Teaching Tools Research Group: Course Management Software

Prepared For: Dan Lecocq

Team Members: Jeff Park, Corey Bell, Tyler Pare, Craig Buche

Executive Summary

With the emergence of computers and the Internet in the world of academia, online course management software has become an integral part of classes. This project focused on the desire to create course management software that is lightweight, consistent, and most importantly, easy to use in comparison to existing systems. The team started from scratch by performing research from both the professor and student perspectives in order to include their suggestions and concerns into the design. We have developed a web-based application using Ruby on Rails (RoR).

Table of Contents

1) Introduction

The client, representing the Teaching Tools Research Group (TTRG), submitted a request for a new, web-based course management software system as an alternative to existing systems. One popular course management system, BlackBoard, used by the Colorado School of Mines and many other institutions is an example of aging software that lacks many of the key characteristics and benefits of agile development. The team conducted research on the subject by interviewing several professors, students, and even TAs (see Appendix A). The general consensus was that, while some of the features of the existing system are nice, overall it is clunky, not-intuitive, and has needless complexity. Consequently, our team set out to create a course management system from the ground up; we've incorporated agile development methods and principles from the start in order to create software that avoids the major pitfalls of the existing system.

2) Requirements

2.1) Functional Requirements:

1. Professors should be able to create and manage multiple courses
2. Professors should be able to manage grades for each course and each enrolled student
3. Professors should be able to post announcements, assignments, and documents
4. Professors should be able to add and remove students from their courses
5. Students should be able to view their grades
6. Students should be able to submit assignments

2.2) Non-Functional Requirements

1. Software will be written using the RoR framework
2. The system will be hosted via virtual machine running Linux
3. The final product must be finished and client approved by June 16th 2009
4. The system should be usable and should display consistently across multiple web browsers
5. Flow, ease of use, and consistency should be maintained throughout the system

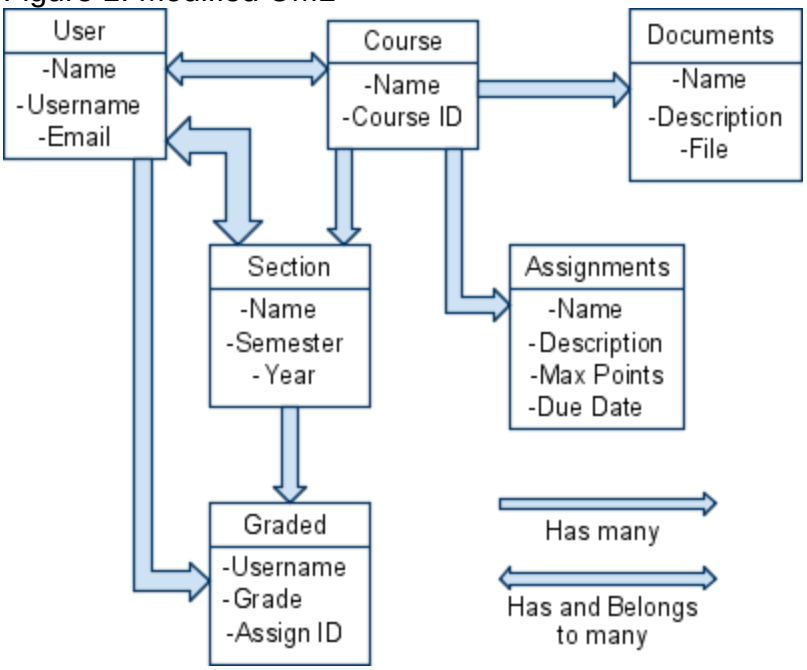
3) Design

3.1) UML

RoR uses a model, view, controller (MVC) design pattern, so traditional UML diagrams fail to accurately portray projects created within the Rails framework. Presented in Figure 1 is a modified UML that better shows the relational nature of classes within our project. Each model in RoR has a corresponding table in the database, and each unique object of the model has its own row within that table. Consequently, RoR typically uses database relationships such as "many to many" (represented by the double arrow) to describe the relationships between classes rather than inheritance – e.g. while a course may have many sections, and while sections could inherit variables such as name, description, etc, a simple database relation replaces the need for inheritance by sections simply holding a foreign key to

the course that the section belongs to. RoR is also designed to read almost like conversational English - for example, the one way arrow means the class at the base of the arrow "has many" of the class at the head of the arrow (e.g. a course has many assignments).

Figure 1: Modified UML



3.2) Database

The database aspect of our project is primarily managed by the Rails framework. When Rails generates framework code, it creates corresponding database tables and fields for each model. Models in ruby are asynchronous to classes in other common programming languages. RoR uses ActiveRecord, which is an object relation mapping tool; ActiveRecord abstracts SQL code from the programming language – letting Rails manage the database queries. RoR defaults to using Sqlite3 as its database management system; the team has kept the Sqlite3 system because it is a clean, lightweight, and fast system. The database schema used in this project is available in Appendix A.

3.3) Use Cases:

1. User logs in

Description: System retrieves data corresponding to the user
Precondition: User has login info
Post condition: User is logged in

Primary flow:

1. User inputs login information
2. System sets user based on login information
3. User's homepage is displayed

Alternate flow

1. System can't find matching login information
2. System returns to login page and displays error

2. Professor adds a course

Description: Professor adds a course to their list of courses
Precondition: n number of courses
Post condition: n+1 number of courses

Primary flow:

1. User clicks add course button
2. System asks for course name
3. Course is added

3. User accesses grades.

Description: User accesses grades
Precondition: User is logged in and enrolled in a course
Post Condition: User has access to grades

Primary flow:

1. User clicks on grades
2. System displays user's grade book

4. Professor adds an assignment

Description: Professor adds an assignment to a course
Precondition: Course exists and professor is logged in
Post Condition: New assignment is added to the course

Primary Flow:

1. Professor selects the appropriate course
2. Professor clicks on the "Add New Assignment" button
3. System asks for assignment name, description, max grade, file, and the due date (name and max grade are required)
4. Assignment is added to the course

5. Uploading a document

Description: Professor adds a document to a course
Precondition: Professor is logged in
Post Condition: New document is added to the course

Primary Flow:

1. Professor selects the appropriate course
2. Professor clicks on the "Add New Document" button
3. System asks for document name, description, and file
4. Professor clicks on the "choose file" button and browses for the appropriate file
5. Professor clicks "create" button
6. Document is added to the course

4) Implementation Details and Results

4.1) Languages and Tools Used

For this project, the client specified that the group develop the software in the Ruby language using the Rails framework. Since the project is web-based, RoR was a perfect fit due to the inherent web-application support it provides. We found RoR to be a powerful platform because it supports countless plug-ins and integrates seamlessly with existing web technologies such as HTML/XHTML, Javascript, and AJAX. RoR also added the versatility our project needed because it is not platform specific; project development is not dependent on a specific operating system and the finished product displays across multiple browsers.

4.2) Development Process

Included in the project are two major plug-ins: an attachment plug-in for uploading files and a calendar plug-in for selecting assignment due dates since these would be beyond the scope and time frame of our project to develop in-house. Ultimately, in order to have the finite control we wanted over the software as well as to learn as much as possible from this field session, a majority of the code was created from scratch.

4.3) Design Issues:

The project presented had a myriad of challenges throughout the project development:

- At the start, our client provided the team with a development environment - a virtual machine running a Linux distribution. While the client offered his expertise and guidance, ultimately learning the Linux console commands and setting up a version control (SVN) repository was left up to the team. Due to the team's inexperience with the Linux operating system, this was another challenge in the beginning stages of the project.
    - These inexperience issues were resolved with the aid of internet searches for command syntaxes, examples, and trial and error.
    - Early on, the client granted the team sudo (admin) privileges, which resolved several issues by allowing server-side commands to be executed as an administrator.
- Another issue related to the development environment was the version control repository itself – the team had issues with compatibility between their Windows based laptops and the Linux server. Often times, team members would have difficulty committing changes or updating their source code.
    - The solution was to use a program called Tortoise SVN; the popular and well-designed program eliminated these compatibility issues as well as streamlined SVN operations for the Windows machines.
- The last major issue involved the fact that RoR has three major components: Ruby, Gems, and Rails. Since RoR is in continual development, new versions of each component are released on a fairly regular basis. Our team ran into several problems with version mismatching; newer versions would often implement new commands that replaced deprecated syntax or procedures. While mismatched versions of the three components still work together, this often led to slow debugging of issues due to the uncertainty of whether problems stemmed from the code or the version mismatch. To

complicate matters further there was a lack of consistency between the machines uses to develop the application.

- The most important solution to this problem was becoming aware that several debugging issues were linked to version mismatch. This knowledge was invaluable in solving many seemingly impossible problems. Fortunately RoR allows you to specify which version of the framework to use - a necessity when unsure of which version each developer is using. Also, RoR has the flexibility of allowing users to manually install previous versions of each component to ensure heightened compatibility.

5) Scope and Project Progression

The team recognized that with a limited development time some features would be have to be left out entirely. Ultimately, our team was able to limit the scope and accomplish several milestones in our project development and they are as follows:

- A working login system
- Instructors are able to create courses
- Instructors are able to add assignments and upload documents to the courses
- Instructors are able to create sections for each course – e.g. for a calculus course, the instructor can create a Section A, Fall 2009 with its own students enrolled in the section
- Instructors are able to add announcements for each course that can email students and update an RSS feed
- A working gradebook for each section that instructors can open and edit grade values for each student and each assignment
- Students are able to view course documents, view their grades, and upload assignments for grading
- Site design that is aesthetically pleasing and easy to use

6) Conclusions and Future Directions

6.1) Lessons Learned

Over the course of project development, our team learned key lessons that are important for developing within the RoR framework.

- Developers using RoR should be acutely aware of what version(s) their development environment is using and the constraints related therein
- Following RoR convention is important to the portability of the code – i.e. creating migration files for adding columns to a table as opposed to manually adding them to the database is essential when working with several team members
- Taking time to familiarize one's self with the MVC design of RoR is an essential part of the ramp-up process

6.2) Future Directions

Our interview research yielded several suggested features that our team would have liked to add but did not have the time to implement (see appendix B for interviews). Some areas for future development include:

- A much more robust gradebook system that includes the following:
  - An option for instructors to implement weighted grades
  - Automatic curving options for tests and even final grades
  - More optimizations and streamlining for instructor and TA use
- More communication/collaboration tools
  - A well developed email system for mass communications
  - The ability to create user groups within sections – i.e. for group projects and communication therein
- A dynamic syllabus – rather than a document to download, the ability to easily create an HTML document that would contain not only the relevant information, but instructor links as well.

6.3) Summary

Ultimately, our team completed the essential elements of a course management system that is inline with the client's request. We have met and exceeded the client's initial expectations by going beyond the original scope set forth at the onset of the project. We also believe that we have setup a solid foundation for future work should the option be pursued. The project was a success.

Glossary of Terms

Ruby	A scripting language similar in nature to Python and Perl – primarily used for web-applications.
Rails	A web development framework created specifically for Ruby. It implements the model, view, controller architecture. The model corresponds to conventional object-oriented programming; it represents the object, its variables, and its functions. The view is the visual representation of the project. The controller is the action handler that handles input and notifies the user of the results.
Gems	A packet manager for Ruby, used to install Ruby programs, libraries, and plug-ins.
RSS	Really Simple Syndication. A web feed format used to publish frequently updated works in a standardized form.
Sqlite	A lightweight, versatile, and open-source database management system.
HTML/XHTML	Hypertext Markup Language and Extensible Hypertext Markup Language respectively. The language web pages are written in; both formats are recognized by web browsers worldwide.
Javascript	A scripting language used in web-page development that is often embedded within HTML/XHTML. It has no relation to the JAVA programming language.
AJAX	Asynchronous Javascript and XML. A highly visual form of Javascript that is commonly used to update portions of a web page without refreshing the entire page.
Sudo	Super User Do. A linux syntax that will run a single command as an admin – e.g. if a program can only be installed by an administrator, the sudo command will allow it to be installed.
SVN	Subversion. A version control system used to track and manage version changes for a project.

Reference Materials

[1] D. Thomas and A. Hunt Programming Ruby: The Pragmatic Programmer's Guide, Addison Wesley Longman Inc., 2001.
Available HTTP: ://www.rubycentral.com/book/

[2] S. Ruby, D. Thomas, and D. Hansson Agile Web Development With Rails, The Pragmatic Programmers LLC., 2009.

[3] Rails API. Available HTTP: http://api.rubyonrails.org/

Appendix

9.1) Appendix A: Database Schema

courses
id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL
name varchar(255) DEFAULT NULL
courseID varchar(255) DEFAULT NULL
created_at date
time DEFAULT NULL
updated_at datetime DEFAULT NULL

announcements
id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL
announcement varchar(255) DEFAULT NULL
course_id integer DEFAULT NULL
created_at datetime DEFAULT NULL
updated_at datetime DEFAULT NULL

assignments
id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL
name varchar(255) DEFAULT NULL
max_points integer DEFAULT NULL
description varchar(255) DEFAULT NULL
due_date date DEFAULT NULL
course_id integer DEFAULT NULL
created_at datetime DEFAULT NULL
updated_at datetime DEFAULT NULL
max float DEFAULT 0
average float DEFAULT 0
median float DEFAULT 0
min float DEFAULT 0
parent_id integer
content_type varchar(255)
filename varchar(255)
thumbnail varchar(255)
size integer
width integer
height integer
db_file_id integer

courses_users
id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL
course_id integer DEFAULT NULL
user_id integer DEFAULT NULL

db_files
id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL
created_at datetime DEFAULT NULL
updated_at datetime DEFAULT NULL
data blob

content_type varchar(255)
parent_id integer

documents
id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL
name varchar(255) DEFAULT NULL
description varchar(255) DEFAULT NULL
docfile blob DEFAULT NULL
course_id integer DEFAULT NULL
created_at datetime DEFAULT NULL
updated_at datetime DEFAULT NULL
parent_id integer
content_type varchar(255)
filename varchar(255)
thumbnail varchar(255)
size integer
width integer
height integer
db_file_id integer

gradeds
id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL
assignment_id integer DEFAULT NULL
username varchar(255) DEFAULT NULL
grade float DEFAULT NULL
section_id integer DEFAULT NULL
created_at datetime DEFAULTNULL
updated_at datetime DEFAULT NULL
parent_id integer
content_type varchar(255)
filename varchar(255)
thumbnail varchar(255)
size integer
width integer
height integer
db_file_id integer
submitted integer DEFAULT 0
average float
min float
max float
median float

users
id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL
name varchar(255) DEFAULT NULL
username varchar(255) DEFAULT NULL
email varchar(255) DEFAULT NULL salt varchar(255) DEFAULT NULL
hashed_password varchar(255) DEFAULT NULL
access_level integer DEFAULT 0

created_at datetime DEFAULT NULL
updated_at datetime DEFAULT NULL)

submissions
id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL
created_at datetime DEFAULT NULL
updated_at datetime DEFAULT NULL
parent_id integer
content_type varchar(255)
filename varchar(255)
thumbnail varchar(255)
size integer
width integer
height integer
db_file_id integer

sessions
id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL
session_id varchar(255) NOT NULL
data text DEFAULT NULL
created_at datetime DEFAULT NULL
updated_at datetime DEFAULT NULL

sections_users
section_id integer DEFAULT NULL
user_id integer DEFAULT NULL

sections
id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL
name varchar(255) DEFAULT NULL
semester varchar(255) DEFAULT NULL
year integer DEFAULT NULL
course_id integer DEFAULT NULL
created_at datetime DEFAULT NULL
updated_at datetime DEFAULT NULL

9.2) Appendix B: Interview Results

LAIS Professor: Jason Delborne

Likes:

- Student access to grades
- Integration to Safe (anti-plagiarism check)
- Ability to post readings and links
- Ability to post announcements and email users/groups/class
- Ability to customize a grading sheet

Dislikes:

- That BB does not keep an archive of sent emails
- That setting up a grading sheet was hard, not intuitive, and was lacking features
- That students had little control over submissions – students cannot post informational items for other students to see
- That the announcement system is not very streamlined

Desired Features:

- Would like an electronic syllabus – i.e. not simply a downloadable document, but rather a page that could be created with a tool that contains things like course information and links to the specific items mentioned within
- An easily viewable deadline / important date list available to students
- A better way to integrate extra credit
- An overall more streamlined experience

LAIS Professor: Sandy Woodson

Likes:

- Rollover between classes/semesters
- Students can enroll themselves

Dislikes:

- Random students can enroll
- Difficult to view/rearrange columns in the gradebook, resulting in only posting major assignments
- Customize tabs instead of using presets that don't reflect user preferences

Desired Features:

- Transfer from the gradebook on BB to Banner

MACS Professor Terry Bridgman

Likes:

- The idea of a central site customized per student for their courses

Dislikes:

- It tends to be slow with a bizarre number of page refreshe
- It has limited testing capabilities
- Also (a very small quirk), in the grade center, BB computes averages differently than most spreadsheet software. If I leave a grade blank (I do like to distinguish between a 0 and a no-submittal), BB will not include that assignment in the average. I would like a blank treated as a 0 in the average

Desired Features:

- LaTeX support and equation parsing for ultimately having quiz support beyond simple true/false. Allowing the user to type in answers would be nice

TAs Drew Huck and Jeff Kim

Regarding TA tasks:

- Updating information was time consuming and unnecessarily complicated
- Adding grades was a hassle – too many steps to be repeated for each student
- Grade view is bulky and unwieldy

Regarding Student Use:

- Liked the ability to see grades, but often found they were poorly updated or not used by the professor at all

User interface is not intuitive – finding and using certain features is harder than it should be

Student Shanley Philip

Dislikes

- Home page is useless—must visit another page to log in
- Buttons and tabs can be ambiguous
- Hard for students to send files to each other

Desires

- Group courses by semester
- Different color schemes
- Classes as tabs

Student Jared Alexander

Dislikes:

- No quick communication with the professor
- Grades never seem to be accurate

Student Grant Gunhus

Likes:

- Easy access to course materials like Syllabuses and documents

Dislikes:

- Grade inconsistencies
- There is a lot of stuff that never seems to be used