Feature-Driven Development

Archene Buena

Niño Rogin Flordeliz

Feature-Driven Development (FDD)

- is a client-centric, architecture-centric, and pragmatic software process.

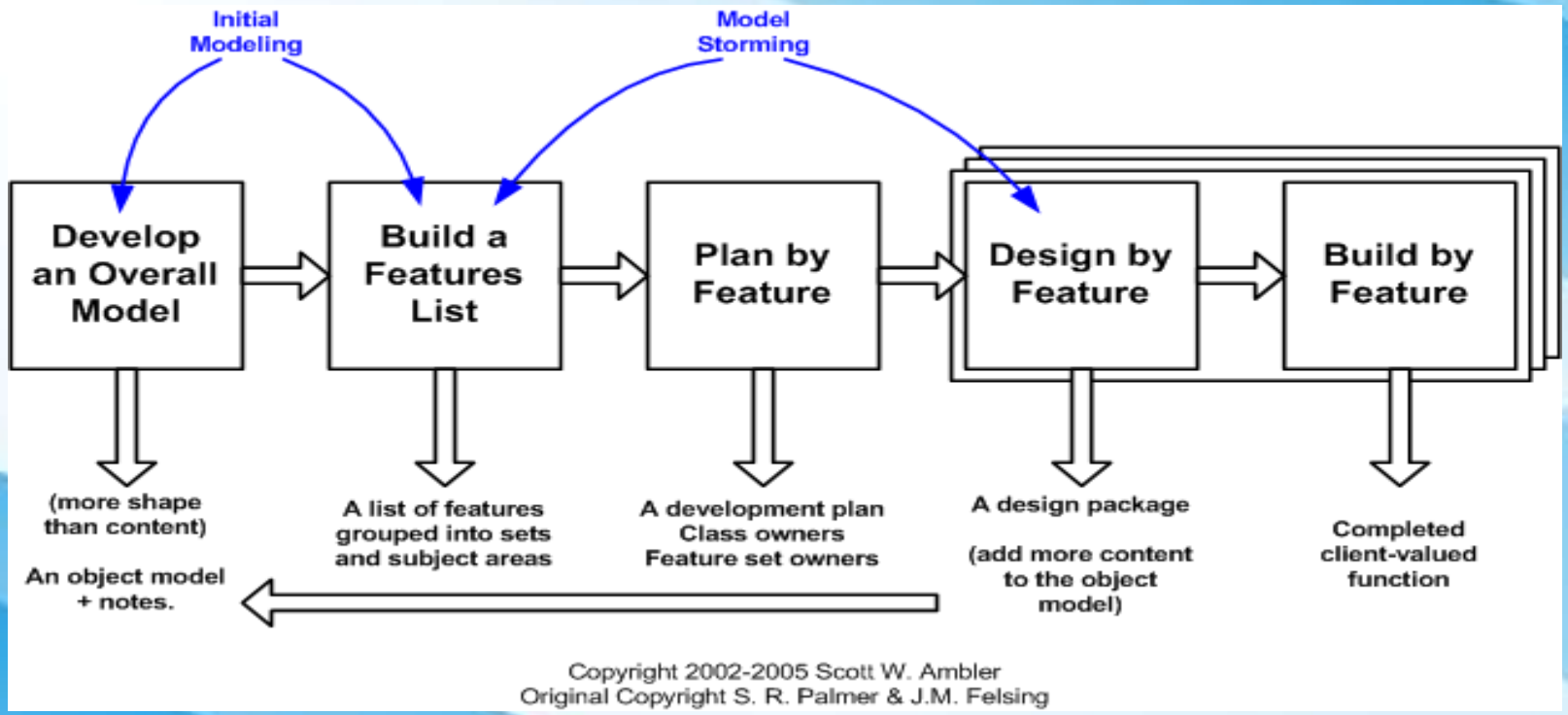
- The term "client" in FDD is used to represent what Agile Modeling (AM) refers to as project stakeholders or eXtreme Programming (XP) calls customers.

FDD’s first

- FDD was first introduced to the world in 1999 via the book Java Modeling In Color with UML, a combination of the software process followed by Jeff DeLuca's company and Peter Coad's concept of features.

- FDD was first applied on a 15 month, 50-person project for a large Singapore bank in 1997, which was immediately followed by a second, 18-month long 250-person project.

As the name implies, features are an important aspect of FDD. A feature is a small, client-valued function expressed in the form <action><result><object>. For example, "Calculate the total of a sale", "Validate the password of a user", and "Authorize the sales transaction of a customer".


Initial Modeling
Model Storming
Develop an Overall Model
Build a Features List
Plan by Feature
Design by Feature
Build by Feature
(more shape than content)
An object model + notes.
A list of features grouped into sets and subject areas
A development plan
Class owners
Feature set owners
A design package
(add more content to the object model)
Completed client-valued function
Copyright 2002-2005 Scott W. Ambler
Original Copyright S. R. Palmer & J.M. Felsing

The History Of Feature Driven Development

The idea of FDD was created by Jeff Luca in 1997 to meet the software development needs of a Singapore bank. His solution was a group of five processes designed to cover the model's development and also its listing, design, planning and the building of its features.

Since its original implication, FDD, and its five basic activities, have continually been used to develop enterprise software because it is seen as both agile and pragmatic.

It takes key advantages of eXtreme Programming and Scrum and combines them with model-centric techniques including Domain-Driven Design by Eric Evan and modelling in colour by Peter Coad. In addition, it is easily scaled to large teams due to its concept of just enough design initially (JEDI), as well as peer reviews and dynamic feature teams.

- Process one – develop an overall model: The FDD model insists that teams exert the adequate amount of effort at the start of the project in order to build an object model highlighting the domain problem. Modelling with feature driven development is time-boxed and collaborative. Domain models should be created in detail by small groups and then presented for peers to review. It is hoped that a proposed model – or potentially a combination of them – will then be used for each area of the domain. They will then be merged over time to produce an overall model.

- Process two – build a feature list: From the knowledge that is obtained during the modelling process, a list of features is established by dividing domains into subject areas that contain information on business activities. The steps that are used for each business activity represent a categorised list of features. Features are expressed in the form of: “action, result, object”. The expectation is that they will not take more than two weeks to complete: if they do, they should be broken into smaller sections.

- Process three – plan by features: Once the feature list has been established, the following process involves assigning the various feature sets to the programmers.
- Process four – design by feature: Programmers then produce a design package for each feature with a chief programmer selecting a group of features that should be developed within a two-week period. The chief programmer will also establish detailed diagrams for each feature while refining the model. When this is complete, prologues are produced and a design inspection is carried out.

- Process five – build by feature: Once design inspections are complete, designers plan an activity for each feature and develop the code for their respective classes. When the inspection is complete and a unit test carried out, the feature is then pushed to the main build.

Feature driven development: best practices

At the heart of FDD are a number of best practises designed for software engineering: all of which are formed from a client's perspective. Some of the best practices that should be followed by developers include:

- Domain object modelling: Explores and explains the problem that needs to be sold and provides a framework from which features can be added.

- Developing by feature: Functions that cannot be implemented within two weeks should be divided into smaller functions: with each sub-problem to be small enough to be labelled as a feature. This should make it easier to modify the system and ensure that the correct functions are offered.
- Individual class ownership: Groups of code should be assigned to individual owners.

- Feature teams: A dynamic small team that focuses on individual activities. This ensures that multiple minds are used on each design decision so that several options are always explored. Typically roles will include – a project manager, chief architect, development manager, domain expert, class owner and chief programmer.
- Inspections: They should be carried out regularly to ensure there are no defects.

- Configuration management: Identifies source code for all features and maintains a history of the changes that are made so that the feature teams can enhance them.
- Regular builds: Ensures that the system is always up to date.
- Visibility of progress: Progress reporting should be carried out on a regular basis to ensure that managers can steer the project in the right direction.