Our Group Software Date: xxx

Project Post Mortem

My Project Title

Note: Text in Yellow to be replaced:

Overview

Company: xxxx Project: YYYYYYY

Purpose

The purpose of this document is to give my findings on what worked and what did not work for our project.

Initial Expectations

The initial expectations on the complexity of this project were pretty low. I was allready given a working prototype of software than merely needed to be improved. A lot of the additions that were to be made were cosmetic. I did not figure that this project would be very difficult to implement.

What Worked

SourceForge.net

Using SourceForge as a cvs repository for our code saved us a lot of time and headache after the initial setup of the project. This was well worth the investment in time. All the code was constantly available on line and it took care of all of our revision control and maintaining of the servers. I can definitely see how e-mailing bits of code would have hindered our project.

Ant

We used Ant as a build tool for our project. This took care of all of the necessary compiling and jar file creation. All that was needed was a double click or a one-line command to build the entire project.

Install Anywhere

This free tool is very powerful and incredibly useful with java applications. It took care of all of our installation needs including options for many different platforms.

Working Prototype

I was effectively given a working prototype of our software from the previous version of PowerPlay. I was able to use this code to create a modified program structure that had already proven itself to work. This also allowed our customers to have a frame of reference to working software to know what features they wanted added, bugs to be fixed, and how they wanted it to work.

Customer Communication

I was in contact with our customers for most of the project. They were given two intermediate versions of the project. From these, they were able to see the progress that we made and well as make suggestions on how they would like it further improved. This greatly improved the quality of the software. This also allowed us to have a wide base of software testers. A lot of people were able to stress the project in order to find errors. This saved us from doing a lot of our own testing and gives us confidence in how well the project works.

Modularization

We divided the project into five major subsystems. Each subsystem was given an owner. Each subsection also had to produce some type of interface or class on how that subsystem would interact with the rest of the subsystems. Once these interfaces were designed the owner could code them in any manner they wished as long as they followed the interface. This resulted in few code conflicts. Most of the conflicts occurred when someone would help someone else with their subsection.

What Did Not Work

Testing

In our project plan we had tasks set aside for creating an extensive test suite that included use of Java's Robot classes to automate user interactive testing. This was a very optimistic view of our project and in the end did not happen at all. We have no formal test suite but have been relying on our use of the project as well as the customer's use of betas of the project. This is something I definitely would change. If we had to do it over again we would have built a formal test suit. Testing is an integral part of any software project. Our project has a significant gap in this area.

CVS Check in Policy

Often times during the project one would do an update and then find that the code no longer compiled for them. This was usually due to the fact that another user added functionality but forgot to check in a new source file or utility. A way to fix this is to have every user maintain two directories where they check out to. One of these directories is devoted to development. The other would be just an update directory to see if the changed code still compiles and does not cause any conflicts.

Project Reflections

I was surprised by the amount of time that this project took us. A lot of the underlining engine for digraph structure control was rewritten to make it less complicated and to make it easier to add functionality. The cosmetic improvements to the project were also quite time consuming. A lot of code went into creating the menu features and toolbar functionality. Overall we were very pleased with the resulting project. We fixed several major bugs, added a lot of useful features, and made the project visually presentable. Our customers were very pleased with our output and it is rewarding to know that we created a system of software that will be used for many years.

Teamwork Evaluation

Was the project work load evenly distributed among the team members?

Was the work of any team members particularly praise worthy?

Was the work of any team members particularly deficient in your opinion?

Course Improvements

Five things I liked about this course:

Five things I did not like about this course (and suggested improvements)