THE DEVELOPMENT OF PROJECT GRADE-UP

by

Dalin Williams

A PROJECT

Presented to the Faculty of

The School of Computing at the Southern Adventist University

In Partial Fulfilment of Requirements

For the Degree of Master of Science

Major: Computer Science

Under the Supervision of Professor Anderson

Collegedale, Tennessee

November, 2015

THE DEVELOPMENT OF PROJECT GRADE-UP

Dalin Williams, M.S.

Southern Adventist University, 2015

Adviser: Scot Anderson, Ph.D.

The university classroom has greatly evolved from a simple syllabus and in class discussion to the modern online documentation and virtual classrooms. These developments have changed the way students review their grades and balance their workloads. With the plethora of new technologies, students are often burdened with a full school schedule, work, and social events, with few tools to help them effectively understand their grades or manage their time. Current solutions addressing this issue do not present data in an organized way that allows the student to easily comprehend their past performance or up coming work load. Our solution builds upon the Moodle system by adding visual, progress-specific information that is comprehensible at a glance. This in turn allows the student to answer the following questions:

1. What have I completed, and what do I have left to complete?
2. What is my current grade and projected grade at my current pace?
3. Given what I've done so far what is the best possible grade I could get if I ace the remaining work?
4. What if I stopped now, what would my grade be?
5. How am I doing compared to the average in this class?
6. If I got a particular grade(s) on a specific assignment(s) how would that change the answers to the preceding questions?

7. Where is the work left concentrated in the temporal domain? I.e. when should I start working on the items left to complete in my course(s)?

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Students today want a well-defined grade and progress visualization system that shows their current grade and projected possible grades. In addition, students want an easily interpreted visualization showing concentrated study times occurring in a single class or more importantly overlaps across all classes within a semester, as this would allow them to plan study time more effectively and efficiently.

Institutional attentiveness to education and technology has taken a more dominate role, making the need for an efficient Learning Management Systems (LMS) the top priority in many educational institutions [1]. This has led to the creation of a wide selection of systems such as Desire2Learn, Blackboard, and Moodle. These different LMS satisfy educational institution's desire for technological innovation, while providing a common interface for teachers and students, and creating an organized system for presenting educational material.

The high demand of work, school, and social obligations force students to pick and choose which classes to work on, and which classes not to. Uninformed decisions tend to lower the student's grade, in turn hampering their overall academic performance [2]. There currently exists two methods of thought to

efficiently address this issue: grade prediction and student self-projection [3].

Desire2Learn [4] provides a method through which students may view a form of self-projection they call "progress projections." These projections allow students to manage their time and class work preference in accordance with time available and course load. The increased student awareness allows a student to pick and choose which assignment or course to focus on. These tools, in theory, positively affect student efficiency and effectiveness in handling the course loads [5]. Desire2Learn's solution utilizes predictive analysis technology to give student's information such as current success rate per course, success rate per course item, and grade per term within a given semester. This allows students to keep a running tally on their progress, while also knowing what areas need more work to improve their grade.

However, this solution does not address the issue of grade projection, and is only currently available on the Desire2Learn LMS. The solution leaves the students to guess at what they must do in order to achieve a certain letter grade, further preventing students from knowing the approximate difficulty and time consumption of any given assignment. This deficiency with in the application prevents the student from maximizing their potential.

Therefore, we believe no application exists which allows a student to visually review a projected grade based off their current grade and future course work or visually show concentrated study times required during all their classes and assignments. We address these issue by our completion of the following goals:

- The creation of a visualization plug-in for the Moodle LMS that shares many similarities to burn-up charts from software engineering. This visualization gives students instant feedback and projected information about their current and potential grade within a class.

- The creation of a visualization plug in for the Moodle LMS that shows workloads over the course of the semester across individual and all classes. This visualization gives students a better understanding of current and upcoming difficulties within their course load.

Giving these additions, a student can visually ascertain their grade trajectory within classes and their work load over time in all classes in a few seconds. We believe this will help students to plan ahead and complete work more efficiently and accurately.

For the remainder of our paper, we first cover the background in Chapter 2. Chapter 3 and Chapter 4 will cover our project and the evaluation respectively. Finally, Chapter 5 presents our conclusion on the research found and our future work on the Master's Project.

Chapter 2

Background

The push to use technology within the educational intuition systems has led to the development of many Learning Management Systems (LMS). These systems allow unprecedented access to information and knowledge not readily available in the past except upon request. These web-based systems allow instructors, parents, students, and even colleagues to view data ranging from assignment scores to overall performance data.

LMS grew out of Content Management Systems (CMS) and hence focus on content. LMS rarely provide *semantic analysis* of student success beyond overall grades and feedback specifically entered by the grader. LMS also rarely provide *planning tools* for success beyond the usual due dates and calenders. In this section we explore the ability of several LMS in these two areas.

There are many LMS solutions available to choose from. Figure 2.1 shows the most popular software packages [6].

From Figure 2.1, we select Moodle, Edmodo, Blackboard, and Desire2Learn for analysis, and take into consideration their weakness and strengths as follows:

- Peer/Instructor Feedback

LMS	Users
Moodle	73,753,035
Edmodo	20,000,000
ConnectEDU	20,000,000
Blackboard	20,000,000
Cornerstone	11,000,000
Instructure	11,000,000
Desire2Learn	11,000,000
Interactyx	10,000,000
Meridian Knowledge Solutions	8,500,000
SkillSoft	7,000,000
Latitude Learning	3,300,000
SumTotal	2,000,000
Schoology	2,000,000
Litmos	1,260,000
Collaborize Classroom	350,000
Docebo	300,000
DigitalChalk	290,900
Rcampus	270,000
Educadium	40,000

Table 2.1: Popular LMS packages.

- Course Communication
- Student Progress/Tracking

2.1 Benefit Analysis: Moodle

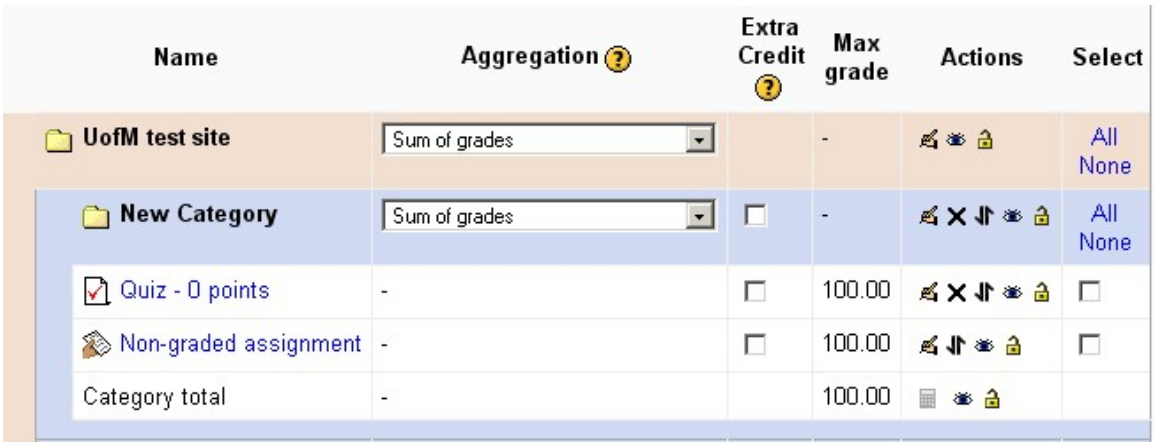
Moodle is an open source CMS, allowing for access to a wider market including small businesses, government facilities, undergraduate education, and graduate education. This software flexibility has led to an enormous software supporting community [7]. Several distinct benefits of Moodle include [7]:

- Ease of Customization: Open-source code can be easily accessed, modified, and distributed.

- Extensibility: Third party add-ons allow for quick and clean customization.
- Ease of Localization: Since the code is open source, there is no location specification.
- Flexibility: Customization, hosting, developing, training, and support services or jobs may be obtained from a variety of resources, rather than from a central host.
- Licensing: Open source software eliminates the overhead of these fees, leaving room for spending in the products lifetime costs.
- Modernization: Generally, open source products evolve at a faster rate than licensed products [7], therefore allowing quicker bug fixes, quicker security patches and faster upgrades.
- Product Satiability: Generally, there is better protection from a vendors collapse and the catastrophic consequences that would cause to any facility using the CMS.

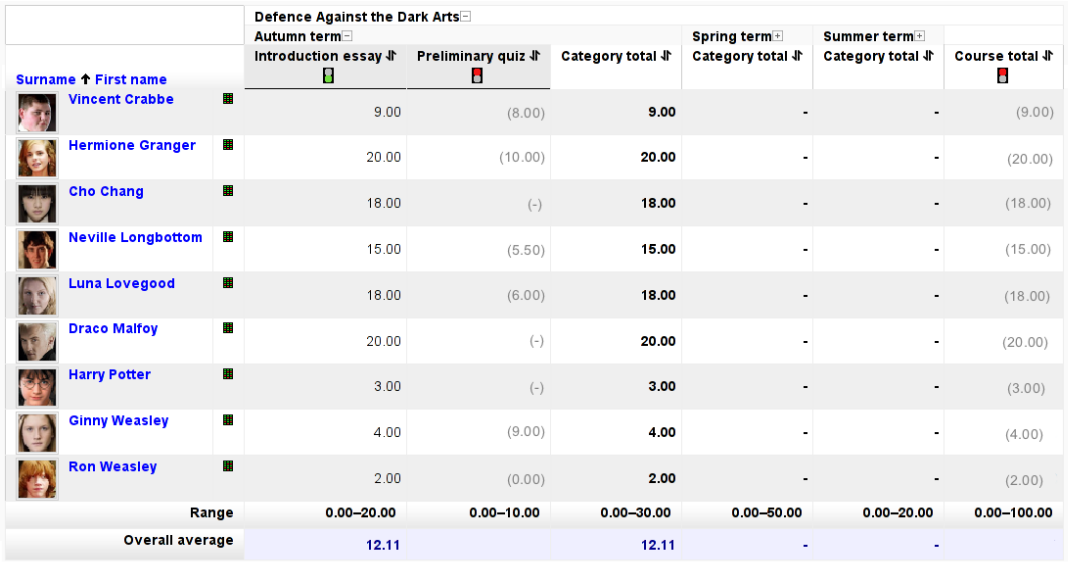
In the following subsection, we explore Moodle according to the aspects listed above, and therefore gain perspective of how the system functions, and if this functionality contributes to or contracts from the usability of the CMS.

Instructor Feedback Moodle provides instructors with a general grade management system. This system, shown in Figure 2.1-2.2 [8], gives instructors an assignment by assignment and student by student method of assessing grades. Completed quizzes and exams are automatically entered into the grade-book. However, if conflict arises, a instructor can go into the grade-book, view the conflicting assignment, and modify the grade. During this modification period, instructors



Name	Aggregation	Extra Credit	Max grade	Actions	Select
UofM test site	Sum of grades		-		All None
New Category	Sum of grades		-		All None
Quiz - 0 points	-		100.00		
Non-graded assignment	-		100.00		
Category total	-		100.00		

Figure 2.1: The simplistic nature of Moodle allows for instructors to quickly modify and update grades on an assignment level.



Surname ↑ First name	Defence Against the Dark Arts — Autumn term: Introduction essay	Autumn term: Preliminary quiz	Autumn term: Category total	Spring term: Category total	Summer term: Category total	Course total
Vincent Crabbe	9.00	(8.00)	**9.00**	-	-	(9.00)
Hermione Granger	20.00	(10.00)	**20.00**	-	-	(20.00)
Cho Chang	18.00	(-)	**18.00**	-	-	(18.00)
Neville Longbottom	15.00	(5.50)	**15.00**	-	-	(15.00)
Luna Lovegood	18.00	(6.00)	**18.00**	-	-	(18.00)
Draco Malfoy	20.00	(-)	**20.00**	-	-	(20.00)
Harry Potter	3.00	(-)	**3.00**	-	-	(3.00)
Ginny Weasley	4.00	(9.00)	**4.00**	-	-	(4.00)
Ron Weasley	2.00	(0.00)	**2.00**	-	-	(2.00)
Range	0.00–20.00	0.00–10.00	0.00–30.00	0.00–50.00	0.00–20.00	0.00–100.00
Overall average	12.11		12.11	-	-	

Figure 2.2: Moodle's viewer shows the grades of each student per assignment and category, allowing teachers the ability to update grades per-student, allowing for a detailed method of modifying grades.

can leave communication concerning an assignment. Instructors are also presented with group communication. This method of communication guarantees that the alert will be delivered to the student [8].

General Communication Moodle provides students and faculty with five main methods of communication, email, instant message, forums, blogs, and chat. Out the five, students and faculty mostly use the instant message and forums communication methods which are the most popular [9]. This form of communication allows for students to communicate in a community like fashion, allowing for a pseudo class chat environment. However, this popularity is negated by the fact that students and faculty use these methods of communication, on average, only fifty-percent of the time [10]. We note that the fifty percent that do not use forums often occurs because course designers do not include this option for the course.

Forum Communication Moodle presents users with a variety or forums ranging from blog to message board style. Users are also given an option for anonymity when submitting questions concerning the course or teacher. There is not, however, a method to anonymously submit responses when replying to a message or a blog entry. This has led to a marginal decrease in forum response, even when grades were associated with forum post completion [9]. Moodle currently does not have a method by which teachers and students may have a live classroom like session together remotely. This led users of Moodle to collaborate with Google Hangouts and other cloud and hosting systems. This draws away from the original purpose of a CMS, to have a succinct modular system to manage courses.

Student Progress/Tracking Moodle gives teachers the grades of each category, then the totals of these categories and non-category items. This sum is then

displayed to the instructor, providing an easy mean by which instructors may view and edit the grade per assignment and per category. The student view reflects this as well [8].

2.2 Benefit Analysis: Edmodo

Edmodo presents a unique form of distance education in the popular form of a social network [11]. This method provides a refined, easy form-based communication between students and teachers. Built on the Web 2.0 standard [12], Edmodo focus on the experience of the student, providing a safe environment for the student to communicate and collaborate with students and faculty. This actually creates better student work-flow, as well as ensuring the development of proper Internet behavior. Edmodo has the appearance of popular social sites such as Facebook, while retaining the monitoring and educational properties of popular CMS such as blackboard [13]. This faux social media site allows for all course and user profile materials to be located entirely on the cloud.

Instructor Feedback and Communication Edmodo functions primary through forum postings [13]. These postings, are similar to Facebook postings, showing on a live feed combined with assignments, forums, and public/private posts. These posts support comments similar to social networks. The functionality allows professors to take individual assignments and comment on each student's post, or send them a private message. This allows a more open-ended method to notify a class of an assignment or any other public event for a student. This also allows for school announcements to appear in a similar fashion. In many campuses where Edmodo is in place, the social interface has become the life-line

of the campus, tying together events, assignments, and social discussions on one platform. Pop [14] attributes Edmodo's popularity with students and overall success primarily to its social network capabilities.

Course Communication Edmodo supports the submission of anonymous information gathering via the poll posts. These posts are placed directly on the live feed of the group/user posting them, allowing their respective subscribers to view and asses the poll. This live-feed nature of Edmodo contributes much to its responsiveness and user awareness.

Unfortunately, Edmodo does lack live communication, restricting teachers to user third party application when in need of a web-class lecture. This overlooked property in the building of Edmodo has led to some universities forfeiting Edmodo for other suites which support this live communication.

Student Progress/Tracking Student grades are viewed in a simple table which may be dumped to a comma separated values (CSV) file. Grades may also be posted from assignments which are not posted on the site. The only down side to this grade-book is the fact that it is very rudimentary. The generic interface offers no more than competing CMS. As for progress tracking, there is no such system for Edmodo. This prevents students from fully utilizing Edmodo.

2.3 Benefit Analysis: Blackboard

Blackboard is currently used by an average of 50 Percent of the U.S. colleges and universities named in Forbes.com's Most Connected Campuses List [15]. Blackboard's usage extends beyond the United States to more than sixty countries, 12 languages, and over 2,200 learning institutions [16]. The black board system is

divided into two components. The first component provides the functionality of managing users, groups, permissions and other administrative functions. Organizations can choose the second component depending on their use case. If they have multiple use cases, they may add another component while maintaining authentication and authorization with the first component. This flexibility allows organizations to use a familiar package across both academic and non-academic areas.

Instructor Feedback and Communication Within Blackboard, there is instructor feedback and automatic grade feedback [17]. These two avenues of feedback allow students to receive a grade for a quiz or an exam faster that with traditional methods. Also if the assignment is a submission based assignment, student can check up on the status of their grade. Alternatively, students can communicate on each others assignments in a central location for a group assignment. This software service increases student/teacher communication, preventing conflicts and other disagreements between instructors and students, increasing overall productivity. Instructors can also receive feedback via Blackboard's survey option [17]. This option allows students to respond anonymously to a set of multiple choice/true-false questions concerning the course. Students generally ignore this built-in solution because it does not pertain to the course content. Also, blackboard does not provide a method for students to 'socialize' outside of the assignments. Schools who do provide a social platform must do so with a separate social network interface[16]. These two detractors do not usually obstruct nor benefit the use of the Blackboard.

Course Communication Blackboard offers forums, announcements, and virtual classroom functionality to enhance communication to and from the students of a course. The announcement feature verifies authorization, and reduces the tedious process of sending out individual emails to each and every student. Discussions provide asynchronous communication with subscribers of a forum. This helps build inter-student relationships while allowing teachers to measure student competency and class attentiveness. The virtual classroom supports a text chat room with live interactions between the participants. However, the supported plug-ins are not generally accessible by all users, thus limiting this method of interacting with the system.

Student Progress/Tracking Blackboard tracks student progress and usage, allowing faculty as well as students a view of their class status. Instructors can measure the success or failure of any assignment. This capability, coupled with the time stamps included, allows for the identification of late or improperly submitted assignments. The instructor may then input the grades into the online grade book, showing the students the grade received within a given assignment. However, there is no method of creating or supporting an analysis tool. This is due to Blackboard being a closed source CMS, hampering the ability for institutions to create and integrate software with their Blackboard instance.

2.4 Benefit Analysis: Desire2Learn

Desire2Learn offers flexibility to its users via an extensively customizable interface. However, as customizable as this CMS is, it is relatively underused, ranking in the lower fifteen LMS/CMS systems[4]. This aspect, however, does not affect the

efficiency and capability of the Desire2Learn CMS. Desire2Learn uses a Learning Tools Interoperability (LTI) compliant architecture, allowing users to configure connections (external or internal) to secure sources by parameterizing the link, allowing for insertion of secure information. When this link is selected, the CMS launches a security checking mechanism which analyzes the link for a proper signature. After this signature is validated, the user is then routed to the selected information. We will explore the pros and cons of Desire2Learn CMS, and therefore gain perspective of how efficiency the system functions overall.

Instructor Feedback and Communication Desire2Learn provides users with the general method of communication between Student and instructor via grading comments, but also provides a social-network framework by which students can intercommunicate similar to Facebook. This encourages students to communicate outside of the classroom, generally increasing student satisfaction, as well as student interactions [18]. However, the organization of the social network interface for Desire2Learn may distract from its original purpose. This is because Desire2Learn's social communication network is not at the heart of the application, thus making the interface slightly unusual. Richard Sertia [18] identifies this as a cause for students to ignore the complication of the site rather than try and learn it. This ignoring may lead students astray from teachers attempting to use this medium of communication, thus taking away from the purpose of the CMS.

Course Communication Users are given an option of anonymously submitting remarks and comments in forums, thus increasing user honesty and the number of participants in the discussion [19]. Also, Desire2Learn Tracks the number of posting and the time spent on each post, increasing the methods by which an

instructor may grade a forum. Desire2Learn provides instructors with analysis tools with which an instructor my view the percentage of participation. However, there is no interface for face-to-face meetings. If Desire2Learn supported a voice stream with student live text response, at least there would be some form of student awareness.

Student Progress/Tracking Apart from the user usage statistics previously mentioned, Desire2Learn provides a method for the teacher to view all of the grades of a current student. This student view can then be expanded to allow an instructor to analyze the student in parallel with the other students within a class. Desire2Learn also provides student statistical analysis for the viewing of the instructor [20, 21]. This tool allows the instructor to forecast the chance of success of a student, the success rate of the class, and the areas in which his/her students rate content up or down. Students are also provided with this view. In the student view, the analytic tool provides information from a course level (the success rate of certain assignments within a course) to an academic level (the success rate within a certain course). There is not, however, a method by which users can view projected grades, or assignment difficulty.

Chapter 3

Project

This project added a plug-in called “Project Grade-Up” (PGU) to the open-source LMS platform Moodle. This project contains several sub-systems to keep our system in a Moodle friendly format. Since a large amount of our code consists of ”interfacing” code, we concentrate on the calculations of the burn-up chart and heat map functionality.

3.1 Architecture

Our project follows the Model View Controller pattern, allowing for a separation of logic from view functionality. Following this paradigm also allows us to improve the user experience via the use of Asynchronous Java-script (AJAX). The user experience, therefore, is not interrupted for the posting or fetching of data to and from the server.

The MVC pattern depends on a layer of abstraction added within Moodle called the Data Control Layer. This control layer, as seen in Figure3.1, acts as an interface to the underlying data in Moodle. Consequently, this Layer allows our

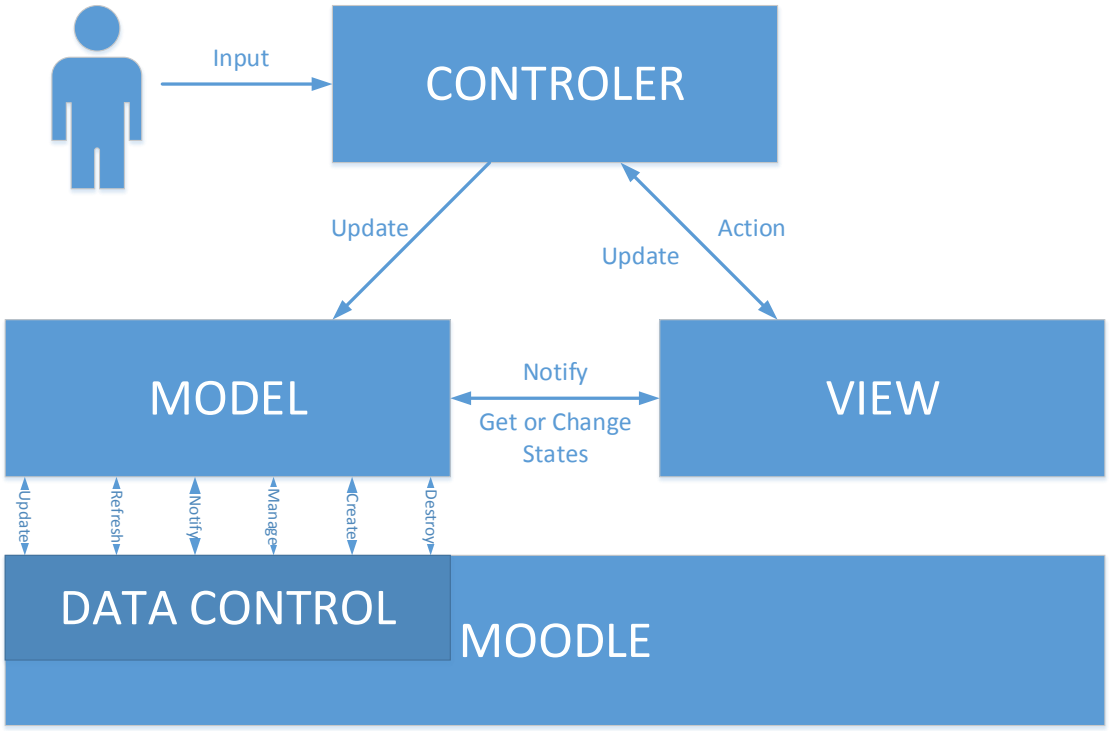


Figure 3.1: The Data Control Layer lies "within" Moodle, allowing for an abstraction which will assist future developers in adapting our solution.

solution to easily adapt to other LMSs. To adapt our solution to other LMSs it is only necessary to implement the data control layer. This design also adheres to Moodle coding practices and principles, and ensures modularity throughout the added code.

Moodle's performance may be viewed through the tools XHProf and XHgui, both of which are components for viewing PHP web-page profile runs and general performance. In a given page execution, Moodle runs several low level functions regardless of the function being run. This is due to Moodle's extreme modularity, which while integrated with the Zend framework modularity mindset, allows

individual components of code to be extremely modular. Even though modules are being called multiple times in separate calls, this modularity reduces space on the server code-wise and also improves the overall performance.

This modularity, however, comes at a price. Documentation in Moodle, as in some open source solutions [22], tends to be lacking. This absence, coupled with the extreme modularity, leads to a situation where developers have used slightly different standards than core developers. This lack of standards within plug-ins and extraneous bodies of code leads to further divergence from prescribed standards and tends to introduce developer errors [23]. This divergence has prompted a need for a re-write of the documentation, some code within Moodle and several large plug-ins This moves the development towards a more standard development approach. This revision is a part of Moodle 3.0, released November 16, 2015 [24].

The methods used within our solutions follow Moodle's coding standards [23]. Within Moodle, there is defined separation between different types of plug-ins. For example, modifications and additions to core student activities are called modules, visual and minor functional changes to the user interface of Moodle are called themes, and local unique-to-campus plug-ins are called locals. For the development of our solution, we felt it best to use the block paradigm of Moodle plug-ins. Using this paradigm allows for a more generic access model than using the report or module plug-in, as these plug-ins are not only vetted and scrutinized at a more granular level, but also Student access to a report would break that paradigm.

Moodle's block development [25] follows a core set of instructions. These instructions call for an adherence to the MVC pattern for the development of blocks. Blocks use a separate language API integration (e.g. English, French,

etc.), core user type authentication, REST service creation, and a centralized client-resource management system. We follow this pattern throughput the entirety of this plug-in development because adhering to the standards grants a better rating on Moodle. This project builds a Model View Control (MVC) block plug-in, with the individual components as follows:

- The *Data Control Layer* provides an abstraction of the underlying Moodle data for use in our plug-in.
- The *Model* Interacts with the Data Control Layer to provide a Model for the MVC pattern.
- The *Control Layer* Manages interactions between the view and Models.
- The *View Layer* provides visualizations using data retrieved from the Control Layer.

Figure 3.2 shows an approximation of the amount of work required in each of the above layers. The following sections discuss each of these layers starting with an overview of the architecture.

3.1.1 Data Control Layer

The Moodle LMS houses the Data Control Layer and manages the data needed to build and display charts and graphs for the two proposed features. We use Moodle's internal data management system and APIs to gather data for the Data Control Layer. The benefit from using the existing infrastructure in the Moodle API is that there is no need for regression testing. This means that when Moodle updates, it will remain backwards compatible []. This also means the Model does

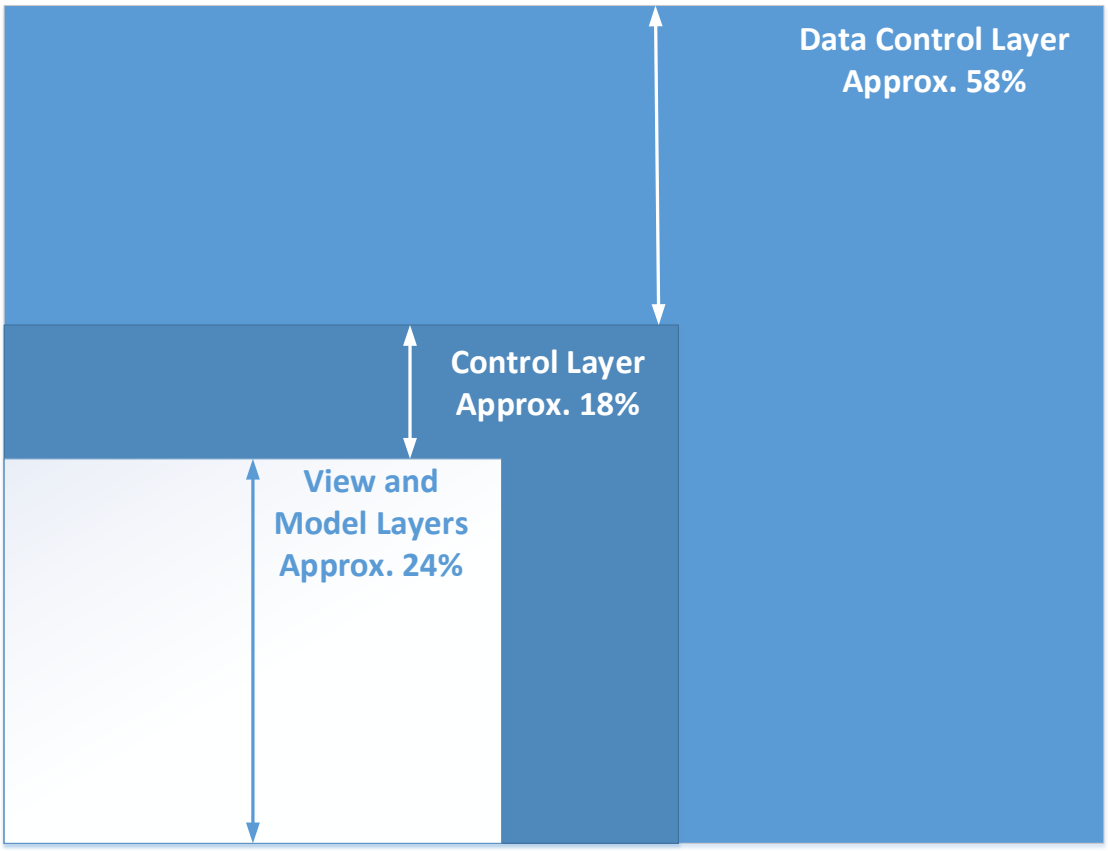


Figure 3.2: The greater percentage of our code lies in Moodle specific code within the Data-Control Layer.

not need to *know* the location or format of the data, due to Moodle handling this interaction [proposal ref 12]. This layer provides several functions within the lib.php library. The purpose of these functions is to extract the data from within Moodle's databases and format the data into PGUs tables. Figure 3.3 shows how our Data Control Layer interacts with Moodle APIs which in turn abstracts the data from the Moodle Data Layer.

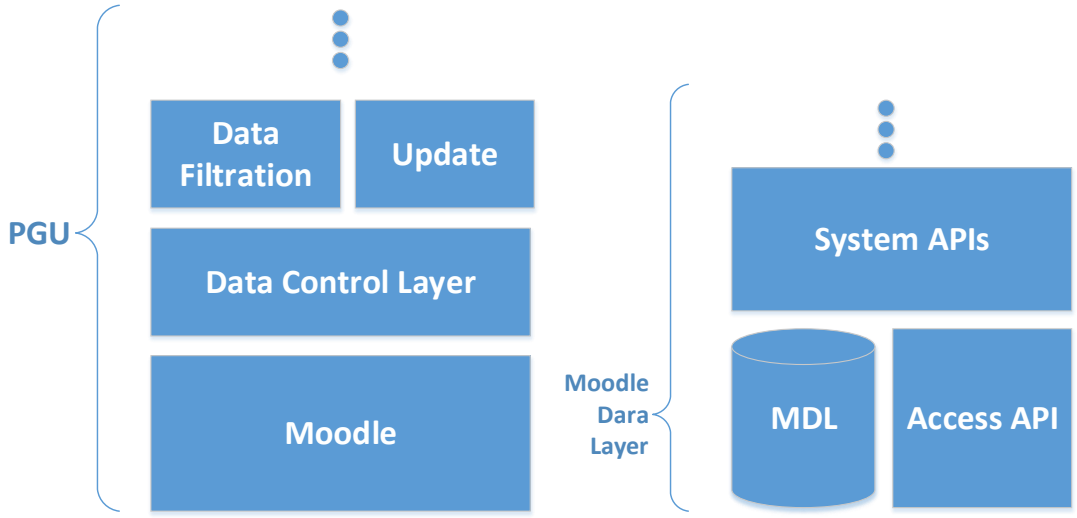


Figure 3.3: The Data Control Layer allows an abstraction of Moodle's data layer, allowing for a more general interface of our Model Layer with the rest of the plug-in

3.1.2 Model

Due to the complexity of the data needed for the visualizations, we created a separate Model from the Data Control Layer in-order to abstract the intermediate queries between the data control layer and the Control Layer. This Model layer contains the intermediate database tables and classes to interact with that data. The .php files contain classes that the controller uses to manage requests to the tables. The layout of this layer can be found in Figure 3.4.

Figure (Figure3.5) shows the schema of the tables that hold the aggregate

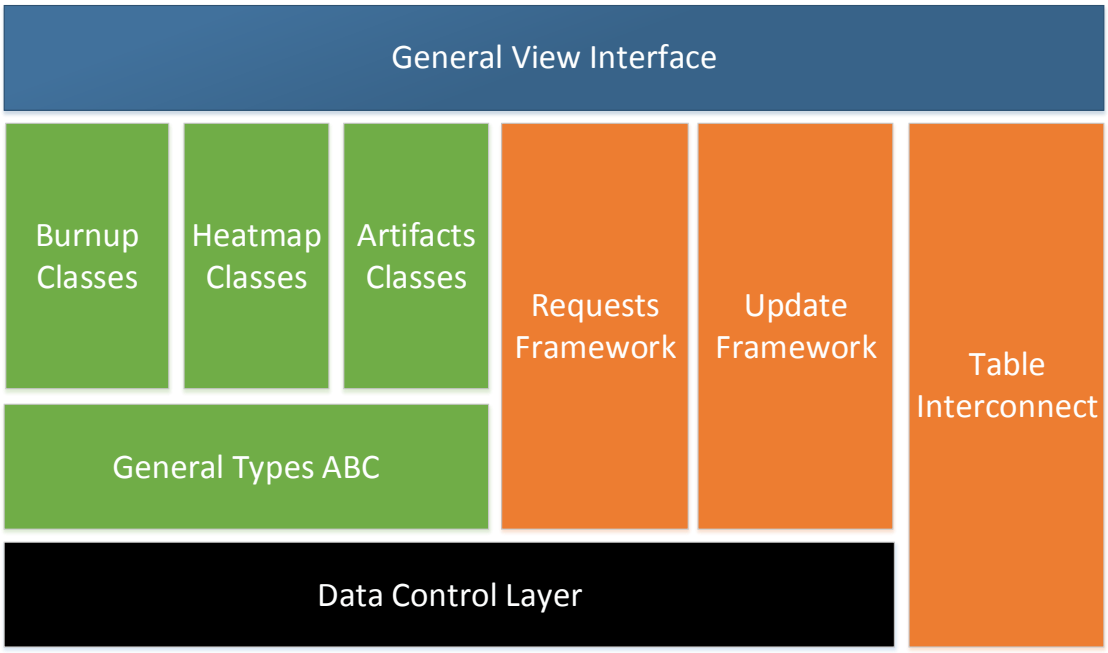


Figure 3.4: The Model Layer consists of a combination of a intermediate database and the associated classes. The classes that exist in this Layer are modeled after their respective tables.

data retrieved from the data control layer. This aggregate data comes from costly queries in Moodle. Consequently the data stored in these tables provides quick access without running those costly queries in Moodle each time a user interacts with our plug-in

The plug-in also uses Moodle's services manager, which allows the plug-in to update the intermediate tables at regular intervals.

The Model also caries all base data structures. These are the artifacts, artifact_date_time, class_date_time classes etc. These individual classes provide the

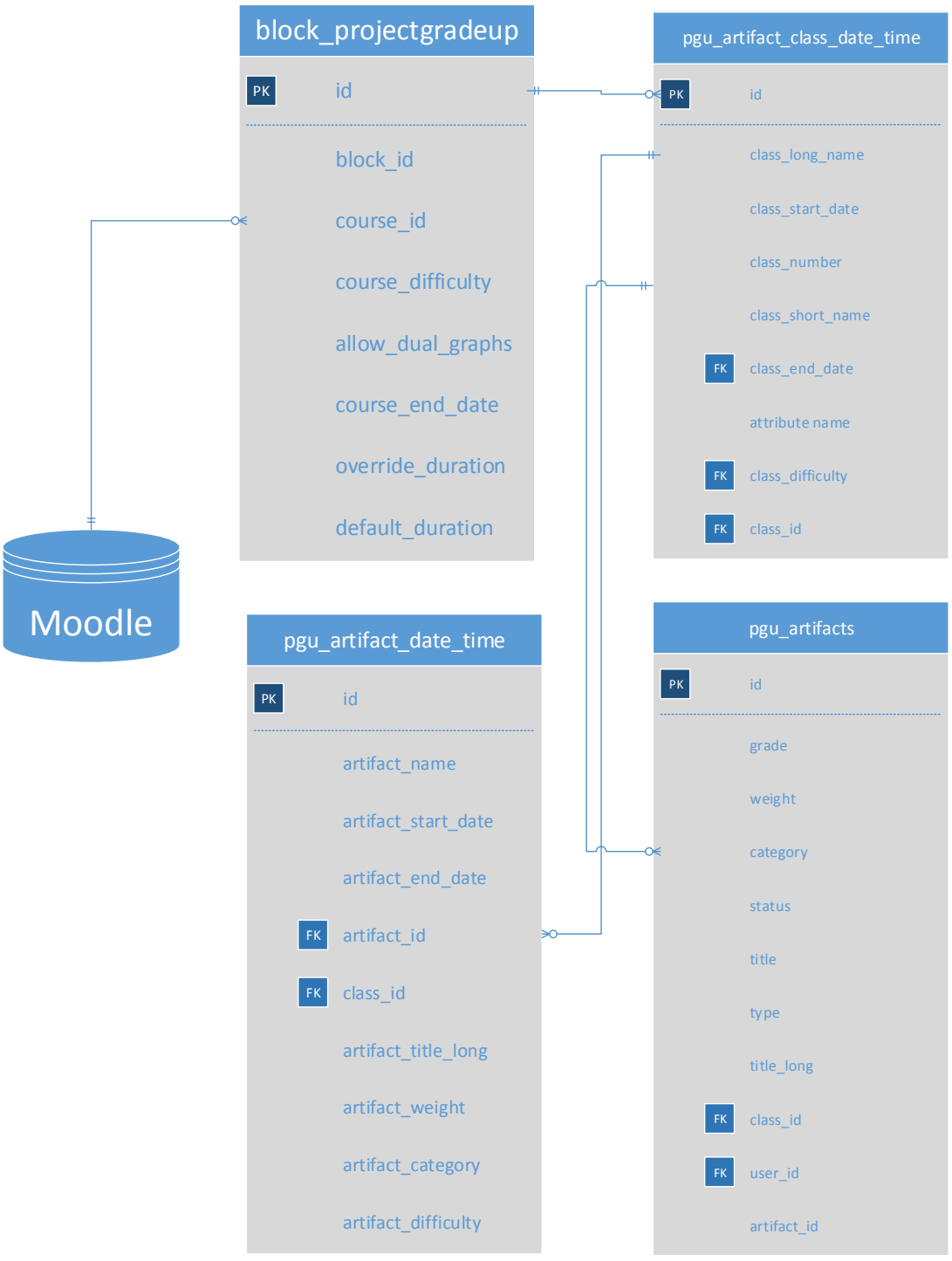


Figure 3.5: The schema of the pgu_tables is such that all objects handled by the Model can be instantiated without queries to any other table besides itself.

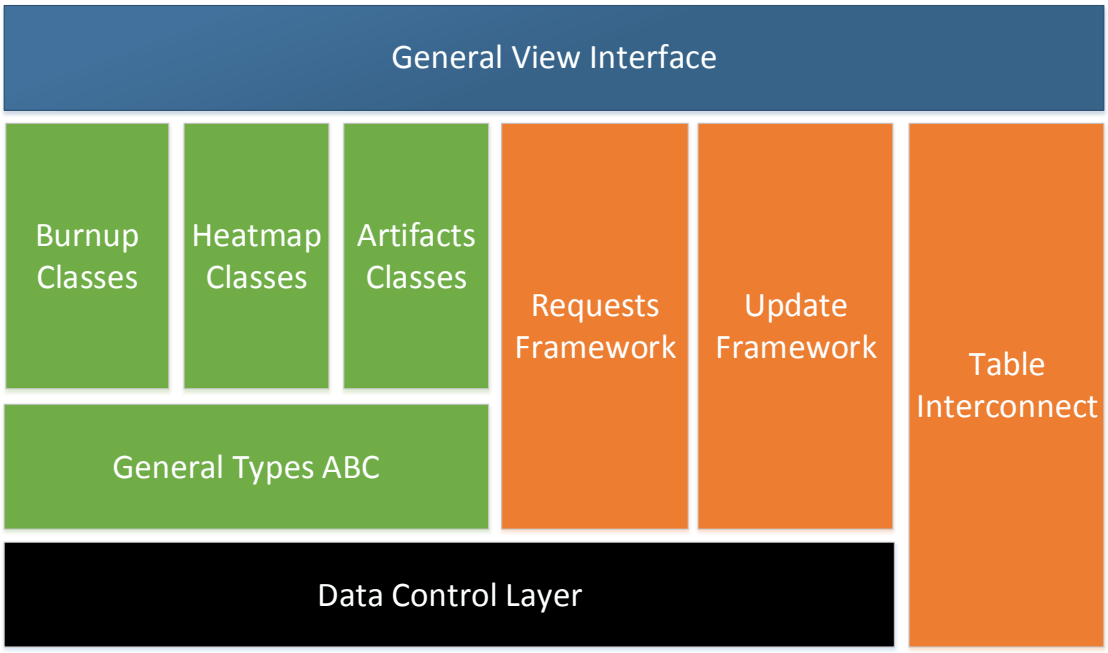


Figure 3.6: The Control Layer contains a sum of interface code, but the majority of the code is contained in the update and Abstract Classes (ABC) and its inherited children.

functionality for the graphs, and provide uniform interface code that allows separation of concerns when building the view and controller layers.

3.1.3 Control Layer

This layer handles all data interactions between the Model, and View Layers. First, it must update all data within our Model. Second, it must relay any and all view requests to the Model. Finally, it must update the view upon user requests. This central Control Layer allows the abstraction of nearly all *business and session logic*

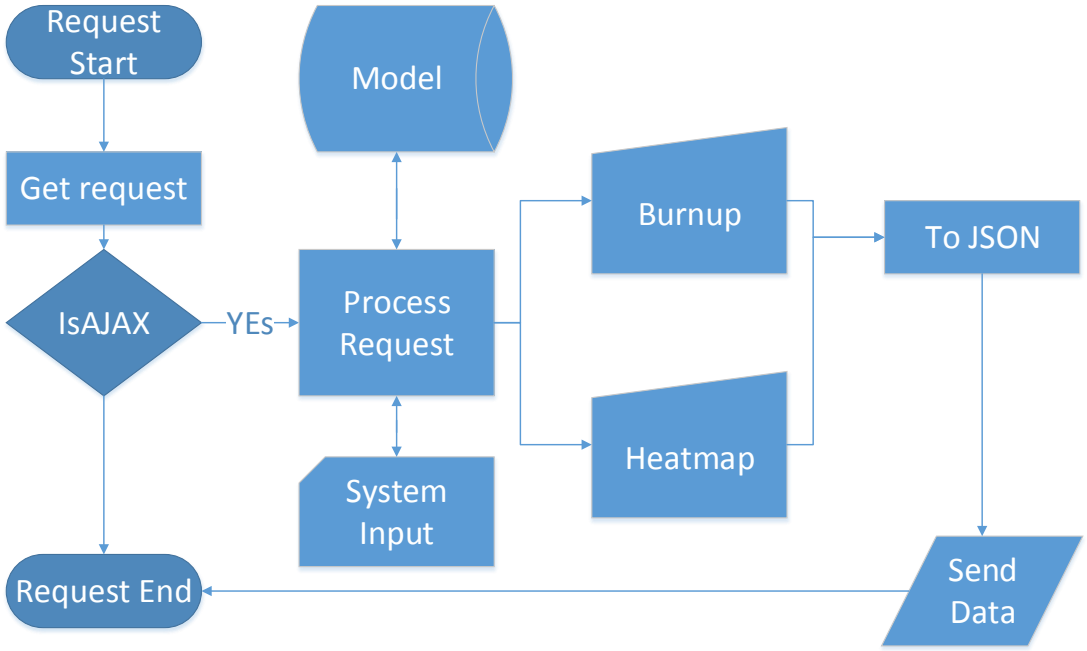


Figure 3.7: A simple diagram demonstrating a request within PGUs Control Layer.

from our View Layer, keeping our code clean and readable. This layer is in two separate components: the table updating and REST service components. Figure 3.6 shows the controller components in orange.

The table updating component contained within this layer consists of two individual components: scheduler and triggered events. The scheduler functionality in this layer directly configures the scheduling of the underlying LMS through the data control layer. The REST service components consist of the Burnup and Heatmap classes, each written to facilitate the transformation the data into JSON which is consumed by the client side. The client side manipulates the data and displays it in the visualization. Client side code handles as much of the user events

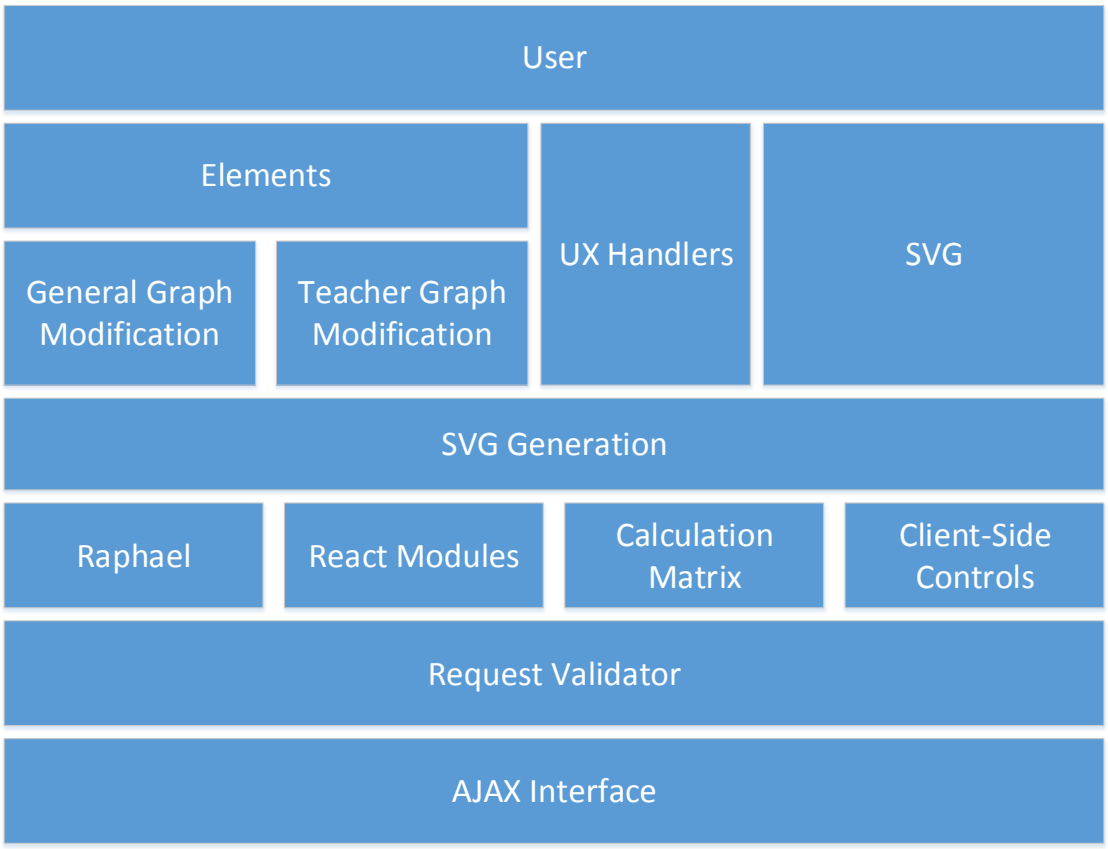


Figure 3.8: The View Layer contains all user experience and interactivity functionality. The layers are arranged as to provide maximum modularity concerning the passing of data as to not overload the client.

as possible without making round trips to the server.

3.1.4 View Layer

The View layer handles all graphics display logic. This includes the PHP views and JavaScript libraries. We would like to note that our layout depends on the coding standards of Moodle [26]. This layer is broken into several sections:

- *View* - This module consists of the encapsulating .php pages and injected HTML 5 elements

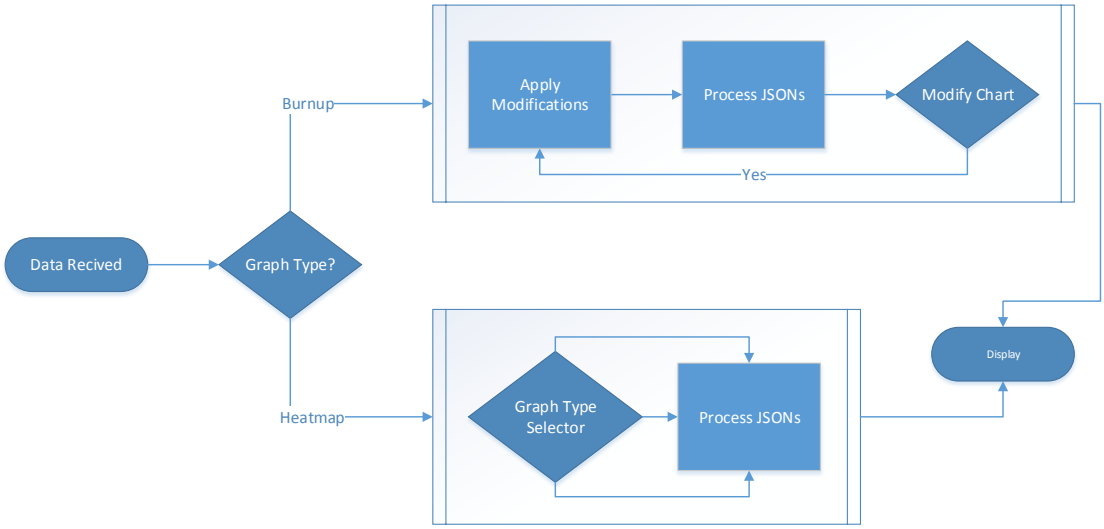


Figure 3.9: This sample process shows the slight difference between the Heatmap and Burnup SVG Generation processes. Users can only interact with the Burnup chart SVG directly, hence its shorter logical path.

- *Element Controls* - This module controls the interactions with the elements within our solution
- *AJAX* - This module controls all request from the client to the server, including client-side parameter sanitizing
- *SVG Generation* - This layer handles all client side SVG generation

The view component consists of the .php page itself, the injected HTML elements, and the theme (Styling). The Element Controls are simply elements

which generate a user input method. AJAX handles client side sanatization, requests data, and parses results. The final segment consists of the Raphael SVG generation engine and our custom component which handler all graphics creations and interactions. Figure 3.8 shows the View Layer. There is a slight difference in how the Heatmap and Burnup SVG Generation methods work. As outlined in Figure 3.9, the Heatmap does not have any direct user interaction, hence the difference in code base between the Heatmap and Burnup charts.

3.2 Charts

This project mainly focuses on the graphics, that is the method of displaying the useful data to a set of users. These graphics are engineered to properly, efficiently, and correctly display student grade and difficulty data. There are two types of graphs created for this project:

- A Course *Burnup Chart* (adapted from agile programming) that shows the grade state, predictions, and possible outcomes.
- The *Load Heatmap* visually shows work loads during the current semester for one or more courses

Our sub systems provide all the data manipulation, filtration, and 'business logic,' see the above sections for more information on these sub systems.

3.2.1 Burnup-Chart

Within our Burnup-Up chart, a user may view several key pieces of data simultaneously. These pieces of information pertain primarily to the grades and current

progress of the selected course. Each of these pieces of data answer a specific question:

- Assignment completion - What have I completed, and what do I have left to complete?
- Current grade trajectory - What is my current grade? That is, what will my final grade be if I continue at my current success rate?
- Best possible final grade - Given what I've done so far what is the best possible grade I could get if I ace the remaining work?
- Final Grade with no additional work completed - What if I stopped now, what would my grade be?
- Individual assignment weight - How much does each artifact contribute to the final grade?
- Grade Projection - If I got a particular grade(s) on a specific assignment(s) how would that change the preceding data?

Figure 3.10 shows our completed graphic as it appears in the plug-in Notice the differences between assignments which are due (*assignment completion*) and those which are incomplete.

Logic Consider the chart as the upper right quadrant in an x-y plane. The x-axis represents the total percentage of points due. The left most position on the x-axis represents 0% work due. The right most position on the x-axis represents 100% of the work due. Although the artifacts required will be in chronological order, the x-axis does not correspond to time but instead shows the percent accomplished

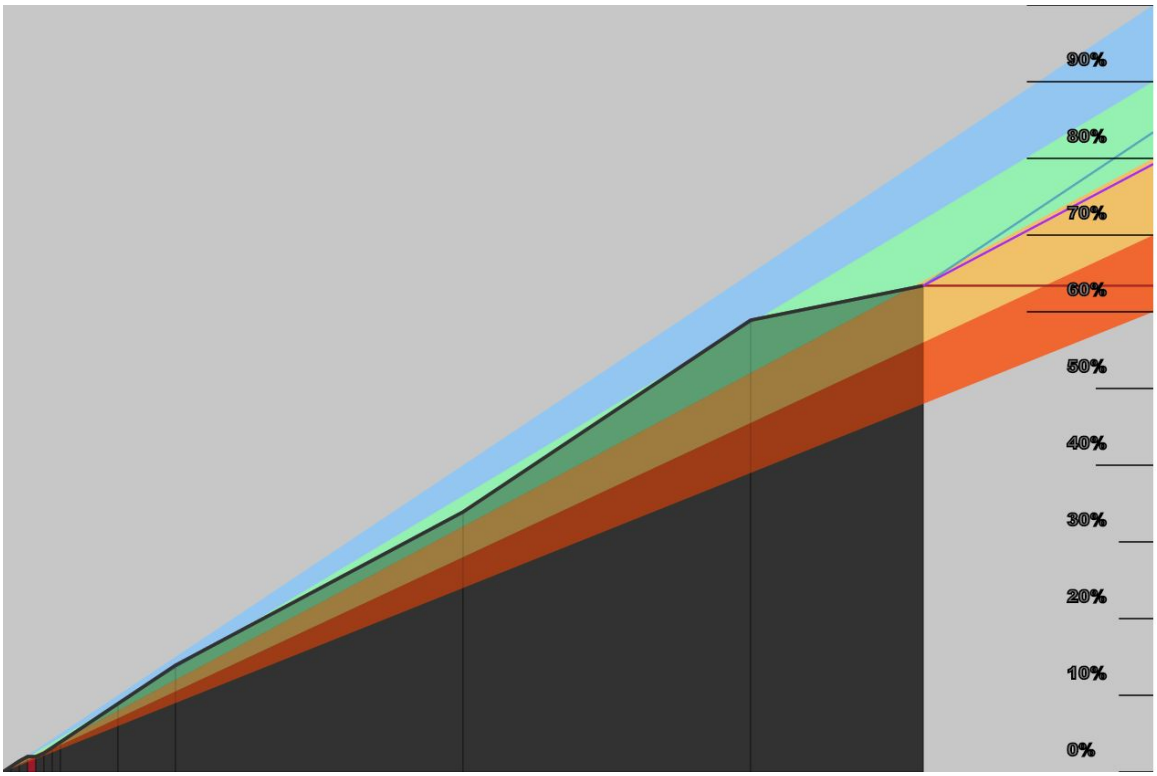


Figure 3.10: This is the completed Burnup chart. The projections (far right) represent the projected grades of the logged-in user.

and weight of assignments. The y-axis represents the percentage of points earned in the course. Clearly a student who earn 100% on all their work would see a line having a slope of 1 from $(0\%, 0\%)$ through $(100\%, 100\%)$. The scale of the axes presented differ between the x-axis and the y-axis so that the slope does not appear to be one, however the mathematics involved need not account for this disparity.

The black line shows the *current grade trajectory*. The current grade trajectory line starts and the origin $(0, 0)$ and goes through the last assignment grade plotted on the chart. The grade point is the percentage earned as of the last assignment due designated c. Let a_i represent the i^{th} artifact and w_i be the weight and p_i be the percentage earned for a_i. The grade point is defined by percentage completed

and the percentage earned. The percentage completed is given by

$$Percentage_{completed} = \sum_{i=1}^{c} w_i \tag{3.1}$$

The percentage earned is given as

$$Percentage_{earned} = \sum_{i=1}^{c} w_i \times p_i \tag{3.2}$$

The grade point is thus defined by

$$(Percentage_{completed}, Percentage_{earned}) \tag{3.3}$$

The line from $(0, 0)$ through $(Percentage_{completed}, Percentage_{earned})$ gives the black, grade-trajectory, line shown in Figure 3.10.

The green line shows the *best possible final grade* that the student can achieve given the work completed (or missed) up to this point. It starts from the last grade given defined by the point $(Percentage_{completed}, Percentage_{earned})$ and has a slope of 1. The y value at $x = 100\%$ gives the best possible grade.

The red line shows the *final grade with no additional work completed*. The line starts at $(Percentage_{completed}, Percentage_{earned})$ and continues to the right most border with a slope of 0. The y value at $x = Percentage_{completed}$ gives the final grade with no additional work completed.

We omitted the *class average* overall represented by a blue line. Clearly this class average must be optional for small classes. This line consists of an average of the current grade trajectory lines from all students from within that specific class. If

this is enabled, we define the slope of this line as

$$Class_{Avg} = \sum_{i=1}^{c} Percentage_{Earned_i} / \sum_{i=1}^{c} Percentage_{Completed_i} \tag{3.4}$$

where c is the number of students in the class.

Figure 3.10 also shows the weight of each assignment visually represented by its width. Widths depend on the category weight and the number of artifacts when assignments are equally weighted. However, no mater what system one uses for calculating grades a method can be used to assign a weight to each individual artifact. For example, the assignment category with a weight of $Category_{weight}$, and a number of assignments $Category_{count}$, the individual assignment weight is given by

$$Assignment_{weight} = Category_{weight} / Category_{count} \tag{3.5}$$

Here, $Assignment_{weight}$ gives the percentage of the X-axis that a particular assignment occupies. All artifact together will then add up to 100% and fill the X-axis of the Burnup chart.

Algorithm Layout The Burnup algorithm primarily uses client side computation and Raphael.js for the production of charts using Scalable Vector Graphics (SVG). The code for this graphic is primarily within the client side code. The algorithm begins by requesting the artifacts for a course. These course artifacts are then processed into the individual components of the graphic; the artifact polygons, projection lines, grade regions, and finally the trend line. These components then are individually formatted into JSON objects before being compiled into a large string for transmission.

After the calculations for the graphic are completed, the code is then passed

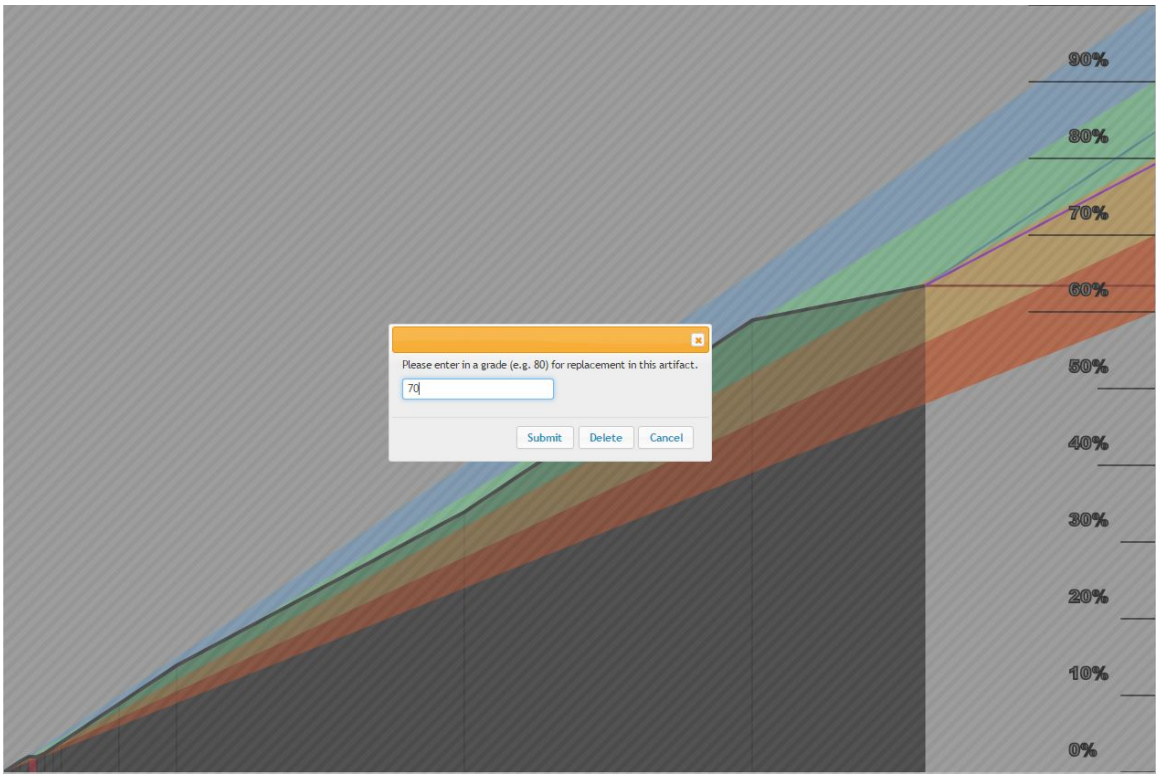


Figure 3.11: Within the 'what if' pop-up, the student may enter in a grade to overwrite the grade in the current artifact, or delete the current artifact. Any and all changes will persist until the user has refreshed the page.

to the displaying logic, which simply parses the Raphael formatted JSON string and sends this information to the display div ([reference an image for this process that encompass both Heatmap and Burnup charts]). For more information on this process, please refer to [reference to Appendices with Doxygen documentation].

Example Figure 3.11 shows a student using the grade projection feature to play out a what if scenario for the final exam. Notice how this compares to the first Burnup Chart. In the second Chart, the student can clearly see that a score of 59% is needed to maintain a 'B' in the course. The student may also use the grade projection feature to lower his grade (or skip it altogether) to see how it would impact his overall grade projection in the course. Also, the student may

use assignment drop option. Delete will allow students to drop an assignment without receiving a zero for the assigned dropped.

3.2.2 Heatmap-Chart

Our Heatmap-Chart is primarily focused on the displaying a student's progression through a course, along with the difficulty per day. This chart contains jagged triangle-like structures which represent the difficulty over set increments of time. These increments are determined by either the start and due dates of the artifacts, or the overall difficulty of an artifact. The finial component of these charts is the annotations signifying artifact due-dates.

Equations Consider an x-y plot where the x-axis represents time over the interval $[t_s, t_e]$ and the y-axis represents a work load in the interval $[0, l_{max}]$. A piecewise linear function $L(t)$ represents the work load at any time $t \in [t_s, t_e]$. Students may view $L(t)$ directly but we will give preference to a simpler single, horizontal bar chart that varies in color over time. For the preferred chart the function $C(L(t))$ maps the load intensity to a color spectrum.

Artifact i affects the magnitude of load by an amount l_i based on a heuristic function which takes the type of artifact, the percentage of the grade that artifact contributes to the overall grade, and the level of the course the artifact comes from. We define l_i as follows.

$$l_i = H(Type(A_i), w_i, Level(Course(A_i))) \tag{3.6}$$

Artifact i due dates and times appear on the time line and affect $L(t)$ for an interval of time $[t_{s_i}, t_{e_i}]$ before it's due date. The length of the time interval

is determined by one of several things. First if the assignment is assigned on a particular date (e.g. in Moodle if a start date/time is defined), then we will consider that the artifact starts to affect $L(t)$ at the assigned date/time. Otherwise we will need to determine time interval length based on a heuristic function based on the following elements.

- Last due date of an artifact in the same category $DueDate(A_{i-1})$.
- Weight w_i of the artifact as a percentage of overall score
- Preparation time $PrepTime(A_i)$ as indicated by a student or teacher.

Whatever heuristic function we develop, it must produce a the time interval $[t_{s_i}, t_{e_i}]$.

Now we define an artifact load function for artifact i by

$$L_{a_i}(t) = \begin{cases} m_i t + b_i & t \in [t_{s_i}, t_{e_i}] \\ 0 & t \notin [t_{s_i}, t_{e_i}] \end{cases}$$

where m_i is the slope defined by $\frac{l_i}{t_{e_i} - t_{s_i}}$ and b_i is a constant to raise the line at the starting time to 0.

Finally we can define the mathematical expression associated with each interval i in the piecewise linear function $L(t)$ as follows.

$$L(t) = \sum_i L_{a_i} t. \tag{3.7}$$

Algorithm Layout The Heatmap algorithm, unlike the Burnup algorithm, keeps all business logic on the server. We keep logic for Heatmap computation on the server in order to save client resources. We do not have any need for keeping business logic on the client because the graphic is not editable, unlike our Burnup

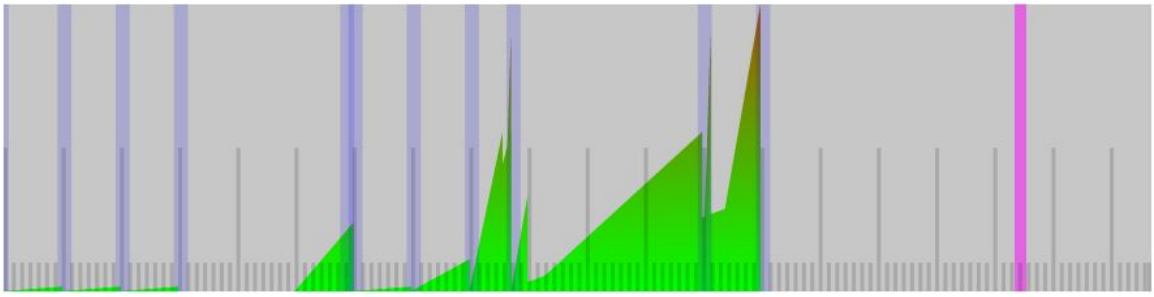


Figure 3.12: This is a sample Heatmap The purple bar is the users current position chronologically in the semester, with the blue intermittent bars representing individual artifacts.

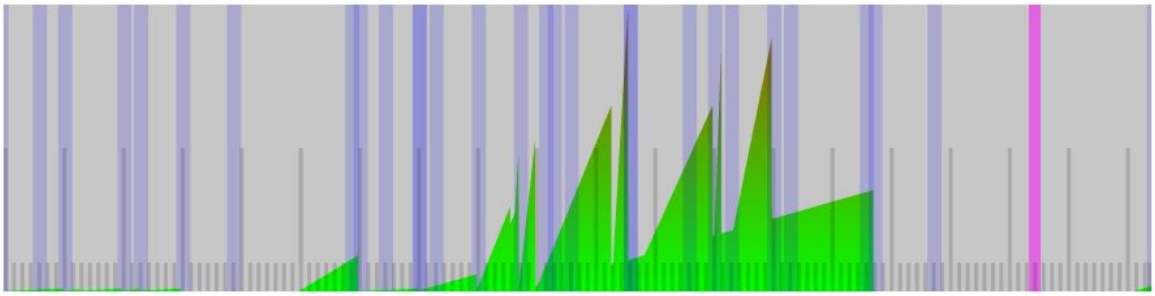


Figure 3.13: This Heatmap spans across two classes within the semester.

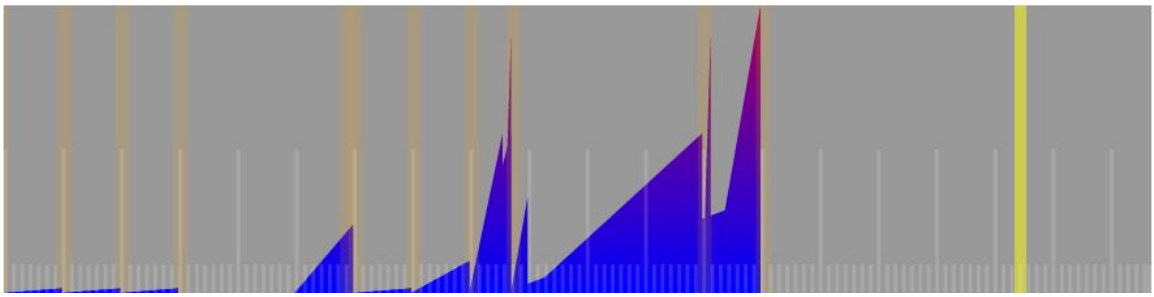


Figure 3.14: This Heatmap has vision assistance enabled, changing the color palate into one who should be able to recognize these colors better than the alternative.

chart. The algorithm selects the artifacts, artifact date time, and class date time items for the selected course. These data types are used to compute the graphs different graphical objects. The data is then sent to the client display logic, near identical to Burnup display logic.

Example Consider the example list of assignments shown in Table 3.1. The table contains added columns to show the category weights and indented individual weights. Figure 3.12 shows a sample load Heatmap visually representing how the work load for the same data as Figure 3.10. The vertical blue bars represent the various artifacts of the course. The student now may view not only their current position in the semester, but also the up and coming difficulty as the semester progresses. This allows the student to plan ahead for harder assignments not only in the class which they currently are viewing, but across their classes as a whole (see Figure 3.13). The students current progress is represented by the purple bar. The Heatmap also includes an assisted viewing feature, for those with visual impairment (see Figure 3.14). However, being that there are many types of visual impairment, users can view the peaks Heatmap chart regardless of their impairment.

3.3 Summary

This chapter described the project solution from architecture through the Burnup and Heatmap charts. It detailed the logic and math necessary to reproduce the results shown and matched real data to show examples of both types of charts.

Category	**Assignment**	**Weight**
Exams		**35**%
3/12/2014	Midterm Exam	17.5%
4/23/2014	Final Exam	17.5%
Presentations		**10**%
3/19/2014	Paper Presentation 1	5%
4/2/2014	Paper Presentation 2	5%
Homework/Discussions		**30**%
1/15/2014	Assignment 1	4.28%
1/22/2014	Assignment 2	4.28%
1/29/2014	Assignment 3	4.28%
2/5/2014	Assignment 4	4.28%
2/12/2014	Assignment 5	4.28%
2/19/2014	Assignment 6	4.28%
2/26/2014	Assignment 7	4.28%
Projects		**25**%
3/26/2014	Campus Project	6.25%
4/2/2014	Archaeology Project	6.25%
4/9/2014	Voting Districts and States Project	6.25%
4/16/2014	Who are my Senators and Representatives Project	6.25%

Table 3.1: The assignment chart showing the weights associated with the assignment.

Chapter 4

Testing and Evaluation Results

This chapter reviews the software requirements, the evaluation process to test those requirements, and the results of that evaluation.

Our evaluation relies on a simple pass/fail process. When possible, we used unit tests using PHPUnit [27] for this purpose. We verify visual component requirements through a visual inspection of the completed application and show a check sheet that lists the requirements and their completion status.

4.1 Unit Testing

Unit tests focus on testing the functionality of the block plug-in added to the Moodle application and not on existing Moodle code. Table 4.1 lists all the unit tests and the functionality that those unit tests covered. Each test tests multiple pieces of functionality on a single class. Some unit test classes such as the tests covering the `lib.php` functions, covers a large set of functions in one unit test class.

Class Tested	Test Information
`ClassAbstracts()`	This class is the base class for all other classes that we built for this project. `Test_ClassAbstracts()` tests the common properties of the class over a variety of valid and invalid instantiations.
`DefinedDataLayer()`	This class provides functionality for the data layer. `Test_DefinedDataLayer()` also tests the common properties of the class over a variety of valid and invalid instantiations
`BurnUp()`	This class provides functionality for formating and serializing the data from the data layer to the client. `Test_BurnUp()` tests `getBurnUp()` by validating the format of the five different components of the chart returned. These include grade artifacts, grade regions, trend lines, projects and grade level markers.
`BurnUp()`	This class provides functionality for formating and serializing the data from the data layer to the client. The `Test_BurnUp()` class tests the function `getBurnUp()` by validating the format of the five different components of the chart returned. These include grade artifacts, grade regions, trend lines, projects and grade level markers.
`HeatMap()`	This class provides functionality for formating and serializing the data from the data layer to the client. The `Test_HeatMap()` class tests the function `getHeatMap()` by validating the format of the five different components of the chart returned. These include chart polygons, the time line markers, artifacts labels, and current time marker. Polygons include cutout polygons defining the various difficulty regions across time.
`AJAX()`	This class provides communication functionality between the client and the server. The `Test_AJAX()` class tests instantiation with both valid and invalid parameters. It also tests requesting a new page, and changing the representation of graphs for colorblind individuals.

Table 4.1: List of features successfully unit tested.

Class Tested	Test Information
`lib.php`	This collection of functions contains all the data creation functionality described in Section 3. Much of the 5000+ lines of code contained here pulls data directly from Moodle, using Moodle APIs. The rest are functions that calculate the information needed to create the charts. We test the following functions • `Block_ProjectGradUp_Update_PGU_Artifacts_Update()` updates the artifact information that the charts display. • `Block_ProjectGradUp_Get_Grade_Categories(int CourseId)` returns an object containing all of the artifact information created in a course. E.g. this would include all categories, weights, artifacts (like assignments) in those categories etc. This has the majority of the calculations described in Section 3. • `Block_ProjectGradUp_Get_AggregationCol(int CatogoryId, int CourseId, int ParentCategoryId, float WeightOne, float WeightTwo, int ArtifactId)` calculates the weights of individual artifacts. provides functionality for formating and serializing the data from the data layer to the client. The `Test_BurnUp()` class tests the function `getBurnUp()` by validating the format of the five different components of the chart returned. These include grade artifacts, grade regions, trend lines, projects and grade level markers.

Table 4.2: List of features successfully unit tested Continued.

4.2 Visual Checklist Evaluation

Visualization features were checked through examining a running test system. Figure 4.1 shows the features successfully tested. Examples of these figures can be seen in Section 3 in Figures 3.10-3.14.

- ✓ Burn-Up Chart is available.
- ✓ Burn-Up Chart shows:
  - ✓ Assignment completion
  - ✓ Current grade trajectory
  - ✓ Best possible final grade
  - ✓ Final Grade with no additional work completed
  - ✓ Grade Projection
- ✓ Load Heat-Map is available for a class.
- ✓ Load Heat-Map is available across all participating classes.
- ✓ Load Heat-Map shows:
  - ✓ Current class load through out the current semester.
  - ✓ Assignment placement within the heat-map to convey relative difficulty.
  - ✓ Schemed color gradient to properly convey areas of congestion within a semester.
- ✓ UI contains Burn-Up and Heat-Map charts appropriately.
- ✓ UI Interacts with Burn-Up chart features where necessary:
  - ✓ Hover over data is displayed properly.
  - ✓ Class add/drop functionality works properly and efficiently.

Figure 4.1: Visualization checklist.

4.3 Side-effect Testing

The functionality added to Moodle in the Project Grade-Up block plug-in should be side-effect free [28] to the original Moodle data. Thus even though we do store data relevant to the block plug-in, it never modifies any original data. The Raw Data tests ensure that PGU code does just that [29].

We completed our data tests with relative ease due to PHPUnits method of asserting a databases state to be the same as prior to the test. These tests show that our solution has no side-effects on Moodle's core tables. The following list shows the unit tests responsible for side-effect testing.

- `Test_DataDefineLayer()`
- `Test_AbstractClasses()`

4.4 Evaluation

Unit tests and visual checklist evaluation shows that our solution meets the requirements. Performance also plays a large role in the adoption of this plug-in.

Performance evaluation showed that this plug-in block performs at a level common to other blocks. Our performance evaluation did lead to storing aggregate data for users in temporary tables for decreased wait times on data requests.

Chapter 5

Conclusion

There is currently a demand for a system which can relay progress information to students within a given course and semester. Current solutions confuse users with obscure graphs, reducing the impact of such a system. Also, the entire range of data is not available to students, further preventing the larger impact a grade analysis tool should have.

In this thesis, we implemented an addition to the Moodle LMS to remedy the above issues. Based on our research, we postulate that the use of our solution may significantly increase student awareness of time constraints, as well as help them prioritize and efficiently schedule study times. The presentation of a burn-up chart's indicating progress and the Heatmap's presentation of time-per-artifact are several features which contribute to student efficiency. Therefore this solution may be of great benefit to the students as well as the teachers, increasing communication and student understanding of progress significantly.

For future work, we suggest a week-at-a-glance feature to the Heatmap, artifact links in the burn-up, and an advanced analysis and research project on the education benefits of this Moodle plug-in.

Bibliography

[1] M. Roqueta, "Learning management systems a focus on the learner," *Distance Learning*, vol. 5, pp. 59–66, 2009. 1

[2] J. Monks and R. Schmidt, "The impact of class size and number of students on outcomes in higher education," *In Real Life (IRL) Collection*, vol. 12, pp. 1–23, 2010. 1

[3] W. M. Reynolds and G. E. Miller, *Handbook of Pschology: Educational Psychology*, I. B. Weiner, Ed. John Wiley & Sons, Inc., 2003, vol. 7. 1

[4] Desire2Learn, "Desire2learn insights learning guide," Desire2Learn.com, Tech. Rep., 2012. 1, 2.4

[5] L. RAMIST, C. LEWIS, and L. McCAMLEY-JENKINS, "Student group differences in predicting college grades: Sex, language, and ethnic groups," *College Entrance Examination Board,*, vol. 2, pp. 1–45, 1994. 1

[6] Capterra. Top learning management system software products. [Online]. Available: http://www.capterra.com/learning-management-system-software/infographic 2

[7] M. Media, "Open-source learning management systems: Sakai and moodle," Monarch Media, Inc, Tech. Rep., 2013. 2.1

[8] Moodle. (2012) Moodle gradebook. [Online]. Available: http://docs.moodle.org/24/en/Gradebook 2.1, 2.1

[9] R. Hijon-Neira and A. Velazquez-Iturbride, "Improving the analysis of studnets' participation & collaboration in moodle forums," *InTech*, vol. 6, pp. 12–60, 2009. 2.1, 2.1

[10] M. Hlbl and T. Welzer, "Students feedback and communication habits using moodle," *ELECTRONICS AND ELECTRICAL ENGINEERING*, vol. 6, no. 102, pp. 63–66, 2010. 2.1

[11] P. Cauley, "A guide to explain it all," IT Babble.com, Tech. Rep., 2010. 2.2

[12] L. Y. Muilenburg and C. Holland, "Supporting student collaboration: Edmodo in the classroom," Master's thesis, St. Marys College of Maryland, 2010. 2.2

[13] edmodo.com, "Edmodo teacher guide," Edmondo, Tech. Rep., 2008. [Online]. Available: http://susd.edmodo.com 2.2, 2.2

[14] A. Pop, "Edmodo e-portfolios in efl a case study," *Dimitrie Cantemir University of Tirgu Mures*, 2010. 2.2

[15] P. Bradford, M. Porciello, N. Balkon, and D. Backus, "The blackboard learning system," *The Journal of Educational Technology Systems*, vol. 35, pp. 301–314, 2007. 2.3

[16] A. Tella, "Reliability and factor analysis of a blackboard course management system success: A scale development and validation in an educational context," *Journal of Information Technology Education*, vol. 10, pp. 25–81, 2011. 2.3, 2.3

[17] *Learning Management System (LMS) Review Summary of Findings*. University of Pennsylvania, 2013. 2.3

[18] K. A. Livingstone, "Learning and teaching effectiveness in the digital age: A case study from a pacific tertiary education provider," *Journal of Business Management & Social Sciences Research*, vol. 3, pp. 1–16, 2014. 2.4

[19] Aimetis. (2005) Desire2learn client sucecess story. [Online]. Available: http://www.brightspace.com/resources/Desire2Learn_Success_Story 2.4

[20] Desire2Learn. (2013) Does desire2learn learning environment integrate with. Short Paper. 2.4

[21] ——. (2003) Desire2learn prediction. [Online]. Available: https://documentation.desire2learn.com/en/understanding-class-dashboard 2.4

[22] M. Levesque. (2004, April) Fundamental issues with opensource development. [Online]. Available: http://firstmonday.org/ojs/index.php/fm/article/view/1137/1057\unhbox\voidb@x\bgroup\@xxiil\egroup3 3.1

[23] M. Community. (2015, April) Moodle coding style. Electronic. Moodle. [Online]. Available: https://docs.moodle.org/dev/Coding_style 3.1

[24] ——. (2015, November) Moodle 3.0 announcement. [Online]. Available: https://moodle.org/mod/forum/discuss.php=323243 3.1

[25] ——. (2014, September) Moodle block coding guide. https://docs.moodle.org/dev/Blocks. [Online]. Available: https://docs.moodle.org/dev/Blocks 3.1

[26] ——. (2015, September) Coding. Electronic. https://docs.moodle.org/dev/Coding. [Online]. Available: https://docs.moodle.org/dev/Coding 3.1.4

[27] *PHPUnit Manual*. 4

[28] G. Sivathanu, C. P. Wright, and E. Zadok, "Ensuring data integrity in storage: Techniques and applications," *StorageSS*, 2005. 4.3

[29] C. Noneman and L. McMillan, "Simple data storage and manipulation for scientists," 2011. 4.3