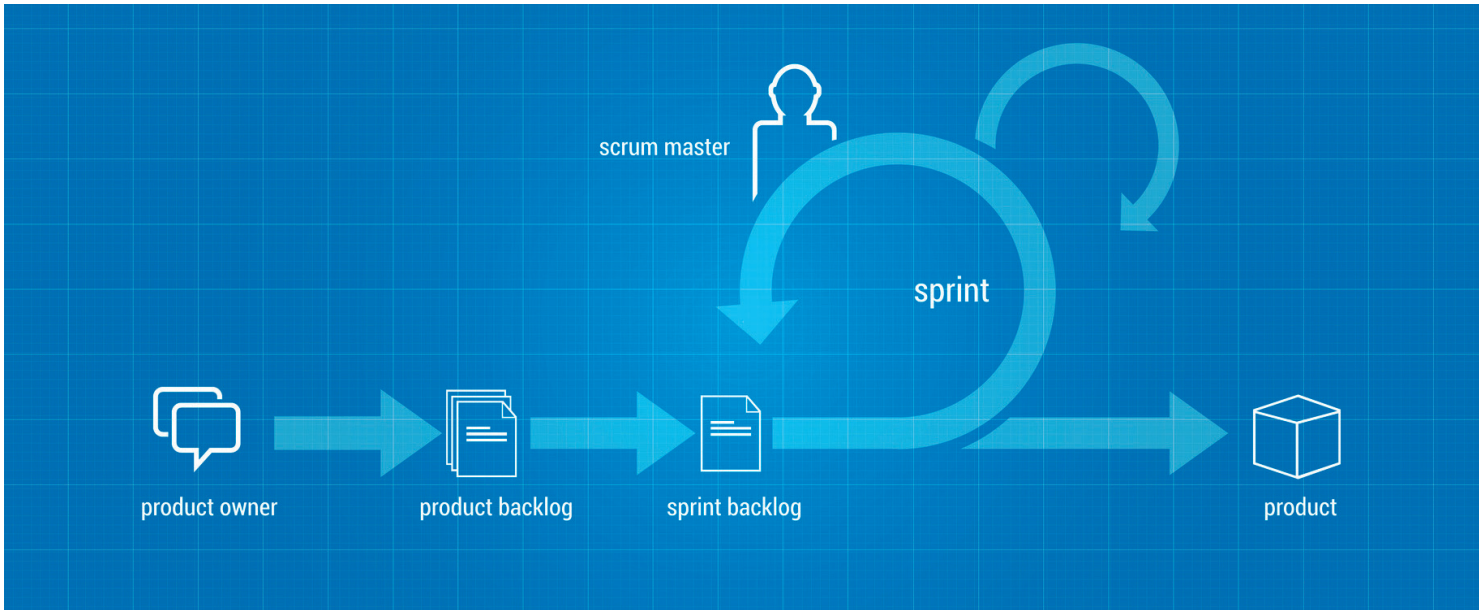

Erfahrungen agiler Softwareentwicklung

H&D International Group

Inhalt

Einleitung

Beim Start eines neuen Softwareentwicklungsprojektes liegt das vom Kunden gewünschte Ergebnis in einer Wolke. Die Anforderungen existieren meistens nur in den Köpfen der Fachbereiche und sind für Außenstehende häufig widersprüchlich, unklar oder chaotisch. Alle Projektbeteiligten, egal ob Fachbereich, Projektleiter oder Entwickler, wissen nur grob, wie das eigentliche Ergebnis aussehen soll. Es ist unscharf und schwer zu greifen. Deshalb werden notgedrungen Annahmen getroffen oder Dinge vereinfacht, die allerdings nur in Teilen korrekt sind und sich im späteren Verlauf als schlichtweg falsch erweisen können.

Auf Basis dieser Annahmen und Vereinfachungen werden mittels des Wasserfallmodells oder des V-Modells Lastenhefte geschrieben, Pflichtenhefte erstellt, DV-Konzepte spezifiziert, Software-Architekturen entworfen und Oberflächen designt. Es werden Projektpläne erstellt und Meilensteine für die aktuelle und anschließende Projektphasen definiert, um den geplanten Fertigstellungstermin möglichst sicher bestimmen zu können.

Nachdem viele Dokumente vom Dienstleister erstellt und vom Kunden abgenommen worden sind, beginnt schließlich die Phase der Softwareerstellung oder Realisierung. Erst in dieser Phase erhält der Kunde einen realistischen Eindruck über das zu erwartende Ergebnis. Wichtig ist hier der Begriff realistisch, denn auch mit Prototypen und Oberflächenentwürfen kann immer nur ein Teil des Ergebnisses dargestellt werden und nie das große Ganze. Die Erfahrung zeigt, dass einzelne Teile für sich genommen zwar performant und intuitiv zu bedienen sein mögen, sich im Zusammenspiel mit anderen Teilen jedoch als langsam und nicht benutzbar erweisen.

Spätestens mit Beginn der Realisierung erwartet der Kunde einen vertraglich zugesicherten Termin für die Fertigstellung des Projektes. Oft gibt er den Termin, aufgrund von eigenen Terminzusagen, selbst vor oder definiert zumindest den erwarteten Funktionsumfang. Dabei wird jedoch häufig vergessen, dass sich die Anforderungen seit der Entwurfsphase geändert haben können bzw. der Entwurf nicht immer mit den Wünschen des Kunden übereinstimmt.

Das Problem

Kurzfristig auf Änderungswünsche zu reagieren, um den Erfolg des Projekts sicherzustellen, ist nur schwer bis gar nicht möglich.

Softwareentwicklungsprojekte zeichnen sich durch individuelle Vorhaben mit individuellen Herausforderungen aus. Von Projekt zu Projekt ändern sich Kunde, Projektthema oder Entwicklungsteam. Dadurch ist jedes Projekt in seiner Form einzigartig.

Mittels des Wasserfallmodells oder des V-Modells wurde in der Vergangenheit versucht, ein Vorgehensmodell zu finden, welches diese Einzigartigkeit berücksichtigt und die Besonderheiten bestmöglich unterstützt. Dabei hat sich gezeigt, dass folgende Herausforderungen nur schwer oder gar nicht gelöst werden können.

Anpassungen in der Software fordern mehr Zeit und Budget.

Änderungen von Anforderungen sind nie auszuschließen, egal in welcher Phase sich das Softwareentwicklungsprojekt befindet. Der Kunde stellt fest, dass er bei den Anforderungen nachbessern muss, um den Erfolg des Projekts sicherzustellen. Dies kann aufgrund von strategischen Entscheidungen erfolgen oder durch geänderte Rahmenbedingungen wie zum Beispiel Gesetzesänderungen. Je früher Änderungen identifiziert werden, desto besser kann darauf reagiert werden, da sich die Anpassungen in einem frühen Stadium des Entwicklungsprojektes meistens noch in Grenzen halten.

Was hat es jedoch zur Folge, sollte eine wichtige Anpassung erst in der Softwareerstellungsphase bemerkt werden? Alle Dokumente, die von den Anpassungen betroffen sind, müssen geändert werden. Zusätzlich muss der relevante, bereits umgesetzte Teil der Software erneuert werden.

Häufig werden diese Anpassungen als Vorwand dazu genutzt, den Termin entsprechend nach hinten zu schieben und zusätzlich weiteres Budget für die erforderlichen Änderungen zu verlangen. Manchmal verzichtet der Kunde auf weniger wichtige, noch nicht realisierte Anforderungen, um die wichtigeren Änderungen umsetzen zu lassen und den Terminplan halten zu können.

Die Herausforderung besteht darin, das vom Kunden gewünschte Ergebnis sicherzustellen.

Eine noch weitaus größere Herausforderung ist die Sicherstellung des gewünschten Projektergebnisses aus Kundensicht. Wie kann der Kunde sicher sein, dass die vom Dienstleister entwickelte Software genau seinen Anforderungen entspricht? Der Kunde, in Form des Fachbereichs, kann in der Regel erst mit der Fertigstellung der Software sicher beurteilen, ob das Ergebnis seinem Bedarf entspricht. Konzepte über Softwarearchitekturen mit UML-Diagrammen und Texten mit technisch überfluteten Begriffen kann und will er häufig nicht verstehen. Erst anhand des Endergebnisses kann die Korrektheit von komplexen Algorithmen und Prozessen bewertet werden. Dies gilt auch für die Benutzeroberfläche und das Antwortverhalten.

Wie können diese Probleme gelöst werden? Gib es Alternativen zu den oben genannten traditionellen Vorgehensmodellen?

Die Lösung

Durch Scrum kann eine agile Softwareentwicklung mit nur wenigen einfachen Regeln erfolgreich betrieben werden. Scrum ermöglicht die kurzfristige Reaktion auf Anforderungsänderungen und stellt sicher, dass der Kunde frühzeitig sieht, welches Ergebnis er vom Dienstleister erhält.

Um die genannten Herausforderungen zu meistern, ist eine agile Softwareentwicklung mittels Scrum eine erfolgreiche Lösung.
Scrum ist als Rahmenwerk zu verstehen, welches aus drei Rollen, fünf Ereignissen und drei Artefakten besteht. Da Scrum mit nur sehr wenigen Regeln auskommt, wird es bewusst nicht als ein Vorgehensmodell bezeichnet. Die Regeln sind im Scrum Guide[1] oder Agile Atlas[2] definiert.

Scrum beruht auf der Erfahrung, dass viele Softwareentwicklungsprojekte zu komplex sind, um sie mit einem vollumfassenden Masterplan erfolgreich durchführen zu können. Deshalb verfolgt Scrum einen empirischen Ansatz, der auf drei Säulen basiert:

1. Transparenz (transparency): Alle am Ergebnis des Projekts beteiligten Personen haben einen Einblick in Fortschritte und Hindernisse des Projektes.
2. Überprüfung (inspection): In regelmäßigen Abständen werden die Artefakte und der Fortschritt in Bezug auf das Ziel überprüft, um unerwünschte Abweichungen zu erkennen.
3. Anpassung (adaptation): Wenn bei der Überprüfung festgestellt wird, dass das resultierende Produkt nicht akzeptabel sein wird oder dass Prozesse von festgelegten Grenzwerten abweichen, müssen Anpassungen vorgenommen werden, um zukünftige Abweichungen zu minimieren. Die Anpassungen betreffen sowohl das Produkt als auch die Prozesse. Beide werden durch dieses Vorgehen kontinuierlich verbessert.

In Scrum werden die Anforderungen an die Software aus der Sicht des Anwenders formuliert und iterativ in festen Intervallen, den sogenannten Sprints, umgesetzt. Am Ende eines jeden Sprints steht die Fertigstellung eines Teilprodukts, das potentiell an den Kunden ausgeliefert werden kann (potentially shippable product). Im Anschluss jeder Iteration, basierend auf dem empirischen Ansatz der drei Säulen, werden Produkt und Prozesse überprüft und im nächsten Sprint weiterentwickelt.

In den nächsten Kapiteln werden die Elemente von Scrum, also der Sprint, die drei Rollen, die fünf Ereignisse und drei Artefakte, genauer erklärt.

Der Sprint

Unter „Sprint“ werden in Scrum Arbeitsphasen verstanden, die jeweils drei bis fünf Wochen dauern.

Der Sprint ist das Herzstück von Scrum und steht für ein festes Intervall. Er beginnt mit einem „Sprint Planning“ und endet mit einem „Sprint Review“ sowie einer „Sprint Retrospective“. Sprints besitzen ein festes Zeitfenster, das in der Regel auf drei bis fünf Wochen ausgelegt ist. Sie folgen unmittelbar aufeinander und sollten idealerweise die gleiche Länge haben. Ein Sprint sollte niemals verlängert werden. Ein vorzeitiger Abbruch kann nur durch den Product Owner erfolgen. Gründe hierfür wären beispielsweise, dass das Entwicklungsteam die Komplexität des Sprints falsch eingeschätzt hat oder dass strategische Anforderungsänderungen und Umpriorisierungen stattgefunden haben. In diesem Fall wird der aktuelle Sprint mit einer „Sprint Retrospective“ vorzeitig beendet und der neue Sprint ganz normal mit einem „Sprint Planning“ begonnen.

[1] *Jeff Sutherland und Ken Schwaber: The Scrum Guide. URL: http://www.scrumguides.org*
[2] *Agile Atlas. Scrum Alliance. URL: http://agileatlas.org/atlas/scrum*

Die Rollen in Scrum

Product Owner

Der Product Owner vertritt die Auftraggeberseite in fachlicher Hinsicht und somit alle Stakeholder im Projekt. Er pflegt das Product Backlog.

Der Product Owner ist dafür verantwortlich, dass im Product Backlog eindeutig priorisierte User Stories existieren. Die Priorisierung ist abhängig von ihrem geschäftlichen Wert und den Story Points, die vom Entwicklungsteam bestimmt werden. Die Erstellung dieser User Stories kann durch den Product Owner eigenständig vorgenommen werden oder er kann auf die Mithilfe vom Entwicklungsteam zurückgreifen, bzw. die Erstellung komplett in die Hände des Teams legen. Wichtig ist jedoch, dass der Product Owner die Anforderungen der anderen Stakeholder konsolidiert und permanent für Fragen zu der geforderten Funktionalität zur Verfügung steht. Die Vermittlung der Produktvision ist eine weitere wichtige Aufgabe des Product Owners. Nur mit einer konkreten und realistischen Vision entwickelt das Team das gewünschte Ergebnis, denn es trifft unbewusst bessere Entscheidungen. Im Kleinen definiert der Product Owner für jeden Sprint das gewünschte Ziel. Des Weiteren bestimmt er den Umfang des Produkts und entscheidet über die Release-Daten. Er ist verantwortlich für das finanzielle Ergebnis des Projektes (ROI).

In diesem Text wird „User Story“ synonym zu „Product Backlog Item“ verwendet. Gemeint sind damit die einzelnen Anforderungen, die entsprechend der Akzeptanz-Kriterien der Stakeholder erfüllt werden müssen, um den Entwicklungsauftrag erfolgreich umzusetzen.

Die Hauptaufgabe des Product Owners besteht darin, die Ergebnisse des Entwicklungsteams, also die umgesetzten User Stories, entweder zu akzeptieren oder zu verwerfen. Bei einer Verwerfung wandern die User Stories wieder zurück ins Product Backlog.

Entwicklungsteam

Das Entwicklungsteam ist eine gemischte, interdisziplinäre Gruppe aus z.B. Entwicklern, Testern und Designern.

Das Entwicklungsteam umfasst alle Entwickler, die an der Umsetzung des Produktes beteiligt sind, inklusive Tester und Anforderungsanalyst. Entwickelt der Scrum Master selbst noch mit, gehört er ebenfalls zum Entwicklungsteam. Die typische Größe eines Entwicklungsteams beträgt zwischen fünf und neun Personen. Zwei Personen mehr oder weniger sind in Ausnahmefällen auch noch akzeptabel. Alle Teammitglieder sollten in Vollzeit mitarbeiten. Besonders wichtig ist, dass alle notwendigen Kenntnisse für die Umsetzung des Produktes im Team verfügbar sind. Nur dann besteht Aussicht auf eine schnelle und qualitativ hochwertige Umsetzung.

Zu Beginn eines Sprints, siehe Sprint Planning, schätzt das Entwicklungsteam, wie viele der am höchsten priorisierten User Stories aus dem Product Backlog es im Sprint umsetzen kann. Anschließend verpflichtet es sich gemeinsam als Team mit einem Commitment auf das jeweilige Sprint-Ziel.
Das Entwicklungsteam organisiert sich und seine Arbeit selbst. Es gibt keinen Projektleiter, der dem Team Vorgaben macht bzw. der die Verantwortung trägt. Dies übernimmt das Entwicklungsteam als Kollektiv, da es für die Umsetzung des Commitments selbst verantwortlich ist.

Scrum Master

Der Scrum Master ist für den Scrum-Prozess verantwortlich.

Zu den ersten Aufgaben des Scrum Masters zählen die Einführung und später die Sicherstellung der Einhaltung der Scrum-Werte und dessen Praktiken. Die „Definition of Ready“ (DoR) und die „Definition of Done“ (DoD) sind die zwei wichtigsten Richtlinien. Beide Definitionen müssen klar formuliert und am besten ständig in visueller Form verfügbar sein. Entwicklungsteam und Product Owner müssen das gleiche Verständnis dafür haben, wenn eine User Story fertig für einen Sprint ist (DoR) bzw. wenn eine User Story wirklich fertig umgesetzt wurde (DoD). Häufig kommt es früher oder später zu großen Problemen, wenn diese Definitionen nicht ganz klar für jeden Beteiligten beschrieben sind. Ein einheitliches Verständnis ist daher ein Schlüssel- und Erfolgsfaktor.

Eine weitere ständige Aufgabe des Scrum Masters ist es, Hindernisse zu beseitigen und sich schützend vor das Entwicklungsteam zu stellen. Externe Störungen, die im allgemeinen über den Product Owner an das Team herangetragen werden, müssen vom Scrum Master aufgefangen werden. Hierzu gehört zum Beispiel das nachträgliche Anpassen von User Stories während eines Sprints (wenn sich zeigt, dass inhaltliche Fragen noch offen waren) oder die schnelle Beseitigung eines Fehlers vom produktiven Release. Der Scrum Master stellt sicher, dass das Entwicklungsteam funktional und produktiv arbeiten kann. Er hilft nicht nur dem Team, sondern auch dem Product Owner seine Arbeiten zu erledigen. Er unterstützt die enge Zusammenarbeit zwischen allen Projektbeteiligten.

Die Ereignisse

Innerhalb jedes Sprints gibt es fünf Ereignisse bzw. Aktivitäten, die nacheinander ablaufen.

Anstelle von fünf Ereignissen findet oft auch der Begriff „fünf Aktivitäten“ Anwendung, um klar zu stellen, dass es sich hier um Besprechungen handelt, bei denen gearbeitet wird. Alle Ereignisse haben strikte Zeitfenster (Time-Boxen).

Sprint Planning

Das Sprint Planning unterteilt sich in die Auswahl der in diesem Sprint zu bearbeitenden User Stories und die Festlegung der sich daraus ergebenden Tasks.

Das Sprint Planning leitet jeden Sprint ein und sollte nicht länger als acht Stunden dauern. Es gliedert sich in zwei Teile.

Im ersten Teil stellt der Product Owner die Product Backlog Items aus dem Product Backlog in Form von User Stories entsprechend ihrer Priorität der Reihe nach vor. Erfüllt die User Story die Definition of Ready (DoR), wird sie mittels der Schätzmethodik „Planning Poker“ vom Entwicklungsteam geschätzt.

Bevor die eigentliche Schätzung beginnen kann, muss eine Referenz User Story vom Entwicklungsteam bestimmt und vom Product Owner vorgestellt werden. Am besten eignt sich hierzu eine kleinere User Storiy. Anschließend wird sie vom Entwicklungsteam bewertet. Die Bewertung erfolgt auf Basis von Story Points, die sich an der Fibonacci-Folge orientieren. Folgende Story Points können beispielsweise zur Verfügung stehen: 0, 1, 2, 3, 5, 8, 13, 21, 50, 100.
Die Story Points einer User Story definieren ihre Größe, das heißt, ihre relative Komplexität. Durch Story Points lassen sich User Stories ins Verhältnis setzen und können so untereinander verglichen werden. Sie sind einheitenlos.
Nachdem die Referenz User Story bestimmt und bewertet wurde, erfolgt die Abschätzung der eigentlichen User Stories. Dabei legt im ersten Schritt jedes Teammitglied eine verdeckte Karte vor sich ab. Mit dieser Karte bestimmt jedes Teammitglied für sich die Story Points der zu schätzenden User Story. Im Anschluss diskutieren die beiden Teammitglieder miteinander, die die kleinste bzw. größte Zahl abgelegt haben. Findet keine Einigung statt, wird die gleiche Story so lange von allen Teammitgliedern geschätzt, bis eine Einigung gefunden wird. Bei der Diskussion für die Einigungsfindung können sich alle Teammitglieder beteiligen. Dadurch wird schnell ein einheitliches Verständnis von technischen und fachlichen Herausforderungen geschaffen. Der Scrum Master übernimmt beim „Planning Poker“ die Rolle des Dealers im Poker-Kartenspiel. Oft ist es ratsam, Stories mit vielen Points (50 und 100) in mehrere kleinere Stories aufzuteilen. Dadurch wird die Gesamtsumme in der Regel kleiner.
Das Entwicklungsteam bewertet so viele User Stories aus dem Product Backlog, wie es meint, im nächsten Sprint liefern zu können. Alle bewerteten Stories wandern dann aus dem Product Backlog in das Sprint Backlog. Manchmal ist es sinnvoll, dass das Entwicklungsteam einige weitere Stories aus dem Product Backlog bewertet. Diese Stories kann der Product Owner anschließend nutzen und sie neu priorisieren lassen, falls es für ihn wichtige User Stories nicht ins Sprint Backlog geschafft haben.

Als Referenz User Story wird eine Anforderung gewählt, bei der alle im Team eine genaue Vorstellung von ihrer Umsetzung und ihrem Ergebnis haben. Eine solche „kleine“ User Story kann etwa sein, dass ein User seinen Login-Namen auf einer Website angezeigt haben möchte (1 Story Point) oder dass ein Administrator die Möglichkeit haben möchte, User in einem Forum zu sperren (2 Story Points).

Nachdem das Sprint Backlog mit User Stories gefüllt ist, erfolgt der zweite Teil des Sprint Plannings. Das Entwicklungsteam erstellt für jede Story die erforderlichen Tasks, um sie in den Status „done“ zu überführen. Dieser Teil wird als Task Planning bezeichnet. Neue und ungeübte Sprintteams schätzen zusätzlich die Tasks mit Aufwandsstunden ab. Dadurch können sie sicherstellen, dass sie sich mit der Menge der User Stories nicht übernehmen. Falls das Entwicklungsteam beim Task Planning feststellt, dass sie nicht alle User Stories des Sprints schaffen werden, muss es dies umgehend dem Product Owner mitteilen.

Zusammen mit dem Product Owner wird dann nach einer Lösung gesucht. Diese sieht meist so aus, dass entweder User Stories komplett aus dem Sprint Backlog wieder zurück ins Product Backlog wandern oder aber auch durch andere Stories aus dem Product Backlog ersetzt werden. Auch das Verändern von Anforderungen ist eine Lösung.

Abschließend erfolgt das Commitment des Entwicklungsteams darüber, dass es am Ende des Sprints alle User Stories aus dem Sprint Backlog in den Status „done“ überführt haben wird.

Daily Scrum

Als Daily Scrum wird ein täglich stattfindender Kurztermin bezeichnet, der zur gegenseitigen Information über den Status Quo dient.

Das Daily Scrum wird jeden Tag zur gleichen Zeit am selben Ort durchgeführt und dauert nicht länger als 15 Minuten. Alle Teammitglieder müssen teilnehmen. Die Anwesenheit des Product Owners ist optional. Wenn er jedoch Tasks vom aktuellen Sprint bearbeitet, ist seine Anwesenheit ebenfalls Pflicht.
Beim Daily Scrum beantwortet jeder Teilnehmer der Reihe nach drei Fragen:

1. Was habe ich seit dem letzten Daily Scrum erreicht?
2. Was werde ich bis zum nächsten Daily Scrum erreichen?
3. Welche Probleme stören mich bei meiner Arbeit?

Das Daily Scrum dient nicht dazu, Probleme sofort zu lösen, sondern lediglich dem Informationsaustausch innerhalb des Entwicklungsteams. Längere Diskussionen, die die 15 Minuten überschreiten würden, sind strikt zu unterbinden. Diese werden meist gesondert in Anschlussbesprechungen und nur mit den betroffenen Personen durchgeführt.

Product Backlog Refinement

Das Backlog Refinement ist eine kontinuierliche Arbeit und dient dazu, die User Stories vom Product Backlog möglichst gut für folgende Sprints vorzubereiten.

Das Product Backlog Refinement dient dazu, die User Stories des Product Backlogs zu verfeinern. Der Product Owner stellt analog zum Sprint Planning User Stories aus dem Product Backlog vor, die dann mittels Planning Poker vom Entwicklungsteam bewertet werden. Der Product Owner kann die Priorität der User Stories während des gesamten Refinements ändern. Auch kann er bestehende User Stories ändern oder löschen sowie neue Stories erstellen.

Die Anwesenheit von Stakeholdern kann dazu führen, dass weitere wertvolle Informationen zum Entwicklungsteam gelangen, die der Product Owner eventuell nicht liefern könnte oder würde. Durch gut organisierte Refinements kann sich die erste Phase des Sprint Plannings stark verkürzen, da auf eine intensive Vorstellung der jeweiligen Stories verzichtet werden kann.

Das Refinement darf nicht mehr als 10 % der Kapazität des Entwicklungsteams beanspruchen. Durch die Ergebnisse des Refinements, kann der Product Owner eine zuverlässige Release Planung vornehmen.

Sprint Review

Im Sprint Review werden dem Product Owner und den Stakeholdern die umgesetzten User Stories vorgestellt.

Am letzten Sprint-Tag wird der Sprint Review durchführt, der maximal eine Stunde je Sprint-Woche dauert. Hier werden die umgesetzten User Stories aus dem Sprint nicht nur dem Product Owner, sondern auch allen Stakeholdern vorgeführt. Der Product Owner entscheidet, ob eine User Story wirklich „done“ ist. Wenn sie nicht „done“ ist, wandert sie zurück in das Product Backlog.

Je mehr Kunden beim Review beteiligt sind, desto höher wird später die Akzeptanz des Product Increments sein, denn diese können den Product Owner in dieser Besprechung entsprechend beeinflussen. Der Product Owner erhält ebenfalls neue Ideen, die er in bestehende oder in zukünftige User Stories aufnimmt. Die Präsentation der User Stories kann sowohl vom Scrum Master, von einem oder mehreren Mitgliedern des Entwicklungsteams als auch vom Product Owner direkt vorgetragen werden.

Sprint Retrospective

Die Sprint Retrospective dient als interne Feedbackrunde des Entwicklungsteams.

Auch die Retrospective findet am letzten Sprint-Tag statt. Für die Länge kann mit ca. 45 Minuten je Sprint-Woche gerechnet werden. Im Gegensatz zum Review findet die Retrospective nicht mit dem Produkt Owner statt. Ein Grund dafür ist, dass es dem Entwicklungsteam so ermöglicht werden soll, offen, ehrlich und ohne Furcht über aktuelle und vergangene Probleme zu reden und diese klar zu analysieren. Dabei sollen nicht die Symptome besprochen, sondern die manchmal verborgenen und versteckten Ursachen gefunden werden.

Dies geschieht häufig dadurch, dass zuerst die guten und anschließend die weniger guten Aspekte des letzten Sprints diskutiert werden. Dabei sollte jeder Entwickler seine Meinung kundtun. Vereinfacht wird dies, wenn während der gesamten Retrospective eine angenehme Atmosphäre herrscht. Die Entwicklungsmitglieder sollten sich wohlfühlen und wissen, dass keine Kritik, die offen angesprochen wird, persönlich aufgefasst werden darf. Sollte die Atmosphäre anders sein, liegt es am Scrum Master, vermittelnd einzugreifen.

Der Scrum Master dokumentiert nicht nur die positiven und die weniger positiven Punkte, sondern auch die Verbesserungsmaßnahmen, die ergriffen werden sollen, um im nächsten Sprint besser zu werden. Gibt es Punkte, die nicht sofort durch Maßnahmen beseitigt werden können, kommen diese auf die Liste der Hindernisse.

Die Umsetzung der Verbesserungsmaßnahmen, die bei der letzten Retrospective für den aktuellen Sprint verfasst wurden, können zu Beginn der Retrospective gemeinschaftlich diskutiert werden. Häufig ergeben sich dadurch wieder Maßnahmen, die als neue User Stories ins Product Backlog wandern können.

Die Artefakte

Product Backlog

Die Artefakte können als Container für User Stories gesehen werden, in denen sie nach Status („noch umzusetzend“, im aktuellen Sprint „in Bearbeitung“ und „umgesetzt“) abgelegt sind.

Das Product Backlog besteht aus der Menge der noch umzusetzenden User Stories und gehört dem Product Owner. Dieser ist für die eindeutige Priorisierung verantwortlich. Das Product Backlog ist dynamisch und ändert sich fortlaufend während der gesamten Entwicklungsphase. Es erhebt nicht den Anspruch auf Vollständigkeit.

Die User Stories des Backlogs sollten aus fachlichen Anforderungsgesichtspunkten erstellt werden und nicht aus technischer Sicht.

Sprint Backlog

Das Sprint Backlog beinhaltet alle User Stories, die für den aktuellen Sprint aus dem Product Backlog ausgewählt wurden. Zusätzlich zu den User Stories kommen noch die zugehörigen Tasks hinzu.

Product Increment

Das Product Increment ist das Ergebnis aller nach der „Definition of Done“ (DoD) umgesetzten User Stories aus dem aktuellen Sprint und der vergangenen Sprints. Im Bereich der Softwareentwicklung ist es in der Regel die prinzipiell produktiv einsatzfähige Software, auch wenn der Product Owner sie aktuell noch gar nicht ausliefern will.

Die Vorteile

Scrum ermöglicht schnelles, flexibles und effektives Arbeiten.

Bei der agilen Softwareentwicklung mittels Scrum können Anforderungsänderungen schnell umgesetzt werden und stellen keine Herausforderung mehr dar. Zusätzlich sieht der Kunde frühzeitig das zu erwartende Ergebnis.

Der Ansatz bei der agilen Softwareentwicklung ist, sich nur auf einige wenige Regeln zu fokussieren und den bürokratischen Aufwand zu minimieren. Dadurch lassen sich schneller und zuverlässiger Softwarestände produktiv nutzen. Folgende Punkte zeigen weitere Vorteile gegenüber den klassischen Vorgehensmodellen auf:

- Hohe Transparenz über Projektstatus und Projektfortschritt für das gesamte Scrum-Team.
- Änderungen an User Stories aus dem Product Backlog stellen kein Problem dar. Im Gegenteil, es ist sogar der Normalfall, dass diese im Laufe ihrer Lebenszeit Änderungen unterliegen. Befindet sich jedoch eine User Story im aktuellen Sprint Backlog, sind Änderungen an dieser nicht mehr zulässig.
- Das Product Increment, also das Softwareergebnis, kann den Stakeholdern oder auch den Endkunden nach jedem Sprint zur Verfügung gestellt werden. Dadurch verkürzen sich einerseits die Auslieferungsphasen erheblich. Andererseits wird das Risiko etwas zu entwickeln, das keiner haben möchte, auf eine Sprintlänge reduziert, denn nach dem Sprint können alle Verantwortlichen das Ergebnis selbständig begutachten.
- Auch die Planungsphasen sind kurz gehalten, denn jeder Sprint beginnt jeweils mit einer eigenen Miniplanungsphase, dem Sprint Planning. Dadurch kann der Product Owner mit jedem Sprint Planungsänderungen, die ein Release betreffen, neu vornehmen. Er kann gleichwertige User Stories austauschen, ohne einen Change Request stellen zu müssen.
- Durch die permanente Ergebnisrückkopplung steigt die Produktivität, da effektiver gearbeitet werden kann. Auch die Qualität der Software ist sehr hoch, da das Sprintergebnis nach jedem Sprint ausgerollt werden kann. Dies führt zu einem ständigen Soll-Ist Vergleich und zu einer Erhöhung der Qualität.
- Die Mitglieder des Entwicklungsteams haben direkten Kontakt zum Product Owner und zu Stakeholdern. Dadurch identifizieren sie sich stärker mit dem Product Increment, was wiederum zur Senkung der Mitarbeiterfluktuation führt.

Erfolgsfaktoren

Damit Scrum erfolgreich eingesetzt werden kann, sind folgende Punkte besonders zu beachten:

Einige wenige Bedingungen müssen erfüllt sein, um agile Entwicklung mit Scrum auf Erfolgskurs zu führen.

- Das Entwicklungsteam sollte aus enthusiastischen Teammitgliedern zusammengesetzt sein, die bereit sind, für ihre Ideale und Vorstellungen einzustehen.
- Nicht nur das Management muss agile Entwicklungsmethodiken unterstützen, sondern auch alle anderen Projektbeteiligten.
- Das Entwicklungsteam sollte mit einem Tester ausgestattet sein. Dieser ist angehalten, so früh wie möglich User Stories auf ihre Akzeptanzkriterien zu testen. User Stories erst geballt in der letzten Sprint-Woche zu testen widerspricht der agilen Vorgehensweise „liefere so früh wie möglich“.
- Das frühe Liefern gilt nicht nur in Richtung des Product Owners, sondern auch in Richtung des Endkunden. Häufiges Ausliefern senkt das Risiko einer schlechten Akzeptanz am Markt, bzw. es kann schneller ausgebessert werden, falls sich der Endkunde anders verhält als erwartet.
- Zu Beginn ist der Product Owner häufig überfordert. Hier liegt es am Scrum Master, den Product Owner zu unterstützen. So können Scrum Master und Product Owner am Anfang Aufgaben gemeinsam erledigen.
- Commitments gelten nicht nur vom Entwicklungsteam zum Product Owner, sondern auch anders herum. Dies gilt insbesondere für Anforderungsänderungen an User Stories im Sprint Backlog.
- Die Rollen Scrum Master und Product Owner sollten immer von zwei verschiedenen Personen ausgeführt werden. Die Interessen widersprechen sich teilweise, daher würde eine Person ständig in Gewissenskonflikte geraten.
- Ein Taskboard oder auch Scrumboard ist unabdingbar. Es ist egal, ob die Tasks analog oder digital erfasst werden. Wichtig ist nur die Transparenz für alle Beteiligten.
- Der Quellcode gehört dem gesamten Entwicklungsteam (Collective Code Ownership) und nicht dem Ersteller des Codes. Jedes Teammitglied darf den Code eines anderen Teammitglieds ändern, wenn es dem Sprint-Ziel dient. Entwickler, die einen starken eigenen Besitzanspruch auf ihren Code legen und schnell verärgert sind, wenn andere Teammitglieder ihren Code ändern, sind in einem agilen Entwicklungsteam wenig hilfreich.

Die Nachteile

Nachteile der agilen Entwicklung können durch entsprechende Gegenmaßnahmen entschärft werden.

Auch Scrum bringt nicht nur Vorteile, sondern ebenso Nachteile mit sich, die ohne entsprechende Maßnahmen sehr große Auswirkungen auf das gesamte Projekt haben können:

Die Besonderheiten von Scrum bergen durchaus auch Gefahren für den Projekterfolg. Entsprechende Gegenmaßnahmen müssen daher frühzeitig ergriffen werden.

- Das Entwicklungsteam organisiert sich selbst. Es gibt keinen Projektleiter, der Arbeitspakete verteilt und Entscheidungen trifft, bzw. diese dem Entwicklungsteam abnimmt. Diese Arbeitsweise kann zu Beginn sehr ungewohnt sein und dazu führen, dass das Entwicklungsteam ziellos vor sich hin entwickelt und nicht mehr effektiv arbeitet. Häufig werden auch unrentable Diskussionen geführt. Bei einigen Teammitgliedern kann dies zu Verunsicherung führen.
- Im Entwicklungsteam gibt es keine Hierarchien und es herrscht Gleichberechtigung. Jedes Teammitglied besitzt in allen Bereichen ein gleiches Stimmrecht. Ideen und Vorschläge neuer Mitglieder werden genauso ernst genommen wie die von langjährigen Teammitgliedern. Manche Teammitglieder empfinden dies als Machtverlust und lehnen sich dagegen auf.
- Der stark iterative Ansatz durch Sprints kann dazu verleiten, nur kurzfristige Lösungen zu entwickeln. Die Gefahr, den Überblick zu verlieren, ist gegeben.
- Wird Scrum in einem Festpreis-Projekt eingesetzt, wird schnell deutlich: Festpreise können am Anfang häufig nicht realistisch abgeschätzt werden, da im Laufe des Projektes immer wieder Änderungen vorgenommen werden, die sich auf den Preis auswirken. Oft entstehen dem Kunden daher höhere Kosten als anfänglich geplant.

Gegenmaßnahmen

Die erwähnten Nachteile können durch folgende Gegenmaßnahmen entschärft werden:

- Der Scrum Master ist dafür verantwortlich, dem Entwicklungsteam bei der Selbstorganisation zu helfen und es zu unterstützen. Diskussionen, die unfruchtbar und nicht lohnend sind, sollten vom Scrum Master eingedämmt werden. Ängste von unsicheren Teammitgliedern müssen ausgesprochen und entsprechend behandelt werden. Ein Lösungsansatz ist die enge Zusammenarbeit von Mitgliedern. Meistens entstehen dadurch schnelle und gute Lösungen.
- Dass jedes Teammitglied gleichberechtigt Vorschläge und Ideen einbringen kann, hält das Projekt und das Team innovativ und vermeidet eine „Null-Bock-Haltung“. Den Teammitgliedern, die um den Verlust ihrer Macht besorgt sind, sollte der Scrum Master vermitteln, dass sie aufgrund ihrer Fähigkeiten im Team sind und nicht wegen ihrer erkämpften Position.
- Auch im nicht agilen Umfeld entstehen kurzfristige Lösungen. Häufig hilft es, die Rolle des Softwarearchitekten zu vergeben oder diese auf zwei Teammitglieder zu verteilen. Dabei empfiehlt sich eine Trennung entsprechend der genutzten Technologie. Der Architekt sollte auf die Nachhaltigkeit der Lösung achten und langfristige Ziele verfolgen. Durch eine stark testgetriebene Entwicklung können Nachjustierungen an Design-Entscheidungen sicher korrigiert werden.

- Um zu vermeiden, dass die festgelegten Preise im Laufe des Entwicklungsprojektes in die Höhe schnellen, sollte auf eine anfängliche Schätzung der Festpreise verzichtet und stattdessen die ersten Sprints des Scrum-Teams abgewartet werden. Erst dann ist die „velocity“ zuverlässig messbar und der Preis für eine komplette Backlog-Umsetzung kann sicher errechnet werden. Dieser Preis ist verlässlicher als ein blind festgelegter Betrag, der am Ende durch die zahlreichen Anforderungsänderungen kaum mehr dem ursprünglichen Angebot entspricht.

Richtigstellungen

Kurz noch zwei Richtigstellungen über Scrum, die häufig anders gesehen werden:

Es geht nicht ohne Dokumentation. Und: Für Scrum gilt „ganz oder gar nicht“.

- Auch bei Scrum ist eine Dokumentation unabdingbar. Jede User Story, die in einen Sprint eingeplant werden soll, muss vollständig mit allen Akzeptanzkriterien erfasst sein. Falls sich Anforderungen ändern – dies gilt auch für bereits umgesetzte User Stories – müssen sie entsprechend nachträglich angepasst werden. User Stories und Umsetzung müssen immer synchron geändert werden, damit die Herausforderungen, die bei der Umsetzung gelöst werden sollen, zu der dazu entwickelten User Story passen.
- Agilität ist keine Garantie für ein besseres Produkt. Nur wenn Scrum zu 100 % angewendet wird, stellt sich der Erfolg ein. Die Nutzung von nur einigen Scrum-Elementen, für die schnellen Quick Wins, führt mittel- oder langfristig nicht zum gewünschten Erfolg.

Das Fazit

Agile Softwareentwicklung ist erwiesenermaßen erfolgreicher als Entwicklung per Wasserfallmethodik.

Eine agile Softwareentwicklung auf Basis von Scrum kann die Anforderungen, die heute an den Softwareentwicklungsprozess gestellt werden, sehr gut bedienen. Insbesondere kann kurzfristig auf Änderungen reagiert werden. Auch die Gefahr, dass etwas vom Entwicklungsteam entwickelt wird, das der Kunde gar nicht haben wollte, wird gesenkt, da dem Kunden Entwicklungsergebnisse schneller sichtbar gemacht werden. Dies belegt auch der Standish Chaos Report 2011. In diesem Report wird deutlich, dass Projekte eine deutlich höhere Wahrscheinlichkeit haben erfolgreich abgeschlossen zu werden, wenn sie mit einer agilen Methodik umgesetzt werden.

Bei Anwendung der Wasserfallmethodik waren laut Standish Chaos Report nur 14 % der Projekte erfolgreich, 57 % problematisch und 29 % scheiterten. Im Gegensatz dazu waren bei der agilen Methodik 42 % der Projekte erfolgreich, 49 % problematisch und nur 9 % scheiterten.[3] Der Standish Chaos Report 2015 stützt und bestätigt diese Ergebnisse erneut.

[3]The Standish Group (Hrsg.): Chaos Manifesto - The Laws of CHAOS and the CHAOS 100 Best PM Practices. S. 25.

Ansprechpartner

Martin Rebenstorff
stellv. Projektleiter
Tel: +49 151 18206683
E-Mail: martin.rebenstorff@hud.de

H&D International Group

August-Horch-Straße 1
38518 Gifhorn

Tel: +49 5371 960 0
E-Mail: kommunikation@hud.de
www.hud.de