On-Line Student Co-Op Evaluation System

Project Plan Document

Team Char c = '5'

*Patrick Flanagan, Tom Small, Whitney Sorenson,
Chris Woodbury, Dan Volpe*

Overview

The Software Project Plan is the basis for software projects. This document will describe a number of topics that are critical to the success of the Student Co-op Evaluation software. This project plan describes the process by which the software will be created, the artifacts that are necessary for successful completion, the available resources for implementing the project, and the scheduling options based on the desired functionality and dates of completion.

High-Level Functionality

Ability for a Student to Log In

A student should be able to log into the system. Currently, students will be able to do this with a DCE account. This summer, the co-op office is planning to change the way students are tracked and move away from the universal student ID for everything.

Fill Out the Co-Op Evaluation Form

Once logged in, the student will be able to fill out an electronic version of the co-op evaluation form. Submissions will go to a central repository that can be accessed by both OCECS and the student’s department. Students also have the ability to save a partially completed evaluation to be finished at a later time.

View Submitted Evaluations

Students will be able view work reports that they have submitted, as well as evaluations that their employers have submitted.

Generate Reports

The system will need to generate reports based on feedback from students. These reports will be a useful analysis tool for the co-op office and other users of the system.

Project Staffing

The schedule for the project has been broken down into four two- to three-week iterations. During each of these iterations our five team members will be working on the developing the project, and will be supervised by Dr. Naveda, our academic advisor, and Jim Bondi from the Co-op Office, our sponsor/customer.

While everyone on the team will be involved in all of the tasks, each person will be primarily responsible for certain aspects of the project. Michael is the team leader; Chris is our point of contact to our customers and stakeholders; Tom will maintain the website and be the primary note taker; Dan will be the technical leader and tools person; and Patrick will handle planning and tracking progress.

Software Process

Process Model (Agile, Incremental)

Our project will adopt a tailored agile, incremental process model that incorporates aspects from Extreme Programming (XP). The strengths of such an agile approach make it ideal for the Student Co-op Evaluation system, since we are anticipating changes requested throughout development due to the inter-department nature of the system.

One of the main characteristics of the XP process model is the fact that the developers of the software are in constant contact with each other. If they have a question or need help, they can simply turn to the person next to them for advice. Also, it provides more chances for the developers to catch each others' mistakes. This type of environment becomes harder and harder to create as the size of the development team increases, but since our development team is small this will work perfectly.

Another important aspect of the XP process model which is very useful for a project like ours is rapid feedback. Rapid feedback (preferably directly from the customer) is very important to our system since we want a quick roll-out. The faster the feedback occurs, the faster the developers can fix potential problems and make required changes.

The last important reason for choosing an agile approach with XP trimmings deals with adaptation to change. The XP process model is centered around the idea that changes to the system are inevitable, and the developers must be ready for it. Although our project is not very large, it has a huge potential for change, mainly due to the fact that much of what needs to be done (new forms, new co-op processes, etc) are highly customizable. Changes may also need to be made with respect to ITS. Our customer has informed us that ITS has very specific requirements for how the system will be deployed, so we may need to make changes to our system to accommodate these requirements. Our custom tailored agile process model will let us tackle these changes easily, and therefore will make a good fit for our project.

Iterations

The project will consist of 4 major iterations. These iterations vary in length from two to three weeks, depending on the planned features for the iteration.

The first iteration includes setting up and configuring the development environment, and implementing the student view of the system and the email subsystem.

The second iteration includes polishing the student view of the system based on feedback from iteration 1, and adding dynamic form editing capabilities.

The third iteration will focus on reporting. This involves allowing the user to search for specific information and generating reports based on these searches, as well as exporting the returned data to an Excel spreadsheet.

The fourth iteration will involve usability testing, bug fixes, and further polishing to get ready for the final launch.

Artifacts

In general, because we have chosen to use an agile approach, emphasis will be on working code, not documentation. As a trade off to small, quick releases, early feedback, and an overall high-performance process, there is less documentation produced when compared to other process models. This is not to say that there will be no documentation, but that the documentation will developed at the same time as the software, resulting in less up-front documentation. The artifacts that we plan to produce are as follows:

- *User Interface Prototype*
  The user interface for this project is very important. To ensure that it is satisfactory, prototypes and mock-ups of the user interface will be presented early on so that the customer can give feedback based on what they want. This also helps the team ensure that all the appropriate functionality is included.
- *Risk Analysis Sheet*
  Another effective document is our risk analysis sheet. This document will prepare our team for possible disasters by outlining the risks that could have the highest impact on our project and what we estimate the likelihood of these risks to be. With this information we will be able to analyze what possible risks we face and plan accordingly in order to try and mitigate those risks as much as possible.
- *Defect Reports*
  A defect tracking sheet will be used to keep track of all bugs found in the project. The sheet may be generated from a defect tracking software package, or manually stored as an Excel spreadsheet or Word document. This sheet will display the amount of bugs, severity of bugs, as well as when the bugs were found and when they will be fixed. It will basically be a summary based on whatever defect tracking tool we decide to use. It is a good way to track project progress and it also makes for a very good way to measure software metrics.
- *Earned Value Chart*
  We will have a schedule of the different activities that need to be performed, when they will be performed, and who will perform them. This chart will provide us with a way to know where we came from, where we are, and where we're headed at all times. Using this chart, we will be able to easily tell if we are ahead, behind, or on schedule.
- *User Manual*
  Along with the system itself, we will be delivering a user manual describing how to perform all of the tasks in the system. This will be included directly in the finished system, with links to specific parts of the user manual wherever the user may want help.

Software Engineering Methods

- Test-First Development

- Object-Oriented implementation using Java

Schedule and Effort

Earned Value Chart

See attached chart.

Resources

We have identified the following resources as being vital to the successful completion of this project:

Material

- CVS Server
- Design Software (UML, etc)
- Testing Computers
- Testing Software
- Development Computers
- Eclipse IDE

Personnel

- Professor Naveda, Advisor
- Jim Bondi, Co-op Office
- Software Engineering Senior Project Team

Milestones

We have identified the following movable and immovable milestones for this project along with the expected date

• Winter Quarter Begins	(November 29, 2004)
• Project Kick-off	(November 30, 2004)
• Initial Meeting with Customer	(December 2, 2004)
• Environment Setup Complete	(December 17, 2004)
• Christmas Break	(December 18 – January 2)
• Iteration 1 Begins	(January 10, 2005)
• Iteration 1 Ends	(February 4, 2005)
• Iteration 2 Begins	(February 7, 2005)
• Iteration 2 Ends	(February 25, 2005)
• Spring Break	(February 27 – March 6)
• Spring Quarter Begins	(March 7) (Immovable)
• Iteration 3 Begins	(March 7, 2005)

- Iteration 3 Ends (March 25, 2005)
- Iteration 4 Begins (March 28, 2005)
- Iteration 4 Ends (April 15, 2005)
- Initial Deployment & Testing (April 15, 2005)
- Spring Quarter Classes End (May 13, 2005) (Immovable)
-

See the attached Gantt chart for more details.

Product Metrics

Defects

The number of bugs/defects and their severity will be identified and tracked using a defect tracking tool. Tracking defects in this manner will ensure that identified bugs will be fixed. Knowledge of what the nature of the bugs and how many are encountered will help identify areas of concern.

Earned Value

By making a list of tasks that need to be accomplished and assigning a point value to each task, we can track how much progress has been made toward the project's completion.

Test Coverage

Since we are modifying an existing system, we want to ensure that we don't break anything. We will need to track what aspects we change and insure that we fully test all of them. We will also need to create, provide, and use a detailed test document to test the overall system when it is completed.

Project Risks

Changing Requirements

Changing requirements may have a large impact on the project, especially if the changes occur late in the project. To reduce the risk, our team has decided to adopt the customized agile process mentioned above.

Security

The project's data must be secure. This risk can be mitigated through constant consideration of the possible security aspects of any features or sections of the system being implemented. Security standards, such as usernames and passwords, can easily be programmed into the system. We can also work with ITS to ensure additional security measures outside the scope of the system are considered and possibly implemented.

Testing/Debugging

Debugging and testing should be done continuously throughout the project, and any known bugs should be addressed as soon as possible. Finding bugs late in a project can cause serious problems that require much rework of the system to fix.

Tight Schedule

It is important that proper scheduling be done to ensure that a project can be finished on time. We have designed our schedule to provide ample time for final deployment issues that may occur at the end of the year, as well as providing some leeway should any slippage occur. This does lead to a tight time constraint for some of our iterations, and the team will need to be constantly aware of this.

CVS Problems

Due to the fact that we are using a remote CVS server for both version control and the storage of our project, there is the concern that the server could go out, causing problems with development. The team should have a backup strategy should this occur, and should have backups of the code created on a regular schedule should any data loss occur.

Lack of Knowledge

There is the possibility of a problem occurring due to a lack of thorough knowledge of some of the members of our team with the technology used by this project. The team needs to take this into consideration and plan to allow time for the developers to research and learn this technology.

Tools

The development of our system will require additional tools that the developers will not have, including a development box to deploy the system to and a test database to use throughout development. These tools need to be provided by OCECS and ITS, and any problems acquiring or setting up these tools could cause delays or problems with development.

Meeting Times

There are a lot of interested groups that we need to potentially meet with. OCECS is our main customer, and we need to keep them updated on all progress. ITS is also a stakeholder since they will have to maintain the system once we have finished, as well as ensure that it is properly deployed. We also need to keep our advisor, Dr. Naveda, updated on project progress. The only meeting times that the whole team is available is generally the 4 hours a week set aside for the Senior Project class. One way to mitigate this problem is to designate one team member as the primary point of contact with our various stakeholders. This person would then be responsible for keeping each of the interested parties up to date and for notifying the other developers of any concerns that may arise.

Quality Assurance

This section details the types of activities that will be performed to ensure the student co-op evaluation system meets the expectations of the customer.

Software Configuration Management

CVS will be used as our version control software. Each developer will have a separate CVS log-in to ensure that proper records of changes are stored.

Weekly Status Meeting

Every week, a meeting will be held to make sure that all developers are on task and that the stakeholders are up-to-date with the latest developments. The meeting will consist of going over a list of action items that each developer has been given. Developers will update the status of their items and receive new action items to work on for the following week.

Defect Tracking

A server based tracking tool (such as Bugzilla) will be set up to record defects in artifacts. Developers will be expected to record any defects found during reviews, inspections, testing, or other development activities. Each defect will be assigned a date found, a date fixed, a priority, a severity, and have fields for assigning a fixer of the defect.

Early project planning sessions will need to account for the time required to decide on an appropriate software package to perform defect tracking. If a custom package is built, time will need to be allotted for the development and testing of the defect tracker.

Usability Testing

One of the main complaints with the employer software is that its laid out poorly. To ensure that the student software doesn’t suffer from that same problem, we will create a simple prototype of the user interface and get feedback from OCECS as well as students that will eventually use the system.

Product Release Checklist

- ITS has product up and running
- ITS and OCECS have received documentation
- Source code and other artifacts archived for ITS or future projects

Product Support Strategy

- Pass project on to ITS to support and maintain the code. For this reason, it is important that ITS be involved in the project early on since they will take over at the end of the project.
- Train ITS, program administrators, and OCECS representatives in the basic installation and usage of the program.