SSW-555-9

Feature Driven Development

Cocomo: How do we scale agile methods to larger projects?

- many more dev working much longer times
- Techniques we have discussed so far won't work for very large projects

Agile Manifesto

- Uncovering better ways to do it

FDD's view on Process

- A system for building systems is necessary in order to scale to larger projects
- A simple, but well defined process works best
- Process steps should be logical and their worth immediately obvious to each team member
- Process pride can keep the real work from happening
- Good processes move to the background so team members can focus on results
- Short, iterative, feature-driven life cycles are best

FDD Origins

- Strong OO modeling at start of project
- Assumes overall model of the system at the beginning
- Adopted short iterations to address shorter business cycles
- Lightweight process specification relative to plan based methods used in the past, but heavier weight than some agile methods

Singapore Project

- Banking domain
- Failed 2 year project by a well known systems integration firm: 3500 pages of useless cases hundreds of OO classes no code
- Using FDD they successfully developed 2000 features in 15 months with 50 stagg
- Developed initial model in 1 month

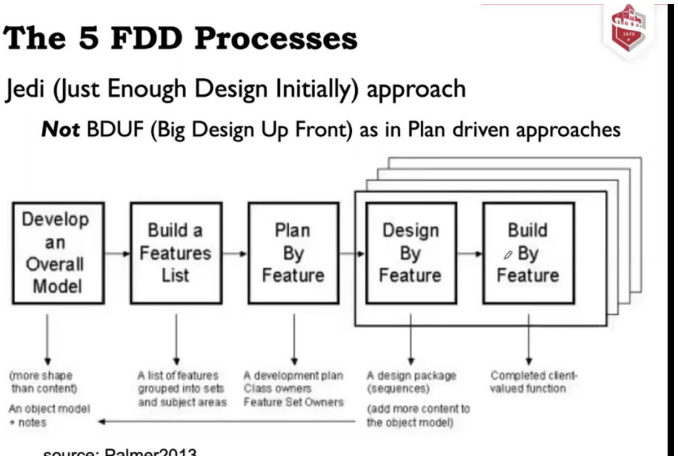
FDD Features

- Cross functional team
- Focus on collaboration
- Time boxed activity
- Start project

Roles in FDD

- Domain experts: voice of customer
- Project manager: admin lead
- Chief architect: responsible for overall design
- Development manager: team management
- CHief programmers: lead teams in design of features
- Feature teams: temp groups of dev implementing features
- Class owners: dev who own individual classes different from XP collective code ownership

5 FDD Processes



-

Develop an Overall Model

1. Form the modeling team led by chief architect
2. Perform domain walk through
3. Study relevant documents to understand domain knowledge
4. Develop the model define overall shape of the model rather than the details and more details will be added during the design by feature process
5. Refine the overall object model
6. Write model notes
7. Internal and external assessment

Build a features list

1. Form the features list chief programmers - goal is to define all of the features (product backlog)
2. Build features list - oriented to the business needs

Plan by feature

1. Form the planning team
2. Determine the development sequence - some dependencies are obvious but others are more subtle
3. Assign business activities to chief programmers
4. Assign classes to developers

Design by Feature

1. Form feature team
2. Domain walk through
3. Study the referenced documents
4. Develop the UML sequence diagrams
5. Refine the object model
6. Write class and method prologues
7. Design inspection

Build by Feature

1. Implement classes and methods
2. Code inspection
3. Unit testing
4. Promote to the build

5. Disband the team

FDD Practices

- Domain object modeling
    - construct class diagrams for all significant object types
    - Supplement UML sequence diagrams showing interactions between objects
    - Coad developed modeling in color during singapore project to standardize class patterns
- Developing by feature
    - A feature is a small client valued function expressed in the form <action><result><object>
    - Feature template provides clues to operations and classes to which they should be applied
- Individual class/code ownership
    - Each class is owned by one developer; opposite of collective ownership in XP
    - Adv: owner maintains conceptual integrity of class each class has an expert associated with it
    - Dis: dependencies between classes risk of loss of owners during development
- Feature Teams
    - Each feature is owned by a feature team
    - Each team consists of 2-5 class owners
    - Each team contains all the owners it needs to create the feature
    - Owners may be on multiple feature teams at the same time
- Inspections
    - FDD inspections usually performed by feature teams on their own features
    - Designs and code are inspected by team members
    - Adv: most effective technique for discovering defects; provide educational benefits; promotes and enforces standards
- Regular builds
    - Adv: early detection of integration problems always have something to demonstrate
    - Generate documentation; run audit, run automated testing
- COnfiguration management
    - Keep code under configuration management to manage conflicts
- Reporting/visibility of results
    - Track by feature; roll up summary
    - Opposite of burndown

Similarities to other Agile

- Short, time boxed iteration
- Lightweight process
- Requirements described as features
- Customer participation in planning

Differences

- Up front modeling
- FDD values documentation

- More emphasis on progress tracking and reporting
- Dynamic feature teams
- Design and code owned by individuals