Project Plan

Co-op Evaluation System

Senior Project 2014-2015

Team Members:

Tyler Geery
Maddison Hickson
Casey Klimkowsky
Emma Nelson

Faculty Coach:

Samuel Malachowsky

Project Sponsors:

Jim Bondi (OCECS)
Kim Sowers (ITS)

Table of Contents

Revision History

Version	Primary Author(s)	Description of Version	Date Completed
v1.0	Emma Nelson, Maddison Hickson, Casey Klimkowsky, Tyler Geery	Initial revision	September 14, 2014
v1.1	Emma Nelson	Update after the Requirements phase, inclusion of the metrics description (3.5)	October 12, 2014
v1.2	Emma Nelson	Update after Architecture phase	November 5, 2014
v1.3	Emma Nelson	Update at the end of Fall Semester	December 10, 2014
v1.4	Emma Nelson	Update to roles, metrics, and technologies	January 8, 2015
v1.5	Emma Nelson	Update to deliverables, assumptions, project lifecycle, project organization, schedule, resource allocation, tracking and control, metrics, technology, and supporting plans	February 5, 2015
v1.6	Emma Nelson	Final updates at the end of the project	May 14, 2015

1 Introduction

1.1 Overview

The purpose of this project is to re-engineer the Co-op Evaluation System in order to leverage newer web technologies while also improving performance and user interaction. The current system uses outdated, under-documented technology, which makes it difficult to maintain. Furthermore, the random errors that occur do not give users confidence that their information was submitted properly. Significant improvements to the user interface will need to be made, but the existing database structures can be used as a reference for modifications.

One of our primary goals is that by the conclusion of this project, we will, at a minimum, have supplied OCECS and ITS with a product that is functionally equivalent to the existing system. Time permitting, we hope to implement several new features, as defined by our project sponsors, as well. We plan to design and implement the system with extensibility in mind, so that in the future, other developers may implement additional features that we were unable to implement during the duration of this project.

1.2 Deliverables

In addition to this document, the following deliverables will be completed by the conclusion of this project. Some of these documents are required by the project sponsors, while others are required for the Software Engineering department.

Deliverable	Description
Project Website	Holds all non-proprietary work products and projects artifacts
Activity Tracking Spreadsheet	Documents the time worked by each team member on any given high-level task as well as the aggregation of time spent by the team as a whole
Software Requirements Specification	Describes the system to be developed and outlines all functional and non-functional requirements. This deliverable also includes our user workflows that played into developing the final set of requirements in a user-centric way
Architecture Document	Outlines the high-level architecture of the project
Design Document	Details the software design at a subsystem and services level. This deliverable also includes mockups of the system in the form of static images and clickables
Test Plan	Details the testing strategy used by the development team including, but not limited to, unit testing, usability testing, and

	beta testing approaches. This deliverable also includes the correlating test scripts and materials
Project Presentations	Two presentations will be delivered: the first describes our interim status in December 2014, and the second concludes the project in May 2015
Project Poster	Presents a description of the project, the technologies used, and other relevant project information in the graphical and textual form of a poster
Technical Report	Outlines the basic requirements of the system, development process, state of the system at time of delivery, and other similar high-level descriptions
Hand-off Support Documentation	Provides ITS with the information they need to continue supporting the final delivered product; No formal documentation, information transferred via code review and artifact hand-off
User Manual	Describes the functionality of the system, and describes to users how to use the system; Part of the product
Software Product	All software written for this project, including tests and build scripts

1.3 Assumptions and Constraints

The development team will work with ITS Application Development and Operations staff as needed to deliver a solution that meets the technology standards of ITS and can be supported by ITS staff once the application has been deployed to production.

Our biggest constraint is the set of application technologies that are supported by ITS. At a minimum, we are limited to using Java to implement our backend, and it is recommended to us by ITS to use AngularJS for our frontend. Furthermore, if we choose to implement our own database instead of using the pre-existing structures, we will be limited to using Oracle SQL Developer. We understand that we are allowed to pursue technologies that are not on the list of application technologies supported by ITS, but will have to present these technologies and have them approved by ITS before we are able to use them for this project.

1.4 Definitions and Acronyms

EWA	Enterprise Web Applications, a division of ITS
IDE	Integrated development environment
ITS	Information and Technology Services
OCECS	Office of Cooperative Education and Career Services
RIT	Rochester Institute of Technology

1.5 Reference Material

We were given the following materials prior to starting the project:

Reference	Description
ITS wiki page	Created by the ITS team and contains sample applications as well as guidelines by which we must abide
CD from Jim Bondi	Content on the CD includes documentation and code created by previous Senior Project teams
Project description	Description of the project given to us by the Software Engineering department, and written by Kim Sowers

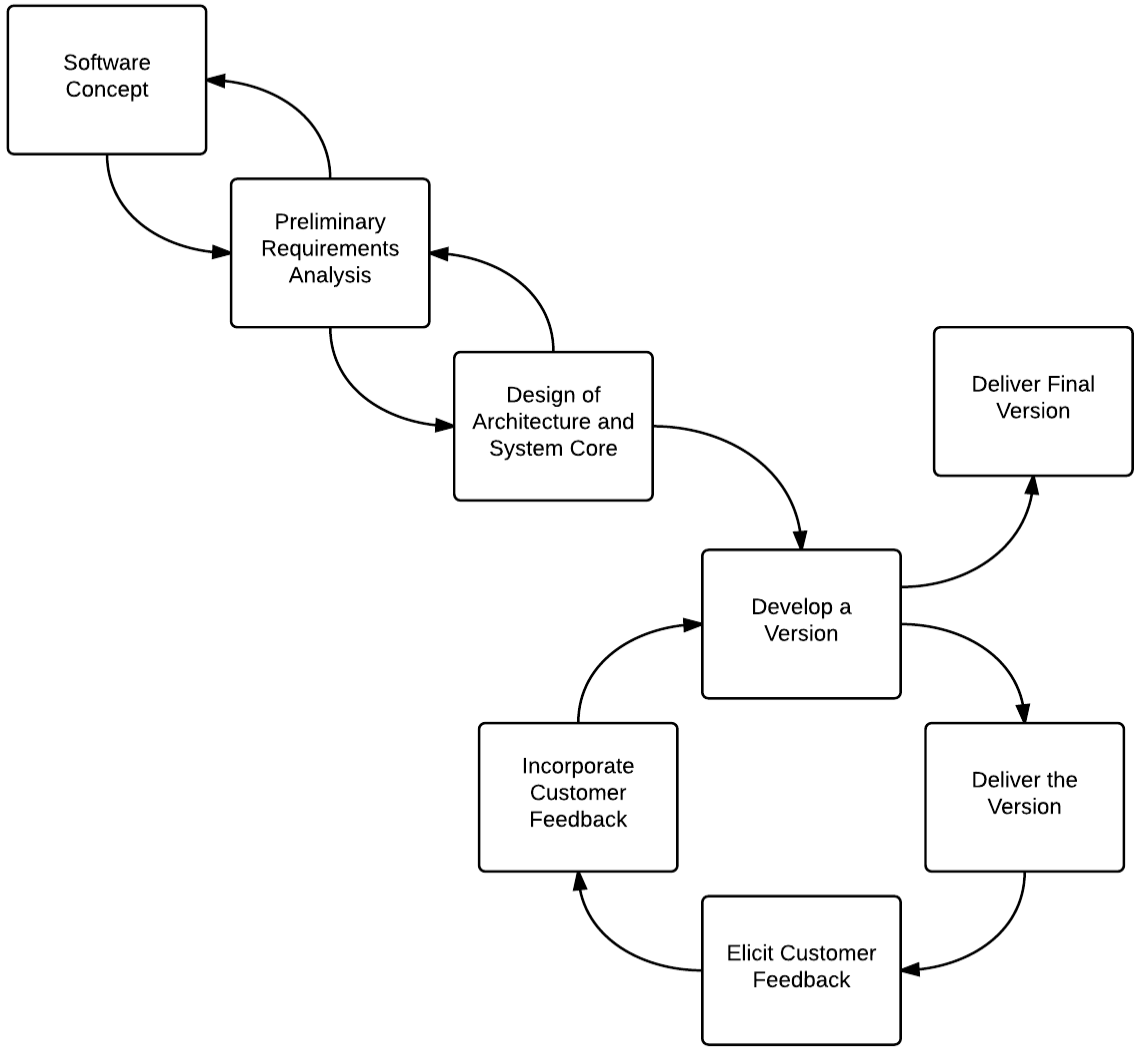
2 Management Structure

2.1 Project Lifecycle

For this project, we will be following the evolutionary delivery lifecycle. Evolutionary delivery is a lifecycle model that straddles the ground between evolutionary prototyping and staged delivery. In this sense, we will develop a version of our product, show it to the project sponsors, and refine the product based on sponsor feedback. Although we would like to accommodate as many customer requests as possible, our accommodation of these requests will based on project schedule and technical feasibility. If we manage to accommodate most requests, our project lifecycle will look a lot like evolutionary prototyping. If we only manage to accommodate a few change requests, our project lifecycle will more like staged delivery.

In evolutionary delivery, our initial emphasis is on the core of the system, which consists of lower level system functions that are unlikely to be changed by customer feedback.

One significant customization we will be making to the evolutionary delivery lifecycle is our risk management. At the beginning of each “Develop a Version” stage, we will place a heavy emphasis on risk management. Refer to 2.3 for a more detailed description of how we intend to manage risk for the duration of this project.



In our process model, “Develop a Version” will include various sub-steps as detailed in the list below:

1. Identify goals, tasks, and features for the cycle
2. Analyze the risks involved and re-evaluate Top 10 Risks table
3. Estimate the time for each goal, task, or feature
4. Update the schedule and project plan to reflect the plan for the cycle
5. Develop the necessary artifacts, functionality, etc.
6. Test any code and user interaction created

For “Deliver the Version”, we will deliver the functionality for the milestones set and demo them during our sponsor meeting that week. The completed functionality will also be deployed to our DEV server by that meeting and to TEST shortly afterwards to be available for acceptance testing.

As a part of “Incorporate Customer Feedback”, the development team will complete an informal post-mortem for the cycle in order to analyze what went well, what went poorly, and where we can improve. This is an important point in that it will allow us to better plan the next cycle and reduce the risk of an error reoccuring.

2.2 Project Organization

2.2.1 Roles and Responsibilities

In addition to the team roles we have assigned below, we have agreed that there will be a certain amount of overlap between the roles. We made this decision as a part of our risk mitigation plan to ensure that all of our tasks are completed on-time.

Name	Role	Responsibility
Emma Nelson	Team Coordinator	Coordinates overall team process and is the person responsible for all project documentation
Maddison Hickson	Webmaster Development Coordinator	Maintains the project website Tracks development progress and is point person for keeping feature development on schedule
Casey Klimkowsky	Communications Coordinator	Serves as a communication point between the development team and project sponsors Records meeting minutes during all meetings, including any major team decisions and action items
Tyler Geery	Testing Coordinator	Track testing progress and ensures any gaps are addressed
Anu Sharma	Contractor	Development and testing support

2.2.2 Relationships to External Organizations

We will work closely with the ITS Application Development and Operations staff to determine approved technologies for the project. Furthermore, at the conclusion of the project, we will perform a formal handoff of the finished product to the ITS Support staff.

OCECS is the other organization that we will be working with for the duration of the project. The requirements for the final system will be given to us by OCECS, as the goal of this project is to replace their existing Co-op Evaluation System.

2.3 Risk and Asset Management

We are maintaining a list of all risks we believe could be a problem throughout the lifecycle of the project. This document serves as a living list of possible risks for the remainder of the project. We actively manage a separate risk table of the top 10 risks that are of the biggest concern to us at the current stage of the project. These top risks will be revisited every week.

Refer to Risk Table for a formal collection of risks that will be actively managed for the duration of this project, and the first indicator and mitigation approach for each.

3 Planning and Control

3.1 Estimate

Estimation will be continually refined at the start of each stage in the process. Before starting the development work of each stage the team will take time to establish what they want to accomplish in the given stage and estimate the time each task will take, thus the exact schedule will not be fully complete until near the end of the project. At a higher level, the team has developed some rough estimates for major milestones as listed below.

Milestone	Estimated Date of Completion	Refinement process
Requirements	3 weeks after reviewing the current documents October 6, 2014	The team will analyze what work has been done and the quality of the work, then re-estimate based on their conclusions.
Architectural design	3 weeks after completing the requirements October 30, 2014	The team will analyze what work has to be done, then re-estimate based on their conclusions.
Detailed design	3.5 weeks after completing the architectural design November 25, 2014, with more wireframes coming as they are completed	The team will analyze what work has to be done, then re-estimate based on their conclusions. At that time, the team will determine the best way to accomplish the work and set a reasonable deadline.
Implementation Cycle 1: Environment Configuration and Student Dashboard	Last 2 weeks of Fall Semester plus Intersession (half time) January 25, 2015	It's a hard deadline, so no refinement is required. Re-estimation will happen if the deadline is slipped.

Implementation Cycle 2: Student Role	3 weeks after C1 ends. February 17, 2015	It's a hard deadline, so no refinement is required. Re-estimation will happen if the deadline is slipped.
Implementation Cycle 3: Employer Role	3 weeks after C2 ends. March 10, 2015	It's a hard deadline, so no refinement is required. Re-estimation will happen if the deadline is slipped.
Implementation Cycle 4: Admin Role Part 1 and Evaluator Role	3 weeks after C3 ends, not including Spring Break Week. April 7, 2015	It's a hard deadline, so no refinement is required. Re-estimation will happen if the deadline is slipped.
Implementation Cycle 5: Admin Role Part 2	3 weeks after C4 ends. April 28, 2015	It's a hard deadline, so no refinement is required. Re-estimation will happen if the deadline is slipped.
Code Freeze (No new features, only bug fixes)	April 28, 2015	It's a hard deadline, so no refinement is required. Re-estimation will happen if the deadline is slipped.
Implementation Cycle 6: Finalization and Documentation	3 weeks after C5 ends. May 21, 2015	It's a hard deadline, so no refinement is required. The team will re-estimate once ITS has provided their expectations for the follow-up documentation.

3.2 Resource Identification

3.2.1 Time

The available calendar time for the project is the length of two academic semesters, a total of 33 weeks, non-inclusive of institute breaks. A limited amount of time will be spent by the development team during the intersession break in order to avoid "summer syndrome", or forget where we were at before the fall semester ended.

3.2.2 Cost

If we find technological tools that require the purchase of a license, and these tools fit our needs better than alternative non-paid options, we may approach the project sponsor to request funding for these technological tools. However, at this time we do not know what the exact monetary cost may be, or if there will be any at all.

3.2.3 Materials

ITS has provided us with a page of EWA Student Development Guidelines on the RIT Wiki. This wiki provides us with expected standards for technology, documentation, and user interface design. The wiki also contains sample applications, which will assist us in the construction of the skeleton for our application.

Jim Bondi has provided us with a CD, which includes documentation and code created by previous Senior Project teams. We will use this content as a basis for the planning of our project documentation.

3.3 Resource Allocation

3.3.1 Schedule

Refer to the Schedule section on the team's website for a detailed schedule for the project, which will be updated as needed.

3.3.2 Team Members

A contractor has been add to this project at twenty hours a week to help with the development of the actual software. She will be in charge of developing features and corresponding tests and is responsible for following the process as set up by the original team and meeting all standards for code style, commit messages, etc. like any other member of the team.

3.4 Tracking and Control

Our schedule for the project will be tracked and controlled throughout the project using a table on our website, as mentioned above in 3.3.1.

We will be using Trello to track high-level process tasks and in-depth development tasks throughout the different phases of the project. In addition, we will be using a Google Drive spreadsheet to track the current state of high-level tasks. The spreadsheet provides information such as who is assigned to each task, the estimated amount of time it will take to complete each task, the actual time it took to complete each task, and any additional comments.

In addition to Trello, we will be using Slack as a communication tool. We will receive notifications on Slack from several sources, including Trello, Google Drive, and GitHub. This provides us with a one-stop-shop for updates to project resources as well as a location for contextualized conversations.

Finally, we are using GitHub Issues to track development tasks. These tasks will be included in Trello at the feature level, so we can easily see which part of the system are being developed at any given time; however, all detail on who exactly is doing what within a given feature set and where they are will be found on GitHub. This method of tracking provides ITS with the full history of our development process in a format that won't disappear with us when we graduate.

The expectations of functionality and quality, and other non-functional requirements, of the software product will be defined in the software requirements document. We will moderate our adherence to these standards predominantly through the use of Trello, as we intend to go more in-depth with Trello and GitHub Issues than our Google Drive spreadsheet.

3.5 Metrics Tracking

The development team is tracking three metrics to aid in measuring the quality of the project: requirements volatility (churn), usability, and backlog management index.

Tracking requirements volatility will give the development team insight into how much the requirements change over time, starting from the time of completion of the initial version of the requirement documents. This is particularly important when it comes to recognizing when, how often, and how many requirements were added or dramatically changed.

Usability will be measured through a variety of ways during usability testing after the completion of the initial clickable mockups. Measures may include completion rate, task time, error count, and/or number of clicks. After each usability test, the subject will complete a form of Likert questions. Their responses will be tallied up with those of other testers to create a SUS score for the product and possibly for each individual user role of the product. This score will act as the core metric on usability.

The backlog management index is the number of bugs introduced (reported) compared to the number of bugs resolved. This metric will be actively tracked starting at the beginning of the development phase.

Finally, multiple metrics on effort of the team are being tracked in the Activity Tracker on the time tracking tabs for each semester. These metrics include total hours per person, average weekly hours per person, total hours by the team, and average weekly hours by the team. The percent completion of estimated time, estimated time, and actual time is also available on the task tracking tabs for each semester.

4 Technical Process

4.1 Technology

4.1.1 Environment

Java will be used as our main programming language with JavaScript on the client side. For Java development, we will be using IntelliJ as our IDE. For client-side development, we will use IntelliJ or Sublime Text.

4.1.2 Methods, Tools, and Techniques

We will be using Maven to help automate our software build. The source code for our application must be written in Java, and will follow an Web Services paradigm through the use of Spring Boot. AngularJS will be used on the client side, and will be used to help implement

the MVC pattern in our application. We will also be utilizing Sass as a substitute for straight CSS in order to simplify the development of our stylesheets. We will be using Twitter Bootstrap as well to help us to implement a responsive user interface.

4.2 Infrastructure

ITS will provide us with a GitHub repository. It will hold all of our source code for and will serve as our version controlled repository.

ITS will also provide us with a Tomcat development server, which will serve as our runtime environment. Our application build will run in our own development environment.

We must build our application with the assumption that it will be deployed in a DEV, TEST, and PROD instance. Any values that are environment specific must be stored outside of the application code.

For a relational database, ITS will provide us with access to existing Oracle databases. We may or may not reuse the existing database structures. If we choose to implement our own data structure, we will be using Oracle SQL developer.

4.3 Project Artifacts

The project artifacts that we plan to deliver are listed in 1.2. All of these documents will be available on the project team's website once they are completed.

5 Supporting Plans

This section will be completed over the course of the project, if and when, we define and write the respective documents.

5.1 Configuration Management

Information concerning Configuration Management can be found on the GitHub wiki for the Co-op Evaluation project.

5.2 Testing

This document can be found on the team website along with all the related materials for specific types of testing. All automated tests are included in our GitHub repository.

5.3 Deployment

There is no formal deployment documentation as our deployment process follows the process defined by ITS on our wiki page and during work sessions.

5.4 Maintenance

Notes regarding maintenance and current state can be found in our Trello board, GitHub Issues, and Technical Report. Any further documentation concerning maintenance and future development will be completed by another senior project team or ITS, whomever is set to complete the project.