Agile Softwareentwicklung bezeichnet Ansätze im Softwareentwicklungsprozess, die die Transparenz und Flexibilität erhöhen und zu einem schnelleren Einsatz der entwickelten Systeme führen sollen, um so Risiken im Entwicklungsprozess zu minimieren. Je nach Kontext bezieht sich der Begriff auf Teilbereiche der Softwareentwicklung – wie im Fall von Agile Modeling – oder auf den gesamten Softwareentwicklungsprozess – exemplarisch sei Extreme Programming oder Scrum aufgeführt.

Agile Softwareentwicklung zeichnet sich durch selbstorganisierende Teams, sowie eine iterative und inkrementelleVorgehensweise aus.

Es wird versucht, mit geringem bürokratischem Aufwand und Regeln auszukommen und sich schnell an Veränderungen anzupassen, ohne dabei das Risiko für Fehler zu erhöhen.

Ziel

Das Ziel agiler Softwareentwicklung ist es, den Entwicklungsprozess flexibler und schlanker zu machen, als das bei den klassischen Vorgehensmodellen der Fall ist. Die agile Softwareentwicklung ist eine Gegenbewegung zu den oft als schwergewichtig und bürokratisch angesehenen traditionellen Softwareentwicklungsprozessen wie dem Rational Unified Process oder dem V-Modell.

Agile Werte

„Wir erschließen bessere Wege, Software zu entwickeln, indem wir es selbst tun und anderen dabei helfen. Durch diese Tätigkeit haben wir diese Werte zu schätzen gelernt:

Individuen und Interaktionen stehen über Prozessen und Werkzeugen

Funktionierende Software steht über einer umfassenden Dokumentation

Zusammenarbeit mit dem Kunden steht über der Vertragsverhandlung

Reagieren auf Veränderung steht über dem Befolgen eines Plans

Das heißt, obwohl wir die Werte auf der rechten Seite wichtig finden, schätzen wir die Werte auf der linken Seite höher ein.“

Agile Methoden

Paarprogrammierung

Testgetriebene Entwicklung

ständige Refaktorierungen

Story-Cards

schnelle Codereviews

Paarprogrammierung

Paarprogrammierung bedeutet, dass bei der Erstellung des Quellcodes jeweils zwei Programmierer an einem Rechner arbeiten. Ein Programmierer schreibt den Code, während der andere über die Problemstellungen nachdenkt, den geschriebenen Code kontrolliert sowie Probleme, die ihm dabei auffallen, sofort anspricht. Diese können dann sofort (im Gespräch zu zweit) gelöst werden. Die beiden Programmierer sollten sich bezüglich dieser beiden Rollen abwechseln. Auch die Zusammensetzung der Paare sollte sich häufig ändern.

Eine Voraussetzung für Paarprogrammierung ist ein gemeinsam vereinbarter Programmierstil des gesamten Teams.

Kombination der Leute, Codierer und Kontrolle sollen ständig ausgewechselt werden.

Testgetriebene Entwicklung

Nach klassischer Vorgehensweise, beispielsweise nach dem Wasserfall- oder dem V-Modell, werden Tests parallel zum und unabhängig vom zu testenden System entwickelt oder sogar nach ihm. Dies führt oft dazu, dass nicht die gewünschte und erforderliche Testabdeckung erzielt wird. Mögliche Gründe dafür sind unter anderem:

Fehlende oder mangelnde Testbarkeit des Systems (monolithisch, Nutzung von Fremdkomponenten, …).

Verbot der Investition in nicht-funktionale Programmteile seitens der Unternehmensführung. („Arbeit, von der man später im Programm nichts sieht, ist vergeudet.“)

Erstellung von Tests unter Zeitdruck.

Nachlässigkeit und mangelnde Disziplin der Programmierer bei der Testerstellung.

Ein weiterer Nachteil klassischer White-Box-Tests ist, dass der Entwickler das zu testende System und seine Eigenheiten selbst kennt und dadurch aus Betriebsblindheit unversehens „um Fehler herum“ testet.

Die Methode der testgetriebenen Entwicklung versucht den Gründen für eine nicht ausreichende Testabdeckung und einigen Nachteilen der White-Box-Tests entgegenzuwirken.

Wasserfallmodell

Ein Wasserfallmodell ist ein lineares (nicht iteratives) Vorgehensmodell, das insbesondere für die Softwareentwicklung verwendet wird und das in aufeinander folgenden Projektphasen organisiert ist. Dabei gehen die Phasenergebnisse wie bei einem Wasserfall immer als bindende Vorgaben für die nächsttiefere Phase ein.

In einem Wasserfallmodell hat jede Phase vordefinierte Start- und Endpunkte mit eindeutig definierten Ergebnissen. Meist beschreibt das Modell auch einzelne Aktivitäten, die zur Herstellung der Ergebnisse durchzuführen sind. Zu bestimmten Meilensteinen und am jeweiligen Phasenende werden die vorgesehenen Entwicklungsdokumente im Rahmen des Projektmanagements verabschiedet.

Der Name Wasserfall kommt von der häufig gewählten grafischen Darstellung der als Kaskade angeordneten Projektphasen. In der betrieblichen Praxis ist es traditionell ein weit verbreitetes Vorgehensmodell, von dem es viele Varianten gibt.

Erweiterungen des einfachen Modells (Wasserfallmodell mit Rücksprung) führen iterative Aspekte ein und erlauben ein schrittweises Aufwärtslaufen der Kaskade. Ein Abweichen vom strengen Vorgänger-Nachfolger-Prinzip wird auch dann erforderlich, wenn in der aktuellen Phase Handlungsbedarf erkennbar wird, der grundsätzlich einer früheren Phase zugeordnet ist, z. B. Anpassungen im Systementwurf oder im Benutzerhandbuch aufgrund von Erkenntnissen beim Testen.

Wasserfallmodelle werden allgemein dort vorteilhaft angewendet, wo sich Anforderungen, Leistungen und Abläufe in der Planungsphase relativ präzise beschreiben lassen.

V-Modell

Das V-Modell ist ein Vorgehensmodell, welches ursprünglich für die Softwareentwicklung konzipiert wurde. Ähnlich dem Wasserfallmodell organisiert es den Softwareentwicklungsprozess in Phasen. Zusätzlich zu diesen Entwicklungsphasen definiert das V-Modell auch das Vorgehen zur Qualitätssicherung (Testen), indem den einzelnen Entwicklungsphasen Testphasen gegenüber gestellt werden. Auf der linken Seite wird mit einer funktionalen/fachlichen Spezifikation begonnen, die immer tiefer detailliert zu einer technischen Spezifikation und Implementierungsgrundlage ausgebaut wird. In der Spitze erfolgt die Implementierung, die anschließend auf der rechten Seite gegen die entsprechenden Spezifikationen der linken Seite getestet wird. So entsteht bildlich das Namensgebende „V“, welches die einzelnen Entwicklungsebenen ihren jeweiligen Testebenen gegenüberstellt.

Refactoring

Refactoring (auch Refaktorisierung, Refaktorierung oder Restrukturierung) bezeichnet in der Software-Entwicklung die manuelle oder automatisierte Strukturverbesserung von Quelltexten unter Beibehaltung des beobachtbaren Programmverhaltens. Dabei sollen die Lesbarkeit, Verständlichkeit, Wartbarkeit und Erweiterbarkeit verbessert werden, mit dem Ziel, den jeweiligen Aufwand für Fehleranalyse und funktionale Erweiterungen deutlich zu senken.

Refactoring ist ein zentraler Bestandteil der Agilen Softwareentwicklung. Dort wird meist von „kontinuierlichem“ Refactoring oder „kompromisslosem“ Refactoring gesprochen. Refactoring ist in der agilen Softwareentwicklung wie Kodieren oder Modultesten ein integraler Bestandteil des Softwareentwicklungsprozesses und nicht auf bestimmte Zeiten bzw. Phasen beschränkt.

Story Cards

Als Story-Card wird das Medium beschrieben, auf dem eine User-Story dokumentiert wird, üblicherweise Zettel (z. B. Klebezettel) oder Computerdatensätze.

Die Story-Cards dienen insbesondere als Kommunikationsmittel zwischen Anforderern und Umsetzern. Die Story-Card enthält die Beschreibung der Story, üblicherweise in Form eines Satzes nach der oben beschriebenen Vorlage. Weiterhin kann das Ergebnis einer Schätzung, zumeist in Form von Story-Points, direkt auf der Story-Card festgehalten werden.

Traditionell enthält die Rückseite der Story-Card die Beschreibung der Akzeptanzkriterien der User-Story. Wenn die Rückseite allerdings nicht zugänglich ist und/oder mehrere Akzeptanzkriterien für eine User-Story definiert werden müssen, können diese auch auf getrennten Zetteln oder in einem Computersystem erfasst sein.

Code Reviews

Mit dem Review werden Arbeitsergebnisse der Softwareentwicklung manuell geprüft. Jedes Arbeitsergebnis kann einer Durchsicht durch eine andere Person unterzogen werden. Der oder das Review ist eine statische Testmethode und gehört in die Kategorie der analytischen Qualitätssicherungsmaßnahmen.

In Anlehnung an die IEEE-Norm 729 ist das Review ein mehr oder weniger formal geplanter und strukturierter Analyse- und Bewertungsprozess, in dem Projektergebnisse einem Team von Gutachtern präsentiert und von diesem kommentiert oder genehmigt werden.

Der untersuchte Gegenstand eines Reviews kann verschieden sein. Es wird vor allem zwischen einem Code-Review (Quelltext) und einem Architektur-Review (Softwarearchitektur, insbesondere Design-Dokumente) unterschieden. Diesen Bereichen zugeordnet sind technische Dokumente wie etwa Readmes, Installationsanweisungen oder Bedienungsanleitungen, aber auch Programme oder Skripte, die für eine Installation gebraucht werden, sowie Dokumente mit Informationen und Anweisungen an andere, ähnlich qualifizierte Entwickler, um diese zu befähigen, den Übersetzungsvorgang der Quellen zu einem späteren Zeitpunkt erfolgreich zu reproduzieren, etwa für ein Bug-Fixing (Fehlerkorrektur) oder eine Weiterentwicklung.

Beim Code-Review wird ein Programmabschnitt nach oder während der Entwicklung von einem oder mehreren Gutachtern Korrektur gelesen, um mögliche Fehler, Vereinfachungen oder Testfälle zu finden. Dabei kann der Gutachter selbst ein Softwareentwickler sein. Für unerfahrene Entwickler bietet der Code-Review durch einen erfahrenen Programmierer eine gute Möglichkeit, sich schnell und praxisorientiert weiterzubilden.

Adaptive Softwareentwicklung

Adaptive Software Development ist ein Softwareentwicklungsprozess, der auf das Rapid Application Development zurückgeht. ASD ist eine Umsetzung des Prinzips der kontinuierlichen Anpassung an immer neue Anforderungen (eher der Normalzustand) und ersetzt damit das verbreitete Wasserfallmodell. Dabei wird alle vier Wochen geprüft, ob eine neu erstellte Programmversion einen Fortschritt zur Vorgängerversion darstellt. Dies geschieht gemeinsam mit dem Kunden. Zwischen jedem der Treffen werden die Phasen 'Spekulieren', 'Zusammenarbeiten' und 'Lernen' durchlaufen.

Christal Family

Crystal Light ist eine Familie von Software-Entwicklungsmethoden, die zu den Agilen Methoden der Softwareentwicklung gerechnet wird. Die Mitglieder dieser Familie sind in der Regel mit Farben bezeichnet. Die einfachste Variante heißt hingegen Crystal Clear (,glasklar‘).

Extrem Programming

Extreme Programming (XP, auch Extremprogrammierung) ist eine Methode, die das Lösen einer Programmieraufgabe in den Vordergrund der Softwareentwicklung stellt und dabei einem formalisierten Vorgehen geringere Bedeutung zumisst. Diese Vorgehensweise definiert ein Vorgehensmodell der Softwaretechnik, das sich den Anforderungen des Kunden in kleinen Schritten annähert.

Feature Driven Development

Feature Driven Development (Abk. FDD) ist eine Sammlung von Arbeitstechniken, Strukturen, Rollen und Methoden für das Projektmanagement im Rahmen agiler Softwareentwicklung.

Kaban

Kanban ist eine Methode, bei der die Anzahl paralleler Arbeiten, der Work in Progress (WiP), begrenzt und somit kürzere Durchlaufzeiten erreicht und Probleme – insbesondere Engpässe – schnell sichtbar gemacht werden sollen.

Scum

Scrum (aus englisch scrum für „[das] Gedränge“) ist ein Vorgehensmodell des Projekt- und Produktmanagements, insbesondere zur agilen Softwareentwicklung. Es wurde ursprünglich in der Softwaretechnik entwickelt, ist aber davon unabhängig. Scrum wird inzwischen in vielen anderen Bereichen eingesetzt. Es ist eine Umsetzung von Lean Development für das Projektmanagement.

Agiles Testen

Unter agilem Testen versteht man das Testen von Software im Rahmen eines agilen Entwicklungsprojekts. Testen in agilen Entwicklungsprojekten bedarf dabei vor allem eines Fokus auf die Unterstützung des Entwicklungsteams. Viele Eigenschaften von agilen Testern helfen auch einem traditionellen Tester, allerdings werden die damit verbundenen Probleme nicht so deutlich und deshalb nicht von jedem Projektteam behoben. Agiles Testen folgt den Prinzipien des agilen Manifests und wendet die agilen Prinzipien auf das Testen an.

Behaviour Driven Development

Behavior Driven Development (BDD, deutsch verhaltensgetriebene Softwareentwicklung), auch als Specification Driven Development (SDD, deutsch anforderungsgetriebene Softwareentwicklung) bezeichnet, ist eine Technik der agilen Softwareentwicklung, welche die Zusammenarbeit zwischen Qualitätsmanagement und Business-Analyse in Softwareentwicklungsprojekten stärkt. Beim Behavior Driven Development werden während der Anforderungsanalyse die Aufgaben, Ziele und Ergebnisse der Software in einer bestimmten Textform festgehalten, die später als automatisierte Tests ausgeführt werden kann und damit die Software auf ihre korrekte Implementierung getestet werden kann. Die Softwareanforderungen werden dabei meist in „Wenn-dann“-Sätzen basierend auf der ubiquitären Sprache des Domain-driven Designs verfasst. Damit soll der Übergang zwischen der Sprache der Definition der fachlichen Anforderungen und der Programmiersprache, mittels derer die Anforderungen umgesetzt werden, erleichtert werden.

Man kann mit der agilen Sofwareentwicklung besser auf interne und externe Veränderungen reagiern.

-Flexibilität

-unvollständiger Auftrag

-Transparenz

-mit Auftraggeber mehr verknüpft

-führt kleiner Wassermodellbausteine aus und zeit das wiederum den Kunden

Vorteile:

-

Extreme Programming

-Entwickler alle gleichwertig

Wasserfallmodell

-Analyse

-Erstellung von Konzepten

-Ausprogrammierung

-Testen der Funktionen

-danach direkt zum Auftraggeber