A Tool for Supporting Feature-Driven Development

Marek Rychlý   Pavlína Tichá

Department of Information Systems
Faculty of Information Technology
Brno University of Technology
(Czech Republic)

2nd IFIP Central and East European Conference
on Software Engineering Techniques,
October 10–12, 2007

Outline

Outline

1. Feature Driven Development
   - FDD's Five Processes
   - FDD as an Agile Method
   - Software Support of the FDD
2. A Tool for Supporting Feature-Driven Development
   - Motivation
   - Design
   - Implementation

Feature Driven Development – Processes

Definition (Feature Driven Development and Features)

FDD is a model-driven short-iteration process. It begins with establishing an overall model shape. Then it continues with a series of two-week "design by feature, build by feature" iterations. The features are small "useful in the eyes of the client" results.

[Coad et al., Java Modeling in Color with UML (1999)]

Five processes:

1. Develop an Overall Model
2. Build a Features List
3. Plan By Feature
4. Design By Feature
5. Build By Feature

Feature Driven Development – Processes

Definition (Feature Driven Development and Features)

FDD is a model-driven short-iteration process. It begins with establishing an overall model shape. Then it continues with a series of two-week “design by feature, build by feature” iterations. The features are small “useful in the eyes of the client” results.

[Coad et al., Java Modeling in Color with UML (1999)]

Five processes:

1. Develop an Overall Model
   (describe key abstractions and their relationships in a system by domain experts and developers)
2. Build a Features List
3. Plan By Feature
4. Design By Feature
5. Build By Feature

Feature Driven Development – Processes

Definition (Feature Driven Development and Features)

FDD is a model-driven short-iteration process. It begins with establishing an overall model shape. Then it continues with a series of two-week "design by feature, build by feature" iterations. The features are small "useful in the eyes of the client" results.
[Coad et al., Java Modeling in Color with UML (1999)]

Five processes:

1. Develop an Overall Model
2. Build a Features List
   (obtain users' requirements and build lists of features, feature-sets and subject-areas)
3. Plan By Feature
4. Design By Feature
5. Build By Feature

Feature Driven Development – Processes

Definition (Feature Driven Development and Features)

FDD is a model-driven short-iteration process. It begins with establishing an overall model shape. Then it continues with a series of two-week "design by feature, build by feature" iterations. The features are small "useful in the eyes of the client" results.
[Coad et al., Java Modeling in Color with UML (1999)]

Five processes:

1. Develop an Overall Model
2. Build a Features List
3. Plan By Feature
   (estimate time intensities of feature-sets and features, sort according to priorities, assign the feature-sets to feature-teams and the features to feature-owners, schedule features implementation into two-weeks intervals)
4. Design By Feature
5. Build By Feature

Feature Driven Development – Processes

Definition (Feature Driven Development and Features)

FDD is a model-driven short-iteration process. It begins with establishing an overall model shape. Then it continues with a series of two-week "design by feature, build by feature" iterations. The features are small "useful in the eyes of the client" results.
[Coad et al., Java Modeling in Color with UML (1999)]

Five processes:

1. Develop an Overall Model
2. Build a Features List
3. Plan By Feature
4. Design By Feature
   (design sequence diagrams and another models and describe interfaces and declarations for classes and methods)
5. Build By Feature

Feature Driven Development – Processes

Definition (Feature Driven Development and Features)

FDD is a model-driven short-iteration process. It begins with establishing an overall model shape. Then it continues with a series of two-week "design by feature, build by feature" iterations. The features are small "useful in the eyes of the client" results.

[Coad et al., Java Modeling in Color with UML (1999)]

Five processes:

1. Develop an Overall Model
2. Build a Features List
3. Plan By Feature
4. Design By Feature
5. Build By Feature
   (implement and integrate new classes, their parts or their modifications)

Feature Driven Development as an Agile Method

FDD is one of agile software development methods, but

- built around the traditional industry-recognised practices, (planning, design and documentation phases, fine-grained decomposition of a system's functionality, accurate progress reporting, frequent verification, etc.)
- iterative and incremental software development process,
- better project management and consistency of a software's design, implementation and documentation,
- suitable for large developer teams (up to 250 members in Sprint and Motorola companies).

FDD in practice:

- 41 percent of organisations have adopted one or more agile methods and at least 19 percent of them use the FDD method,[1]
- at least 6 tools for supporting FDD, the most are commercial.

[1] A survey made in March of 2006 by Scott Ambler (IBM Rational's Methods Group), 4232 respondents (multiple responses allowed).

Software Support of the Feature Driven Development

Need to track **the five processes**:

1 describe an overall domain model with versioning,
(many features are arising during the next processes and they can affect the domain model)

2 keep track of all features and their grouping to feature-sets and subject-areas (top-down and bottom-up approaches),

3 scheduling of feature-sets and features,
(contrary requirements – keep the start-time and duration of a whole project and permit individual planing of features)

4+5 lists of activities leading to implementation of the features, track relationships between features, developers and parts of classes.
(the activity is required modification of a class or its part related to a feature)

+ provide a set of **visual reports** (for correct planing decisions).
(should continuously show progress for individual features, feature-sets, subject-areas and a whole project, as well as a proper form of workload overviews for individual developers and feature-teams)

Software Support of the Feature Driven Development

Need to track **the five processes**:

1 describe an overall domain model with versioning,
(many features are arising during the next processes and they can affect the domain model)

2 keep track of all features and their grouping to feature-sets and subject-areas (top-down and bottom-up approaches),

3 scheduling of feature-sets and features,
(contrary requirements – keep the start-time and duration of a whole project and permit individual planing of features)

4+5 lists of activities leading to implementation of the features, track relationships between features, developers and parts of classes.
(the activity is required modification of a class or its part related to a feature)

+ provide a set of **visual reports** (for correct planing decisions).
(should continuously show progress for individual features, feature-sets, subject-areas and a whole project, as well as a proper form of workload overviews for individual developers and feature-teams)

Software Support of the Feature Driven Development

Need to track **the five processes**:

1 describe an overall domain model with versioning,
(many features are arising during the next processes and they can affect the domain model)

2 keep track of all features and their grouping to feature-sets and subject-areas (top-down and bottom-up approaches),

3 scheduling of feature-sets and features,
(contrary requirements – keep the start-time and duration of a whole project and permit individual planing of features)

4+5 lists of activities leading to implementation of the features, track relationships between features, developers and parts of classes.
(the activity is required modification of a class or its part related to a feature)

+ provide a set of **visual reports** (for correct planing decisions).
(should continuously show progress for individual features, feature-sets, subject-areas and a whole project, as well as a proper form of workload overviews for individual developers and feature-teams)

Software Support of the Feature Driven Development

Need to track **the five processes**:

1 describe an overall domain model with versioning,
(many features are arising during the next processes and they can affect the domain model)

2 keep track of all features and their grouping to feature-sets and subject-areas (top-down and bottom-up approaches),

3 scheduling of feature-sets and features,
(contrary requirements – keep the start-time and duration of a whole project and permit individual planing of features)

4+5 lists of activities leading to implementation of the features, track relationships between features, developers and parts of classes.
(the activity is required modification of a class or its part related to a feature)

+ provide a set of **visual reports** (for correct planing decisions).
(should continuously show progress for individual features, feature-sets, subject-areas and a whole project, as well as a proper form of workload overviews for individual developers and feature-teams)

Software Support of the Feature Driven Development

Need to track **the five processes**:

1 describe an overall domain model with versioning,
(many features are arising during the next processes and they can affect the domain model)

2 keep track of all features and their grouping to feature-sets and subject-areas (top-down and bottom-up approaches),

3 scheduling of feature-sets and features,
(contrary requirements – keep the start-time and duration of a whole project and permit individual planing of features)

4+5 lists of activities leading to implementation of the features, track relationships between features, developers and parts of classes.
(the activity is required modification of a class or its part related to a feature)

+ provide a set of **visual reports** (for correct planing decisions).
(should continuously show progress for individual features, feature-sets, subject-areas and a whole project, as well as a proper form of workload overviews for individual developers and feature-teams)

Outline

Motivation

- open-source information system providing all team-members with instruments to follow the FDD method,
  (including multiple user roles and multiple projects and domains)
- top-down and bottom-up approaches to (de)composition of features and feature-lists,
  (the variable approaches for variable projects and requirements-obtaining strategies)
- better planning of features' implementation and their verification,
  (get responsibilities of the owners of features and classes in planning and implementation phases)
- the safety of development process in a software project.
  (traceability of user requirements into modification of classes with connection to responsibilities of active team-members)

Motivation

- open-source information system providing all team-members with instruments to follow the FDD method,
  (including multiple user roles and multiple projects and domains)
- top-down and bottom-up approaches to (de)composition of features and feature-lists,
  (the variable approaches for variable projects and requirements-obtaining strategies)
- better planning of features' implementation and their verification,
  (get responsibilities of the owners of features and classes in planning and implementation phases)
- the safety of development process in a software project.
  (traceability of user requirements into modification of classes with connection to responsibilities of active team-members)

Motivation

- open-source information system providing all team-members with instruments to follow the FDD method,
  (including multiple user roles and multiple projects and domains)
- top-down and bottom-up approaches to (de)composition of features and feature-lists,
  (the variable approaches for variable projects and requirements-obtaining strategies)
- better planning of features' implementation and their verification,
  (get responsibilities of the owners of features and classes in planning and implementation phases)
- the safety of development process in a software project.
  (traceability of user requirements into modification of classes with connection to responsibilities of active team-members)

Motivation

- open-source information system providing all team-members with instruments to follow the FDD method,
  (including multiple user roles and multiple projects and domains)
- top-down and bottom-up approaches to (de)composition of features and feature-lists,
  (the variable approaches for variable projects and requirements-obtaining strategies)
- better planning of features' implementation and their verification,
  (get responsibilities of the owners of features and classes in planning and implementation phases)
- the safety of development process in a software project.
  (traceability of user requirements into modification of classes with connection to responsibilities of active team-members)

Design of the Tool – Features and Activities

Decomposition of features:

- implementation of a feature can involve multiple classes or their parts,
- the feature can be decomposed to activities leading to implementation of the feature,
- feature owner decomposes the feature to a list of activities and establish a team together with class owners maintaining modified classes.

Planning of activities:

- features and related activities are scheduled by members of a feature-team (cooperation of feature owners and class owners),
- class owners implement individual activities, i.e. parts of the feature,
- feature owners check progress of implementation of features and verifies their finished activities.

Design of the Tool – Features and Activities

Decomposition of features:

- implementation of a feature can involve multiple classes or their parts,
- the feature can be decomposed to activities leading to implementation of the feature,
- feature owner decomposes the feature to a list of activities and establish a team together with class owners maintaining modified classes.

Planning of activities:

- features and related activities are scheduled by members of a feature-team (cooperation of feature owners and class owners),
- class owners implement individual activities, i.e. parts of the feature,
- feature owners check progress of implementation of features and verifies their finished activities.

Design of the Tool – User Roles

Supported user roles:

System Administrator and Domain Administrator	– maintains technical issues of the whole information system and of its domain specific parts,
Project Administrator	– manages a project, assigns and withdraws users and their roles, and obtains reports relevant to the project and its parts,
Chief Architect	– manages a project's domain model and features, feature-sets and subject-areas, and makes planing decisions,
Feature Owner	– maintains a feature-team, i.e. controls activities leading to implementation of the feature, assigns classes that are modified by the activities and class-owners responsible for realisation of individual activities, watches progress and verifies finished activities,
Class Owner	– represents a developer, who implements a part of assigned feature (an individual activity connected to the owned class) and is able to modify information about the progress of the activity
Guest	– represents external supervisor of a project (a customer's representative), able to view progress reports.

Design of the Tool – Planning of Workload

Problems of the FDD:

- features and activities can be implemented concurrently,
- a class owner can be an active member of many feature-teams,
- some of classes can be modified more often than others.

Proposed solutions:

- provide feature-teams with tools to create better timetables for their features,
  (according to an actual and scheduled workload, a history of related classes and its owners, etc.)
- provide a wide range of visual reports for high management,
  (for long-term planning, fine-tuning of human resources, etc.)
- support modification of a system's design and refactoring the code (classes).
  (to avoid "overloaded" classes and their owners, etc.)

Design of the Tool – Planning of Workload

Problems of the FDD:

- features and activities can be implemented concurrently,
- a class owner can be an active member of many feature-teams,
- some of classes can be modified more often than others.

Proposed solutions:

- provide feature-teams with tools to create better timetables for their features,
  (according to an actual and scheduled workload, a history of related classes and its owners, etc.)
- provide a wide range of visual reports for high management,
  (for long-term planning, fine-tuning of human resources, etc.)
- support modification of a system's design and refactoring the code (classes).
  (to avoid "overloaded" classes and their owners, etc.)

Implementation of the Tool – Technologies

- the framework ASP.NET 2.0, coded in the C# language,
- a web-based application for the Microsoft Internet Information Services web-server,
- database server Microsoft SQL Server via ADO.NET and own data abstraction layer.

Summary and Future Work

- open-source information system providing all team-members with instruments to follow the FDD method,
- top-down and bottom-up approaches to feature-lists (de)composition,
- traceability of user requirements into modification of classes with connection to responsibilities of active team-members.

Current and future work

- versioning of the features, feature-lists and activities,
- integration of human-resource management into the tool,
- source code management in connection to the activities.

Thank you for your attention!