Agile Method Analysis

Scott Beddall
Lee Vangundy
Steven Long
Chandler Abraham

October 23rd, 2010

Abstract

The goal of this publication is to analyze various Agile methods and determine which best fits the requirements of the GUS project.

1 Introduction

Before development of GUS code can begin, a method of implementation must be decided. As development of the F-22 Joint Strike Fighter has proven, selecting the right project management methodology is paramount to the success of any software development process.

GUS will be developed as a core module first, with several additional features being bolted on later if time allows.This makes it readily obvious that in choosing our Agile methodology, we will want a method that will promote design of a core module along with development driven by features. In this way, our team will be able to distribute workload and keep the code size comprehensible. Additionally, if done correctly, we will also promote the design goals sketched out in class. That is: minimal complexity, high cohesion, and low inter-dependence between modules.

To this end, an analysis of possible Agile methods must be performed. This analysis need not be extremely in-depth, as the description of each method will make its specialty readily apparent. Following is a description of several Agile methods, along with summaries that relate them to the GUS team. Once each is described, it will be easy to determine which methodology best fits development for GUS.

2 Method Proposals

Dr. Jeffery suggested a different management methodologies relevant to GUS. The correct rationale must be applied to decide on which one to use for our project.

2.1 Extreme Programming

Extreme Programming, as a method of software development, focuses on producing code FIRST. This can be beneficial in that code (and a lot of it) is delivered. Programmers who are using this method need to be able to interact with customers effectively; as they are the primary resource from which the direction of the project will be derived. XP argues that most important resource is code. This code should be well designed, simple, and implemented with confidence that the new code is correct. This confidence is gained from customer feedback, unit tests, and direct interaction with the customer during development.

Unfortunately, while GUS will need to develop a lot of code, a more structured approach utilizing less customer input will be needed. In essence, the team is its own customer.

2.2 Feature-Driven Development

Feature-Driven development is the one of the easiest Agile methods to understand. Its goal is simple: develop and overall model, build feature list, plan by feature, design by

feature, build by feature. This method ensures that each build is its own milestone, with a coded and tested feature added at each iteration. The overall development model shouldn't change over the course of the project, but if this does need to happen, it shouldn't be a problem unless the change is to an existing feature. Adding features is not difficult with this method.

In my opinion, this method is the most likely candidate for implementation by the GUS team. Not only is it easy to understand, but it will also allow us to separate the different modules of GUS into individual development cycles.

2.3 Scrum

The SCRUM Agile method is one that is currently heavily used. (example:F-22) It is based around 3 main characteristics that define the skeleton. The first characteristic specifies three roles: SCRUM Master (project director), Product Owner (customers), and the "Team" (developers). These roles come together in a sprint of preset length. This length is determined at the outset of the project, and should not be changed unless under the most dire of circumstances. Daily 15-minute meetings will take place every day to discuss progress since the last meeting. Only developers can speak during these meetings, as they are the ones doing the actual coding. The end of each sprint should result in a working increment of the software; with that increment's backlogs being added to the product backlog, and used as a starting point for the next sprint.

The Scrum Agile method is one that makes sense for the core module of GUS implementation. While it will need to be adjusted for the fact that students won't be able to meet every day to discuss progress, the core of the idea will be the same. This method, like FDD, will ensure that at each cycle the team will have a working version of our software that can be used for demos or implementation.

2.4 Test-Driven Development

Test-Driven Development is, as a method, used exactly as the name suggests. Development teams operate on a very short development cycles. First, a test-case is designed using developer and customer input, then code is produced to successfully pass that case. If necessary, code is refactored to meet any industry standards. These test-cases might not even be specific to a new feature, they might be relevant to a core module that is already completed. It must be mentioned that all tests are archived, and executed along with any new test-case. In this way, the development team is assured that they are not breaking any old features with their new code.

As a method, TDD could fit GUS's needs very well. However, the test-cases for an entire feature might be a bit too complicated to work correctly. If that is the case, then we might scrap the method for a more feature-oriented one.

2.5 Pair Programming

Pair Programming as a method could be described as a facet of Extreme Programming. It promotes synergy between coders and/or customers. The method boils down to having programmers work in teams. One types in code (the driver) while the other reviews each line of code as it is entered (the observer). The entire point of the method is cut down on the amount of debugging that will be required at the end of each iteration of the project. Some could argue, however, that the loss of an entire programmer's work day is more detrimental than the amount of time saved in debugging.

Unfortunately, this is merely a programming method designed to ensure the correctness of code inside a much larger framework. GUS needs a Agile method that primarily directs the way our project timeline is laid out. This doesn't rule our Pair Programming from use at some point, but it will not be our method of choice.

3 Conclusion

As a whole, while most of these methods have promise, I don't feel they all quite satisfy our needs for GUS development. There are, however, a couple that seem to fit our requirements exactly. As earlier stated, both Agile methods Feature-Driven Development and SCRUM were very likely candidates. That point still stands, but I don't feel that individually they will cut it. Therefor, I propose a compromise. We do both. This isn't a proposal that seems even close to standard, but I do feel that it fits our needs as a team and as individuals.

The rational is quite simple, SCRUM is designed for more complex, very highly coupled projects; projects that need to have their entire core built concurrently with its modules. Feature-Driven Development is almost the exact opposite. This method is also incremental, but is more focused on implementing specific features one at a time.

With this information in mind, I propose that as a team we implement GUS using an amalgamation of the two afore-said methods. The core module of GUS needs to be written and tested for completedness before any additional features can be added. The SCRUM Agile method should be used in this endeavor. However, once the core module is implemented and feature implementation begins, we should switch methods to Feature Driven Development. Our overall model of features and requirements elicitation will have been completed earlier, so we can move on to designing and implementing each individual feature. In this way we will maintain stability of our core APIs while also successfully completing our project goals.