Air Traffic Control: A Modern Software

Josh Scotton

Nick Cannon

Abdallah Al-Sulaihat

Yahya Osman-Husein

Version 2

Revision History

* Document Created: February 26, 2015 - Josh Scotton
* First Revision: April 8, 2015 - Josh Scotton, updated all content

Purpose

To create a software that satisfies the FAA's request of a modernized and current means to train their air traffic controllers. To have a centralized and singular means of managing the air traffic controllers training. Also, to test the team's capabilities as software engineers.

Table of Contents

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

Part 1:

1.1: Project Overview

1.2: Project Deliverables

1.3: Evolutions for the SPMP

1.4: Reference Materials

1.5: Definitions and Acronyms

Part 2:

2.1: Process Model

2.2: Organizational Structure

2.3: Organizational Interfaces

2.4: Project Responsibilities

Part 3:

3.1: Management Objectives and Priorities

3.2: Risk Management

Part 4:

4.1: Software Documentation

4.2: Project Support Functions

Part 5:

5.1: Work Breakdown Structure (WBS) and Dependencies

Part 1

Project Overview

The project was brought to us by the FAA through a Request for Proposal (RFP). In this request they have expressed the desire for a modern, current and centralized software, known as a "learning content management system" or LCMS for short, be created for the singular use of training their air traffic controllers. This software is to be the main source of information relating to any aspect of the training of air traffic controllers. In this software there is the lessons, simulations, quizzes, and various reports.

There are four different permission levels: trainee, training administrator, manager and auditor. Within these different access levels there are a variety of things that can be done ranging from completing lessons, creating lessons to viewing reports on different trainees and even adding, removing and editing different employees.

The product should be able to act as a database for several different types of reports, employee information, simulations and lessons. It should also be versatile enough to generate simulations and quizzes that appropriately reflect the lessons outlined by the training administrator. As well as generate GUIs that enable the user to perform several different activities described earlier.

As for ourselves, the goal of this project is to determine our capabilities in creating a proto-type software within an allotted amount of time. It is also to test our abilities in project management and creation, understanding of various software engineering topics, programming abilities, and our abilities to work in teams toward a common goal.

Project Deliverables

* Requirements - February 19, 2015 - Blackboard / In Class
* Use Case - February 19, 2015 - Blackboard / In Class
* High-Level Architecture - February 26, 2015 - Blackboard / In Class
* Object Classes - February 26, 2015 - Blackboard / In Class
* Sequence Diagram - February 26, 2015 - Blackboard / In Class
* Software Project Management Plan - March 5, 2015 - Blackboard
* Group Status Presentation - March 26, 2015 - In Class
* Software Deployment - April 9, 2015 - In Class
* Group Product Presentation - May 14, 2015 - In Class

Evolutions of the SPMP

The plans for anticipated and unanticipated changes are to follow protocol; in which we as a team come up with the most efficient and productive alteration in order to respond to said change and present it to our Program Manager, Rebecca Broadwater. From there, the team's idea will be evaluated and considered. If the change is desirable, then it is approved and then the team begins working on adjusting all necessary paperwork, plans and models in order to adapt to the changes.

After this has been done, the team begins to implement the changes into the software itself. Any minor changes to previously created code must be documented and integrated into the repository and the team must be informed of said changes, major changes must be discussed with the team first then follow the aforementioned protocol.

Since this project is delivered in increments, it should be very adaptive and capable of handling several unanticipated changes. Every change and alteration will be documented in here, in the code itself and every other document that pertains to the desired change.

Definitions and Acronyms

* RFP - Request for Proposal
* FAA - Federal Aviation Administration
* SPMP - Software Project Management Plan
* LCMS - Learning Content Management System
* GUI - Graphical User Interface
* WBS - Work Breakdown Structure
* IEEE - Institute of Electrical and Electronics Engineers

Reference Materials

* The FAA's RFP (Request for Proposal)
* Project requirements outline - Created by Rebecca Broadwater

Part 2

Process Model

The LCMS is responsible for the main interactions between user and the software, every user request goes through the LCMS which then contacts the area containing the appropriate data. Upon opening the LCMS, there should be the appropriate GUI brought up that enables interaction between the software and the user. From there the user enters a login and the LCMS pulls the corresponding information from the database in order to verify their username and password. From that information, the LCMS draws the appropriate GUI based on the employee type. The LCMS also contacts the database for everything data-related.

In terms of simulation, the LCMS requests the lesson number information and corresponding simulation type then goes to the simulator and draws the correct simulation pertaining to the lesson information gathered from the database and proceeds to evaluate the user. Any data gathered or altered from the user is then stored by the LCMS into the database for later use.

Organizational Structure

The client, FAA, corresponds with our Program Manager, Rebecca Broadwater. She, in turn, corresponds with the development team (us). Any questions that the development team is given to the program manager who in turn asks the question to the client. Josh Scotton is responsible for the administrative aspects, Yahya Osman-Husein is responsible for the database, Nick Cannon is responsible for the repository and joint software developer and Abdallah Al-Sulaihat is responsible for GUI design and also a joint software developer.

Client (FAA)

Project Manager

Rebecca Broadwater

Team

Repository

Nick Cannon

GUI

Abdullah Al-Sulaihat

Database

Yahya Osman-Husein

Administrative

Josh Scotton

Organizational Interfaces

As of right now, everything is planned to be independent of other entities and should be compatible with both Microsoft and Apple platforms. As a team we decided it best to subcontract the simulator for the air traffic control lessons, since no one on the team has any experience with such content it was thought to be best to outsource that material elsewhere to someone who has the appropriate knowledge to adequately create a simulator.

Later on however, it is planned to make this LCMS have the potential to integrate with any sort of simulator in case the FAA or other companies have a different preference for the simulator that they use, supposing that training differs on the training site or things of that nature.

As of yet there are no plans for the software to be available for any sort of mobile device compatibility or plans for this to become a fully commercialized software available to the general public.

Project Responsibilities

The administrative and paperwork aspect of this software is Josh's responsibility, he is responsible for updating and making changes where necessary in the documents correlating to changes made by the client or the programming team. He is also responsible for assisting where necessary with coding and researching information vital to the completion of the software.

Yahya's main responsibility is to keep the database current, linked to the software and connected in the right methods with the right access to each of the different respective employee types. On top of that, he is also responsible for assisting the main programmers with any secondary objectives or relieving their workload.

Abdullah is creating the GUIs that are going to be generated by the LCMS for the different pages brought up in the software. He is also responsible for assisting in the method development of the main code and making sure that the prototype will be deployable within the time constraints.

Nick has the responsibility of maintaining the repository, managing it and keeping it as current as possible. This means integrating all of the methods and classes. He assists in generating the code for all of the methods and making sure that everything is in its proper order and functional.

Any and all major changes must be agreed upon by the team and brought to Program Manager Broadwater to ensure that the client accepts the change and that it flows well with the entire goal of the program. Minor changes are at the discretion of one or two team members but must be clearly documented.

Every team member will be knowledgeable and current with the code and the documents corresponding to the code. This is to ensure that there is the maximum flow of ideas and productivity.

Part 3

Management Objectives and Priorities

The management philosophy is a very free reign sort of philosophy. The Program Manager gave the team an outline of the client requirements. From there it was up to the team to decide how we wanted to manage ourselves. As a team we thought it were best to follow a sort of Spiral philosophy with ideally a gradual developing, testing, changing sort of routine. There is much emphasis on the incremental delivery in order to see if our work is up to par with the customer's requests and to ensure that we would make our deadline.

Another aspect of the management philosophy was the emphasis on documenting every change and revision. Minor changes have to be documented and major changes have to be discussed as a team and then brought to the program manager for discussion and approval. From there the change is implemented, documented in the paperwork and in the code itself and then vigorously tested to ensure its viability.

The main priority of this management system is the incremental delivery and to make sure that every aspect of the prototype works cohesively within the allotted time. Beautifying and streamlining the software is a secondary priority that would come after successful operation of the program prototype. There was no defined budget for this software because the team is working as unpaid employees.

Risk Management

The most glaring risk of the project so far is within the team, determining whether or not we have the appropriate knowledge in order to complete this project within the predefined schedule. Taking on a task this size in a limited amount of time is easily managed through thorough research and adaptability through adjusting code to reflect our current capabilities. Currently, it seems as if time is our biggest enemy and risk.

Another risk is the "client acceptance" risk, where we as a team develop a prototype that does not match the client's desires. If we fail to satisfy our program manager and the FAA, the results could be disastrous. To mitigate this risk as best as possible, the team is determined to use incremental deliveries. This would catch any problems that a client has early on and help the team reduce the amount of major changes and overhauls they would need to do.

Part 4

Software Documentation

Every time a document is updated, there should be a revisions page early on in the document that needs to be altered naming the current version, who updated the document and a general summary of what they have updated. This is in order to keep track of all the changes as the software develops and the requirements progressively change in order to adapt to the capabilities of the software engineers and the desires of the client.

Each time any portion of code is updated, revised or changed documentation of the version type, who made the changes and what changes they made must be noted in the repository. This is in order to keep track of all changes made to code and so teammates are aware of who is working on different portions of the software. It also enables t he team to determine where the errors have occurred, if they were working in the previous version or model.

As long as all the data and information is consistently updated through proper documentation conventions, it will help streamline the whole engineering process and reduce the amount of complications and errors spanning all of the documents and differing portions of code.

Project Support Functions

In order to ensure the most quality product possible, the prototype shall undergo vigorous testing that shall challenge every aspect of the software. This will reduce the amount of potential bugs upon the final release of the product. The IT members of the FAA training facilities shall also be thoroughly educated on the system and be given advanced privileges in order to ensure that the software remains consistently smooth and any post-release bugs will be caught and easily fixed.

Any unsolvable problems shall be forwarded to the software engineering team in order to get the most familiar people with the software to fix the problem(s). Our software shall adhere by the management and configuration plans outlined by the IEEE. (Institute of Electrical and Electronics Engineers)

Part 5

Work Breakdown Structure and Dependencies

* Outline Use Cases and Requirements
* Design High-level architecture, Object Classes and Sequence Diagram
* Generate Software Program Management Plan
* Begin implementation of plan
* Test piece by piece aspects of code
* Review software performance
* Make any necessary changes
* Integrate simulator and database
* Put all the pieces together and deploy
* Test as a whole
* Make any additional / last minute changes
* Test again.
* Demonstrate prototype for client approval.
* Begin developing final product and put finishing touches.

In order to progress anywhere, the entirety of the software is dependent on outlining the clearest, most concrete requirements and use cases from the client. Without any sort of direction the product could quickly become a massive project and unfeasible. From there, careful planning and outlining of how to turn those requirements into reality. This is where the development of architecture, classes and a sequence diagram come in. These things are vital to any sort of efficient software engineering. Without sufficient planning and coordination, the team would fall to "cowboy" coding and the whole software could quickly turn into a mess.

After that, thoroughly detailing every aspect of progression and the whole plan for every step of the way is necessary. This is done in case of confusion or a problem, every situation is documented that would detail what to do in order to solve a problem or give clarity to a person who is confused. Once all of this paperwork and documentation is done, finally the coding can begin. Tackling every aspect of the software in manageable pieces is critical. Then, testing each piece individually to ensure that each piece is functioning prior to full integration and deployment allows for problems to be caught in these pieces preventing massive errors throughout the entire program upon integration.

Next, integrating all working pieces allows for the beginnings of a prototype that can be demonstrated. This part depends on the testing and fixing of the individual pieces in order to make sure that there is as few errors as possible that do not spread to other parts of code and cause mass confusion. Once the prototype works as desired, the potential product is then demonstrated to the client for approval. One cannot present an unfinished or error-ridden product to the client and expect much. If the approval is given, then finally the development of the final product begins and the cycle begins anew.