ECE 458: Engineering Software For Maintainability

Senior Design Course

Spring 2015

Evolution 1 Analysis

Brian Bolze, Jeff Day, Henrique Rusca, Wes Koorbusch

Contents

1 Introduction **3**

2 Project Plan **3**

2.1 Design Goals 3

2.2 Language Choice 3

2.3 Timeline 3

2.4 High-Level API 3

3 Design Review **4**

3.1 Status 4

3.2 Design 4

3.3 Alternate Designs 4

4 Design Process Notes **4**

4.1 Designed and Conducted Experiment 4

4.2 Analyzed and Interpreted Data 5

4.3 Designed System Component to Meet Desired Needs 5

4.4 Deal with Realistic Constraints 5

4.5 Contributed to Team Work and Interacted with Team Members 5

5 Next Steps **6**

1 Introduction

Good software design is often seen not only as a science but as an art. It is a craft and, like any other type of engineering, is only mastered over time. The fundamentals of good software design, however, remain sound. In this senior design course, we plan on synthesizing our four years of knowledge through the development of a robust and long-lasting software application implementing a web-based calendar. We plan on applying the core principles of good design to our code and to our design process, while continuously evaluating, refining, and improving on our skills of the software engineering trade.

For our project, we built an application using the Ruby on Rails framework. This framework aided us in developing highly modular and reusable code due to the MVC architecture and Rails' powerful web application stack. With the additional help of great documentation and a strong community, we were able to develop a functional product within weeks, despite minimal domain expertise. While our current application has noticeable areas for improvement, our robust model, well thought out API, and extensive front-end templates promise a maintainable foundation for future development.

2 Project Plan

2.1 Design Goals

2.2 Language Choice

stuf stuf stuf

2.3 Timeline

stuf stuf stuf

2.4 High-Level API

stuf stuf stuf

3 Design Review

3.1 Status

stuf stuf stuf

3.2 Design

As a result of utilizing the Ruby on Rails framework, we separated our program into four distinct large scale sections: the Model, View, Controller, and Database. A browser is used to

3.3 Alternate Designs

stuf stuf stuf

4 Design Process Notes

4.1 Designed and Conducted Experiment

Jeff's Contribution

Once the team finalized our schema for the database, we began developing individual aspects of the entire design on each of our own git branches. However, we noticed that when testing newly added features on our own individual branches, the database would for some reason roll back to an older version of the database that would cause the website to break. For a very long time, we thought it was because the database schema and model files were being transferred over to new branches incorrectly. However, after trying to implement new features on both individual branches and on the master branch, we realized that in order to update the database layout using rake db:migrate (the rails command to automatically update the database) on an individual branch, both the migrations and the database files needed to be committed to master whereas we were only committing the schema files. After running the experiment of developing on two different branches and seeing the different outcomes (along with some help from stackoverflow), we were able to solve this bug using an experiment

4.2 Analyzed and Interpreted Data

Jeff's Contribution

Included in the rails framework is the ability to write tests for all of the different models that we create. In order to confirm the functionality of our models and the associated database, we developed a number of test files that would create instances of our models, manipulate them in some way (add, edit, or delete specific fields of the model), and then display the result. In confirmation of our model design and function, we developed different data schemes to test and analyzed their outcomes to make sure that our implementation was sound.

4.3 Designed System Component to Meet Desired Needs

Jeff's Contribution

Our team as a whole spent many hours designing our database schema and model to make sure that our design had the ability to meet all the desired needs specified by the evolution document. Although not every single requirement was implemented perfectly by the specified deadline, we our confident that our design deals with every possible relationship necessary to facilitate the completion of the requirements.

4.4 Deal with Realistic Constraints

Jeff's Contribution

The fact that we made a decision part way through the development process to switch from a Django framework to a Rails framework made the time of total development much smaller. This restraint required us to quickly design our new model in Rails while still having to consider good design principles in what we decided to implement.

4.5 Contributed to Team Work and Interacted with Team Members

Jeff's Contribution

I attended every single team meeting and highly contributed to the initial model design. Specifically, I helped white board out the model and collaborated with Henrique on integrating aspects

of how events are displayed and how the overall look and feel of the application should be.

5 Next Steps

stuf stuf stuf