Quantitativer und qualitativer Vergleich von Anforderungen bei agilen und konventionellen Softwareprojekten

Melanie Hennemann und Eric Knauss
FG Software Engineering, Leibniz Universität Hannover
{hennemann, knauss}@se.uni-hannover.de

Einleitung

In der Softwareentwicklung hängt der Erfolg oder Misserfolg eines Projektes wesentlich davon ab, ob die Wünsche des Kunden umgesetzt wurden oder nicht. Im Kundengespräch werden die Anforderungen an das System erhoben, die letztendlich die Wünsche des Kunden widerspiegeln. Um den Kunden zufriedenzustellen, müssen diese Anforderungen mit hoher Priorität umgesetzt werden. In konventionellen und agilen Methoden gibt es dabei wesentliche Unterschiede, wie mit diesen Anforderungen umgegangen wird. In diesem Beitrag zeigen wir die Vor- und Nachteile beider Softwareentwicklungsmethoden in Bezug auf die Struktur und Art der Anforderung. Zu diesem Zweck werden studentische Softwareprojekte analysiert. Dies ist ein erster Schritt zu einer Grundlage für die Entscheidung, welche Methode in einem konkreten Projekt eingesetzt werden soll.

Situation in konventioneller Softwareentwicklung

Konventionellen Entwicklungsmethoden [1,2] liegt eine feste Abfolge von Tätigkeiten zugrunde. Die Resultate aller Schritte werden dabei ausführlich dokumentiert. Gespräche zwischen dem Kunden und den Entwicklern sind nach der Anforderungsanalyse sehr unwahrscheinlich und in der Regel wird auf Basis von Dokumenten kommuniziert. Änderungen der Kundenwünsche können in späteren Phasen nur noch schwer und mit erheblichem Kostenaufwand berücksichtigt werden. Dafür wird das Ergebnis sehr nah an der Anforderungsspezifikation liegen, die dann als Vertragsgrundlage dienen kann. Aber viele späte Anforderungsänderungen, schnellere Releasezyklen und Marktdruck machen es heute oft schwierig, mit konventionellen Methoden Kundenwünsche angemessen zu berücksichtigen.

Situation in agiler Softwareentwicklung

Bei agilen Projekten [3,4] wird zugunsten der Flexibilität auf eine detaillierte Dokumentation verzichtet. Kunde und Entwickler bleiben ständig in Kontakt und auf Änderungswünsche kann gut reagiert werden. Schon früh wird dem Kunden lauffähige Software geliefert. Allerdings ist es schwer, einen Kunden ohne genauen Vorgehensplan von einer Idee zu überzeugen, und es besteht ein gewisses Risiko, sich in Detailfragen zu verzetteln.

Aufbau der Messung

Unterschiede in Zeitpunkt, Art und Granularität von Anforderungen in konventionellen und agilen Softwareentwicklungsmethoden wurden nach der GQM-Methode untersucht [5]. Dazu wurden Hypothesen aufgestellt, wie sich die agile von der konventionellen Softwareentwicklung aus Sicht der Anforderungen unterscheidet. Dann wurde in einer Fallstudie versucht, diese Hypothesen zu belegen.

H1 (*Mehr nicht-funktionale Anforderungen*) Der Kunde bekommt durch die direktere Kommunikation mit den Entwicklern und durch früh gelieferte nutzbare Software schon früh eine genaue Vorstellung von dem Produkt. Es ist anzunehmen, dass der Kunde dadurch im Projektverlauf viel mehr auf nicht-funktionale Anforderungen eingehen wird, als er das bei der Anforderungsspezifikation in einem konventionellen Projekt tun würde.

H2 (*Usability-Fokus*) Der Kunde kann die Software bei agilem Vorgehen schon früh im Projektverlauf testen, Feedback geben und so großen Einfluss auf die Qualität der Benutzbarkeit nehmen. Die Usability-Anforderungen werden daher einen größeren Anteil an den nicht-funktionalen Anforderungen haben als bei konventionellen Methoden.

H3 (*Details und niedrige Abstraktion*) Viele kleine Gespräche lassen viele Änderungen und Ergänzungen erwarten. Anforderungen werden stärker ins Detail gehen und Subfunktionen beschreiben.

Fallstudie

Beobachtungsgegenstand waren studentische Softwareprojekte: Auf der einen Seite sechzehn konventionelle Softwareprojekte (je 5 Bachelor-Studenten, 4 Monate Laufzeit, Teilzeit) und auf der anderen Seite zwei parallel laufende eXtreme-Programming-Projekte (XP) (je 8 Master-Studenten, 2 Wochen Blockzeit).

Die konventionellen Softwareprojekte fanden im Rahmen einer Pflichtveranstaltung für Informatikstudenten statt, in der die Studenten praktische Erfahrungen und Einblicke in die Softwareentwicklung erhalten sollten. Jedes Team bestand aus fünf Studenten und jedem Team stand ein technischer Berater zur Verfügung. Aufgabe war es, ein Softwareprodukt in einem Zeitrahmen von drei Monaten in einem wasserfallartigen Prozess fertig zu stellen. Kunde war ein fiktives Unternehmen, das durch Mitarbeiter des Lehrstuhls repräsentiert wurde.

Die XP-Projekte wurden innerhalb einer Intensivübung für agile Softwareentwicklung durchgeführt. Zwei Teams mit jeweils acht Studenten haben in einem Zeitraum von zwei Wochen acht Stunden täglich ein Softwareprodukt entwickelt. Dabei wurde auf die Einhaltung der Praktiken des XP besonders

Wert gelegt (vergleiche [6]). So war beispielsweise der Kunde während der ganzen Entwicklungsphase vor Ort (Onsite-Customer) und die Entwicklung wurde durch Test-First gesteuert.

Die Anforderungen wurden im Rahmen dieser Untersuchung von einer unabhängigen Person protokolliert, klassifiziert und dann ausgezählt, um anschließend anhand dieser Ergebnisse die Hypothesen bewerten zu können.

Ergebnisse der Messung

H1 (*Mehr nicht-funktionale Anforderungen*) Die Messung zeigte, dass in Softwareprojekten konventioneller Entwicklungsmethoden im Durchschnitt lediglich 31% der Anforderungen nicht-funktional waren, während man in den XP-Projekten zu einem Anteil von durchschnittlich mehr als 50% nicht-funktionaler Anforderungen kam. Es scheint durch die konkretere Kommunikation bei XP tatsächlich viel mehr über die nicht-funktionalen Eigenschaften der Software gesprochen zu werden. So gestaltet der Kunde das Programm schon während der Entwicklung mit. Dies ist ein Vorteil, der von XP erwartet wird und nun auch belegt werden konnte.

H2 (*Usability-Fokus*) Während bei den XP-Projekten der Anteil der Usability-Anforderungen bei rund 70% lag, lag dieser in konventionellen Softwareprojekten durchschnittlich nur bei 24%.

H3 (*Details und niedrige Abstraktion*) In den XP-Projekten wurde viel mehr über Subfunktionen sowie konkrete, messbare Anforderungen gesprochen als über Geschäftsziele und abstrakte Anforderungen. Der Anteil der Subfunktionen und messbaren Anforderungen gemessen an der Gesamtzahl der Anforderungen lag in den beiden ausgewerteten Projekten bei 67% und 80%.

Diskussion der Messergebnisse

In den beobachteten XP-Projekten waren mehr als die Hälfte der nicht-funktionalen Anforderungen Usability-Anforderungen (70%). Der Kunde in XP hat eine viel bessere Chance, sich genaue Vorstellungen über sein gewünschtes Endprodukt zu machen. Frühzeitig gelieferte und testbare Software geben dem Kunden mehr Aufschluss über die Eigenschaften seiner gewünschten Software. Häufige Kommunikationsmöglichkeiten mit den Entwicklern ermöglichen es dem Kunden, diese Wünsche auch zu vermitteln und ihre Umsetzung zu überprüfen. Bei konventionellen Softwareprojekten scheint es dem Kunden dagegen nur schwer möglich, die Entwickler bei der Berücksichtigung der Benutzerfreundlichkeit zu unterstützen. Die Entwickler sind viel mehr auf sich selbst gestellt und die Gefahr scheint groß, den Kunden am Ende nicht zufrieden stellen zu können.

Die XP-Projekte haben gezeigt, dass sehr wohl von der Möglichkeit Gebrauch gemacht wird, Anforderungen nachträglich ändern zu können. Mehr als die Hälfte der Anforderungen wurden erst nach dem ersten Planning Game (Iteration 0) erhoben. In P1 waren dies 168 von 257 Anforderungen (65%), in P2 sogar 67 % (109 von 163 Anforderungen).

Eine Erklärung können die eher skizzenhaften Aufzeichnungen der Basisanforderungen im Planning Game sein. Diese Aufzeichnungen können versteckte Anforderungen enthalten, die zwar nicht explizit besprochen wurden, die der aber Kunde voraussetzt. Es ist auch denkbar, dass die eher skizzenhaften Aufzeichnungen dadurch zustande kommen, dass der Kunde zu Beginn des Projekts selber noch gar nicht so genau weiß, wie die Details der fertigen Software aussehen sollen, und sie deshalb erst später im Projektverlauf den Entwicklern beschreiben kann.

Bewertung der Messung

Dieser Beitrag vergleicht quantitativ Anforderungen agiler und konventioneller Projekte, Als Metriken kamen dazu a) das Verhältnis funktionaler / nicht-funktionaler Anforderungen, b) Anteil von Anforderungen an die Bedienbarkeit und c) Granularität von Anforderungen, Diese Metriken wurden als Prozentzahlen angegeben. Die Ergebnisse waren dadurch gut und einfach analysierbar. Beobachtungsgegenstand waren Projekte im studentischen Umfeld. Um eine realitätsnähere Durchführung der Projekte zu geben, wäre der Einsatz eines realen Kunden denkbar und wünschenswert gewesen, da Konflikte möglich sind, wenn der Kunde auch zu den Veranstaltungsleitern zählt.

Fazit

Beide Arten der Softwareentwicklung zeigen Vor- und Nachteile und sowohl die agilen als auch die konventionellen Methoden bieten Lösungsansätze, um die Auswirkungen der Nachteile gering zu halten. Dieser Beitrag betrachtet die Entscheidung aus der Perspektive der Anforderungen. Je nach Projektart kann entschieden werden, welche Entwicklungsmethode am sinnvollsten ist oder ob man die für das Projekt wichtigsten Eigenschaften der verschiedenen Entwicklungsmethoden sogar kombinieren kann, um die Kundenwünsche bestmöglich zu erfüllen.

Referenzen

[1] Royce, W. W. "Managing the Development of Large Software Systems", Proceedings of IEEE WESCON, pp 328 - 338, 1970

[2] Rausch, A. & Höppner, S.; Petrasch, R. & Höppner, S. (ed.) "V-Modell XT – eine Einführung" (3), Kapitel 1, pp 1-24, Logos-Verlag, 2005

[3] Kent Beck +, "Manifesto for Agile Software Development", http://www.agilemanifesto.org, am 06.07.2008 eingesehen

[4] Schwaber, K. & Beedle, M., "Agile Software Development with Scrum", Prentice Hall, 2002

[5] van Solingen, R. & Berghout, E., "The Goal/Question/Metric Method: a practical guide for quality improvement of software development," McGraw-Hill Publishing Company, 1999

[6] Stapel, K.; Lübke, D. & Knauss, E. "Best Practices in eXtreme Programming Course Design", Proceedings of ICSE 2008, Leipzig, pp 769-776, 2008