Seminar in Software Engineering

Agile vs. klassische Methoden der Software-Entwicklung

Feature-Driven Development

Wintersemester 2003/04

Daniel Gyger

Betreuer: Christian Seybold

Institut für Informatik
der Universität Zürich
Prof. Dr. M. Glinz

1. Einleitung

In der Softwareentwicklung ist es heute nicht selten, dass die Entwicklung einer Software zwei Jahre oder länger dauert. In einem schnelllebenden Markt mit immer kürzer werdenden Produktlebenszyklen wird die Entwicklung von Software mit langen Realisierungsphasen zunehmend schwieriger. Es kann deshalb vorkommen, dass ein vor zwei Jahren geplantes und begonnenes Softwareprojekt beim Abschluss nicht mehr den Marktbedürfnisse entspricht und deshalb als gescheitert betrachtet werden muss. Eine lange Projektdauer ist demnach mit einem hohen Risiko verbunden.

Feature-Driven Development (FDD) reagiert auf diese Entwicklung mit kurzen Entwicklungsschritten und der laufenden Fertigstellung von Teilprogrammen. Etabliert wurde diese Entwicklungsmethode von Jeff De Luca und Peter Coad. In einem grösseren Software-Projekt mit Java in Singapur wurde FDD Mitte der Neunziger Jahre entwickelt und erstmals eingesetzt [2]. FDD ist für den Einsatz mit einer objektorientierten Programmiersprache und mit UML als Modellierungssprache gedacht.

2. Lösungsansatz von FDD

Ein Feature wird von den Erfindern von FDD in [1] wie folgt definiert:

```
The features are small "useful in the eyes of the client"
results.
```

FDD ist eine kundenorientierte Entwicklungsmethode mit kleinen Teilresultaten. Die Grösse eines Feature wird weiter eingeschränkt durch die Bedingung, dass ein Feature in maximal zwei Wochen entwickelt werden sollte, andernfalls muss es weiter aufgeteilt werden bis diese Vorgabe erfüllt werden kann.

Der Vorteil einer solchen Vorgabe wird in der Motivation der Programmier und in der Messbarkeit der Resultate gesehen. Durch die häufige Erreichung eines brauchbaren und sichtbaren Resultats, d.h. alle zwei Wochen, kann die Motivation der Programmierer Aufrecht erhalten werden. Der Abschluss eines Feature kann zudem von den Projektverantwortlichen und den Manager zum Feststellen des Projektfortschritts verwendet werden. Da ein Feature als kundenorientiertes Resultat definiert ist, ist die Fertigstellung eines Feature auch ein für Nichtinformatiker verständlicher Meilenstein: ein Manager oder der Kunde kann sich unter dem Resultat etwas vorstellen. Der Aufwand für kleine Teilresultate kann besser geschätzt werden und deshalb erlaubt FDD eine bessere Projektkontrolle.

3. Namenskonventionen

Ein Feature wird mit folgendem Template beschrieben:

<**action**>the<**result**><by|for|of|to>a(n)<**object**>
Zum Beispiel:
- **Calculate** the **total** of a **sale**
- **Asses** the **fulfillment timelines** of a **sale**
- **Calculate** the **total purchases** by a **customer**

Features können gruppiert und in einer Menge (Feature Set) zusammengefasst werden. Dabei kommt folgendes Template zur Anwendung:

<**action**><-ing>a(n)<**object**>

Beispiel:
- **mak**ing a **product sale**

Die nächste Abstraktionsstufe wird von den Autoren als "Major Feature Set" bezeichnet und mit folgender Namenskonvention beschrieben:

<**object**>management

Beispiel:
- **product-sales** management

4. Die Prozesse

4.1. Einführung

Die Prozesse für das Feature-Driven Development werden bewusst kurz und simpel gehalten, weil die Autoren die Meinung vertreten, dass eine Überspezifikation mehr schadet als nützt. Einen bis ins kleinste Detail definierten Prozess kann die schlussendlich benötigte Entwicklung eines Programms nicht ersetzen. Nichtsdestotrotz wird auf Prozesse im FDD aus folgenden Gründen nicht verzichtet:

- Transformation auf andere Projekte und Wiederholbarkeit des Erfolgs
- Schnellere Einarbeitungszeit für neue Mitarbeiter
- Setzen von Prioritäten

Beim ersten Punkt geht es darum, ein bestimmtes Verhalten bei den involvierten Personen zu schulen, welches auch in späteren und neuen Projekten zum Erfolg führt. Ein simpler und klar definierter Prozess soll zu einer guten Gewohnheit werden.

Zweitens kann mit einem simplen und klar definierten Prozess die Einarbeitungszeit für neue Mitarbeiter verkürzt werden.

Drittens sollte sich die Entwicklung auf die wichtigen Punkte konzentrieren und Prioritäten setzen. Die Anforderungen sollen gewichtet sein nach "must-have", "nice-to-have" etc.

4.2. Art der Prozessbeschreibung

Der FDD-Prozess wird nach dem ETVX-Template beschrieben. ETVX steht für: Entry, Task, Verfication, Exit und umfasst folgende Schritte:

1. Entry: Vorbedingungen des Prozesses, d.h. Bedingungen welche erfüllt sein müssen, bevor mit dem Prozess begonnen werden kann.
2. Tasks: Liste aller Aufgaben, die während des Prozesses ausgeführt werden.
3. Verification: Die Verfikation beschreibt die Mittel, wie die Erfüllung des Prozesses ermittelt werden kann.
4. Exit: Schlussbedingungen und Output des Prozesses.

4.3. Die fünf Prozesse des FDD

In diesem Abschnitt werden die fünf Prozesse des Feature-Driven Development näher vorgestellt.

1. Entwickeln eines Gesamtmodells (develop an overall model)
2. Erstellen einer Feature-Liste (build a feature list)
3. Planung pro Feature (plan by feature)
4. Entwurf pro Feature (design by feature - DBF)
5. Implementierung pro Feature (build by feature - BBF)

Die beiden letzen Prozesse werden pro Feature wiederholt. Abbildung 1 veranschaulicht den Aufbau der Prozesse.

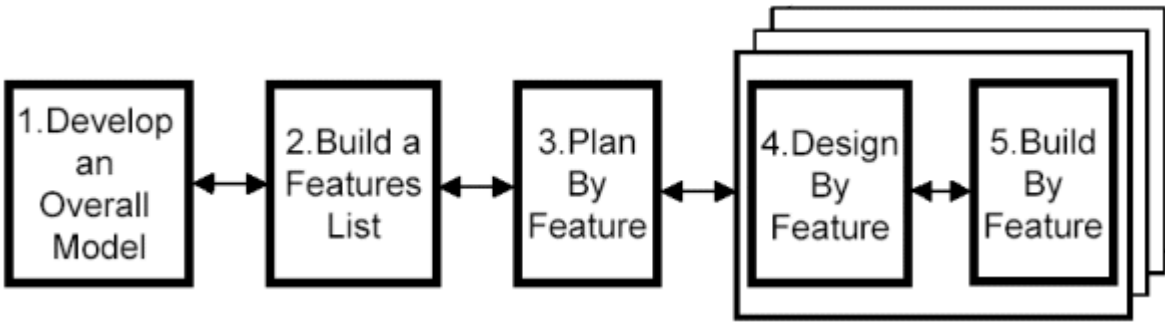

Abbildung 1 - Die fünf Prozesse von FDD

4.3.1. Entwickeln eines Gesamtmodells (develop an overall model)

Die Einstiegsbedingung (vgl. Abschnitt 4.2) dieses Prozesses ist, dass der Kunde die Weichen zum Erstellen der Applikation gestellt hat. Der Kunde sollte eine grobe Vorstellung haben, was das System leisten soll, aber er benötigt keine detaillierte Spezifikation.

Die Aufgaben des ersten Prozesses umfassen folgende Punkte:

- **Bildung des Modellierungteams**
  Das Modellierungsteam besteht aus permanenten Mitgliedern aus dem Fach- und dem Entwicklungsbereich. Diese Personen sind Experten und werden während der Arbeit durch rotierendes Personal ergänzt.
- **Übersicht über die Fachmaterie gewinnen**
  Die Fachexperten stellen ihre Gebiete dem Modellierungsteam vor.
- **Dokumente studieren**
  Das Team studiert vorhandene Dokumentationen zum Fachbereich, falls solche Dokumente überhaupt vorhanden sind. Diese Aufgabe ist deshalb optional.
- **Entwurf einer ersten Feature-Liste**
  Die Chefprogrammier und der Architekturchef erstellen eine erste Liste mit Features. Auf die Rolle des Chefprogrammiers wird im nächsten Kapitel eingegangen.
- **Erstellen von Teilmodelle**
  In Gruppen, d.h. bis zu drei Personen, modellieren jeweils einen Teilbereich des Gesamten. Es kommt eine objektorientierte Modellierung zum Einsatz.
- **Erstellen eines Geamtmodells**
  Jede Kleingruppe präsentiert ihr Teilmodell. Änderungsvorschläge können vom Architekturchef und in der Diskussion angebracht werden. Die einzelnen Teilmodelle werden zu einem einheitlichen Gesamtmodell (Klassendiagramm) zusammengefügt.
- **Notieren von Alternativvorschlägen**
  Nicht berücksichtigte Alternativvorschläge werden archiviert, so dass notfalls auf diese zurückgegriffen werden kann.

Die Verifikation des Prozesses geschieht durch die beteiligten Fachexperten, welche darüber entscheiden, wie weit das Gesamtmodell dem gewünschten System entspricht. Ein externes Review kann bei Bedarf durchgeführt werden.

Die Ergebnisse des ersten Prozesses sind die Klassendiagramme, eine informale Feature-Liste sowie Notizen zu den Alternativvorschlägen.

4.3.2. Erstellen einer Feature-Liste (build a feature list)

Die Einstiegsbedingung für den 2. Prozess von FDD ist die korrekte Beendigung des ersten Prozesses.

Die Aufgaben umfassen:

- **Bildung des Teams zur Erstellung der Feature-Liste**
  Das Team wird wiederum durch Fachexperten und Leute aus dem Entwicklungsbereich gebildet.
- **Erstellen der Feature-Liste**
  Ausgehend von den Resultaten des 1. Prozesses werden nun die Features beschrieben. Dazu werden die Methoden der Klassendiagramme in Features transformiert. Bei der Beschreibung der Features kommen die im Kapitel 3 diskutierten Namenskoventionen zum Einsatz. Die Features werden zudem in "Feature Sets" und "Major Feature Sets" gegliedert.

- **Setzen der Prioritäten**
  Ein Subteam setzt für alle Features die Prioritäten nach folgendem Schema fest:
  - must have
  - nice to have
  - add it if we can
  - future

  Bei dieser Beurteilung steht die Kundenzufriedenheit im Vordergrund. Ausschlaggebend ist also, ob ein Feature die Kundenzufriedenheit erhöht oder erniedrigt.
- **Aufteilen komplexer Features**
  In einem weiteren Schritt wird jedes Feature dahin beurteilt, ob dieses gemäss der Definition eines Feature (vgl. Kapitel 2) innerhalb zwei Wochen implementiert werden kann oder nicht. Ist dies nicht der Fall, wird das entsprechende Feature aufgeteilt.

Die Verifikation des zweiten Schrittes erfolgt wiederum durch die in den Prozess involvierten Fachexperten.

Das Resultat des Prozesses ist eine vollständige, gegliederte und geprüfte Feature-Liste.

4.3.3. Planung pro Feature (plan by feature)

Vor dem Beginn dieses Prozesses muss der vorangehende Prozess korrekt beendet worden sein.

Die Aufgaben des dritten Prozesses sind:
- **Bildung des Planungsteams**
  Das Planungsteam besteht aus dem Projektmanager, dem Entwicklungschef und den Chefprogrammierern.
- **Bestimmung der Reihenfolge**
  Das Planungsteam legt die Reihenfolge sowie das Enddatum fest wie die "Major Feature sets" implementiert werden sollen.
- **Zuweisung der Klassenverantwortlichkeiten**
  Jede Klasse (basierend auf dem Klassendiagramm) wird einem Entwickler zugewiesen, welcher die Verantwortung dieser Klasse übernimmt.
- **Zuweisung der Verantworlichkeiten für die Features**
  Die Verantwortung für Features und "Feature Sets" werden den Chefprogrammierern zugewiesen.

Die Verifikation erfolgt durch die Mitglieder des Planungsteams sowie bei Bedarf durch das höhere Kader.

Die Ergebnisse des dritten Prozesses sind ein Projektplan mit Enddatum und zugewiesene Verantwortlichkeiten für jedes Feature und jede Klasse.

4.3.4. Entwurf pro Feature (design by feature - DBF)

Einstiegsbedingung ist die Beendigung des dritten Prozesses. Der vierte und fünfte Prozess werden mehrmals, d.h. für jedes Feature einmal ausgeführt.

Während dem vierten Prozess werden folgende Aufgaben ausgeführt:

- **Bildung des Teams**
  Der Chefprogrammierer bestimmt die Klassen, welche voraussichtlich für die Implementierung des Features betroffen sind und kontaktiert die betreffenden Klassenverantwortlichen. Zusammen bilden diese ein Team, welches innerhalb von zwei Wochen dieses Feature implementieren soll.
- **Übersicht über Fachgebiet**
  Bei einem komplexen Feature ist es nützlich, wenn ein Fachexperte die betreffende Domäne zuerst vorstellt. Dieser Schritt ist optional.
- **Dokumente studieren**
  Das Team studiert vorhandene Dokumentationen zum Fachbereich. Diese Aufgabe ist wiederum optional.
- **Erstellen eines Ablaufdiagramms**
  Für die Implementation dieses Features wird ein detailliertes Ablaufdiagramm erstellt. Entwurfsalternativen werden festgehalten und die Projektdokumentation mit diesem Entwurf ergänzt.
- **Schnittstellen festlegen**
  Die Klassenverantwortlichen aktualisieren ihre Klassen mit den Methoden, welche sie gemäss dem Entwurf implementieren müssen. Parameter und Rückgabewerte von Methoden werden festgelegt.
- **Entwurfsinspektion**
  Das Team führt eine Entwurfsinspektion für das betreffende Feature durch. Bei Bedarf und bei hoher Komplexität des Features können für die Inspektion teamexterne Personen hinzugezogen werden.
- **Befunde festhalten**
  Befunde der Inspektion werden festgehalten und gegebenenfalls Aufgaben an die betreffenden Klassenverantwortlichen gestellt.

Die Verifikation dieses Prozesses erfolgt durch das Team anhand des erstellten Ablaufdiagramms.

Das Ablaufdiagramm und die Aktualisierungen der geänderten Klassen bilden das Resultat des vierten Prozesses.

4.3.5. Implementierung pro Feature (build by feature - BBF)

Die Startbedingung dieses Prozesses ist die Vollendung des dazugehörigen DBF-Prozesses.

Die Aufgaben des Prozesses umfassen:

- **Implementierung**
  Die Klassenverantwortlichen implementieren die im vorangehenden Prozess deklarierten Methoden. Zusätzlich müssen auch Testmethoden hinzugefügt werden.

- **Codeinspektion**
  Das Team führt unter der Leitung des Chefprogrammierers eine Codeinspektion für die implementierten Methoden durch.
- **Befunde festhalten**
  Befunde der Inspektion werden festgehalten und den Klassenverantwortlichen Aufgaben zur Behebung aufgetragen.
- **Unit Test**
  Die Klassenverantwortlichen führen einen Unit Test durch und zeigen so, dass die restlichen Methoden der Klasse nach der Implementierung dieses Features weiterhin korrekt funktionieren. Der Chefprogrammierer macht einen Gesamttest des implementierten Features.
- **Freigabe des Codes**
  Nachdem eine Klasse erfolgreich implementiert, inspiziert und getestet wurde, wird sie im Konfigurationsmanagementsystem freigeschaltet. Der Chefprogrammierer aktualisiert den Status des Features.

Die erfolgreiche Codeinspektion aller beteiligten Klassen bildet die Verifikation dieses Prozesses.

Die Resultate des fünften Prozesses umfassen die Implementierungen des Features, die Testresultate und die neue Version im Konfigurationsmanagementsystem.

5. Rollen im FDD

Wie im vorangehenden Kapitel bereits mehrfach erwähnt, gibt es im FDD verschiedene Rollen wie z.B. Chefprogrammierer, welche eine Person in einem Team wahrnehmen kann. Diese Rollen werden nun diesem Kapitel genauer beschrieben.

5.1. Chefprogrammierer (chief programmer)

Der Chefprogrammierer leitet ein Feature-Team (vgl. 5.3) während zwei Wochen bei der Implementierung eines Features. Für die Position des Chefprogrammiers kommen überdurchschnittliche produktive Programmierer einer Unternehmung in Frage. Er trägt die Verantwortung für die Implemenation und das Testen eines Features.

5.2. Klassenverantwortlichen (class owner)

Für jede Klasse der Applikation gibt es einen Klassenverantwortlichen, welcher die Klasse entwirft und implementiert. Diese Organisation führt zu einer konsistenten Implementation einer Klasse und erhöt die Motivation der Mitarbeiter.

5.3. Feature Team

Ein Feature Team besteht höchstens zwei Wochen, also die Zeit, in welcher jedes Feature entwickelt werden sollte. Der Chefprogrammierer stellt das Team je nach Klasse zusammen, welche nach seiner Einschätzung zur Implementierung des Features notwendig sind. Die Klassenverantwortlichen dieser Klassen bilden dann zusammen für die Implementationszeit des Features das Team dazu. Die Zusammensetzung der Teams wechselt also für jeden Iterationschritt von Prozess 4 und 5 (DFB/BBF) . Ein Klassenverantwortlichen kann gleichzeitig auch in mehreren Feature Teams Mitglied sein. Abbildung 2 veranschaulicht diesen Teambildungsprozess.

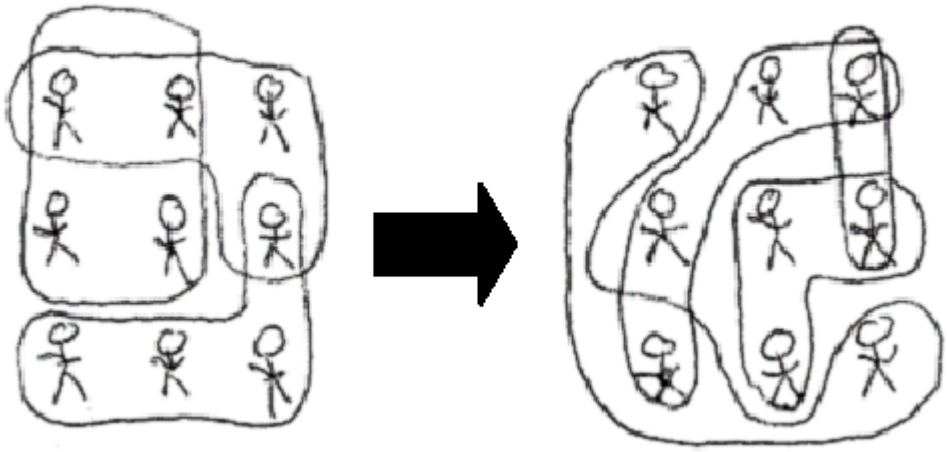

Abbildung 2 - Klassenverantwortliche in verschiedenen Teams

6. Fortschrittskontrolle

Wie im Kapitel 2 angesprochen wurde, sollte beim Einsatz von FDD eine genaue Fortschrittskontrolle möglich sein. Diese Kontrolle basiert auf Erfahrungswerten was die Aufteilung des Arbeitsaufwandes auf die verschiedenen Prozesse anbelangt und auf die Zählung der bereits implementierten Features zur Fortschrittskontrolle in der Iteration der Prozesse 4 und 5. Die Zeitaufteilung auf die fünf Prozesse wird von den Autoren der Methode [1] wie folgt angegeben:

	Initial	**Laufend**
1, Entwickeln eines Gesamtmodells	10%	4%
2. Erstellen einer Feature-Liste	4%	1%
3. Planung pro Feature	4%	2%
4. Entwurf pro Feature	77 %	
5. Implementierung pro Feature		

Tabelle 1 - Verteilung des Aufwands

Bei der Angabe des Aufwands wird unterschieden zwischen dem Initialaufwand und dem Aufwand, welcher für spätere Änderungen hinzukommt. Die meiste Zeit wird für

das Design und die Implementation der einzelnen Features eingeplant. Innerhalb des Zyklus sieht die Fortschrittskontrolle wie folgt aus:

Übersicht über Fachgebiet	1%
Entwurf	40%
Entwurfsinspektion	3%
Implementierung und Testen	45%
Codeinspektion	10%
Freigabe des Codes	1%

Tabelle 2 - Verteilung des Arbeitsaufwand innerhalb einer DBF/BBF-Iteration

Mit Hilfe dieser in Tabelle 2 aufgelisteten Meilensteine und den kleinen Iterationsschritten, d.h. zwei Wochen, ist es möglich, eine überdurchschnittlich gute Fortschrittskontrolle durchzuführen.

Für die Berichterstattung ist ein wöchentliches Treffen zwischen den Chefprogrammierern und der Projektleitung vorgesehen. Der Status jedes Feature wird dabei auf einer Übersichtskarte aktualisiert. Dem höheren Management und dem Kunden wird jeweils der Status der "Major feature sets" mitgeteilt.

7. Fazit

Feature Driven Development scheint auf den ersten Blick eher eine wenig agile Methode zu sein, da Planung und Prozesse bewährte Eigenschaften von klassischen Methoden sind. Gemäss Jeff De Luca, dem Erfinder von FDD, ist dies kein Wiederspruch, weil beispielsweise das vierte Statement des manifesto for Agile Software Development [3] nicht sagt, dass man nicht planen soll, sondern dass man offen für Änderungen sein soll und gegebenenfalls sein Plan anpassen muss. FDD kann als Gegenbewegung zu ausgeprägten Prozesskulturen verstanden werden, in welchen viel geplant und wenig produziert wird. Als Kritikpunkt kann die vielleicht all zu starke Kundenfokusierung gesehen werden. Für informatikspezifische Probleme wie z.B. technische Verbesserungen, welche nur indirekt einen Kundennutzen bringen, kann es schwierig sein, diese kurz und bündig als Feature zu beschreiben. Für grosse Integrationsprojekte mit langen Planungs- und Abklärungsphasen ist diese Methode deshalb nicht zu empfehlen.

Abbildungs- und Tabellenverzeichnis

Literaturverzeichnis

[1] Coad P., Lefebvre E., De Luca J.: Java Modeling in Color with UML, Prentice Hall 1999
[2] Nebulon Pty. Ltd, http://www.nebulon.com
[3] Manifesto for Agile Software Development, http://www.agilemanifesto.org
[4] About Peter Coad, http://www.pcoad.com