Agile Softwareentwicklung mit Scrum und User Stories

Ralf Wirdemann
Zert. ScrumMaster und Scrum Practitioner, Coach für agiles PM

Das Vorgehensmodell Scrum erfreut sich durch wenige Regeln und klare Verantwortlichkeiten in der Softwareentwicklung zunehmender Beliebtheit. So erlaubt Scrum ein einfaches und flexibles Vorgehen – wie die Beteiligten den Entwicklungsprozess gestalten und organisieren, bleibt ihnen selbst überlassen. (Siehe auch "Agiles Projektmanagement. Scrum – eine Einführung", projektmagazin, Ausgabe 21/2009.)

Allerdings enthält Scrum keine konkreten Vorgaben, wie die Anforderungen an die zu entwickelnde Software erfasst und spezifiziert werden sollen. Die agile Softwareentwicklung verfolgt jedoch den Ansatz, die Anforderungen auf eine Art und Weise zu beschreiben, die es dem Entwicklungsteam ermöglicht, nach jedem Sprint nutzbare Software für den Kunden zu liefern.

Ein etablierter Weg, dies zu tun, ist die Verwendung von User Stories. Dabei handelt es sich um einfache Formulierungen der Anforderungen aus Sicht des späteren Anwenders. Dieser Artikel beschreibt, wie man die Anforderungen eines Scrum-Projekts mit Hilfe von User Stories beschreiben und verwalten kann.

Praxisbeispiel

Zum besseren Verständnis soll das Prinzip der User Stories innerhalb eines Scrum-Projekts am Beispiel eines webbasierten Job-Portals erläutert werden: Eine Gruppe von Investoren möchte ein Portal für die Stellenvermittlung im High Professional-Umfeld entwickeln, das Firmen und Recruitern Zugriff auf nachgewiesen hoch qualifiziertes Fachpersonal ermöglichen soll. Im Gegenzug bietet das Portal Job-Suchenden die Möglichkeit, Kontakte zu renommierten und interessanten Arbeitgebern aufzubauen.

Ein Auszug aus der zentralen Anforderungsliste des Job-Portals, dem sog. Product Backlog, könnte wie folgt aussehen:

- Datenbankmodell entwerfen
- OR-Mapping-Framework (Mapping zwischen den Objekten des Systems und der unterliegenden relationalen Datenbank) entwickeln

- User Interface für die Job-Suche entwerfen
- Security Framework (Authentifizierung und Autorisierung von Nutzern) implementieren
- Suchmaschine entwickeln und integrieren
- ...

Das Problem: Scrum enthält keine Spezifikation der Anforderungen

Das Product Backlog in Scrum ist eine priorisierte Liste mit sämtlichen Anforderungen des Projekts. Die Einträge im Backlog werden als Backlog Items bezeichnet. Für jeden Eintrag liegt eine Schätzung des Aufwands vor; die Priorität des Eintrags bestimmt seine Position in der Liste. Während Scrum zwar mit dem Product Backlog sehr genau vorgibt, wie und womit Anforderungen verwaltet werden, trifft es keinerlei Aussagen darüber, wie die einzelnen Einträge des Backlogs formell genau auszusehen haben. Von "Die Software muss in Java programmiert werden" bis hin zu nicht-funktionalen Anforderungen, wie "Die Applikation muss skalierbar sein" ist alles erlaubt, was irgendwie getan werden muss.

Beide Beispiele sind notwendige Anforderungen, liefern aber keinen offensichtlichen Mehrwert für die Nutzer der Anwendung. Und darum geht es schließlich bei Scrum – der Entwicklung und regelmäßigen Auslieferung von nutzbarer Software für den Kunden. Entsprechend sollte das Product Backlog nur Einträge enthalten, die einen klar ersichtlichen Wert für die späteren Nutzer der Anwendung liefern. Hier hat der Product Owner die Aufgabe, ein Backlog zu erstellen, das diesen Kriterien genügt.

Ein Beispiel für Anforderungen ohne praktische Funktion für den Nutzer ist das oben genannte Backlog Item "OR-Mapping-Framework entwickeln". Die Entwicklung eines solchen OR-Mappers ist sinnvoll und notwendig, nur bietet der OR-Mapper für sich allein genommen keinerlei Mehrwert für die späteren Anwender des Portals. Entsprechend schwierig ist es für den Product Owner, der ja die Sicht des Kunden repräsentiert, dieses Backlog Item im Verhältnis zu den anderen Anforderungen im Sinne des Geschäftswerts zu priorisieren. Der Begriff Geschäftswert beschreibt in diesem Zusammenhang den Wert, den ein neues Feature für das Geschäftsfeld liefert, für das die Software entwickelt werden soll. Dieses kundenorientierte Anforderungsmanagement zielt folglich auf die regelmäßige Produktion und Lieferung von Features mit dem größtmöglichen Geschäftswert ab.

Die Lösung: User Stories im Kontext von Scrum

User Stories sind ein Werkzeug der agilen Softwareentwicklung, mit dem sich Software-Anforderungen kurz und prägnant aus Sicht des Nutzers beschreiben lassen. Viele Teams in der Softwareentwicklung setzen aktuell auf Scrum, so dass sich ein genauerer Blick auf die Verwendung von User Stories im Kontext von Scrum lohnt.

Ein wichtiges Prinzip von Scrum ist, dass die Auftraggeber regelmäßig in kurzen Abständen die aktuelle Version der Software zu sehen bekommen, um möglichst früh Kurskorrekturen vornehmen zu können. Deshalb muss der Product Owner die wichtigsten Anforderungen so früh wie möglich vom Team umsetzen lassen, um dem Kunden funktionsfähige Software präsentieren zu können. Damit diese relevanten Anforderungen auch hoch priorisiert werden, eignen sich User Stories, da sie für den Product Owner sehr viel greifbarer sind und sich gegeneinander abwiegen lassen.

Im Kontext des Job-Portals könnte eine User Story z.B. das Einstellen oder die Suche nach Stellenangeboten sein. An diesem Beispiel wird außerdem offensichtlich, dass das Einstellen von Angeboten vor einer möglichen Suche stattfinden muss und folglich höher priorisiert werden sollte.

Im Gegensatz zu den vorher exemplarisch genannten Anforderungen zielt eine User Story immer auf einen konkreten Mehrwert für den Kunden ab: Statt der Entwicklung eines OR-Mappers beschreibt eine User Story immer eine konkrete und für den Nutzer sinnvolle Funktion. Technische Details für die Programmierer werden bei dieser Sichtweise nicht ignoriert, sondern zu einer notwendigen Voraussetzung für die Entwicklung einer User Story. Der OR-Mapper liefert für sich genommen keinen Mehrwert, ist aber erforderlich für die Entwicklung eines sinnvollen Features für den Nutzer. Während im Backlog nur die User Story auftaucht, wird das Team den OR-Mapper im Rahmen der Entwicklung aufgreifen und als technisches Detail der Story implementieren.

Was ist eine User Story?

Eine User Story beschreibt eine Anforderung an das zu entwickelnde Softwaresystem. Jede User Story besteht aus drei Teilen: Einer Story Card, ihren Akzeptanzkriterien und die für ihre konkrete Umsetzung erforderliche Konversation zwischen Product Owner und Entwicklungsteam.

Story Card

Der schriftliche Teil einer User Story, die sogenannte Story Card, besteht häufig nur aus einem einzigen Satz, der den Kern der Anforderung auf den Punkt bringt. Beispiele für User Stories sind:

- "Als Nutzer kann ich nach Stellenangeboten suchen" oder
- "Als Arbeitgeber kann ich Stellenanzeigen aufgeben".

Beide Beispiele bringen die jeweilige Anforderung auf den Punkt und machen in einem Satz klar, um wen es geht und was derjenige mit dem System machen kann. Der Product Owner kann dadurch den Geschäftswert der jeweiligen Anforderung bestimmen und sie entsprechend priorisieren.

Story Cards lassen sich am einfachsten mit Hilfe von Karteikarten oder großen Post-Its verwalten. Diese lassen sich schnell schreiben und für alle sichtbar an die Wand hängen. Sie sind weder zu groß, noch zu klein und bieten ausreichend Platz für den Titel der Story, vermeiden dabei aber das Schreiben von zu langen Texten. Physische Story Cards funktionieren allerdings

nur für zentral arbeitende Teams; verteilt sich das Team über mehrere Orte, bietet sich der Einsatz elektronischer Tools, wie Scrumy, Mingle oder ScrumNinja (siehe Referenzen) an.

Akzeptanztests

Die Story Card ist nur einer von drei Teilen einer User Story. Über die Card hinaus besteht eine User Story aus ihren Akzeptanztests. Dabei handelt es sich um beispielhafte Beschreibungen der Funktionen einer User Story. Mit den Akzeptanztests legt der Product Owner fest, welche konkreten Funktionen die Story liefern muss. Einige Akzeptanztest-Beispiele für die Suche nach Stellenangeboten im Kontext des Jobportals:

- Eine Stichwortsuche nach "Java" liefert alle Angebote, die den Begriff "Java" entweder im Titel oder in der Beschreibung des Angebots enthalten.
- Eine Suche nach "Ort" liefert alle Angebote, die sich entweder direkt auf den gesuchten Ort oder auf einen Umkreis von 50 Kilometer außerhalb dieses Orts beziehen.

Akzeptanztests werden vom Product Owner formuliert und sind gleichermaßen Test und exemplarische Spezifikation. Für den Product Owner sind sie Abnahmekriterien, die er nach der Fertigstellung der Story überprüfen muss. Für das Team sind Akzeptanztests während des Sprints eine Richtschnur, was genau zu entwickeln ist und wann die Story aus Geschäftssicht fertig ist.

Ein geeigneter Platz für das Festhalten der Akzeptanztests ist die Rückseite der jeweiligen Story Card. Elektronische Tools bieten i.d.R. gesonderte Eingabebereiche, um die Tests zu notieren.

Konversation zwischen Product Owner und Entwicklungsteam

Der dritte und wichtigste Teil einer User Story ist die Konversation zwischen dem Product Owner und dem Team. Wenn der schriftliche Teil einer User Story nur aus einem einzigen Satz und den zugehörigen Akzeptanztests besteht, wo bleiben dann die Details, d.h. die konkreten Ausprägungen der jeweiligen Story? Ganz einfach: Sie werden vom Product Owner dem Team erzählt und von ihm während des Sprints maßgeblich mitbestimmt. Anstatt die Details der Anforderungen im Vorfeld der Entwicklung sehr genau aufzuschreiben, verlagern User Stories die Anforderungsbeschreibung von der schriftlichen auf die verbale Kommunikation. Dadurch rückt der Product Owner während des Sprints nah an das Team heran, sieht die entwickelte Software sehr früh und erhält so viel bessere Steuerungsmöglichkeiten, die Software im Sinne des Kunden zu entwickeln.

Der typische Ablauf für die Planung und Umsetzung einer Story im Rahmen eines Sprints gestaltet sich wie folgt: Der Product Owner erklärt dem Team im Sprint Planning seine Vorstellungen von der Story "Als Job-Suchender will ich nach Stellenangeboten suchen" und das Team entscheidet die Story in den anstehenden Sprint zu nehmen. Anschließend startet das Team in den Sprint und beginnt mit der konkreten Umsetzung der Story; es arbeitet Aufgabe für Aufgabe (sog. Tasks) ab und entwickelt möglichst schnell eine erste vorzeigbare Version der Story.

Zusammen mit dem Product Owner geht das Team diese erste Version direkt vor dem Rechner durch und diskutiert die konkreten Details ihrer Weiterentwicklung. Die Story wird Schritt für Schritt und unter ständiger und sehr enger Einbeziehung des Product Owners weiter entwickelt. Durch diese Art der engen Zusammenarbeit wird von vornherein sichergestellt, dass das Team etwas entwickelt, was der Product Owner auch haben will.

Der Anforderungsworkshop

Um User Stories zu sammeln, sollte zu Beginn des Projekts ein Anforderungsworkshop durchgeführt werden. Der Product Owner initiiert den Workshop und lädt neben dem Entwicklungsteam alle Stakeholder, wie z.B. Marketing, Vertrieb, Investoren oder auch die Geschäftsführung ein. Jeder, der in irgendeiner Form etwas zu den Anforderungen der anstehenden Produktentwicklung sagen kann, sollte an diesem Workshop teilnehmen.

Der Workshop beginnt damit, dass der Product Owner den Teilnehmern seine zuvor in Zusammenarbeit mit dem Kunden ausgearbeitete Produktvision präsentiert. Die Produktvision ist ein kurzes und knappes Statement, das beschreibt, was entwickelt werden soll. Die Vision bringt den Kern des Produkts auf den Punkt und soll möglichst alle Stakeholder überzeugen und ins Boot holen. Die Produktvision des Job-Portals könnte z.B. folgender Satz sein: "Wir wollen das einzige Job-Portal für die Stellenvermittlung im High Professional-Umfeld in Europa werden."

Wurde die Vision vermittelt und von den Teilnehmern verstanden, ist der nächste Schritt im Workshop die Modellierung der Nutzerrollen. Eine Nutzerrolle ist eine Gruppe von Nutzern mit identischen Zielen in Bezug auf die Nutzung des Systems. Die zentrale Frage bei der Modellierung der Nutzerrolle ist also, welche Bedürfnisse der Anwender der Software hat. Denken Sie beispielsweise an eine der diversen Job-Seiten im Internet, dann drängen sich einige offensichtliche Nutzerrollen unmittelbar auf: Job-Suchende, als Gruppe von Nutzern die über die Plattform neue Stellen finden wollen, oder Job-Anbieter, als Gruppe von Nutzern, die Stellenangebote ausschreiben wollen, um neue Mitarbeiter zu finden.

Mit Nutzerrollen die Anforderungen festlegen

Nutzerrollen, deren Beschreibung und Ziele sind ein wichtiger Ausgangspunkt für das Schreiben der ersten User Stories. Das funktioniert gut, indem sich das Anforderungsteam Rolle für Rolle vornimmt und anhand der jeweiligen Ziele die User Stories überlegt, die für die Zielerreichung notwendig sind. Das Anforderungsteam besteht idealerweise aus Vertretern sämtlicher am Projekt beteiligen Stakeholder, wie dem eigentlichen Scrum-Team, der Marketingabteilung, dem Vertrieb oder auch der Geschäftsführung. Auf diese Art wird sichergestellt, dass alle relevanten Parteien von Anfang an beteiligt sind und den Product Owner beim Schreiben der User Stories unterstützen.

Nehmen wir als Beispiel die Nutzerrolle "Job-Suchender". Das primäre Ziel dieser Rolle ist es, einen neuen Job zu finden. Dazu muss der Job-Suchende den Stellenmarkt nach bestimmten Kriterien durchsuchen können. Enthält die Ergebnisliste interessante Angebote, dann will er die

Angebote im Detail betrachten. Interessiert ihn ein Angebot besonders, dann will er sich schließlich darauf bewerben. Ein einfacher Blick auf das Ziel "Job finden" führt zu drei offensichtlichen User Stories:

1. Als Arbeitssuchender will ich die Angebote nach bestimmten Kriterien durchsuchen.
2. Als Arbeitssuchender will ich die Details der gefundenen Angebote sehen.
3. Als Arbeitssuchender will ich mich auf ein Angebot bewerben.

Beim Betrachten dieser User Stories fällt auf, dass die Stories einem Muster folgen: Als <Nutzerrolle> will ich <das Ziel> [, so dass <Grund für das Ziel>].

Die Verwendung dieses Musters hat sich in der Praxis bewährt und ermöglicht es dem Product Owner mit sehr wenigen Worten sehr viel auszudrücken: Für wen ist die Anforderungen gedacht, welches Ziel verfolgt derjenige mit der Anforderung und was ist der Grund für dieses Ziel. Außerdem zwingt das Muster den Product Owner, den Kern der Anforderung und damit deren Wert für den Kunden auf den Punkt zu bringen.

Der letzte Teil des Musters, der "Grund für das Ziel", ist optional und sollte immer dann verwendet werden, wenn das Ziel alleine nicht ausreicht, um den Geschäftswert der Anwendung zu formulieren. Ein Beispiel ist die Story: "Als Job-Anbieter will ich mich einloggen". Die Funktion "Einloggen" stellt keinen offensichtlichen Geschäftswert dar. Erweitert man hingegen die Story um den Grund "so dass kein anderer Anbieter unter meinem Namen Job-Angebote erstellen kann", werden der Wert und damit der Grund fürs Einloggen deutlich.

Ergebnis des Workshops

Das Ergebnis des ersten Anforderungsworkshops ist ein initiales Product Backlog, gefüllt mit den wesentlichen User Stories des Projekts. Die initialen User Stories sind häufig noch zu groß, als dass sie sich für die direkte Umsetzung in einem der ersten Sprints des Projekts eignen würden. Deshalb müssen sie auf die richtige Größe geschnitten werden. Hierbei kommt es darauf an, dass die resultierenden Teil-Stories weiterhin ihren Geschäftswert behalten. Ein Beispiel für eine zu große Story ist die Suche nach Stellenangeboten. Diese kann die Stichwortsuche, die Suche nach Ort, Bezahlung oder Position einbeziehen. Jede dieser Teilfunktionen sind wieder eigenständige User Stories die weiteren Wert für den Kunden schaffen und sich eigenständig umsetzen lassen.

Mit zunehmender Projekterfahrung wird relativ schnell offensichtlich, wann eine Story zu groß ist und wie sie sich am sinnvollsten zerschneiden lässt. Grundsätzlich gilt es, Stories so klein wie möglich zu schneiden, sofern sie weiterhin eine nützliche Funktion für den Anwender liefern. Die natürliche Obergrenze für die Größe einer Story ist dabei immer ein Sprint; schätzt das Team die Entwicklung einer Story auf länger als einen Sprint, ist klar, dass die Story zerschnitten werden muss.

Weitere Anhaltspunkte für zu große User Stories liefern die regelmäßig stattfindenden Estimation-Meetings. In diesen Meetings schätzt das gesamte Team die bis dahin geschriebenen User Stories.

Kommt es dabei zu sehr großen Schätzungen, oder kann der Product Owner die Story dem Team nicht ausreichend weit erklären, so dass das Team nicht in der Lage ist, die Story zu schätzen, ist es offensichtlich, die Story zu verkleinern.

Der erste Schritt vor dem Schätzen und dem daraus ggf. folgendem weiteren Zerschneiden der Stories ist ihre Priorisierung.

User Stories priorisieren und schätzen

Wie eingangs erwähnt, ist es ein wesentliches Scrum-Prinzip, die wichtigsten Dinge zuerst zu erledigen, um dem Kunden früh funktionsfähige Software zu präsentieren. Folglich muss sich der Product Owner für sein Backlog überlegen, welches die wichtigsten Stories sind und sie entsprechend priorisieren. Die wichtigsten Stories wandern im Backlog nach oben und werden damit zu Kandidaten für den nächsten Sprint.

Eine einfache und schnelle Variante der Priorisierung ist die sogenannte "MuSCoW-Priorisierung", die Stories hinsichtlich der folgenden vier Kriterien bewertet:

Must Have: Diese User Stories sind zwingend erforderlich. Ohne sie würde das System nicht funktionieren.

Should Have: Diese User Stories sind sehr wichtig, das System funktioniert aber auch ohne sie, weil z.B. eine provisorische Lösung (Workaround) existiert.

Could Have: User Stories dieser Kategorie haben eine geringe Bedeutung und werden nur umgesetzt, wenn neben Must Have- und Should Have-Stories noch Kapazitäten zur Verfügung stehen.

Won't have this time: Diese User Stories sind zurzeit nur vorgemerkt, werden aber aktuell nicht umgesetzt.

Nach dem ersten Anforderungsworkshop enthält das Backlog meistens noch viele zu große Stories. Mit dem Prinzip MuSCoW kann der Product Owner eine schnelle und einfache Priorisierung des initialen Product Backlog durchführen und die Stories hinsichtlich ihrer Notwendigkeit für ein Funktionieren des Systems bewerten. Alle Must Have- und Should Have-Stories werden weiterverfolgt und auf konkretere Stories "heruntergebrochen". Die resultierenden Teil-Stories werden ihrerseits nach MuSCoW priorisiert. Auf diese Weise ist es möglich, relativ schnell zu einem sinnvoll priorisierten Product Backlog zu kommen, das vom Team geschätzt und als Grundlage für die Planung des ersten Sprints genutzt werden kann.

Stories schätzen

Nach der Priorisierung steht das Schätzen der Stories auf dem Programm. Zusammen mit dem Team geht der Product Owner das Backlog von oben nach unten durch und lässt das Team gemeinsam den Aufwand der jeweiligen User Story schätzen. Das relative Schätzen findet im

Rahmen agiler Vorgehensweisen und damit auch im Rahmen von Scrum zunehmende Verbreitung. Bei dieser Art des Schätzens werden Anforderungen nicht in Personentagen, sondern ausschließlich in ihrer Größe geschätzt.

Der entscheidende Vorteil des relativen Schätzens ist die Trennung von Dauer und Größe. Während man beim traditionellen Schätzen versucht, in Personentagen vorherzusagen, welches Feature wie viel Entwicklungszeit benötigen wird, trifft man beim größenbasierten Schätzen ausschließlich Aussagen über die relative Größe von Features zueinander. So lässt sich sehr viel einfacher bestimmen, dass Feature A kleiner als Feature B ist, als vorherzusagen, dass Feature A genau einen Personentag beansprucht, während Feature B zwei Personentage dauern wird. Das Schätzen in relativen Größen erhöht die Wahrscheinlichkeit, dass die Schätzungen zutreffen, erheblich.

Der offensichtliche Nachteil relativer Schätzungen ist, dass Aussagen über die Entwicklungsdauern der jeweiligen Features fehlen. Schließlich wollen Projektleiter und Geschäftsführung nicht wissen, was wie groß im Verhältnis zueinander ist, sondern wann welches Feature geliefert wird. Ein Punkt, auf den ich später im Artikel zurückkommen werde.

Für die Schätzungen verwendet man "Story Points", die die relativen Größenverhältnisse der Stories zueinander ausdrücken. Eine Story von zwei Punkten ist doppelt so groß, wie eine 1-Punkt-Story. Eine 8-Punkte-Story ist viermal so groß, wie eine 2-Punkte-Story, und so weiter. Größe ist dabei zunächst ein abstrakter Begriff, der nichts darüber aussagt, wie lange die Entwicklung der jeweiligen Story dauern wird. Im Abschnitt "Sprints messen und planen" (s. unten) erkläre ich, wie Story Points für das Messen der Entwicklungsgeschwindigkeit des Teams und so für die Release-Planung genutzt werden können.

User Stories im Sprint

Hat der Product Owner die Stories priorisiert und liegen ausreichend geschätzte Stories für mindestens einen Sprint vor, dann kann das Team in seinen ersten Sprint starten. Am Anfang jedes Sprints steht das Sprint Planning Meeting, das in zwei Teile gegliedert ist: das "Sprint Planning 1" und das "Sprint Planning 2". Das Sprint Planning 1 ist ein Analyse-Meeting, in dem der Product Owner dem Team die User Stories der Reihe nach vorstellt. Das Team stellt Fragen, diskutiert die Story und versucht die Anforderungen ausreichend weit zu verstehen, um entscheiden zu können, ob es die Story im anstehenden Sprint vollständig umsetzen kann. Sieht das Team anschließend Raum für eine weitere Story, setzt der Product Owner das Vorstellen fort.

Einzig und allein das Team bestimmt, wie viele Stories es in den Sprint aufnimmt. Der Product Owner gibt ausschließlich die Reihenfolge der Stories, jedoch nicht deren Anzahl vor. Diese Regel ist wichtig im Hinblick auf die Teilung der Verantwortlichkeiten in Scrum: Das Team ist für die Software verantwortlich. Wenn das Team bestimmen darf, wie viel Arbeit es in den anstehenden Sprint nimmt, dann übernimmt das Team damit auch die Verantwortung dafür, die Software am Ende des Sprints zu liefern. Die Übernahme von Verantwortung basiert also auf

Freiwilligkeit, was nicht gegeben wäre, wenn der Product Owner bestimmen würde, wie viele Stories das Team im anstehenden Sprint zu liefern hat.

Das im Anschluss an das Sprint Planning 1 folgende Sprint Planning 2 ist ein Design-Meeting. In diesem Meeting entwirft das Team das Software-Design der Stories und zerbricht jede Story in die konkret auszuführenden Arbeitsaufgaben, die sogenannten Tasks. Die Task-Liste der Angebotssuche könnte z.B. wie folgt aussehen:

- Domain-Modell entwerfen und implementieren
- Controller-Logik programmieren
- Layout entwerfen und Basis-CSS erstellen
- Suchtechnologie evaluieren
- ...

Stories und Tasks werden auf Karteikarten notiert und ans Taskboard gehängt. Das Taskboard ist die To-Do-Liste des Teams, das sämtliche noch ausstehende Arbeit des Sprints visualisiert.

Während des Sprints arbeitet das Team Story für Story ab. Die Stories werden im Sprint in der Reihenfolge ihrer Prioritäten nacheinander umgesetzt. Alle am Ende des Sprints nicht fertigen User Stories nimmt der Product Owner zurück ins Product Backlog. Ist die Story bei der nächsten Priorisierung noch genauso wichtig wie im gerade abgeschlossenen Sprint, schafft sie es unmittelbar zurück in den nächsten Sprint. Hier hat Scrum einen großen Vorteil: Die noch nicht umgesetzten Anforderungen werden ständig darauf überprüft, ob die ursprüngliche Fassung noch gültig ist.

Sprints messen und planen

Auch Scrum-Teams reihen nicht Sprint um Sprint aneinander und gucken mal, wann sie fertig sind. Sowohl der Kunde als auch der Product Owner haben ein berechtigtes Interesse an Planung und einer darauf basierenden Kosten- und Fertigstellungsvorhersage. Planung in Scrum basiert auf der sogenannten "Velocity". Diese bezeichnet die Entwicklungsgeschwindigkeit des Teams gemessen an der tatsächlich entwickelten und am Ende des Sprints gelieferten Funktionalität.

Dazu ein Beispiel: Hat das Team im Sprint Planning Meeting Stories mit einer Summe von 20 Story Points angenommen, schafft während des Sprints aber nur drei statt vier Stories, dann beträgt die tatsächliche Velocity dieses Sprints 20 minus der Story Points der vierten Story. Angenommen die vierte Story wurde vom Team auf fünf Punkte geschätzt, dann liefert das Team 15 Story Points ab, was die tatsächliche Velocity dieses Sprints ist. Basierend auf diesem Wert kann der Product Owner eine erste Releaseplanung vornehmen, indem er das Product Backlog in Häppchen zu je 15 Punkten aufteilt. Die sich daraus ergebende Anzahl an Sprints multipliziert mit der ursprünglich geplanten Sprint-Dauer (z.B. vier Wochen) liefert das geschätzte Fertigstellungsdatum. Voraussetzung dafür ist, dass sämtliche der für das anstehende Release geplanten Stories vom Team geschätzt wurden.

Die Velocity wird von Sprint zu Sprint genauer. Während die tatsächliche Velocity nach dem ersten Sprint noch ein sehr grober und ungenauer Wert ist, pendelt sie sich nach einigen Sprints auf einen mittleren und damit für die Planung geeigneten Wert ein. Für eine sichere Planung empfiehlt es sich, mit einer niedrigeren als der mittleren Velocity zu planen, da auch eine mittlere Velocity durchaus in dem einen oder anderen Sprint unterschritten werden kann. Außerdem ist zu beachten, dass ein Product Backlog für Änderungen offen ist und Änderungen am Backlog folglich zu Änderungen am Release-Plan führen.

Fazit

Dieser Beitrag zeigt die Vorteile der Verwendung von User Stories im Rahmen von Scrum. User Stories fokussieren Team und Product Owner auf den Geschäftswert: Jeder Sprint liefert sichtbaren Mehrwert im Sinne des Kunden. Das Product Backlog enthält mit den User Stories ausschließlich werthaltige Items und kann vom Product Owner entsprechend priorisiert werden. User Stories verlagern den Fokus von der schriftlichen auf die verbale Kommunikation und erzwingen die Vollzeit-Verfügbarkeit des Product Owner. Statt genau aufzuschreiben, was das Team zu liefern hat, bleibt er eng an der Entwicklung dran und kann die Entwicklung des Teams sehr zeitnah und in kurzen Feedback-Schleifen steuern. Auf diese Art wird sichergestellt, dass am Ende jedes Sprints etwas herauskommt, was der Product Owner wirklich haben will, und nicht etwas, was er vor zwei Monaten aufgeschrieben hat.

Referenzen

- http://scrumy.com
- http://www.thoughtworks-studios.com/mingle-agile-project-management
- http://scrumninja.com/scrum_software
- Wirdemann, Ralf: Scrum mit User Stories, Hanser 2009