Alexander Ramsey  
SWEN 603  
July 20, 2016

Feature-driven Development Process

Feature driven development “FDD” is a methodology that follows an iterative process that is designed to lay the groundwork for larger projects. Originally proposed by Jeff De Luca in 1997, the process has remained relevant to today as an agile related process that embraces a mix of traditional waterfall style structure with some of the people related flexibility of Agile.

Feature driven development builds upon the traditional waterfall model in that the requirements phase marks the start of the project. Committees form to analyze the requirements and outline what models are needed to accomplish each of the requirements. If the requirements are clear to the committee, they will move on to present them to the larger group of developers. This way, the micro model can be consumed with the macro model of the overall team. This allows for more scalability throughout the process, hence why FDD is best for larger projects.

An overall Object Model is then refined at the larger team level. If requirements are not clear to the team, they will take more notes and form some communication channel with the client to ensure that the requirements are known and understood well. At this point, an additional iteration will occur, where the model is adjusted to conform to any new information brought to light.

Once the high level walkthrough, domain model, and peer review occurs, the next step is to outline the features list. The features list allows the development team to put the requirements into an easy to understand perspective that may offer additional insight into the business requirements of the program. A pattern for formulating these features is followed: action, result, object. A good example of a feature as stated is, “Display the username in the upper header of each webpage.” These features are developed into a domain categorized list. Each domain is then distributed as working projects to smaller development teams. These development teams take ownership of the problem domain and develop classes that work off of the agreed upon model established in the requirements phase.

Once each development team takes ownership, they enter the plan by feature phase. Chief programmers enter each of the domain areas to examine the design packages created by the small teams. Sequence diagrams and further documentation is generated, following the FDD documentation. At this point, building of features occurs. Unit testing of the feature occurs and if it matches the requirements of the client, it can be implemented into the final build of the project.

A simple diagram displaying the FDD process:

Build by Feature

Design Classes by Feature

Plan Classes by Features

Build Features

Develop Model

FDD allows the team to regain control over the project while maintaining enough client communication to allow for flexibility in changing requirements. In addition, it establishes a clear group oriented approach that is helpful for larger development projects. The structure of FDD is centered largely around the time boxing of feature development. This can be initially a challenging process for budding teams. However, by estimating the development of features over time, the team can learn with each other about estimating future time boxes. A downside of FDD is that it requires a significant amount of time to plan each step. Although this time is useful, it has been shown that planning affects the result of the project being within budget and timeframe (Hunt, 2006).

In review, FDD is a feature-centric, time box focused, and adaptive for changing requirements. It meets the needs of modern, agile teams while maintaining the best of the structural documents and processes utilized in older iterative based methodologies.

References:

Hunt, J. (2006). Feature-driven development. *Agile Software Construction*, 161-182.