports the Aalborg PBL model as defined in the system definition.

16.2 Sub-Project

In the second part of this report the authors defined the developed subsystem as:

> *A sub-system of MyMoodle that implements project groups in Moodle and allow for administration and usage thereof. The sub-system includes a virtual meeting place, which integrates the other sub-systems.*

We implemented an administration tool in accordance with Moodle standards. By using this tool administrators are able to add, edit, and delete project groups. In order to accommodate for a large number of project groups we have created a page with a list of all project groups. Finding a specific project group or a specific set of project groups is possible by using filtering.

Additionally, we created a project group room page based on requirements gathered from interviews and demo meetings conducted during the project. Each project group has one of such pages associated with it. This page shows all the group members with name and profile picture, and allows other Moodle blocks to be displayed.

To allow integration between the sub-systems of MyMoodle we provide a code library that is available in every Moodle page. This library has data retrieval functions to project groups. These include retrieval of project groups related to a specific user and retrieval of project group information (name, members, etc.) of a specific project group. For our administrative tool we have created a separate library. Our libraries are documented and tested to an acceptable extend for the kind of system we have developed.

The structure of our system allows for further development in a modular fashion. It also allows for changes in the implementation while keeping the system robust.

The system has not been tested on a large scale or in a real world scenario. It has only been tested with test project groups, and only been used a few times by administrative personnel. We cannot be certain that the system will work as intended on a large scale and operated by current staff.

Chapter 17

Future Work

In this chapter we present the ideas that we believe to be the most valuable additions to MyMoodle. While Chapter 15 focuses on alterations and improvements to the system, this chapter focus on new features and alternative usages of the system. There are several features and extensions to the system, but only those we regard as highest prioritized are mentioned. This chapter is relevant because students should continue on our work later.

17.1 Physical Group Rooms

In the current implementation of MyMoodle there is a concept of group rooms. This concept is strictly virtual – there is no understanding of actual physical group rooms.

If there were an implementation of physical group rooms in Moodle it would be possible to implement a room booking system. As mentioned in Section 10.1 some groups have to book group rooms daily. This takes the time of administrators and students. Such a system could also be expanded to handle the booking of lecture rooms.

There are several constraints that have to be considered before implementing such a system. First of all there should be priorities based on the person wanting to reserve a room, e.g. a lecturer should take precedence over a student when trying reserve a lecture room. Additionally there should be a system in place that handles a situation where there are not a sufficient number of group rooms available; some groups should not be able to make a significantly greater number of reservations than some other groups.

We believe such a system has a sufficient complexity and relevance that it could serve as a project for one of next year's 6^{th} semester software groups.

17.2 Central Project Group Database

An issue which is discussed in Section 15.2.4 is the lack of a single system for managing all project groups at Aalborg University. This issue can be solved by letting Moodle become the authoritative database for project groups. To become this several issues must be overcome.

- The structure of the project groups must be very general to support all the possible project groups structures there exist at Aalborg University.
- There should be only one Moodle installation with one database.
- The system must have a strong backup system.
- The system should have an accessible API from which project groups can be exported to other systems if needed.

The implementation issues concerned with Moodle becoming a primary database is minor compared to the task of getting the administrative board at Aalborg University to decide to implement and use the system. We do not recommend this as a feature valuable of implementation as long as there exist more than one Moodle installation at Aalborg University.

17.3 Virtual Meeting Place Templates

skriv om at bruge templates til at oprette project group og hvordan det kunne gøres. Dette emner er berørt i requirements.

The feature will enable administrative personnel to create virtual group rooms with predefined templates. In the current system we have one template for the virtual meeting place. This template specifies a set of activated blocks for the virtual meeting place. These blocks have been chosen by us, but our choice for this may not be optimal for all departments and by having multiple templates each department can have their own template.

To implement this feature the function `blocks_add_default_projectgroup_blocks` must be changed so it instead of getting the list of default blocks from a configuration it should get the name of a list. Based on this name the specific set of blocks should be added to the virtual group room.

To figure out which blocks are best suited as default for the virtual meeting places in the various departments the students and supervisors at the departments must be interviewed or answer questionnaires.

Alternatively, the students could grant privileges to create their own templates and share these with each other. This way we avoid the risk of having conflicting wishes about templates in a given department.

Implementing this feature may or may not constitute for an entire semester project, but is never the less a feature that can be considered. Perhaps it can be implemented as part of a bigger feature.

Bibliography

[1] Adobe connect. URL http://www.adobe.com/products/adobeconnect.html. Last viewed: 22/5-2012.

[2] Definition af e-læring, . URL http://www.itst.dk/ferdigheder/e-lering/om-e-lering/definition-af-e-lering/definition-af-e-lering. Last viewed: 18/4-2012.

[3] Definition of e-learning, . URL http://www.webopedia.com/term/e/e_learning.html. Last viewed: 2/5-2012.

[4] Dropbox. URL https://www.dropbox.com/. Last viewed: 22/5-2012.

[5] E-læringssamarbedet ved aalborg universitet. URL http://www.elsa.aau.dk/. Last viewed: 19/4-2012.

[6] Google docs. URL https://docs.google.com/. Last viewed: 22/5-2012.

[7] Gnu general public license. URL http://www.gnu.org/copyleft/gpl.html. Last viewed: 19/3-2012.

[8] Litmos. URL http://www.litmos.com. Last viewed: 9/5-2012.

[9] Mahara. URL https://mahara.org. Last viewed: 9/5-2012.

[10] Moodle, . URL http://moodle.org/. Last viewed: 19/3-2012.

[11] Unit test api, . URL http://docs.moodle.org/dev/Unit_test_API. Last viewed: 7/3-2012.

[12] Master in problem based learning in engineering and science. URL http://www.mpbl.aau.dk/. Last viewed: 19/4-2012.

[13] Sharepointlms. URL http://www.sharepointlms.com. Last viewed: 9/5-2012.

[14] Simpletest: Unit testing for php. URL http://www.simpletest.org/. Last viewed: 7/3-2012.

[15] Skype. URL http://www.skype.com/. Last viewed: 22/5-2012.

[16] Apache subversion. URL http://subversion.apache.org/. Last viewed: 6/3-2012.

[17] Scrum of scrums, 2007. URL http://www.scrumalliance.org/articles/46-advice-on-conducting-the-scrum-of-scrums-meeting/. Last viewed: 2/5-2012.

[18] Analysis of git and mercurial, 2008. URL http://code.google.com/p/support/wiki/DVCSAnalysis. Last viewed: 14/3-2012.

[19] Studieordningen for bacheloruddannelsen i software, 2009. URL http://www.sict.aau.dk/digitalAssets/3/3331_softwbach_sept2009.pdf. Last viewed: 22/5-2012.

[20] Blocks/appendix a. Webpage, December 2011. URL http://docs.moodle.org/dev/Blocks/Appendix_A. Last viewed: 23/05.

[21] Activities. Webpage, 2012. URL http://docs.moodle.org/22/en/Activities. Last viewed: 03/05.

[22] Activities block. Webpage, 2012. URL http://docs.moodle.org/22/en/Activities_block. Last viewed: 03/05.

[23] Eventum issue / bug tracking system, 2012. URL http://dev.mysql.com/downloads/other/eventum/. Last viewed: 13/3-2012.

[24] Eventum issue / bug tracking system, 2012. URL http://dev.mysql.com/downloads/other/eventum/features.html. Last viewed: 12/3-2012.

[25] Activities modules. Webpage, March 2012. URL http://docs.moodle.org/dev/Activity_modules. Last viewed: 23/05.

[26] Navigation block. Webpage, 2012. URL http://docs.moodle.org/21/en/Navigation_block. Last viewed: 04/05.

[27] Plugins. Webpage, 2012. URL http://docs.moodle.org/dev/Plugins. Last viewed: 02/05.

[28] Morten Mathiasen Andersen, 2012. URL http://personprofil.aau.dk/121493?lang=en. Title: Assistant.

[29] Mads Peter Bach, 2012. URL http://personprofil.aau.dk/102508?lang=en. Title: Systems Administrator.

[30] Scott Barge. Principles of problem and project based learing - the aalborg pbl model, Sep 2010. Last viewed: 18/4-2012, Folder describing the Aalborg PBL Model.

[31] Alex Moesgård Bek, Dianna Hjorth Kristensen, Rasmus Oxenbøll Lund, Nikolaj Dam Larsen, and Kenneth Eberhardt Jensen. E-learning mangement system : Implementing customisable schedules, 2011. URL http://projekter.aau.dk/projekter/files/52575492/Report.pdf.

[32] Barry Boehm and Richard Turner. Observations on balancing discipline and agility. In *Agile Development Conference, 2003. ADC 2003. Proceedings of the*, pages 32 – 39, june 2003. doi: 10.1109/ADC.2003.1231450.

[33] Mikkel Boysen, Christian de Haas, Kasper Mullesgaard, and Jens Laurits Pedersen. E-lms - authentication, database, and educator, 2011. URL http://projekter.aau.dk/projekter/files/52596547/document.pdf.

[34] Individual bugzilla.org contributors. Bugzilla: Features, April 2011. URL http://www.bugzilla.org/features/. Last viewed: 12/3-2012.

[35] Lillian Buus, 2012. URL http://personprofil.aau.dk/profil/103311?lang=en. Title: Staff Member with University Degree.

[36] Mark Drechsler. Moodle structural overview, 2012. URL http://www.slideshare.net/mark.drechsler/moodle-structural-overview. Last viewed: 9/5-2012.

[37] Ryann K. Ellis. Field guide to learning management systems, 2009. URL http://conference.unctlt.org/proposals/presentations/conf4/900_LMSfieldguide1.pdf. Last viewed: 19/4-2012.

[38] Lene Winther Even, 2012. URL http://personprofil.aau.dk/107261. Title: Senior Secretary.

[39] Moodle Forum. Extending moodle navigation block. Webpage, March 2012. URL http://moodle.org/mod/forum/discuss.php?d=170325&parent=753095. Last viewed: 2012-05-09.

[40] Marie Glasemann, 2012. URL http://personprofil.aau.dk/116492?lang=en. Title: Head of Section.

[41] Boris Gloger. Your scrum checklist. Webpage, March 2011. URL http://people.cs.aau.dk/~jeremy/SOE2012/resources/Scrum%20CheckList%202011.pdf. Last viewed: 2012-05-17.

[42] Marcus Green. Moodle. Webpage, 2011. URL http://www.examulator.com/er/. Last viewed: 2012-05-09.

[43] The PHP Group. Package information: Phpdocumentor, 2012. URL http://pear.php.net/package/PhpDocumentor/. Last viewed: 6/3-2012.

[44] Kim Hammond, Helen Foster, chris collman, Jason Hardin, Jonathan Harker, Nadav Kavalerchik, Tony Butler, and Abed Islam. Google apps integration. Webpage, May 2012. URL http://docs.moodle.org/22/en/Google_Apps_Integration. Last viewed: 2012-05-28.

[45] Mikael Møller Hansen, 2012. URL http://personprofil.aau.dk/106741?lang=en. Title: IT Administrator.

[46] Jette Egelund Holgaard, 2012. URL http://personprofil.aau.dk/103630?lang=en. Title: Associate Professor.

[47] Steven Kerschenbaum and Barbara Wisniewski Biehn. Lms selection: Best practices. URL http://www.trainingindustry.com/media/2068137/lmsselection_full.pdf. Last viewed: 26/4-2012.

[48] Pia Kruse Knudsen, 2012. URL http://personprofil.aau.dk/109828?lang=en. Title: Senior Clerk.

[49] Anette Kolmos, Flemming K. Fink, and Lone Krogh. *The Aalborg PBL model*. Aalborg University Model, 2004. ISBN 87-7307-700-3.

[50] Craig Larman. *Agile & Iterative Development*. ADDISON WESLEY, first edition, 2004. ISBN 0-13-111155-8.

[51] Camilla Gæraa Larsen, 2012. URL `http://personprofil.aau.dk/124239?lang=en`. Title: Trainee Clerical Assistant.

[52] Nilesh Parekh. The waterfall model explained, September 2011. URL `http://www.buzzle.com/editorials/1-5-2005-63768.asp`. Last viewed: 19/4-2012.

[53] Ron Patton. *Software Testing*. Sams Publishing, July 2005. ISBN 0-672-32798-8.

[54] Mary Poppendieck. A rational design process – it's time to stop faking it, April 2000. URL `http://www.leanessays.com/2010/11/rational-design-process-its-time-to.html`. Last viewed: 6/3-2012.

[55] Thomas Ryberg, 2012. URL `http://personprofil.aau.dk/109500`. Title: Associate Professor.

[56] Mikkel Todberg, Peter Hjorth Dalsgaard, and Jeppe Lund Andersen. E-lms - administration, calendar, model, educations, courses, 2011. URL `http://projekter.aau.dk/projekter/files/52553822/E_LMS.pdf`.

[57] Gareth Watts. Phpxref, December 2010. URL `http://phpxref.sourceforge.net/`. Last viewed: 6/3-2012.

[58] Moodle Wiki. License, August 2011. URL `http://docs.moodle.org/dev/License`. Last viewed: 29/5-2012.

[59] Moodle Wiki. About moodle. Webpage, April 2012. URL `http://docs.moodle.org/22/en/About_Moodle`. Last viewed: 2012-05-09.

[60] Moodle Wiki. Access api. Webpage, 2012. URL `http://docs.moodle.org/dev/Access_API`.

[61] Moodle Wiki. Admin tools. Webpage, January 2012. URL `http://docs.moodle.org/dev/Admin_tools`. Last viewed: 2012-05-09.

[62] Moodle Wiki. Core apis. Webpage, 2012. URL `http://docs.moodle.org/dev/Core_APIs`.

[63] Moodle Wiki. Course categories. Webpage, March 2012. URL `http://docs.moodle.org/22/en/Course_categories`. Last viewed: 2012-05-09.

[64] Moodle Wiki. Course formats. Webpage, April 2012. URL `http://docs.moodle.org/22/en/Course_formats`. Last viewed: 2012-05-09.

[65] Moodle Wiki. Database. Webpage, April 2012. URL `http://docs.moodle.org/dev/Database`. Last viewed: 2012-05-14.

[66] Moodle Wiki. Data manipulation api. Webpage, 2012. URL `http://docs.moodle.org/dev/Data_manipulation_API`.

[67] Moodle Wiki. File:moodle-contexts-1.8.png. Webpage, 2012. URL `http://docs.moodle.org/dev/File:Moodle-contexts-1.8.png`.

[68] Moodle Wiki. Form api. Webpage, 2012. URL `http://docs.moodle.org/dev/Form_API`.

[69] Moodle Wiki. Output api. Webpage, 2012. URL `http://docs.moodle.org/dev/Output_API`.

[70] Moodle Wiki. Page api. Webpage, 2012. URL `http://docs.moodle.org/dev/Page_API`.

[71] Moodle Wiki. Roles and modules. Webpage, 2012. URL `http://docs.moodle.org/dev/Roles_and_modules`.

[72] Moodle Wiki. Unit test api. Webpage, 2012. URL `http://docs.moodle.org/dev/Unit_test_API`.

List of Figures

List of Code Snippets

Appendix

Appendix A

Interviews & Demo Meetings

A.1 Lene Winther Even

Interviewers: Kim A. Jakobsen and Mikael Midtgaard

This interview was conducted on March 15th at Institute of Computer Science, Cassiopeia, at Aablorg University. At the time of writing Lene is a Senior Secretary at Aalborg University, and one of her tasks is to handle many aspects of Moodle for the Institute of Computer Science.

The following is a summery of the interview.

Moodle in General

It is a problem that there are 13 different Moodle systems at Aalborg University, especially since some student are required to use more than one Moodle system.

When there is a change in a calender event it is difficult to administer in Moodle.

Only students, administrative personnel, teachers, supervisors use Moodle. Research groups do not use Moodle to share information.

Project Groups

Project groups exist in a document written manually into Moodle. They are saved as part of a semester course page. Currently the university do not differ between groups and projects. Messages for project groups could be improved by sending the messages through Moodle instead of email. Messages to entire semesters are already sent through Moodle. A forum can be created for each project group to be used for a group's internal and supervisor communication.

Other Tools

Lene tells us of some of the other tools that are used in relation to the Moodle system.

ADMDB

ADMDB is the Administration Database and is maintained by the Information Services Technology department(IST). The database is not currently linked with Moodle; all information from the database has to be written manually into the Moodle system. The database was linked with the old TYPO3 system.

CalMoodle

CalMoodle is a calender system for courses. Calender events are created in CalMoodle and imported to the Moodle system. Calender events have no direct relation to courses.

Office

Different programs from the Microsoft Office suite are used to store and manage data locally.

Courses

Some courses at Aalborg University share some of the same information. However, the common information has to be written to the courses individually. It is highly wanted for courses to share some common information.

Administrative personnel are enrolled to many courses, which all are listed on the front page and in the navigation menu. This clutters the main page and makes navigating to a specific course difficult.

kontroller at vi ikke har brugt denne forkortelse tidligere

School of Information and Communication Technology (SICT) is responsible for creating a Moodle course page for each course as well as a Moodle course page for each semester. The administrative personnel write some general information to the courses. Additional information are written by the lecturer. Course information is often copied and pasted different places.

Students are automatically added to the semester course, but they have to manually enroll to every other course. Teachers are added to courses with the role of a student, and an administrator has to manually promote them.

Archiving

Finished reports are the only thing that is currently being archived. They are archived in the project database. Supervisor contact and meetings are relevant candidates for being archived. ELSA have plans for archiving all courses and course information for future reference.

A.2 Morten Mathiasen Andersen

Interviewers: Kim A. Jakobsen and Alex Bondo Andersen

This interview was conducted on March 22th at Department of Development and Planning at Aablorg University. Morten is working at MPBL. MPBL educate students in Problem based learning. All education is conducted online

and the students are often residents in foreign countries. At the time of writing Morten is an Assistant at Aalborg University, and supports lectures and students in daily work with Moodle.

The following is a list of topics that summarize interview.

Courses

At the start of a new semester, ELSA creates new Moodle course pages, Morten – in cooperation with the course lecturer – populates the course page. In this process Morten uses archived course material when filling out the course page.

Projects

When projects are conducted the students are responsible for forming groups. The formed groups are not administered by the university. Morten explains that Skype and or Adobe Connect is used as communication medium, this is necessary because of geographical differences. A common problem for the group are how to share files, thus are all groups using different tools. If all tools where integrated in Moodle, it would make the administration task easier.

Improvements to Moodle

All information is provided through Moodle forums. To insure that everybody reads the information, a system could be developed that alerts the user of unread posts. Moodle course pages are static in the appearance, Morten wishes that it was be possible to add banners to the individual course page. Lectures at MPBL are held online via video steaming tools. It is desirable to chat through Moodle during the lectures.

A.3 Jette Due Nielsen and Pia Knudsen

Interviewers: Kim A. Jakobsen and Alex Bondo Andersen

This interview was conducted on March 19th at Department of Communication and Psychology. Two persons are interviewed, Jette is the target of the interview and Pia supports with comments and omissions. At the time of writing Jette and Pia are employed as Senior Clearks at Aalborg University.

They primarily use Moodle to send information to the students.

The following is a list of topics that summarize interview.

Courses

When new course pages needs to be created, an email is send to ELSA with an excel form containing the participants and lectures. Archiving of course pages are done by ELSA and the end of a semester. Jette and Pia do not use old archived courses in their work with Moodle.

Projects

When the student form groups in the start of the semester, a list of the group composition is send to the administrative personal. A forum is created for each group, it is then used for communication between the members of the group. Group rooms are available on a day-to-day basis. The students acquire group rooms in a first come, first served manner.

Improvements to Moodle

When writing a forum post it is only possible to attach one file with the post. If several files need to be attached several post must be made.

A.4 Mikael Møller Hansen

Interviewers: Kim A. Jakobsen and Rasmus Veiergang Prentow

This interview was conducted on April 12th at Cassiopeia at Aablorg University. At the time of writing Mikael is IT Administrator, he is engaged in developing a system that supports the daily work at the university

The following is a list of topics that summarize the interview.

AdmDB

AdmDB is a tool used by administrative personnel at Cassiopeia. It contains information about student and the project groups. It was developed as a supportive system to the old Typo3 system. When project groups are typed into the system, a repository is created for the given group. A project group have the following properties:

- A group room alias
- Room number
- Supervisor
- Students
- A period where the project group is active
- Repository

Mikael explains that AdmDB i necessary at the moment but he hopes that a central system will be developed to administrate project groups in all department on Aalborg University.

MyMoodle

We explain the concepts behind MyMoodle and asks Mikael to comment. We talk about using AdmDBs groups in MyMoodle thus allowing us to automatically create project groups. He replies that we should wait because he currently developing a new API for AdmDB using RESTful architecture.

Other Systems

Mikael mentions that AdmDB is not used on the entire university. Other departments are using their own systems to administrate project groups, some of which are analog – post-its on a wall.

A.5 Thomas Ryberg

Interviewers: Kim A. Jakobsen, Lasse Rørbæk Nielsen and Alex Bondo Andersen

This interview was conducted on March 8th over the VoIP service Skype. At the time of writing Thomas is a Associate Professor and PHD supervisor at Aalborg university. Thomas has conducted research in the field of e-learning.

The following is a list of topics that summarize interview.

E-Learning

After we explained the concept of our project to Thomas he recommends a few system that are used in other e-learning contexts. The suggested systems are Bigbluebutton, Mahara and Podio.

Courses

Thomas explains that populating of the course pages in Moodle is done by administrative personal and professors.

Advising

In Thomas's experience the students use dropbox to share documents internal in their project groups. The student study on the humanistic faculty.

A.6 Lene Winther Even Demo Meeting Sprint 3

Group members present: Alex Bondo Andersen, Kim Ahlstrøm Jakobsen, and Mikael Midtgaard
Administrative personnel present: Lene Winther Even

This demo meeting was conducted on April 4th our the group room at Cassiopeia at Aalborg University. We briefly showed Lene the administrative tool that we implemented and asked her to use it and give feedback.

The following is the relevant parts of the demo meeting.

Using the System

Lene tried to add a project group with a few members. She noted that the list of members added and the list of users not added was reversed compared to the creation of courses in the Moodle template she was used to. She also mentioned

that the search field for users in the system is usually located under the two lists of users, not above. She had trouble with the the filtering functionality because a new filter did not replace an old one, but added it in conjunction with the existing one(s).

After the project group was created she was asked to find the group and alter it. She could not find the project group because there were simply too many project groups. She tried to use the internal search field, but it did not help. She suggested that there should be a search functionality similar to that of members to add to a project group.

Additional Comments

After Lene tried using the system she had a few comments and ideas, which are presented below.

- The list of members does not contain the full name of the users. It would greatly help to have the full name to identify the students and supervisors.
- When asked if she could see any use of a recursive tree structure, she could not come up with any work task where she would benefit from it.
- The member overview seems useful to identify users.
- Perhaps instead of using the term "Full Name", the term "Project Group Name" would be better.
- To avoid double work an integration with ADMDB would be great. ADMDB is the system where project groups are inserted at semester start.
  - It should be possible to get every project group of a semester at a time.
  - Supervisors of project groups should be retrieved from from ADMDB as well.
  - ADMDB is only used at Institute of Computer Science.

A.7 Lea Gustavson

The documentation of this interview is written by the Project Planning group.

Date: 07/05/12 Participants: Lea Gustavson, Anders Eiler, Kim Jakobsen, Henrik Koch, Jesper Dalgas
Interviewer: Anders Eiler

Lea is given a complete walk-through of our system. A casual conversation is conducted about the different elements. The following is a summary of the conversation between Lea and Eiler:

Meetings

- A pop-up appears when creation of a meeting is complete, it is disruptive.
- When a user enters the page for a meeting, no back button is visible.

Timeline

- It would be nice if it was possible to show scheduled lectures on the timeline.

Tasks

- When creating a task, the calender does not close when a date is selected.

Wall

- If several comments exist on an item, it can disrupt the clarity of the wall. It could be nice if older comments collapsed and only new comments are displayed in full.
- The current filter options were not useful, Lea suggest more filtering options.
- It is not all comments that are relevant for the supervisor, and some comments are for the group only. An option to send the message to the supervisor would resolve this problem.
- The structure is very clear and gives a good overview.

Blackboard

- It is currently not possible to assign a name to a blackboard.

Other

- The overall layout seems simple and easy to use.
- The meeting feature is very attractive and Leas thinks that her group will use it often.
- They use SVN for writing the report, it could be intergrated with moodle in some way.
- Lea explains that group rooms should be private for the group.

Comments from the observers

- Meetings: It would be nice if it was possible to see what other activities that is arranged on a given date. This is relevant when creating a new meeting.(Henrik Koch)
- General: The name of the buttons are not consistent, some are labeled " submit" and others "save". (Kim Ahlstrøm)
- If the wall only contains a few elements, it displays the next page button even though only one page is available. (Kim Ahlstrøm)
- Lea was not able to edit group settings because she was not admin. It is a problem that admin rights are required to edit a group. (Kim Ahlstrøm)

- It would be nice with a small calender view in one of the sitebars (Anders Eiler)

A.8 Mathilde Gammelgaard

The documentation of this interview is written by the Project Planning group.

Date: 09-05-12 Participants: Mathilde Gammelgaard , Anders Eiler, Kim Jakobsen
Interviewer: Anders Eiler

Mathilde is given a complete walkthrough of our system. A casual conversation is conducted about the different elements. The following is a summary of the conversation between Mathilde and Eiler:

General

- Not very user friendly.

Members

- Considering that the group members know each other, it takes up a lot of space.

Meetings

- Data is not danish standard.
- Standard meeting should not include supervisor.

Timeline

- It is a good thing that it is dynamic.
- Four months makes the horizontal timeline chaotic.
- Good for overview.

Tasks

- When entering the name of the title one must first delete the “Task title” text.
- A color should be uniquely attached to a type.
- If a huge amount of tasks exit, a view with tasks of the day is wanted.
- A task should contain subtasks.
- In task overview there need to be added a “assigned to” column.

Wall

- It is important that the supervisor is not able to see all messages and files
- It should be an active task to post something to the supervisor
- There is to much emphasizes on filters and item information on the wall. It is disturbing.
- It is a good thing that it is similar to Facebook, it makes it easy to learn.
- Less spaceing between text
- When handling multiple documents, an explorer like appearance would be appreciated.

Blackbaard

NOTE: Did not work, but we talked about it.

- Not very usable.
- Add arrow functionality.
- Add equation options.

Comments from the observers

- Task: It is possible to assign task to the supervisor, seems odd. –Kjakob09 09:19, 9 May 2012 (UTC)
- Task: Assign task to members form is not consistent with the rest of the system. (Eiler)
- Task: Overdue tasks should be marked with an ! (Eiler)

A.9 Lene Winther Even Demo Meeting Sprint 4

Group members present: Alex Bondo Andersen, Kim Ahlstrøm Jakobsen, Mikael Midtgaard, and Rasmus Veiergang Prentow
Administrative personnel present: Lene Winther Even and Camilla Gæraa Larsen

This demo meeting was conducted on May 9th in the group room at Cassiopeia at Aalborg University. Along with Lene was her office trainee Camilla Gæraa Larsen. Lene used the system and gave feedback, while Camilla observed and gave her thoughts on the system.

The following is the relevant parts of the demo meeting.

Using The System

On the page to add a new project group Lene used the filter function to search for users to add to the group. She successfully found and added the first user. When attempting to add an additional user to the group she added another filter. This resulted in no users matching both filter. However, she quickly discovered the mistake and removed the first filter, which showed the user she was looking for.

When using the list of the project groups Lene expected to be redirected to the group page when pressing on a group link. Instead she was redirected to a list of group members. From the list she added another member to a group without any problems.

On a project group page Lene used the functionality developed by the Supervisor Communicationgroup. She uploaded a file and commented on the problem, and was satisfied with the functionality of it. She did, however, think that the user interface needed to be improved.

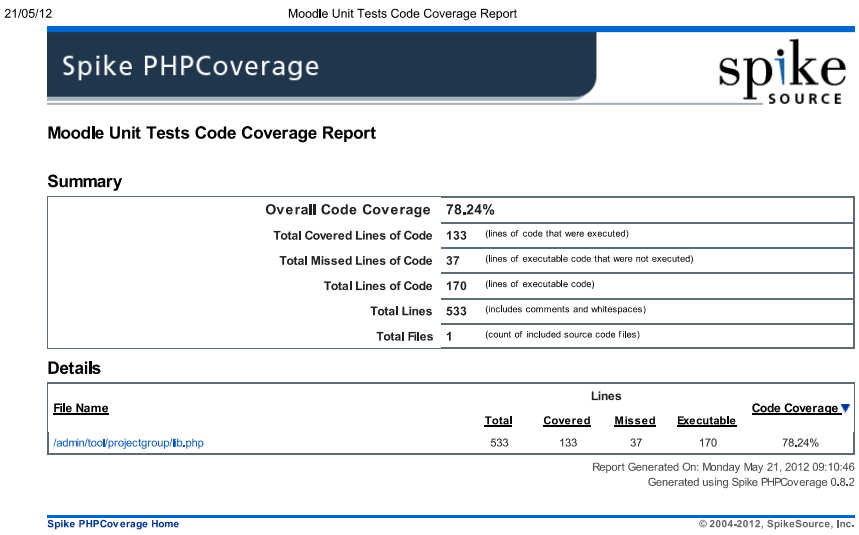
Additional Comments

- It would be preferred to have central database that contained information on all the groups of Aalborg University. Additionally, people should be able to create groups for different purposes.
- The fullname attribute is not used in practice.
- There are a number of problems with having several different Moodle systems at Aalborg University insted of having just one.
- Camilla thinks that overall the system we developed seems to be working well.
- There should be a way to remove the old filters in the same action as adding a new filter.
- The list of project groups should have a column displaying the supervisor.
- It should be possible to sort the list by the different columns - both ascending and descending order.
- In the future the system could be expanded to show how many hours supervisors use on their groups, so it could be used for hour calcutions, which is a task that the administrative personnel at Aalborg University do.

Appendix B

Code Coverage

B.1 Admin Tool



21/05/12 Moodle Unit Tests Code Coverage Report

Spike PHPCoverage

spike SOURCE

Moodle Unit Tests Code Coverage Report

Summary

Overall Code Coverage	**78.24%**	
Total Covered Lines of Code	**133**	(lines of code that were executed)
Total Missed Lines of Code	**37**	(lines of executable code that were not executed)
Total Lines of Code	**170**	(lines of executable code)
Total Lines	**533**	(includes comments and whitespaces)
Total Files	**1**	(count of included source code files)

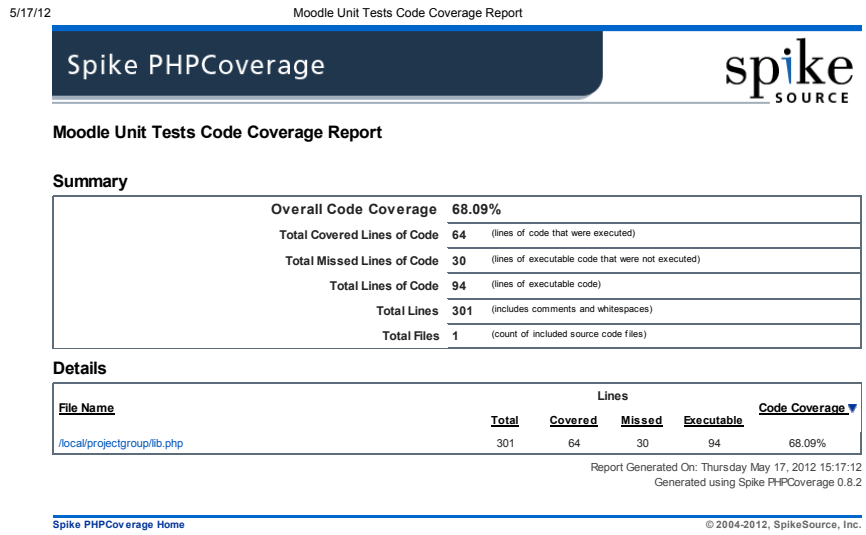
Details

File Name	Lines				Code Coverage
	Total	Covered	Missed	Executable	
/admin/tool/projectgroup/lib.php	533	133	37	170	78.24%

Report Generated On: Monday May 21, 2012 09:10:46
Generated using Spike PHPCoverage 0.8.2

Spike PHPCoverage Home © 2004-2012, SpikeSource, Inc.

B.2 Project Group



5/17/12 Moodle Unit Tests Code Coverage Report

Spike PHPCoverage

spike SOURCE

Moodle Unit Tests Code Coverage Report

Summary

Overall Code Coverage	**68.09%**	
Total Covered Lines of Code	**64**	(lines of code that were executed)
Total Missed Lines of Code	**30**	(lines of executable code that were not executed)
Total Lines of Code	**94**	(lines of executable code)
Total Lines	**301**	(includes comments and whitespaces)
Total Files	**1**	(count of included source code files)

Details

File Name	Lines				Code Coverage
	Total	Covered	Missed	Executable	
/local/projectgroup/lib.php	301	64	30	94	68.09%

Report Generated On: Thursday May 17, 2012 15:17:12
Generated using Spike PHPCoverage 0.8.2

Spike PHPCoverage Home © 2004-2012, SpikeSource, Inc.

Appendix C

Source Code

The source code for our component of MyMoodle is found on the attached CD in the folder labeled Appendix C