Erfolgreiche Durchführung studentischer Softwareprojekte auf Basis der Dualen Softwareentwicklung nach Bertrand Meyer

Timo Acquistapace[*], **Joachim Goll**, **Manfred Dausmann**

Fakultät Informationstechnik der Hochschule Esslingen – University of Applied Sciences

Wintersemester 2015/2016

Agile Ansätze halten immer mehr Einzug in die Welt der Softwareentwicklung und sind in vielen Firmen und Projekten bereits zum Standard geworden. Der Ablauf agiler Prozesse unterscheidet sich erheblich von den zuvor eingesetzten spezifikationsorientierten Vorgehensweisen. Dieser Unterschied tritt vor allem im Hinblick auf eine durchdachte Planung, insbesondere Up-Front-Entscheidungen, und die Einbindung des Kunden zutage.

Studenten, welche sich während ihres Studiums immer wieder mit Gruppenprojekten befassen, müssen sich entscheiden, ob sie einem agilen oder einem spezifikationsorientierten Ansatz zur Umsetzung ihrer Projekte folgen sollen. Beide Herangehensweisen bieten sowohl Vor- als auch Nachteile, welche in der Folge diskutiert werden sollen. Im Zuge der durchgeführten Gruppen-Bachelorarbeit entstand ein kombinierter Ansatz, der nach Bertrand Meyer als „Duale Entwicklung“ zu charakterisieren ist und sich auch über Studentenprojekte hinaus hervorragend eignet.

Vorteile der beiden Vorgehensweisen

Agile Prozesse zeichnen sich dadurch aus, dass infolge einer iterativen, inkrementellen Entwicklung möglichst früh eine lauffähige Version des Systems bereitgestellt wird und somit auch in einer frühen Phase des Entwicklungsverlaufs Feedback vom Kunden zum System eingeholt werden kann. Der Kunde, bei Studentenprojekten in Gestalt des betreuenden Professors, kann somit frühzeitig Einfluss auf die Entwicklung nehmen, die Entwicklung verfolgen und hat durch das zyklisch ausgelieferte, lauffähige Produktinkrement einen ständigen Fortschrittsnachweis des Projekts. Die Architektur des Systems wird bei einer agilen Vorgehensweise nach und nach durch Refactoring gewonnen und optimiert.

Bei einem spezifikationsorientierten Vorgehen werden zunächst die Anforderungen aufgestellt und der Problembereich analysiert. Die daraus abgeleitete Architektur des Systems wird in einer Dokumentation festgehalten. Sind die Anforderungen, die Systemanalyse und die Architektur vom Team und dem Kunden akzeptiert und wurde somit ein sogenanntes „Shared Understanding“ geschaffen, so startet der Implementierungsprozess. Dieser steht aufgrund der ausgiebigen Vorüberlegungen auf einer soliden Basis und kann dadurch meist sehr geradlinig und ohne große Änderungen an der Systemarchitektur verlaufen.

Nachteile der beiden Vorgehensweisen

Wird rein spezifikationsorientiert vorgegangen, so werden viele Up-Front-Entscheidungen getroffen und sehr viel Zeit in die Dokumentation investiert. Die Erfahrung hat gezeigt, dass Studenten zu viel spezifizieren und dokumentieren und dass dadurch letztendlich die Zeit für die Umsetzung der erdachten Features fehlt. Das eigentliche Software-Produkt wird erst zu einem späten Zeitpunkt ausgeliefert, sodass es auch keine Möglichkeit zu einem frühen Feedback gibt.

Durch die große Zeitspanne, welche für die der Implementierung vorausgehenden Planung benötigt wird, entsteht ein weiteres Problem: Die Spezifikation veraltet. Anforderungen können sich in der Zwischenzeit geändert haben und dadurch kann das modellierte System bereits obsolet sein. Zudem besteht die Möglichkeit, dass die Kommunikation zwischen dem Kunden und dem Requirements-Ingenieur durch Missverständnisse und Fehlinterpretationen negativ beeinträchtigt wurde und somit falsche Anforderungen festgehalten und modelliert wurden. Derartige Fehler werden bei einer spezifikationsorientierten Vorgehensweise zu spät bemerkt, wodurch die entstehenden Fehlerkosten ein beträchtliches Ausmaß annehmen.

Agile Prozesse hingegen sind flexibel gegenüber Änderungswünschen des Kunden bzw. gegenüber Änderungen der Anforderungen, solange diese zwischen den einzelnen Entwicklungszyklen eingebracht werden[1]. Der Verzicht auf eine erste Planung und generell auf Up-Front-Entscheidungen birgt jedoch

[*] Diese Arbeit wurde durchgeführt bei der Firma IT-Designers GmbH, Esslingen-Zell

[1] Bei Scrum existiert hierfür die „Closed Window“-Regel. Diese besagt, dass ein laufender Sprint gegenüber Änderungen der Anforderungen geschlossen ist.

große Risiken, wenn ein komplexes System entwickelt werden soll. Es ist naiv, stets davon auszugehen, dass sich bei einer Erweiterung des ersten lauffähigen Inkrements die Komplexitäten lediglich addieren. Kommt es zu einer Zusammenarbeit der hinzugefügten Programmteile mit den bisher bereits implementierten, so kommt es meist auch zu einer sogenannten multiplikativen Komplexität, welche ohne vorherige Analyse des Gesamtsystems nur schwer handhabbar ist.

Generell bedarf ein Agiles Vorgehen großer Erfahrung bzw. einer kompetenten Anleitung durch erfahrene Softwareentwickler, die agile Verfahren sehr gut beherrschen und mit dem zu realisierenden Systemtyp bereits vertraut sind. Studenten-Projekte, welche agil durchgeführt werden, scheitern daher oftmals.

Die Duale Softwareentwicklung als erfolgreiche Kombination

Um die Vorteile beider Ansätze nutzen zu können, deren Nachteile jedoch auszumerzen, empfiehlt Bertrand Meyer die Duale Softwareentwicklung. In der frühen Phase eines Prozesses liegt hierbei der Fokus auf der Planung und auf Design-Entscheidungen, welche ein erweiterbares und skalierbares System mit der erforderlichen Infrastruktur garantieren. Diese Entscheidungen werden analog zu einer spezifikationsorientierten Vorgehensweise auf Basis initial aufgestellter Requirements und einer ersten Systemanalyse getroffen. Auch bei einer gewissen Unschärfe der initialen Anforderungen sollte es nach Meyer erfahrenen Softwareentwicklern möglich sein, derartige Entscheidungen zu fällen und das Projekt somit auf ein solides Fundament zu stellen.

Im weiteren Verlauf wird der Fokus auf die Entwicklung der Software gelegt und diese vorangetrieben, wobei regelmäßig Produktinkremente entwickelt und ausgeliefert werden. Das Refactoring der grundlegenden Architektur fällt, falls zuvor gute Vorüberlegungen und Design-Entscheidungen getroffen wurden, in seinem Umfang deutlich kleiner aus als bei einer rein agilen Vorgehensweise und dient nun eher als zusätzliches Hilfsmittel, denn als Allheilmittel.

Der hier angedeutete sequentielle Ablauf dieser beiden Schritte kann auch in einen parallelen Ablauf umgewandelt werden. Dadurch ist es möglich, dem Kunden erste Prototypen vorzustellen und dessen Feedback zu gewinnen, während parallel an der grundlegenden Systemarchitektur gearbeitet wird. Die Beteiligten beider Prozesse befinden sich in regelmäßigem Austausch, sodass deren Erkenntnisse kombiniert werden können. Dadurch wird sowohl erreicht, dass die erste Systemarchitektur den Kundenwünschen gerecht wird, als auch sichergestellt, dass Teile der ersten Prototypen auch später im Projekt verwendet werden können [1].

Einsatz der Dualen Softwareentwicklung in Studenten-Projekten

Die Idee der Dualen Softwareentwicklung in ihrer sequentiellen Variante eignet sich hervorragend für Studenten-Projekte. In der frühen Phase werden die Vision und die Ziele des Systems definiert sowie eine erste Architektur erstellt. In der zweiten Phase werden zyklisch mehrere Sprints durchlaufen, an deren Anfang stets die Aufstellung der Requirements und die Systemanalyse für die im Sprint zu entwickelnde Funktionalität steht. Daraufhin werden diese Features von den Gruppenmitgliedern umgesetzt und am Ende des Sprints ein Review durchgeführt, um die Ergebnisse, aber auch das Vorgehen selbst, zu bewerten und Verbesserungspotential aufzuzeigen.

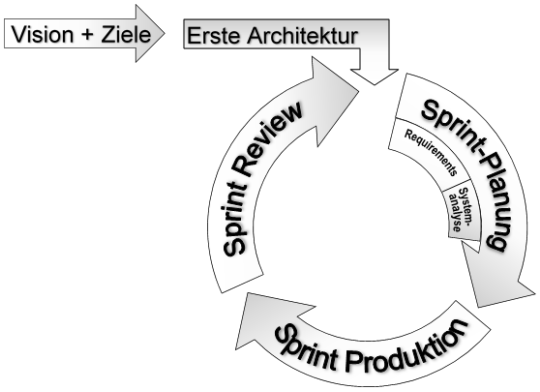

Abbildung 1: Gewähltes Vorgehensmodell

Dieses Vorgehen mag zunächst etwas an Scrum erinnern, da bei Scrum oft von einem sogenannten „Sprint 0“ die Rede ist, in welchem ebenfalls die Ziele des Systems und eine erste Architektur definiert werden. Dieser vorbereitende Sprint findet jedoch im offiziellen Scrum-Guide keine Erwähnung und auch bekannte Vertreter der Scrum-Philosophie wie Mike Cohn[2] zweifeln die Sinnhaftigkeit dieses initialen Sprints an. Eine erste Architektur könne auch im ersten produktiven Sprint entstehen. Werden wichtige Entscheidungen schon vorab benötigt, so solle ein „Projekt vor dem Projekt“ eingeführt werden [2].

[1] Meyer, B. (2014). Dual Development. In B. Meyer, Agile! (S. 74-75). Zürich: Springer International Publishing.

[2] Cohn, M. (05.03.13). Using Scrum on an Analysis Project: https://www.mountaingoatsoftware.com/blog/using-scrum-on-an-analysis-project, Zugriff am: 10.11.15

[2] Mike Cohn ist zertifizierter Scrum-Trainer und Mitbegründer der Scrum Alliance sowie der Agile Alliance.