Agile Softwareentwicklung von Anfang an

Wie flexibel soll's denn sein?

André Schütz

LinuxTag 2014 / Berlin

Version 1.0

Andre Schütz (andre@wegtam.com)

Die Agile Softwareentwicklung hat sich in den letzten 25 Jahren fortwährend weiterentwickelt und zahlreiche Methoden und Prozessgrundlagen geschaffen. Neben großen Unternehmen können insbesondere kleinere Firmen von den Vorteilen profitieren. Agile Softwareentwicklungsmethoden können die Produktivität positiv beeinflussen und den Mitarbeitern neue Wege der Zusammenarbeit und Variabilität aufzeigen. Darüber hinaus müssen aber auch verschiedene Herausforderungen berücksichtigt werden, welche sich für die beteiligten Manager, Projektleiter und Softwareentwickler ergeben.

Bei der Durchführung von Projekten, welche einen proprietären und gleichzeitig auch Open Source zugewandten Ansatz verfolgen, können agile Softwareentwicklungsmethoden dazu beitragen, die nötige Flexibilität beizubehalten und ein Höchstmaß an Arbeitserleichterung zu generieren. Dennoch muss man auch den zusätzlichen Aufwand berücksichtigen, der durch die Einführung von agilen Prozessen entstehen kann. Es muss dann individuell abgewogen werden, welche Prozesse und Methoden in welchen Situationen und mit welchen Ressourcen sinnvoll sind bzw. bis zu welchem Maße eingesetzt werden.

Unter diesen Gesichtspunkten kann sich von Projekt zu Projekt und abhängig von den Fähigkeiten der vorhandenen Personen ein Mix aus traditioneller Individualprogrammierung und Paarprogrammierung als äußerst effektiv erweisen. In der Projektplanung und Projektverwaltung kann es von Vorteil sein, grundsätzliche Ansätze aus Scrum oder Kanban einfließen zu lassen. Die Einführung eines Continuous Integration Werkzeuges in Verbindung mit diversen Testframeworks (Unittests, Akzeptanztests, etc), kann zu einer erheblichen Erleichterung im Deployment Prozess führen und die kontinuierliche Weiterentwicklung vereinfachen.

Inhaltsverzeichnis

1. Agile Softwareentwicklung

Die Softwareentwicklung setzt sich aus verschiedenen Teilgebieten zusammen, welche Prinzipien, Methoden und Werkzeuge für die ingenieurmäßige Entwicklung und Anwendung von Softwaresystemen beinhalten [Balz2001].

Neben den etablierten klassischen Softwareentwicklungsprozessen entwickelten sich in den letzten 25 Jahren auch eine Reihe von agilen Vorgehensweisen und -modellen, welche neue Ansichten, Methoden und Prozessgrundlagen hervorgebracht haben.

Geschichte der agilen Softwareentwicklung

Seit Anfang der 90er Jahre des letzten Jahrtausends konnten sich eine Vielzahl von Ansichten und Ideen zur agilen Softwareentwicklung fortwährend weiterentwickeln und haben kontinuierlich an Popularität gewonnen.

Ein entscheidender Meilenstein bildete die Veröffentlichung des **Agilen Manifest**, welches im Februar 2001 von 17 Erstunterzeichnern in Utah vorgestellt worden ist. Dieses Fundament der agilen Softwareentwicklung beinhaltet 4 Werte und weitere 12 Prinzipien.[AM2001]

Klassische Vorgehensmodelle

Die klassischen Vorgehensmodelle beschreiben einen schrittweisen Weg von der Problemstellung zur Lösung und befolgen dabei oftmals klar definierte und **bindende** Phasen. Diese sollen die Entwicklung von Software planbar, kontrollierbar und vorhersagbar machen und weitere Unterstützungen mittels Prozessoptimierungen und Zertifizierungen liefern.[CB2010]

Zu den bekanntesten Vertretern dieser Vorgehensmodelle gehören u.a. das Wasserfallmodell, das V-Modell und das Spiralmodell.

Kernaussagen der agilen Softwareentwicklung

Im Gegensatz zu den klassischen Modellen, versucht die agile Softwareentwicklung den gesamten Entwicklungsprozess **flexibler** und

schlanker zu gestalten. Durch die Minimierung des bürokratischen Aufwandes, weniger Regeln und einem häufigen iterativen Vorgehen, sollen die technischen und sozialen Probleme der Softwareentwicklung in den Vordergrund gestellt und die zu erreichenden Ziele stärker betont werden. [AgilS2014]

Das Fundament der agilen Softwareentwicklung basiert auf **4 Werten**, bei denen jene Aussagen auf der linken Seite einer größeren Bedeutung zugemessen werden als denen auf der rechten Seite.[AMDE2014]

1. **Individuen und Interaktionen** mehr als Prozesse und Werkzeuge
2. **Funktionierende Software** mehr als umfassende Dokumentation
3. **Zusammenarbeit mit dem Kunden** mehr als Vertragsverhandlung
4. **Reagieren auf Veränderungen** mehr als das Befolgen eines Plans

Aus diesen Werten haben sich **12 Prinzipien**[1] entwickelt, welche Handlungsgrundsätze für die **Methoden** der agilen Softwareentwicklung darstellen.

In der Gesamtheit münden die Werte, Prinzipien und Methoden in den agilen **Prozessen**, welche Gemeinsamkeiten aus den folgenden Punkten aufweisen:

- Schlanke Entwurfsphase
- Iteratives Vorgehen in kurzen Abständen
- Frühzeitig und regelmäßig ausführbare Software erstellen
- Höchstmaß an Flexibilität und Anpassbarkeit

2. Agile Softwareentwicklungsmethoden

Nach einer aktuellen Studie [VOne2014] von Version One[2], gaben 90% der befragten Personen an, dass ihnen agile Methoden bei der Bewältigung von sich verändernden Prozessen geholfen haben. Die Erfahrungen nehmen immer

1 Prinzipien hinter dem Agilen Manifest [AMPrinzi2001]
2 2013/2014 mit 3500 befragten Individuen

mehr zu und in rund 50% der durchgeführten Projekte wurden agile Softwareentwicklungsmethoden eingesetzt.

Story-Cards

Die Nutzung von Story-Cards in agilen Prozessen soll die Beschreibung von Kundenanforderungen mit Hilfe von einfachen Kurzbeschreibungen erleichtern. [SA2011]

Der Kunde kann das gewünschte Verhalten mit Hilfe seiner eigenen Sprache beschreiben und jede Funktion des Systems mit einer einzelnen Story-Card definieren.

Paarprogrammierung

Die Paarprogrammierung ist ein wichtiger Bestandteil des Extreme Programming und wird von jeweils 2 Entwicklern an einem Rechner durchgeführt. Dabei schreibt einer abwechselnd den Code („Driver“), während der andere ein Codereview durchführt („Navigator“).[KBeck2004]

Eine Kritik an der Paarprogrammierung ist die vermeintliche Verringerung der Produktivität, wenn 2 Entwickler an einem Rechner gleichzeitig agieren. Dabei wird oftmals irrtümlicherweise die Tippgeschwindigkeit[CR2009] als Indikator heran gezogen.

Die Vorteile der Paarprogrammierung werden wie folgt angesehen:[ACock2001]

- Einfache Möglichkeit das Wissen im Team zu verteilen
- Höhere Codequalität und weniger Fehler
- Stärkere Fokussierung auf die momentane Aufgabe
- Teambildung und Motivation
- Schnellere Lösung von auftretenden Fragen / Problemen

Test Driven Development (TDD)

Beim Test Driven Development geht es um das Erstellen von Testfällen vor der

eigentlichen Implementierung des Produktionscodes. Dadurch entsteht ein Fehler, welcher durch den erstellten Code behoben wird und den Test lauffähig macht. Diese Vorgehensweise schafft eine extrem hohe Testabdeckung und verringert die Komplexität der zu testenden Komponenten.[KBeck2002]

Oftmals unterscheidet man das Testen im Kleinen durch Unit-Tests und das Testen im Großen durch System- und Akzeptanztests.

Ständiges Refactoring

Beim ständigen Refactoring geht es um das fortwährende manuelle oder automatisierte optimieren von Quelltexten.[MF2000] Dabei soll die Wartbarkeit, Lesbarkeit, Verständlichkeit und Erweiterbarkeit verbessert und komplexe Strukturen möglichst elegant vereinfacht werden.

Die kontinuierliche Durchführung des Refactoring innerhalb eines agilen Prozesses kann dazu beitragen, unerwünschte Seiteneffekte und verborgene Fehler schneller zu finden.

3. Agile Softwareentwicklungsprozesse

Die am meisten genutzten Prozesse im Jahr 2013 waren Scrum (55%), Custom (10%), Scrumban (7%), Kanban (5%) oder das Feature Driven Development (2%). Die Entscheidung über den einzusetzenden Prozess wurde in erster Linie durch das Management (61%), gefolgt vom DEV-Staff (17%), den Executives (14%) und den Consultants (8%) getroffen.[VOne2014]

Extreme Programming (XP)

Das Extreme Programming (XP) stellt die zu lösende Arbeitsaufgabe in den Vordergrund und versucht durch kurze Entwicklungszyklen, User-Stories und der Lieferung einer einsatzfähigen Software am Ende einer jeden Iteration, die anfänglichen Unsicherheiten bei der Definition der Anforderungen zu verringern.[CR2009]

Diese Herangehensweise schafft neue Möglichkeiten aber auch

Herausforderungen für die Kunden und das Projektteam.[TDM2003]

Kunden können jederzeit **steuernd** eingreifen und neue Features frühzeitig bewerten, müssen sich aber auch stärker während der gesamten Entwicklungszeit **einbringen** und auf eine formale vollständige Projektdefinition und -planung verzichten.

Das Projektteam auf der anderen Seite kann sich auf die wirklich **relevanten** Anforderungen konzentrieren, welche einen wirklichen **Mehrwert** für den Kunden liefern, muss sich aber andererseits auch erst einmal mit dem Kunden einigen, damit die Vorteile der agilen Entwicklung nicht durch die bürokratischen Vorgaben und Richtlinien von vornherein zum Scheitern verurteilt sind.

Agile Unified Process

Der Agile Unified Process (AUP) übernimmt viele agile Techniken des Extreme Programming (XP) und erstellt auch einige der formalen Artefakte aus dem Rational Unified Process (RUP). Daher wird AUP oft zwischen diesen beiden Prozessmodellen eingeordnet.

Für das Management eines Unternehmens kann AUP insbesondere durch die zusätzlichen Artefakte von Interesse sein, wenn XP als zu wenig formal angesehen wird. Entwickler bevorzugen die geringere Anzahl von Artefakten im Vergleich zum RUP Modell.[SA2006]

Crystal

Die Methoden der Crystal Familie werden mit Farben gekennzeichnet und basieren u.a. auf den Prinzipien des passiven Wissenstransfers, einem fokussierten Arbeiten, laufenden Verbesserungen und häufigen Releases, automatisierten Tests und einem kundigen Nutzer.[AC2005]

Die Farben in den Methoden spiegeln grundsätzlich die Anzahl der beteiligten Personen wider und werden aufsteigend mit Crystal Clear, Crystal Yellow, Crystal Orange, Crystal Orange Web, Crystal Red, Crystal Magenta und Crystal Blue bezeichnet. Im Fall von Crystal Clear werden 2-6 Personen als optimal

angegeben.

Darüber hinaus werden noch Stufen des Risikos definiert (Möglichkeit des Scheiterns) und hinsichtlich der Gefährdung von Kundenzufriedenheit, dem Verlust von Geld, einem Imageschaden und der Gefahr von Leben, verglichen. Zusammen mit der Anzahl der beteiligten Personen sollen so die jeweils zu den Projektdefinitionen passenden Regelsätze gewählt werden.[AC2004]

Scrum

Scrum versucht die Komplexität von Softwareprojekten über die Säulen Transparenz, Überprüfung und Anpassung besser handhabbar zu machen. Es basiert auf der Erkenntnis, dass ein ständig verfügbares Feedback den Erfolg eines Projektes sichern kann.[KS]

Grundlegende Praktiken sind die Ausformulierung von Funktionalitäten, Daily Scrum, wiederholte Intervalle von 2-4 Wochen (Sprints) und die Lieferung von fertigen Funktionalitäten und lauffähigen Programmen am Ende eines jeden Sprints.

Kanban

Kanban stammt aus dem Japanischen und bedeutet so viel wie Signalkarte. Es wird versucht, die Anzahl der parallel durchgeführten Arbeiten zu reduzieren und dabei schnellere Durchlaufzeiten zu erreichen. Grundlegende Praktiken sind die Visualisierung des Arbeitsflusses, die Begrenzung angefangener Arbeiten, explizite Regeln, Leadership auf allen Ebenen und Vereinfachungen durch Modelle.[DA2010]

Das Ziel ist die Vermeidung und Verringerung von Verschwendung und die Gewährleistung eines gleichmäßigen Flusses in der Arbeit.

Feature Driven Development (FDD)

Feature Driven Development wurde 1997 von Jeff De Luca für ein zeitkritisches Projekt entwickelt und stellt die Features eines Projektes in den Mittelpunkt.

Dabei soll jedes Feature einen Mehrwert für den Kunden liefern und über die Feature-Liste abgearbeitet werden.[NB1998]

Adaptive Software Development (ASD)

Das Adaptive Software Development (ASD) basiert auf dem Rapid Application Development (RAD) und führt alle 4 Wochen eine Prüfung der gerade aktuellen Programmversion hinsichtlich ihrer Neuerungen und Verbesserungen in Bezug auf die vorherigen Versionen durch.[AD]

Dabei werden kontinuierlich die Phasen Spekulieren, Zusammenarbeiten und Lernen wiederholt durchlaufen und in Kooperation mit dem Kunden erarbeitet.

Behavior Driven Development (BDD)

Bei der verhaltensgetriebenen Softwareentwicklung werden in der Anforderungsanalyse die Aufgaben, Ziele und erwarteten Ergebnisse schriftlich festgehalten. Dies ermöglicht automatisierte Testverfahren, welche eine korrekte Implementierung verifizieren können.[DN2006]

Im Laufe des Prozesses werden die ursprünglichen Mock-Objekte (Frameworks u.a. Mockito, PowerMock (Java); rr (Ruby); Mock (Python), Rhino Mocks (C#/.net)), ScalaMock, welche für die Automatisierung der Fallbeispiele erstellt worden sind, durch die implementierten Softwareteile ersetzt.

Agiles Testen

Das agile Testen soll den Entwicklern helfend zur Seite stehen, permanent ein schnelleres Feedback liefern, einen hohen Automatisierungsgrad erreichen, Grenzen zwischen Testern und Entwicklern auflösen und die agile Softwareentwicklung optimal unterstützen.[AT2013]

4. Unterstützende Maßnahmen und Werkzeuge

Bei der täglichen Arbeit in agilen Softwareprojekten gibt es eine Vielzahl von Maßnahmen und Tools, welche eine Unterstützung und Automatisierung für die

gewählten Methoden und Prozesse bieten können.

Verwaltung der Projekte

Für die Durchführung von agilen Projekten wird eine Vielzahl von allgemeinen oder spezialisierten Software-Tools eingesetzt. Die am meisten genutzten Programme in diesem Bereich sind [VOne2014] (Mehrfachangaben waren möglich):

1. Excel (66%)
2. Microsoft Project (48%)
3. VersionOne (41%)
4. JIRA / Greenhopper (36%)
5. Microsoft TFS und HP Quality Center (je 26%)
6. Bugzilla (21%)
7. Google Docs (20%)

Weitere Produkte waren: Vendor Y, Rational, Pivotal Tracker, IBM Rational, etc.

Continuous Integration

Systeme aus dem Bereich des Continuous Integration stellen eine dauernde Integration des gerade bearbeitenden Codes mit dem Gesamtsystem her. Die häufigste und bekannteste Form ist das automatische Ausführen von Tests, welche bei der Zusammenführung von Komponenten erfolgreich durchlaufen werden müssen.

Weitere Grundsätze einer erfolgreichen Continuous Integration sind u.a. eine gemeinsame Codebasis, eine automatisierte Übersetzung, häufige und kleine Integrationen und ein automatisiertes Reporting.[MF2006]

Einige Software-Vertreter in diesem Bereich sind: TeamCity von JetBrains[1], Continuum[2] als Subprojekt von Apache Maven, Bamboo[3] von Atlassian, das

1 http://www.jetbrains.com/
2 http://continuum.apache.org/
3 https://www.atlassian.com/software/bamboo

Bitten-Plugin[1] für Trac, CruiseControl[2], Apache Gump[3] oder der Team Foundation Server von Microsoft.

Build-Tools

Programme für die Build-Automatisierung liegen nach Bug-Trackern und noch vor Wikis und Testwerkzeugen auf dem zweiten Rang der am häufigsten eingesetzten Software-Werkzeuge in der agilen Softwareentwicklung. [VOne2014] Sie werden für die Kompilierung von Quellcode, der Ausführung von Tests, dem automatischen Einspielen von Code in das Produktionssystem oder der Erstellung von Dokumentation eingesetzt.

Einige bekannte Vertreter nach Programmiersprache sind:

Java:	Ant, Continuum, Maven, Leiningen (Clojure), SBT (Scala)
Cross-Language:	Bamboo, CMake, CruiseControl, Jenkins, make, Visual Build
Ruby:	Rake

Deployment

Die automatisierte Auslieferung von Software kann dazu beitragen, die Installation, Wartung und Konfiguration von Systemen zu erleichtern. Speziell zusammengestellte und auf den jeweiligen Anwendungszweck angepasste Konfigurationen erleichtern das kontinuierliche und reibungslose Ausliefern von Programmen und Softwareumgebungen.

Vertreter für das kontinuierliche Deployment sind u.a. Teamcity, Chef, Puppet oder Vagrant.

5. Herausforderungen

Damit Methoden und Prozesse der agilen Softwareentwicklung erfolgreich in Unternehmen und Teams eingesetzt und durchgeführt werden können, müssen verschiedene Herausforderungen gemeistert werden.

1 http://bitten.edgewall.org/
2 http://cruisecontrol.sourceforge.net/
3 https://gump.apache.org/

Die größten Bedenken bei der Einführung von agilen Methoden und Prozessen lagen im Jahr 2013 [VOne2014]:

- Mangel an Planung im vornherein
- Verlust an Managementkontrolle
- Fehlende Dokumentation
- Fehlende Vorhersagbarkeit
- Fehlende Disziplin der Entwickler

Offenheit und Mitwirken des Kunden

Die agile Softwareentwicklung bedarf der kontinuierlichen Mitarbeit und Beteiligung durch den Kunden [TNM2005]. Er muss in diesem agilen Umfeld mitarbeiten wollen und sich auf die Unsicherheit eines Projektes einlassen, welches keine stabilen Anforderungen definiert.

Auch wenn die ersten agilen Methoden und Prozesse schon seit etwa 25 Jahren existieren, bedarf es immer noch einer erheblichen Aufklärung über die Vorteile, Risiken und Herausforderungen, welche diese Ansätze mit sich bringen. Das man dabei gezielte Anforderungen an den Auftragnehmer aber auch an den Auftraggeber stellt, muss gezielt kommuniziert und eingefordert werden.

Ein häufiger Fall von nicht korrekt durchgeführter Beteiligung ist die fehlende Rückmeldung zu ausgelieferten Versionen, welche neue Features implementiert und frühere Fehler korrigiert haben. Dadurch ist eine schnelle Korrektur bei Fehlentwicklungen nicht gegeben und ein späteres Eingreifen in den bestehenden Code wird immer aufwändiger.

Akzeptanz und Loslassen des Management

Über Jahrzehnte wurde die Softwareentwicklung durch klassische Ansätze geprägt und von ihnen beherrscht. Dadurch haben sich bestimmte Grundsätze entwickelt, welche durch die agilen Ansätze teilweise in Frage gestellt werden. [VOne2014]

Das führt unweigerlich zu Problemen in der Bewertung und Betrachtung dieser „neuen“ Herangehensweisen und demzufolge auch zu Hindernissen, welche einem erfolgreichen agilen Ansatz im Wege stehen.

Möchte man agile Methoden und Prozesse in einer Unternehmung integrieren, muss man zunächst darauf achten, dass die Unternehmenskultur und das strategische und operative Personal diesen neuen Grundsätzen offen gegenüber stehen. Dadurch kann man generelle Ablehnungen von Anfang an verhindern und die Unterstützung und Akzeptanz im Management erhöhen.

Umsetzen der Methoden im Team

Agile Softwaremethoden und -prozesse bedürfen einer Menge Disziplin von den beteiligten Personen und ein gut organisiertes Teamwork.[ASD]

Damit diese Voraussetzungen gewinnbringend umgesetzt werden können, bedarf es des Commitment des Management und aller beteiligten Personen. Die Idee der agilen Softwareentwicklung muss gelebt werden, um mittels Reflektion der eingesetzten Prozesse und der tatsächlichen Umsetzung eine kontinuierliche Verbesserung zu erreichen.

Kreativität braucht Raum und Vertrauen

Klassische Methoden der Softwareentwicklung versuchen häufig den Entwicklungsprozess zu standardisieren und von den umsetzenden Personen zu abstrahieren.[CR2009] Der Versuch einer solchen Teilung wird oftmals der individuellen Einzigartigkeit und dem unterschiedlichen Leistungsvermögen der beteiligten Entwickler nicht gerecht und kann bei der Umsetzung von Projekten zu erheblichen Problemen führen.

Die Softwareentwicklung ist ein komplexer und je nach Aufgabenstellung und Anforderungsdefinition höchst anpassbarer Prozess, welcher in erster Linie von dem Leistungsvermögen und der Kreativität der involvierten Entwickler beeinflusst wird.

Daher sollte man den Entwicklungsprozess nicht zu sehr eingrenzen und der gestalterischen Kraft und dem Einfallsreichtum der beteiligten Entwickler

genügend Raum und Flexibilität lassen.

Zeit, Preis, Umfang - Projektdefinition

Bei der Durchführung von agilen Softwareprojekten geht man von instabilen Anforderungen aus, die sich im Laufe des Projektes ständig ändern und angepasst werden müssen. Die Kunden sollen das Produkt erhalten, welches sie wirklich brauchen und unnötige Features ohne Mehrwert gar nicht erst definieren.[TNM2005]

Daher ist es insbesondere bei agilen Projekten erforderlich, dass der Auftraggeber sich auf anpassbare Komponenten bei der Vertragsgestaltung einlassen muss, wenn er die Vorteile in Gänze nutzen möchte. Dies betrifft entweder die Anpassung der Zeit, des Preises oder des Umfanges der zu erbringenden Arbeit.

Ein Ansatz sieht vor, dass der Preis und die Zeit als eine fixe Komponente definiert, aber der Umfang der zu erbringenden Leistungen in einem kontrollierbaren Rahmen variieren kann.[TNM2005]

6. Wo bieten sich agile Methoden an?

Grundsätzlich spricht erst einmal nichts gegen den Einsatz und die Integration von agilen Methoden und Prozessen. Oftmals stehen einer erfolgreichen Einführung eher persönliche Interessen der beteiligten Personen, die Ablehnung von Veränderungen, Probleme bei der Vermittlung der neuen Ansätze in den Teams, fehlendes Wissen oder eine mangelhafte Kommunikation zwischen Teams, Management und Kunden im Wege. [VOne2014]

Projekte

Agile Methoden können dazu beitragen, die **Entwicklung** innerhalb eines Projektes zu **beschleunigen** und bei instabilen Anforderungen, mittels erster lauffähiger Versionen, ein klareres Bild von den kommenden Anforderungen zu erhalten.

Insbesondere bei Projekten, welche einen **forschenden** oder noch nicht klar umrissenen Charakter besitzen, können agile Softwareentwicklungsmethoden helfen, die wirklichen Anforderungen an das zu entwickelnde System besser zu verstehen und während der Implementierung klarer herauszustellen.

Team

Entwickler und Teams können per Definition von den agilen Ansätzen profitieren, da sich die agilen Methoden stärker an dem **Menschen** orientieren und versuchen, diesen in jeder auszuführenden Tätigkeit einzubeziehen. Dadurch wird von den beteiligten Personen eine stärkere **Selbstorganisation** und Übernahme von **Verantwortung** verlangt, welche bei richtiger Anwendung positiv auf die **Motivation** und die Entwicklung des Teamwork wirken können.

Damit diese Ansätze übernommen und in der täglichen Arbeit auch einfließen können, müssen die Ziele, Denkweisen und Fähigkeiten der beteiligten Personen berücksichtigt werden. Entwickler, welche noch nie mit agilen Methoden gearbeitet haben, sollten behutsam an diese herangeführt und vorbereitet werden.

Das **direkte** Gespräch ist ein wesentlicher Grundbaustein der agilen Entwicklung und muss nicht nur verlangt, sondern auch selbst vollzogen werden. Dadurch lässt sich das Wissen schneller, **transparenter** und kontinuierlich im Team und der Unternehmung verteilen.

Ein weiterer wesentlicher Punkt ist die Notwendigkeit von Generalisten. Nur, wenn sich auch die Spezialisten einbringen und über ihre Grenzen schauen, kann die Durchführung von agilen Projekten funktionieren. Es soll keine Barrieren im Team geben, welche auf den durchzuführenden Aufgaben basieren.

Darüber hinaus haben die **Fähigkeiten** der beteiligten Personen einen nicht zu unterschätzenden Einfluss auf den Erfolg oder Misserfolg. Denn für die Umsetzung der agilen Ansätze bedarf es einer gewissen **Routine** und **Erfahrenheit**, damit sich ändernde Anforderungen effektiv integrieren lassen.

Kunden

Kunden müssen sich auf die Anforderungen der agilen Softwareentwicklung einlassen, wenn es für den Auftraggeber und den Auftragnehmer funktionieren soll. Dafür müssen sie von Anfang an eine klare Vorstellung davon haben, welche Aufgaben von ihnen während des Projektes gefordert sind.

Sind die Bedingungen für die Durchführung eines agilen Projektes akzeptiert und von beiden Seiten klar eingefordert, können die Kunden viele Vorteile aus der Zusammenarbeit ziehen. Ihre **Einflussnahme** ist von Anfang an erwünscht und sie bekommen lauffähige Versionen mit den neuesten Features in klar definierten Zeitabständen. Das erlaubt ihnen jederzeit **steuernd** einzugreifen und die am Anfang vielleicht noch nicht vollständig vorhandenen Anforderungen weiter zu ergänzen.

Unternehmensgröße und -kultur

Die Größe einer Unternehmung sollte keinen Einfluss auf die Einführung von agilen Methoden und Prozessen haben. Vielmehr muss man darauf achten, dass alle beteiligten Personen während der Einführung **einbezogen** werden und diesen Umbruch auch mittragen.

Darüber hinaus ist es nicht immer sinnvoll, alle klassischen oder über die Jahre entwickelten eigenen Prozesse sofort durch agile Ansätze zu ersetzen. Es kommt immer auch auf die Aufgabenstellung und die beteiligten Personen an, ob und in welchem Umfang eine Methoden- und Prozessänderung erforderlich und gewünscht ist. Durch die gestellten Anforderungen und verfügbaren Fähigkeiten kann es notwendig sein, die Projekte mit klassischen Methoden oder einem **Mix** aus klassischen und agilen Methoden zu bewältigen (z.B. öffentliche Ausschreibungen; Kooperationen mit klassischen Unternehmen; fehlendes Wissen über agile Methoden in der Unternehmung).

Unabhängig von der Unternehmenskultur und -größe müssen Unternehmen darauf achten, dass sie eine **motivierende** Umgebung für ihre Angestellten schaffen, welche ein unterbrechungsfreies Arbeiten und eine möglichst durch Software automatisierte Entlastung fördert.

7. Agilität in Open Source Projekten

Die Ansätze der agilen Softwareentwicklung finden sich auch in vielen Open Source Projekten aufgrund ihrer strukturellen und personellen Grundlage wieder oder können in diesen positive Effekte für die Produktivität und Zusammenarbeit bewirken. Dennoch gibt es auch genügend Open Source Projekte, die auf klassische Weise, oder ohne besondere Ansätze, geführt werden.

Flexibilität

Open Source Projekte leben von dem **Engagement** und der **Zusammenarbeit** von Kern-Entwicklern und Community-Committern. Diesbezüglich ist das Kern-Team oftmals über einen längeren Zeitraum mit der Konzeption und Weiterentwicklung des Projektes beschäftigt und wird zeitlich begrenzt oder in definierten Intervallen durch zusätzliche Entwickler bei der Entwicklung von speziellen Features oder der Beseitigung von vorhandenen Bugs unterstützt.

In diesem Zusammenhang ist es besonders wichtig, neue Ideen und Einflüsse in die Entwicklung zu integrieren und die fortlaufende Planung und Weiterentwicklung nicht durch zu starr definierte und von Anfang an schwerfällige Methoden und Prozesse zu behindern.

Daraus ergibt sich ein Umfeld, in dem agile Ansätze und Denkweisen gut integriert und ihre Vorteile zum Wohle der Projekte eingesetzt werden können. Diese flexible Sichtweise ermöglicht eine äußerst **offene** Umgebung, in der die **Spontanität** und **Motivation** der beteiligten Personen nicht durch zu straffe Vorgaben und Regularien eingegrenzt werden.

Fortlaufende Anpassbarkeit

Ein Grundsatz der agilen Entwicklung ist die Erstellung und fortwährende Lieferung von lauffähigen Versionen, welche regelmäßig mit den neuen Features und Verbesserungen geliefert werden.

Ein großer Vorteil von Open Source Projekten gegenüber proprietären

Entwicklungen ist die **schnelle Anpassbarkeit** und Reaktion auf Veränderungen, welche sich durch die technische Entwicklung, die Erwartungen der Nutzer oder den Ideenreichtum der Community ergeben. Diese Sichtweisen und Grundsätze finden sich auch in den **Prinzipien** der agilen Softwareentwicklung wieder, wo es um die vordringliche Zufriedenstellung der Nutzer, der Sicherung von Wettbewerbsvorteilen oder der nachhaltigen und zukunftsorientierten Entwicklung geht.

Zielgruppe

Committer in Open Source Projekten schätzen die dynamische Anpassung und schnellere Integration von Verbesserungen und neuen Anforderungen in lauffähige Versionen. Daher werden oftmals gerade hoch motivierte und kreative Personen von solchen Projekten **angezogen**, die ihre Ideen umsetzen und mit anderen teilen wollen.

Die agilen Strukturen machen einen Einstieg in das Projekt auch zu einem späteren Zeitpunkt möglich und **erleichtern** das Zurechtfinden und Verinnerlichen der bestehenden und zukünftigen Entwicklungen durch eine geringere Anzahl von Vorgaben, stabilen Anforderungen und Regularien. Dadurch können schneller Mitstreiter gewonnen werden, die zeitnah ein Feedback über ihr Mitwirken in den iterativen Releases erhalten.

Agile Softwareentwicklungsmethoden beachten und berücksichtigen insbesondere die Bedürfnisse und Eigenarten der beteiligten Personen. Man findet diese Sichtweisen in den Prinzipien des agilen Manifesto wieder, wonach **selbstorganisierte** Teams die besten Architekturen, Anforderungen und Entwürfe erstellen. Diese Ansätze, bei denen die **Menschen** wieder in den Vordergrund gestellt werden, finden bei Entwicklern großen Anklang und werden von ihnen durch ihre Motivation und Beteiligung honoriert.

Zusätzlicher Aufwand

Agile Ansätze können in einem Unternehmen, welches bereits eingespielte und vorhandene Strukturen besitzt, oder durch die vorherrschende

Unternehmenskultur und -arbeitsweise (z.B. verteilte Teams, Freelancer) nicht einfach agile Methoden einführen kann, einen zusätzlichen Aufwand bedeuten.

Es bedarf einer **geeigneten** Infrastruktur, welche die sich fortwährenden Anpassungen möglichst einfach integriert und durch eine Vielzahl von **automatisierten** Prozessen eine Erleichterung mit sich bringt.

Ist es nicht möglich, das **direkte** Gespräch mit den beteiligten Personen durchzuführen, müssen die sich stetig ändernden Anforderungen auf eine andere Weise vermittelt werden. Dadurch kommt es zu einem zusätzlichen Aufwand für die Aufbereitung der Informationen und die Sicherstellung, dass alle Teammitglieder auch tatsächlich die Informationen in korrekter Weise verinnerlicht haben.

Möchte man agile Strukturen und Prozesse nutzen und vielleicht auch parallel mit klassischen Ansätzen in der täglichen Arbeit integrieren, müssen bestimmte Bedingungen geschaffen werden, welche die agilen Grundsätze und Prinzipien unterstützen.

8. Mix aus klassischen und agilen Methoden

Agile und klassische Ansätze haben sich im Laufe der letzten Jahrzehnte beständig weiterentwickelt und konnten, je nachdem welche Methoden und Prozesse hinsichtlich der vorhandenen Anforderungen und Strukturen am besten für die jeweiligen Aufgaben geeignet sind, ihre jeweilige Daseinsberechtigung in einer Vielzahl von Projekten nachweisen.

Dennoch wäre es ein Fehler, eine der beiden Sichtweisen als das „Allheilmittel“ oder die am besten geeignete Methode hervor zu heben und diese unberücksichtigt der auftretenden Anforderungen auszuwählen. Vielmehr ist es ratsam, sich auf die in den jeweiligen Projekten gestellten Anforderungen und Rahmenbedingungen **einzustellen** und eine geeignete Methode zu wählen, welche einen möglichst **effizienten** und **produktiven** Arbeitsablauf gestattet. Das kann bei eher agil ausgerichteten Firmen dazu führen, dass projektbezogene klassische Ansätze oder ein Mix aus klassischen und agilen Methoden und Prozessen eingesetzt werden. Andersherum kann es auch

klassisch ausgerichteten Firmen je Projekt und Möglichkeiten helfen, agile Ansätze in die tägliche Arbeit zu integrieren oder ganze Projekte mit einem agilen Prozessmodell durchzuführen.

Ein Beispiel sind u.a. öffentliche Ausschreibungen oder Unternehmen, welche sich über Jahrzehnte einem klassischen Ansatz verschrieben haben. In diesem Umfeld sind die Rahmenbedingungen oftmals sehr klar und unabänderbar definiert, so dass die fehlenden stabilen Anforderungen, welche bei agilen Ansätzen als Vorteil gelten, nicht gegeben oder erwünscht sind.

Letztlich kann es auch von Vorteil sein, einen Teilbereich oder eine Teilkomponente der agilen Methoden und Prozesse zu **vernachlässigen** oder gänzlich zu ignorieren. Diese gewollte Missachtung von bestimmten agilen Konzepten oder Ideen kann dazu beitragen, dass auch **kleine** (z.B. 2 Entwickler) oder stark **verteilte Teams** (z.B. 4 Freelancer in 4 unterschiedlichen Städten) die Vorteile der agilen Softwareentwicklung nutzen können, ohne die zusätzlichen Aufwände einer ganzheitlichen Einführung der Prozesse zu bewältigen.
Daher kann es durchaus nützlich sein, wenn man zum Beispiel nur 80% der Methodiken umsetzt und sich nicht die Mühe macht, die restlichen 20% auch noch in die tägliche Arbeit zu integrieren. Es muss sich letztendlich lohnen, auch diesen Teil nutzen zu wollen, damit die Gesamtproduktivität steigt und sich eine Verbesserung des Entwicklungsprozesses ergibt.

Agile Softwareentwicklung kann dazu beitragen, die Flexibilität, Produktivität und Qualität der Softwareentwicklung zu verbessern und sich von klassischen Ansätzen zu lösen, welche in bestimmten Anforderungsprofilen und Rahmenbedingungen nicht die erwünschten Ergebnisse liefern. Dabei ist es wichtig, dass sich die Methoden und Prozesse nach den Anforderungen und Bedingungen der jeweiligen Projekte richten und demzufolge auch gezielt ausgewählt werden.

Agile und klassische Ansätze sind **gleichberechtigte** Wege, welche unterstützend zur Seite stehen, um die Herausforderungen der modernen Software-Entwicklung zu bewältigen.

Literaturverzeichnis

Balz2001: Helmut Balzert, Software-Entwicklung, 2001, ISBN:3-8274-0480-0

AM2001: 17 Erstunterzeichner, Agiles Manifest, 2001, http://http://agilemanifesto.org/

CB2010: Christian Baranowski, Anforderungsanalyse - Teil 1, 2010, http://www.slideshare.net/tux2323/anforderungsanalyse-grundlagen-und-prototyping

AgilS2014: Wiki-Autoren, Agile Softwareentwicklung, 2014, http://de.wikipedia.org/wiki/Agile_Softwareentwicklung

AMDE2014: Erstunterzeichner, Manifest für Agile Softwareentwicklung, 2001, http://agilemanifesto.org/iso/de/

AMPrinzi2001: Erstunterzeichner, Prinzipien hinter dem Agilen Manifest, 2001, http://agilemanifesto.org/iso/de/principles.html

VOne2014: Version One, 8th annual State of Agile Development Survey 2013, 2014, www.versionone.com/pdf/2013-state-of-agile-survey.pdf

SA2011: Scott W. Ambler, Introduction to User Stories, 2011, http://www.agilemodeling.com/artifacts/userStory.htm

KBeck2004: Kent Beck, Extreme Programming Explained, 2004, ISBN:978-0321278654

CR2009: Collin Rogowski, Agile Software Entwicklung, 2009, http://collin.rogowski.de/Skript.pdf

ACock2001: Alistair Cockburn, Laurie Williams, The Costs and Benefits of Pair Programming, 2001, ISBN:0-201-71040-4

KBeck2002: Kent Beck, Test Driven Development, 2002, ISBN:0-321-14653-0

MF2000: Martin Fowler, Refactoring . Wie Sie das Design vorhandener Software verbessern, 2000, ISBN:3827316308

TDM2003: Tom DeMArco, Timothy Lister, Bärentango, 2003, ISBN:3-446-22333-9

SA2006: Scott W. Ambler, The Agile Unified Process (AUP), 2006, http://www.ambysoft.com/unifiedprocess/agileUP.html

AC2005: Alistair Cockburn, Crystal Clear, 2005, ISBN:0201699478

AC2004: Alistair Cockburn, Crystal Clear: A Human-Powered Methodology for Small Teams: A Human-Powered Methodology for Small Teams, 2004, ISBN:0201699478

KS: Ken Schwaber and Jeff Sutherland, Scrum, , https://www.scrum.org/Scrum-Guide

DA2010: David J. Anderson, Kanban. Successful Evolutionary Change for Your Technology Business, 2010, ISBN:978-0-9845214-0-1

NB1998: Jeff De Luca, The Original FDD Processes, 1998, http://www.nebulon.com/articles/fdd/originalprocesses.html

AD: adaptivedevelopment.com, Adaptive Development Software Development Model, , http://www.adaptivedevelopment.com/Adaptive%20Development%20Software%20Development%20Model.pdf

DN2006: Dan North, Introducing Behaviour Driven Development, 2006, http://dannorth.net/introducing-bdd/

AT2013: Manfred Baumgartner, Martin Klonk, Helmut Pichler, Richard Seidl, Siegfried Tanczos, Agile Testing - Der agile Weg zur Qualität, 2013, ISBN:978-3446431942

MF2006: Martin Fowler, Continuous Integration, 2006, http://www.martinfowler.com/articles/continuousIntegration.html

TNM2005: Martin Fowler, The New Methodology, 2005, http://www.martinfowler.com/articles/newMethodology.html

ASD: Scrummethodology, Agile Software Development, , http://www.scrummethodology.org/agile-software-development.html

André Schütz

Wegtam UG (haftungsbeschränkt)

Hansestrasse 21, 18182 Bentwisch

Mail: andre@wegtam.com

http://www.wegtam.org