Thomas Ecker, Michael Wesinger

Stärken und Schwächen agiler Softwareentwicklungsmodelle

Diplomarbeit

Examicus

Stärken und Schwächen agiler Softwareentwicklungsmodelle

Diplomarbeit zur Erlangung des akademischen Grades
eines Magister der Sozial- und Wirtschaftswissenschaften

Vorgelegt von Thomas Ecker und Michael Wesinger
Wels/Schlüßlberg, im Februar 2005

Institut für Wirtschafsinformatik - Software Engineering
Johannes Kepler Universität Linz

Kurzfassung

Über die leichtgewichtigen bzw. agilen Softwareentwicklungsmodelle wurde in den letzten Jahren verstärkt diskutiert. Das zeigt sich durch die vermehrte Anzahl an den in Buchform erschienen Modellen bzw. an vielen veröffentlichten Fachartikeln.

Diese Modelle bilden den Rahmen, um Software effizient und rasch entwickeln zu können. Im Unterschied zu den Schwergewichtigen versuchen sie, Software mit weniger Vorgaben und Richtlinien zu erstellen und rasch auf Kundenwünsche zu reagieren. Sie sind weniger formal beschrieben als die Schwergewichtigen und lassen den Softwareentwicklern einen größeren Freiraum, u. a. beim Entwurf. Doch nicht alle agilen Modelle sind für jede Art von Projekt gleich gut geeignet. Einige Modelle eignen sich auf Grund ihrer reglementierten Richtlinien nur für kleine bis mittelgroße Projekte, die anderen sind weniger formal beschrieben und auch in größeren Projekten anwendbar.

Die in dieser Diplomarbeit beschriebenen Modelle erschienen alle in Buchform, doch auf verschiedene Kriterien wie Stärken und vor allem Schwächen gingen die Modellentwickler nicht detailliert ein. In dieser Arbeit werden acht agile Modelle beschrieben und anhand verschiedener Kriterien verglichen, deren Einsatzbereiche eruiert und Stärken und Schwächen aufgezeigt. Darüber hinaus wird ein Fragebogen erstellt, der Hinweise darüber geben soll, ob bzw. welche agilen Modelle in der Praxis eingesetzt werden und welche Stärken und Schwächen bei agiler Entwicklung auftreten können. Die Durchführung der Befragung ist nicht mehr Teil dieser Diplomarbeit, sie kann jedoch im Rahmen einer darauf aufbauenden Arbeit erfolgen.

Abstract

Agile software development approaches have become more popular during the last few years, shown by the increased number of methods appeared in book form or discussed in many subject articles.

They form the framework to be able to develop software efficiently and quickly. In addition to this, they try to ensure that the software meets customer's changing needs. In comparison to traditional development methods, agile development is less document-centric and more code-oriented. Further, it is adaptive rather than predictive and people-oriented rather than process-oriented. But all agile methods aren't equally well suitable for every kind of project. Due to more exact guidelines, some methods are only suitable for small till medium-sized projects, others are described less formally and therefore also applicable to greater projects.

All the methods described in this paper appeared in book form but the method developers didn't go into different criteria like strengths and primarily weaknesses. The description and the comparisons of the agile methods, the fields of application as well as the strengths and weaknesses derived from it are the main part of the work. Further, we prepare a questionnaire which examines the hypotheses put forward in this work. This questionnaire gives information about which agile models are used in the practice and declares which strengths and weaknesses appear at an agile development. The evaluation is no more part of the work. It can, however, be evaluated in the context of a master thesis building on it.

Danksagung

Wir möchten unserem Betreuer, Herrn Dr. Joachim H. Fröhlich, für diese interessante Aufgabenstellung und seine engagierte Betreuung danken. Unser Dank geht an dieser Stelle auch an unsere Eltern, die uns das Studium der Wirtschaftsinformatik ermöglicht und uns in jeder Hinsicht dabei unterstützt haben. Ebenso bedanken wir uns bei unseren Freunden und Kollegen, die durch kritische Anmerkungen und fachliche Diskussionen uns immer wieder neue Anregungen und Motivation gegeben haben.

Inhaltsverzeichnis

Kapitelaufteilung

Michael Wesinger erklärt sich verantwortlich für die folgenden Kapitel:

- Kapitel 1
- Kapitel 2
- Kapitel 4.1, 4.2, 4.3
- Kapitel 6.6, 6.7, 6.8, 6.9, 6.10
- Kapitel 7

Thomas Ecker erklärt sich verantwortlich für die folgenden Kapitel:

- Kapitel 3
- Kapitel 4.4, 4.5
- Kapitel 5
- Kapitel 6.1, 6.2, 6.3, 6.4, 6.5
- Kapitel 8

1 Einleitung

Die leichtgewichtigen Softwareentwicklungsmodelle werden von ihren Entwicklern und zahlreichen Befürwortern als Lösungen für Softwareentwicklungsprobleme gesehen, weil sie im Gegensatz zu den schwergewichtigen Modellen durch mehr Flexibilität und mehr Kundennähe gekennzeichnet sind. Ihnen eilt der Ruf voraus, dort zu helfen, wo schwergewichtige Modelle scheitern. Mitunter werden sie als Allheilmittel gesehen. Sie gewähren den Projektbeteiligten ein hohes Maß an Gestaltungsfreiheit, fordern bzw. fördern ein regelmäßiges Überprüfen und Anpassen des Vorgehens und bauen auf ähnliche Grundsätze auf. Der bekannteste Vertreter ist „eXtreme Programming“. Dieses Modell war wichtig für die Weiterentwicklung anderer agiler Modelle. Das Ziel ist, mit geringem finanziellen Einsatz so schnell als möglich auf Änderungen zu reagieren, Erfolge früh sichtbar zu machen und die Lebensdauer der Software zu verlängern. Diese Modelle kommen ursprünglich aus den USA, haben jedoch auch bei den IT - Verantwortlichen in Europa ihre Anhänger.

Im Dezember 2001 hat das *Cutter IT Consortium*, das ist ein Zusammenschluss von amerikanischen IT-Beratern, 200 Softwareunternehmen mit mindestens 25 Mitarbeitern nach dem Einsatz von leichtgewichtigen Modellen befragt. Dem Umfrageergebnis zufolge hatten mehr als 48 % der Firmen die Absicht, bis zum Jahre 2003 leichtgewichtige Modelle in mindestens einem Projekt einzusetzen. Das zeigt, wie groß die Euphorie bezüglich des Einsatzes von leichtgewichtigen Modellen im Jahre 2001 war. Weiters zeigte sich die höhere Kundenzufriedenheit. Wurde ein Projekt leichtgewichtig abgewickelt, so waren 77 % der Kunden mit dem Ergebnis eher zufrieden. Bei Teams, die schwergewichtig vorgingen, waren es nur 63 %. Dies bedeutet aber auch, dass noch 23 % der Kunden mit dem leichtgewichtigen Modell unzufrieden waren [Coldewey 2002a S. 237]. Der Grund der Unzufriedenheit geht aus der Studie nicht hervor. Man darf trotz aller Euphorie um die neuen Modelle nicht vergessen, dass auch sie Problemen und Risiken ausgesetzt sind.

1.1 Motivation

Das wissenschaftliche Computermagazin *ACM* veröffentlichte eine Studie[1], welche die unterschiedlichen Prämissen theoretischer und praxisorientierter Forschung

[1] Aus “Communications of the ACM”, September 2004, Volume 47, Number 9, S. 91-94; “ACM” ist die Bezeichnung für “Association for Computing Machinery”.

bezüglich der IT Analyse, Planung und Implementierung im Zeitraum von 1970 bis 2002 zeigt. Laut dieser Studie, dessen Angaben auf den Quellen von 13 wissenschaftlichen Journalen basieren, klafft eine große Lücke zwischen diesen beiden Forschungsrichtungen. Hat man auf theoretischer Basis vor 1990 hauptsächlich auf Planungsaktivitäten Wert gelegt, so wurde in den folgenden Jahren der Fokus auf Anforderungen, Systemspezifikationen und Systemanalysen erweitert. In der Praxis standen vor 1990 Methoden, Werkzeuge, Systemspezifikationen und Systemanalysen im Vordergrund, danach jedoch hauptsächlich Methoden und Werkzeuge. Das Ziel ist, die Qualität von Softwareprodukten zu verbessern und die Fertigstellung zu beschleunigen. Um dies zu erreichen, ist es notwendig, die theoretischen und praxisorientierten Ansätze zu kombinieren und den Schwerpunkt sowohl auf interne Kommunikation als auch auf Kommunikation mit dem Kunden zu setzen. Dies hat uns veranlasst, die aus der Praxis stammenden leichtgewichtigen Softwareentwicklungsmodelle zu analysieren, da die Befürworter dieser Modelle überzeugt sind, eine erhöhte Softwarequalität und eine schnellere Entwicklung u. a. durch die Aspekte der Zusammenarbeit und Kommunikation zu erhalten. Die Modelle werden in der Literatur hinsichtlich ihrer historischen Entwicklung und ihrer Merkmale ausreichend beschrieben. Auf die Unterschiede, Stärken und Schwächen sowie auf die Risiken der leichtgewichtigen Modelle wurde jedoch noch nicht detailliert eingegangen. Ebenso sind uns keine Fragebögen bekannt, die Hinweise darüber geben, ob Softwareentwicklungsunternehmen diese Modelle einsetzen.

1.2 Aufgabenstellung und Zielsetzung

Wir möchten der zunehmenden Bedeutung der leichtgewichtigen Modelle Rechnung tragen und sie aus neutraler Sicht beschreiben. Diese Modelle sollen anhand typischer Merkmale charakterisiert und untereinander - bzw. auch mit typisch schwergewichtigen Modellen – verglichen werden, um daraus Stärken und Schwächen ableiten zu können. Weiters möchten wir herausfinden, ob es „das“ leichtgewichtige Modell gibt, das in allen Projekten, unabhängig der Größe eingesetzt werden kann, das alle Phasen der Softwareentwicklung abdeckt und mit denen allen Schwächen und Risiken entgegengetreten werden kann. Dies wäre ein Modell, das man Softwareentwicklungsunternehmen empfehlen könnte. Durch eine neutrale Sicht sollen die Stärken und Schwächen besser erkennbar und die Modelle vergleichbar werden.

Um die Aussagekraft der Literaturarbeit zu erhöhen, schaffen wir mit einem Fragebogen die Voraussetzungen, die in unserer Arbeit aufgestellten Hypothesen untersuchen zu können. Der Fragebogen soll so konzipiert sein, dass die

Untersuchung in allen Softwareentwicklungsfirmen, unabhängig der Unternehmensgröße und Branche, durchgeführt werden kann. Der Grund einer unternehmens- und branchenunabhängigen Untersuchung liegt darin, dass Erkenntnisse über den Vorgang der Softwareentwicklung gewonnen werden sollen. Umfassende Erkenntnisse über den praktischen Einsatz von Softwareentwicklungsmodellen erhält man, wenn keine unternehmensspezifische Festlegung hinsichtlich Größe und Branche vorgenommen wird. Die Auswertung des Fragebogens wird von uns nicht durchgeführt, dies könnte jedoch im Rahmen einer weiterführenden Arbeit erfolgen.

1.3 Aufbau der Arbeit

In Kapitel 2 „Grundlagen“ nehmen wir einige Begriffsdefinitionen vor, erörtern die Unterschiede zwischen leicht- und schwergewichtigen Modellen und beschreiben die für die Softwareentwicklung notwendigen Phasen und Prozesse, da wir einerseits die Modelle anhand von Phasen vergleichen und andererseits einen einheitlichen Sprachgebrauch diesbezüglich liefern wollen. In Kapitel 3 „Agile Softwareentwicklungsmodelle“ gehen wir auf die leichtgewichtigen Modelle ein. Wir analysieren das „Manifest der agilen Softwareentwicklung“, erläutern die Eigenschaften und gehen auf die Einsatzbereiche ein. Kapitel 4 „Ausgewählte Vertreter agiler Modelle“ beschreibt die typischen Vertreter der leichtgewichtigen Modelle. Kapitel 5 „Vergleich der Vertreter agiler Modelle“ stellt die Modelle zum Vergleich gegenüber und behandelt die Unterschiede anhand verschiedener Kriterien. Weiters analysieren wir, inwieweit die leichtgewichtigen Softwareentwicklungsmodelle die im zweiten Kapitel beschriebenen Phasen abdecken. Kapitel 6 geht auf die Stärken sowie auf die Schwächen der leichtgewichtigen Softwareentwicklungsmodelle ein. In Kapitel 7 „Fragebogen“ beschreiben wir die Zielsetzung und die Strukturierung des Fragebogens. Im letzten Kapitel „Zusammenfassung und Ausblick“ fassen wir die gewonnenen Erkenntnisse der Diplomarbeit zusammen und liefern einen kurzen Ausblick.

2 Grundlagen

Ziel dieses Kapitels ist die Einführung in den komplexen Bereich der Softwareentwicklungsmodelle und der Erwerb von grundlegendem Wissen, das für das Verständnis der weiteren Kapitel notwendig ist. Wir nehmen eine Begriffsdefinition vor, führen weiters in den Bereich der Softwareentwicklungsmodelle ein und eruieren die Unterschiede zwischen leicht- und schwergewichtigen Modellen. Darüber hinaus beschreiben wir die Phasen der Softwareentwicklung inklusive Tätigkeiten und Ergebnissen. Die von uns beschriebenen leichtgewichtigen Modelle werden im Rahmen dieser Arbeit anhand der Phasen gegenübergestellt und analysiert, inwieweit sie diese berücksichtigen. Abschließend beschreiben wir noch Prozesse, die bei der Softwareentwicklung unterstützend eingesetzt werden.

2.1 Begriffsdefinitionen

Wir definieren hier jene Begriffe, die für das Verständnis der folgenden Kapitel notwendig sind. In der Literatur werden hierfür oft keine einheitlichen Ausdrücke verwendet. Damit wir in dieser Arbeit ein Begriffschaos vermeiden, erläutern wir an dieser Stelle die wesentlichen Ausdrücke und Bedeutungen. Eine umfassende Begriffsdefinition führen wir im „Glossar“ an.

Aktivität: Eine in sich abgeschlossene Folge von Tätigkeiten, deren Unterbrechung kein sinnvolles Ergebnis liefert [Heinrich et al. 2004 S. 49].

Arbeitsergebnis: Ein Produkt, dass bei einer Aktivität entstanden ist.

Basislinie (engl. Baseline): Eine überprüfte und genehmigte Spezifikation [OOSE 2003]. Im Kontext des Konfigurationsmanagements stellt sie die Ausgangskonfiguration eines Produkts dar.

Entwurf: Die Phase eines Projekts, mit dem Zweck der Überführung der Grundkonzeption in ein logisches Modell des Sollzustands [Heinrich 2001a S. 72].

Feature: Ein funktionales Merkmal bzw. eine unbedingte notwendige Eigenschaft eines Softwareprodukts.

Inkrement: Darunter versteht man die Erweiterung eines Produktes. Ein Inkrement ist gewöhnlich gekennzeichnet durch die Differenz zwischen zwei Versionen eines in Entwicklung befindlichen Systems. Auch das Teilergebnis, das innerhalb einer Iteration entstanden ist, wird Inkrement genannt [OOSE 2003].

Konfigurationseinheit: Versionisierter und verwalteter Teil einer Software. In der Regel ist ein Entwickler für eine Konfigurationseinheit verantwortlich [OOSE 2003].

Konfigurationsidentifikation: Die Basis für die nachfolgende Kontrolle der Softwarekonfiguration. Es ist der Prozess, wobei ein System in einzelne identifizierbare Komponenten für den Zweck des Konfigurationsmanagement getrennt wird (*Konfigurationseinheiten)* [Koskela 2003 S. 11].

Kritikalität: Bezeichnet die möglichen negativen Folgen für die Organisation, für die Software entwickelt wird, wenn die gelieferte Software fehlerhaft ist.

Methode: Ein auf einem System von Regeln aufbauendes Problemlösungsverfahren (z.B. Algorithmus) [Heinrich et al. 2004 S. 427].

Modell: Synonym für Verfahren bzw. Ansatz im Kontext der Softwareentwicklung. Wir unterscheiden in unserer Arbeit zwischen leichgewichtigen (agilen) und schwergewichtigen Softwareentwicklungsmodellen.

Phase: Eine nach zeitlichen und logischen Gesichtspunkten gebildete Teilmenge eines Projekts [Heinrich 2001a S. 72].

Praktik: Tätigkeit, die von den Beteiligten beim Softwareentwicklungsvorgang durchgeführt wird.

Prozess: Menge von Operationen, die durch eine Eingabe in ein System, interne Funktionen im System und einem Ergebnis aus dem System beschrieben wird [Heinrich et al. 2004 S. 534].

Refactoring: Restrukturierung des Programms durch den Softwareentwickler, ohne Funktionen zu ändern.

Release: Eine aus Anwendersicht lauffähige Version des Systems mit vorher genau definierter Funktionalität.

Rolle: Ein System von Verhaltensregeln, das an den Inhaber einer Position gerichtet ist [Heinrich 2001b S. 290].

Softwareentwicklungskultur: Die in einer Organisation üblichen Softwareentwicklungsrichlinien.

Softwareentwicklungslebenszyklus (engl. Software Development Life Cycle): Zusammenfassender Begriff für die Phasen der Softwareentwicklung.