

AALBORG UNIVERSITY
STUDENT REPORT
sw608f12

**Implementing the
Aalborg Problem Based Learning Model
in Moodle**

Theme:
Application Development

Authors:
Alex Bondo Andersen
Kim Ahlstrøm Jakobsen
Mikael Midtgaard
Rasmus Veiergang Prentow

Supervisor:
Kurt Nørmark

Title:

Implementing the Aalborg Problem
Based Learning Model in Moodle

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 Nørmark

Page count: 107

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 project groups and their associated 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 February 1, 2012 to June 4, 2012 by the software student group sw608f12. The project is the outcome of the 6th 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 throughout the report the notation of three dots (...) can be seen. This notation illustrates that 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 two and three is a header, which very briefly describes what the given chapter contains and why it is important to the project.

In the report the notation `< /directory/file.ext >` is used to specify a file in the source code. The path is relative in relation to the base directory of the Moodle installation.

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 83. A copy of the code can be found on the attached CD. We refer to the source code as Appendix C.

The product is hosted on a computer by 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 our interviews and demo meetings, which was a tremendous help.

Contents

Preface	I
I Inter Group Work	1
1 Introduction	2
1.1 E-Learning	2
1.1.1 Learning Management Systems	3
1.2 The Aalborg PBL Model	3
1.3 Moodle	4
1.4 Problem Definition	5
2 Related Work	6
2.1 Learning Management Systems	6
2.1.1 SharePointLMS	6
2.1.2 Litmos	6
2.1.3 Mahara	7
2.1.4 Comparison	7
2.2 Relevant Student Reports	7
3 Initial Requirements	9
3.1 Discussion with Expert Users	9
3.1.1 ELSA	9
3.1.2 MPBL	10
4 System Definition	12
4.1 Decomposing MyMoodle	12
4.1.1 Project Group Management	13
4.1.2 Project Planning	13
4.1.3 Intra-group Collaboration	13
4.1.4 Supervisor Communication	13
5 Methods & Tools	14
5.1 Development Method	14
5.1.1 Considered Methods	14
5.1.2 Choosing a Development Method	16
5.1.3 Refining Scrum	19
5.2 Tools	19
5.2.1 Version Control	19

5.2.2	Bug Tracking	19
5.2.3	Code Documentation	20
5.2.4	Testing	20
II	Development	21
6	Introduction	22
7	Development Practices	23
7.1	Utilizing the Development Method	23
7.2	Inter Group Development Method	24
7.3	Additional Practices	24
8	Project Scope	26
8.1	End Users	26
8.1.1	Users of the Virtual Group Room	28
8.1.2	Managers of Project Groups	28
8.2	Requirements	29
8.2.1	Requirements Selection	31
8.2.2	Non-functional Requirements	32
8.3	System Definition	32
9	Platform	33
9.1	Constraints	33
9.2	Courses	33
9.3	Groups	33
9.4	Plugins	34
9.5	Framework	35
9.5.1	Database Access Layer	35
9.5.2	Context System	36
10	Analysis & Design	38
10.1	Virtual Meeting Place	38
10.1.1	Structure of the Virtual Meeting Place	38
10.1.2	Assignment of Virtual Meeting Places	40
10.1.3	Division of Projects and Groups	40
10.1.4	Virtual Project Tools Integration	42
10.1.5	Navigation to Virtual Group Room	43
10.1.6	Presentation of Members	43
10.1.7	Combining the Aspects	43
10.2	Project Group Management	44
10.2.1	Automatic Management	44
10.2.2	Student Management	45
10.2.3	Administrative Personnel Management	45
10.2.4	Choosing an Approach	45
10.3	Architecture	46
10.4	Database	48
10.4.1	Project Group Relation	48
10.4.2	Project Group Members Relation	49

11 System Presentation	52
11.1 Virtual Group Room	52
11.2 Administration	54
11.2.1 List of Project Groups	54
11.2.2 Project Group Members	55
11.2.3 Add & Edit	56
12 Implementation	58
12.1 Project Group Library	58
12.1.1 Context of Project Groups	59
12.1.2 Ensuring Permissions	59
12.2 Virtual Group Room	60
12.2.1 Blocks	60
12.2.2 Navigation	61
12.3 Managing Project Groups	61
12.3.1 Add & Edit	62
12.3.2 Lisiting	63
13 Test	64
13.1 Strategy	64
13.2 Test Implementation	65
13.2.1 Test Case Examples	66
13.3 Results	68
III Evaluation	69
14 Development Approach	70
14.1 Inter Group Collaboration	70
14.1.1 Initial Period	70
14.1.2 Choosing a Development Method	70
14.1.3 Sprints	70
14.1.4 Meetings	71
14.1.5 Group Room Location	71
14.1.6 Tools	71
14.1.7 Summary	72
14.2 Intra Group Development	72
14.2.1 Scrum Roles	72
14.2.2 Management of Requirements	73
14.2.3 End User Representatives	73
15 Product	74
15.1 MyMoodle	74
15.1.1 Decomposition	74
15.1.2 Architecture	75
15.1.3 Database	75
15.2 Project Group Management	75
15.2.1 Project Group Structure	75
15.2.2 Virtual Group Room	76
15.2.3 Navigation	77

15.2.4 Administration User Interface	77
15.2.5 Management	77
15.2.6 Testing	78
16 Conclusion	79
16.1 Multi-Project	79
16.2 Sub-Project	80
17 Future Work	81
17.1 Physical Group Rooms	81
17.2 Central Project Group Database	81
17.3 Virtual Meeting Place Templates	82
Bibliography	83
List of Figures	88
List of Code Snippets	90
 Appendix	 91
A Interviews & Demo Meetings	92
A.1 Thomas Ryberg Interview	92
A.2 Lene Winther Even Interview	93
A.3 Jette Due Nielsen & Pia Knudsen Interview	95
A.4 Morten Mathiasen Andersen Interview	96
A.5 Lene Winther Even Demo Meeting Sprint 3	97
A.6 Mikael Møller Hansen Interview	98
A.7 Lea Gustavson Demo Meeting Sprint 4	99
A.8 Mathilde Gammelgaard Demo Meeting Sprint 4	100
A.9 Lene Winther Even Demo Meeting Sprint 4	102
B Code Coverage	104
B.1 Project Group	104
B.2 Admin Tool	105
C Source Code	106
D Login Information	107

Part I

Inter Group Work

Chapter 1

Introduction

Today, computers are heavily integrated with the educational system. 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.

Compared to the previous semesters, the bachelor project is differently structured, since the goal of this project is to collaborate on a multi-group project. At the beginning of the semester two larger development projects were proposed, each of which were then divided into sub-projects. This meant that every “sub-group” working on a larger project was supposed to work together towards creating a single system. The goal of the multi-project described in this report was to develop an extension for Moodle, which 14 students chose to assign themselves to. The large multi-project was then divided into smaller sub-projects consisting of three to four students. This part of the report is shared among every sub-group and contains the 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 classrooms. 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 courses taught exclusively online in the Master of Problem Based Learning (MPBL) education [14], and regular courses that make use of online quizzes and more as a method of teaching.

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 that administrates, tracks, and reports on teaching. It is considered robust if it contains the following functionality [38]:

- 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 suited to a specific target group. The general characteristics of an LMS are [48]:

- 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 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) [12] is currently the primary e-learning platform at Aalborg University (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. Unfortunately, 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 AAU's problem based learning model. It originates from the philosophy of the university's staff. 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 [31] with supplementary literature from [50, 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 staff at the university is responsible for teaching the students to connect their project work to broader theoretical knowledge via the curriculum.
- **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 creating 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+) [8][60]. 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. The version of Moodle that we have decided to use is Moodle version 2.2.

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 [37]. Topics are an integrated part of courses, and resources can be links to external sources or uploaded files. 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 course page to 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.

My courses

- Computability and Complexity(CC)
- Database Systems (DAT6, SW6, DE8, MI8, SSE8)
- Design, Implementation and Evaluation of User Interfaces (DIEB)
- Multi Project Management (SW6)
- News Studyboard of Computer Science
- Professional Communication in Computer Science and Theory of Science (DAT6, SW6, DE8, MI8, SSE8)
- Programming Paradigms (PP)
- Real-Time Software (TSW)
- Software 5
- Software 6
- Software Engineering (DAT6, SW6, DE8, MI8, SSE8)
- Study Board CS, Exam, Aal
- Test and Verification of Software (SW6, GL.DAT8)
- All courses ...

People

Participants

Administration

Test and Verification of Software

Teacher: Ullrik Nyman

Description

Software is becoming increasingly complex and there is a growing awareness within software engineering practice that both formal verification techniques as well as testing techniques are needed in order to deal with this growing complexity. This is in particular true in areas such as embedded systems used in consumer electronics, time dependent systems occurring in safety critical systems and communication protocols from the telecommunication industry. The focus of this course is on techniques and software-tools that can be used to assess the quality and correctness of software systems. The (logical) first part of the course will focus on test and testing techniques. The (logical) second part of the course will focus on verification with a particular focus on real-time verification using UPPAAL. These two parts might be slightly interleaved.

Sidste års kursus: <https://intranet.cs.aau.dk/education/courses/2011/tov/>

Purpose, content and evaluation

Literature

Participants: SW6, DAT8old

Number of Students: SW6 (34), DAT8old (7)

Study Secretaries: Lene Even (SW6), Ulla Øland (DAT8old)

Semester coordinator: Kurt Nørmark (SW6), Jeremy Rose (DAT8old)

News forum

General Course Materials

Miniprojekt

Calendar

May 2012

Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	2	3	4	5	6	
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

Events Key

Global Course

Group User

Upcoming Events

Course: SICT moodle - Note: 2.

pinseidag / Whit Monday - Time: 09:00 - 16:30

Monday, 28 May, 09:00 AM - 04:30 PM

Go to calendar...

New Event...

Help me!

Missing course (teach./secc.)

Editing rights (teach./secc.)

Unsubscribe a course (stud.)

Missing course (stud.)

Course structure

View all help subjects

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

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 support the concept of project groups. To accommodate for this, students must use other 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 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

In this chapter existing LMSs will be examined to see how they support the Aalborg PBL model to gain inspiration for how it could be implemented in Moodle. Furthermore, we will look at student reports from the previous year in order to draw on their experiences with multi-projects, since they worked with a similar topic.

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 [15] 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 that 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 a clear distinction exists between courses and projects.

2.1.2 Litmos

Litmos [10] is a lightweight LMS with focus on being easy to set up and use. 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 [11] 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 members of communities within their area of interest. However, it does not come with a built-in calendar, which makes planning of projects an issue when using only Mahara. Mahara compensates for this by providing a sign-on bridge to Moodle, allowing users to access their Moodle accounts directly from Mahara without having to sign 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 had 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 to develop a solution which fulfilled the needs of teachers and students in relation to courses, whereas our project focuses on developing a solution to better facilitate PBL.

Despite the differences between our project and theirs, we concluded that it would be beneficial for us to examine their development methods and try

to learn from their experiences, since 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* [34], *E-Learning Management System: Implementing Customizable Schedules* [32] and *E-LMS - Administration, Calendar, Model, Education, Courses* [58].

The following is a list of relevant and interesting observations 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 used a traditional method. Integration testing became a problem as the groups were not synchronized and were always in different 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 took place between the groups.
- 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.

Even though we cannot 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 them.

Chapter 3

Initial Requirements

To determine how to enable Moodle to support the Aalborg PBL model we conducted two informal discussions with E-L ringssamarbejdet ved Aalborg Universitet (ELSA) [5] and the Master in Problem Based Learning (MPBL) department [14] at Aalborg University. This section accounts for the results of the discussion. Based on the results 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 which 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 [36], Marie Glasemann [41], Mads Peter Bach [30], and us
- Date: February 6, 2012

ELSA is responsible for the technical, organizational and pedagogical support of Moodle at Aalborg University. [5] However, the actual development on Moodle is outsourced to the different IT departments of the university. ELSA receives all feature requests and is responsible for providing support to the departments of the university. Because of this, 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, thus we have chosen five of the subjects ELSA proposed that we deem 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 filled with content, which requires manual work. This is primarily done by the administrative personnel and the lecturers. ELSA would like this

to 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 feature in Moodle that renders a more simple overview, which eases the task of finding the wanted item. We need to ensure that similar problems do 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 features to be more general such that they could be used in different contexts. For example, this could be the ability to share quiz questions between courses.

3.1.1.5 Archiving

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

3.1.2 MPBL

- Type: Informal meeting
- Participants: Jette Egelund Holgaard [47], Morten Mathiasen Andersen [29], and us
- Date: 08-02-12

The MPBL department defines itself 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)”. [14] 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 contexts we conducted a meeting and discussed how Moodle can be improved to better support PBL. We chose 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 [17] and Adobe Connect [1] are used. MPBL would like the necessary tools to be available in Moodle to avoid having to use several external tools.

3.1.2.2 Nearness

When working as a group it is advantageous to be in the same room. When that is impossible, which is often the case for MPBL, tools must create the feeling of nearness. A virtual meeting place 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. A need exists in Moodle to better handle sharing of documents. The ability to comment on documents is a common way to give feedback on the work of others. Currently, document sharing is conducted via emails, Dropbox [4], and Google Docs [7].

3.1.2.4 Planning

When group members are geographically separated it is essential to coordinate and plan. MPBL would like Moodle to contain a tool for planning 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 for Moodle that allows Moodle to support the Aalborg PBL model.

Since MyMoodle is an extension it adds functionality without changing existing functionality.

4.1 Decomposing MyMoodle

Since the project is large it is decomposed into sub-projects. This is due to the educational context in which the project is being conducted. The educational context requires us to work as one large group divided into four sub-groups.

Each sub-group will have its own dedicated project, of which the goal is to make a part of MyMoodle. The parts from each sub-group will be integrated with each other, and the development process and results of each part will be documented in the corresponding sub-group's report [21].

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 core principles we find relevant to this project are:

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

These principles are important for MyMoodle, and must be considered in each sub-system. However, each sub-system prioritizes the core principles differently. Notice that two of the core principles are not considered. The core principles, 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, collaboration between the members, and 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 each other. 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 Collaboration

During a project, the participants must be able to collaborate regardless of geographic location. The collaboration must be efficient such that deadlines are met and progress is made. 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 with, and receive feedback from, their supervisor. This sub-system should accommodate the MBPL departments requests for collaboration and planning described in 3.1.1

Chapter 5

Methods & Tools

In this chapter we describe the development methods and tools that are considered for this project. A choice is made for tools and development methods along with reasons for the choices.

5.1 Development Method

For the collaboration between the four groups 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 each used different development methods, which caused problems, see Section 2.2. In this section different development methods are presented and one is chosen for this project.

5.1.1 Considered Methods

We consider a selection of traditional and agile development methods. In general, traditional methods follow a schedule planned upfront, where every task is handled as a single unit and the result is not changed afterwards [56, sec. 2.7]. This method is inspired by the methods used in the construction industry.

Projects using agile methods are developed in iterations, where some part of the product is developed in every iteration [51, p. 25]. This should help with adapting to changes that the end-users or customers might pose during the development.

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

5.1.1.1 Extreme Programming

Extreme Programming (XP) is an agile development method that consists of 12 recommended core practices [51, 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 [51, p. 156-157].

There are roles assigned to different people involved in the project [51, p. 145]. These roles are: customer, programmer, tester, coach, tracker, and consultant. All these roles are important, but Larman, the author of [51], stresses that an on-site customer or at least an on-site customer proxy is needed [51, p. 152-156].

When using XP, the development is done in iterations lasting 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 the “Iteration planning game”.

5.1.1.2 Scrum

This section describes the agile development method Scrum, and is based on [51, 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. The sprint backlog cannot be changed during the sprint. 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 customer’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 [42, pp. 23-30][51, 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 for each implementation. One version of the waterfall model defined at [54] 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 whether their implementation is feasible 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 in their pure form they do not allow to move back to a phase once the next is started. This suggests a big and heavy upfront design plan that is to be followed until the project is finished.

Of the methods presented, Waterfall is the one that requires the most amount of 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 the students are divided into two major groups, one of which is working with Moodle. At AAU, a group limit of four members is imposed during the sixth semester, therefore the Moodle project has been divided into four sub-groups.

2. **Diverse target group**

The relevant target groups are: 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, who are available to test the product and provide feedback.

4. **Hard deadline**

Because this is a semester project the project must be handed in when the semester ends.

5. **Pass on project**

This project will be passed on to the sixth semester students next year.

6. **Known framework and platform**

The Moodle platform is open source and 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 lectures beside the project, therefore the development time is limited.

9. **No manager/product owner**

Because this is a semester project, there is no project manager or product owner as 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, authors of [33], have identified five factors that can be used to determine whether to use a traditional or agile development method. 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 their methodological skills. The methodological skills are divided into five levels: -1, 1b, 1a, 2, and 3 [33, 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 [51, pp. 36-37].

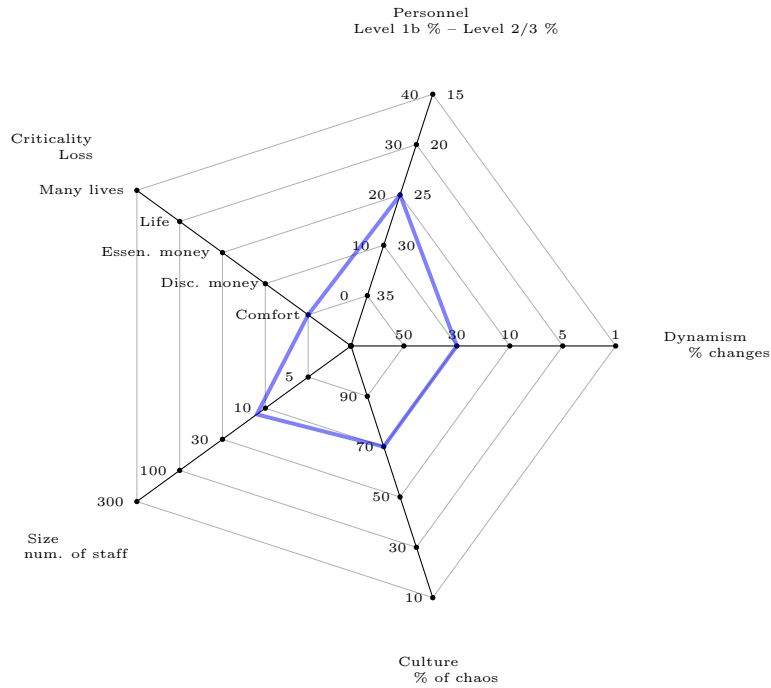


Figure 5.1: Radar chart showing the deciding factors for choosing the development method of our project.

A polar chart showing the factors for our project can be 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 upfront analysis and design difficult to perform. This lead us to choose an agile method due to the risk of “I know it when I see it” (IKIWISI). IKIWISI is when 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, that 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 in the previous section state, 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 [19] 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 item 7) so we will strive to allow every member to be Scrum master for some 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 described briefly in this section. The tasks that must be handled by tools are: version control, bug tracking, code documentation and testing.

5.2.1 Version Control

All of us have been using subversion (SVN) for version control in previous projects. SVN is a centralized solution [18] with a single repository the group members can update from and commit changes to. However, this project differs from previous projects with respect to the organization of groups; we are not one group of x individuals, but rather one group divided into sub-groups. This has lead us to choose a distributed solution rather than a centralized one.

The solutions considered are the distributed systems Git [6] and Mercurial (Hg) [9]. These systems are quite similar and the main difference is that Hg is simpler than Git and Git is more flexible than Hg [20]. 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 discover during our project. The most important requirement for the tool is that it should be able to track the bugs and make them easily available to the team that will continue this project next year. The tools considered for bug tracking are Bugzilla [35] and Eventum [25]. These tools are very similar in their features. When a bug is discovered a bug report must be added in the

tool. In the tool, the members of a team can see the bugs and their status, and mark them as fixed when appropriate.

Since we have used Bugzilla as part of the course Test and Verification, choosing Bugzilla will save us the overhead of having to learn a new tool. Furthermore, the development of Eventum seems to have been discontinued since the beginning of 2009 [24], which means that if defects exist they are 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 write documentation externally from the code. We plan to use a tool like PHPDocumentor [44] or PHPXref [59] to handle our documentation, since both of these read comments in the source code and use it for documentation. This gives us the ability to write code and documentation in the same file. The syntax both of these tools use is the same, which means that as long as the syntax is used either tool can be used to generate documentation.

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

5.2.4 Testing

We want the tests to remain available after our work on the project ends, since the groups that are to take over this project next year should be able to reuse our test cases. We will use the built-in testing framework of Moodle [13], which is based on SimpleTest [16]. We do not consider other tools, since it reduces the number of tools we have to use by using one which is already a 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

The foundation for this project has been set up in Part I. We now proceed to the part where the actual development is presented. The sub-system that we are developing is called Project Group Management and is briefly described in Section 4.1.

From this part on the report has four authors; “we” refers to those four authors.

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

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

We refer to the students working on the other sub-projects as our “peer-groups”.

As we are working with Project Group Management we are responsible for integrating the concept project group into Moodle. Recall from Section 1.2 that we define a project group to be a team of students working on a project.

When referring to a “block” we mean a specific plugin type in Moodle and not as a general concept.

The members of a project group will have access to shared tools through which they can conduct their project work. These tools include planning the progress of the project, communicating internally while the project is being conducted, and communicating with supervisors. These tools are created by our peer-groups (as mentioned in Section 4.1). Our responsibility is to make all these tools available to the members of the project groups. To make MyMoodle usable at any educational institution we provide a tool to manage project groups.

Chapter 7

Development Practices

An important part of this project is the collaboration between the peer-groups, which is guided through a development method. This chapter gives a description of how we are using the development method Scrum of Scrums in our sub-group, and which additional practices we have chosen to use.

We distinguish between a development method and a development practice. A development method is a collection of development practices that in combination should enhance development in general. A development practice is a predefined set of actions that are performed during development.

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 entire system should be structured and created. We solve this problem by using the development method Scrum and dividing the development process into a number of sprints.

Our first sprint consists of information gathering; we study the Moodle platform as well as interview our end users 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 a programming sprint.

Different sources suggest different notations for what is in a backlog [42, p. 17][51, pp. 123-124]. Our backlog items are all features. See Figure 7.1 that illustrates the relation between the backlogs.

The requirements we gather are used to create feature descriptions, which are used to fill the product backlog. At the beginning of each succeeding sprint 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.

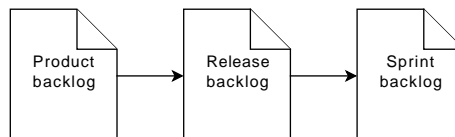


Figure 7.1: *Illustration of how backlog items move through the different backlogs.*

Before an item is moved to a sprint backlog it is moved to the release backlog. This means that the release backlog consists of all items implemented in the sprints of this semester.

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. If there is a significant difference between the estimates we discuss the scope of the feature and argue until an agreeable estimate is found. Additionally, we assign the items a priority. Based on the estimate and priority we choose which items we are able to implement in the current sprint.

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

7.2 Inter Group Development Method

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

At the sprint planning meeting 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 sub-group should accomplish in the given sprint.

At the end of each sprint the sub-groups meet and present what they have created during the sprint. End users can be invited to be showcased or try the system in demo meetings in order to acquire feedback between sprints.

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 physical group room to ask for help or request that some specific work should be done. Additionally we hold Scrum of Scrums meetings approximately twice a week. 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.

7.3 Additional Practices

Since we are continuously integrating – a Scrum practice – and are passing the project on to new developers next year (described in item 5 in Section 5.1.2), we want automated tests that can be run to ensure no regression in functionality. We have already decided that we are using the built-in test framework of Moodle in Section 5.2.4. To make sure that test cases are written and to avoid that these are created in the last minute we use test driven development (TDD) [51, pp. 292-294] as an additional development practice. TDD states that test cases must be written before the functionality to be tested is written. A functionality is not considered implemented until the test case(s) for it are written and pass.

These test cases can be run whenever a change is made to existing code to test whether the change broke something.

Chapter 8

Project Scope

In this chapter the problem of our sub-system is defined by examining the characteristics of the end users and by gathering requirements from them. From the requirements of the end users the sub-system is defined, which is important since it explains the choices made in the project.

8.1 End Users

Since we are using Scrum as our primary development practice we prefer to have a product owner [51, p. 115] who can speak on behalf of the end users. As mentioned in Section 5.1.2 we do not have a product owner, which is why we facilitate the contact and communication with the end users ourselves. In this section the end users for our sub-system are defined and the people that we use as representatives are presented shortly – we cannot speak with all potential end users because there are too many.

The end users of the entire system are very diverse. In our sub-system we are interested in two different categories of end users:

- Users of the virtual group rooms.
- Managers of the project groups.

The two categories of end users can overlap. When choosing representatives for our end users there are several properties that we want to cover. These properties are: Type, Faculty, Department, and LMS experience. The type of an end user is one of four values: Administrative personnel, technical personnel, supervisors, or 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 LMS experience as important factors to be diverse. We would like to have representatives that cover these two properties as much as possible.

The end users we are using come from Aalborg University. The reason for this is three-fold. Firstly we are implementing the Aalborg PBL model into 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 easier for us to find people to participate from this university compared to other universities. Thirdly the system is to be used at Aalborg University.

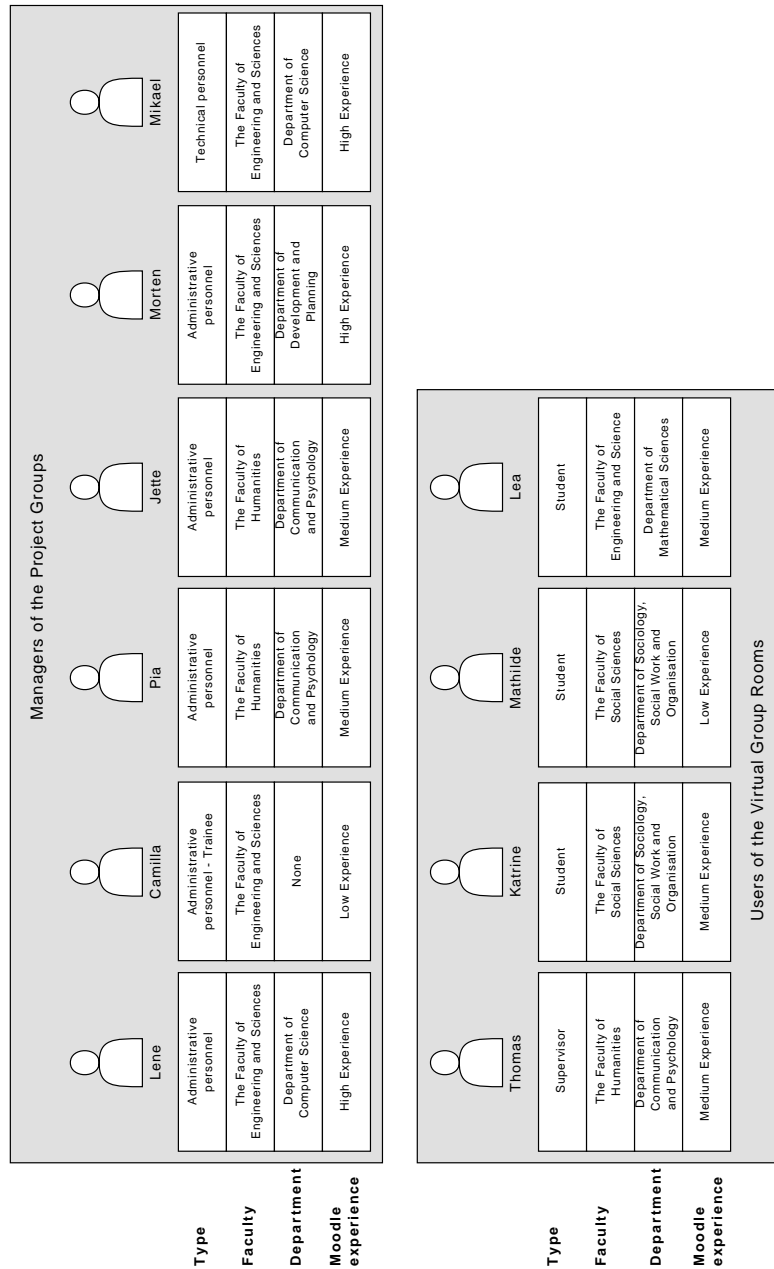


Figure 8.1: An overview of our end users. The gray boxes contain the two categories of end users. The upper box contains the members of the project groups, the lower box is the managers of the project groups.

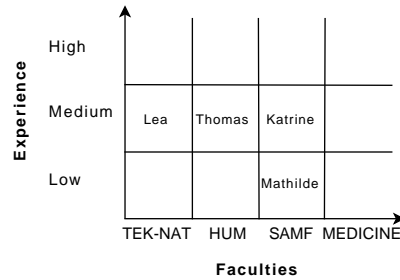


Figure 8.2: A graph showing how the end user representatives of the users of virtual group room are spread between faculties and how much experience they have with regard to LMSs.

In Section 8.1.1 and Section 8.1.2 we explore the two different categories of end users. An overview of the chosen end users can be seen in Figure 8.1. The coverage of properties is illustrated in Figure 8.2 and Figure 8.3.

8.1.1 Users of the Virtual Group Room

As discussed in Chapter 4 our responsibility is to create the concept of a project group and virtual group room. Also we want to create a tool to manage project groups. Since we are working together with three other peer-groups we receive requests for functionality from them. Our peer-groups have their own end users, both students and supervisors, where they get their requests from.

To obtain an understanding of the members of the project groups we consider two different approaches:

- Regarding students as a part of our end users and make our own field studies.
- Relying on the requests of our peer-groups and effectively use our peer-groups as end user representatives.

We choose a compromise and rely on the requests of our peer-groups and cooperate in the field studies related to the entire system and the concept of project groups. By doing so we receive feedback directly from end users, which helps us avoid misunderstandings.

The end users of the virtual group room are students and supervisors. We want to have different kinds of student representatives – 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 [57], who is supervisor under The Faculty of Humanities. In Figure 8.2 end user representatives of the virtual group room can be seen along with their LMS experience and faculty.

8.1.2 Managers of Project Groups

We choose the administrative personnel as our end users because they have a good knowledge of project groups and the related management tasks. From our

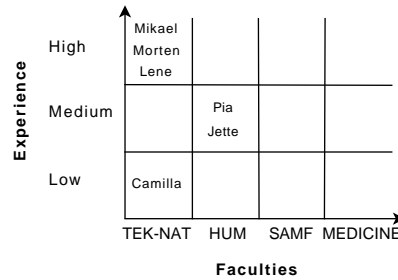


Figure 8.3: A graph showing how the end user representatives of the managers of project groups are spread between faculties and how much experience they have with regard to LMSs.

own faculty we have the representatives Lene Winther Even [39], Camilla Gæraa Larsen [52], and Mikael Møller Hansen [46]. Where Lene is a senior secretary, Camilla is an office trainee, and Mikael is an IT administrator. We have been in contact with Jette Due Nielsen [47] and Pia Knudsen [49] as our representatives from The Faculty of Social Sciences. We have also been in contact with Morten Mathiasen Andersen [29] from MPBL education and The Faculty of Engineering and Sciences. Figure 8.3 shows these representatives along with their faculty and LMS experience.

8.2 Requirements

During this project we have several meetings with end user representatives. We hold meetings before, during, and after the development of the system. We do this to gather the requirements of the end users based on the system under development. In this section we present our final requirements. The continuous change in our requirements is evaluated in Chapter 14 in Part III. The interviews that have been conducted to gather these requirements are found in Appendix A.

Since we use Scrum as our development method we use a product backlog to capture our requirements [51, p. 114]. Since this multi-project is to be proceeded by a new multi-project group next year, we allow our product backlog to be larger than we are able to complete in this semester. In this section we focus on the items that are part the project of this semester. 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. The following three paragraphs will describe how we derived some of our backlog items.

Virtual Meeting Place The concept of a virtual meeting place comes from the meeting with people from the MPBL education described in Section 3.1.2. It is supposed to be a virtual place in which members of a project group can meet and conduct project related work. Additionally, it must create a feeling of nearness as mentioned in our discussion with people from the MPBL education.

Find Project Groups For the managers of project groups there must be functionality available to find a specific project group or a set of specific project

Name	Description
Manage Project Groups	A subset of the end users should be able to manage the project groups, this includes creating, editing, and deleting project groups.
Virtual Meeting Place	Create a virtual space that is compliant with the principles of the Aalborg PBL model and create a feeling of nearness.
Introduce Roles	Add the possibility to assign roles to the members of a project group.
Find Project Groups	Produce functionality to find a given project group.
Navigate to Virtual Group Room	The users of project groups must be able to navigate to the their virtual group room.
Presentation of Members	It should be possible to see who the members of the group is, to create the feeling of nearness.

(a) *Items in the release backlog.*

Name	Description
Archiving Project Group	ELSA has requested that data archiving should be possible.
Recursive Project Groups	Allow project groups to contain other project groups. This also includes the administrative tools to manage the new structure of project groups.
Manage Roles	Create Functionality to define and manages roles. An example of a new role could be secondary supervisor.
Project Group Synchronization	Project groups should be added automatically by synchronizing the system with a central database containing all the project groups of the university.
Virtual Meeting Place Template	To account for the end users 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.

(b) *Items not in the release backlog.*

Figure 8.4: *Lists showing features from our product backlog.*

groups. This requirement is based on a demo meeting that we conducted with Lene (see Appendix A.5).

Manage Project Groups It should be possible to manage the project groups. We let us inspire by how courses are currently created at Aalborg University. ELSA is responsible for creating new courses at the start of every semester. This is done by the administrative personnel. Every department informs ELSA of which courses that should be created. ELSA creates the courses based on a general template. The administrative personnel and lecturers then populate the courses with data. Information about this process can be read in Appendix A.1, Appendix A.2, Appendix A.3, and Section 3.1.1. Similar to the creation of courses it should be possible to create and in other ways manage project groups. The design of project groups are explained in Chapter 10.

Not all backlog items are explained because we believe that the derivation of them are of similar nature. In the following section the choices of the release backlog are accounted for.

8.2.1 Requirements Selection

Recall the table seen in Figure 8.4. To explain the selection of requirements we present a short description of and an argumentation for why other requirements are excluded. The four items are described below.

Recursive Project Groups The idea of allowing recursive groups is that multi-projects such as ours are better structurally supported. In an interview with Lene (see Appendix A.5), we discussed the possibility of having a recursive project group structure. Lene could not imagine any concrete cases where such a structure would be beneficial in practice, which is why we felt that it is not a feature that will improve the system substantially.

Project Group Synchronization In the Department of Computer Science at Aalborg University AdmDB (Administration Database) is, among other things, used to manage project groups. During the interview with Mikael (see Appendix A.6) we learned that he is currently developing a new Application Programming Interface (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 elaborate further on this subject in Section 10.2.1.

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 places, 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.

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 explains that it is to preserve the content for future reference (see Appendix A.2). Morten (see Appendix A.4) explains that teachers use the archived versions of a course to create a new course of the same subject. Archiving is worth considering because it allows students and supervisors to 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.

8.2.2 Non-functional Requirements

To capture our non-functional requirements, also referred to as the “-ilities” [51, p.288], we rely on our end users’ experiences with the system. Common non-functional requirements include availability, robustness, effectiveness, efficiency, extensibility, and usability [53, sec. 9.1]. We primarily focus on robustness, extensibility, and usability. The reason for the two first is that new developers will take over the project next year. They may want to change something, which is why we must ensure that the software is robust. They probably want to extend our system with new features which calls for an extendable system. We believe usability is very important, since the system will potentially be used by many students with varying experience with LMSs. Availability is not considered since our system is an extension to another system and our availability is directly connected to that of the main system, Moodle.

8.3 System Definition

The selected requirements are: Managing Project Groups, Virtual Meeting Place, Introduce Roles, Find Project Groups, Navigate to Project Groups, and Project Group Members. Based on the definition of the entire system and the selected requirements we define the sub-system we are implementing as:

A sub-system of MyMoodle that implements project groups in Moodle and allows 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. This definition and the definition of the entire system is concluded upon in our conclusion in Chapter 16.

Chapter 9

Platform

In this chapter we present the constraints that Moodle sets for MyMoodle, the core concepts of Moodle, and the framework in which MyMoodle is integrated. This is important since it gives an understanding of the environment in which MyMoodle should function, and which features it can use from the underlying system.

9.1 Constraints

Moodle is written in the programming language PHP [61] and uses an SQL database. This restricts us to write MyMoodle in PHP as the server side scripting language. Since Moodle uses an SQL database system we choose to restrict ourselves to use that as well, to avoid adding unnecessary dependencies to third party systems.

9.2 Courses

Courses is 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 [66]. Every course in Moodle is part of a course category [65]. 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

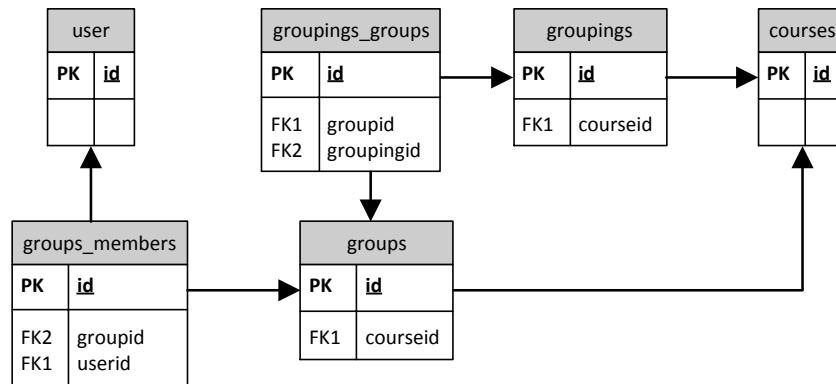


Figure 9.1: The database scheme for groups and groupings. Data fields that are neither part of a primary key nor foreign key are omitted for clarity [43].

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 [28]. The four most common plugins are explained below:

Blocks Moodle uses the term block to refer to a box that can contain a wide spectrum of information and functionality. A block can as per 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 [27] is a very common block, which as per default is displayed on all pages. It contain links such as My Profile and My Courses. Blocks are not limited to navigational purposes, but can also display data from the database.

Each block has settings regulating which pages it can be displayed on.

Activity modules “An activity is a general name for a group of features in a Moodle course.” [22] 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 [23], that contains available activities for the course.

Activity modules often contain a new page where an activity is available. An example of this is 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, which 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 Documentation defines the functionality of an admin tool as the following: “Provides utility scripts useful for admins to examine and modify a Moodle site”. [28] The only users that can access the admin tools are administrators. These are made available through the Site administration menu point in the Settings block [63].

Local plugins Local plugin is 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 is communication with an external system or for extending the navigation block.

9.5 Framework

Moodle includes various APIs [64], which aid the developer in the process of creating an extension. Some of the APIs are:

- Data Manipulation Language (DML) API [69] for all persistent data access and is described further in Section 9.5.1.
- Form API [71] for rendering HTML form objects and handle the validation of the data sent through the form.
- Output API [72] for general HTML rendering.
- Page API [73] for setup of a standalone page including methods for adding JavaScript and configuring display options.
- Access API [62] for controlling access levels of users throughout the Moodle installation.
- Unit test API [75] for testing components in Moodle.

Moodle APIs are used in different ways. The DML API, Page API, and Output API are used through global variables, whilst the others are used through class extensions. In Code Snippet 9.1 usage of Page API and Form API is shown.

```
1 global $PAGE;
2 $PAGE->set_context($somecontext);
3
4 class some_form extends moodleform {
5     ...
6 }
```

Code Snippet 9.1: *Example of the Page API and Form API in Moodle.*

9.5.1 Database Access 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 methods for updating, inserting, deleting, and selecting. The basic methods for selecting and

inserting are `get_record` and `insert_record`. These either accept an argument of type `stdClass` or return one generated from data in the database. `stdClass` is a standard PHP class which is empty, and allows for arbitrary addition of properties at runtime like any other PHP object.

This gives a unified procedure for data manipulation. A simple update procedure is seen in Code Snippet 9.2. For simple operations it is not necessary to write any SQL queries, but for more complex queries, such as queries using joins, SQL statements must be written directly.

```

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 a user.*

9.5.2 Context System

The context system in Moodle is used to set the context of a given page to determine the capabilities of the user, who is currently logged in, and which blocks and activity modules to present [74].

When a user is logged in to a Moodle system he can have various roles in various contexts. One user may be a student in one course and a teacher in others. The context system is structured hierarchically. The hierarchical layout of the context system can be seen in Figure 9.2 [70]. The highest level of context is the system context, which all other contexts inherit from.

Every webpage in Moodle 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 {
4     print_error('nocontext');
5 }
6 ...

```

Code Snippet 9.3: *Code snippet from `< /course/view.php >` showing how the context of a course is set.*

The function `get_context_instance` requires a context level and in this case also a course id. The context level used in the function call is a constant telling which context level that needs to be loaded. This is a numeric value. In this example an error is outputted if the context fails to load. This happens if the context level provided is illegal or in this instance if the course id is invalid. It is required to set the context for each page. In some pages Moodle sets the context itself [73].

If a user adds an instance of a block and the context is set to the system context, the block instance will be shown on all course pages. If the context is

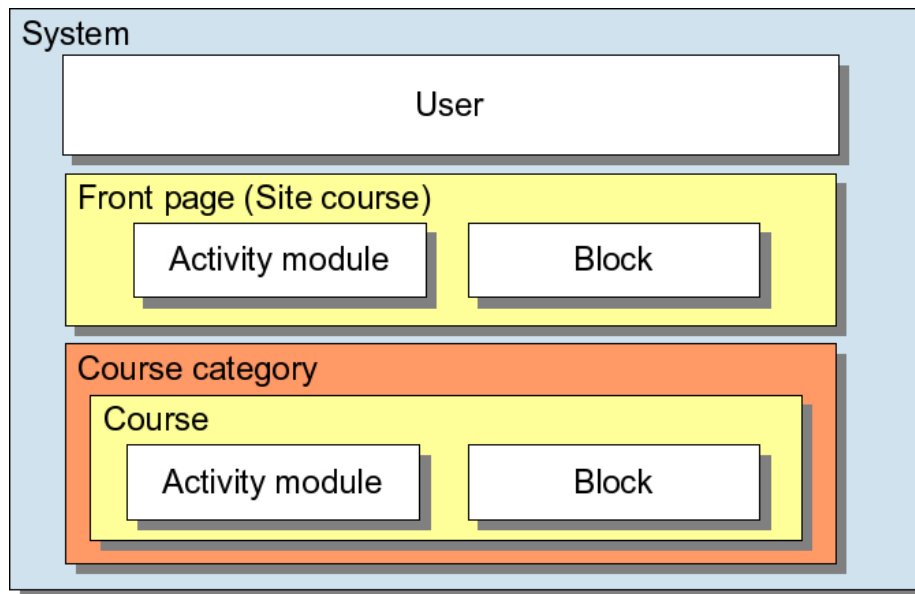


Figure 9.2: *The hierarchical context structure of Moodle. Picture source: [70].*

set to the course context the block instance will only be presented on the page of the course.

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 in Section 10.1 and Section 10.2. The complete architecture of MyMoodle is presented in Section 10.3, and in Section 10.4 the database design of our sub-system is presented. 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 sub-system. Analysis and design is presented together for simplicity.

10.1 Virtual Meeting Place

In this section we analyze the requirement 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 analyze our requirements based on the conducted interviews and demo meetings and our own experiences.

We refer to the tools that our peer-groups develop as virtual project tools. The aspects of the requirement Virtual Meeting Place that are presented here are: Structure of virtual meeting place, assignment of virtual meeting place, division of projects and groups, and virtual project tools integration. These are presented below.

10.1.1 Structure of the Virtual Meeting Place

The members of a project group should have a place where they can meet and engage in project work using 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. We have to decide the structure of the virtual meeting place. We consider two different approaches, which are described below.

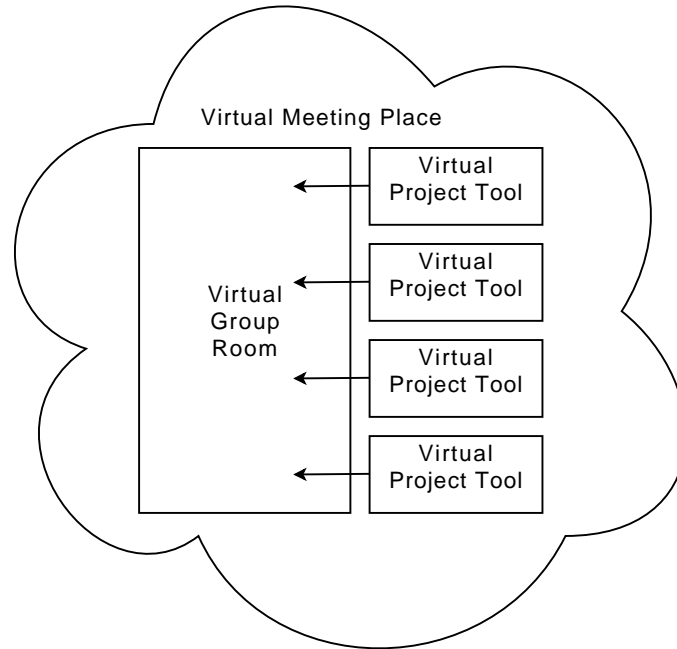


Figure 10.1: *The connection between the virtual meeting place and the virtual group room.*

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 requires less effort than the Team Group idea to implement since no permissions need to be considered. However, the Team Group is closer to the way that the Aalborg PBL model should work in practice.

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 a lot to find the relevant information 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 tools and possibly adding new relevant tools. The virtual place where these tools can be found is called the “virtual group room”. A virtual meeting place for a project group thereby consists of the virtual group room and the tools available. This is illustrated on Figure 10.1.

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

10.1.2 Assignment of Virtual Meeting Places

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

- **Never** There are no physical 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 physical group rooms on a daily basis (as stated in Appendix A.3).
- **Half-yearly** Physical group rooms are assigned to each project group at the start of each semester.

The virtual meeting place should be a hub where tools central to group work are available. It should be similar to physical group rooms at Aalborg University in usage, and be available on a half-yearly basis.

10.1.3 Division of Projects and Groups

So far we have regarded project groups as an atomic entity, but projects 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 a single entity, where the group and project are one. Below are the properties of the two variants presented.

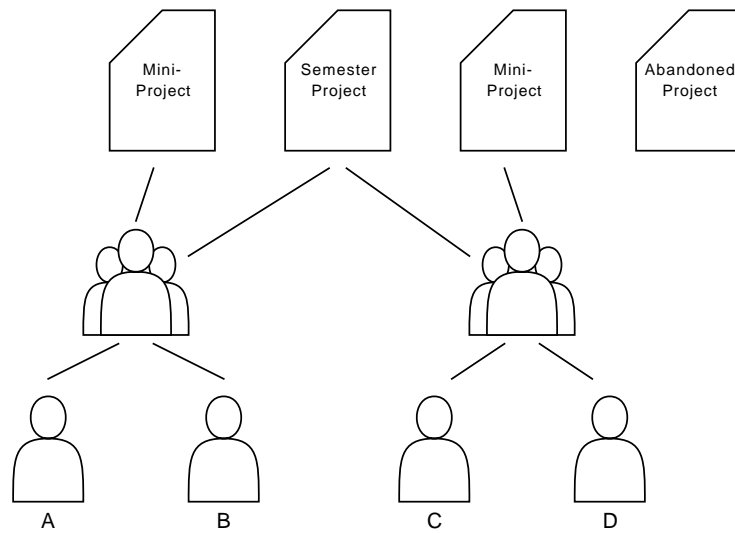
Groups and Projects Divided

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

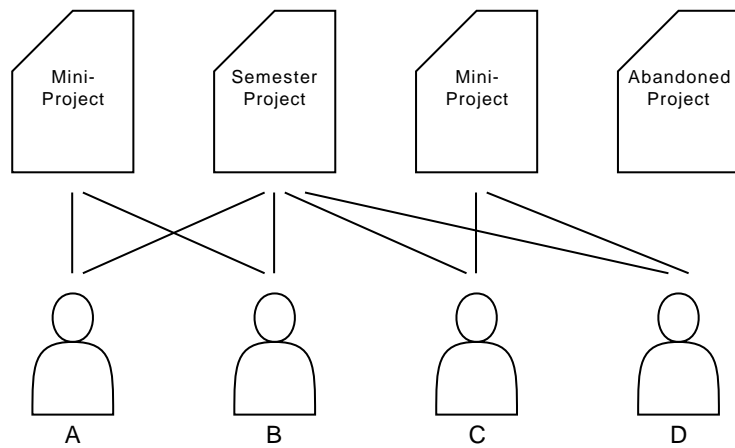
Groups and Projects United

- A group is linked to a single project; they are inseparable.
- Simple to implement.

The differences between these variants are illustrated with an example in Figure 10.2. 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 the 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.



(a) *Divided.*



(b) *United.*

Figure 10.2: *The difference between the two approaches to the division of project and group illustrated by an example.*

Intuitively the option to divide groups and projects (seen in Figure 10.2a) might seem more appropriate since it introduces a level of indirection. However, our interview with Lene (seen in Appendix A.2) 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 entity (seen in Figure 10.2b) in accordance with the needs of our end users and to simplify our system. The database design for storing project groups and members of the project group is found in Section 10.4.

10.1.4 Virtual Project Tools Integration

As mentioned previously we are developing a virtual group room where different virtual project tools can be used to perform project work. 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 virtual project tool could be a request to make the tool render itself. Alternatively we could integrate every tool directly into the virtual group room. This, however, is not extensible enough. Recall that we are four peer-groups of three to four students working together, which means that we would likely “step 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 a tool. This should be much easier if each tool is encapsulated in a component with a common interface than if every tool were intertwined in a single component.

Now that we have established that there is a need for an interface between the 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: Using an existing interface or developing a new interface. These are not mutually exclusive. We may take an existing interface and modify it to fit our needs. Since we are making a plugin for an established LMS, we need to 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: 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 is a joint decision.

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 [26]. We do not use 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 blocks or develop a new interface. Developing a new interface would give us more freedom because we may choose exactly how it is defined. On the other hand a block ensures that MyMoodle will have a similar look and feel as the rest of Moodle. Additionally, the students that take over next year will have accessible documentation through the Moodle Documentation about how to modify virtual project tools or add new ones. The

advantages of blocks outweigh the additional freedom that we may gain from defining our own interface. This leads us to choose to use block plugin type that every virtual project tool must be implemented as. For the virtual group room to communicate with virtual project tools it can use calls to member methods of blocks.

10.1.5 Navigation to Virtual Group Room

For a user to navigate to a virtual group room that he is a member of, there should be a 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 project groups we show a link to a page that has a list of all the project groups that the user is a member of.

10.1.6 Presentation of Members

Recall that the Presentation of Members requirement must ensure nearness for project group members. If members of 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. The Supervisor Communication team has requested a feature for supervisors to see photographs of 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 virtual group room should show all the members of the project group belonging to the room. There should be a photography 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

In this section we combine the aspects discussed in the previous sections. First and foremost there is a project group room where virtual project tools can be used directly or alternatively linked to. Secondly there are all the virtual project tools that are not located directly in the virtual group room. Only students and supervisor(s) associated with a project group are allowed to access the virtual group room of the given project group. An exception to this is the

administrative personnel, who have privileges to access and change any virtual group room.

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

Each virtual project tool related to a virtual group room is implemented as a 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 virtual group rooms, we extend the existing navigation menu of Moodle. To avoid expanding the navigation menu 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 simple tool 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 manage the project groups: Automatic management through external application, manual management by administrative personnel, and management by students. We will now examine the approaches and choose the one we feel is the best solution.

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 project 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 project groups are changed. If the data is input 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 Mikael of IST (see Appendix A.6) we discovered that Aalborg University does not have a single system containing the project groups. 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 at Aalborg University. Furthermore, some departments do not keep official records of the project groups. In an interview Mikael explained that not all the systems have an API from which the groups can be extracted, and he 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 project groups is discussed.

In some cases the administrative personnel do not know which project groups are created (see Appendix A.6), and it is not guaranteed that all students want an online project group in Moodle. Therefore, a solution is to allow students to manage, by allowing them 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 project groups or misuse the system by vandalizing the project groups of other students. This approach must include various security constraints to ensure proper use. Students should not be able to delete project groups they are not members of or to delete project groups by mistake. A mistaken deletion of a project 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 project group must accept new members. Alternatively they should not be able to add or remove members from a project group after creation since a student should not be able to join a project group they do not belong to.

10.2.3 Administrative Personnel Management

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

The administrative personnel plays a central role in the administration of the courses and project groups of Aalborg University 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 to 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 project group database at Aalborg University, and the decentralized system at our department, AdmDB, is in the process of getting a new API developed. Therefore the current API is going to be deprecated in the near future. If a centralized system with an API to extract project group information existed this would be the ideal approach, since it will not add workload to the administrative personnel and all project groups will exist in Moodle.

From the two remaining approaches we choose to have the administrative personnel manage the project groups. We choose this because the student management approach has several constraints, which must be taken into account. 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 still 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 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 project groups. This list can potentially grow very large, so there needs to be a way of finding a specific project group or a specific set of project groups as specified by the 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 project groups. The filter attributes should be those that administrative users are likely to use. During the demo meeting described in Appendix A.5 we experienced that our end users need to find a 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 we 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 plugins. The architecture does not specify how each plugin should be created, but specifies a general structure of the components of the system. The complete architecture can be seen in Figure 10.3.

The architecture consists of a total of five layers. The three uppermost constitute MyMoodle, they have a common dependency, namely the Moodle platform. Layers four and five are Moodle and the Database system respectively.

We explain the three uppermost layers as:

1. The uppermost layer is the virtual group room and the administration tool. The virtual group room is described in Section 10.1. The administrative tool is 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 uppermost layer is the middle layer, which consists of the four components: Timeline, Task, Supervisor, and Blackboard. These four components are created by our peer-groups and are not explained further. We call this layer “Content Layer”, since the components in

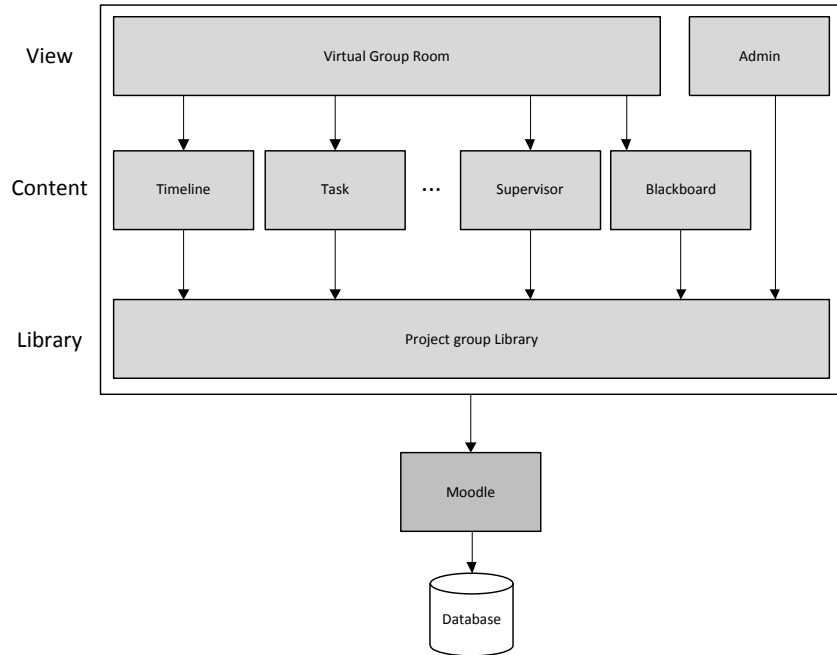


Figure 10.3: *The overall architecture of MyMoodle.*

this layer generate content for the virtual group room component. This layer has three dots in the center in order to show that there exist more components and more can be added here.

3. Below the Content Layer is the project group library, which contains common functionality. This layer handles all communication between the components in the Content Layer. We call this layer the “Library Layer”.

There are two primary factors we need to consider when planning our architecture. Firstly, we are four sub-groups working together. This creates the need for a structured way of communicating between the different component and it lets every sub-group know how their component is connected to the rest of the project. Secondly, the project should be passed on, which requires an architecture that grants great comprehensibility and extensibility 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 Content Layer, but depends directly on the project group library and Moodle. 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 10.3 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 relative size of the components do not imply anything, e.g.

the Moodle component is code wise much larger, than all of the components in MyMoodle combined, but is illustrated with a rectangle the same size as any of the Content Layer components.

10.4 Database

In this section we present the design of our database schema. Our design supports 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 Relation

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, which 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 humans 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 avoid race conditions. An example of a race condition in this system is 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 project 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 [67]. 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 a human identifier as well. However, we deem it much easier for 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 of the time of the creation of the project group.

The four fields of **projectgroup** are as follows.

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

An instance of **projectgroup** with a single tuple (representing our own project group) is shown Figure 10.4. Notice in the figure that **shortname** might seem somewhat cryptic. It translates to: “Software Engineering, 6th semester, project group number 8, spring (forår in Danish), 2012”. This illustrates the need for **longname**, since supervisors might find it difficult to distinguish several project groups of the same semester simply by their group number. The field **longname** gives a more descriptive name that should clearly give an idea of which project group it is.

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

Figure 10.4: *Example of an instance of **projectgroup**.*

10.4.2 Project Group Members Relation

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 group. Moodle saves users in a relation called **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 [68]. 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 Chapter 4) that current functionality of Moodle should not be altered. The first option 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 option we would make n fields in **projectgroup**, each containing a member of the project group or a value indicating no member, e.g. **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 option 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 set too 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 **projectrgroup** tuple will be wasted. If n is too low the system will be unacceptable, because some desired project groups cannot be created.

The final option 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 option 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 option is greater complexity by an increase in the number of relations. The strengths of this option outweighs its weaknesses and it is preferred to the previous option by allowing a true many-to-many link without wasting space or making the system unacceptable. We choose this option 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 the primary keys of the relations. The fields that hold these foreign keys are called **user** and **projectgroup**. 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 field called **id** to comply with Moodle guidelines [67]. We still make the combination of **user** and **projectgroup** unique, to ensure that no user 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 with that member. We can choose to assign a role to a user, which specifies which privileges he has in a project group. 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 students earlier in their education. Secondly we would have to make a new relation to contain only this information and thus increase complexity if the former option 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 the fields **created** and **updated** respectively. It may be useful to have these timestamps for administrative personnel, to see whether 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 [67].
- **projectgroup** references to the related project group.
- **user** references to the related user.
- **role** denoting the type of membership the user has to the project group.
- **created** is the timestamp for its creation.
- **updated** is the timestamp for the last update of the membership.

Figure 10.5 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 a given user to the same project group, namely the one shown in Figure 10.4. An example of an instance of the relation **user** can be seen in Figure 10.6.

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 10.5: *Example of an instance of **projectgroup_members**.*

id	firstname	lastname	email	...
1	Rasmus Veiergang	Prentow	rprent09@student.aau.dk	
2	Alex Bondo	Andersen	aban09@student.aau.dk	
3	Mikael	Midtgaard	mmidt09@student.aau.dk	
4	Kim Ahlstrøm	Jakobsen	kjakob09@student.aau.dk	
17	Kurt	Nørmark	normark@cs.aau.dk	

Figure 10.6: *Example of an instance of **user**.*

The database schema can be seen in Figure 10.7. The user entity is a built-in part of Moodle and is not modeled by us.

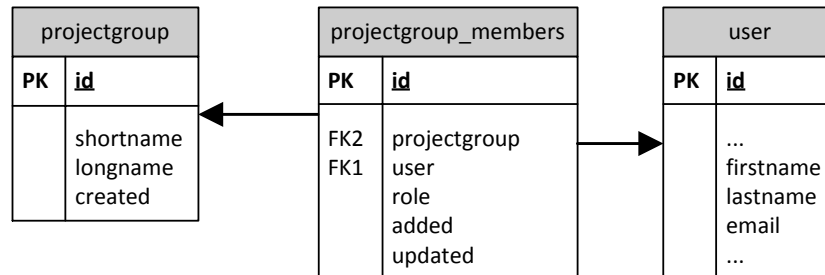


Figure 10.7: *The database schema of project groups and memberships. Most of the data fields of the user relation are omitted for brevity. The user table consists of more than 50 fields.*

Chapter 11

System Presentation

This chapter presents how the system works from the perspective of the user, which is key knowledge in order to comprehend the Implementation. This chapter is divided into two sections, one presents the virtual group room and the other presents the administration of MyMoodle. Information for logging into the system can be found on the CD in Appendix D.

11.1 Virtual Group Room

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

The virtual group room consists 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 a standard Moodle component, and we affect it by adding a headline – the rest of the header is added automatically. The header shows the name of the project group. The middle part contains the content of the virtual group room, and it is divided into three columns. The left column contains the standard navigation block in Moodle and is not created by us, though it is extended by us. The center and right columns both contain custom blocks. The center area is much wider than the left and right and can therefore contain bigger blocks. The various blocks presented in the virtual group room are described in Section 12.2.1. If a user wants to edit the layout for the virtual group room he can press the “Turn editing on” button. This will allow the user to remove, add, move, and edit blocks. A special “add new block”-block is added in editing mode to allow for adding new blocks. If a user edits the page, the change can be seen by all group members. Note that the virtual group room in Figure 11.1 is in editing mode.

In the figure an example of a block can be seen; it is denoted with a highlighted box. The highlighted box contains the members block and is created by our sub-group; we elaborate on the implementation of this in Section 12.2.1. The block shows a photograph and the name of each member of the project group.

Software 6 #8 Spring 2012

You are logged in as Alex Andersen (Logout)

Home » My project groups » SW608F12

Turn editing off

Navigation

Home

My home

Site pages

My profile

Courses

Users

My project groups

SW608F12

Settings

My profile settings

Site administration

Search

Software 6 #8 Spring 2012

Project Group Members

Rasmus Prentow

Mikael Midtgaard

Alex Andersen

Kim Jakobsen

Kurt Normark

Timeline

2012-05-28

Page 22

2012-06-03

Group Wall

Display newest first

Display oldest first

Sort by

Comments

Original Post

View

52

Files

52

Messages

52

Meetings

Indisend

previous

Jump to page: 1 of 1

next

Message

Sender: Kurt Normark

Time: Thursday, 10 May 2012, 11:02 AM

Dette er en test.

Kommentar: Gruppe ID ala SW608F12 siger mig ingenting. Jeg identificerer altid grupperne via grupperummet (evt. med A og B tilføjet hvis der er flere grupper i samme rum).

Posted by: Kim Jakobsen on Thursday, 31 May 2012, 06:48 PM

Det er modtaget

Meeting - Some meeting

Created by: Alex Andersen

Created on: Monday, 21 May 2012, 03:39 PM

Time: 2012-05-29 16:00

Duration: 0:30

Location: 3.1.36

Participants: Rasmus Prentow, Mikael Midtgaard, Alex Andersen, Kim Jakobsen, Kurt Normark.

Notes:

Task - Some Task

Sender: Alex Andersen

Time: Monday, 21 May 2012, 03:38 PM

Deadline: Thursday, 31 May 2012, 01:35 PM

Status: In progress

Completed: Not completed yet

Assigned to: Cannot find username.

Complete this to get reward

previous

Jump to page: 1 of 1

next

Blackboards

Title

Created

Last Modified

Untitled Blackboard

31 May 2012

31 May 2012

New Blackboard

Moodle Docs for this page

You are logged in as Alex Andersen (Logout)

Home

Figure 11.1: The virtual group room. The highlighted box shows the project group block and the dashed lines show the different areas on the page.

53

11.2 Administration

This section presents the administrative tool used by the administrative personnel.

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

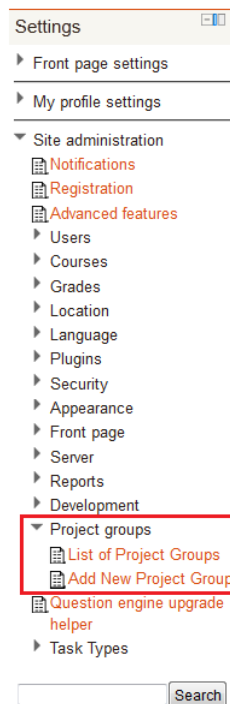


Figure 11.2: The settings block, which contains the site administration menu. The highlighted box displays the administrative tool for project groups.

The menu provides two links: One link to a list of all project groups, which is further described in Section 11.2.1, and another link to a page where a project group can be created, which 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 number 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 containing 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.

Project Groups

Search filters

Short name

contains

Add filter

Show advanced

Add New Project Groups

Short name	Full name	Actions
SW607F12	SW607F12	
SW606F12	SW606F12	
SW608F12	Software 6 #8 Spring 2012	
SW605F12	SW605F12	
SW604F12	SW604F12	
5.1.59A	Software 606	

Figure 11.3: A screenshot of a page listing all proejct groups in the system.

11.2.2 Project Group Members

The “Project Group Overview” page (seen in Figure 11.4), contains a list of the members of the project 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 confirmation from the administrative user.

SW608F12: Project Group Overview

[Edit Group](#)

[View Group](#)

Full name	Email	Role	Added to Project Group	Actions
Kurt Normark	normark@cs.aau.dk	Supervisor	Wednesday, 9 May 2012, 10:25 AM	✕
Alex Andersen	abondoa@gmail.com	Student	Wednesday, 9 May 2012, 10:25 AM	✕
kim jakobsen	spacepest@gmail.com	Student	Wednesday, 9 May 2012, 10:25 AM	✕
Mikael Midtgaard	mikaelmidt@gmail.com	Student	Wednesday, 9 May 2012, 10:25 AM	✕
Rasmus Prentow	rprentow@gmail.com	Student	Wednesday, 9 May 2012, 10:25 AM	✕

[Edit Group](#)

[Back to index](#)

Figure 11.4: A screenshot of the “Project Group Overview” page, which shows all current members of the project group.

11.2.3 Add & Edit

The add & edit page serves two purposes; it allows adding a project group and it allows for editing of a project group. The page is divided into four parts and can be seen on Figure 11.5. The first part contains fields allowing the user to enter basic information of the project group which is either edited or added. Beneath this there is a filter. 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.

SW608F12: Edit

General

Full name*

Software 6 #8 Spring 2012

Short name*

SW608F12

New filter

User full name

contains

[Show advanced](#)

[Add filter](#)

Users in list

Users ?

Available

All users (21)

Alex Andersen

Anders Eiler

Bjarke Søndergaard

Chuck Norris

Emil Custic

Esben Møller

Henrik Koch

Jeg er ikke Admin

Jesper Dalgas Zachariassen

kim jakobsen

kim jakobsen

Kurt Normark

Lasse Rørbaek Nielsen

Martin Sørensen

Selected

All selected (5/21)

Alex Andersen

kim jakobsen

Kurt Normark

Mikael Midtgaard

Rasmus Prentow

Selected user list...

[Add to selection](#)

[Remove from selection](#)

?

[Add all](#)

[Remove all](#)

Supervisors

Highlight

Supervisor(s)

None

Alex Andersen

kim jakobsen

Kurt Normark

Mikael Midtgaard

Rasmus Prentow

[Save changes](#) [Cancel](#)

There are required fields in this form marked*.

Figure 11.5: A screenshot of the add & edit page.

Chapter 12

Implementation

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

From Section 10.3 we know that there are three components, which must be implemented by us – the administration tool to manage project group, the project group, and the virtual group room. The administration tool is implemented as an admin tool, which, as described in Section 9.4, is a special Moodle plugin type.

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. By doing so we solve the problem of both implementing project groups and the virtual group room. This will make it possible to use activity modules, described in Section 9.4, in the virtual group room. Another option 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 built-in navigation block. With a local plugin we cannot use activity modules since they are too strongly connected to courses. Instead we can use blocks for the functionality. We choose to make a local plugin.

The local plugin includes both the project group library and the virtual group room.

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 of our peer-groups and several helper files. Our contribution to the library is located in the library file itself. The library consists of approximately 300 lines of code, including comments and whitespace and excluding any included files. All the functionality in the library is auxiliary functionality for the entire system.

The following sections explain how project groups are put into a Moodle context and how we ensure that users have 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 have different blocks for different virtual group rooms 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 34 different plugins types available. 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** (a class that is part of Moodle core code), 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 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 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($instance = 0)
2 {
3     return context_projectgroup::instance($instance);
4 }
```

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

The **instance** argument denotes a project group id.

With the new context and the function to instantiate it we can now add blocks to specific virtual 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 virtual 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 virtual group room 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 project group. The administrator check is necessary since an administrator should be able to see any project group even if he is not a member of the project group.

The function `has_projectgroup_write_permission` checks that the user has write permissions and uses the read permission function to check that the user has read permission. If he does not have read permission he should not be able to edit the virtual group room. 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.

Because the functions are separate it is possible to change them individually later. An example is that an end users requires that supervisors should only have read permissions. Then the change will only be in one place. Both functions can be found in the project group library.

12.2 Virtual Group Room

In this section the implementation of the virtual group room, its custom blocks, and the navigation to the virtual group room is described.

12.2.1 Blocks

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

```

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
6      * the right
7      */
8     $format['defaultprojectgroupblocks'] = ':projectgroup_members,
9         timeline,groupwall,blackboard:upload,tasks';

```

Code Snippet 12.3: *The default block configuration.*

The syntax for the format is `left:middle:right`. Left, middle, and right represent the three columns in the virtual group room. The blocks Timeline, groupwall, blackboard, upload, and tasks are 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 photographs of the project group members. It is implemented as a block in the same manner as the blocks created by our peer-groups, although our block is less complicated. All it does is fetch the members of the project group

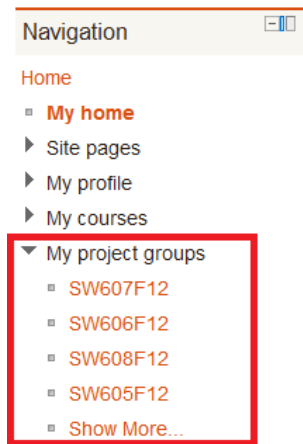


Figure 12.1: *Navigation block. The highlighted box shows our extension with a list of project groups, which the user is a member of.*

using the function `get_projectgroup_members` and present these along with a photograph for each member, which is fetched through a Moodle API call.

The layout of a virtual group room 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. The implementation that satisfies this requirement is presented here. Since we implement the virtual group room as a local plugin we can use a built-in functionality of Moodle to extend the navigation block [40]. The navigation menu can be seen in Figure 12.1. The extended part is highlighted by a rectangle.

The code to add project groups to the navigation block can be seen in Code Snippet 12.4. The id of the currently logged in user is used to get the project groups that he is a member of. The code checks if something should be displayed at all and then runs a loop to print out the link to each project group. If more than four project groups are associated with the given user a link called “Show More...” is added and no more links are shown. The “Show More...” link takes the user to an overview showing all project groups that he is a members of.

12.3 Managing Project Groups

Before any project group can be useful it has to be created. Administrative personnel needs to have the ability to add, edit, and delete project groups. The features we provide to manage project groups are known as admin tools in Moodle. This section describes how we implement the administration tools needed to manage project groups.

The main functionality is placed in the file `< /admin/tool/projectgroup/lib.php >`, which is a code library consisting of approximately 500 lines of code.

```

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.*

The library is now 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 inclusion is the implementation of the dependency between the admin component and the project group library component seen on Figure 10.3.

In the following sections creation, editing, and listing of project groups are presented. Deletion of project groups is 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 or 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 adding of new project groups. The only difference between editing and adding a project group is that

when editing a group an URL 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 a project group is being edited, a row in the database can be updated. If not, a new project group is being created and a row is inserted in the database instead.

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 of 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 roles. 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. The role of a member is as standard set to “0”, which indicates a student. If a member is highlighted in the view when the form is submitted (as Kurt is in Figure 11.5) the role value is changed to supervisor (indicated by a value of “1”) before the call to **save_or_update_projectgroup**.

12.3.2 Lisiting

In the list of project groups, mentioned in Section 11.2, we provide a filtering functionality, which makes it possible 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 a 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 as well. 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 bugs in the program and makes the entire system more robust.

13.1 Strategy

Our development is to some extent test driven. This means that we are creating unit tests for our functionality and integration tests to test whether our subsystem as a whole works as expected and works 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. 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 requirements, or expected 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 behaves as expected, and through the demo meetings where the system is tried out and validated by an end user.

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 [55, pp. 39-40].

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 is 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 partitioning [55, pp. 67-69] and boundary value analysis (or data testing) [55, pp. 70-79] in the creation of test cases. Without going into too great detail of these concepts, equivalence partitioning means that for every function we test, we divide input values into partitions. Within these partitions the outcome of each element is expected to be the same or share some similarity. 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” [55, 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 show some examples of our test cases. The test cases are found in Appendix C in `< /admin/tool/projectgroup/simpletest/ >` and `< /local/projectgroup/simpletest/ >`. The examples are 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 initially members of the project group. We create partitions and perform boundary value analysis on both these inputs.

The `group_id` is divided 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 is tested with a value m , where both ids $m - 1$ and $m + 1$ belong 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 an integer as input, we choose to use a string. We assume 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 removed can either be all, partly, or not at all valid, depending on whether each of the the given ids 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 additional invalid inputs. The boundary values for the 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 a trusted API or sudden database failure (though the database management system is responsible for recovery).

13.2.1 Test Case Examples

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 project group. In line 12 a project group of six users is created. The function **createTestGroup** utilizes the function that is used to create project groups in the admin tool library, but it also ensures removal of the project group after the tests are run, so the database does not get polluted with test data. 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 9.5.1), since we use an actual database for data storage during the test.

To make most of the unit tests run we have 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

```

1 function testRemoveMembersFromGroupEveryone() {
2     global $DB;
3
4     $groupmembers = 6;
5
6     //The data
7     $projectgroup = new stdClass();
8     //Set data - shortname, longname, 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 project 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.*

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.

```

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 erroneous input of an empty set of users.

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 functionality that we are testing are the project group library and the admin tool library. Code coverage reports are generated for these and can be seen in Appendix B.1 and Appendix B.2 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 unit and integration tests are combined with several UI tests, which help in finding bugs and ensure better quality of MyMoodle.

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.5, Appendix A.7, Appendix A.8, 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 method and practices. This is split into the two sections Inter Group Collaboration and Intra Group Development. The evaluation of our development approach is 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 Inter Group Collaboration

In this section the chosen development method and its utilization is discussed.

14.1.1 Initial Period

Prior to choosing a development method we conducted whole team assemblies. This practice worked well in the initial part of the project, but they became ineffective later on. A reason for this is that the topics discussed early in the development were important and interesting knowledge to everyone, but later some topics were uninteresting for some sub-groups while important to others. That made the enthusiasm at the whole team assemblies fall drastically and we organized fewer of these meetings as a result.

14.1.2 Choosing a Development Method

In the beginning of the project we decided to use Scrum of Scrums based on the 11 deciding factors described in Section 5.1.2. We have not tried any other development method for a multi-project, but we still believe we made the right choice based on the deciding factors.

14.1.3 Sprints

During our project we have had four sprints, the first one being a non-programming sprint. The sprints had a varying lengths, but they were always time boxed. The varying lengths of the sprints was necessary due to the fact that our course

schedule was unevenly distributed. The number of sprints is limited to four because of the hand in deadline of this report.

14.1.4 Meetings

Before the first sprint started we had a collaborate sprint planning meeting where all tasks for every sub-group was presented and estimated in collaboration. In successive sprints we decided that every sub-group was responsible for handling these meetings themselves, because there was only a limited gain from sitting in on the selection and estimation of sprint items of other sub-groups compared to the gain from Scrum of Scrums meetings that we conducted during the sprint.

Ideally we would have conducted our demo meetings with end users immediately after a sprint ended, but in some cases we had to wait because the end user representatives were not available in the given time period. This resulted in some feature requests that came in during a sprint and therefore had to be delayed – we prohibited ourselves from adding work to our sprint backlog during a sprint as Scrum suggests.

We conducted Scrum of Scrums meetings approximately twice a week, however, we planned meetings on an as need basis, which worked well. Scrum of Scrums meetings were also used to evaluate our development method, which in some cases caused the meetings to last longer than is normally allowed in Scrum of Scrums.

Scrum dictates that after every sprint a sprint retrospective meeting should be held to evaluate the development during the sprint in order to make the next sprint more pleasant and effective. We never conducted any formal sprint retrospective meetings in collaboration with our peer-groupss. We compensated for the missing meetings by evaluating the development method along the way or as part of the Scrum of Scrums meetings.

14.1.5 Group Room Location

Another issue is the lack of a shared room, which is preferred in Scrum of Scrums, but it helped that the group rooms are located close to each other. We quickly discovered that leaving the door open during working hours helped on the inter sub-group communication.

14.1.6 Tools

In this section we evaluate on the used tools during development.

Version Control We chose to use HG as our version control system. This worked well with the structure we set up where each person and group had its own repository on the server, which allowed for constant integration. There were some minor problems regarding permissions, but this was fixed by changing the server configuration.

Bug Tracking Bugzilla was installed and configured, but it was never used. The reason is that we did not spend time to document bugs, but fixed them immediately.

Documentation We documented the code using code comments, which can be parsed by PHPDocumentor or PHPXref. PHPXref was installed on the server and used frequently. PHPDocumentor was installed, but we never compiled the code with it.

Testing The use of the built-in testing framework in Moodle worked well and the test cases are now an integrated part of the code.

14.1.7 Summary

In general the development method worked well and it enabled us to adapt to changes. An example of a large change during development is when we realized that it was impossible to use activity modules in virtual group rooms without linking them through a course. This was overcome by changing the already implemented activity modules to blocks. The changed requirement was embraced, as agile methods recommend, and quickly implemented. To make this change quickly and effectively it helped a great deal that our physical group rooms were located close to each other, since we could communicate with our peer-groups as soon as we found the solution. The solution was found in collaboration with some of our peer-groups.

14.2 Intra Group Development

In this section the most relevant aspects of our development approaches are evaluated.

14.2.1 Scrum Roles

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 had an interchangeable Scrum master; 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 preferred 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 even from a single sprint.

An issue in the implementation of Scrum of Scrums is the lack of a product owner, which is described as a 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 ourselves, 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 [51, p. 128]. Furthermore, we have conflicting interests – we as programmers want to implement something exciting and challenging, while the end users may in fact need something we consider to be simple and boring.

14.2.2 Management of Requirements

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.

We tried several different approaches for estimating how many story points the size of a backlog item corresponds to. In the first sprint we estimated our sprint log items in collaboration with our peer-groups as described in Section 14.1.4.

In the second sprint we chose a few items from our release backlog and after estimating them using planning poker we decomposed the items into smaller tasks, which were then estimated. This led to a large number of small tasks, but as progress went on we realized that some tasks were redundant and could be combined. In other cases a few tasks were missing to fully implement a backlog item.

In the third sprint we tried to estimate only the backlog items. This made it difficult to delegate small tasks and estimate the number of story points burned.

In the fourth sprint we created one task for each backlog item and the person who took a task had the job of starting to program and create smaller tasks that other people could 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 also served as a motivational tool.

We continued to use a burndown chart when we transferred from mainly programming to mainly writing report. After the transition we just had report tasks such as “Write test chapter”. These tasks were given a story point value (usually just 1) and the burndown chart showed the progress by having the story points drop or rise from day to day.

14.2.3 End User Representatives

During the development we relied on several end user representatives in the gathering of our requirements. The selected representatives were not as spread out as we initially wished for. This is mainly because it is difficult to find end users who are willing to volunteer for interviews. For both categories of end users – the users of the virtual group rooms and the managers of project groups – we lack end user representatives from the Faculty of Medicine. For the usage of the virtual group room we need users with high experience and to interview more than one supervisor. 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 sub-groups collaboratively have developed. In this section we evaluate on 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 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 with 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 sub-group got the task of implementing the project group administration and the virtual group room, which is two very separate tasks with a diverse set of end users. The black board tool could be merged with the message board tool by integrating an existing drawing application, such the drawing tool in google docs, which is already integrated in Moodle by a 3rd party extension [45]. A new decomposition will then be:

- Management
- Planning
- Communication
- Virtual Group Room

This division would have given a more distinct grouping of end users to all components. The distribution of end users would be: Administrative personnel to Management, students to the other three sub-systems. Communication should consider supervisors as well.

15.1.2 Architecture

The architecture was not the result of an upfront 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 by some or all sub-systems 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 functionality. 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 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 the project group is updated (described in Section 12.3.1) 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` or deleting entries of members that are supposed to persist.

15.2 Project Group Management

In this section the sub-system implemented by our sub-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 theme of a semester, e.g. Application Development in our case. The project groups would be the instantiations, so to speak, of the project – MyMoodle in our case. A project can have a virtual meeting place in the same way a project group has a virtual group room, 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 sub-groups, which would make a student part of his sub-group and its parent group. It can be implemented using a recursive tree structure. Consider our case: We are four people working 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. 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.

The consequences of making a nested group structure is a more complex group structure for the administrative personnel to manage. Furthermore, it will not add any value for the members of the project groups but only create a group structure closer to how it is in reality. It may, however, be considered at a later stage in the development of MyMoodle.

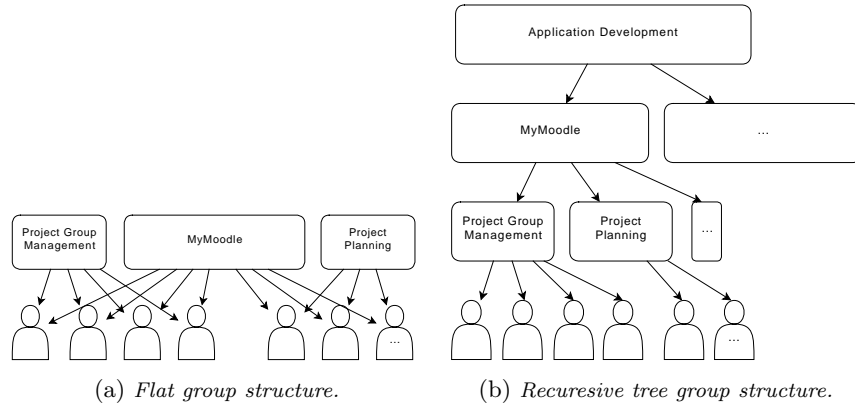


Figure 15.1: Recursive tree structure and our flat structure.

15.2.2 Virtual Group Room

In Section 10.1 we decided to implement the virtual group room to satisfy the need for a virtual meeting place. Each project group has its own virtual group 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 project groups 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 Aalborg PBL model.

15.2.3 Navigation

As shown in Section 12.2.2 at most four project groups are shown in the navigation menu. This number is quite low in order to avoid flooding the navigation menu. However, the four project groups that are shown are simply the four project groups that the users has been added to first. This can potentially pose a problem if old discontinued projects are not deleted. Users that are members of these would have to go to the overview page for project groups whenever they want to access a newer projects that are still active, which is considered to be more likely than accessing an old discontinued project.

A way to fix this is to reverse the order of the project groups and thereby always showing the newest project groups in the navigation menu. A more flexible solution would be to log whenever a user enters the virtual group room of a project group and then show the four most recently visited project groups in the navigation menu. This solution assumes that when a project group is visited at some point in time it is likely that it is visited again in a near future.

Furthermore, the users should be able to choose the number of project groups to be shown in the navigation menu themselves. If a user is working on, say, six projects, it would perhaps not bother him to have all these shown in the navigation menu instead of having to go to the overview page of project groups for one third of his virtual group rooms.

15.2.4 Administration User Interface

In the last demo meeting with Lene (seen in Appendix A.9) 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 old filters should be discarded per default 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 physical group room (if they have one), their email, and the city in which they study.

15.2.5 Management

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 is 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 already 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 is discussed in Section 10.1.3, must be changed to be more general to support all structures at the university (see Appendix A.6).

15.2.6 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 by a separate system on a separate server.

Chapter 16

Conclusion

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

Based on our problem definition in Section 1.4 we defined the system that we wanted to develop in Chapter 4. This definition applies to the entire system, MyMoodle. For our sub-system we have made a definition in Section 8.3. The rest of Part II describes how we have striven to meet the system definition by analyzing our requirements and designing, implementing, and testing our system.

The following sections conclude to which extent our system definitions have been realized.

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 for Moodle that allows Moodle to support the Aalborg PBL model.

Through our analysis we decided that to support the Aalborg PBL model we would 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 planning 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 planning meetings and writing on a message board. 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.

The virtual group room is implemented as a “local plugin” to Moodle. The content of the virtual group room is 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.

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. We have discussed issues in Chapter 15 that could be addressed to enhance PBL support.

16.2 Sub-Project

In the second part of this report we define the developed sub-system as:

A sub-system of MyMoodle that implements project groups in Moodle and allows 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 virtual group room based on requirements gathered from interviews and demo meetings conducted during the project. Each project group has one of such rooms associated with it. This room shows all the group members with name and profile picture, and allows other 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. 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. The developed sub-system is, however, a foundation on which development can be continued.

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 administrative personnel 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 to reserve a lecture room. Additionally there should be a system in place that handles a situation where there are not a sufficient number of physical group rooms available; no project groups should be able to make a significantly greater number of reservations than other project groups.

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

17.2 Central Project Group Database

An issue which is discussed in Section 15.2.5 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 group structures that exist at Aalborg University.
- There should be only one Moodle installation with one database system.
- 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.

We do not believe that this is a valuable feature to implement as long as there exist more than one Moodle installation at Aalborg University.

17.3 Virtual Meeting Place Templates

Templates for virtual meeting places will enable administrative personnel to create virtual group rooms with predefined virtual project tools. 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 a list of default blocks from a configuration file should get a name of a template. Based on this name the specific set of virtual project tools 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 be granted 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 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 semester project.

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] Git. URL <http://git-scm.com/>. Last viewed: 2012-05-31.
- [7] Google docs. URL <https://docs.google.com/>. Last viewed: 22/5-2012.
- [8] Gnu general public license. URL <http://www.gnu.org/copyleft/gpl.html>. Last viewed: 19/3-2012.
- [9] Mercurial. URL <http://mercurial.selenic.com/>. Last viewed: 2012-05-31.
- [10] Litmos. URL <http://www.litmos.com>. Last viewed: 9/5-2012.
- [11] Mahara. URL <https://mahara.org>. Last viewed: 9/5-2012.
- [12] Moodle, . URL <http://moodle.org/>. Last viewed: 19/3-2012.
- [13] Unit test api, . URL http://docs.moodle.org/dev/Unit_test_API. Last viewed: 7/3-2012.
- [14] Master in problem based learning in engineering and science. URL <http://www.mpbl.aau.dk/>. Last viewed: 19/4-2012.
- [15] Sharepointlms. URL <http://www.sharepointlms.com>. Last viewed: 9/5-2012.
- [16] Simpletest: Unit testing for php. URL <http://www.simpletest.org/>. Last viewed: 7/3-2012.
- [17] Skype. URL <http://www.skype.com/>. Last viewed: 22/5-2012.

- [18] Apache subversion. URL <http://subversion.apache.org/>. Last viewed: 6/3-2012.
- [19] Scrum of scrums, 2007. URL <http://www.scrumalliance.org/articles/46-advice-on-conducting-the-scrum-of-scrums-meeting/>. Last viewed: 2/5-2012.
- [20] Analysis of git and mercurial, 2008. URL <http://code.google.com/p/support/wiki/DVCSAnalysis>. Last viewed: 14/3-2012.
- [21] Studieordningen for bacheloruddannelsen i software, 2009. URL http://www.sict.aau.dk/digitalAssets/3/3331_softwbach_sept2009.pdf. Last viewed: 22/5-2012.
- [22] Activities. Webpage, 2012. URL <http://docs.moodle.org/22/en/Activities>. Last viewed: 03/05.
- [23] Activities block. Webpage, 2012. URL http://docs.moodle.org/22/en/Activities_block. Last viewed: 03/05.
- [24] Eventum issue / bug tracking system, 2012. URL <http://dev.mysql.com/downloads/other/eventum/>. Last viewed: 13/3-2012.
- [25] Eventum issue / bug tracking system, 2012. URL <http://dev.mysql.com/downloads/other/eventum/features.html>. Last viewed: 12/3-2012.
- [26] Activities modules. Webpage, March 2012. URL http://docs.moodle.org/dev/Activity_modules. Last viewed: 23/05.
- [27] Navigation block. Webpage, 2012. URL http://docs.moodle.org/21/en/Navigation_block. Last viewed: 04/05.
- [28] Plugins. Webpage, 2012. URL <http://docs.moodle.org/dev/Plugins>. Last viewed: 02/05.
- [29] Morten Mathiasen Andersen, 2012. URL <http://personprofil.aau.dk/121493?lang=en>. Title: Assistant.
- [30] Mads Peter Bach, 2012. URL <http://personprofil.aau.dk/102508?lang=en>. Title: Systems Administrator.
- [31] Scott Barge. Principles of problem and project based learning - the aalborg pbl model, Sep 2010. Last viewed: 18/4-2012, Folder describing the Aalborg PBL Model.
- [32] Alex Moesgård Bek, Dianna Hjorth Kristensen, Rasmus Oxenbøll Lund, Nikolaj Dam Larsen, and Kenneth Eberhardt Jensen. E-learning management system : Implementing customisable schedules, 2011. URL <http://projekter.aau.dk/projekter/files/52575492/Report.pdf>.
- [33] 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.

- [34] 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>.
- [35] Individual bugzilla.org contributors. Bugzilla: Features, April 2011. URL <http://www.bugzilla.org/features/>. Last viewed: 12/3-2012.
- [36] Lillian Buus, 2012. URL <http://personprofil.aau.dk/profil/103311?lang=en>. Title: Staff Member with University Degree.
- [37] Mark Drechsler. Moodle structural overview, 2012. URL <http://www.slideshare.net/mark.drechsler/moodle-structural-overview>. Last viewed: 9/5-2012.
- [38] 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.
- [39] Lene Winther Even, 2012. URL <http://personprofil.aau.dk/107261>. Title: Senior Secretary.
- [40] 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.
- [41] Marie Glasemann, 2012. URL <http://personprofil.aau.dk/116492?lang=en>. Title: Head of Section.
- [42] 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.
- [43] Marcus Green. Moodle. Webpage, 2011. URL <http://www.examulator.com/er/>. Last viewed: 2012-05-09.
- [44] The PHP Group. Package information: Phpdocumentor, 2012. URL <http://pear.php.net/package/PhpDocumentor/>. Last viewed: 6/3-2012.
- [45] Kim Hammond, Helen Foster, chris collman, Jason Hardin, Jonathan Harker, Nadav Kavalarchik, 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.
- [46] Mikael Møller Hansen, 2012. URL <http://personprofil.aau.dk/106741?lang=en>. Title: IT Administrator.
- [47] Jette Egelund Holgaard, 2012. URL <http://personprofil.aau.dk/103630?lang=en>. Title: Associate Professor.
- [48] 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.
- [49] Pia Kruse Knudsen, 2012. URL <http://personprofil.aau.dk/109828?lang=en>. Title: Senior Clerk.

- [50] Anette Kolmos, Flemming K. Fink, and Lone Krogh. *The Aalborg PBL model*. Aalborg University Model, 2004. ISBN 87-7307-700-3.
- [51] Craig Larman. *Agile & Iterative Development*. ADDISON WESLEY, first edition, 2004. ISBN 0-13-111155-8.
- [52] Camilla Gæraa Larsen, 2012. URL <http://personprofil.aau.dk/124239?lang=en>. Title: Trainee Clerical Assistant.
- [53] Lars Mathiassen, Andreas Munk-Madsen, Peter Axel Nielsen, and Jan Stage. *Object Oriented Analysis & Design*. The Johns Hopkins University Press, 1 edition, 2000. ISBN 87-7751-150-6.
- [54] Nilesh Parekh. The waterfall model explained, September 2011. URL <http://www.buzzle.com/editorials/1-5-2005-63768.asp>. Last viewed: 19/4-2012.
- [55] Ron Patton. *Software Testing*. Sams Publishing, July 2005. ISBN 0-672-32798-8.
- [56] 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.
- [57] Thomas Ryberg, 2012. URL <http://personprofil.aau.dk/109500>. Title: Associate Professor.
- [58] 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.
- [59] Gareth Watts. Phpxref, December 2010. URL <http://phpxref.sourceforge.net/>. Last viewed: 6/3-2012.
- [60] Moodle Wiki. License, August 2011. URL <http://docs.moodle.org/dev/License>. Last viewed: 29/5-2012.
- [61] Moodle Wiki. About moodle. Webpage, April 2012. URL http://docs.moodle.org/22/en/About_Moodle. Last viewed: 2012-05-09.
- [62] Moodle Wiki. Access api. Webpage, 2012. URL http://docs.moodle.org/dev/Access_API.
- [63] Moodle Wiki. Admin tools. Webpage, January 2012. URL http://docs.moodle.org/dev/Admin_tools. Last viewed: 2012-05-09.
- [64] Moodle Wiki. Core apis. Webpage, 2012. URL http://docs.moodle.org/dev/Core_APIs.
- [65] Moodle Wiki. Course categories. Webpage, March 2012. URL http://docs.moodle.org/22/en/Course_categories. Last viewed: 2012-05-09.
- [66] Moodle Wiki. Course formats. Webpage, April 2012. URL http://docs.moodle.org/22/en/Course_formats. Last viewed: 2012-05-09.

- [67] Moodle Wiki. Database. Webpage, April 2012. URL <http://docs.moodle.org/dev/Database>. Last viewed: 2012-05-14.
- [68] Moodle Wiki. Database schema introduction. Webpage, May 2012. URL http://docs.moodle.org/dev/Database_schema_introduction. Last viewed: 2012-05-15.
- [69] Moodle Wiki. Data manipulation api. Webpage, 2012. URL http://docs.moodle.org/dev/Data_manipulation_API.
- [70] Moodle Wiki. File:moodle-contexts-1.8.png. Webpage, 2012. URL <http://docs.moodle.org/dev/File:Moodle-contexts-1.8.png>.
- [71] Moodle Wiki. Form api. Webpage, 2012. URL http://docs.moodle.org/dev/Form_API.
- [72] Moodle Wiki. Output api. Webpage, 2012. URL http://docs.moodle.org/dev/Output_API.
- [73] Moodle Wiki. Page api. Webpage, 2012. URL http://docs.moodle.org/dev/Page_API.
- [74] Moodle Wiki. Roles and modules. Webpage, 2012. URL http://docs.moodle.org/dev/Roles_and_modules.
- [75] Moodle Wiki. Unit test api. Webpage, 2012. URL http://docs.moodle.org/dev/Unit_test_API.

List of Figures

1.1	<i>A Moodle course page for the Test and Verification course.</i>	5
5.1	<i>Radar chart showing the deciding factors for choosing the development method of our project.</i>	18
7.1	<i>Illustration of how backlog items move through the different backlogs.</i>	23
8.1	<i>An overview of our end users. The gray boxes contain the two categories of end users. The upper box contains the members of the project groups, the lower box is the managers of the project groups.</i>	27
8.2	<i>A graph showing how the end user representatives of the users of virtual group room are spread between faculties and how much experience they have with regard to LMSs.</i>	28
8.3	<i>A graph showing how the end user representatives of the managers of project groups are spread between faculties and how much experience they have with regard to LMSs.</i>	29
8.4	<i>Lists showing features from our product backlog.</i>	30
9.1	<i>The database scheme for groups and groupings. Data fields that are neither part of a primary key nor foreign key are omitted for clarity [43].</i>	34
9.2	<i>The hierarchical context structure of Moodle. Picture source: [70].</i>	37
10.1	<i>The connection between the virtual meeting place and the virtual group room.</i>	39
10.2	<i>The difference between the two approaches to the division of project and group illustrated by an example.</i>	41
10.3	<i>The overall architecture of MyMoodle.</i>	47
10.4	<i>Example of an instance of <code>projectgroup</code>.</i>	49
10.5	<i>Example of an instance of <code>projectgroup.members</code>.</i>	51
10.6	<i>Example of an instance of <code>user</code>.</i>	51
10.7	<i>The database schema of project groups and memberships. Most of the data fields of the user relation are omitted for brevity. The user table consists of more than 50 fields.</i>	51
11.1	<i>The virtual group room. The highlighted box shows the project group block and the dashed lines show the different areas on the page.</i>	53
11.2	<i>The settings block, which contains the site administration menu. The highlighted box displays the administrative tool for project groups.</i>	54
11.3	<i>A screenshot of a page listing all project groups in the system.</i>	55
11.4	<i>A screenshot of the “Project Group Overview” page, which shows all current members of the project group.</i>	56

11.5	<i>A screenshot of the add & edit page.</i>	57
12.1	<i>Navigation block. The highlighted box shows our extension with a list of project groups, which the user is a member of.</i>	61
15.1	<i>Recursive tree structure and our our flat structure.</i>	76

List of Code Snippets

9.1	<i>Example of the Page API and Form API in Moodle.</i>	35
9.2	<i>Example of how to change the name of a user.</i>	36
9.3	<i>Code snippet from < /course/view.php > showing how the context of a course is set.</i>	36
12.1	<i>The context_projectgroup class header and constant definition.</i>	59
12.2	<i>The function to get projectgroup context.</i>	59
12.3	<i>The default block configuration.</i>	60
12.4	<i>The code for extending the navigation block.</i>	62
12.5	<i>The invocation of the filtering mechanism.</i>	63
13.1	<i>A test case for the function <code>remove_projectgroup_members</code>. The test case tests if the function correctly removes all the members of the project group when instructed to.</i>	67
13.2	<i>A test case for the function <code>remove_projectgroup_members</code>. The test case tests if the function correctly handles the erroneous input of an empty set of users.</i>	68

Appendix

Appendix A

Interviews & Demo Meetings

A.1 Thomas Ryberg Interview

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.2 Lene Winther Even Interview

Interviewers: Kim A. Jakobsen and Mikael Midtgaard

This interview was conducted on March 15th at Institute of Computer Science, Cassiopeia, at Aalborg 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 summary of the interview.

Moodle in General

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

When there is a change in a calendar 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 does 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 calendar system for courses. Calendar events are created in CalMoodle and imported to the Moodle system. Calendar 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.

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.3 Jette Due Nielsen & Pia Knudsen Interview

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 Morten Mathiasen Andersen Interview

Interviewers: Kim A. Jakobsen and Alex Bondo Andersen

This interview was conducted on March 22th at Department of Development and Planning at Aalborg 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.5 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 Casiopeia 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.6 Mikael Møller Hansen Interview

Interviewers: Kim A. Jakobsen and Rasmus Veiergang Prentow

This interview was conducted on April 12th at Cassiopeia at Aalborg 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 is 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 groups from AdmDB 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.7 Lea Gustavson Demo Meeting Sprint 4

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.

A.8 Mathilde Gammelgaard Demo Meeting Sprint 4

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 too much emphasis 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 spacing 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.

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 Casiopeia 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.

In a virtual group room 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 longname 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 calculations, which is a task that the administrative personnel at Aalborg University do.

Appendix B


Code Coverage

B.1 Project Group

5/17/12

Moodle Unit Tests Code Coverage Report

Spike PHPCoverage



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.

B.2 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.

Appendix C

Source Code

The source code for our sub-system of MyMoodle is found on the attached CD in the folder labeled Appendix C. The source code on the CD follows the structure of Moodle but only code produced by our sub-group and the shared project group library is included.

Appendix D

Login Information

Server information and login are located on the CD in a file called “Appendix D”.