mehr zum thema:
www.featuredrivendevelopment.com/
de.wikipedia.org/wiki/Feature_Driven_Development

FEATURE DRIVEN DEVELOPMENT

„Feature Driven Development" (FDD) ist eine leichtgewichtige agile Entwicklungsmethode, die sich seit über zehn Jahren in kleinen und großen Softwareprojekten bewährt hat. Mit seiner vorgelagerten Modellierungsaktivität beantwortet FDD auch Fragen nach Gesamtaufwänden und bietet so eine belastbare Basis für Festpreisprojekte. Trotzdem ist FDD in Deutschland relativ unbekannt. Der Artikel gibt einen Überblick über das Thema.

der autor

Stefan Roock
(E-Mail: stefan.roock@akquinet.de) ist Senior IT-Berater bei der akquinet AG in Hamburg. Er verfügt über mehrjährige Erfahrung aus agilen Softwareprojekten als Projektleiter, Entwickler, Kundenberater, Entwicklerberater und XP-Coach.

Feature Driven Development (FDD) wurde von Jeff De Luca im Jahre 1997 als schlanke Methode definiert, um ein großes zeitkritisches Projekt (15 Monate, 50 Entwickler) durchzuführen. Seitdem wurde FDD kontinuierlich weiterentwickelt. FDD stellt den Feature-Begriff in den Mittelpunkt der Entwicklung. Jedes Feature stellt einen Mehrwert für den Kunden dar.

Die Entwicklung wird anhand eines Feature-Plans organisiert. Eine wichtige Rolle spielt der Chef-Modellierer, der ständig den Überblick über die Gesamtarchitektur und die fachlichen Kernmodelle behält. Bei größeren Teams werden einzelne Entwicklerteams von Chef-Programmierern geführt. FDD ist ganz im Sinne der agilen Methoden sehr kompakt. Die FDD-Definition passt auf nur zehn Seiten (vgl. [Luc]).

FDD-Prozesse

FDD-Projekte durchlaufen fünf Prozesse (siehe Abb. 1 sowie [Luc] und [Pal02]).

Die Abfolge der Prozesse sieht auf den ersten Blick wasserfallartiger aus, als sie es in der Praxis ist. Zum einen werden für die ersten drei Prozesse nur kurze Zeiträume investiert (häufig nur wenige Tage). Zum anderen werden die Prozesse #4 und #5 in ständigem Wechsel durchgeführt, weil jedes Feature in maximal zwei Wochen realisiert wird.

Prozess #1: Entwickle Gesamtmodell

Im ersten Prozess definieren Fachexperten und Entwickler unter Leitung des Chef-Modellierers Inhalt und Umfang des zu entwickelnden Systems. In Kleingruppen werden Fachmodelle für die einzelnen Bereiche des Systems erstellt, die in der Gruppe vorgestellt, gegebenenfalls überarbeitet und schließlich integriert werden. Das Ziel dieses ersten Prozesses ist ein gemeinsames fachliches Verständnis über den zu unterstützenden Anwendungsbereich. Es wird sowohl analysiert als auch auf grober Ebene modelliert (das fachliche Kernmodell).

Meistens verwendet man für die Modellierung *Modeling in Color* (vgl. [Coa99]), einen Modellierungsansatz, der durch wiederkehrende Archetypen und standardisierte Beziehungen zwischen fachlichen Klassen sehr schnell zu einheitlichen Modellen führt.

Prozess #2: Erstelle eine Feature-Liste

Im zweiten Prozess detaillieren die Chef-Programmierer die im ersten Prozess festgelegten Systembereiche in Features. Dazu wird ein dreistufiges Schema verwendet: Fachgebiete (*Subject Areas*) bestehen aus Geschäftstätigkeiten (*Business Activities*), die in Schritten (*Steps*) ausgeführt werden. Die Schritte werden durch Features abgebildet.

Die Features werden sehr prägnant nach dem einfachen Schema *Aktion/Ergebnis/Objekt* aufgeschrieben, z. B. „Berechne Gesamtsumme der Verkäufe". Ein Feature darf maximal zwei Wochen zu seiner Realisierung benötigen.

Das Ergebnis dieses zweiten Prozesses ist eine Feature-Liste, kategorisiert nach Geschäftstätigkeiten und Fachgebieten. Die Feature-Liste muss für die Fachexperten verständlich sein.

Prozess #3: Plane je Feature

Im dritten Prozess planen der Projektleiter, der Entwicklungsleiter und die Chef-Programmierer die Reihenfolge, in der Features realisiert werden sollen. Dabei richten sie sich nach den Abhängigkeiten zwischen den Features, der Auslastung der Programmierteams sowie der Komplexität der Features.

Auf Basis des Plans werden die Fertigstellungstermine je Geschäftsaktivität festgelegt. Jede Geschäftsaktivität wird von einem Chef-Programmierer verantwortet. Außerdem werden für die bekannten Kernklassen Entwickler als Besitzer festgelegt (*Class Owner List*).

Prozess #4: Entwirf je Feature

Im vierten Prozess weisen die Chef-Programmierer den Feature-Teams die anstehenden Features zu. Ausschlaggebend für die Zuordnung ist der Klassenbesitz. Die Feature-Teams erstellen ein oder mehrere Sequenzdiagramme für die Features und die Chef-Programmierer verfeinern die Klassenmodelle auf Basis der Sequenzdiagramme. Die Entwickler schreiben dann erste Klassen- und Methodenrümpfe. Schließlich werden die erstellten Ergebnisse inspiziert (*Design Inspection*). Bei fachlichen Unklarheiten werden die Fachexperten hinzugezogen.

Prozess #5: Konstruiere je Feature

Im fünften Prozess programmieren die Entwickler die im vierten Prozess vorbereiteten Features. Bei der Programmierung werden Komponententests und Code-Inspektionen zur Qualitätssicherung eingesetzt.

Wann FDD?

FDD definiert ein Prozess- und ein Rollenmodell, die gut mit existierenden

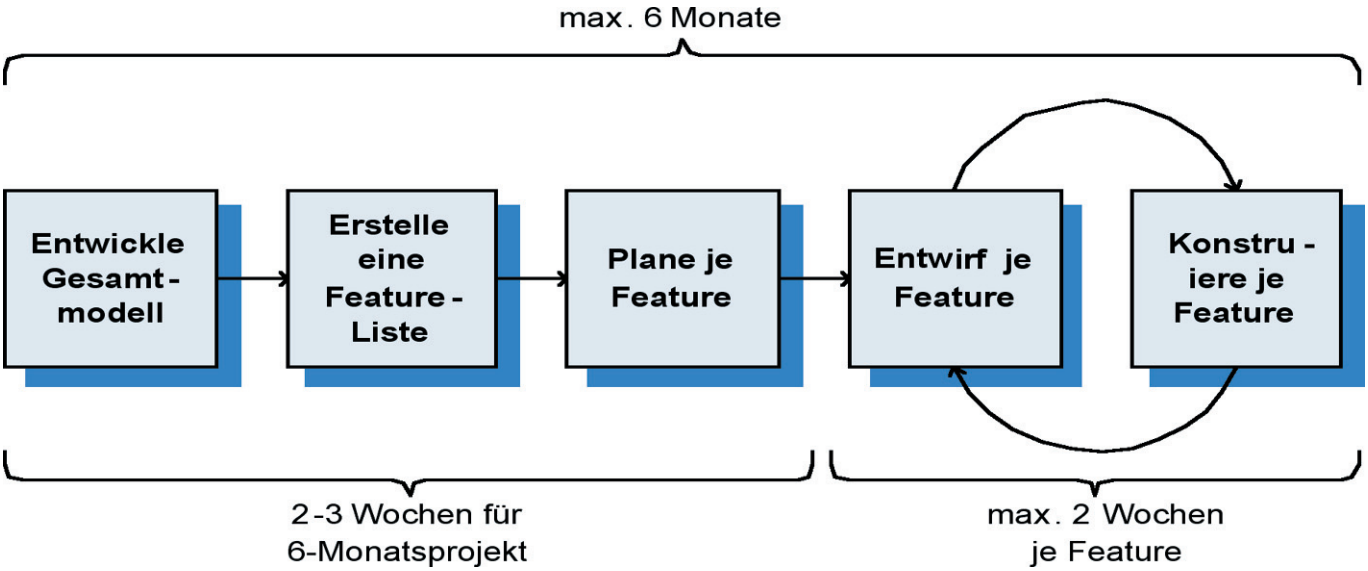

Abb. 1: FDD-Prozesse

Strukturen in größeren Unternehmen harmonieren. Daher fällt es vielen Unternehmen leichter, FDD zu verwenden als andere agile Methoden.

Die Voraussetzung für FDD ist ein klarer Projektumfang beim Projektstart und ein moderates Tempo für Anforderungsänderungen. Ist das Projektteam groß und heterogen oder arbeiten die Entwickler nur einen begrenzten Zeitraum zusammen, bietet FDD eine gesunde Basis für ein diszipliniertes und zielgerichtetes Vorgehen.

Ein auffälliger Unterschied zwischen FDD und XP ist die *Code-Ownership*: Während FDD klassische *Class-Ownership* verwendet (d. h. Klassen sind Entwicklern fest zugeordnet), setzt XP auf *Collective-Ownership* (d. h. jeder im Team kann jede Klasse ändern). FDD meidet die mit *Collective-Ownership* verbundenen Gefahren (*No-Ownership*, Strukturverlust im Code) und muss dafür mehr in die Planung investieren, insbesondere die Zuordnung von Features zu Entwicklern.

Bewertung

FDD wirkt in seiner Beschreibung wasserfallartig: einzelne Prozesse folgen sequenziell hintereinander und ein freier Wechsel zwischen den Prozessen ist nicht vorgesehen. Allerdings sind die einzelnen Prozesse sehr kurz, sodass FDD dennoch Reaktionen auf veränderte Randbedingungen und Anforderungen erlaubt. Allerdings kann FDD nicht ganz so schnell reagieren wie z.B. XP oder Scrum. Der höhere Planungsaufwand macht FDD etwas schwerfälliger, allerdings auch weniger anfällig für bestimmte Störungen im Entwicklungsprozess (z. B. „Kunde entscheidet sich ständig um"). ■

Literatur & Links

[Coa99] P. Coad, E. Lefebvre, J. De Luca, Java Modeling in Color with UML, Prentice Hall, 1999

[it-a] it-agile, Schulung und Coaching zu FDD in Deutschland, siehe: fdd.it-agile.de

[Luc] J. de Lucca, FDD-Beschreibung auf 10 Seiten, siehe: www.nebulon.com/articles/fdd/download/fddprocessesA4.pdf

[Neb-a] Nebulon Pty Ltd., Community-Website zu FDD, siehe: www.featuredrivendevelopment.com/

[Neb-b] Nebulon Pty Ltd, Homepage (Web-Site des FDD-Vaters Jeff De Luca), siehe: www.nebulon.com

[Pal02] S.R. Palmer, J.M. Felsing, A Practical Guide to the Feature Driven Development, Prentice Hall, 2002