

Build Your Project Using Feature Driven Development

#4 of a Series, by Pavan Kumar Gorakavi, M.S., M.B.A, G.M.C.P, C.A.P.M.

1. What is Feature Driven Development?

Feature Driven Development is a agile software methodology that gained significance in the early 2000s. Feature Driven Development (FDD) was first reported by Peter Coad, and was further developed by Jeff Luca and Stephen Palmer. According to Palmer and Felsing, *Feature Driven Development* focuses on the design and building phases. FDD is flexibly designed to work with any specific process model of software development. Like other agile methodologies, FDD advises incremental development by practicing best practices and frequent monitoring.

In the 1990s, when many project developers were looking for alternative software development methods because of their dislike for rigid conventional methodologies, agile alternatives came into the picture. FDD is one of the agile practices that looks promising for the problems caused by the long development cycles of traditional development models. FDD is an enhancement of iterative and incremental approaches. FDD enables the reliable delivery of working software in a disciplined fashion with information freely flowing across the project.

In early 2000 when software developers were feeling the heat of increasing complexity of agile methodologies due to wide range of factors, Feature Driven Development rose in popularity. FDD practices apply a *Feature centric process*. A Feature centric process helps in handling a project by considering the following factors [1]

- What must we do next to add value to the customer or client?
- How are we progressing against time and budgets?
- What issues and risks do the project face?
- How can the issues and risks be addressed or mitigated?
- What should we do next?

Feature-driven development (FDD) is just one example of a feature-centric process. There are other feature centric development processes apart from FDD. The development process EVO is feature-centric and so is, to some extent, DSDM, with its requirements catalogue.

Feature Driven Development consists of four sequential processes during system design and development of solution. FDD classifies its players into three categories: key roles, supporting roles, and additional roles (Palmer and Felsing). Roles include Project Manager, Chief Architect, Development Managers, Team Leads, Class Owners, and Domain Experts, all playing a vital role in formulating features, prioritizing, and designing a business solution.

2. Life cycle of FDD

FDD provides a flexibility of changing the requirements at later part of the game. This characteristic helps in building a product using trial and error prototype. FDD consists of five sequential processes during design and development of business solution. The five sequential processes includes:

- Develop an overall model,
- Build a feature list,
- Plan by features,
- Design by feature and build by feature.

The Life cycle of FDD methodology is illustrated in figure 1, below.

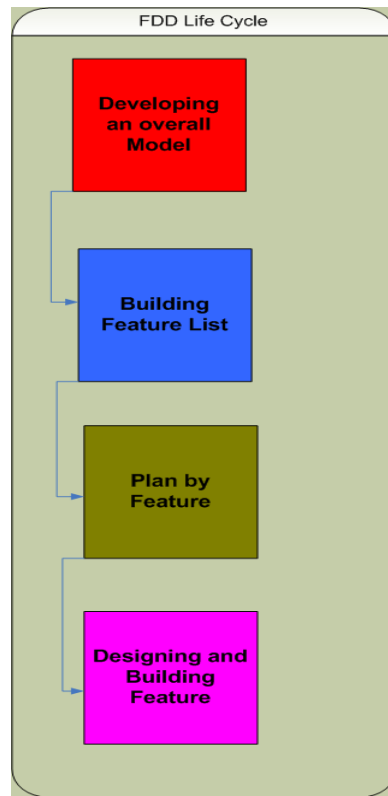


Figure 1: Life cycle of FDD

Phase 1: Building Overall Model

In this phase overall domain descriptions are prepared. Having gained an understanding of needed business functionality, domain expertise, and overall scope of the project, the key players initiate overall model design. Documented requirements are organized with proper use-cases and technical specifications. Domain experts walkthrough the requirements with team members, chief architects and other key team players. After the initial walkthrough, the overall domain is further divided into different domain areas. A detailed walkthrough is held for domain members. After each walkthrough, small development teams work together effectively to produce an object model. This can be illustrated in figure 2.



Figure 2: Building Overall Model

Phase 2: Building Features List

This phase conducts a initial project wide activity to identify all the features to support the Overall model(s) designed in Phase 1. The Feature list designed in this phase is intended to address all identified requirements. This Phase can also be considered to be the functional decomposition of the Domain Model obtained from Phase 1. Major features list are prepared, then further divided into feature sets. A Review Panel, including Domain Experts, Chief Programmers, Architects and other key players, performs value-analysis on the feature list. Features are listed in a <Requirement> <Result expected> format.

Eg: Calculate the total of course fees for all students; these results are used to calculate total revenue of an organization.

Phase 3: Plan by Feature

This phase is an initial project-wide activity which produces the development (design and build) High-level Plan. This High-level plan lists all features, based on their priority and their dependencies. The Project Manager, Development Managers, Chief Programmers and other key players act together to prepare a sequential list of all features listed in phase 2, and sorts the features based on their priority. Identifying interdependencies between features prior to implementation is an important part of this phase.

Schedule and major milestones are set for this feature test. A project schedule is identified, considering the following:

- Interdependencies between the features.
- Balancing workload across different teams and class owners.
- Risk factors involved in implementing the features.
- Complexities involved in implementing the features.
- Any other external factors.

Phase 4: Design and build features

After obtaining their set of features list with priorities, Class owners help form their feature teams. Feature teams handle a small group of features, which are a subset of the feature list developed in phase 3. Design and development are incremental in nature.

Each iteration is generally scheduled for 1-2 weeks. In this phase, the system goes through a sequential process of product development: Design, Development, Unit Testing, Integration and System Testing. After successful iteration, the completed feature is pushed to the main build, while the next Design and Build iteration starts with a new set of features.

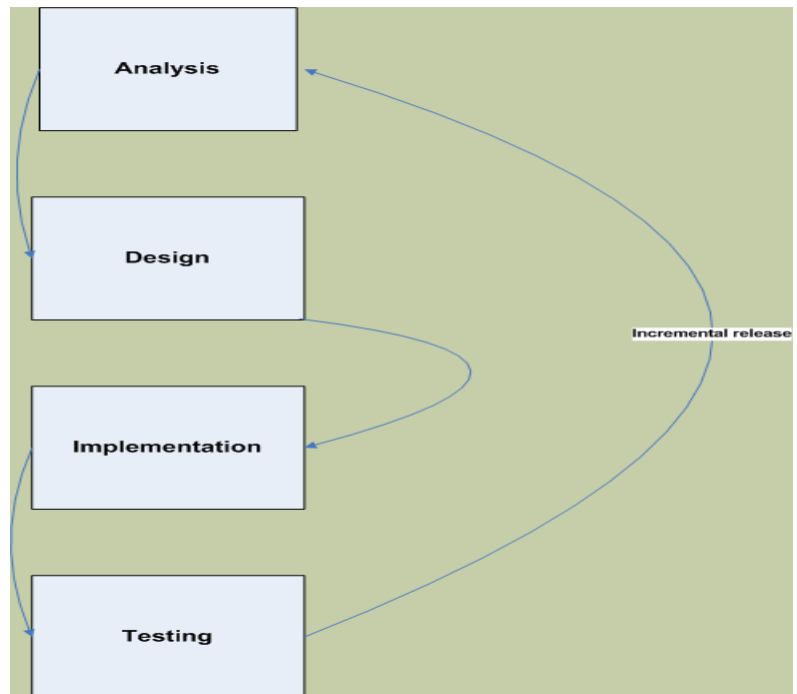


Figure 3: SDLC cycle

Classes are migrated to the build after a successful code inspection. The Chief Programmer is accountable for individual classes being promoted, through feedback from the developers.

The Chief Programmer /Team Leads are responsible for helping form Feature Teams by identifying the owners of the classes likely to be involved in the development of each set of features. Each Feature team documents the developmental sequence using sequence diagrams for the assigned features. The teams update the Object Model based on the content of each sequence diagram. The Feature team developers write class and method prologues. A periodic design Inspection assures effective design.

3. Different Actors of Feature Driven Development

According to Palmer and Felsing 2002, FDD classifies its roles into three categories: Key roles, supporting roles, and additional roles. The six key roles in a FDD project are: Project Manager, Chief Architect, Development Manager, Chief Programmer, Class Owner, and Domain Experts.

The five supporting roles comprise Release Manager, Language Lawyer, Build Engineer, Tool Smith, and System Administrator. Testers, Deployers and Technical Writers also play a vital role in FDD development. Figure 4 illustrates the key roles.

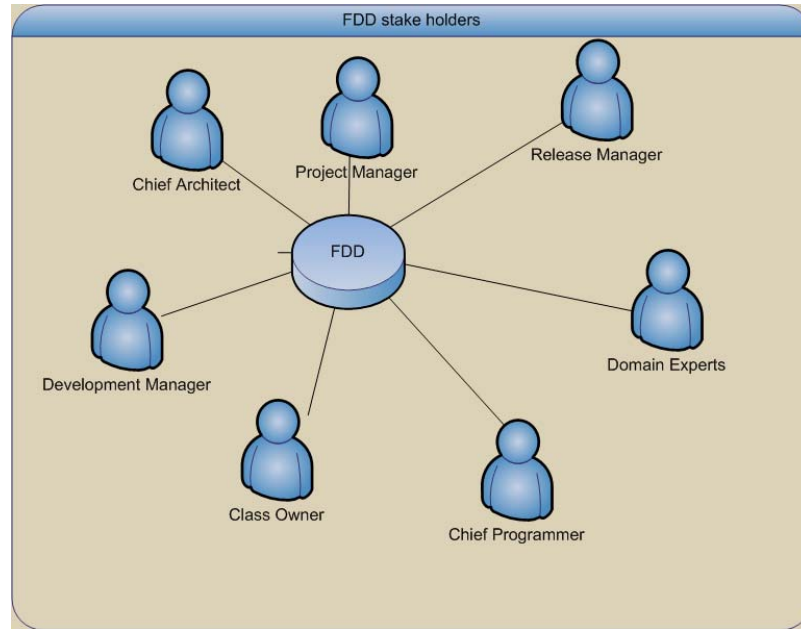


Figure 4: FDD stakeholders

Project Manager

The Project Manager is the person who has ultimate say in the lifecycle of FDD. The Project Manager looks after administration and financial aspects of the project, and drives and controls the team focus towards project outcomes. The Project Manager also acts as a facilitator for the team members and strives to provide suitable working conditions for team players.

Chief Architect

The Chief Architect is responsible for the system design. He/She is responsible for overall design of the system and he conducts design sessions, and review sessions held with the team to help in identifying risks proactively. Based on the complexity of the system, the Chief Architect can distribute the architectural work among different system architects.

Development Manager

The Development Manager is responsible for monitoring daily developmental activities, identifies risks proactively, resolves any issues, plans releases and plans resources. The Development Manager acts as a coordinator so the team will have a smooth product release.

Class Owner

Class Owners are responsible for the formation of feature teams and building the assigned class(es). Class Owners report to Chief Programmers. Class Owners are accountable for analysis, design, implementation, testing and documentation of the class. Class owners plan for the next iteration, monitor current iteration progress, and support any previously developed classes.

Chief Programmer

The Chief Programmers are responsible for identifying different classes and the Class Owners. They play an active role in the analysis, design, implement, test and documentation phases of the project, and a vital role in Phase 4: design and development of features. Chief Programmers communicate frequently with each other in order to identify risks, reduce complexity, and analyze technical feasibilities.

Domain Experts

Domain experts are persons who possess wide domain knowledge and understand system behavior. Domain experts can be users, business owners, business analyst, and clients. Domain Experts communicate the business logic to key players of the project. A Domain Manager leads Domain Experts and strives for smooth communication flow.

4. Conclusion

Palmer and Felsing suggest that Feature Driven Development is suitable for new projects, project enhancements, and project upgrades. FDD must be adopted gradually. Feature driven development offers salient features, such as: Domain object modeling, Class level accountability, Feature teams, Regular builds, Configuration management, and frequent monitoring, all helping to develop a robust system.

References:

- [1] Coad, P., LeFebvre, and De Luca. Java Modeling in color with UML. Prentice Hall
- [2] Palmer, S.R. and Felsing, J.M. A practical guide to feature driven development, Prentice Hall.
- [3] Hunt, John. Agile Software Construction. Springer. © 2006
- [4] Wysocki, Robert K. Effective Software Project Management. John Wiley & Sons.
- [5] Koch, Alan S. Agile Software Development: Evaluating the Methods for Your Organization. Artech House.
- [6] Schuh, Peter. Integrating Agile Development in the Real World. Charles River Media.

About the Author

Pavan Kumar Gorakavi is working as a Senior Software Developer in Dallas, TX. He is settled in Dallas, TX with his family (wife Swapna Gorakavi and son Anish Gorakavi). He is VP - programs for *asapm* Young Crew. He is also acting as Associate Director [Marketing] for PMI-ISSIG.

Pavan earned his Bachelor's degree in computer science from Jawaharlal Nehru Technological University and Masters in computer science from Lamar University. He did his MBA from University of Texas at Dallas and GMCP from Southern Methodist University. Pavan holds SUN, IBM and PM Institute certifications.

Pavan Gorakavi authored a book on 'Artificial Intelligence' published by Rahul publications - India, and 'Digital Electronics' published by Subhash publications, India. His research interests are Artificial Intelligence, Agile methodologies, and Software development in ADA, Prolog and Java. You can reach Pavan at gorakavi@gmail.com.



About this Series

This article is the fourth in a series by Mr. Gorakavi on Agile, posted on the *asapm* website; watch for the others in the series. And, although the concepts of Agile are most-common in Software Development projects, increasingly Agile and Lean PM methods are also turning up in many other project areas, including Engineering and Manufacturing, where some assert they actually originated.