GUS

Method of Development

Project Goal

Create a useful and powerful group management tool within the timeframe of Software Engineering 1 and 2. Because of the nature of this time frame, as well as the project itself the style of development needs to work quickly, and efficiently.

Description

The project team has been tasked with using an agile method of development. There are many different kinds of agile development. Those which seem most suited to this project are Test Driven Development, Scrum, Feature Driven Development, and Extreme Programming.

Test Driven Development seems like a good choice at first. However, at second glance it isn't quite what is needed. In using TDD programmers write a test, and create code to pass the test. This project has users, who will be able to perform many different tasks, which often cannot be done as easily by code. Therefore creating a test, and then a program to pass it will be quite difficult.

While the concept here is useful, the method itself is not the best. The programmers will need to be able to understand what the program should be doing, and write code to perform those tasks. However, the idea of writing a test first and passing that test is un-realistic. The tests will only truly be done when the end user is satisfied with the functionality.

Scrum also appears to be a good fit, because of the time constraints. This style allows for fast development, and many meetings about problems that occur during the development and testing, as well as solutions to those problems. However, Scrum requires that after each 'sprint' (a two to four week period of programming) that the thing which was created be a shippable product increment (working and tested software). Because of other requirements in the Software Engineering class, this is unrealistic. Much of the programming is to take place during the second semester of the class, rather than the first.

Scrum's also say to create a 'shippable' product each few weeks will only have small, separately unusable, features (many features only work when others are also working, ex: events and calendars will potentially use each other). The overall concept here is still useful however. Working to create small shippable products can still work, if applied to the right pieces, and at the right times. Because of that constraint, Scrum is not the best method to use for the whole project, but perhaps only on a few pieces.

Feature Driven Development is the next step up. It uses a similar form as Test Driven Development, in that programmers work toward solvable tasks, rather than an overall view of the project. Chunks of code are created in an effort to create specific features which have been identified, rather than trying to solve the problem of the entire project at once. However, this can cause a focusing of work down to individual features, and the whole project may not come together well.

Much like the Scrum method, creating for features is still useful. The team will need to be able to create features on the fly, if and when the customer refines their requirements. The software specification also has what amounts to a list of features, rather than a problem statement. This plays well to Feature Driven Development. The problem occurs when one applies FDD to an entire project of this scope, with this many people working on it. The features can be spread out enough that when re-assembled the end product may be entirely unusable.

Extreme programming is a combination of several of these ideas, as well as many others. It focuses on making sure features are needed before they are created, to prevent un-needed coding. Programming is often done in pairs, which suites the class requirements well. There is extensive code review, and unit testing of all code which allows for more robust, simple, and clean coding.

Extreme Programming also allows for a regular change to the requirements from the customer, letting the concept of GUS be able to change on a more regular basis, as features appear and customers refine what they want in the software. This also allows for frequent communication within the project team, preventing the kind of focusing that might happen in Feature Driven Development. XP also uses an aspect of Scrum, small shippable increments. The big difference here, is there is no week limit, each 'sprint' is done at a sustainable pace, rather than a time limit.

When those shippable increments are done, the Test Driven piece of the puzzle can also be applied. Those increments can be sent to select customers, or simply be used in house by a different section (than who created those features) of the team; and be extensively tested and fixed before moving on to the next section of the project.

Resources

The way Extreme Programming works is a constant interaction of both team and customers. This requires a small amount of resources. Those resources are things such as time and locations. Time will be needed by each team member, in order to be able to keep in good contact with the rest of the team. They will also need to be able to find times to meet with customers and present what they have, and receive feedback. Locations will also be required

for any meetings done not through digital communication. Both of these things can be easily attained with the use of online calendaring, and university meeting/conference rooms. These requirements will likely be the same no matter what development method is chosen.

Action

Extreme Programming should be used as the agile method of development for the GUS project. It meets the requirements for the Software Engineering class, as well as creates a smooth combination of other methods of development. There are many different methods of development, but each has something useful. Extreme programming brings all those different useful bits together, while doing the best at keeping the less useful pieces out of the process. Once those methods are combined they create a method of development which allows for well written and effective programs.

Jacob Flynn, 10/24/10
jacobflynn@vandals.uidaho.edu