WIKIPEDIA

Feature-driven development

Feature-driven development (**FDD**) is an iterative and incremental software development process. It is a lightweight or Agile method for developing software. FDD blends a number of industry-recognized best practices into a cohesive whole. These practices are driven from a client-valued functionality (feature) perspective. Its main purpose is to deliver tangible, working software repeatedly in a timely manner in accordance with the Principles behind the Agile Manifesto.[1]

Contents

History

FDD was initially devised by Jeff De Luca to meet the specific needs of a 15-month, 50-person software development project at a large Singapore bank in 1997. This resulted in a set of five processes that covered the development of an overall model and the listing, planning, design, and building of features. The first process is heavily influenced by Peter Coad's approach to object modelling. The second process incorporates Coad's ideas of using a feature list to manage functional requirements and development tasks. The other processes are a result of Jeff De Luca's experience. There have been several implementations of FDD since its successful use on the Singapore project.

The description of FDD was first introduced to the world in Chapter 6 of the book *Java modelling in Color with UML*[1] by Peter Coad, Eric Lefebvre, and Jeff De Luca in 1999. Later, in Stephen Palmer and Mac Felsing's book *A Practical Guide to Feature-Driven Development*[2] (published in 2002), a more general description of FDD was given decoupled from Java modelling.

Overview

FDD is a model-driven short-iteration process that consists of five basic activities. For accurate state reporting and keeping track of the software development project, milestones that mark the progress made on each feature are defined. This section gives a high level overview of the activities. In the figure on the right, the meta-process model for these activities is displayed. During the first two sequential activities, an overall model shape is established. The final three activities are iterated for each feature.

Develop overall model

The FDD project starts with a high-level walkthrough of the scope of the system and its context. Next, detailed domain models are created for each modelling area by small groups and presented for peer review. One or more of the proposed models are selected to become the model for each domain area. Domain area models are progressively merged into an overall model.

Build feature list

Knowledge gathered during the initial modeling is used to identify a list of features by functionally decomposing the domain into subject areas. Subject areas each contain business activities, and the steps within each business activity form the basis for a categorized feature list. Features in this respect are small pieces of client-valued functions expressed in the form "<action> <result> <object>", for example: 'Calculate the total of a sale' or 'Validate the password of a user'. Features should not take more than two weeks to complete, else they should be broken down into smaller pieces.

Plan by feature

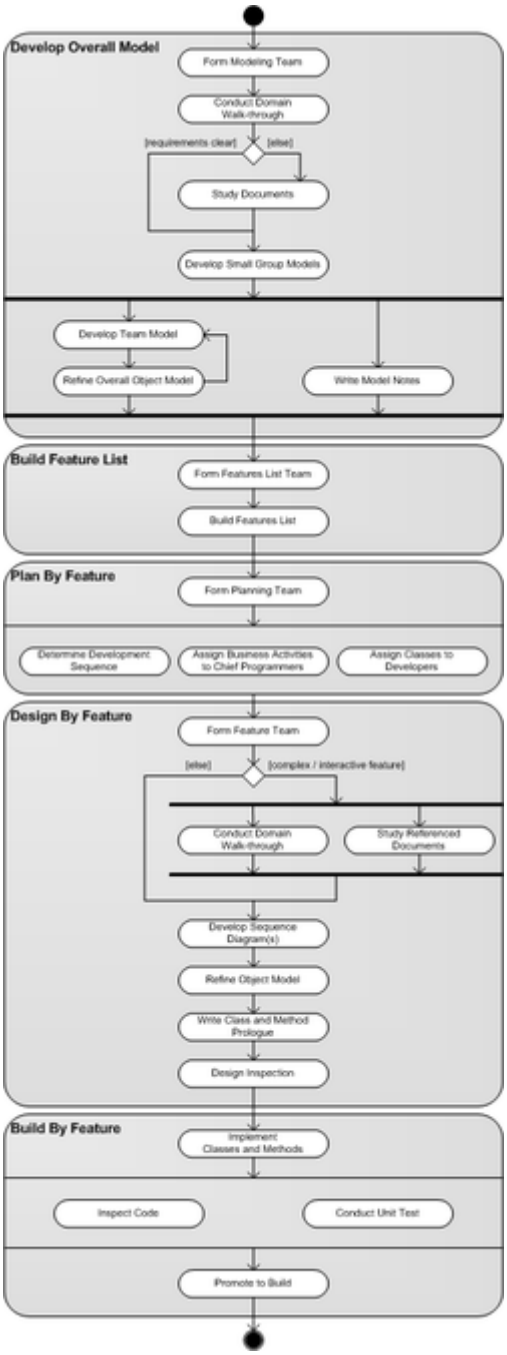
After the feature list is completed, the next step is to produce the development plan and assign ownership of features (or feature sets) as classes to programmers.

Design by feature

A design package is produced for each feature. A chief programmer selects a small group of features that are to be developed within two weeks. Together with the corresponding class owners, the chief programmer works out detailed sequence diagrams for each feature and refines the overall model. Next, the class and method prologues are written and finally a design inspection is held.

Build by feature

After a successful design inspection for each activity to produce a feature is planned, the class owners develop code for their classes. After unit testing and successful code inspection, the completed feature is promoted to the main build.



Process model for FDD

Milestones

Since features are small, completing a feature is a relatively small task. For accurate state reporting and keeping track of the software development project, it is important to mark the progress made on each feature. FDD therefore defines six milestones per feature that are to be completed sequentially. The first three milestones are completed during the Design By Feature activity, and the last three are completed during the Build By Feature activity. To track progress, a percentage complete is assigned to each milestone. In the table below the milestones and their completion percentage are shown. At the point that coding begins, a feature is already 44% complete (Domain Walkthrough 1%, Design 40% and Design Inspection 3% = 44%).

Table 1: Milestones

Domain Walkthrough	Design	Design Inspection	Code	Code Inspection	Promote To Build
1%	40%	3%	45%	10%	1%

Best practices

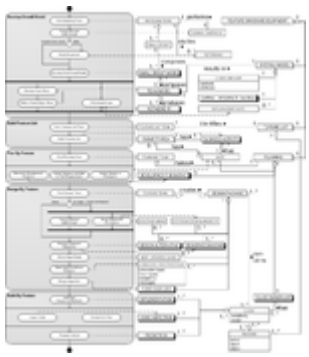
Feature-driven development is built on a core set of software engineering best practices aimed at a client-valued feature perspective.

- **Domain Object modelling**. Domain Object modelling consists of exploring and explaining the domain of the problem to be solved. The resulting domain object model provides an overall framework in which to add features.
- **Developing by Feature**. Any function that is too complex to be implemented within two weeks is further decomposed into smaller functions until each sub-problem is small enough to be called a feature. This makes it easier to deliver correct functions and to extend or modify the system.
- **Individual Class (Code) Ownership**. Individual class ownership means that distinct pieces or grouping of code are assigned to a single owner. The owner is responsible for the consistency, performance, and conceptual integrity of the class.
- **Feature Teams**. A feature team is a small, dynamically formed team that develops a small activity. Multiple minds are always applied to each design decision, and multiple design options are evaluated before one is chosen.
- **Inspections**. Inspections are carried out to ensure good quality design and code primarily by detection of defects.
- **Configuration Management**. Configuration management helps with identifying the source code for all features that have been completed to date and maintaining a history of changes to classes as feature teams enhance them.
- **Regular Builds**. Regular builds ensure there is always an up-to-date system that can be demonstrated to the client and helps highlight integration errors of source code for the features early.
- **Visibility of progress and results**. Managers steer a project using frequent, appropriate, and accurate progress reporting from all levels inside and outside the project based on completed work.

Metamodel (Metamodelling)

Metamodelling helps visualize both the processes and the data of a method. This allows methods to be compared, and method fragments in the method engineering process can easily be reused. Usage of this technique is consistent with UML standards.

The left side of the metadata model shows the five basic activities involved in a software development project using FDD. The activities all contain sub-activities that corresponding to sub-activities in the FDD process description. The right side of the model shows the concepts involved. These concepts originate from the activities depicted in the left side of the diagram.



Process-Data Model for FDD

Tools used

- CASE Spec (http://www.casespec.net). CASE Spec is a commercial enterprise tool for feature-driven development.
- TechExcel DevSuite (https://web.archive.org/web/20070220235300/http://www.techexcel.com/solutions/alm/fdd.html). TechExcel DevSuite is a commercial suite of applications to enable feature-driven development.
- FDD Tools (http://fddtools.sourceforge.net/). The FDD Tools project aims to produce an open source, cross-platform toolkit supporting the Feature Driven Development methodology.
- FDD Viewer (http://www.featuredrivendevelopment.com/node/687). FDD Viewer is a utility to display and print parking lots.

See also

- Agile software development
- Behavior-driven development
- Project lifecycle

- Software architecture
- Software development process
- Software engineering

References

1. "Principles behind the Agile Manifesto" (http://agilemanifesto.org/principles.html). 2019-06-11.

- 1. ^ Coad, P., Lefebvre, E. & De Luca, J. (1999). *Java modelling In Color With UML: Enterprise Components and Process*. Prentice Hall International. (ISBN 0-13-011510-X)
- 2. ^ Palmer, S.R., & Felsing, J.M. (2002). *A Practical Guide to Feature-Driven Development*. Prentice Hall. (ISBN 0-13-067615-2)

External links

- Feature Driven Development Community (http://www.featuredrivendevelopment.com/)
- Feature-driven development (https://curlie.org/Computers/Programming/Methodologies/Agile/Feature_Driven_Development) at Curlie
- Nebulon FDD Page (http://www.nebulon.com/fdd/index.html) - Nebulon is the consulting practice of Jeff De Luca
- Successful Web Development Methodologies (http://www.sitepoint.com/article/successful-development) - Use of FDD for Web Development projects
- Delivering Real Business Value using Feature Driven Development (http://www.methodsandtools.com/archive/archive.php?id=19) - Article gives basic overview of FDD
- FDD and Agile modelling (http://www.agilemodeling.com/essays/fdd.htm)
- Better Software Faster (https://web.archive.org/web/20071102073447/http://www.bettersoftwarefaster.com/index.htm) - Another book in the Coad Series referencing Feature Driven Development. Authors Andy Carmichael and Dan Haywood ISBN 0-13-008752-1
- Interview with FDD-Creator Jeff DeLuca (http://www.se-radio.net/2008/01/episode-83-jeff-deluca-on-feature-driven-development/) (Podcast)