Electronic version of an article published in **Fahney, R.; Herrmann, A.; Weißbach, R. (Hrsg): Anforderungsbasiertes Projektmanagement. Beiträge zum Workshop Fulda 14./15.06.2007. Erschienen in der Reihe "Prozessgestaltung"**

Feature Driven Development zwischen Wasserfall und Agilität

Andrea Herrmann

Lehrstuhl Software Engineering, Institut für Informatik
Universität Heidelberg
Im Neuenheimer Feld 326, 69120 Heidelberg
herrmann@informatik.uni-heidelberg.de

Bei der Projektplanung baut ein Projektmanager mehr oder weniger bewusst auf ein Vorgehensmodell auf. Das Wasserfallmodell erlaubt eine einfache Planung, funktioniert jedoch leider in der Praxis nicht gut. Agile Methoden erlauben hohe Flexibilität, verlangen aber einen Paradigmenwechsel auch beim Kunden. Feature Driven Development (FDD) stellt einen Kompromiss zwischen Wasserfallmodell und Agilität dar. In FDD sind die als Features gruppierten Anforderungen auch die Grundlage für die Projektplanung und –kontrolle. Es wird eine Fallstudie beschrieben, in der die Randbedingungen den Einsatz von FDD nahe legten.

1 Einleitung

Beim Planen eines Projektes baut ein Projektmanager mehr oder weniger bewusst auf ein Vorgehensmodell auf. Das Wasserfallmodell erlaubt eine einfache Planung, da das Projekt nur einige wenige Meilensteine und Arbeitsergebnisse vorsieht. Leider funktioniert es in der Praxis nicht gut (Abschnitt 2). Agile Methoden erlauben eine hohe Flexibilität, verlangen jedoch einen Paradigmenwechsel auch beim Kunden (Abschnitt 3). Feature Driven Development (FDD) stellt einen Kompromiss zwischen Wasserfallmodell und Agilität dar (Abschnitt 4). Hier wird eine Fallstudie beschrieben, insbesondere deren Randbedingungen, die den Einsatz von FDD nahe legten, sowie die Risiken, die durch dieses Vorgehen abgefangen wurden (Abschnitt 5). In Abschnitt 6 wird zusammenfassend der Nutzen von FDD dargestellt.

2 Das Wasserfallmodell

Das Wasserfallmodell [Ro70], das älteste heute noch zitierte Vorgehensmodell, unterscheidet klar voneinander getrennte Projektphasen (Anforderungen, Entwurf, Realisierung, Test, Wartung), die hier sequentiell ausgeführt werden. Beim Übergang von einer Phase in die folgende muss jeweils am Meilenstein das Arbeitsergebnis der vorherigen Phase fertig gestellt sein. Im Wasserfallmodell stellen die Anforderungen nur ein Projektdokument unter vielen dar. Dieses Modell bietet eine hohe Planungssicherheit und erlaubt es den Kunden, sich nach der Anforderungs-Phase zurück zu ziehen.

In der Praxis jedoch funktioniert kaum ein Projekt auf diese Art. In jeder Phase werden Unvollständigkeiten in der vorigen Phase entdeckt. Durch neue Erkenntnisse entstehen neue Fragen und Ideen. Diese Schwierigkeiten führten in der Fachliteratur zur Definition zweier Uncertainty Principles: „uncertainty is inherent and inevitable in software development processes and products“ [ZR96] und: “for a new software system, the requirements will not be completely known until after the users have used it” [Hu95].

2 Agile Methoden und Extreme Programming

Die sich in der Praxis ergebenden Iterationen werden durch die agilen Softwareentwicklungs-Methoden berücksichtigt. Außerdem werden Projektrisiken reduziert, indem man das Projekt in kleinere (d.h. in Inkremente) aufteilt. In den praktischen Details unterscheiden sich die verschiedenen agilen Methoden, doch sie basieren alle auf den Idealen des „agile manifesto“ [BB01]:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

Dies bedeutet einen Paradigmenwechsel in der Softwareentwicklung, der nicht nur das Projektteam betrifft, sondern auch die gesamte Organisation auf Seiten des Softwareherstellers und des Kunden.

Eine der bekanntesten und vom Wasserfallmodell besonders weit entfernten agilen Methoden ist das Extreme Programming (XP) [Be00]. Es verwendet die folgenden Praktiken:

- Planung: Kunde vor Ort, kurze Releasezyklen, Planungsspiel (planning game) mit story cards, Testen der Akzeptanz
- Programmierung: einfaches Design, Programmieren in Paaren, Testen des Kodes, Refactoring
- Unterstützende Praktiken: gemeinsame Verantwortlichkeit, Programmier-standards, fortlaufende Integration, Metapher, 40-Stunden-Woche.

Damit diese Praktiken angewendet werden können, müssen verschiedene Voraussetzungen erfüllt sein. Beispielsweise muss zwischen den Mitgliedern des Projektteams eine intensive Kommunikation möglich sein, z.B. indem die räumlichen Voraussetzungen gegeben sind. Hierzu gehört auch, dass mindestens ein Vertreter des Kunden zur Verfügung steht, am besten „on site“, d.h. dort, wo die Software entwickelt wird. Außerdem darf dem Projekt kein Werkvertrag mit festem Lieferumfang zugrunde liegen, und es muss Vertrauen zwischen den beiden Vertragsparteien herrschen, so dass keine ausführlichen Dokumente nötig sind [Fa02]. Idealerweise wird XP in Teams von fünf bis zehn Personen angewendet. Agile Entwicklung im Großen ist jedoch ebenfalls möglich [Ec04].

In agilen Softwareentwicklungsmethoden sind die Anforderungen und deren Prioritäten (nicht unbedingt deren Dokumentation) die wesentliche Grundlage für die Projektplanung. So spielen beispielsweise die Story Cards im XP eine wichtigere Rolle als die Anforderungsspezifikation im Wasserfallmodell.

Allerdings sorgen durch den Kunden oder auch die softwareerstellende Firma selbst vorgegebene Randbedingungen dafür, dass viele agile Prinzipien in der Praxis nicht umgesetzt werden können. Beispielsweise stellt der Kunde nicht leicht einen Mitarbeiter für die Projektarbeit ab, und Werkverträge sind Standard.

3 FDD als Kompromiss zwischen Wasserfallmodell und Agilität

Da das Wasserfallmodell riskant ist und agile Methoden sich an Randbedingungen stoßen, ist ein Kompromiss nötig: ein Vorgehensmodell, das zwischen beiden liegt und deren Vorteile so weit möglich kombiniert. Als ein solches wird hier Feature Driven Development (FDD) vorgestellt.

FDD wurde von Jeff De Luca und Peter Coad in einem größeren Software-Projekt in Singapur Mitte der Neunziger Jahre entwickelt und erstmals eingesetzt [Ne]. Das zentrale Konzept von FDD ist das Feature. Es wird von den Erfindern von FDD wie folgt definiert [LD99]: Ein Feature ist ein kleines Ergebnis, das „useful in the eyes of the client“ ist. Das Feature umfasst meist mehrere Anforderungen. Ein Feature muss innerhalb von maximal zwei Wochen realisierbar sein oder aufgeteilt werden. Die Features dienen sowohl dem Projektleiter als auch dem Kunden als Mittel der Kontrolle des Projektfortschritts. Jedes Feature wird als ein Projekt für sich behandelt mit den Phasen Planung, Entwurf und Implementierung. Bevor jedoch jedes einzelne Feature realisiert werden kann, werden ein Gesamtmodell des Systems und eine Featureliste erstellt [Gy03].

4 Erfahrungen aus einer Fallstudie

Mit FDD habe ich in der Rolle als Projektleiterin und Anforderungsingenieurin in einem Projekt Erfahrungen gemacht, in dem die Randbedingungen die Anwendung von FDD nahe legten. Weder das Wasserfallmodell noch eine agile Methode waren hier ideal. Ohne dieses Projekt im Detail zu beschreiben, soll hier zusammengefasst werden, welche Randbedingungen dazu führten, dass FDD hier so gut passte. Außerdem wird beschrieben, welche Risiken in diesem konkreten Fall durch FDD erfolgreich abgefangen werden konnten.

4.1 Randbedingungen, die ideal zu FDD passten

Einerseits sollte das Projekt als eines durchgeführt werden, ausdrücklich nach dem Wasserfallmodell, andererseits bestand es aus voneinander teilweise unabhängigen Teilprojekten und Features, die sich unterschiedlich zu entwickeln drohten.

Sowohl in der Firma, die das Softwareprojekt durchführte, als auch beim Kunden galt das Wasserfallmodell als Standard, der eingehalten werden muss. Das bedeutete praktisch, dass die im Wasserfallmodell üblichen Dokumente erstellt werden mussten. Dazu gehörten ein unterschriebener Werkvertrag, der den Lieferumfang und den Preis des zu erstellenden Gewerks definiert, offizielle Meilensteine (für den Statusbericht an das Management beider Firmen), eine vorhersehbare Personalplanung, eine Anforderungsspezifikation und deren Abnahme, Entwurf, Entwicklerdokumentation, Testfälle und Testprotokolle. Change Requests waren möglich und üblich. Es verstieß jedoch gegen keine Regel, die Projektplanung und Projektdokumente nach Features zu gliedern.

Das Projekt sollte eine bereits beim Kunden produktiv laufende Software erweitern. Das Projekt sollte die Anforderungen von vier Benutzergruppen umsetzen. Zwei dieser Gruppen (Nr. 1 und 2) waren bereits Benutzer des Systems, die Gruppen Nr. 3 und 4 jedoch noch nicht. Die Idee, das Projekt in zwei aufzuteilen, wurde verworfen. Der Kunde bestand darauf, dass die Einführung der neuen Softwareversion und die Schulungen für alle vier Benutzergruppen gleichzeitig stattfinden. Um den knappen End- und Schulungstermin einzuhalten, mussten alle Entwickler sofort beginnen.

Zu dem Zeitpunkt, zu dem das Projekt beginnen sollte, war die Situation die folgende: Die Anforderungen für die Benutzergruppe 1 waren detailliert als Szenarien erfasst, teilweise sogar bereits der Entwurf erstellt, für die zweite Benutzergruppe, die komplexere Erweiterungen wollte, waren die Anforderungen zum größten Teil spezifiziert. Für Gruppe 3 wurde noch eine Kosten-Nutzen-Abwägung durchgeführt und überlegt, ob das Projekt überhaupt durchgeführt werden soll, während Gruppe 4 die Entscheidung von Nr. 3 abwartete. Ausgerechnet die Ansprechpartner dieser Gruppen 3 und 4 waren schlecht erreichbar. Für deren Anforderungen existierte nur eine Featureliste.

Die von den verschiedenen Gruppen beauftragten Features waren jeweils (aus Benutzersicht) weitgehend unabhängig voneinander, zum großen Teil durch das Berechtigungskonzept ohnehin für die anderen Gruppen nicht sichtbar.

4.2 Durchführung von FDD

Die im Wasserfallmodell üblichen Dokumente wurden erstellt. Das Projekt und alle Dokumente wurden nach Features gegliedert. Eine direkte Zuordnung von Features zu Arbeitspaketen erlaubte die Traceability zwischen Projektplan und Anforderungsspezifikation sowie zwischen allen anderen Dokumenten. Die Realisierungsreihenfolge der Features wurde nach folgenden Kriterien festgelegt: Reife des Features (reife zuerst), Komplexität (komplexe zuerst), Abhängigkeiten (technische und aus Nutzersicht grundlegende zuerst, d.h. solche, deren Funktionieren für das Testen der anderen nötig sein würde) und Verfügbarkeit der betreffenden Mitarbeiter (Entwickler und Ansprechpartner beim Kunden).

Es gab jedoch drei Anforderungsspezifiktionsdokumente: eines für die ersten beiden Benutzergruppen und jeweils eines für Nr. 3 und 4. Dies war nötig, um die Kommunikation mit den Kunden zu vereinfachen.

FDD wurde folgendermaßen abgewandelt: Es wurden weder die FDD Templates noch UML für die Beschreibung von Features verwendet, sondern natürlichsprachige Szenariobeschreibungen, zusammen mit Entwürfen von Bildschirmmasken und Datenfeldtabellen. Obwohl in FDD nicht explizit vorgesehen, erfolgte für jedes Feature die Entwicklung iterativ, d.h. im Detail: Spezifikation, Entwurf, interne Tests, Tests durch den Kunden, Neuspezifikation, ggf. Neuverhandlung des Preises, Entwurf, usw..

4.3 Risiken, die durch FDD abgefangen wurden

Die Features stellten trotz der Volatilität der Anforderungen eine stabile Grundlage für die Projektplanung dar. Der Werkvertrag legte fest, welche Features zu realisieren waren. Es fielen während des Projektes keine davon weg. Volatil war nur das „wie“.

Änderungswünsche des Kunden konnten leicht berücksichtigt werden. Gruppe 2 hatte beim Testen neue Wünsche. Hier wurde nachverhandelt und entweder durch einen Change Request das Projektvolumen erhöht oder ein anderes Feature schlichter realisiert. Für solche Verhandlungen sind Features ein geeignetes Konzept, denn der Kunde kann sie sich vorstellen und kennt ihren Nutzen.

Da die Ansprechpartner auf Kundenseite nicht vor Ort waren und ihr Tagesgeschäft Priorität vor der Projektarbeit hatte, mussten die Entwickler oft auf Rückmeldung warten. Dies konnte abgefangen werden: Da jeder Entwickler an mehreren Features arbeitete, füllte er diese Wartezeiten mit der Arbeit an einem anderen Feature.

Die Flexibilität hatte aber auch ihre Schattenseiten. Der Endtermin des Projektes verschob sich einerseits dadurch, dass die Ansprechpartner der Gruppen 3 und 4 schlecht erreichbar waren und sich folglich die Anforderungsspezifikation verzögerte. Außerdem beauftragten die Gruppen 2 bis 4 weitere Features. Als bekannt wurde, dass sich die Auslieferung verzögert, fügte auch Gruppe 1 noch Change Requests hinzu. FDD erlaubt es, weitere Features in das Projekt zu integrieren. Trotzdem musste dem Anwachsen des Projektvolumens ein Riegel vorgeschoben werden. Da alle Benutzergruppen von Anfang an eine schnelle Einführung der neuen Version gewünscht hatten, waren sie zu einer sinnvollen Versionierung bereit. Ohne die organisatorische Vorgabe, dass alle Features zum selben Zeitpunkt ausgeliefert werden sollen, hätten hier Teillieferungen die Lage entspannt.

Durch das schleppende Vorangehen der Anforderungsspezifikation mit den Gruppen 3 und 4 konnten für die entsprechenden Features bis zum Abnahmetermin weniger Iterationen durchgeführt werden. Schließlich wurde durch die Gruppen 3 und 4 die Abnahme verweigert.

Die Features waren so definiert worden, dass sie nicht nur aus Kundensicht Sinn machen, sondern auch technisch möglichst unabhängig sind. Dies war insbesondere dort nicht möglich, wo verschiedene Benutzergruppen auf dieselben Daten zugriffen. Trotzdem war mit etwas Mehraufwand die pünktliche Produktivsetzung der Features für die Gruppen 1 und 2 möglich, während diejenigen für die Gruppen 3 und 4 wo technisch machbar nicht ausgeliefert wurden. Im Sinne einer höheren Flexibilität sollten Features technisch unabhängig voneinander realisiert werden, um Teillieferungen zu erlauben.

5 Zusammenfassung

Es wurde eine Fallstudie beschrieben, in der FDD in einer Situation angewendet wurde, in der zwar das Wasserfallmodell verlangt, jedoch der Komplexität der Situation nicht angemessen war. FDD stellt folglich nicht nur einen Kompromiss zwischen dem starren Wasserfallmodell und den agilen Methoden dar, sondern auch eine Möglichkeit, in einer auf das Wasserfallmodell fixierten Umgebung agile Methoden umzusetzen. Es gab in dieser Fallstudie wie in einem sequentiellen Projekt üblich einen Vertrag und definierte Meilensteine. Es wurden außerdem die im Wasserfallmodell üblichen Dokumente erstellt, die nach Features gegliedert waren, was die Traceability zwischen den Dokumenten auf einfache Weise sicherstellte. FDD hat sich aus Projektleitungssicht ausgezeichnet bewährt, da sich anhand von Features sehr gut mit dem Kunden kommunizieren und verhandeln lässt. Sie erlauben eine modulare Projektsteuerung und -kontrolle und die einfache Integration zusätzlicher Features in das Projekt.

Literaturverzeichnis

[BB01] Beck, K.; Beedle, M.; van Bennekum, A.; Cockburn, A.; Cunningham, W.; Fowler, M.; Grenning, J.; Highsmith, J.; Hunt, A.; Jeffries, R.; Kern, J.; Marick, B.; Martin, R.C.; Mellor, S.; Schwaber, K.; Sutherland, J.; Thomas, D.: Manifesto for Agile Software Development. http://agilemanifesto.org/, 2001.

[Be00] Beck, K.: Extreme programming explained. Addison-Wesley, Upper Saddle River, 2000.

[Ec04] Eckstein, J.: Agile Softwareentwicklung im Großen. dpunkt.verlag, Heidelberg, 2004.

[Fa02] Favaro, J.: Managing Requirements for Business Value. In: IEEE Software 19(29) 2002; S. 15-17.

[Gy03] Gyger, D.: Feature-Driven Development. Seminar in Software Engineering, Universität Zürich, 2003, http://www.ifi.unizh.ch/groups/req/courses/seminar_ws03/08_Gyger_Fdd_Ausarbeitung.pdf

[Hu95] Humphrey, W.S.: A Discipline for Software Engineering. SEI Series in Software Engineering. Addison-Wesley, Upper Saddle River, 1995.

[LD99] Lefebvre, E.; De Luca, J.: Java Modeling in Color with UML. Prentice Hall, NJ, 1999.

[Ne] Nebulon Pty. Ltd: http://www.nebulon.com

[Ro70] Royce, W.: Managing the Development of Large Software Systems. In: Proc. IEEE WESCON 26, 1970; S. 1-9.

[ZR96] Ziv, H.; Richardson, D.: The Uncertainty Principle in Software Engineering. In: Proc. 19th Int. Conf. on Software Eng., ICSE, Boston, MA, 1997.