PROJEKTMANAGEMENT-WISSEN

Abgestimmt mit The Scrum Guide™

SCRUM

Agiles Projektmanagement und Scrum erfolgreich anwenden

Proconis

ROLAND WANNER

SCRUM

Agiles Projektmanagement und Scrum erfolgreich anwenden

ROLAND WANNER

Kontakt zum Autor:
Roland Wanner
E-Mail: info@rolandwanner.ch
Internet: www.rolandwanner.ch/

Herstellung:
Amazon Media EU S.à r.l, Luxembourg

Haftungsausschluss
Dieses Buch enthält Informationen über Scrum und agiles Projektmanagement. Es wurde zu Informations- und zu Weiterbildungszwecken geschrieben. Für den professionellen Einsatz empfiehlt sich die Unterstützung durch eine kompetente Fachperson.
Trotz größter Sorgfalt dieses Buch so vollständig und korrekt wie möglich zu machen ist nicht auszuschließen, dass es Fehler enthält, typografische oder inhaltliche. Deshalb ist dieser Text nur als genereller Leitfaden und nicht als alleinige Informationsquelle über Scrum und Agiles Projektmanagement zu verwenden.
Der Autor, Herausgeber und die zitierten Quellen haften nicht für etwaige Verluste, die aufgrund der direkten oder indirekten Umsetzung der in diesem Buch verwendeten Beschreibungen und Formeln entstehen könnten.
Themen in diesem Buch sind: Scrum, agiles Projektmanagement, IT und Software Projekte, Sprint, Timeboxing, Product Owner, Daily Scrum und weitere.

Bei Fragen oder Anregungen kontaktieren Sie bitte:
info@rolandwanner.ch

ISBN: 978-1983653995
1. Auflage Januar 2018, V1.0

Bibliografische Information der Deutschen Bibliothek:

Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.ddb.de abrufbar.

Inhalt

Dieses Buch basiert auf „The Scrum Guide™“ vom November 2017 von Ken Schwaber and Jeff Sutherland

Vorwort

Ich gratulieren Ihnen, dass Sie sich mit agilem Projektmanagement und Scrum beschäftigen wollen! Agiles Projektmanagement wird in Zukunft stark an Bedeutung gewinnen und mit dem Wissen in diesem Buch sind Sie für die nächste Zukunft gut vorbereitet. In diesem Buch erhalten Sie eine kurze und allgemeine Einführung in das agile Projektmanagement. Diese legt dann die Basis für den größten Teil des Buches, dass agile Projektmanagement mit Scrum für Software-Entwicklungsprojekte.

Agile Methoden sind im Vormarsch, nicht nur im Projektmanagement. Auch in anderen Managementbereichen versucht man langsam den Nutzen daraus zu ziehen, Prozesse zu verschlanken, sich noch mehr auf den Kunden und den Kundennutzen zu fokussieren, noch schneller auf Marktveränderungen zu reagieren und mehr mit selbstorganisierenden Teams zu „experimentieren“. Wenn Sie schon etwas älter sind, dann kommen Ihnen diese Themen vielleicht bekannt vor.

Die Prinzipien, Methoden und Werte, auf denen das agilen Projektmanagement aufbaut, gibt es schon seit Jahrzehnten. Ich habe mich schon vor 30 Jahren damit beschäftigt. Sie stammen praktisch alle aus den 1950er und 1960 vom Toyota Production System und anderen japanischen Unternehmen. Daraus entstanden dann in den Folgejahren Management-Methoden, wie z.B. Lean Production, Lean Management, Business Process Reengineering, Six-Sigma oder das Wissensmanagement. Auch selbstorganisierende Team sind ein alter Hut. Auch diese wurden schon vor vielen Jahrzenten in japanischen und einzelnen amerikanischen Unternehmen erfolgreich eingesetzt.

Alle diese Management-Methoden waren in den betreffenden Unternehmen sehr erfolgreich, haben sich aber erstaunlicherweise in der Wirtschaft nie richtig durchgesetzt und gingen fast vergessen. Einzig

die Autoindustrie hat vom Toyota Production System und Lean Production gelernt und davon stark profitiert.

Die Arbeit in Projekten in der Industrie, auf dem Bau und in anderen Branchen wird schon seit Jahrhunderten sequentiell in Phasen ausgeführt, die aufeinander aufbauen. Als die Informationstechnologie und Softwareentwicklung in den 1960er Jahren entstand wurden die sequentiellen Vorgehensweisen auch in diese Projekte übernommen. In den Folgejahren wurden verschiedene mehr oder weniger erfolgreiche iterative Softwareentwicklungs-Methoden entwickelt.

1995 präsentierten Jeff Sutherland und Ken Schwaber zusammen ein Dokument, welches die Scrum Methode für Softwareprojekte beschrieb. Sie entwickelten, basierend aus den sehr erfolgreichen, aber schon fast vergessenen Werten, Prinzipien und Methoden, die Sie vorhin kurz kennengelernt haben, die erfolgreichste Projektmanagement Methode für die Softwareentwicklung – seit Software entwickelt wird – und dies ist SCRUM.

Für wen wurde dieses Buch geschrieben?

Dies ist ein Buch für alle, die am agilem Projektmanagement interessiert sind und wissen wollen, wie die bekannteste agile Methode in der Softwareentwicklung, Scrum, funktioniert.

Dies ist kein Buch mit allen Details, vielen Beispielen, Geschichten und Ausschweifungen. Für das gibt es umfassendere und viel teure Bücher. Hier lernen Sie konzentriert das Wichtigste über agiles Projektmanagement und Scrum.

Ob Sie bereits im Softwarebereich arbeiten, Student, Projektauftraggeber für Software oder bereits in einem agilen Projekt arbeiten – dieses Buch liefert Ihnen in kompakter Form das notwendige Wissen, agiles Projektmanagement und besonders Scrum besser zu verstehen und erfolgreich anzuwenden.

So ist das Buch aufgebaut

Zuerst lernen Sie, wie das agile Projektmanagement entstanden ist und was der Unterschied ist zum „normalen" Projektmanagement ist. Dann lesen Sie, was das agile Manifest ist und was die agilen Prinzipien für eine Bedeutung für Scrum und die anderen agilen Methoden hat.

Der Hauptteil dieses Buches beschäftigt sich natürlich mit Scrum. Zuerst gebe ich Ihnen **ab Seite 42** einen ganz schnellen **Überblick über das Scrum Framework**, damit erhalten Sie innerhalb von fünf Minuten ein Grundwissen und verstehen so später die Zusammenhänge viel besser.

Die folgenden Kapitel geben Ihnen einen **Überblick** über die Werte von Scrum, Scrum Ereignisse (Events) und Artefakt. Die weiteren Kapitel vertiefen dann das gelernte und gehen ins Detail der Scrum Artefakte und Ereignisse.

Sie werden während dem Lesen sicher feststellen, dass viele Themen in diesem Buch mehrfach in verschiedener Granularität vorkommen. Das ist kein Fehler oder nur um das Buch mit Seiten zu füllen, sondern dient dazu, damit Sie sich immer weiter in Scrum vertiefen und immer mehr Details der verschiedenen Themen kennenzulernen.

Ein ideales Nachschlagewerk

Wenn Sie dieses Buch gelesen haben, dann ist es für Sie später auch ein ideales Nachschlagewerk. Am Schluss des Buches finden Sie ein umfangreiches **Glossar** und ein hilfreiches **Stichwortverzeichnis**, mit dem Sie schnell ein bestimmtes Thema im Buch finden und sich darüber informieren können.

Nicht zuletzt eignet sich dieses Buch ausgezeichnet **für Schulungen** und **das Studium** im Bereich Softwareentwicklung.

Der Scrum Guide

The Scrum Guide™ von Ken Schwaber und Jeff Sutherland ist die offizielle Guideline für Scrum. Sie wird periodisch aktualisiert. Dieses Buch basiert auf der neusten Version des Scrum Guides vom November 2017.

Der Scrum Guide beschreibt in sehr knapper Form die Mindestanforderung an Scrum. Er kann hier kostenlos heruntergeladen werden: https://www.scrum.org/resources/scrum-guide

Das Agile Manifest

Das Agile Manifest für Softwareentwicklung ist der kleinste gemeinsame Nenner aller agilen Vorgehensmodelle – und damit auch für Scrum. Es beschreibt die „Zwölf Prinzipien Agiler Softwareentwicklung", die Sie hier nachlesen können:

http://agilemanifesto.org/

Terminologie in diesem Buch

Ich verwende in diesem Buch bewusst die bekannten, englischsprachigen Scrum-Begriffe für die *Rollen*, *Artefakte* und *Ereignisse*. So werden Sie nicht mit neuen Wortschöpfungen unnötigerweise verwirrt.

Dieses Buch ist absichtlich in der männlichen Form geschrieben, natürlich nicht um das weibliche Geschlecht auszugrenzen oder zu diskriminieren, sondern damit der Text einfacher zu lesen ist. Ich hoffe die weiblichen Leser haben Verständnis dafür.

Agiles Projektmanagement

Agile Methoden werden in immer mehr Projekten eingesetzt – in der Softwareproduktentwicklung schon seit vielen Jahren erfolgreich, bei anderen Projektarten stehen wir hier noch am Anfang. Unternehmen, die agile Methoden einsetzen oder planen einzusetzen, steigt jedoch konstant.

Seit kurzem wird versucht agile Methoden auch in anderen Geschäftsprozessen einzusetzen als bei Projekten. Dies ist eine spannende Herausforderung, welche die gesamte Unternehmenskultur, Führungssysteme und die Zusammenarbeit stark beeinflussen wird. Ich bin mir aber nicht so sicher, ob viele Großunternehmen diesen Schritt in den nächsten Jahren schaffen werden. Kleinunternehmen sind diesbezüglich viel anpassungsfähiger.

Agile Methoden wie Scrum zu lernen und zu verstehen ist relativ einfach. Die agilen Werte und Grundprinzipien zu verinnerlichen und zu leben ist hingegen einiges schwieriger – und hier haben und werden viele Unternehmen noch Mühe haben.

Der Erfolg agiler Methoden auf Unternehmensebene kommt schlussendlich aber nur zustande durch radikale Veränderungen innerhalb der Organisationen und hier sind wir noch weit davon entfernt große Fortschritte zu machen. Auf Projektebene sind wir hier aber hingegen schon sehr weit fortgeschritten.

Agile Methoden haben einen radikalen Einfluss auf die Führungs- und Kompensationssysteme in Unternehmen. Manager verlieren Macht und Einfluss bei selbstorganisierenden Teams. Dies wird die Veränderung, speziell in der Unternehmenskultur nicht einfach machen.

Das Prinzip von selbstorganisierenden und funktionsübergreifenden Teams, Chefs als Coaches ohne Führungsfunktion und der Abbau von Managementebenen, war schon vor Jahrzehnten kurz ein „interessantes Thema". Ich hoffe in den nächsten Jahren haben wir damit mehr Erfolg.

In der Softwareentwicklung mit Scrum werden agile Methoden schon seit einigen Jahren erfolgreich angewendet. Ein erster, wichtiger Schritt ist also schon gemacht!

Agiles Projektmanagement im Überblick

Imposante Projekte wurden schon vor tausenden von Jahren durchgeführt. Denken Sie z.B. an die Steinstruktur von Stonehenge, die 3500 Jahre v. Chr. und die ägyptischen Pyramiden, die 2500 v. Chr. erbaut wurden oder in der neueren Zeit an die mittelalterlichen Burgen, Festung, Schlösser und Kathedralen, die Dampfmaschine, Autos, die Atombombe oder die Wolkenkratzer. Dies waren teilweise riesige und komplexe Projekte für Ihre Zeit. Software und Softwareprojekte gibt es jedoch erst seit ein paar Jahrzenten. In den 1950er Jahren war Software noch Teil der Hardware und wurde als Programmcode bezeichnet. Ich kann mich auch noch gut an die Lochkarten erinnern, mit denen in den 1970er Jahren Werkzeugmaschinen gesteuert wurden. Die Lochkarten waren die Programme, um mit Werkzeugmaschinen z.B. Teile zu fräsen.

Softwareentwicklungsmethoden gab es bis in die 1960er Jahre noch keine. Der Systems Development Life Cycle (SDLC) war der erste, der in dieser Zeit entstand, mit dem Ziel, große, funktionale Business Systeme zu entwickeln. Alle Projekte wurden bis in die 90er Jahre nach dem sequentiellen Wasserfallmodell abgewickelt (Abbildung Seite 17). Das heißt, alle Anforderungen wurden zu Projektbeginn festgelegt, dann wurden Konzepte, Spezifikationen und Pläne erstellt und dann das Produkt gebaut und eingeführt.

Die Softwareentwicklung in den 1990er Jahren wurde durch die objektorientierte Programmierung und durch den Aufstieg des Internets und den Dot.com-Boom geprägt. Hier war Time-to-Market und Firmenwachstum entscheidend.

Das Wasserfall-Modell war nicht mehr geeignet

Mit dem starren Wasserfall-Prozess war man in der Software-Entwicklung immer weniger zufrieden, besonders weil die Projekte immer komplexer wurden, Produktlebenszyklen immer kürzer und das Umfeld und die Anforderungen dynamischer. Man benötige immer schneller brauchbare Software, nicht mit allen Funktionen, aber den Wichtigsten. Leichtgewichtiger, flexibler und schneller sollte die Softwareentwicklung werden und weniger Administration waren gewünscht.

Neue Methoden sollten den Softwareentwicklungsprozess flexibler und schlanker gestalten – als Gegenbewegung zu den eher schwergewichtigen und bürokratischen, traditionellen Methoden, wie eben z.B. dem Wasserfall-Modell. Diese Forderungen stießen eine aktive Entwicklung von Methoden in der Softwareentwicklung an:

- 1986 entwickelte Barry Boehm erste Ansätze eines iterativen Software-Entwicklungsprozesses mit dem risikoorientierten Spiralmodel.
- 1991 wurde Rapid Application Development (RAD) vorgestellt
- 1995 präsentierten Jeff Sutherland und Ken Schwaber zusammen ein Dokument, welches die Scrum Methode erstmals beschrieb
- 1998 wurde der Rational Unified Process (RUP) vorgestellt
- 1999 wurde Extreme Programming (XP) vorgestellt, welches auf großes Interesse bei den Softwareentwicklern stieß.

Wie Sie sehen, sind erste Ansätze zu agiler Softwareentwicklung bereits Anfang der 1990er Jahre zu finden. Popularität erreichte die agile Softwareentwicklung erstmals 1999, als Kent Beck das erste Buch zu Extreme Programming (XP) veröffentlichte.

Das Interesse an Extreme Programming ebnete den Weg auch für andere agile Prozesse und Methoden. Die Bezeichnung „agil“ für diese Art der Softwareentwicklung wurde ausgewählt von 17 Vertretern von verschiedenen Softwareentwicklungsmethoden im Februar 2001, bei einem Treffen in Utah (USA). Dies als Ersatz für das bis dahin gebräuchliche leichtgewichtig (engl. lightweight). Bei diesem Treffen wurde auch das Agile Manifest (siehe Seite 23) formuliert. Daraus hat sich dann im Laufe der Jahre die Bezeichnung „agiles Projektmanagement“ entwickelt, denn nicht nur Softwareprojekte lassen sich agil planen und steuern, sondern auch andere Projektarten.

Das Ziel agiler Softwareentwicklung ist es, den Entwicklungsprozess flexibler und schlanker zu machen, als dass bei den klassischen Vorgehensmodellen der Fall ist. Agile Softwareentwicklung zeichnet sich durch selbstorganisierende Teams, sowie eine iterative und inkrementelle Vorgehensweise aus. Es wird versucht, mit geringem bürokratischem Aufwand und weniger Regeln auszukommen und sich so schnell an Veränderungen anzupassen, ohne dabei das Risiko für Fehler zu erhöhen.

Der Unterschied zwischen „klassischen“ und „agilen“ Projekten

Bei „**klassischen“ Projekten** werden vom internen oder externen Kunden (Auftraggeber) zu Projektbeginn klare Ziele und Anforderungen definiert, die sich während der Projektdurchführung möglichst nicht mehr ändern. Am Ende des Projektes erhält der Kunde, was er am Anfang definiert hat.

Das Projekt wird strikt in nacheinander folgenden Phasen durchgeführt. Eine vorhergehende Phase muss beendet sein, bevor mit der nächsten gestartet werden kann. Das Projektresultat entsteht im Ablauf der Phasen, bis es dann am Ende der letzten Phase vollständig fertiggestellt ist. Dieser Ablauf wird Wasserfall-Modell genannt.

Je weiter das Projekt fortschreitet, desto weniger Einfluss kann der Kunde auf das Ergebnis nehmen. Eine große Einschränkung beim Wasserfallmodell ist, dass jede Änderung oder neue Anforderungen, die der Kunde in einer späteren Projektphase noch umgesetzt haben will, ein Mehrfaches kostet, als wenn er diese am Anfang definiert hätte.

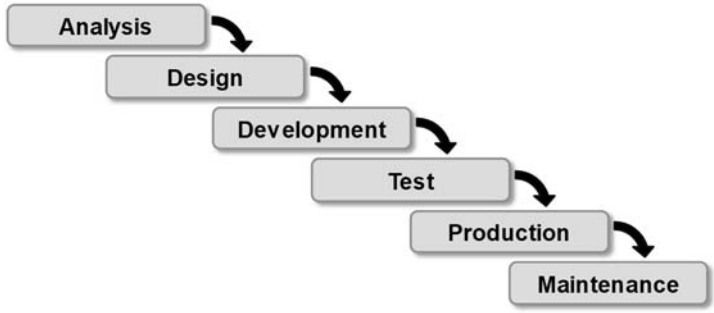


Abbildung 1: Das Wasserfall-Modell

Sie können sich vermutlich gut vorstellen wie man ein Haus baut. Wenn bei diesem Projekt die Bauarbeiter bereits die Wände mauern und Sie wünschen zu diesem Zeitpunkt noch ein Zimmer mehr, dann wird dies sehr teuer oder es ist fast unmöglich.

Agile Methoden, z. B. Scrum werden in Entwicklungsprojekten (besonders von Software) verwendet, um sich der hohen Dynamik der Ziele, Anforderungen, Umfeld und Erwartungen anzupassen. Man setzt dabei u.a.

- auf enge Zusammenarbeit zwischen Kunden, dem Product Owner und dem sich weitgehend selbst organisierenden Team
- auf kurze Entwicklungszyklen, nach denen Änderungen und neue Anforderungen in die Planung zusätzlich aufgenommen werden können (Iterationen mit definierter Länge; typisch sind 14 bis maximal 30 Tage).

So macht es das agile Projektmanagement dem Auftraggeber oder den Stakeholdern möglich neue Anforderungen während dem gesamten Projektablauf einzubringen oder bestehende zu ändern und so auf kurzfristige Marktbedürfnisse flexibel einzugehen – und dies ohne ein exponentielles Kostenwachstum zu verursachen. Dabei muss natürlich das Gesamtbudget trotzdem im Auge behalten werden.

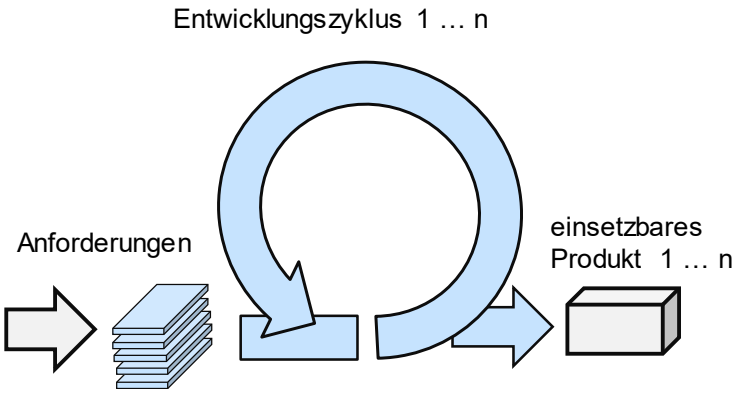


Abbildung 2: Agiles Vorgehen

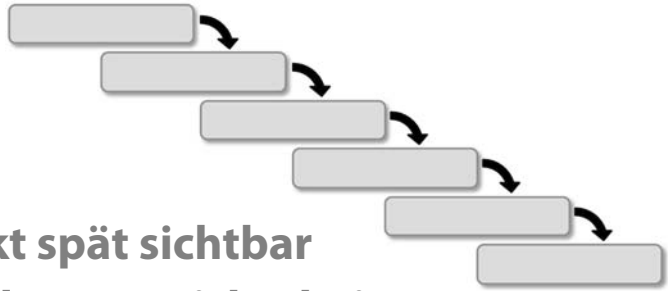
Agiles Projektmanagement für alle Projekte?

Agiles Projektmanagement setzt sich immer mehr durch, auch außerhalb der traditionellen Softwareentwicklung. Viele sagen dem Wasserfallmodell schon den Tod voraus. Soweit wird es aber nicht kommen. Versuchen Sie ein Haus, Flugzeugträger, Werkzeugmaschine oder eine Fabrikationsanlage mit agilem Projektmanagement zu entwickeln und zu fertigen. Unmöglich! Dort müssen praktisch alle Anforderungen zu Projektbeginn bekannt sein, und der Spielraum diese während dem Projektablauf zu ändern oder neue hinzuzufügen ist sehr gering.

Das heißt aber nicht, dass agile Methoden nicht für bestimmte Lieferobjekte von „Wasserfallprojekten" eingesetzt werden können. Überall, wo zum Beispiel Software entwickelt werden muss, können wahrscheinlich agile Entwicklungsmethoden eingesetzt werden, zum Beispiel für die Software einer Werkzeugmaschine oder eines Car Entertainment Systems.

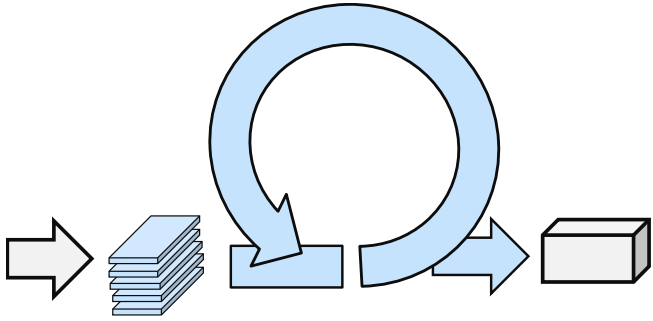
Bestimmte Prinzipien, Praktiken und Werte des agilen Projektmanagements werden sicher auch das traditionelle Projektmanagement in Zukunft stark beeinflussen. Ich denke nur schon an die Zusammenarbeit im Team. Aber überall wo physische Produkte entwickelt werden ist die Flexibilität im Entwicklungsprozess aber eindeutig geringer als bei Software.

Wasserfall-Modell



- starr
- Produkt spät sichtbar
- hohe Planungssicherheit
- konstante Anforderungen notwendig

Agiles Vorgehen



- wenig Struktur, iterativ
- früh einsetzbare Resultate
- hohe Flexibilität, marktnah
- ändernde Anforderungen willkommen

Was macht Teams in der Produkteentwicklung erfolgreich?

Ein Kernpunkt im agilen Projektmanagement ist die Zusammenarbeit in Teams und das Anwenden von Wissen. Um mehr darüber zu erfahren machen wir jetzt einen kurzen Abstecher in die Vergangenheit.

Hirotaka Takeuchi und Ikujiro Nonaka beschrieben 1986 im Harvard Business Review Artikel "The New New Product Development Game" die Eigenschaften erfolgreicher Produkteentwicklungsteams. Der Inhalt des Artikels ist stark beeinflusst vom „Toyota Production System" und japanischen Produkteentwicklungsmethoden z.B. bei Canon, Honda oder NEC aus den 1970er Jahren. In diesem Artikel werden wesentliche Eigenschaften von erfolgreichen Product-Development-Teams beschrieben:

1. **Built-in instability:** Das Management stößt den Entwicklungsprozess an, setzt herausfordernde Ziele und Anforderungen und gibt dem Team größtmögliche Freiheit in der Entwicklung.
2. **Self-organizing project teams**: Das Team arbeitet wie ein Start-up Unternehmen, es ist initiativ, geht Risiken ein und hat eine unabhängige Agenda: Die Teams charakterisieren sich durch folgende drei Eigenschaften:
   - *Authonomy*: Die Teams organisieren sich selber.
   - *Self-transcendence:* Stetiges lernen. Das Team strebt nach vorne und will immer besser werden.
   - *Cross-fertilization:* Die Teams arbeiten funktionsübergreifend. Diese Diversität fördert neue Ideen und Konzepte, was zu erfolgreicheren Produkten führt.
3. **Overlapping development phases**: Mit stark überlappenden Phasen wird der Entwicklungsprozess schneller und flexibler.

4. **Multilearning:** Durch Lernen auf verschiedenen Unternehmensebenen und in Gruppen, durch stetigen Erfahrungsaustauch und durch den Kontakt mit der Umwelt kann man schneller auf verändernde Marktbedingungen reagieren.
5. **Subtle control:** Das Management setzt genügend Checkpunkte, obwohl sich das Projektteam selbst organisiert. Damit sollen Instabilität, Unklarheiten und Spannungen vermieden werden.
6. **Organizational transfer of learning:** Gelerntes Wissen soll in die Organisation einfließen, um sich stetig zu verbessern.

Außerhalb Japans haben nur wenige Industrien Elemente dieses Konzepts einließen lassen, und wenn, dann oft eher halbherzig. Abgeleitet aus diesem Wissen entstand in den 1990er Jahren das Lean Management und das Knowledge Management.

Die geringe Erfolgsquote bei Softwareprojekten in den 1990er Jahren lag nicht in der Ausbildung der Mitarbeiter oder an der fehlenden Reife, der noch jungen Informatik und deren Werkzeugen, sondern in der Durchführung und der Zusammenarbeit in Projekten. Ken Schwaber und Jeff Shuterland haben dies erkannt und dann aus dem Wissen von "The New New Product Development Game" und eigenen Erfahrungen ein erfolgreiches Softwareentwicklungs-Framework entwickelt – SCRUM.

Das Agile Manifest

Im Frühjahr 2001 verabschiedeten 17 projekterfahrene Software-Entwickler in Utah (USA) das sogenannte „Manifesto for Agile Software Development", heute vor allem unter der Kurzbezeichnung „Agile Manifesto" bekannt. Diese Erstunterzeichner, darunter auch die beiden Scrum-Gründer Ken Schwaber und Jeff Sutherland, formulierten mit dem Agile Manifesto erstmals die zentralen Werte agiler Softwareentwicklung – ein Meilenstein und zugleich das Fundament des agilen Projektmanagements.

Die Werte des Agilen Manifests (www.agilemanifesto.org) sind paarweise beschrieben, wobei die Werte auf der linken Seite jeweils höher eingeschätzt werden, als die Werte auf der rechten Seite. Dies heißt aber nicht, dass diese bedeutungslos sind.

Das Agile Manifest lautete folgendermassen:

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

Individuen und Interaktionen

Funktionierende Software

Zusammenarbeit mit dem Kunden

Reagieren auf Veränderung

Die Bedeutung des Agile Manifests

In der Bedeutung für das agile Projektmanagement hat das Agile Manifesto bis heute nichts eingebüßt. Im Laufe der Jahre sind sogar noch Tausende weiterer Befürworter und Unterzeichner hinzugekommen. Den Verfassern ist damals gelungen, die Kerngedanken moderner Software-Entwicklung, trotz teils höchst unterschiedlicher Auffassungen und Herangehensweisen, ein für alle Mal auf einen gemeinsamen Nenner zu bringen.

Der große Fortschritt bestand und besteht darin, dass mit dem agilen Manifest endlich ein Wertesystem in Stein gemeißelt wurde, dass eine konkrete Vorgehensweise umreißt. Gleichzeitig konnte damit auch der bis dahin gängige, vergleichsweise unscharfe Begriff der „leichtgewichtigen Software-Entwicklung“ abgelöst werden. Insofern lässt sich das Agile Manifesto am ehesten als die „Geisteshaltung der Agilität“ verstehen, und diese Geisteshaltung lebt bis heute in agilen Methoden wie Scrum oder Kanban bzw. deren Regeln und Rollen fort.

Das Agile Manifest hatte auch zur Folge, dass Mitarbeiter fortan nicht einfach nur als Ressourcen betrachtet wurden, sondern als Akteure im Mittelpunkt stehen.

Agilität im Projektmanagement fordert und fördert die individuellen Fähigkeiten der Mitarbeiter, indem man ihnen mehr Verantwortung überträgt und kreative Gestaltungsmöglichkeiten einräumt. Dadurch wird der Weg geebnet für effektivere und erfolgreichere Projektverläufe.

Agile Prinzipien

Das Agile Manifest kann missverstanden oder falsch interpretiert werden, deshalb wurden die Aussagen durch zwölf Prinzipien detaillierter erklärt.

Es ist zum Beispiel nicht die Meinung, dass es keine Dokumentationen mehr geben soll. Gemeint waren die mehreren hundert Seiten langen Dokumente, die niemand liest noch pflegt.

Im Zentrum der Prinzipien steht der Mensch, sei es das Scrum-Team oder der Kunde. Die agilen Prinzipien sind eine wesentliche Orientierungshilfe für erfolgreiche agile Projekte und sie sind auch heute immer noch uneingeschränkt gültig.

Die zwölf agilen Prinzipien hinter dem Agilen Manifest:

1. **Den Kunden zufriedenstellen:** Unsere höchste Priorität ist es, den Kunden durch frühe und kontinuierliche Auslieferung wertvoller Software zufrieden zu stellen.
2. **Änderungen willkommen heissen:** Anforderungsänderungen, selbst spät in der Entwicklung sind willkommen. Agile Prozesse nutzen Veränderungen zum Wettbewerbsvorteil des Kunden.
3. **Häufige Auslieferungen:** Liefere funktionierende Software regelmäßig, innerhalb weniger Wochen oder Monate, bevorzugt in einer kürzeren Zeitspanne.
4. **Bereichsübergreifende Zusammenarbeit:** Fachexperten und Entwickler müssen während des Projektes täglich zusammenarbeiten.
5. **Unterstützung leisten und Vertrauen schenken:** Errichte Projekte rund um motivierte Individuen. Gib ihnen das Umfeld und die Unterstützung, die sie benötigen und vertraue darauf, dass sie die Aufgabe erledigen.
6. **Persönliche Kommunikation:** Die effizienteste und effektivste Methode, Informationen an und innerhalb eines Entwicklungs-Teams zu übermitteln ist im Gespräch, von Angesicht zu Angesicht.

7. **Funktionierende Software:** Funktionierende Software ist das wichtigste Massstab für den Fortschritt.
8. **Nachhaltige Geschwindigkeit:** Agile Prozesse fördern nachhaltige Entwicklung. Die Auftraggeber, Entwickler und Nutzer sollten ein gleichmäßiges Tempo auf unbegrenzte Zeit halten können.
9. **Streben nach technischer Exzellenz:** Ständiges Augenmerk auf technische Exzellenz und gutes Design fördert Agilität.
10. **Einfach ist besser:** Einfachheit – die Kunst, die Menge der Arbeit die nicht getan wird zu maximieren – ist entscheidend.
11. **Selbstorganisiert agieren:** Die besten Architekturen, Anforderungen und Entwürfe entstehen durch selbstorganisierte Teams.
12. **Überprüfen und anpassen:** In regelmäßigen Abständen reflektiert das Team, wie es effektiver werden kann und passt sein Verhalten entsprechend an.

Bitte beachte Sie, die „agilen Werte“ unterscheiden sich von den „Scrum Werten“. Diese finden Sie ab Seite 65.

Der Mensch, die Zusammenarbeit und das Wissen stehen im Zentrum

Die agilen Werte und Prinzipien stellen den Menschen in den Vordergrund, sowie deren Zusammenarbeit und neue Führungsmodelle, aber auch das Wissen und schlanke Prozesse. Hier steht der Mitarbeiter im Zentrum, genauso wie der Kunde, für den das Produkt entwickelt wird.

Das Wissen im Unternehmen und der Wissensmitarbeiter wird leider immer noch viel zu wenig beachtet, dabei ist dies heute der entscheidende Wettbewerbsvorteil gegenüber der Konkurrenz.

Wissensmitarbeiter brauchen wenig Führung und wollen mitentscheiden. Zeigen von Anerkennung und Wertschätzung ist wichtig, aber auch das Fördern von Kreativität und Mut. Hier braucht es auch eine offene Fehlerkultur.

Bei der Zusammenarbeit geht es um das Miteinander und nicht das Gegeneinander. Hier sind Werte wie Rücksicht, Vertrauen, Toleranz, Offenheit und Respekt gefragt.

Agilität ist eine Haltung die immer mehr gefragt ist, nicht nur in der Softwareentwicklung. Wir werden immer mehr zu einer Wissensgesellschaft. Veränderungen, speziell im Berufsumfeld, aber auch mit IT-Werkzeugen mit denen wir arbeiten verändern sich immer schneller. Dabei geht die persönliche Kommunikation oft unter – und genau auch die will die agile Bewegung wieder stärken.

Agile Methoden

Wie Sie bereits ein paar Seiten früher gelesen haben entstanden viele Grundelemente agiler Methoden schon in den 1970er Jahren in Japan bei Toyota (Toyota Production System), Canon und bei NEC und wurden 1986 im Harvard Business Review Artikel " The New New Product Development Game" beschrieben.

Im Projektmanagement wurden viele neue Methoden entwickelt, die allerdings fast ausnahmslos auf die Softwareentwicklung zugeschnitten sind. Erste Ansätze entstanden in der agilen Softwareentwicklung bereits Anfang der 1990er Jahre. Popularität erreichte die agile Softwareentwicklung erstmals 1999, als Kent Beck das Buch zu Extreme Programming veröffentlichte. In der Industrie findet man agile Methoden hauptsächlich bei den Automobilproduzenten.

In der folgenden Aufstellung finden Sie einige wichtige agile Methoden in der Softwareentwicklung. Je weiter oben Sie steht, desto verbreiteter und bekannter ist sie.

- Scrum
- Extreme Programming (XP)
- Unified Process (RUP, AUP, OUP)
- Feature Driven Development (FDD)
- Adaptive Software Development (ASD)

Der Rational Unified Process (RUP) wird von vielen Vertretern agiler Methoden als nicht-agiler, schwergewichtiger Prozess aufgefasst. Das ist allerdings umstritten. Deshalb wurde versucht mit dem Agile Unified Process eine agile Variante von RUP zu entwickeln.

Was ist Agilität?

Agilität ist die Eigenschaft von Schnelligkeit, Leichtgewichtig und Anpassungsfähigkeit. Dies umfasst:

- Die Fähigkeit Neues entstehen zu lassen und **sich anpassen können,** in einem sich schnell verändernden Geschäftsumfeld.
- Die Fähigkeit Ressourcen **schnell zu repriorisieren**, wenn sich Anforderungen, Technologie und Wissen verändern.
- Auf Markveränderungen und entstehende Trends **schnell reagieren** können **durch intensiven Kundenkontakt.**
- Verwenden von **evolutionären, inkrementellen** und **iterativen** Methoden, um eine optimale Kundenlösung zu liefern.

Agil sein, mit dem Ziel den Geschäftswert (**Business Value)** zu maximieren, mit der gerade richtigen Lösung, zur richtigen Zeit.

Die Unterschiede zwischen den Agilen Methoden

Betrachtet man die Projektcharakteristik bei den verschiedenen agilen Softwareentwicklungs-Prozessen, so unterscheiden sich diese wesentlich.

Das Gesamtmodell von **FDD** ist ein Festpreisprojekt mit einem vereinbarten Funktionsumfang und Fertigstellungsdatum. Dadurch ermöglicht es eine übersichtliche Auflistung aller Features, welche Teil des Vertrages sind und somit die Kennzahlen, das Budget und die Zeit vorgeben.

Beim Modell **XP** entfällt eine komplette Auflistung aller Anforderungen. Es baut darauf auf, dass mit dem Kunden ein Vertrag mit einem festen Budget und mit einem verhandelbaren Umfang von Features vereinbart wird. Innerhalb des Budget- und Zeitrahmens wird das Projekt umgesetzt, wobei so viele Features entwickelt werden, wie machbar sind. Hier kann die Situation auftreten, dass eventuell benötigte Features nicht entwickelt und ausgeliefert werden, sondern einen Folgeauftrag benötigen.

Scrum bildet einen allgemeinen Managementrahmen für beliebige Projekte. Somit soll Scrum nicht nur für Softwareprojekte genutzt

werden können, sondern auch für anderweitige Projekte. Auch gibt es bei Scrum kein von Beginn an festgesetztes Fertigstellungsdatum. Die Entwickler geben lediglich an, wie viele Einheiten sie pro Iteration entwickeln können und bestimmen dadurch das Zeitmanagement der Entwicklung. Somit hat Scrum, wie auch XP einen auf den Kunden zugeschnittenen Funktionsumfang, der immer wieder angepasst werden kann.

Agil oder Lean?

Toyota und andere Autobauer nutzen einen agilen Entwicklungsprozess, den sie als „Set-Based Concurrent Engineering" bezeichnen. Bei diesem wird eine Vielzahl von Prototypen während des gesamten Entwicklungsprozesses erstellt. Dieses scheinbar ineffiziente System hat Toyota zum schnellsten und effizientesten Autohersteller gemacht. Dieser Entwicklungsprozess kann auch als „lean development" oder „lean innovation" bezeichnet werden. Die „lean" Prinzipien (Fokus auf die Kundennutzenorientierung, kontinuierliche Verbesserung, hohe Qualität, Abfallreduzierung und hohe Integration in einer schlanken Wertschöpfungskette) können auch auf jegliche technische oder Service-Prozesse angewendet werden.

Die Struktur des agilen Projektmanagements

Agiles Projektmanagement besteht aus verschiedenen Merkmalen, die aufeinander aufbauen. Die folgende Abbildung und Beschreibung hilft Ihnen besser zu verstehen, was hinter den Merkmalen: Methoden, Praktiken, Prinzipien, Werten und Einstellung steht und wie sie aufeinander aufbauen.

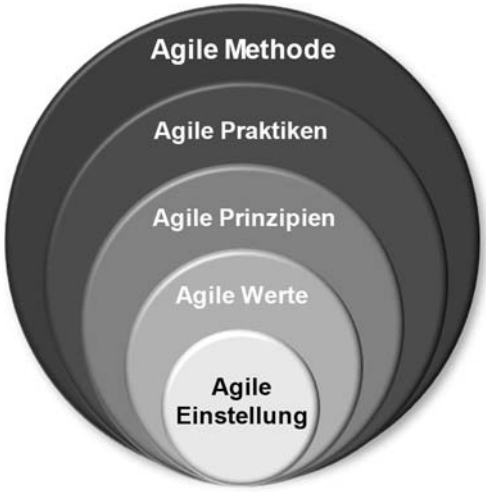


Abbildung 3: Struktur des agilen Projektmanagements

Agile Methoden: Agile Prinzipien und Praktiken werden zu einem strukturierten Vorgehensmodel kombiniert. Dies sind dann Methoden, bzw. Vorgehensmodelle wie z. B. Scrum, XP oder FDD. Sie geben den Agilen Praktiken eine Grundstruktur

Agile Praktiken: (Doing) – Aktivitäten, die benutzt werden, um agile Prinzipien und Werte im Projekt umzusetzen, z.B.: User Stories, Test-Driven Development, Daily Stand-up Meetings, Retrospektiven

erarbeitet, ein Proof of Concept erstellt oder Prototypen gebaut werden und ggf. Architekturentscheidungen getroffen werden.

Vor dem eigentlichen Projektbeginn, z.B. während der Releaseplanung, sollte sich der Product Owner mit den Entwicklern zusammensetzen und über die technische Umsetzung sprechen. Auch wenn er selbst über keinen Entwicklungshintergrund verfügt empfehle ich dies, um zumindest einen ganz groben Überblick zu erhalten.

Es ist wichtig, dass das gleiche Team in Explorationssprints arbeitet, wie dann auch im normalen Sprint, denn nur so können Sie Wissensverlust vermeiden.

Wie viele, und ob überhaupt Explorationssprints für Ihr Projekt sinnvoll sind, hängt von den Risiken und der Wissensbasis ihres Projektes ab. Die folgende Abbildung zeigt Ihnen schematisch eine Releaseplanung mit Vorprojekt.

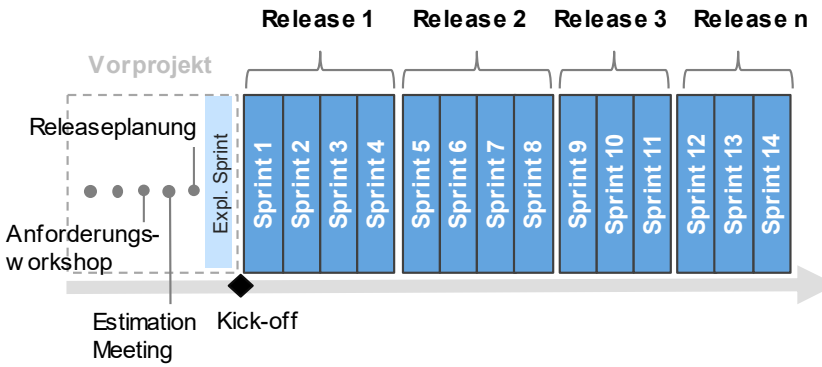


Abbildung 18: Vorprojekt, Projektstart, Releaseplanung

Releaseplanung

Planung ist auch in Scrum eine wichtige Aktivität, damit Sie das Projektziel sicher erreichen. In Scrum wird jedoch nicht wie bei traditionellen Projekten ein detaillierter Plan mit allen Aktivitäten zu Beginn des Projektes erstellt, sondern wie Sie bereits ein paar Seiten vorher gelesen haben, auf drei Ebenen geplant:

- Releaseplanung (Mittelfristplanung)
- Sprintplanung (siehe Seite 139)
- Tagesplanung im Daily Scrum (siehe Seite 159)

Es gibt agile Projekte, die verwenden keinen Releaseplan und planen von Sprint zu Sprint. Dies kann sinnvoll sein für sehr kurze, unkritische Projekte. Normalerweise wird aber ein Releaseplan benötigt, um dem Kunden eine gewisse Sicherheit zu geben, was er bis wann voraussichtlich an Funktionalität bekommt.

Wenn Sie bei agilen Projekten Releases planen, dann werden Ihnen vom Kunden oder Auftraggeber eigentlich immer zwei Frage gestellt:

- Wann wird eine bestimmte Anzahl von Anforderungen geliefert (fester Projektumfang) und welche Kosten entstehen dadurch?
- Wie viel Funktionalität wird bis zu einem bestimmten Zeitpunkt geliefert (fester Zeitpunkt)

Diese Fragen kann der Releaseplan beantworten. Der Product Owner hat die Aufgabe diesen zu erstellen. Damit er dies kann, benötigt er drei Angaben:

1. Die Größe aller Backlog Items (Gesamtaufwand)
2. Die Priorisierung der Backlog Items
3. Die Entwicklungsgeschwindigkeit (Velocity) des Scrum Teams

Wenn diese Punkte bekannt sind, kann der Product Owner berechnen, zu welchem Zeitpunkt welche Funktionalitäten ungefähr fertig sein werden.

Wie Sie den Aufwand der Product Backlog Items bestimmen erfahren Sie im Kapitel „Aufwandschätzung“ ab Seite 127. Die Priorisierung der Backlog Items haben Sie bereits kennengelernt und die Entwicklungsgeschwindigkeit lernen Sie ab Seite 137 kennen.

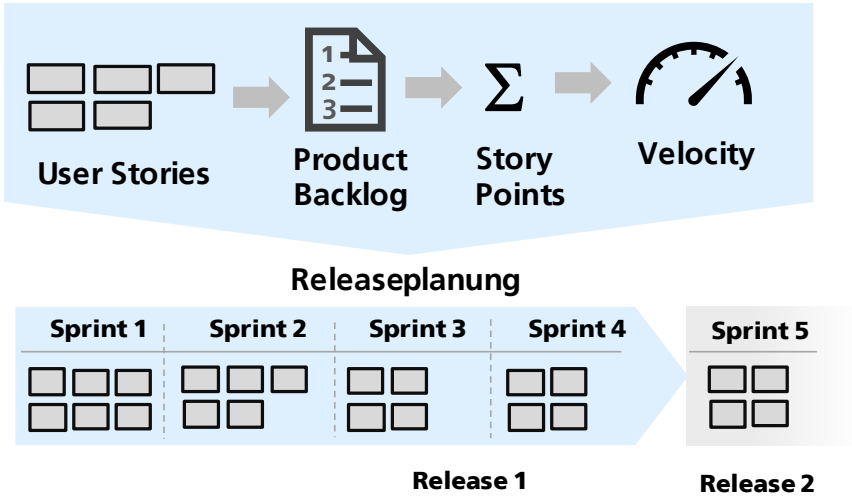


Abbildung 19: Erstellen des Releaseplans

Der initiale Releaseplan kann erstellt werden, sobald ein erstes Product Backlog vorhanden ist. Der Releaseplan zeigt, in welchem Zeit- und Kostenrahmen die bereits definierten Anforderungen voraussichtlich umgesetzt werden und wie viele Sprints dafür notwendig sind.

Der Product Owner ist verantwortlich, dass der Releaseplan erstellt, aktualisiert und kommuniziert wird. Er bestimmt, wann Releases geliefert werden.

In den letzten Jahren wurden die Releases bei Scrum Projekten immer kürzer und es gibt Teams, die haben einen Release nicht nach einigen Sprints, sondern nach jedem Sprint oder sogar innerhalb von

Sprints. Deshalb sollte man den Begriff Releaseplanung in Scrum gemäß Mike Cohen heute vielleicht einfach durch Mittelfristplanung ersetzen (Cohn, 2012).

Dem Kunden eine Übersicht geben

Mit der Releaseplanung geben Sie dem Kunden und den Stakeholdern über den Sprint hinaus eine grobe Übersicht, was in den nächsten Wochen und Monaten an Funktionalität wahrscheinlich an ihn ausgeliefert wird und was dies ungefähr kostet. Dies ist zwar nur eine Prognose, gibt ihm aber trotzdem eine gewisse Sicherheit, bezüglich Terminen und seinem Budget.

Die Releaseplanung basiert auf den bekannten und bereits priorisierten Anforderungen, Epics und Themen, deren geschätzten Aufwand (z. B. in Story Points) und der *durchschnittlichen* Entwicklungsgeschwindigkeit (Velocity) des Development-Teams. Das heißt, die Releaseplanung wird bezüglich Terminen erst genauer, wenn das Development-Team bereits ein paar Sprints gearbeitet hat und die durchschnittliche Entwicklungsgeschwindigkeit bekannt ist. Deshalb wird der Releaseplan periodische überprüft und aktualisiert.

Sobald ein paar Sprints gearbeitet wurde sollte der Product Owner zusammen mit dem Kunden den Releaseplan überprüfen, optimieren und eine grobe Planung für weitere Releases machen, z. B. mit Epics oder Themen. Dies gibt allen Beteiligten eine gewisse Sicherheit, dass man am Richtigen arbeitet und eine Aussicht auf zukünftige Arbeiten.

Release-Management

Der Product Owner erstellt und pflegt den Releaseplan und trägt die fachliche und finanzielle Verantwortung gegenüber dem Auftraggeber. Er entscheidet, welche Funktionen im Releaseplan neu hinzukommen oder wegfallen. Dies stimmt er natürlich mit dem Auftraggeber und den Stakeholdern ab.

Das Product Backlogs ist dynamisch. Die Anforderungen verändern sich mit zunehmendem Wissen und wachsender Erfahrung, aber

auch das Umfeld verändert sich stetig. Genau deshalb wird das Produkt ja mit Scrum inkrementell-iterativ entwickelt. Das bedeutet, alle Anforderungen, die noch nicht umgesetzt sind, können jederzeit den neuen Umständen und Wissen angepasst oder entfernt werden.

Der Product Owner wird beim Releasemanagement auch vom Development-Team unterstützt, wenn es darum geht die Machbarkeit, Kosten und Abhängigkeiten zu anderen Anforderungen und Sprints zu bewerten. Dies sind die Experten für die technische Umsetzung, die auch gewisse Punkte kritisch hinterfragen können. Je länger das Development-Team zusammenarbeitet, desto besser kennt es das Produkt und kann diesbezüglich wertvolle Hilfe leisten.

Ein wichtiges Hilfsmittel für das Scrum Team

Nur mit einem Releaseplan oder einer Mittelfristplanung ist es dem Product Owner möglich das Projekt zu überwachen und zu steuern, Entscheidungen zu treffen und Anforderungen, Termine und Kosten in Einklang zu bringen.

Der Releaseplan gibt auch dem Scrum-Team ein gemeinsames Verständnis, an was in den nächsten Wochen und Monaten gearbeitet wird. Dies hilft den Arbeitsanfall auf das Team gleichmäßig zu verteilen und soll auch Personalengpässe vorbeugen. Der Releaseplan zeigt auch, was für Fachwissen in den nächsten Sprints notwendig ist und ob eventuell neue Development-Teammitglieder notwendig sind.

DevOps – Development and Operations

DevOps und Agile ergänzen sich gegenseitig, mit dem Ziel, funktionierende Funktionalitäten schneller in die Produktion zu bringen.

Das Development-Team produziert am Ende eines jeden Sprints funktionierende Software. Diese muss jedoch oft warten, bis das festgesetzte Releasedatum im Unternehmen erreicht ist. Selbst am Release-Termin kommt es immer wieder zu Verzögerungen, wenn das Ops-Team nicht auf die Integration und Deployment vorbereitet ist oder das Unternehmen nicht bereit ist, mit der neuen Funktionalität

live zu gehen. Kürzeres Time-to-Market, ein wesentlicher Vorteil von Scrum, wird deshalb oft nicht vollständig realisiert.

Das Development-Team muss innerhalb der ersten Sprints nachweisen, dass das Produkt realisierbar ist und Wert schafft. Um dies zu erreichen, benötigen es eine Betriebsumgebung und eine anfängliche Architektur, in der die Ziele der Service Level Agreements erreicht werden. Auch dies wird durch DevOps unterstützt .

Mit DevOps werden die Abteilungen Development und Operations nicht mehr getrennt betrachtet und betrieben, sondern sie werden als ein Team angesehen und arbeiten eng zusammen. Dies über den gesamten Lebenszyklus einer Applikation hinweg – vom Entwurf über die Entwicklung bis hin zum Betrieb einer Applikation – um gemeinsam ein Produkt mit dem größtmöglichen Mehrwert zu schaffen und Software jederzeit und schneller in Betrieb zu nehmen.

Öfteres und schnelleres Deployment wird durch Automatisierung nicht nur im Development (Build, Test), sondern auch im Deployment umgesetzt. Das Ziel ist, möglichst alles zu automatisieren. Dies geschieht in einem agilen Umfeld nach agilen Werten und Prinzipien (Scrum.de, 2015). Daraus ergeben sich kürzere Zyklen, Sie erhalten früheres Feedback der Anwender und sehen auch, ob die Stabilität und die Zuverlässigkeit so sind, wie gedacht.

Menschen, die vorher vielleicht sogar nicht die beste Meinung voneinander hatten, arbeiten zusammen und schätzen die Arbeit der anderen Disziplin als wertvoll ein. Beide profitieren vom Wissen des anderen und bauen gemeinsam ein besseres Produkt. Mehrwert wird schneller, in höherer Qualität und zu niedrigen Kosten geliefert und die Risiken sinken.

Gute, ergänzende Informationen zu DevOps:

https://www.scrumalliance.org/community/articles/2014/april/devops-and-agile

Aufwandschätzung

Aufwandschätzung auf zwei Ebenen

Bei Scrum machen Sie Aufwandschätzung auf zwei Ebenen: Auf der Story-Ebene und auf der Aktivitäten-Ebene. Um dabei gute und zuverlässige Schätzungen zu erhalten, sollte immer das ganze Scrum Team anwesend sein. Wenn Aktivitäten geschätzt werden, ist der Product Owner nicht unbedingt notwendig.

1. Aufwandschätzung auf der Story-Ebene

Vor dem Start des ersten Sprints sollten Sie möglichst viele Backlog Items geschätzt haben, damit Sie einen Releaseplan bzw. die Mittelfristplanung erstellen können. Je mehr Backlog Items Sie geschätzt haben, desto weiter in die Zukunft können Sie Releases planen.

Für diese Schätzung verwenden die meisten Scrum-Teams Story Points. Geschätzt wird im Estimation Meeting. Danach startet der erste Sprint mit dem Sprint Planning.

2. Aufwandschätzung auf Aktivitäten-Ebene

Sobald die Aktivitäten für die Umsetzung jeder Story bestimmt sind, wird im *Sprint Planung 2* der Aufwand für die Aktivitäten bestimmt. Dieser sollte nicht in Story Points oder Task Points geschätzt werden, sondern in Stunden und ist der Aufwand, der das gesamte Team für die Aktivität benötigt.

Das Estimation Meeting

Das Estimation Meeting (auch Backlog Grooming oder Backlog Refinement Meeting genannt) ist das Planungsmeeting mit dem Product Owner, um Klarheit für den nächsten Sprint und die darauf folgenden zu erhalten. Ziel dabei ist, Details von Stories zu klären, Stories zu detaillieren, Aufwände zu schätzen und die Priorisierung der Backlog Items zu überprüfen – aber auch Feedback des Teams über die Backlog Items zu erhalten. Das Estimation Meeting soll aber auch

dem Development-Team eine gewisse Sicherheit und Perspektive für die Zukunft geben.

Das Product Backlog ändert sind während der ganzen Projektdauer. Es kommen immer wieder neue User Stories hinzu und bestehende werden angepasst oder entfallen. Es gehört deshalb zur Aufgabe des Product Owners, zu jedem Sprint Planning mit einem aktuellen, geschätzten und priorisierten Product Backlog zu erscheinen. Damit er das kann, wird immer vor dem Sprint Planning ein Estimation Meeting durchgeführt.

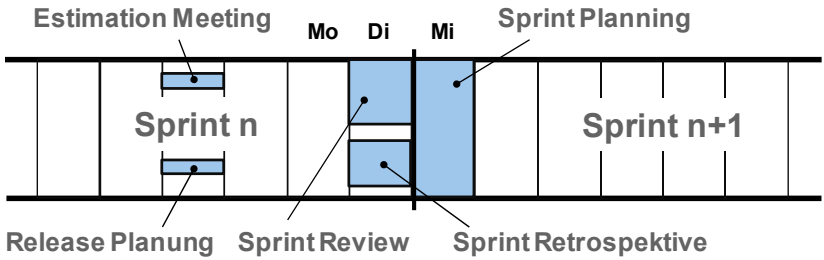


Abbildung 20: Estimation Meeting vor dem Sprint Planning

Das Estimation Meeting hat folgenden Zweck:

1. **Backlog-Pflege (Grooming):** Der Prodcut Owner bespricht mit dem Development-Team die User Stories, die es im nächsten Sprint oder auch im darauf folgenden Sprints voraussichtlich umsetzen wird.

   Epics und große User Stories werden in kleine User Storys aufgebrochen, damit diese im Sprint umgesetzt werden können.

   Manche User Stories werden ganz aus dem Product Backlog entfernt, wenn Aufwand und Nutzen nicht mehr übereinstimmen oder kein Bedarf mehr da ist.

   Feedback des Development-Teams für den Product Owner bezüglich Machbarkeit, technischen Anforderungen und Abhängigkeiten.

2. **Estimation:** Für die geänderten und neuen User Stories wird der Aufwand bestimmt und eventuell zu große Stories aufgebrochen.

Das Estimation Meeting dient hauptsächlich zur Vorbereitung auf das Sprint-Planning. Damit ist gewährleistet, dass dann bereits alle Aufwandschätzung vorhanden sind und das Development-Team die umzusetzenden User Stories schon kennt. Wenn für neue User Stories eine längere Diskussion notwendig ist, dann vertagt der Scrum Master diese in ein eigenes Meeting.

Das Meeting ist aber auch dafür gedacht, das Development-Team über zukünftige Stories zu Informieren und Feedback einzuholen.

Teilnehmer, Organisation und Ablauf des Estimation Meetings

Am Estimation Meeting nimmt das ganze Scrum Team teil. Es kann sinnvoll sein auch externe Spezialisten (z. B. Systemarchitekten, Datenbankentwickler oder Mitglieder von Fachabteilung) hinzuzuziehen. Der Scrum Master sorgt als Moderator für einen reibungslosen Ablauf.

Ich empfehle Ihnen das Estimation Meetings etwa zwei Tage vor dem nächsten Sprint Planning durchzuführen. Dann kann der Product Owner bis zum Sprint Planning die Priorisierung und den Releaseplan nochmals überprüfen und falls notwendig anpassen.

Es hat sich bewährt das Estimation Meeting direkt nach einem Daily Scrum durchzuführen, damit der Tagesablauf des Development-Teams nicht wieder durch ein Meeting unterbrochen und gestört wird (Cohn, 2017). Für einen 4-wöchigen Sprint sollte das Meeting nicht mehr als 2-3 Stunden dauern. Es gibt aber auch Scrum-Teams, die das Estimation Meeting zweimal pro Sprint oder wöchentlich durchführen, dafür ist es dann aber viel kürzer. Gemäß Scrum Guide sollte das Estimation Meeting (Refinement) nicht mehr als 10% der Kapazität des Development-Teams beanspruchen.

Wer schätzt die User Stories?

Ein wichtiger Grundsatz bei Scrum ist: Es schätzt nur das Development-Team, das heißt, diejenigen, die für die Umsetzung der Anforderung verantwortlich sind. Der Scrum Master kann auch mitschätzten, falls er das technische Verständnis hat und das Development-Team ihn dazu einlädt. Beachten Sie, es sollte immer das komplette Development-Team beim Schätzen anwesend sein!

Beim Estimation erklärt der Product Owner die Anforderungen und beantwortet Fragen. Das Development-Team schätzt und der Scrum Master moderiert und stellt sicher, dass das Meeting pünktlich fertig wird.

Die Benefits des Estimation Meetings

- Der Product Owner erhält zusätzlich Hilfe, um das Product Backlog zu optimieren
- Wissen wird an alle Projektbeteiligten weitergegeben und vertieft
- Das Backlog enthält nur wirklich relevante Items
- Fragen und Unklarheiten lassen sich schnell klären (das „Was“ ist geklärt)
- Das Sprint Planning lässt sich vorbereiten (nur noch das „WIE“ ist dann zu klären)
- Risiken werden gemindert, da das Backlog vor dem Sprint Planning diskutiert wird

User Stories sollten aus Prinzip „klein“ sein

Backlog Items sind zu diesem Zeitpunkt ja vom Product Owner bereits priorisiert. Wenn Sie jetzt das Product Backlog anschauen, sehen Sie User Stories in den nächsten zwei Sprints bei denen die Aufwände grösser als „8“ sind? Hier sollte das Scrum Team versuchen diese weiter aufzusplitten. Für Stories die später kommen und grösser sind lohnt sich diese Arbeit nicht. Denn diese sehen bis in ein paar Wochen vielleicht ganz anders aus oder werden gar nie realisiert. Diese Stories können auch grösser sein.

Warum ist das Aufsplitten von User Stories eigentlich wichtig? Die kurze Antwort darauf ist: Damit diese in einen Sprint passen. Aber das ist nur die halbe Wahrheit. Ich zeige Ihnen zwei wichtige Gründe dafür:

1. „The power of small wins“ – Je häufiger Teams ein Fortschrittsgefühl erfahren, auch wenn es in kleinen Schritten geschieht, desto eher wird das Team langfristig kreativ, produktiv und motiviert sein (Amabile, 2011).
2. Stories aufsplitten in Kleinere ist auch wichtig, um das 90%-Syndrom zu vermeiden. Tom Cargill von den Bell Labs hat dies 1985 für die Softwareentwicklung mit der „ninety-ninety rule“ folgendermaßen ausgedrückt:

> *The first 90 percent of the code accounts for the first 90 percent of the development time. The remaining 10 percent of the code accounts for the other 90 percent of the development time*

Hervorgerufen wird dieser Effekt, typischerweise durch die erworbene Kenntnis des Lösungsweges und die Unkenntnis der noch auftretenden Störungen und Probleme. User Stories aufsplitten ist nicht immer einfach – machen Sie sich aber die Mühe!

16 Glossar

Nachfolgend finden Sie eine Übersicht über die wichtigsten Begriffe von Scrum und dem agilen Projektmanagement, die ich in diesem Buch verwendet habe.

Agile Manifesto – Das im Jahr 2001 verabschiedete Manifest ist die Grundlage aller agilen Vorgehensweisen und beschreibt die zentralen agilen Werte.

Agile Prinzipien – Das Agile Manifesto beruht auf 12 Prinzipien, zu deren Einhaltung sich jeder Anwender agiler Methoden verpflichtet.

Agile Werte – Sind die wesentlichen Grundwerte für ein erfolgreiches, agiles Projektmanagement. Zusammengefasst: Mehr Flexibilität, weniger Strukturen, produktivere Zusammenarbeit. (siehe auch Agiles Manifesto)

Akzeptanzkriterien – Jedes Backlog Item (User Story) hat Akzeptanzkriterien. Diese sind Bestandteil der Fertigstellungskriterien (Definition of Done) und werden durch den Product Owner beim Review der fertigen User Story verwendet.

Anforderungen – Anforderungen werden in Scrum üblicherweise in einer User Story definiert (verpackt). Deshalb spricht man meistens von User Stories und nicht von Anforderungen.

Anforderungsliste – siehe Product Backlog

Artefakt – Artefakte sind die Ergebnisse von Aktivitäten im Softwareentwicklungsprozess. Zentrale Scrum Artefakte sind: Product Backlog, Sprint Backlog und das (auslieferbare) Inkrement.

Benutzergeschichte – siehe User Story

Burndown-/Burnup-Chart: Eine Grafik mit der das Development-Team die noch verbleibende Arbeit im Sprint in Relation zu der noch verbleibenden Zeit im Sprint visualisiert. Sie zeigt den Sprint-Fortschritt. Sie kann auf der Projekt-, Release- und Sprint-Ebene eingesetzt werden

Business Value – Er ist der (meist finanzielle) Wert eines Backlog Items für ein Unternehmen. Das Product Backlog wird immer wieder so sortiert, dass Items mit dem höchsten Geschäftswert als Erstes erledigt werden.

Commitment – (Verpflichtung) Das Development-Team verpflichtet sich im Sprint Planning eine „Prognose" zu liefern, welche Backlog Items im nächsten Sprint umgesetzt werden.

Daily Scrum – Ist ein Meeting von 15 Minuten, innerhalb dem das Development-Team seine Aktivitäten synchronisiert und an der Planung für die nächsten 24 Stunden arbeitet. Es wird oft als Stand-up Meeting durchgeführt.

DEEP – Ein gute Product Backlog ist gemäß Mike Cohen DEEP. Dies bedeutet: Detailed, Estimated, Emergent (entwickelt sich), Prioritized.

Definition of Done (DoD) – Ist die Einigung eines agilen Teams darauf, was getan werden muss, damit ein Feature als fertig angesehen wird. Die DoD ist eine Liste von Fertigstellungskriterien.

Definition of Ready (DoR) – Ist eine Liste von Kriterien, die an die Product Backlog Items gestellt werden, damit diese bereit (Ready) sind, um im Sprint daran zu arbeiten. Im Gegensatz zur DoD ist der Product Owner für die Einhaltung der DoR verantwortlich.

DevOps – Durch die enge Zusammenarbeit von *Development and Operations* im agilen Umfeld und durch Automatisierung im ganzen Lebenszyklus können Releases schneller, in höherer Qualität, zu niedrigen Kosten und Risiken geliefert werden.

Dienende Führung – (Servant Leadership) Der Scrum Master ist dienender Führer für das Scrum Team. Das heißt, er verfügt zwar über Einfluss, hat aber keine Autorität bezüglich der Arbeitsorganisation im Team. Er agiert als Coach und als Change Agent.

Dokumentation – Diese ist auch in agilen Projekten notwendig und hilfreich. Es gelten jedoch vorrangig die Grundsätze des Agilen Manifests.

Done – In Scrum wird unterscheiden zwischen »nicht fertig« und „fertig". Ein Backlog Item gilt als „Done", wenn alle Tasks erledigt sind und die Definition of Done erfüllt ist.

Development-Team – Das Development-Team führt alle Arbeiten aus, um die Anforderungen in ein auslieferbares Produkteinkrement umzusetzen. Es organisiert seine Arbeit selbst.

Entwicklungsgeschwindigkeit (Velocity) – Ist die Summe aller Aufwände (Story Points), die das Scrum Team in einem Sprint umgesetzt hat. Über die Zeit pendelt sich die Entwicklungsgeschwindigkeit auf einen relativ stabilen Wert ein. Die Entwicklungsgeschwindigkeit zu kennen ist für den Product Owner wichtig, damit er den Releaseplan erstellen und im diesen im Verlauf des Projektes justieren kann.

Epic – Ist eine grobgranulare Anforderung (große User Story), die zu groß oder zu komplex ist, um Sie in einem Sprint umzusetzen. Sie wird später, wenn die Kundenanforderungen klarer und mehr Details bekannt sind, im Product Backlog aufgeteilt in Stories.

Estimation – Damit ist meistens das Schätzen des Aufwandes von Product Backlog Items, bzw. User Stories gemeint. Schätzmethoden sind z.B. Planning Poker und Magic Estimation.

Estimation Meeting – Das Estimation Meeting (auch Backlog Grooming oder Backlog Refinement genannt) ist das Planungsmeeting mit dem Product Owner, um Klarheit für den nächsten Sprint zu erhalten. Ziel dabei ist: Details zu klären, Aufwände zu schätzen und die Priorisierung zu überprüfen.

Event (Ereignis) – Siehe Scrum Event.

Extreme Programming – Ist eine agile Softwareentwicklungsmethode, die mehrheitlich von Kent Beck entwickelt wurde. Sie folgt

den fünf zentralen agilen Werten: Kommunikation, Einfachheit, Respekt, Feedback und Mut.

Feature — Ist eine funktionale Eigenschaft des zu entwickelnden Produktes.

Geschäftswert – siehe Business Value

Hindernis (Impediment) – Sind Probleme jeglicher Art, welche die Development-Mitglieder bei der effektiven Ausführung ihrer Arbeit behindern.

Impediment – siehe Hindernis

Impediment Backlog – Ist eine öffentliche im Teamraum aufgehängte Arbeitsliste des Product Owners mit allen Hindernissen.

Increment – Ein Increment ist ein Teil eines Ganzen. Unter einem Product Increment im Scrum versteht man das Ergebnis eines Sprints, als fertiges Teilstück des Gesamtproduktes.

Inspect & Adapt – Ein Prinzip der permanenten Verbesserung. Die eigene Arbeit wird in regelmäßigen Abständen überprüft und das Vorgehen in der nächsten Iteration angepasst.

INVEST – Gute User Stories entsprechen INVEST-Kriterien. Sie sind: (I) Independend *unabhängig*, (N) Negotiable *verhandelbar*, (V) Valueable *nützlich*, (E) Estimateable *schätzbar*, (S) Small *klein*, (T) Testable *testbar*

Iteration – Ein einzelner Zeitabschnitt, in dem ein Inkrement des Produktes entwickelt wird. In Scrum wird dies „Sprint" genannt.

Iterativ-inkrementell – In Scrum wird diese Technik verwendet, um durch kurze Feedbackzyklen Software zu erstellen. Iterativ (in einzelnen Sprints), werden Inkremente (fertige Produktteile) geliefert.

Kunde – Ist derjenige, der das Produktergebnis beauftragt (der Auftraggeber) und bestimmt, was dieses leisten soll und auch dafür bezahlt.

Lean Management – Ist eine Management-Philosophie, die Denkprinzipien, Methoden und Verfahrensweisen zur effizienten Gestaltung der gesamten Wertschöpfungskette von Produkten umfasst.

Magic Estimation –Ähnliche Aufwandschätz-Methode für User Stories wie Planning Poker®, jedoch einiges effizienter.

Manager – (Linien-) Manager sind wichtige Stakeholder auch bei Scrum-Projekten. Sie können das Projekt unterstützen und Hindernisse beseitigen, aber auch ein Umfeld schaffen, indem agile Prinzipien und Projekte sich entwickeln können.

Minimal Marketable Feature (MMF) – Ist die kleinstmögliche Menge an Funktionen, die für sich genommen vermarktbar ist. Software, die nur eines dieser Merkmale aufweist, hat einen Nutzen für den User, für den dieser bezahlen würde.

Minimal Viable Product (MVP) – Das Minimal Viable Product ist die minimale Menge von Features, die notwendig sind, um herauszufinden, was ein Kunde möchte. Es ist eine Strategie, um schnelles und breites Feedback vom Kunden zu erzielen, ohne das Produkt bis zu seiner Marktreife auszuarbeiten.

MuSCoW – Die MuSCoW-Methode ist eine Technik zur Priorisierung, um Backlog Items grob in vier Kategorien einzuordnen: Must have, Should have, Could have, Won't have.

PDCA-Zyklus – Ein vierteiliger Problemlösungsprozess, der zur ständigen Verbesserung bei der Softwareentwicklung verwendet wird: Plan (P), Do (D), Check (C) und Act (A)

Planning – siehe Sprint Planning

Planning Poker® – ist eine Technik für die Aufwandsschätzung von Backlog Items im Estimation Meeting oder Backlog Grooming. Dabei wird besonders Wert darauf gelegt, dass die Grösse eines Features geschätzt wird und nicht der Aufwand.

Product Backlog – Das Product Backlog eine Liste von Anforderungen (User Stories), die im laufenden Projekt umgesetzt werden sollen. Sie wird vom Product Owner priorisiert, bereitgestellt, verantwortet und gemeinsam mit dem Development-Team gepflegt.

Product Increment – Ist lauffähige, getestete und dokumentierte Software, die Anforderungen aus dem Product Backlog umsetzt. Das Product Increment ist das Ergebnis eines Sprints.

Product Owner – Der Product Owner ist für den wirtschaftlichen Erfolg und das „Was" im Scrum-Projekt verantwortlich. Er plant und

steuert die Entwicklung in Scrum. Er ist Owner des Product Backlogs, priorisiert es und legt damit fest, welche Features zuerst realisiert werden.

Product Backlog Item (PBI) – Dies ist eine Anforderung, üblicherweise beschrieben in einer User Story. Es ist eine Arbeitseinheit, die klein genug ist, damit das Scrum-Team diese in einem Sprint abschließen kann.

Product Backlog Grooming – So wird die laufende Pflege des Product Backlog bezeichnet. Alternativ Backlog Refinement genannt.

Produktvision – Die Produktvision wird vom Product Owner vor dem ersten Sprint erstellt. Sie leitet das Scrum in die richtige Richtung und ist ein übergeordnetes Ziel, dass jeder teilen muss, das Scrum Team, der Kunde und die Stakeholder. Die Vision beschreibt, warum das Projekt unternommen wird und was der erwünschte Endzustand ist.

Prognose – Das Development-Team, liefert im Sprint Planning eine Prognose, welche Backlog Items es im nächsten Sprint umsetzt.

Release – Ist eine Softwareversion, die vom Anwender produktiv eingesetzt werden kann. Der Release ist das Ergebnis des Scrum-Projektes.

Release-Burndown-Chart – Dies ist ein Burndown-Chart, das für jeden Zeitpunkt eines Projekts anzeigt, wie viele Story Points bis zu einem Release noch umzusetzen sind.

Releaseplanung – Ist die Mittelfristplanung des Projektes und gibt dem Kunden eine Übersicht, wann er welche Funktionalitäten geliefert bekommt.

Scrum Ereignisse (Events) – Als Scrum Ereignisse werden fünf Arbeitstreffen des Scrum Teams bezeichnet: Sprint, Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospektive.

Scrum Guide – Wird herausgegeben von Ken Schwaber und Jeff Sutherland und beschreibt die offiziellen Scrum Grundlagen. Die neuste Version ist vom November 2017.

Scrum of Scrums – Bei grossen Projekten ist dies ein Meeting, das der Koordination von mehreren Scrum-Teams dient. Scrum Teams müssen sich untereinander synchronisieren, besonders dann, wenn

funktionale Abhängigkeiten bestehen. Jedes Team entsendet nach dem Daily Scrum einen Vertreter an das tägliche Scrum of Scrums. Der Ablauf ist gleich wie beim Daily Scrum.

Scrum Master – Der Scrum Master ist eine Scrum-Rolle. Er hilft, Scrum richtig anzuwenden, unterstützt das Team und stellt die direkte Zusammenarbeit zwischen Product Owner und Development-Team sicher. Der Scrum Master beseitigt Hindernisse und hilft dem Team, seine Produktivität kontinuierlich zu steigern.

Scrum-Rollen – Sind: Product Owner, Scrum Master, Development-Team

Scrum-Team – Besteht aus den drei Scrum Rollen: Product Owner, Scrum Master, Development-Team

Scrum-Werte – Scrum basiert auf fünf Werten: Verpflichtung, Mut, Fokus, Offenheit und Respekt. Wenn diese Werte durch das Scrum Team verkörpert und gelebt werden, werden die Scrum-Säulen *Transparenz, Überprüfung* und *Anpassung* lebendig und bauen bei allen Beteiligten Vertrauen zueinander auf.

Selbstorganisation – Ist ein wesentliches Merkmal von Scrum-Teams. Jedes einzelne Teammitglied übernimmt Verantwortung für den Gesamterfolg. Entscheidungen werden im Team getroffen, das Team einigt sich auf teaminterne Regeln und Arbeitsweisen. Einzig die Erreichung des Sprint-Ziels ist ein Maßstab für das Team.

Selected Product Backlog – Im ersten Teil des Sprint Plannings wählt das Development-Team aus dem priorisierten Product Backlog die Items aus, die es im kommenden Sprint erledigen kann. Das Selected Product Backlog entsprich dem Sprint Backlog

Servant Leadership – Der Scrum Master verkörpert die Prinzipien der „dienenden Führung". Servant Leadership beschreibt die grenzenlose Selbstverpflichtung der Führungskraft gegenüber dem Team bzw. der Organisation.

Sprint – In Scrum werden Iterationen als Sprint bezeichnet. Ein Sprint hat eine feste Dauer, in welcher das Development-Team die Anforderungen des Kunden in funktionsfähige Software umsetzt.

Sprint Backlog – Im Sprint Planning 1 entscheidet das Development-Team, wie viele der am höchsten priorisierten Backlog Items es in

den folgenden Sprint nehmen will. Die gewählten Items sind dann das Sprint Backlog für den folgenden Sprint. Das Sprint Backlog entspricht der *Forecast* des Development-Teams, was es im startenden Sprint gemeinsam erreichen will.

Sprint Burndown Chart – siehe Burndown Chart

Sprint Planning Meeting – Das Sprint Planning Meeting findet zu Beginn eines Sprints statt und besteht aus zwei Teilen. Beide Meetings dauern jeweils max. vier Stunden.
Im ersten Teil definiert das Scrum-Team das Sprint Backlog, also WAS es im nächsten Sprint umsetzen will. Im zweiten Teil werden die Implementierungsdetails, das WIE geklärt. Am Ende des Sprint Plannings sind alle User Stories (WAS) und die dazugehörigen Tasks auf dem Taskboard dargestellt.

Sprint Retrospektive – Wird nach jedem Sprint-Review durchgeführt. Der Sprint und die Geschehnisse werden analysiert und es werden Maßnahmen für die nächsten Sprints definiert, mit dem Ziel stetig zu lernen und besser zu werden.

Sprint-Review – Findet am Ende des Sprints statt, mit dem Ziel die entstandenen Arbeitsergebnisse dem Product Owner, dem Kunden und den Stakeholdern vorzustellen und Feedback zu erhalten. Der Scrum Owner überprüft dabei, ob das Development-Team alle User Stories, gemäß Prognose, umgesetzt hat.

Sprint-Abbruch – Der Product Owner kann einen Sprint abbrechen, wenn er sieht, dass das Sprint-Ziel entweder nicht erreicht werden kann oder inzwischen nicht mehr sinnvoll ist. In diesem Fall wird ein Sprint-Review durchgeführt und alle abgeschlossenen „Done" Product Backlog-Einträge begutachtet und wenn brauchbar vom Product Owner abgenommen.

Sprint-Ziel – Das Sprintziel wird vom Product Owner definiert und beschreibt das auszuliefernde Produktinkrement – möglichst in einen Satz. Es ist das, was der Product Owner am Ende des Sprints an den Kunden ausliefern will.

Stakeholder – (Interessensvertreter) werden Personen bezeichnet, die vom Projektergebnis betroffen oder daran interessiert sind, aber

nicht am Projekt aktiv mitwirken, z.B. Anwender, Kunde, Vertrieb, Marketing, Manager.

Sprint 0 – Wird oft initialer Sprint genannt, ist meist kürzer als ein normaler Sprint und produziert keinen Business Value. Hier wird z.B. das Product Backlog vorbereitet, die Infrastruktur aufgesetzt, Architekturfragen geklärt und Regeln definiert, wie das Scrum Team zusammenarbeitet.

Story – siehe User Story

Story Card – Beschreibt ein Backlog Item (User Story) auf einer Karteikarte mit max. zwei Sätzen. Auf der Vorderseite steht zusätzlich der Business Value und der Aufwand in Story Points. Auf der Rückseite stehen z.B. noch Abnahmekriterien und die Teststrategie.

Story Point – Story Points sind eine abstrakte Einheit, mit welcher der Aufwand (Grösse) eines Backlog Items bewertet wird, immer relativ zur einer anderen Story. Der Aufwand kann nur bestimmte vordefinierte Werte annehmen, die meistens der Fibonacci-Sequenz entsprechen.

Task – Im Sprint Planning 2 definiert das Development-Team die Tasks (Aktivitäten, Aufgaben) mit denen die User Stories umgesetzt werden. Diese werden dann auf dem Taskboard neben den User Stories befestigt.

Taskboard – Das Taskboard visualisiert das Sprint Backlog. Darauf lässt sich jederzeit erkennen, welche Product Backlog Einträge für den Sprint ausgewählt wurden, welche Aufgaben (Tasks) dazu zu bearbeiten sind, und in welchem Bearbeitungszustand diese Aufgaben sind.

Team – siehe Scrum-Team

Timebox – Alle Scrum-Ereignisse haben eine zeitliche Beschränkung (Time Box). Das Ereignis wird nach einer bestimmten Zeit beendet, auch wenn diese noch nicht inhaltlich abgeschlossen ist. Jedes Scrum Meeting hat z.B. eine festgelegte Timebox. Die Timebox hat zum Ziel, Verschwendung zu vermeiden, um sich auf die wirklich wichtigen Dinge im Sprint zu fokussieren.

User – Als User bzw. (End-) Nutzer bezeichnet man die Personen, die das Projektergebnis nutzen. Scrum stellt das Liefern von Funktionalität für einen User in den Vordergrund, nicht die Erfüllung von Aufgaben. Es ist eine bewährte Vorgehensweise, die Backlog Items aus Sicht des Nutzers in einer User Story zu beschreiben.

User Story – Auch Product Backlog Items (PBI), bzw. Anforderungen oder Features genannt, werden meistens in User Stories beschrieben. User Stories beschreiben Anforderungen aus der Sicht des Anwenders. Verfasst werden User Stories in einer speziellen Schreibweise: Als <Rolle> möchte ich <Ziel/Wunsch>, um <Nutzen/Grund> erreichen.

Velocity – siehe Entwicklungsgeschwindigkeit

Vision – siehe Produktvision

Werte – siehe Scrum-Werte

17 Anhang

Internet-Links und empfohlene Literatur

Auf der Internetseite http://www.rolandwanner.ch finden Sie eine Liste mit Links und Artikel zum agilen Projektmanagement, Projektcontrolling, Earned Value Management und Risikomanagement in Projekten.

Auf meinem Blog http://www.rolandwanner.ch/blog finden Sie interessante Artikel zum agilen Projektmanagement, Projektcontrolling, Earned Value Management und Risikomanagement in Projekten.

Folgende ergänzenden Scrum Bücher empfehle ich:

Scrum in der Praxis: Erfahrungen, Problemfelder und Erfolgsfaktoren, Sven Röpstroff, Robert Wiechmann, dpunkt.verlag, 2015

Scrum: Produkte zuverlässig und schnell entwickeln, Boris Gloger, Hanser Verlag, 2016

Agiles Produktmanagement mit Scrum: Erfolgreich als Product Owner arbeiten, Roman Pichler, dpunkt.verlag, 2013

Folgende Blogs empfehle ich zu verfolgen:

Mike Cohn's Blog, https://www.mountaingoatsoftware.com/blog

Über den Autor

Roland Wanner ist schon seit über 30 Jahren im Projektgeschäft tätig und hat schon viele Projekte miterlebt – erfolgreiche und gescheiterte. Nach seiner Ausbildung als Maschineningenieur und Wirtschaftsingenieur war er zuerst 5 Jahre als Projektleiter und dann mehrere Jahre als Projektcontroller und Projektportfolio-Manager im Maschinen- und Anlagenbau tätig. Seit mehr als 10 Jahren arbeitet er als Projektmanagement-Spezialist, Projektportfolio-Manager und Project Office Manager im Banken- und Versicherungsbereich. Dort unterstützte er verschiedene Softwareprojekte, welche Scrum und agiles Projektmanagement angewendet haben.

Auf seinem Blog http://www.rolandwanner.ch/blog finden Sie interessante Artikel über agiles Projektmanagement, Projektcontrolling, Earned Value Management und Risikomanagement in Projekten.

Ihre Meinung ist uns wichtig!

Herzlichen Dank, dass Sie dieses Buch gekauft haben. Wir haben unser Bestes gegeben, beim Inhalt wie auch bei der Aufmachung. Es wurde viel Aufwand geleistet, um dieses Buch so vollständig und korrekt wie möglich zu machen. Es ist jedoch nicht ganz auszuschließen, dass uns an der einen oder anderen Stelle des Buches ein Missgeschick unterlaufen ist, ob inhaltlich oder in der Rechtschreibung. Vielleicht vermissen Sie auch bestimmte Informationen oder sind der Meinung, gewisse Themen sollten vertieft werden, oder sind bei gewissen Themen anderer Meinung. Wir sind auf Ihre Meinung angewiesen!

Für Ihre Ideen, Gedanken und Korrekturvorschläge bedanken wir uns ganz herzlich. Senden Sie diese bitte an: info@rolandwanner.ch

Literaturverzeichnis

Amabile, T. (2011). *The Progress Principle: Using Small Wins to Ignite Joy, Engagement, and Creativity at Work.* Harvard Business Review Press.

Cohn, M. (2012). *Release Planning: Retiring the Term but not the Technique*. Von https://www.mountaingoatsoftware.com/blog/release-planning-retiring-term-not-technique abgerufen

Cohn, M. (2014). *Agile Estimating and Planning.* Prentice Hall.

Cohn, M. (2016). *Don't Estimate the Sprint Backlog Using Task Points*. Von https://www.mountaingoatsoftware.com/blog/dont-estimate-the-sprint-backlog-using-task-points abgerufen

Cohn, M. (2017). *Planning Poker Cards: Effective Agile Planning and Estimation*. Von https://www.mountaingoatsoftware.com/tools/planning-poker abgerufen

Cohn, M. (2017). *When Should We Estimate the Product Backlog*. Von https://www.mountaingoatsoftware.com/blog/when-should-we-estimate-the-product-backlog abgerufen

Dräther, R. (2013). *Scrum kurz & gut.* O'Reilly.

Gloger, B. (2016). *Scrum - Produkte zuverlässig und schnell entwickeln.* Hanser.

Gloger, B. (2017). *Was ist Scrum*. Von https://borisgloger.com/scrum/ abgerufen

Greaves, K. (2012). *Release Planning with Scrum*. Von https://www.growingagile.co.za/2012/10/release-planning-with-scrum/ abgerufen

Joas, H. (1999). *Entstehung der Wert.* suhrkamp.

Kerth, N. (2017). *The Retrospective Prime Directive*. Von http://retrospectives.com/pages/retroPrimeDirective.html abgerufen

Pichler, R. (2011). *The Product Vision Board*. Von https://www.romanpichler.com/blog/the-product-vision-board/ abgerufen

Pichler, R. (2013). *Scrum - Agiles Projektmanagement erfolgreich einsetzen.* dpunkt.Verlag.

Schwaber, K. (2004). *Agile Project Management with Scrum.* Microsoft Press.

Scrum.de. (2015). *Scrum.de*. Von https://www.scrum.de/wir-brauchen-devops/ abgerufen

Wintersteiger, A. (2012). *Scrum Schnelleinstieg.* entwickler.press.

Stichwortverzeichnis

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

Z