XP Controlled

Ein automatisierter Prozess für die Agile Softwareentwicklung

von Ursula Meseberg

Jeder Softwareentwickler kennt das: Termine werden trotz realistischer Planung nicht eingehalten. IT-Systeme entsprechen trotz ausführlich dokumentierter Anforderungen nicht den Wünschen des Kunden. Agile Methoden – allen voran das eXtreme Programming – versprechen Abhilfe. Ihr Rezept: Anforderungsgetriebenes, iteratives und selbstorganisiertes Vorgehen des Projektteams. Für manchen Projektleiter klingt das nach Chaos und Anarchie. Sind Agile Projekte also unkontrollierbar?

Die Agilen Methoden [1] und speziell das eXtreme Programming [2] stellen den aktuellsten Versuch dar, die Hauptprobleme der Softwareentwicklung – sich häufig ändernde Anforderungen, Terminverzögerungen und mangelnde Qualität – in den Griff zubekommen. Die Kernkonzepte, die sie dazu anbieten, sind:

- Einbeziehung des Kunden in das Projektteam,
- konsequente Orientierung an den Anforderungen, genauer an ihrem Geschäftswert für das Unternehmen des Kunden,
- kurze Releasezyklen,
- iteratives Vorgehen, das vom Projektteam selbst organisiert wird,
- integriertes Testen.

Will man diese Konzepte erfolgreich in die Praxis tragen, muss man daraus ein konkretes Vorgehensmodell ableiten, an dem sich alle Projektbeteiligten orientieren können. Einen solchen Agilen Prozess hat die microTOOL GmbH, ein Softwarehersteller aus Berlin, für die eigene Produktentwicklung und für ihre Kundenprojekte erstellt. Der Prozess, dessen wichtigste Schritte nachfolgend beschrieben werden, ist für die objektorientierte Entwicklung mit der UML auf Windows-Plattformen konzipiert. Er sieht den Einsatz eines UML-Tools vor und ist unter anderem auf die Implementierung in C# mit Microsoft Visual Studio .NET und auf das Testen mit Hilfe von NUnit zugeschnitten.

Der Prozess lernt Laufen

Wie gestaltet man einen Agilen Prozess, der eine echte Chance hat in der Praxis gelebt zu werden? Ein wichtiger Tipp findet sich bei Jim Highsmith, einem der Vordenker der Agilen Methoden. Er rät [3]: „Think flow, not batch." Wir haben diesen Rat beherzigt und den Agilen Prozess workflow-fähig angelegt. Das heißt, der Prozess kann in einem Projekt durch ein Workflow-Managementsystem maschinell unterstützt werden. Das im konkreten Fall eingesetzte Workflow-Managementsystem macht die Projektaktivitäten mit ihren Vorgängern, Nachfolgern und aktuellen Bearbeitungszuständen sichtbar. Es zeigt außerdem für jede Aktivität die Ein- und Ausgangsprodukte mit ihren Ist- und Sollzuständen . Dadurch wird der Projektverlauf für alle Beteiligten nachvollziehbar.

Will man iterativ arbeiten, wird die Versionskontrolle besonders wichtig. Es wird deshalb ein Workflow-Managementsystem benutzt, das auch Funktionen eines Konfigurationsmanagement-Tools besitzt. Es übernimmt die Versionierung der Produkte, die im Projektverlauf entstehen, und führt ihre Historie. Das verwendete Workflow-Managementsystem realisiert das Source Code Control (SCC) Interface von Microsoft und wird als Standard SCC-Provider eingesetzt. Damit wird die Versionierung gerade der Ergebnisse besonders leichtgängig, die unter Microsoft Visual Studio .NET entwickelt werden.

EXtreme Programming ist in den Worten seines Erfinders Kent Beck „eine leichte, effiziente, risikoarme, flexible, kalkulierbare, exakte und vergnügliche Art und Weise der Softwareentwicklung." Gehen Leichtigkeit, Flexibilität und Spaß nicht verloren, wenn die Arbeit in den Rahmen eines Workflow-Managementsystems gezwängt wird? Das Workflow-Managementsystem belastet die Entwicklungsarbeit nicht. Unsere Erfahrung zeigt das Gegenteil: Es macht sie sicherer und vereinfacht die Kommunikation. Denn viele Rückfragen („Ist... schon angefangen/fertig?", „Wo finden ich denn...?") entfallen. Das Sichtbarmachen des Projektablaufs hilft dem Team, das Richtige zur rechten Zeit zu tun. Tritt das Unerwartete ein, kann der Projektleiter erkennen, welche Optionen er besitzt. Und der Kunde behält den Überblick darüber, was mit seinem Geld geschieht. Was verlangt das Workflow-Managementsystem vom Team im Gegenzug? Jeder Projektbeteiligte muss

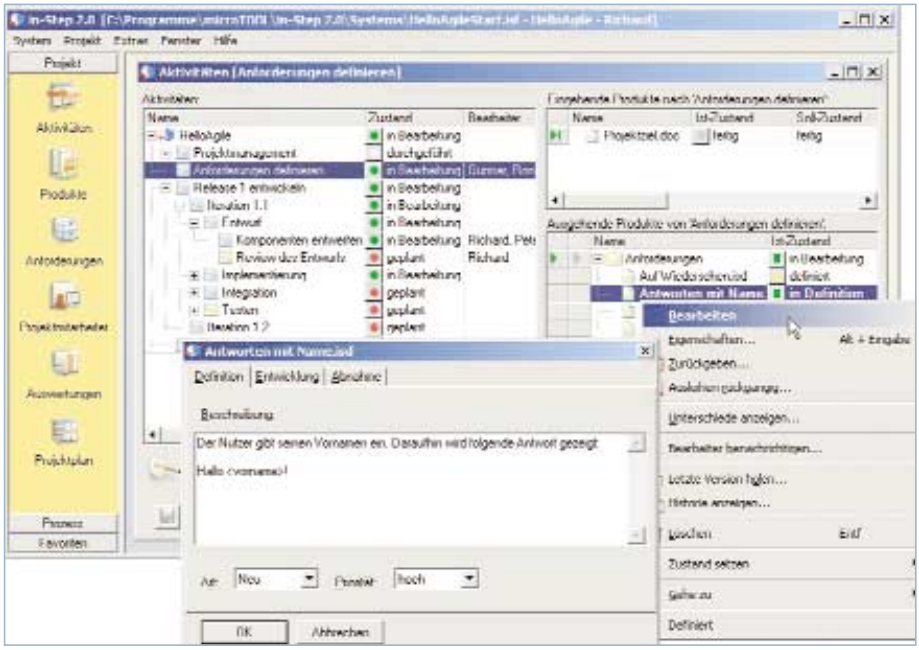

Abb. 1: Stories erfassen – eine einfache Lösung für die Anforderungsdefinition

den Beginn und das Ende seiner Aktivitäten per Mausklick signalisieren, seine Ergebnisse zum Versionieren ein-/auschecken und einen entsprechenden Zustand setzen, wenn ein Produkt fertig ist.

Anforderungen – der Motor der Agilen Entwicklung

Mit dieser technischen Ausstattung läuft der Prozess wie folgt ab: Sobald die Ziele des Projekts und der Geschäftszweck der neuen Anwendung geklärt und dokumentiert sind, werden die Anforderungen definiert. Sie werden als so genannte „Stories" in wenigen Sätzen formuliert. Dies geschieht im Team gemeinsam mit dem Kunden. Noch während der Teamsitzung werden die Stories maschinell festgehalten. Dazu wird ein einfaches Bildschirmformular verwendet, das vom Workflow-Managementsystem angeboten wird (siehe Abb. 1). Die erfassten Anforderungen werden als XML-Dateien gespeichert und wie alle Ergebnisse versioniert.

Alle Stories sollten etwa den gleichen Realisierungsumfang von fünf bis zehn Personentagen besitzen. Unsere Erfahrung zeigt: Einem Projektteam, das mit dem Anwendungsbereich des zu entwickelnden Systems vertraut ist, fällt es leicht, Anforderungen in gleichgewichtige Stories zu gliedern. Was aber, wenn nicht genug Klarheit über den Anwendungsbereich besteht, um Stories zu formulieren? Dann ist jedes Mittel recht: Der Prozess bietet bewusst Spielraum für die Entwicklung freier Produkte. Sie können in den Workflow maschinell eingebunden und aus dem Workflow-Managementsystem direkt zur Bearbeitung mit dem passenden Werkzeug aufgerufen werden. So kann man zum Beispiel nach Bedarf mit einem UML-Tool Aktivitätsdiagramme zur Analyse der Business-Prozesse oder Anwendungsfalldiagramme zur Abgrenzung des Systems erstellen. Zur Stabilisierung von Anforderungen ist auch das Modellieren von Business-Klassen mit einem UML-Tool und ein anschließendes Prototyping hilfreich – ein Vorgehen, das zum Beispiel der Anwendungsgenerator JANUS [4] der Dortmunder otris AG unterstützt. Ganz gleich, welche zusätzlichen Mittel eingesetzt werden: Ziel ist nicht, eine umfassende Dokumentation der Anforderungsanalyse zu erstellen. Es geht ausschließlich darum, dass alle Projektbeteiligten den Anwendungsbereich und seine Probleme verstehen. Das Team soll in die Lage versetzt werden, Stories zu formulieren und damit kleine, realisierbare Einheiten zu definieren. Sie sind die Voraussetzung für die nächsten Projektaktivitäten: Die Release- und Iterationsplanung.

Abb. 2: Wie geht es voran? Die Auswertung der Anforderungen und ihrer Zustände liefert ein aktuelles Maß für den Projektfortschritt

Agil – aber nicht planlos

Ein IT-System wird über mehrere Releases realisiert. Wir haben uns auf einen Release-Zyklus von ca. drei Zeitmonaten festgelegt. Wichtig dabei: Jedes Release muss einen Mehrnutzen für den Kunden schaffen. Gegenstand der Release-Planung ist es, aus der Menge der Anforderungen diejenigen festzulegen, die pro Release realisiert werden sollten. Dies geschieht wiederum im Team. Auch hier ist der Kunde dabei. Denn ein zentrales Kriterium zur Auswahl der Anforderungen für ein Release bilden die spezifischen Geschäftsinteressen des Kunden. Ein weiteres wichtiges Auswahlkriterium sind mögliche funktionale Abhängigkeiten, die es notwendig machen, Anforderungen gemeinsam zu realisieren.

Der Weg zu einem Release führt meist über zwei bis drei Iterationen. Pro Iteration wird ein Teil der Anforderungen an das Release realisiert – welcher Teil, das wird bei der *Iterationsplanung* bestimmt. Die Anforderungen werden jetzt unter ganz anderen Kriterien als bei der Release-Planung ausgewählt. Hier sind es der Aufwand, die technischen Abhängigkeiten, die gemeinsame Testbarkeit und die Verfügbarkeit der benötigten Mitarbeiter. Das Team schätzt den Aufwand für jede Story, die innerhalb einer Iteration realisiert werden soll. EXtreme Programming [5] schlägt vor, ganz pragmatisch mit dem „Wetter von gestern“ zu planen, also in die Aufwandschätzung Erfahrungen früherer Projekte und Releases einfließen zu lassen. Wie soll man sich aber am Wetter von gestern orientieren, wenn es keine „Wetterdokumentation“ gibt? Der Prozess sieht deshalb vor, dass der geschätzte Aufwand für jede Story zusammen mit ersten Lösungsideen, die bei der Iterationsplanung entstehen, maschinell festgehalten werden: Das Bildschirmformular, das zur Erfassung der Anforderungen verwendet wird, bietet dazu eine entsprechende Eingabemöglichkeit. So entsteht im Projekt eine Art Gedächtnis, auf das man in späteren Releases oder Folgeprojekten zurückgreifen kann.

Der Prozess erlaubt jederzeit, neue Anforderungen zu formulieren, Anforderungen zu streichen oder zwischen Iterationen und Releases zu verschieben. Diese Aktionen schlagen natürlich auf die Planung durch. Das Workflow-Managementsystem kennt nicht nur die geplanten Aufwände, sondern hält auch die Plan- und Ist-Dauer der Projektaktivitäten fest. Das macht es dem Projektleiter leichter, immer wieder Plan und Realität in Einklang zu bringen.

Der Kunde ist mit im Boot

Wie im eXtreme Programming trägt der Entwickler auch bei unserer Vorgehensweise die Verantwortung dafür, dass sein Code läuft. Er testet die Methoden der von ihm entwickelten Klassen maschinell und erstellt zu diesem Zweck Testklassen. Diese Testklassen werden vorab nicht geplant oder entworfen. Sie werden direkt implementiert und im Workflow-Managementsystem versioniert.

Ganz anders wird beim Test der höherwertigen Systemleistungen vorgegangen. Hier ist nicht allein der Entwickler, sondern auch der Kunde in der Verantwortung: Im Rahmen der Release- und Iterationsplanung wird im Team festgelegt, welches Verhalten in welchem Umfang geprüft werden soll. Damit hat der Kunde, der in die Planung eingebunden ist, direkten Einfluss darauf, wo „good enough“ reicht, beziehungsweise für welche Systemleistungen besonders hohe Qualitätskriterien erfüllt sein müssen. Das Workflow-Managementsystem stellt für Testanforderungen ebenfalls ein Bildschirmformular zu Verfügung und speichert sie als XML-Dateien. Es werden drei Arten von Testanforderungen unterschieden: Testanforderungen an das Bedienverhalten, an das fachliche Verhalten (sie entstehen in der Regel im Verlauf der Release-Planung) und an das technische Verhalten (sie sind ein Ergebnis der Iterationsplanung). Für die Testanforderungen werden Testsuites erstellt. Sie werden – anders als die Testklassen für elementares Verhalten – mit Klassendiagrammen im UML-Tool entworfen und unter Verwendung des Frameworks NUnit implementiert. Doch halt, bei der Implementierung sind wir noch nicht. Zurück zur treibenden Kraft im Prozess, den Anforderungen.

Immer wissen, wo man steht

Die Anforderungen durchlaufen in einer Iteration mehrere Zustände, bis sie der Kunde schließlich als abgenommen kennzeichnet. Alle Zustände werden maschinell festgehalten. Das Workflow-Managementsystem erlaubt es, beliebige Auswertungen eines Projekts vorzunehmen. Es bietet sich an, von Zeit zu Zeit die Zustände der Anforderungen auszuwerten und alle aufzulisten, die zum Beispiel im Zustand *beauftrag* oder *implementiert* sind. Mit diesem einfachen Mittel gewinnt man ein Maß für den inhaltlichen Projektfortschritt. Es ist weit aussagestärker als der übliche Soll-/Ist-Vergleich von Start- oder Endterminen der Projektaktivitäten.

Nichts ist stabil – Entwurf durch Refactoring

Die Anforderungen, die einer Iteration zugeordnet wurden, sind die Eingangsinformation für die Aktivitäten *Entwurf*, *Implementierung*, *Integration* und *Testen*. Diese Aktivitäten kommen in jeder Iteration vor. Das hat erhebliche Auswirkungen auf die Entwurfsstrategie. Denn wenn der Entwurf in jeder Iteration erweitert und – unvermeidlich – auch geändert wird, gibt es bis zur letzten Iteration für das letzte Release kein stabiles Entwurfsdokument. In unserem Prozess wird diese Tatsache akzeptiert: Pro Iteration wird nur das entworfen, was nötig ist, um die Anforderungen zu erfüllen, die der Iteration zugeordnet wurden. Es wird nicht versucht, einen flexiblen Entwurf zu erstellen, der zukünftige Anforderungen bereits vorausschauend berücksichtigt. Dieses Vorgehen hat eine wichtige Konsequenz. Sie heißt: Kontinuierliches Refactoring. Ziel des Refactoring ist es, die Änderungen und Erweiterungen des Entwurfs so vorzunehmen, dass zum Ersten das Ergebnis verständlich bleibt und zum Zweiten die in den vorangegangenen Iterationen entwickelten Teile ihr beobachtbares Verhalten nicht ändern, was im Regressionstest nachzuweisen ist. Für den Entwurf wird ein UML-Tool eingesetzt. Klassendiagramme sind ein Muss. Package-Diagramme, die die Softwarearchitektur beschreiben, werden ebenfalls verwendet. Der Einsatz anderer UML-Mittel – wie Sequenz- oder Zustandsdiagramme – liegt im Ermessen der Entwickler. Damit der Tool-Einsatz den Entwicklungsprozess nicht bremst, wurde das UML-Tool technisch so in das Workflow-Managementsystem integriert, dass die zu bearbeitenden Diagramme direkt aus dem Kontext einer Projektaktivität aufgerufen werden können (siehe Abb. 3).

Das Refactoring wird dadurch erleichtert, dass das eingesetzte UML-Tool eine ganze Reihe von Refactoring-Methoden maschinell unterstützt. Dazu gehören das Extrahieren von Interfaces und von Super- oder Unter-Klassen, das Verschieben von Attributen und Methoden, das Ändern der Namen von Klassen, Attributen, Methoden und Parametern, sowie das Entzerren von Vererbungsstrukturen.

In jeder Iteration wird nach der Implementierung ein neues Release erstellt, das

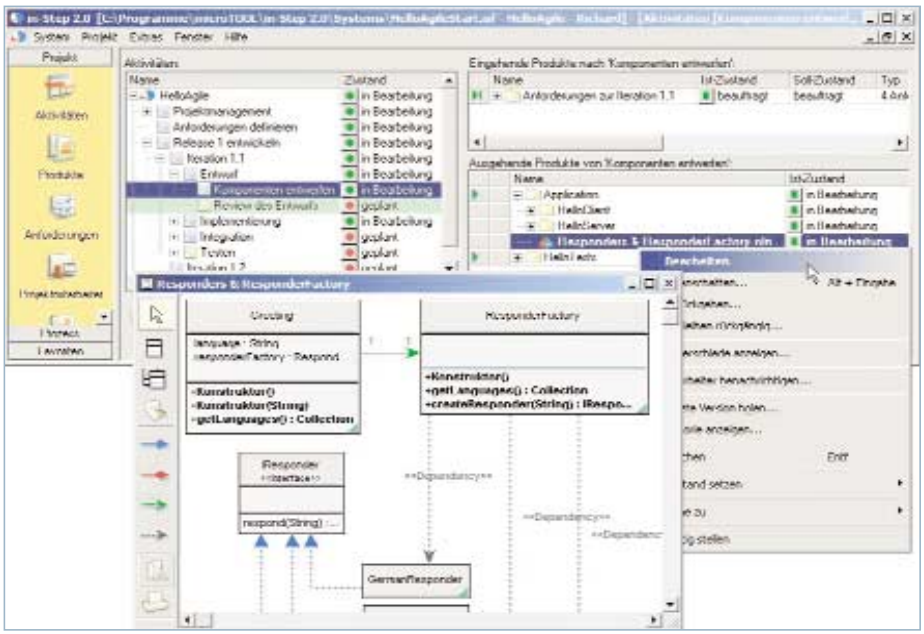

Abb. 3: Kurzer Weg – Aufruf eines UML-Diagramms im Workflow-Managementsystem

einem Regressionstest unterzogen wird. Bis auf das Ergebnis der letzten Iteration, das verteilt und eingeführt wird, sind diese Zwischenreleases nicht für den Echtbetrieb bestimmt. Jedes davon bildet den Ausgangspunkt für die nachfolgende Iteration, den nächsten Schritt auf dem Weg zum fertigen System.

Zusammenfassung

Ziel der Agilen Softwareentwicklung ist es, so früh wie möglich einen Nutzen für den Kunden zu schaffen. Dabei wird akzeptiert, dass sich im Projektverlauf die Prioritäten des Kunden ändern, dass neue Anforderungen entstehen oder vorhandene modifiziert werden. Durch einen definierten Prozess wird die Instabilität der Anforderungen beherrschbar. Wird der Prozess ablauforientiert angelegt, kann der Projektablauf kontrolliert und gesteuert werden. Das Mittel dazu ist ein Workflow-Managementsystem. Eine solche maschinelle Unterstützung hat eine weitere interessante Wirkung: Der agile Prozess wird dadurch skalierbar und bleibt nicht auf sehr kleine Projekte beschränkt.

Nach dem Agilen Prinzip „mit leichtem Gepäck reisen“ werden Tools und Methoden bedarfsgerecht eingesetzt. Drei Projektaufgaben sollten jedoch unbedingt durch Tools unterstützt werden, denn sie sind ein Schlüssel zum Projekterfolg: Das Management der Anforderungen und die Verfolgung ihrer Realisierung, der Softwareentwurf mit seinem kontinuierlichen Refactoring sowie die Entwicklung und Durchführung der Tests.

Der hier vorgestellte Prozess bietet sich nicht nur für die Produktentwicklung eines Softwareherstellers an. Weitere Einsatzgebiete finden sich überall dort, wo stark anforderungsorientiert entwickelt wird, wie in Auftragsprojekten von IT-Dienstleistern oder im Bereich des Change Managements in IT-Organisationen.

Ursula Meseberg ist bei der microTOOL GmbH (www.microTOOL.de) für den Bereich Methoden verantwortlich. Sie blickt auf langjährige Erfahrungen mit objektorientierten Methoden zurück und ist Autorin eines UML-basierten Basisprozesses, der unter dem Namen actiF veröffentlicht wurde.

Links & Literatur

[1] www.agilealliance.org/ und www.agilemanifesto.org/
[2] Kent Beck: Extreme Programming, Das Manifest, Addison-Wesley 2000
[3] Jim Highsmith: Agile Software Development Ecosystems, Addison-Wesley 2002
[4] www.otris.de/
[5] Kent Beck, Martin Fowler: Extreme Programming planen, Addison-Wesley 2001