Spotlight

Arbeiten mit dem Backlog

Eine themenspezifische Zusammenstellung der besten, auf projektmagazin.de erschienenen Artikel, Methoden und Tipps.

www.projektmagazin.de
Mehlbeerenstr. 4, 82024 Taufkirchen
Tel: +49 89 2420798-0
Fax: +49 89 2420798-8

Arbeiten mit dem Backlog

Ein Backlog schafft Überblick, ermöglicht bessere Priorisierung und sorgt für Transparenz zu Anforderungen und Aufgaben. Ob als Produktbacklog für Anforderungen oder als Impediment Backlog – viele Gründe sprechen für die Nutzung eines Backlogs. In diesem E-Book erfahren Sie, welche Arten von Backlogs es gibt und was diese beinhalten. Zudem lernen Sie das entsprechende Vorgehen bei der Pflege und der Schätzung der Aufwände kennen.

Inhalt

Das Backlog – Arten und Inhalte

User Stories erstellen und priorisieren

Aufwände schätzen

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 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 ge-

meinsam 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
- Wirdemann, Ralf: Scrum mit User Stories, Hanser 2009

Requirements Engineering für Projektleiter

Anforderungen prüfen, abstimmen und aktuell halten

Bettina Zastrow
Informatikerin, GF der Zastrow information development GmbH

Elisabeth Wagner
IPMA Level D, Kommunikationsberaterin, Journalistin

Für Projektleiter, die für die Beauftragung von Software verantwortlich sind, ist ein professionelles Auftragsmanagement das A und O. Eindeutig formulierte und vollständige Anforderungen sorgen dafür, dass sich Stakeholder und Projektleitung ebenso verstehen wie Projektleitung und Software-Entwickler. Die Gefahr von Missverständnissen, die unnötig Zeit und Geld kosten, wird dadurch minimiert. In der Artikelserie "Requirements Engineering für Projektleiter" wurde beschrieben, wie Projektleiter Schritt für Schritt zu vollständigen Use Cases aus Sicht der künftigen Nutzer gelangen und aus diesen detaillierte technische Anforderungen (Requirements) für die Software-Entwicklung ableiten. Im vorliegenden Artikel stehen drei weitere Aufgaben eines professionellen Requirements Engineering im Mittelpunkt:

1. Die Qualität durch eine übergreifende Anforderungsanalyse zu sichern,
2. die Anforderungen zu priorisieren und
3. die Änderungen von Anforderungen im Laufe des Projektlebens-zyklus zu verwalten.

Als Projektbeispiel dient wie in den erwähnten vorangegangenen Beiträgen die Programmierung eines Online-Shops für Musikclips ("Jingleshop"), die von Komponisten hochgeladen und von Interessenten gekauft werden können. Als interner Auftraggeber fungiert der Produktmanager, als interner Auftragnehmer der Projektleiter.

Qualität durch übergreifende Anforderungsanalyse sichern

Es wird vorausgesetzt, dass alle Anforderungen von den Stakeholdern abgefragt und tabellarisch (wie in der Artikelserie "Requirements Engineering für Projektleiter" beschrieben) erfasst wurden. Das heißt, die einzelnen Anforderungen haben zu diesem Zeitpunkt bereits eine Qualitätsprüfung durchlaufen: Sie sind bewertet, eindeutig, konsistent, prüfbar, verfolgbar und vollständig.

Bestehende Mängel identifizieren

Damit ist ein wichtiger Schritt getan. Doch erst in der Gesamtsicht wird erkennbar, ob die Liste der Anforderungen vollständig und in sich konsistent ist. Es kann noch sein, dass Anforderungen

- fachübergreifend noch nicht komplett verständlich,
- nicht aktuell,
- in sich oder untereinander inkonsistent bzw. widersprüchlich,
- überflüssig oder
- nicht umsetzbar sind.

Wie kommen solche Mängel trotz Sorgfalt bei der Definition der Anforderungen zustande? Die Ursachen sind vielfältig. Erstens sind Stakeholder aus verschiedenen Bereichen mit unterschiedlichen Perspektiven involviert. Das kann zu divergierenden Wünschen und Anforderungen führen. Zweitens gibt es möglicherweise noch Begrifflichkeiten, die je nach Abteilung anders definiert sind. Drittens kann es sein, dass sich in der Zwischenzeit technische Neuerungen ergeben haben – wie z.B. eine neue Betriebssystemversion – oder es sind Ereignisse eingetreten, die sich auf die Anforderungen auswirken, etwa ein Personalwechsel beim Produktmanagement. Viertens kann es neue oder zunächst nicht bekannte technische, organisatorische oder juristische Beschränkungen geben, die verhindern, dass Anforderungen wie ursprünglich geplant umgesetzt werden können, etwa Vorgaben der Corporate Identity und der Compliance.

Am häufigsten kommt es vor, dass sich mehrere Anforderungen widersprechen: Eine Anforderung sieht eine Bestellung ohne Registrierung vor ("Anmelden als Gast"), eine andere Anforderung verlangt, dass der Nutzer sich vor der Bestellung mit Name und Adresse registrieren muss. Manchmal erkennt auch der Auftraggeber erst im Laufe der Abstimmung mit den Entwicklern, was umsetzbar ist und was nicht – bzw. nur mit einem hohen Aufwand – und ordnet eine Kurskorrektur an.

Beispiel Jingleshop: Mögliche Mängel in Anforderungen

Die folgende Tabelle enthält für jeden der oben genannten möglichen Mängel jeweils ein Beispiel und dazu einen Vorschlag, wie das Problem behoben werden könnte.

Mangel / Kriterium	Anforderung	Erläuterung	Lösungsvorschlag
Verständlichkeit	Der Benutzer mit der Rolle "Produktpflege" muss die Möglichkeit haben, Metadaten einzugeben.	Der Begriff "Metadaten" ist nicht definiert. Entweder sollte er in das Glossar aufgenommen oder wie rechts angegeben formuliert werden.	Der Benutzer mit der Rolle "Produktpflege" muss die Möglichkeit haben, folgende Produktmerkmale einzugeben: Name des Komponisten Name des Songs Länge Genre instrumental/vokal …
Aktualität	Das System muss das Client-Betriebssystem Windows XP unterstützen.	Dieses Betriebssystem wird von Microsoft nicht mehr unterstützt.	Das System muss das Client-Betriebssystem Windows 8.x unterstützen.

Konsistenz	Anforderung 1: Das System muss bei einer Zah-lung, die seit 2 Wochen aussteht, eine Benachrichti-gungs-E-Mail an den Kunden versenden. Anforderung 2: Steht eine Zahlung länger als 10 Kalendertage aus, so hat das System die Bestel-lung zu stornieren.	Es muss eine Entscheidung für eine der beiden wider-sprüchlichen Varianten erfol-gen.	Das System muss bei einer Zah-lung, die seit 2 Wochen aussteht, eine Benachrichtigungs-E-Mail an den Kunden versenden.
Notwendigkeit	Das System soll Benutzern mit der Rolle "Komponist" die Möglichkeit bieten, beliebig viele Produktfotos hochzula-den.	Mehr als 2 Produktfotos sind bei Musikdateien nicht ge-bräuchlich.	Das System soll Benutzern mit der Rolle "Komponist" die Möglichkeit bieten, bis zu 2 Produktfotos hochzuladen.
Umsetzbarkeit	Das System muss eine Ver-fügbarkeit von 100 % haben.	Die Anforderung ist schon al-leine wegen geplanter War-tungsfenster nicht realisier-bar.	Das System muss eine Verfügbar-keit von 99,5 % haben.

Tabelle 1: Beispiele für mögliche Mängel und passende Lösungsvorschläge.

Anforderungen prüfen

Sinnvoll ist es, als Projektleiter das Anforderungsdokument nicht nur selbst zu prüfen, sondern es verschiedenen weiteren Personen zu Qualitätsprüfung zu geben, so dass unterschiedliche Perspektiven berücksichtigt werden. Produktmanager, Business Analyst, Technischer Architekt, Trainer und/oder Entwickler können hier unterstützen. Hilfreich sind in jedem Fall Personen, die über ein stark ausgeprägtes logisch-analytisches Denkvermögen verfügen, wie Prozessdesigner oder Technical Writer. Bei längeren Anforderungsdokumenten bietet es sich an, dieses für die verschiedenen Prüfer zu unterteilen: Der Business Analyst ist dann beispielweise zuständig für den fachlichen, der Entwickler für den technischen Teil. Die Prüfung erfolgt am besten einzeln, nicht im Team.

Im Mittelpunkt der Qualitätsprüfung stehen folgende Fragen:

- Ist die Anforderung noch aktuell?
- Ist sie notwendig? Ist sie sinnvoll?
- Passen die verschiedenen Requirements zusammen oder gibt es Widersprüche?
- Wurde ein Begriff in unterschiedlichen Bedeutungen verwendet?

Identifizierte Mängel, unklare Punkte und Fragen werden zunächst nicht bearbeitet, sondern nur vermerkt. Denn die besten Ergebnisse werden erzielt, wenn Qualitätsprüfung und Korrektur nacheinander in zwei getrennten Durchläufen erfolgen. Der erste steht im Zeichen der Kritik und konzentriert sich auf die Fehlersuche; im zweiten wird umgeschaltet auf "Kreativitätsmodus" und es gilt, Lösungen zu erarbeiten.

Anforderungen korrigieren

Die Behebung der Mängel und Klärung der offenen Fragen geschieht im Korrekturdurchlauf. Viele Lösungen ergeben sich innerhalb des Fachbereichs, in unserem Beispiel etwa die Frage, wie viele Fotos pro Jingle als Möglichkeit vorgesehen werden sollen. Andere Fragen, etwa wie das Mahnwesen bei ausstehenden Zahlungen oder das Recht auf Rücktritt vom Kaufvertrag organisiert sein sollen, sind besser bereichsübergreifend zu bearbeiten, denn hier gilt es, z.B. kommerzielle, juristische und technische Aspekte zu berücksichtigen. Es ist nicht ungewöhnlich, dass zwischen Fachbereich und Entwicklungsabteilung mehrere Feedbackschleifen durchlaufen werden, um einzugrenzen, was mit vertretbarem Aufwand machbar ist und was nicht. Bei größerem Änderungsumfang bietet sich deshalb ein gemeinsamer Termin unter Beteiligung der Entwickler an, bei dem alle Fragen, auch zu technischen Möglichkeiten und Aufwänden, abschließend beantwortet werden.

Sind Konflikte nicht abschließend lösbar, ist es Aufgabe der Projektleitung, die möglichen Optionen inklusive ihrer inhaltlichen, technischen und finanziellen Konsequenzen zu beschreiben und die Entscheidung an die verantwortliche Instanz zu eskalieren. Während der initialen Anforderungserstellung und im Verlauf des Projekts ist das üblicherweise der Produktmanager, nach der Betriebsübergabe der Lenkungsausschuss bzw. die ITIL-Rolle "Change Control Board" (CCB). Das Change Control Board setzt sich aus Stakeholdern zusammen, die fachlich und finanziell von den Änderungen betroffen sind und deren Beratern. In der Regel sind dies der Auftraggeber, die Projektleitung, deren Vorgesetzter, die Leitung der IT-Abteilung, mindestens ein fachlich versiertes Mitglied des Entwicklerteams und der Business Analyst.

Wenn alle Ergebnisse in das Anforderungsdokument eingearbeitet sind, ist es sinnvoll, die geänderten Anforderungen mit den Stakeholdern durchzusprechen, die die Änderungen eingereicht haben, um sicherzustellen, dass alles verstanden und berücksichtigt wurde.

Spätestens jetzt steht einer Abnahme des geänderten Projektauftrags durch den Auftraggeber nichts mehr entgegen. Dabei ist eine schriftliche Form dieser Freigabe unter Bezug auf das vollständige Anforderungsdokument dringend geboten, denn die Anforderungen sind die Grundlage für den Vertrag zwischen Projekt und Kunde, unabhängig davon, ob es sich um ein intern oder extern vergebenes Dienstleistungsverhältnis handelt.

Am Ende der Qualitätssicherung steht eine Liste von geprüften, fehlerbereinigten und abgenommenen bzw. freigegebenen Anforderungen.

Anforderungen priorisieren

Entscheidungskriterien festlegen

Da nun unter Umständen ein langer Anforderungskatalog entstanden ist, aber nicht alles sofort erledigt werden kann, kann es notwendig sein, die Anforderungen mit Prioritäten zu versehen. Die Projektleitung erarbeitet dazu einen Vorschlag für eine Priorisierung der Anforderungen

und stimmt diese mit dem Auftraggeber ab. Dabei liegen manche Entscheidungen auf der Hand. Mit hoher Priorität werden in aller Regel eingeschätzt:

- Realisierung von gesetzlichen und unternehmenseigenen Vorgaben (z.B. Widerrufsrecht, Corporate Design)
- Basisfunktionalitäten, ohne die das System nicht sinnvoll genutzt werden kann (z.B. Hochlade- und Bestellfunktion)
- Beiträge zum Alleinstellungsmerkmal (z.B. Marktplatzfunktion für unabhängige Anbieter)
- Merkmale, die den Marktstart erleichtern (z.B. einfache Kaufvertragsabwicklung)

Mit niedriger Priorität hingegen werden in der Regel eingeschätzt:

- Hilfs- und Beistellfunktionen (z.B. kontextsensitive Hilfe, umfangreiches Schulungsangebot)
- Unterstützungsfunktionen (Vereinfachung der Handhabbarkeit, Expertenmodus)
- Ergänzende Funktionen (selten genutzte Angebote wie Mengenrabatte)
- Zusatzfunktionen, die wenig Mehrwert bieten (z.B. die Darstellung der Cover als Bilderkarussell)

Bei der Festlegung der Priorität können Produktmanager, Entwickler und Key User den Projektleiter unterstützen.

Für Priorisierung strukturiert vorgehen

Sind die Anforderungen – wie in der vorausgegangenen Artikelserie beschrieben – gemäß der SOPHIST-Satzschablone formuliert, ist ihre Priorität bereits mit den Wörtern MUSS, KANN und WIRD im Anforderungstext kenntlich gemacht. Dies reicht aber nicht aus, da eine Pflege der Priorität im Text der Anforderung zu einem immensen Änderungsaufwand führen kann. Weiterhin ist es grundsätzlich kritisch, eine Priorisierung für die Zukunft vorzunehmen und eine Sortierung der Anforderungstabelle nach Priorität ist nicht möglich.

Zudem hat die Priorisierung zwei Dimensionen: Dringlichkeit (zeitliche Priorität) und Verbindlichkeit. Dringlichkeit beantwortet die Frage: Wie kurzfristig muss diese Funktion realisiert werden? Verbindlichkeit beantwortet die Frage: Wird diese Funktion unbedingt benötigt oder kann sie zurückgestellt werden?

Es ist somit sinnvoll, die Priorität mit den beiden Merkmalen Dringlichkeit und Verbindlichkeit auszudrücken und diesen jeweils eine eigene Spalte in der Anforderungstabelle zu widmen. Eine erprobte Kategorisierung für die zeitliche Priorisierung wäre "1 - kritisch", "2 - wichtig", "3 - mittel" und "4 - niedrig". Die Spalte "Verbindlichkeit" unterscheidet die Optionen "zwingend" und "optional".

Die zeitliche Priorisierung lässt sich direkt in die Releaseplanung übernehmen. Die Priorität "1 - kritisch" wird in Release 1.0 aufgenommen, "2 - wichtig" in Release 2.0, usw. Je später die

Funktion eingeplant wird, desto unschärfer gerät die Priorisierung und muss gegebenenfalls für folgende Releases erneut überprüft werden. Sind zwei oder mehr Releases geplant, gilt weiterhin, alle kritischen Anforderungen im Release 1.0 und die weiteren später einzuplanen. Entscheidend dafür sollte immer die Kundenperspektive sein, die vom Produktmanager und von den Key Usern vertreten wird. Spielen Marktgegebenheiten wie Wettbewerbsdruck und Expansionsstrategien eine wichtige Rolle bei der Priorisierung, sind Marketing und Produktmanager die entscheidenden Ansprechpartner für die Releaseplanung.

Synergien können genutzt werden, indem ähnliche oder verwandte Funktionen zusammen realisiert werden – ein Optimierungspotenzial, das vor allem von Entwicklern erkannt und eingeschätzt werden kann.

Beispiel Jingleshop: Priorisierte Anforderungstabelle

Einen Auszug aus der priorisierten Anforderungsliste enthält Tabelle 2. Aus Gründen der Darstellbarkeit wurde auf die Spalten "Offene Punkte", "Quelle" und "Hinzugefügt am" verzichtet.

Nr.	Name	Beschreibung und Zielsetzung	Release	Verbindlichkeit	Vorbedingungen	Fallbeispiel
Req-21	Anmeldung mit Benutzername und Kennwort	Der Kunde muss jederzeit in der Lage sein, sich mit Benutzername und Kennwort am System anzumelden.	1.0	zwingend	Anmeldung ist noch nicht erfolgt Benutzer ist registriert	Der Kunde befindet sich in einem Bestellprozess und möchte sich anmelden.
Req-22	Eigenschaftenfeld Benutzername: Länge	Das System muss fähig sein, einen Benutzernamen mit der Länge von 130 Zeichen zu verarbeiten.	3.0	zwingend	Keine	Keine
Req-23	Eigenschaftenfeld Kennwort: Länge	Das System muss fähig sein, ein Kennwort mit der Länge von 20 Zeichen zu verarbeiten.	3.0	zwingend	Keine	Keine
Req-161	Upload durch Stellvertreter	Das System muss einem Stellvertreter die Möglichkeit bieten, Jingles hochzuladen.	3.0	optional	Keine	Ein Agent vertritt mehrere Komponisten.

Tabelle 2: Anforderungen werden mit zeitlicher Priorität und Verbindlichkeit versehen (Auszug).

Für die Planung hat es sich als hilfreich erwiesen, wenn die zu realisierenden Features in etwa ausgewogen auf die Releases verteilt werden. Orientierungspunkt ist dabei weniger die Anzahl der Anforderungen als der von den Entwicklern geschätzte Zeitaufwand für die Umsetzung.

Änderungen von Anforderungen verwalten

Es ist völlig normal, dass sich im Laufe eines Entwicklungsprojekts oder bereits während der Erstellung des Anforderungsdokuments Änderungen ergeben. Die Wahrscheinlichkeit dafür ist umso höher, je länger eine Software-Entwicklung dauert, aber auch je dynamischer das Projektumfeld ist. Auslöser außerhalb des Projekts können z.B. das Verhalten der Wettbewerber, die Einführung von neuen Technologien oder Gesetzesänderungen sein. Auch Änderungen der Interessenslage im Unternehmen und Budgetkürzungen können zu Änderungen von Anforderungen führen.

Doch auch im Projekt selbst können sich Gründe ergeben, die Anforderungen zu ändern, wie z.B. neue Erkenntnisse der Entwickler zu Aufwänden, Risiken und Chancen sowie Verbesserungsvorschläge der Anwender vor allem nach ersten Systemtests.

Darüber hinaus kann die Software selbst Grund für eine Änderung der Anforderungen sein, etwa wenn sich herausstellt, dass es wider Erwarten zu einem fehlerhaften oder nicht gesetzeskonformen Systemverhalten kommt. Folgendes Beispiel skizziert ein solch fehlerhaftes Systemverhalten: Stammkunden werden Rabatte gewährt. Bei jeder Bestellung wird die Bestellhistorie des Kunden abgeprüft, um zu ermitteln, ob es sich um einen Stammkunden handelt. Im Test mit einer geringen Anzahl von Bestellvorgängen war die Performance einwandfrei, aber im Laufe der Zeit steigt die Anzahl der Einträge in der Bestellhistorie, so dass es zu immer längeren Antwortzeiten kommt.

Klaren Prozess für neue Anforderungen definieren

In jedem Fall sind bei der Software-Entwicklung Änderungen von Anforderungen einzuplanen, denn diese haben Auswirkungen auf die Zeit- und Budgetplanung, den Fertigstellungstermin und Projektrisiken, gegebenenfalls auch auf andere Projekte. Deswegen ist es dringend geboten, einen klaren Prozess für die Verwaltung von Anforderungen und für den Umgang mit Änderungswünschen (Change Requests) festzulegen und zu implementieren. In dem Prozess sollte eindeutig definiert sein,

- wer die Verantwortung für die Anforderungsdokumentation hat,
- wie und wo die Änderungen dokumentiert werden,
- in welcher Form und mit welchen Informationen Änderungsanträge gestellt werden,
- wer wann wie oft geänderte und neu hinzugekommene Anforderungen analysiert, korrigiert und abstimmt und
- wer wann neue Anforderungen priorisiert und genehmigt.

Change-Prozess umsetzen

Folgende Festlegungen haben sich in kleineren Software-Projekten bewährt: Generell sollte nur ein einziges aktuelles und verbindliches Anforderungsdokument existieren. Darauf sollte nur eine Instanz Schreibrechte haben, nämlich die Projektleitung oder die Teilprojektleitung

"Software-Entwicklung" sowie deren Stellvertreter. Diese hat unbedingt dafür zu sorgen, dass die Änderungshistorie nachvollziehbar dokumentiert wird.

Der Änderungsantrag sollte als einfaches, aber schriftliches Formular bereitgestellt werden, da bei weniger formellem Vorgehen, etwa über persönliche E-Mails, maßgebliche Informationen oft verloren gehen. Folgende Angaben sollten immer mit eingereicht werden:

- Datum des Änderungsantrags
- Bezug auf den Entwicklungsstand
- Name des Antragstellers
- Gewünschte Änderung in ein oder zwei erläuternden Sätzen
- Grund für die Änderung
- Erwarteter Nutzen der Änderung bzw. negative Folgen bei Nicht-Umsetzung
- Einschätzung der zeitlichen Priorität und Wichtigkeit
- Geschätzter Aufwand und Kosten

Sieht sich ein Antragsteller nicht in der Lage, Aussagen zu Priorität und Dringlichkeit sowie Aufwand und Kosten zu machen, können diese Felder leer gelassen und später bei der Bearbeitung des Änderungsantrags behandelt werden. Generell sind diese Werte im Rahmen der Entscheidungsvorbereitung zu verifizieren bzw. zu korrigieren.

Kurze Wege zur Bearbeitung von Änderungsanträgen bieten Teamsitzungen, in denen die Antragsteller ihre Vorschläge erläutern und Entwickler die Chance haben, unmittelbar darauf zu antworten und den Aufwand abschätzen – entweder quantitativ in Tagen oder zumindest qualitativ in "geringfügig", "mittel", "hoch". Diese Konstellation ermöglicht es zudem, gemeinsam darüber zu diskutieren, ob das angestrebte Ziel ggf. auf anderem Weg besser zu erreichen wäre. Sind auch Projektleitung und Auftraggeber mit dabei, steht einer schnellen Entscheidung, ob und wann die Änderung in die Programmierung einfließen soll, nichts mehr entgegen.

Änderungen ausreichend dokumentieren

Bei der weiteren Dokumentation der Änderungsanträge kann in gering budgetierten Projekten auf Formalismen verzichtet werden. Hier genügt es, in der Anforderungstabelle vier weitere Spalten für "Änderungsart" (neu, geändert oder gelöscht), "Geändert durch", "Genehmigt am" und "Genehmigt durch" einzufügen.

Handelt es sich um ein größer dimensioniertes Projekt, ist sinnvollerweise ein Change Control Board einzurichten, bei dem Änderungsanträge bzw. "Change Requests" formal eingereicht und begründet werden müssen. Die Mitglieder des Change Control Boards treffen sich regelmäßig, z.B. alle zwei Wochen oder monatlich. Sie beurteilen den Änderungsaufwand, führen eine Kosten-Nutzen-Analyse

durch, fassen den Beschluss, den Änderungsantrag zu genehmigen, abzulehnen oder zurückzustellen, versehen ihn im positiven Fall mit Priorität und Verbindlichkeit und weisen ihn einem Release zu.

Das Anforderungsdokument ist für das Entwicklungsteam verbindlich. Werden nach der Beauftragung Änderungen vorgenommen, ist es erforderlich, den ursprünglichen Auftrag um die neuen Punkte zu ergänzen. Unter Umständen muss der interne oder externe Auftragnehmer eine neue Aufwandsschätzung vornehmen und es ergeben sich auf Seiten des Auftraggebers Zusatzkosten für die Umsetzung.

Fazit

Das Erstellen, Dokumentieren und Präzisieren von Anforderungen ist der erste große Anforderungsblock im Requirements Engineering. Doch die Ergebnisse sind keineswegs statisch. In einem weiteren Schritt sind wie im Artikel beschrieben Widersprüche und Fehler zu beseitigen. Danach sind die Anforderungen bereit für die Festlegung der Verbindlichkeit und der zeitlichen Priorität.

Nach Beginn der Umsetzung ist es erforderlich, die Änderungen mit der gleichen Sorgfalt zu dokumentieren, mit denen die initialen Anforderungen erarbeitet worden sind. Änderungen in der Implementierung "auf Zuruf" sind unbedingt zu vermeiden, da diese später nicht mehr nachvollzogen werden können und schnell zu Inkonsistenzen und Problemen bei der Abnahme führen.

Projektleiter, die (auch) für Software-Entwicklung zuständig sind, profitieren enorm von diesem systematischen Vorgehen im Requirements Engineering. Hier ausreichend Ressourcen einzusetzen, zahlt sich im späteren Projektverlauf mehrfach aus. Es ist ein entscheidender Erfolgsfaktor für die Einhaltung von Zeit und Budget und erspart dem Projektteam belastende Diskussionen über Missverständnisse und unterschiedliche Interpretationen bei der Beauftragung und Umsetzung.

Literaturverzeichnis

- Pohl, Klaus; Rupp, Chris: Basiswissen Requirements Engineering: Aus- und Weiterbildung nach IREB-Standard zum Certified Professional for Requirements Engineering Foundation Level, dpunkt, 2015
- Grande, Marcus: 100 Minuten für Anforderungsmanagement: Kompaktes Wissen nicht nur für Projektleiter und Entwickler, Vieweg+Teubner, 2011

Verborgene Hindernisse aufdecken

Bessere Prozesse mit dem Impediment Backlog

Peter Rubarth
Leiter Softwareentwicklung

Management Summary

- Bei der Arbeit mit projekt- wie produktorientierten Prozessen kann es gleichartige Probleme geben.
- Wohl durchdachte und gemanagte Arbeitsweisen können dennoch zu unerwarteten und nachteiligen Nebeneffekten führen.
- Besser, als nach einer neuen, vermeintlich besseren Struktur zu suchen, ist bereichs- und hierarchieübergreifend ein Impediment Backlog aufzubauen und zu bearbeiten.
- Ein gemeinsames Verständnis von unbeabsichtigten Nebeneffekten etablierter Prozesse ist dabei zentral.
- Um Veränderungen anzustoßen, werden begrenzte Experimente konzipiert und durchgeführt. Mit ihnen werden Ideen getestet und die Entstehung neuer Nebeneffekte vermieden.
- Für den Prozess wird eine übergreifende Arbeitsgruppe gebildet, die Experimente entwickelt und begleitet.
- Entscheidend für den Erfolg ist ein offener Umgang mit den Experimenten, Unterstützung durch die Geschäftsführung und Glaubwürdigkeit bei den Mitarbeitern.

Dieser Tage wird viel darüber geschrieben, dass die Arbeit in Projekten überholt ist. Unter dem Hashtag #NoProjects werden die Argumente hierzu ausgetauscht. Im Juli 2018 ist ein Buch gleichen Namens erschienen. Die zentrale These des Buches ist, dass ein kontinuierlicher Ansatz zur Entwicklung von Produkten – und nicht zeitlich begrenzte Projekte – der richtige Weg zum Erfolg ist.

Bei dieser Diskussion entsteht leicht der Eindruck, dass es nur auf die richtige Struktur ankommt und sich damit alle Probleme lösen. Dieser Artikel beschreibt, warum es sein kann, dass weder Produkt- noch Projektorganisation bei der Produktentwicklung funktionieren.

Best Practice Produktorganisation?

Die Produktorganisation wird im Vergleich zur Arbeit in Projekten häufig als besserer Ansatz gesehen. Die Argumente dafür sind unter anderem:

- Bessere Ausrichtung an gemeinsamen wirtschaftlichen Zielen statt mehr oder weniger losgelöste Projektziele
- Flexible Reaktion auf sich ändernde Rahmenbedingungen gegenüber einem vor Beginn definierten und starren Projektscope, der zu erfüllen ist
- Einbindung aller benötigten Funktionen in einer interdisziplinären Struktur statt „Silo-Denke“ zwischen Fachabteilungen und entsprechend mühsamer Abstimmung
- Vermeidung von Problemen, die daraus resultieren, dass Projektmitarbeiter sowohl dem Projektleiter als auch dem Linienvorgesetzten zuarbeiten
- Teams bleiben langfristig zusammen und werden nicht für jedes Projekt neu geformt und dann wieder auseinander gerissen (was die Entwicklung zu einem performanten Team verhindert)

Zu Besuch bei IrgendeinShop.de

Wir befinden uns bei einem etablierten E-Commerce-Unternehmen mit ungefähr 500 Mitarbeitern, welches einen spezialisierten Online-Marktplatz betreibt. Bei IrgendeinShop.de handelt es sich um ein imaginäres Beispiel, in das Erfahrungen aus mehreren realen Unternehmen eingeflossen sind.

Ein zentrales Organisationselement von IrgendeinShop.de sind die vier agil organisierten Produktentwicklungsteams. Jedes dieser Teams fokussiert sich auf einen Aspekt des Geschäftsmodells. Die fachlichen Themen der Teams sind so festgelegt:

- **Anbieter**: Dieses Team kümmert sich um alle Funktionen der Plattform im Zusammenhang mit Benutzern, die Produkte auf dem Marktplatz anbieten.
- **Nachfrager**: Das Team verantwortet die Funktionalität für potentielle Käufer, die den Marktplatz nutzen.
- **Partnerschaften und Integrationen**: Das Team bindet Partner, z.B. für Finanzierungen an das eCommerce System an. Es geht vor allem um öffentliche Programmierschnittstellen (APIs) und nicht um Benutzeroberflächen.
- **Suchmaschinenoptimierung**: Dieses Team arbeitet eng mit dem Marketing zusammen und entwickelt Funktionen für die Findbarkeit in Suchmaschinen. Dazu gehört auch die Erstellung von Landing Pages für bezahlte Werbekampagnen in Suchmaschinen und Sozialen Medien.

Die Teams bestehen aus jeweils fünf bis neun Mitarbeitern. Dazu gehören Softwareentwickler mit unterschiedlicher Spezialisierung und Software-Tester. Grafikdesigner und andere Spezialfunktionen werden als gemeinsame Ressource eingesetzt. Jeweils ein Product Owner und ein Tech Lead leiten die Teams. Der Product Owner verantwortet das Product Backlog, erhebt und formuliert die Anforderungen und nimmt die Umsetzung ab. Der Tech Lead ist der fachliche und disziplinarische Vorgesetzte der Softwareentwickler und verantwortet die Softwarearchitektur.

Es gibt zwei Scrum Master, um die Teams zu unterstützen. Die Scrum Master sind dem Chief Technology Officer (CTO) unterstellt. Die Product Owner berichten an den Chief Product Officer (CPO). Die Teams arbeiten nach Scrum mit zweiwöchentlichen Sprints.

Ein Blick hinter die Kulissen bei IrgendeinShop.de

IrgendeinShop.de scheint ein Paradebeispiel für agil arbeitende Teams und agile Prozesse zu sein. Doch das Team stand dennoch vor einem Problem bzw. gleich mehreren im Zusammenhang mit einem etablierten Prozess: dem Project Pitch. In diesem Artikel nehme ich Sie mit zur Ausgangssituation dieses Prozesses, erkläre, warum Prozesse (egal ob in klassisch geleiteten oder agil abgewickelten Projekten) manchmal trotz besten Wissens und Gewissens nicht funktionieren und schildere als mögliche Lösung den Aufbau eines Impediments Backlogs sowie die Einführung zeitlich begrenzter Experimente wie im Beispielunternehmen.

Project Pitches mit abenteuerlichen Auswüchsen

Um die Arbeit der Teams bei IrgendeinShop.de zu priorisieren, wurde ein Prozess etabliert, bei dem interne Stakeholder ihre Anforderungen und den zu erwartenden Business Value präsentierten. Interne Stakeholder waren Führungskräfte aus anderen Unternehmensbereichen, wie dem Marketing, Business Development, Lager oder Einkauf. Die Präsentation erfolgte vor einem Gremium, zusammengesetzt aus Produktmanagement und Vertretern aller internen Stakeholder. Dieses Gremium musste sich auf die Priorität der vorgeschlagenen Initiativen einigen, wobei dem CPO das letzte Wort zukam. Anfragen mussten diesen Test bestehen und wurden dann in der Reihenfolge des kalkulierten Business Value in eine Roadmap aufgenommen.

Angenommene Initiativen wurden vom Chief Product Owner an die Product Owner zur Umsetzung mit den Teams übergeben. Die Aufgabe der Product Owner bestand dann darin, die Anforderungen der Stakeholder im Detail zu analysieren und mit den Teams umzusetzen. Dieses Vorgehen entsprach womöglich nicht der ursprünglichen Vorstellung eines Product Owners, ist jedoch eine sehr häufig anzutreffende Interpretation der Rolle.

Der Project Pitch Prozess brachte einige unerwünschte Nebeneffekte mit sich:

- Der Aufwand des Pitch lohnte sich nur für größere Initiativen. Kleinere, in der Summe womöglich wertvollere Initiativen wurden nicht verfolgt.
- Dem kalkulierten Business Value kam eine enorme Bedeutung zu. Um die Chancen auf eine Genehmigung zu erhöhen, wurden teilweise sehr optimistische Annahmen getroffen, z.B. zu den erwarteten Umsätzen oder Einsparungen.
- Die Stakeholder versuchten, sich gegenseitig die Annahmen zum geschätzten Business Value zu „zerlegen“, um dadurch bessere Chancen für ihre eigenen Initiativen zu bekommen. Letzten Endes waren sie darauf angewiesen, dass sich Entwickler Zeit für ihre Initiativen nah-

men, um ihre Ziele zu erreichen. Grundsätzlich ist es positiv zu sehen, wenn Annahmen geprüft werden. Allerdings sind sowohl Argumente als auch Gegenargumente unbestätigte Annahmen. Der Aufwand für diese Auseinandersetzungen verlangsamte den Prozess weiter. Zeit und Kosten, bis Annahmen mit realen Kunden bestätigt oder widerlegt wurden, stiegen.

- Stakeholder brachten alle möglichen Anforderungen mit in die Initiativen ein. Es ging darum, möglichst viel umzusetzen, wenn man schon einmal zum Zug kam. Dadurch vergrößerte sich auch die Verhandlungsmasse für unausweichliche spätere Kürzungen. Im Ergebnis landeten Anforderungen in der Umsetzung, die einzeln betrachtet weder dringend, noch wichtig waren.

Roadmaps, mit denen sich die Entwickler nicht identifizieren

Aus dem Pitch-Prozess resultierten Roadmaps, die der Chief Product Owner an die Product Owner der Teams übergab. Für die Teams war es schwierig, zu erkennen, warum genau diese Themen in dieser Reihenfolge in der Roadmap standen und dementsprechend den Nutzen ihrer Arbeit für den Unternehmenserfolg zu erkennen.

Die Frage, ob die Themen für die Teammitglieder plausibel waren oder nicht, war eine rhetorische, da die Umsetzung bereits beschlossen war.

Zusammenarbeit im Team – Wasserfall und Ping Pong

Die Arbeit im Sprint ließ sich mit dem Begriff „Mini-Wasserfall“ treffend beschreiben. Entwickler bearbeiteten parallel zueinander Tickets und gaben sie gegen Ende des Sprints an die Software-Tester. Jeder war der Meinung, seinen Teil getan zu haben. „Ping Pong“ zwischen Entwicklern und Testern war die Regel; die Erreichung von Sprintzielen Glück.

Zusammenarbeit zwischen Teams – gegeneinander statt miteinander

In der Zusammenarbeit zwischen Teams dominierten Abstimmungsprobleme, Schuldzuweisungen und der Fokus auf das Erreichen des „eigenen“ Sprintziels.

Lieferprobleme und Qualitätsprobleme als letztliche Folge

In der Folge der beschriebenen Symptome kam es insbesondere bei größeren Initiativen in der Regel zu erheblichen Zeitverzögerungen und Qualitätsproblemen.

Same Shit, different Name

Es gibt in der agilen Welt viele Scrum Master und Agile Coaches, die bei solchen Problemen einwenden, dass das Unternehmen Scrum eben nicht richtig einsetzt. Verfechter von Kanban argumentieren, dass es eben genau an Scrum und seinen starren Iterationen liegt. Und Verfechter klassischer Managementansätze (oft Führungskräfte aus nicht-technischen Bereichen) sind sich sicher, dass die Selbstorganisation ein

Irrweg ist und mit definierten Prozessen, abgegrenzter Verantwortung und sequenziellem Vorgehen es all diese Probleme nicht gäbe. Ein starker Projektleiter würde die Teams schon auf Linie bringen.

Aus meiner Sicht greifen all diese Argumente zu kurz. Die beschriebenen Symptome sind, unabhängig von der angewendeten Methode, so oder ähnlich immer wieder zu beobachten. Ob nach Scrum oder Kanban gearbeitet wird oder klassisch in Projekten, ist dabei nicht maßgeblich.

Grundsätzlich sind bestimmte Strukturen oder Prozesse besser geeignet, um eine Herausforderung zu bearbeiten als andere. In vielen Situationen sind dies agile Produktentwicklungsprozesse und agile Teams. Jedoch sind Strukturen und Prozesse kein Allheilmittel. Oft stecken hinter den oberflächlichen Problemen tieferliegende Ursachen, die durch andere Strukturen nicht beseitigt werden.

Strukturtransformation ist Symptombekämpfung!

„Produkt oder Projekt?“ ist die falsche Frage und lenkt lediglich davon ab, die eigentlichen Ursachen aufzudecken.

Dennoch wird oft eine aufwendige Transformation durchgeführt, um derartigen Problemen mit anderen, vermeintlich besseren Strukturen zu begegnen. Nachdem die Transformation unter Aufwand und Schmerzen bewältigt wurde, zeigt sich oft, dass die alten Probleme sich wieder zeigen - nur vielleicht mit etwas anderen Symptomen.

Ein alternativer Ansatz besteht darin, einen Prozess auf Unternehmensebene zu etablieren, mit dem fortlaufend Hindernisse identifiziert und beseitigt werden, statt nach einer Struktur oder einem Prozess zu suchen, mit dem „von Zauberhand“ alles besser wird.

Der ein oder andere Leser denkt jetzt bestimmt an den „Kontinuierlichen Verbesserungsprozess“ aus dem Lean Management. Und das ist kein Zufall. So, wie sich die agile Idee aus den Konzepten von Lean entwickelte, ist der hier beschriebene Ansatz nicht neu, sondern eine Adaption bestehender Konzepte.

Besser: Ungewollte Nebeneffekte aufspüren

Viel wichtiger als das "Herumdoktern" an Strukturen und Prozessen ist eine ehrliche Auseinandersetzung mit der Frage, welche Anteile an der aktuellen Arbeitsweise die gemeinsame Fokussierung auf Ergebnisse behindern und ungewollt zu unerwünschten Nebeneffekten führen. Dazu gehört insbesondere alles, was zu einer Optimierung auf einzelne Interessen hin einlädt und verhindert, über die Grenzen des eigenen Bereichs hinaus zu schauen, Verantwortung für den Gesamterfolg zu übernehmen und gemeinsam nach dem richtigen Weg zu suchen.

Ein paar typische Kandidaten für behindernde Arbeitsweisen sind:

Leistungsorientierte Vergütung anhand von Kennzahlen und Jahreszielen

- Kennzahlen sind sinnvoll. Sie bilden aber entweder nur einen kleinen Bereich ab oder sind wie der Gewinn zu allgemein, um handlungsleitend zu sein.

- Ein solches Vergütungssystem fordert dazu auf, das Gesamtbild zu vernachlässigen und auf die eigenen Performancekriterien hin zu optimieren.
- Jahresziele sind ungeeignet, um auf sich schnell verändernde Rahmenbedingungen zu reagieren. Entweder werden Entscheidungen getroffen, die nicht mehr mit der Lage übereinstimmen oder eine unterjährige Nachverhandlung findet statt und lenkt von den eigentlichen Zielen ab.

Planungs- und Priorisierungsverfahren

Umfangreiche Planungs- und Priorisierungsverfahren, um eine interessengeleitete Priorisierung auszugleichen, verlangsamen den Umsetzungsprozess und erhöhen den Abstand zwischen den Bereichen mit Kundenkontakt und den Umsetzungsteams. Der Zusammenhang geht verloren und die Umsetzungsteams optimieren auf andere Ziele hin, z.B. die eigene Karriere.

Fokus auf Performancekennzahlen

Die Fokussierung auf abstrakte Performancekennzahlen (wie Velocity) lädt zu überhöhten Schätzungen und Abstrichen bei der Qualität ein.

Impediment Backlog: Eine gemeinsame Sichtweise schaffen

Ein geeigneter Start für einen solchen Verbesserungsprozess kann sein, intern nach solchen Regeln oder Prozessen zu suchen, die für sich genommen sinnvoll erscheinen, aber zu ungewollten Nebeneffekten führen.

Im agilen Umfeld werden alle Aspekte, die eine ergebnisorientierte Arbeit behindern, **Impediments** (Hindernisse) genannt. Im weiteren Verlauf des Artikels wird der Begriff Impediment für all diese ungewollten Nebeneffekte oder gut gemeinten, aber unwirksamen Prozesse verwendet, die eine bereichsübergreifende, ergebnisorientierte Arbeit behindern.

Das Ziel ist zunächst, eine gemeinsame Vorstellung von den Impediments zu bekommen, diese strukturiert zu erfassen und zu priorisieren. Das Ergebnis dieses Prozesses ist ein **Impediment Backlog**, auf dessen Basis eine Arbeitsgruppe Experimente konstruiert, um alternative Arbeitsweisen zu testen, die die Impediments auflösen.

Kick-off-Workshop planen

Ein guter Weg zu diesem Impediment Backlog ist ein **Workshop**, bei dem Vertreter verschiedener Bereiche und Ebenen zusammenkommen. Wichtig für den Erfolg der Veranstaltung ist ein klares Mandat der Geschäftsführung und, dass die eingeladenen Teilnehmer von der Belegschaft als Vertreter anerkannt werden. Gute Kandidaten sind meiner Erfahrung nach Führungskräfte der mittleren Ebene.

Arbeitsgruppe festlegen

Damit die Arbeit des Workshops von allen anerkannt wird, sollten jedoch Vertreter aus allen Ebe-

nen dabei sein, insbesondere auch von der Basis. Sogar Werkstudenten oder Praktikanten können mit ihrer unvoreingenommenen Sichtweise wertvollen Input liefern. Ein guter Ansatz ist, erst die zu repräsentierenden Ebenen und Bereiche festzulegen und diese Gruppen dann ihre Vertreter durch eine Wahl bestimmen zu lassen.

Eine gute Gruppengröße ist zwischen 10 und 15. Ist für die Repräsentation aller relevanten Gruppen eine größere Teilnehmerzahl notwendig, muss der Workshop entsprechend komplexer aufgebaut werden. In diesem Fall ist die Arbeit in Teilgruppen zwingend erforderlich und die Selektion dieser und der Austausch zwischen den Gruppen muss organisiert werden. Um solch einen großen Workshop effektiv durchzuführen, sind in der Regel mehrere professionelle Moderatoren erforderlich.

Kick-off-Workshop durchführen

Der Ablauf eines solchen Workshops kann so aussehen:

1. Begrüßung der Teilnehmer und Vorstellung
2. Vorstellung der Ziele der Veranstaltung durch die Geschäftsführung
3. Vereinbarung von Spielregeln (z.B. keine Mobiltelefone, Du oder Sie, wertschätzender Umgang, Einwände sofort ansprechen u.Ä.)
4. Vorstellung des Impediment Konzepts
5. Erhebung von Impediments (z.B. Brainstorming in Kleingruppenarbeit)
6. Vorstellung der Ergebnisse des Brainstormings und Konsolidierung
7. Überprüfung auf Übereinstimmungen und unterschiedliche Perspektiven
8. Priorisierung der gefundenen Impediments
9. Erstellung des Impediment Backlog
10. Ausarbeitung der weiteren Schritte
11. Resümee des Workshops und Verabschiedung

Der Workshop sollte nicht im Verborgenen erfolgen, um der Entstehung von Gerüchten vorzubeugen. Damit ist jedoch nicht gemeint, dass jeder nach Belieben die Arbeitsgruppe beim Workshop stören kann. Vielmehr geht es darum, dass der Workshop von der Geschäftsführung angekündigt wird (als Rundmail, im Intranet oder andere Kommunikationswege):

- Was wird gemacht? Warum?
- Wer nimmt teil? Nach welchem Prinzip werden die Teilnehmer ausgewählt? Warum?
- Was passiert danach?

Nach der Durchführung sollten die Ergebnisse auf demselben Kanal geteilt werden. Durch diese Offenheit ist es möglich, von Anfang an eine breite Unterstützung zu erreichen.

Vorstellung der Ziele

Ein guter Einstieg in die Vorstellung der Ziele ist, mit der Notwendigkeit zu beginnen. Welche Herausforderungen bestehen aus Sicht der Geschäftsführung und woran konkret zeigt sich, dass diese mit den bisherigen Arbeitsweisen unzureichend bewältigt werden.

Im zweiten Teil der Einführung erläutert Geschäftsführung, wo das Unternehmen im Hinblick auf Arbeitsweisen in z.B. 6 Monaten stehen soll und was dann ganz konkret besser ist. Ein Beispiel: Am 30. Juni 2019 haben wir unsere Innovationsfähigkeit verbessert, 5 konkrete Experimente durchgeführt und die Ergebnisse offen geteilt und besprochen.

Der dritte Teil ist eine Erläuterung, warum das Impediment Konzept ausgewählt wurde, welche Resultate am Ende des Workshops erwartet werden, wie diese der erste Schritt zur Erreichung des vorher benannten Ziels sind und was der nächste Schritt sein wird.

Sind bislang Powerpoint Präsentationen am Konferenztisch die Norm, wäre eine Vorstellung mit Flipchart im Stuhlkreis vielleicht eine schöne Gelegenheit zu zeigen, dass tatsächlich etwas fühlbar anders gemacht und bisherige Arbeitsweisen hinterfragt werden sollen.

Vorstellung des Impediment Konzepts

Auch für die Vorstellung des Impediment Konzepts bietet sich Flipchart oder Whiteboard an. Der Moderator erklärt anhand von Beispielen ungewollte Nebeneffekte außerhalb des Unternehmensumfelds, z.B. aus dem Bereich Kindererziehung. Zu diesem Thema haben viele einen persönlichen Bezug, es ist weit genug weg vom professionellen Kontext und komplex genug, dass sich Beispiele finden lassen, so wie das hier: “Stellen Sie sich vor, Sie haben mit ihrem Kind vereinbart, dass es sich Punkte verdient, wenn es Aufgaben im Haushalt erledigt und diese dann in Spielzeug eintauschen kann. Welche ungewollten Nebeneffekte können Sie sich vorstellen?".

Erhebung von Impediments

Ist den Teilnehmern des Workshops klar, nach welchen unerwünschten Nebeneffekten gesucht wird, leitet der Moderator (Methodenexperte, mittlere Führungskraft) zu den Impediments des Unternehmens oder -bereiches über. Wichtig bei der Erfassung der Nebeneffekte ist, im Blick zu behalten, dass die gefundenen Impediments und die identifizierten Zusammenhänge Annahmen sind. Diese Annahmen wurden von den Teilnehmern des Findungsprozesses als plausibel bewertet. Das bedeutet aber auch, dass andere Erklärungen ebenfalls möglich, nur noch nicht bekannt sind. Es geht auch nicht darum, herauszufinden, was jetzt richtig ist. Das Zwischenziel ist, strukturiert und mit möglichst wenig Aufwand Kandidaten für Impediments zu finden.

Vorstellung der Impediments

Das Ergebnis dieser Arbeit (Schritt 4 bis 7) ist ein gemeinsamer Überblick über die identifizierten Impediments. Dieser Überblick für sich kann einen großen Beitrag dazu leisten, dass Beteiligte aus ihrer bis-

herigen, auf den eigenen Bereich begrenzten Perspektive heraustreten und die Zusammenhänge würdigen. Möglicherweise lassen sich auch bereits wiederkehrende Muster erkennen. Damit sind Wechselwirkungen zwischen bisher isoliert betrachteten Effekten gemeint oder gemeinsame Ursachen.

Ein Beispiel für Wechselwirkungen könnte sein: “Unsere Projekte dauern immer so lange, dass man selten zum Zug kommt. Deswegen packen wir möglichst viel in die Projekte rein, wodurch die Projekte lange dauern.”

Ein Beispiel für gemeinsame Ursachen: “Wir orientieren uns alle an unseren jeweiligen Zielvorgaben und KPIs und werden nach deren Erreichung bewertet. Wenn ein anderer Bereich Unterstützung braucht, machen wir das nur widerwillig, weil es sich nicht in unseren KPIs abbildet. Auch denken wir selten über Zusammenhänge mit anderen Bereichen nach, da uns auch das (zumindest kurzfristig) nicht mit unsern jeweiligen KPIs hilft”.

Priorisierung der Impediments

Die Arbeit im Workshop geht weiter, indem die identifizierten Impediments durch die Teilnehmer nach Ausmaß der Behinderung (bzw. Verbesserungspotenzial) und nach Aufwand zur Behebung bewertet werden.

Die Priorisierungskriterien sollten die Teilnehmer dabei selbst festlegen. Eine einfache Form ist, dass die Teilnehmer individuell Wichtigkeit und Aufwand der einzelnen Themen auf einer Skala einordnen und der Mittelwert aller abgegebenen Stimmen verwendet wird.

Thema	Wichtigkeit (1-10)	Aufwand (1-10)
Aufgeblähte Projekte	8	10
Stakeholder brauchen zu lange für Abnahme	6	3
Viel Zeit zur Koordination verbraucht	4	8

Tabelle 1: Eine Tabelle mit Wichtigkeit und Aufwand ermöglicht einen schnellen Überblick, welches Thema am wichtigsten ist bei überschaubarem Aufwand.

Das gewählte Verfahren und konkrete Ergebnis sind dabei nicht entscheidend. Wichtig ist, dass die Beteiligten ein gemeinsames Verständnis entwickeln und eine Selektion treffen, die klein genug ist, um bearbeitet werden zu können.

Sie sollten auf jeden Fall vermeiden, dass die Priorisierung der Themen hinter verschlossenen Türen erfolgt oder, dass im Nachgang des Workshops Themen von der Geschäftsführung „kassiert“ werden. Im ersten Fall werden immer Gerüchte aufkommen, was da jetzt aus welchen Motiven heraus passiert ist. Letzteres entwertet die Arbeit des Workshops und wird es sehr schwer machen, dass sich Mitarbeiter im weiteren Prozess einbringen.

Die Priorisierung kann im ersten Workshop erfolgen. Oftmals ist es besser, ein oder zwei Wochen später einen zweiten Termin zu vereinbaren, da so Gelegenheit zur Reflexion besteht. Ein

Folgetermin bietet auch die Möglichkeit, die Gruppe der Teilnehmer zu verändern. So können z.B. mehr Vertreter aus allen Bereichen einbezogen werden oder sogar die gesamte Belegschaft - sofern das im Hinblick auf die Unternehmensgröße noch praktikabel ist. Die ursprüngliche Gruppe stellt ihre Ergebnisse vor, andere Teilnehmer können Fragen stellen und über die Wichtigkeit der einzelnen Themen abstimmen. Zum Abschluss würdigt die Geschäftsführung die Ergebnisse und nimmt gegebenenfalls Themen begründet heraus.

Das Impediment Backlog erstellen

Als Resultat dieser Arbeit steht ein Impediment Backlog zur Verfügung, das die aus Sicht der Arbeitsgruppe plausibelsten Impediments, geordnet nach der empfohlenen Bearbeitungsreihenfolge enthält. Wie ein Product Backlog auch, ist dies kein Bericht, sondern ein lebendiges Dokument, das regelmäßig an neue Erkenntnisse angepasst wird. Prioritäten werden geändert, Impediments hinzugenommen oder als erledigt markiert.

Als Teil des Prozesses müssen Sie festlegen, wer die Hoheit über dieses Backlog hat – also der „Product Owner“ ist. Dies kann die Geschäftsführung sein, die Arbeitsgruppe oder eine andere konkrete Person. Wie auch immer die Entscheidung aussieht, Änderungen müssen nachvollziehbar und transparent erfolgen und nicht beliebig.

Veränderungsexperimente konstruieren

Experimente konstruieren

Der weitere Verlauf folgt dem agilen Grundsatz des „Inspect & Adapt“ (also Untersuchen & Anpassen). Während es im Workshop darum ging, Kandidaten für die Weiterarbeit auszumachen, geht es jetzt darum, diese konkreten Impediments zu untersuchen und Veränderungen an vielversprechenden Stellen auszuprobieren.

(Neue) Arbeitsgruppe festlegen

Für diesen Schritt muss neu festgelegt werden, wer dafür zuständig ist. Im Hinblick auf die Akzeptanz ist es günstig, eine bereichs- und hierarchieübergreifende, offen arbeitende Arbeitsgruppe einzusetzen. Fünf bis neun Mitglieder sind dabei eine gute Größe, um effektiv zu arbeiten. Die Teilnehmer des ersten Workshops sind gute Kandidaten für diese Gruppe. Die Befugnisse dieser Gruppe sind ebenfalls zu klären. Dazu wird von der Geschäftsführung einmal formal beschrieben, was die Gruppe darf, z.B.:

- Die Gruppe berichtet monatlich an die Geschäftsführung über den Inhalt des Impediment Backlogs und Fortschritte. Die Geschäftsführung kann dabei Einfluss auf die Priorisierung nehmen.
- Die Gruppe kann Gespräche mit allen Mitarbeitern im Unternehmen führen. Die jeweiligen Vorgesetzten sind über die Durchführung der Gespräche und die Zielsetzung zu informieren. Inwieweit Inhalte der Gespräche mit den Vorgesetzten besprochen werden, obliegt der Gruppe.

- Die Gruppe kann Einschätzungen und Empfehlungen zu den bearbeiteten Themen aussprechen. Diese Einschätzungen werden mit der Geschäftsführung und der Leitung der beteiligten Bereiche geteilt. Im Ermessen der Gruppe können die Einschätzungen auch darüber hinaus verfügbar gemacht werden.

Die Aufgabe der Gruppe ist es, Impulse zu setzen. Meine Empfehlung ist, dass die Gruppe Hoheit über das Impediment Backlog hat und Gespräche mit jedem im Unternehmen ansetzen kann, wenn sie das für richtig hält. Die Gruppe gibt dann auf dieser Grundlage Einschätzungen und Empfehlungen ab oder organisiert Workshops, damit die direkt Beteiligten konkrete Maßnahmen erarbeiten können. Ob und welche Maßnahmen ergriffen werden, obliegt den direkt Beteiligten.

Wichtig ist auch hier, die Unterstützung der Geschäftsführung sicher zu stellen. Niemand gewinnt, wenn die Arbeitsgruppe Themen startet und dann „eingefangen“ wird. Ein guter Weg ist deshalb ein regelmäßiges Impediment Backlog Review, bei dem die Geschäftsführung teilnimmt und ein Vetorecht hat. Bei diesem Termin werden Veränderungen am Impediment Backlog besprochen und Prioritäten überprüft. Dieses Review sollte wiederum offen für jeden Interessierten sein. Die Häufigkeit dieser Review hängt vom Unternehmen ab und liegt in der Regel zwischen sechs und zwölf Wochen.

Bei einem Veto der Geschäftsführung wird dieses mit Begründung dokumentiert und das Thema geschlossen. Möglicherweise zeigt sich später, dass das Thema doch weiter relevant ist und die Geschäftsführung ändert ihre Sichtweise. Auch möglich, dass durch eine andere Maßnahme, an dieser Stelle ebenfalls eine Verbesserung eintritt und sich das Thema auf diese Weise erledigt.

Impulsgeber finden

Die Arbeitsgruppe kann alleine keine Veränderungen durchführen. Für jedes konkrete Thema sind die relevanten Personen zu identifizieren und einzubeziehen. Dabei schaut sich die Arbeitsgruppe an, welche Bereiche betroffen sind. Dann werden neben den formal verantwortlichen Führungskräften noch diejenigen Mitarbeiter hinzugezogen, die in diesem Bereich hohes Ansehen genießen. Sie dienen als Impulsgeber für die jeweiligen Bereiche.

Wenn es also um Themen der Produktentwicklung geht, sind Leiter des Produktmanagements und Leiter der Entwicklung gute Kandidaten. Neben diesen wird z.B. mit einem Entwickler gesprochen, der lange im Unternehmen ist und immer „die Kastanien aus dem Feuer holt“, wenn irgendetwas schiefläuft, und mit der jungen Produktmanagerin, die sich sehr offen und kritisch zur aktuellen Arbeitsweise äußert.

Die Auseinandersetzung mit den fachlich Verantwortlichen ist nicht leicht, da es darum geht, bisher als richtig oder notwendig erachtete Vorgehensweisen auf den Prüfstand zu stellen. Eine gute Vorbereitung dieser Gespräche und eine gute Illustration der Impediments ist entscheidend.

Wichtig ist an dieser Stelle ebenfalls, Vorwürfe zu vermeiden und die gute Absicht hinter der bisherigen Arbeitsweise zu würdigen. Die meisten Regeln und Prozesse wurden einmal aus guter Absicht eingeführt. Es kann jedoch sein, dass sich mit der Zeit unerwartete Nebeneffekte eingestellt haben. Oder die

Rahmenbedingung haben sich so verändert, dass das Festhalten an der bisherigen Vorgehensweise jetzt nachteilig ist. Diesen Umstand anzuerkennen, ist für die Beteiligten eine große Herausforderung.

Lösungen erarbeiten

Liegt eine grundlegende Übereinstimmung mit den fachlich Verantwortlichen über das Vorhandensein eines Impediments vor, kann die Arbeitsgruppe beginnen, mögliche Lösungen zu erarbeiten.

Die Veränderungen als Experiment zu betrachten, ist wichtig. So wie die bisherigen Regeln und Prozesse mit besten Absichten eingeführt wurden, ist es auch bei den angestrebten Veränderungen. Nur weil eine Idee in der Theorie plausibel ist, bringt ihre Umsetzung nicht automatisch eine Verbesserung und kann ebenfalls unerwartete Nebeneffekte zeigen. Es wäre aber ein Fehler, deshalb keine Veränderungen durchzuführen.

Der Königsweg ist, Veränderungen so zu konstruieren, dass sie mit geringem Risiko ausprobiert werden können. Dadurch wird die Veränderung in der Praxis erlebbar und die angenommene positive Wirkung oder unerwartete Nebeneffekte können sich zeigen.

Ein gutes Experiment zeichnet sich dadurch aus, dass

- Aussagen über die Wirkung getroffen werden können, die über das Experiment heraus Gültigkeit haben (also kein Laboreffekt, bei dem das Verhalten nur im Labor eintritt, aber nicht in der „freien Wildbahn“).
- die Veränderung ohne erheblichen Aufwand rückgängig gemacht werden kann, wenn sich die Erwartungen nicht erfüllen oder erhebliche negative Nebeneffekte auftreten.
- die Veränderung keine erheblichen Risiken birgt, deren Kosten den Nutzen der angestrebten Verbesserung übersteigen würden (z.B. der Verlust einer Zulassung oder eines wichtigen Kunden).

Die Definition solcher Experimente ist erfahrungsgemäß oftmals schwierig. Es ist durchaus möglich, dass Aufwand betrieben werden muss, um notwendige Voraussetzungen erst zu schaffen, z.B. durch die Einführung eine Technologie oder eines (temporären) Prozesses, um eine bestimme Anzahl von Fällen eines Typs durch einen neuen Prozess abzuwickeln und die Masse über das bisherige Verfahren (in der Webentwicklung ist dieses Verfahren als A/B Test bekannt und de-facto Standard).

Bei einem A/B Test werden beispielsweise unterschiedliche Layouts oder unterschiedliche Formulierungen von Vorteilen getestet. Dafür wird eine Variante für jede zu testende Version erstellt. Diese Varianten werden dann zufällig ausgewählten Kunden angezeigt. Es wird gemessen, inwieweit dabei eine Variante besser von den Kunden angenommen wird als die anderen (also im Beispiel, dass danach mehr aus dieser Gruppe etwas kaufen). Gibt es einen Gewinner, wird diese Version als Standard verwendet. Gibt es keinen Gewinner, werden weitere Experimente definiert. Auch bei solch einem vermeintlichen Fehlschlag gibt es Lessons Learned: Entweder ist der Ansatzpunkt unerheblich oder eine bessere Variante wurde noch nicht gefunden.

Erfolg oder Misserfolg bestimmbar machen

Für jedes Experiment legt die Arbeitsgruppe gemeinsam mit den Beteiligten aus der jeweiligen Abteilung vor Beginn des Experiments Messkriterien fest: Welche Ausprägung wird erwartet? Welche Werte werden als Erfolg betrachtet und welche sind gerade noch akzeptabel? Durch die Festlegung im Vorfeld wird erreicht, dass

1. ein sehr viel genaueres Bild davon entsteht, was konkret erwartet wird,
2. durch die Konkretisierung ein Abgleich unterschiedlicher unausgesprochener Vorstellungen erfolgt,
3. später die beobachteten Veränderungen nicht je nach Interesse interpretiert werden.

Die Messkriterien müssen zum Thema und den Ausgangsbeobachtungen passen. Anders gesagt, die Kriterien müssen die Frage beantworten helfen, ob ein alternatives Vorgehen die Ausgangsprobleme vermeidet oder reduziert. Weiterhin sind Kriterien sinnvoll, die zu erkennen helfen, ob an anderer Stelle Probleme entstehen.

Beispiel

Ein Beispiel für Messkriterien eines solchen Experiments ist die Veränderung des beschriebenen Project Pitch Prozesses. Messkriterien für den Erfolg einer Veränderung waren hier:

- die Zeit von der Aufnahme einer Initiative bis zum Abschluss der Umsetzung
- durchschnittliche Größe von Initiativen (wenn die Vermutung besteht, dass durch das bisherige Vorgehen Projekte unnötig groß werden und ein besseres Verfahren diesen Effekt vermeidet).

Im Beispiel wurden zunächst nur die Kriterien definiert, nach denen eine Veränderung als Erfolg galt und nicht das Veränderungsexperiment selbst.

Löst die Veränderung das Problem?

Bei der Konstruktion des Experiments wird die Kenntnis der Kriterien helfen, zu beurteilen, ob eine diskutierte Veränderung solche Wirkung grundsätzlich bewirken kann oder ob es sich um eine Veränderung handelt, die nicht zum Problem passt. Ich habe einige Veränderungsideen erlebt, für die das zu lösende Problem nicht benannt werden konnte („Wir müssen agiler werden!“ ist keine gültige Problembeschreibung).

Ein Beispiel für Messkriterien für zu erwarteten, unerwünschten Nebeneffekten ist, dass ausprobiert werden soll, ob es zu schnellerer Umsetzung kommt, wenn Softwareentwickler und Tester in einem Team sind. Ein Messkriterium für den Erfolg ist dann die Umsetzungsgeschwindigkeit. Eine begründete Befürchtung ist in diesem Fall, dass durch solch eine Struktur die Qualität leidet, weil sich die Tester mit den Entwicklern verbrüdern oder dem Druck, vorschnell freizugeben, nicht standhalten. In diesem Fall bietet es sich an, ein Qualitätskriterium mit aufzunehmen, um die neue, temporäre Struktur auf das Eintreten dieser Befürchtung hin überprüfen zu können.

Die Anzahl der Messkriterien sollte überschaubar bleiben. Meinen Erfahrungen nach wird es bei mehr als fünf Kriterien zu unübersichtlich. Die (aktive) Beobachtung der Kriterien sollten Sie auch wieder einstellen, sobald sie ihren Zweck erfüllt hat. Andernfalls wird über die Hintertür mit den zu verwaltenden Metriken gleich wieder neuer Ballast aufgebaut.

Laufzeit festlegen

Auch der Zeithorizont eines Experiments hängt vom konkreten Thema ab. Der Zeitraum sollte so kurz wie möglich sein, aber lang genug, um der Veränderung die Gelegenheit zu geben, Resultate zu zeigen. In vielen Fällen reichen zwei bis vier Wochen aus. In anderen Situationen sind es drei Monate. Bei deutlich längeren Experimenten bietet es sich an, nach anderen Ansätzen zu suchen, um schneller Ergebnisse zu bekommen. Mit der Dauer steigt auch die Investition und damit der Druck, einen Erfolg vorweisen zu müssen. Auch wird es dann schwieriger, eine Änderung wieder rückgängig zu machen, sollte sie sich als nicht vorteilhaft herausstellen, da sich die Beteiligten inzwischen daran gewöhnt haben.

Den Zeitraum legen Sie wie die Messkriterien vor Beginn fest. Es ist wichtig, während des Verlaufs des Experiments die Auswirkungen zu beobachten. Bewahren Sie dabei Ruhe. Es kommt oft vor, dass eine Veränderung Zeit benötigt, um zu wirken. Am Anfang verschlechtern sich alle Werte, weil sich die neue Art und Weise erst einspielen muss. Eingreifen sollten Sie nur, wenn andernfalls erheblich irreversible Schäden entstehen oder wirklich sicher ist, dass es sich nicht um einen vorübergehenden Effekt handelt. Halten Sie ansonsten den eingangs definierten Zeitraum ein und bewerten dann die Ergebnisse. Das letzte Wort sollte der Initiator des Experiments haben.

Die Konzeption des Experiments samt Messkriterien, Zeitraum und weiterer Aspekte obliegt der Arbeitsgruppe zusammen mit allen direkt Beteiligten. Es ist unbedingt notwendig, dass die direkt beteiligten Personen

- die Notwendigkeit einer Veränderung nachvollziehen können,
- das zu testende Konzept für grundsätzlich umsetzbar halten,
- die Risiken als vertretbar bewerten,
- den Zeitraum als angemessen einschätzen,
- zustimmen, dass die Messkriterien und die angestrebten Werte plausibel sind.

Experimente umsetzen

Nachdem das Veränderungsexperiment konstruiert ist, geht es an die Umsetzung. In vielen Experimenten geht es darum, in einem begrenzten Rahmen eine andere Arbeitsweise auszuprobieren.

Bei IrgendeinShop.de identifizierte die Arbeitsgruppe den „Project Pitch“ als Impediment. Als Veränderungsexperiment wurde entwickelt, dass ein Team mit seinem Product Owner für drei Monate aus diesem Prozess ausgenommen wird und direkt mit seinen Stakeholdern arbeitet. Messkriterien waren:

- Anzahl der in dieser Zeit bearbeiteten Themen: mindestens gleich
- Geschwindigkeit von der Aufnahme eines Themas bis zum Abschluss der Umsetzung: 30% schneller
- Zufriedenheit der Stakeholder: Verbesserung der Bewertung um 2 Punkte auf einer Skala von 1 bis 10
- Zufriedenheit des Teams und Product Owners: Verbesserung der Bewertung um 2 Punkte auf einer Skala von 1 bis 10

Die Arbeitsgruppe musste entscheiden, welches Team das Experiment durchführt. Das Team und der Product Owner sollten den neuen Ansatz ausprobieren wollen. Auch einigte sich die Arbeitsgruppe darauf, dass die beteiligten Stakeholder keine wesentlichen Bedenken haben sollten oder diese im Vorfeld ausgeräumt werden. Ganz wichtig war, dass der Chief Product Officer als der gegenüber der Geschäftsführung Verantwortlicher gewillt war, das Risiko einzugehen, vorübergehend schlechtere Ergebnisse zu tragen und dem Team über den Zeitraum den Rücken frei zu halten. Der Chief Technology Officer war ebenfalls ein wichtiger Beteiligter, da er Vorgesetzter der Entwickler war und seine Zustimmung deshalb wichtig, damit die Teammitglieder sich darauf einließen.

Generell ist für solche Veränderungsexperimente wichtig, dass die Teilnehmer des Experiments offen für die Veränderung sind und der Veränderung eine Chance geben. Gleichzeitig ist eine Führungskraft erforderlich, die an das Experiment glaubt und es beschützt. Oft wird anfangs eine Verschlechterung zu beobachten sein, wenn die neue Arbeitsweise gelernt wird und es erfordert Mut, dem Druck stand zu halten und nicht abzubrechen.

Die Arbeitsgruppe festlegen

Es ist möglich, ein bestehendes Team zu nehmen oder ein Team speziell für das Experiment zusammenzustellen. Wichtig ist auch hier wieder, dass Mitglieder freiwillig mitmachen. Bei der Rekrutierung hilft, wenn der vorangegangene Prozess der Themenfindung und Konzeption öffentlich und transparent erfolgt. In den Veranstaltungen und bei den Gesprächen in den beteiligten Bereichen wird sichtbar, wer Interesse an dem Thema zeigt und auf eine Beteiligung angesprochen werden kann. Für die Zusammenstellung des Teams hat es sich bewährt, dass die letztendlich verantwortliche Führungskraft eine Person auswählt, der sie zutraut, das Experiment erfolgreich durchzuführen. Diese Person stellt dann ihr Team aus interessierten und geeigneten Kandidaten zusammen.

Finden sich keine Freiwilligen, ist dies ein wichtiges Signal für die Arbeitsgruppe, dass die angedachte Veränderung theoretisch sinnvoll ist, aktuell aber keine Unterstützung findet. In diesem Fall ist es besser, ein anderes Thema vom Impediment Backlog vorzuziehen und in der Zwischenzeit die Hintergründe für die Zurückhaltung genauer zu untersuchen. Womöglich gibt es später geänderte Voraussetzungen und dass Experiment kann angegangen werden. Es ist gut möglich, dass zuerst Vertrauen in die Idee von Experimenten fehlt und die Befürchtung dominiert, bei einem Scheitern des Experiments mit dem vermeintlichen Makel assoziiert zu werden. Mit der Durchführung kleinerer, gefahrloser Experimente und deren Handhabung können solche Befürchtungen zerstreut werden (oder bestätigt; aber dann ist der ganze Ansatz hinfällig).

Ein Experiment mit einem Team, das von dem Experiment selbst nicht überzeugt ist, wird kaum erfolgreich sein. Die Freiwilligkeit ist auch aus einem anderen Grund wichtig: Überzeugte Teilnehmer am Experiment werden zu internen Botschaftern und tragen die Idee weiter.

Vertrauen schaffen durch Öffentlichkeit

Die Experimente müssen ergebnisoffen und sichtbar für jeden umgesetzt werden. Nicht nur Erfolge, auch Rückschläge und Misserfolge sind transparent zu machen: Sie sollen Glaubwürdigkeit aufbauen.

Die Arbeitsgruppe sorgt für Transparenz, indem sie regelmäßig öffentlich über geplante, laufende und beendete Experimente berichtet – inklusive der Ergebnisse und dabei auch auf Probleme eingeht. Die Beteiligten an einem konkreten Experiment schaffen Transparenz indem sie – ebenfalls – regelmäßig von ihren Fortschritten und Erfahrungen berichten, guten wie schlechten. Interessierten Kollegen wird die Gelegenheit gegeben, die Arbeit darüber hinaus im Detail kennen zu lernen, z.B. mit Führungen oder tageweiser Hospitation.

Der Kommunikationskanal und die Häufigkeit der Kommunikation richtet sich einerseits nach den Gepflogenheiten des Unternehmens, andererseits nach den Fortschritten des Experiments. Es kann eine kurze Präsentation in einem Townhall-Meeting sein, ein E-Mail-Newsletter, Artikel im internen Blog oder etwas anderes. Wichtig ist, dass die Mitarbeiter die Kommunikation erhalten können und wissen, wie und bei wem sie weitere Details erfahren können.

Bei IrgendeinShop.de wurde der Bericht der Arbeitskommission in das monatliche Company Meeting integriert. Fortschritte zum konkreten Experiment wurden z.B. in die Sprint Review des Teams aufgenommen, zu der jeder Interessierte eingeladen war.

Experimente sind immer erfolgreich

Die Durchführung solcher Experimente ist in jedem Fall ein Gewinn:

- Durch die Durchführung wird ein Ansatz zur Veränderung aufgezeigt und gelernt, dass die Infragestellung von etablierten Gewohnheiten erlaubt und möglich ist.
- Der experimentelle Ansatz hilft, Ängste vor dem Ausprobieren von Alternativen abzubauen und nimmt auf diese Weise auch die Bedenken von Skeptikern ernst.
- Scheitern neue Ansätze, bietet sich eine Gelegenheit, zu lernen, konstruktiv mit dem Scheitern umzugehen. Dies ist bei Experimenten leichter, da hier ein Scheitern als Möglichkeit einkalkuliert ist (sonst wäre es kein Experiment).
- Die Bestätigung der veränderten Arbeitsweise durch ein Experiment liefert handfeste Argumente und erlaubt es, die neue Vorgehensweise, möglicherweise mit Zwischenschritten, weiter zu verbreiten. Anstatt Skeptiker überzeugen zu müssen, entwickelt sich eine positive Eigendynamik, weil jeder den erfolgreicheren Ansatz übernehmen möchte.

- Auch die Widerlegung einer Idee durch ein Experiment ermöglicht Lernen und liefert Material für weitere Experimente.

Den Prozess etablieren: Aufgabe der Arbeitsgruppe

Im Verlauf des Artikels ist mehrfach die Arbeitsgruppe erwähnt worden. Auch wenn dieses Gremium keine formalen Befugnisse hat (außer mit jedem reden zu dürfen), ist die Bedeutung enorm. Richtig aufgesetzt und eingeführt verfügt sie über Autorität, weil sie auf Willen der Geschäftsführung arbeitet und Authentizität, weil Mitglieder von den Mitarbeitern ausgewählt werden und die Arbeit öffentlich stattfindet. Diese Voraussetzungen erfüllt kaum ein anderes Gremium.

Für die Glaubwürdigkeit ist die Transparenz der Arbeit entscheidend und der kontinuierliche Abgleich mit der Geschäftsführung. Besser lehnt die Geschäftsführung begründet Lösungsvorschläge ab (u.a. weil sie als zu teuer oder riskant angesehen werden), als dass ein Thema hinter verschlossener Tür beerdigt wird.

In vielen Unternehmen ist die Arbeit mit dem Impediment Backlog und Experimenten ungewohnt. Sie muss gelernt werden und wird nicht ohne Rückschläge stattfinden. Der Einsatz eines erfahrenen und neutralen Moderators hilft, Diskussionen konstruktiv und im Sinne des Prozesses zu gestalten. Klug ist die Aufnahme von Skeptikern in die Arbeitsgruppe, um sie so in die Verantwortung zu nehmen. Allerdings müssen diese sich konstruktiv beteiligen.

Die Arbeitsgruppe sollte sowohl repräsentativ sein als auch klein genug, um effektiv arbeiten zu können. Fünf bis sieben Mitglieder sind ideal. Spätestens bei mehr als zwölf permanenten Mitgliedern wird die Koordination innerhalb der Gruppe zu einem Problem.

Für Beschlüsse der Arbeitsgruppe bietet sich der *Konsent* als Verfahren an. Diese Methode bedeutet im Kern, dass Beschlüsse angenommen werden, wenn niemand stichhaltige Gründe dagegen vorbringt. Für Einzelthemen kann die Entscheidung vom Gremium auf die fachlich qualifizierteste Person übertragen werden - unter der Bedingung, dass diese alle relevanten Personen identifiziert und konsultiert.

Happy End bei IrgendeinShop.de

Bei IrgendeinShop.de wurde, nachdem die Probleme überhandnahmen, genau solch eine Arbeitsgruppe eingesetzt - mit dem Namen „Excellence Board“.

Diesem Board gehörten ein Product Owner, zwei Software-Entwickler, eine Kollegin von HR, der CPO und CTO an. Bei den regulären Impediment Backlog Reviews war auch mindestens ein Mitglied der Geschäftsführung anwesend. Das Excellence Board fand sich während des ersten Workshops und wurde von der Geschäftsführung im monatlichen Company Meeting vorgestellt. Dabei wurde auch erläutert, dass die Aufgabe des Boards darin besteht, Impediments zu identifizieren und zu bearbeiten. Die Geschäftsführung äußerte die Erwartung an alle Mitarbeiter, das Exellence Board bei

seiner Arbeit zu unterstützen. Die Termine der öffentlichen Treffen wurden bekannt gegeben und, dass das Excellence Board als Teil des Company Meetings über Fortschritte berichten wird.

Selbstverständlich gab es Widerstände und Hindernisse:

- Die Geschäftsführung sah sich zuerst nicht imstande, bei den Sitzungen dabei zu sein.
- Teile des mittleren Managements fühlten sich unwohl dabei, „normale“ Mitarbeiter bei Diskussionen dabei zu haben.
- Die Teilnehmer brauchten Zeit, sich von gegenseitigen Schuldzuweisungen weg und hin zu gegenseitigem Verständnis zu bewegen.
- Die Mitarbeiter brauchten Zeit, sich ernstgenommen zu fühlen.
- Die Idee des Konsent brauchte Zeit, um akzeptiert zu werden.
- Es fiel schwer, Experimente zu starten, weil „dafür keine Zeit ist“ oder „es sowieso nicht funktioniert“
- Offene Reflexion zu Misserfolgen musste geübt werden.

Im Verlauf von ungefähr zwölf Monaten wurden dennoch erhebliche Fortschritte erzielt:

- Der bisherige Prozess des „Project Pitch“ wurde durch ein stark auf die Teams fokussiertes Verfahren ersetzt. Die Teams, geführt durch den Product Owner, erhoben neue Themen bei ihren Stakeholdern und erarbeiteten mit diesen zusammen Lösungen. Im Vergleich zu früher waren diese Lösungen oft überraschend einfach und entsprechend weniger aufwendig. Die Priorisierung der Produkte wurde vom Product Owner mit dem Team erarbeitet und durch ein Gremium aus CPO, CTO und Vertretern der Stakeholder bestätigt. In diesem Gremium wurden auch Fortschritte und Veränderungen besprochen.
- Zu Zeiten des Project Pitch bestand die Arbeit der Product Owner zum großen Teil darin, Initiativen abzuarbeiten, über die sie keine Kontrolle hatten. Durch die Abschaffung dieses Prozesses waren die Product Owner in der Lage, mit ihren Teams gemeinsam voraus zu planen und zwar nicht, wann welche Themen geplant sind, sondern auch, was überhaupt getan werden sollte. Durch diese inhaltliche Auseinandersetzung mit den Themen stieg auch die Identifikation der Teammitglieder mit den Themen.
- Jedes Team entwickelte für sich selbst ein Set mit Metriken, um die eigene Weiterentwicklung zu reflektieren. Diese Metriken waren Selbsteinschätzungen wie “Qualität der Anforderungen” oder “Zusammenarbeit im Team”, welche auf einer Skala von 1-10 eingeschätzt wurden, oder gemessene Werte wie die Dauer vom Start der Bearbeitung einer Anforderung bis zur Abnahme durch den Stakeholder. Maßnahmen zur Verbesserung und die Erfahrungen damit wurden geteilt. In regelmäßigen Abständen passten die Teams die Zusammensetzung der Metriken an, um den Fokus auf die jeweils aktuellen Herausforderungen zu legenanzupassen. Im Zuge dessen stieg die Auseinandersetzung der Teams mit ihrem eigenen Erfolg und es wurde zunehmend aktiv Verantwortung für Verbesserung übernommen statt auf Gründe außerhalb der eigenen Kontrolle zu verweisen und es dabei zu belassen.

- Das verstärkte Augenmerk darauf, warum Dinge getan werden, führte auch zu einem Lernen, wie Initiativen zielgerichtet umgesetzt werden. Die Teams lernten, angemessene Planungsverfahren einzusetzen, um ihren Fortschritt zu erfassen und Maßnahmen zu ergreifen.

Zurückblickend war für die Probleme bei IrgendeinShop.de ein Vorgehen ausschlaggebend, bei dem „quasi agile“ Teams über einen zentralen Prozess Aufgaben zugeteilt wurden. Die Idee, über diesen Prozess eine zentrale Priorisierung vorzunehmen, war durchaus vernünftig. Allerdings wurden dadurch Product Owner und Teams entmündigt und der direkte Austausch mit Stakeholdern reduziert. Anstatt gemeinsam inhaltliche Ziele zu verfolgen, wurde der Prozess zu einer Hülle und das Erreichen von "100% Aufgabe erledigt" wichtiger, als der Wert dieser Aufgaben.

Dadurch, dass der Project Pitch Prozess ersetzt wurde, wurde der bisherige „Top-Down“-Ansatz umgekehrt: Product Owner und Teams entwickeln mit ihren Stakeholdern eine Roadmap und diese wird vom Management überprüft und bei Bedarf angepasst. Durch diese Veränderung und die gewonnene Autonomie wurde der Weg bereitet für weitere, schrittweise Verbesserungen.

Ebenso wertvoll war die Entwicklung des Veränderungsprozesses selbst. Durch die Einbeziehung der Beteiligten aller Ebenen und die Würdigung ihrer Sichtweisen war es möglich, die jeweils anderen Sichtweisen immer besser zu verstehen und von einem „wir und die“ zu einer gemeinsamen Wahrnehmung von Verantwortung zu kommen.

Fazit

Häufig wird derzeit über die richtigen Strukturen und Prozesse – Projekt- oder Produktorientierung – diskutiert und nach einer vermeintlich perfekten Struktur gesucht. Bei dieser Diskussion wird übersehen, dass viele Probleme nichts mit der Struktur zu tun haben. Eine Veränderung der Struktur wird diese Effekte deshalb auch in den wenigsten Fällen beseitigen.

Vielversprechender ist es, deshalb bereichs- und hierarchieübergreifend konkrete Impediments (Hindernisse) zu identifizieren, die die Organisation davon abhalten, ihr Potential zu entfalten. Diese Impediments werden in einem Impediment Backlog gesammelt und priorisiert. Um diese Impediments zu beseitigen, werden Veränderungsexperimente konstruiert, um alternative Vorgehensweisen auszuprobieren.

Der Veränderungsprozess wird so gestaltet, dass die Arbeit transparent ist und Fortschritte geteilt werden. Dadurch wird eine offene Auseinandersetzung mit Problemen und die Übernahme von Verantwortung gefördert.

Gegenstand dieser Experimente kann auch sein, alternative Strukturen auszuprobieren - allerdings mit einem deutlich besseren Verständnis der Gründe und Zusammenhänge als Grundlage.

Gamification in Scrum

Motivieren Sie Ihr Team mit dem Blind Sprint Backlog

Dave Boddin
Softwareentwickler,
Projektleiter und Ausbilder

Dieser Beitrag richtet sich an alle, die ihr Scrum-Vorgehen über neue Methoden bereichern und ihr Team durch Abwechslung motivieren möchten. Im Folgenden stelle ich Ihnen die von mir konzipierte Methode "Blind Sprint Backlog" vor und berichte von meinen Erfahrungen damit.

In eine Sackgasse programmiert

Die Idee kam mir von ein paar Jahren, als ich eines meiner ersten Projekte leitete. Mein Team und ich entwickelten eine anspruchsvolle Software und verzettelten uns mit Technologien, die wir – rückblickend ist es deutlich – nicht verstanden. Der Worst Case trat ein: Wir hatten uns in eine Sackgasse programmiert. Verschärfend kam hinzu, dass ich weder einen Mentor besaß, noch auf zusätzliche Ressourcen hoffen durfte.

Wir änderten unsere technologische Strategie und wechselten den Kurs, damit verbunden war ein Wechsel beim technologischen Ansatz. Daraus folgte, dass wir bei der Entwicklung mehrere Schritte zurückgingen – doch die Uhr tickte unerbittlich weiter. Ich stand also vor der Herausforderung, die Mitarbeiter zu Mehrarbeit zu motivieren – ohne weitere Ressourcen oder Legitimationen.

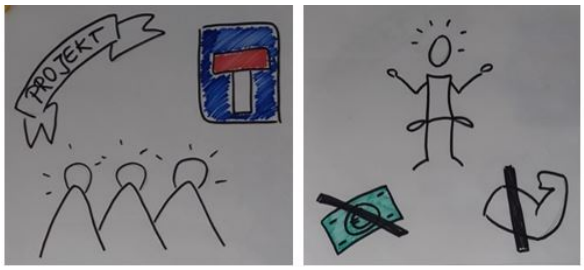


Bild 1: Das Projekt steckt in einer Sackgasse, das Team ist am Anschlag und der Projektleiter hat weder zusätzliche Mittel noch Unterstützung – jetzt hilft nur eine findige Idee

Intrinsisch motivieren – aber wie?

In meiner Not zog ich die Literatur zu Rate. Die Empfehlung: Motivieren Sie Ihre Mitarbeiter intrinsisch. Nur wie, das beantwortete mir kein Buch (das Projekt Magazin kannte ich damals noch nicht). Also konsultierte ich den Hype Cycle von Gartner. Diesen nutze ich bereits seit meinem Studium der Wirtschaftsinformatik, um mir schnell einen Überblick zu verschaffen, welche Trends und Technologien es aktuell gibt.

Der Hype Cylce inspirierte mich, Gamification zu nutzen, um unser Vorgehen in Scrum leicht abzuwandeln und neue Akzente zu setzen. Scrum sollte (wieder) Spaß machen: Das Team sollte lachen und am besten zeitweise vergessen, dass es hier um Arbeit ging.

Gamification

Der Begriff Gamification wurde erstmals 2002 verwendet und meint das Nutzen spieltypischer Elemente in einem spielfremden Zusammenhang. Im wirtschaftlichen Kontext wurden solche Elemente erstmals im Marketing verwendet. Ein Beispiel ist das Sammeln von Punkten im Einzelhandel oder Bonusmeilen bei Flugreisen für Prämien, es soll den Sammeltrieb des Kunden anregen und seine Bindung ans Unternehmen stärken.

Das Pull-Prinzip um den Zufall erweitert

Nach einigen Überlegungen, wie ich Scrum spielerischer gestalten könnte, kombinierte ich das klassische "TO-DO" des Sprint Backlogs mit dem Ziehen von Losen: Die Teammitglieder sollten Ihre Aufgaben zufällig ziehen, das sollte für Abwechslung und Unterhaltung sorgen.

Mein Team bestand ausschließlich aus ausgebildeten Software-Entwicklern, alle männlich und zwischen 20 und 30 Jahren alt. Wir alle befanden uns damals noch in der agilen Entdeckungsphase, hatten also noch nicht viel Erfahrung mit Scrum und Co.

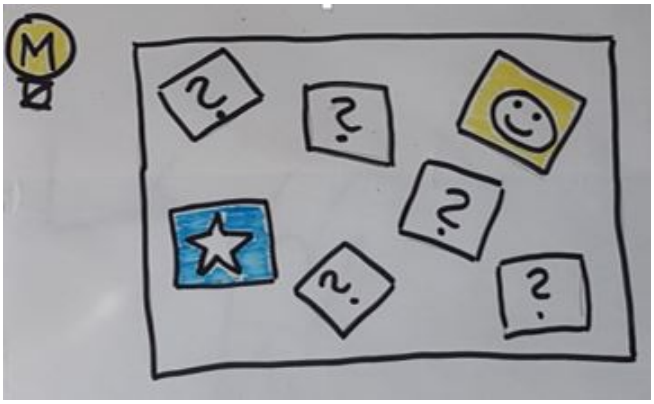


Bild 2: Die Methode "Blind Sprint Backlog" weckt spielerisch die Neugier der Teammitglieder und steigert so kurzfristig deren Bereitschaft für zusätzliche Leistung

Rosinenpickerei stoppen

Nebenbei wollte ich damit einen Trend stoppen, der die Motivation vieler Teammitglieder und den Zusammenhalt untereinander schwächte: Mir war aufgefallen, dass einige Teammitglieder Energie darauf verwendeten zu analysieren, welche Aufgaben in der TO-DO-Spalte einfacher waren und sich diese sicherten – ganz entgegen der klassischen Selbstbestimmung von agilen Teams mittels Pull-Prinzip. Ich wollte dieses Rosinenpicken verhindern und dies über ein spielerisches Vorgehen erreichen, um die Betroffenen nicht zu demotivieren, denn meiner Meinung nach taten sie es nicht aus Faulheit, sondern weil sie sich die Aufgaben nicht zutrauten.

Vorgehen: Neuer Sprint als Startpunkt

Als ein neuer Sprint anstand, probierte ich meine Idee aus. Ich erstellte freitags wie üblich Karteikarten mit den Tasks für die TO-DOs des Sprint Backlogs. Zusätzlich gestaltete ich einige zusätzliche Karten mit Witzen, Zitaten oder Gutscheinen, z.B. für eine Kaffeepause oder eine Runde Kicker. Ich vermischte die Aufgaben mit den Witzkarten und am Abend, nachdem alle

Teammitglieder gegangen waren, heftete ich die Karten umgedreht an das Sprint Board, damit niemand sehen konnte, was er zog.

Als ein neuer Sprint anstand, probierte ich meine Idee aus. Ich erstellte freitags wie üblich Karteikarten mit den Tasks für die TO-DOs des Sprint Backlogs. Zusätzlich gestaltete ich einige zusätzliche Karten mit Witzen, Zitaten oder Gutscheinen, z.B. für eine Kaffeepause oder eine Runde Kicker. Ich vermischte die Aufgaben mit den Witzkarten und am Abend, nachdem alle Teammitglieder gegangen waren, heftete ich die Karten umgedreht an das Sprint Board, damit niemand sehen konnte, was er zog.

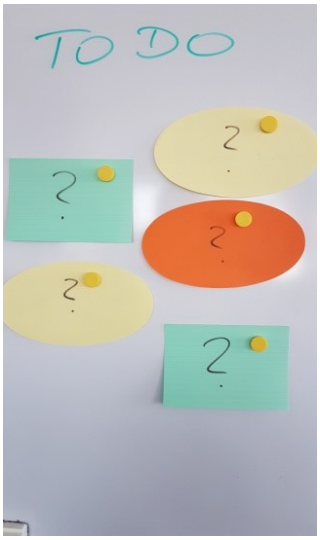


Bild 3: Zu Beginn des "Blind Sprints" bietet das Board den Teammitgliedern einen ungewohnten Anblick

Am Montagmorgen war ich als erster im Büro und überrasche das Team mit meiner Idee. Die Teammitglieder reagierten zunächst verhalten, mit einem für den Berliner Raum typischen, langgezogenen "Okayyy???". Rechnen Sie also mit anfänglicher Skepsis. Ich lächelte und erklärte das neue Vorgehen. Ich zog als erster eine Karte, um auch mich zum Betroffenen der Änderung zu machen.

Die Teammitglieder mit freien Ressourcen folgten meinem Beispiel. Fast alle waren gespannt und hatten viel Spaß, denn Witze oder Zitate muss der betreffende Mitarbeiter laut vorlesen, danach zieht er eine weitere Karte, bis er eine Aufgabe erwischt.

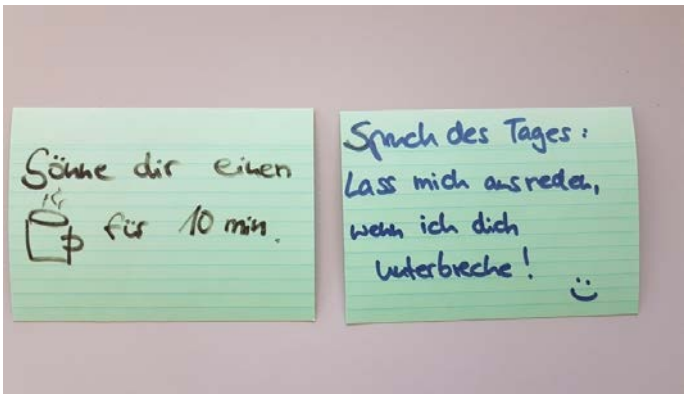


Bild 4: Aktionskarten (links) und Witze (rechts) motivieren die Teammitglieder und lockeren die Stimmung auf

Motivation und Teamgeist wachsen

Ich hatte das Gefühl, dass jeder sich auf das Ziehen einer neuen Karte freute, aufkeimende Neugierde lag in der Luft. In dieser positiven Atmosphäre arbeitete es sich besser und alle kamen mir engagierter vor. Das Ziehen der Aufgaben wurde schnell zu einer teamöffentlichen Angelegenheit: "Ich muss ziehen", hieß es dann und alle kamen zusammen. Zog jemand eine schwerere Aufgabe, kam gerne auch mal ein "Oh neee!". Trotzdem lachten alle, einschließlich des "Pechvogels" und ein erfrischender Zusammenhalt war zu spüren.

Zufällig bekam ich mit, wie ein Mitarbeiter nach dem Ziehen einer schweren Aufgabe einem Kollegen gegenüber seine Hoffnung äußerte, beim nächsten Mal "etwas Cooles zu ziehen." Abwechslung und Unvorhersehbarkeit motivierten nachhaltig und erzeugten Zuversicht. So mancher zog eine Aufgabe, um die er gewöhnlich einen großen Bogen gemacht hatte – und fast alle meisterten sie dennoch, u.a. indem sie sich Unterstützung holten, was zuvor eher die Ausnahme war. Dies ließ das Team stärker zusammenwachsen. Viele blieben abends auch länger als üblich.

Das Ende der Rosinenpickerei

Auch dass niemand sich mehr absichtlich eine einfache Aufgabe herauspicken konnte, stärkte den Teamgeist. Mehrere Teammitglieder meldeten mir das von sich aus zurück. So löste ich eine Konfliktsituation, ohne persönlich an die Verursacher herantreten zu müssen. Besonders freute ich mich, dass dieser neue Teamgeist über das zweiwöchige Experiment hinaus Bestand hatte. Er äußerte sich u.a. dadurch, dass Mitarbeiter weiterhin Angebote zur Unterstützung machten und dass niemand mehr Rosinenpickerei betrieb.

Bild 5: Eine Karte geht noch – während des Blind Sprints unterstützen sich die Teammitglieder gegenseitig bei kniffligen Aufgaben, viele bleiben länger als üblich

Tipps und Empfehlungen

Zum Verhältnis von "Ereigniskarten" und Aufgaben muss jeder ein gutes Maß finden, das zur Kultur seines Unternehmens passt. Das Unternehmen, für welches ich tätig bin, strukturiert sich aktuell stark um, so besteht neben dem erhöhten Workload auch viel Unsicherheit. Hier empfehle ich einen hohen Anteil an "Ereigniskarten," z.B. eine auf drei Aufgaben.

! Die Ereigniskarten sollen aufheitern, nicht albern oder letztendlich sogar störend wirken. Witze sollten daher politisch korrekt und genderneutral sein.

Lassen Sie dem Zufall freie Hand

Ein Teammitglied zog fast nur schwere Aufgaben. Ich überlegte einzugreifen, indem ich ihn bei einer Aufgabe unterstützte. Jedoch entschied ich mich dagegen, da dies den Zufallseffekt abgeschwächt hätte, der eines der Hauptelemente dieses Vorgehens darstellt. Deshalb bleibe ich hier konsequent.

Sparsam einsetzen

! Bitte gehen Sie mit dieser oder ähnlichen Ideen nicht inflationär um. Sie sollen eine Abwechslung vom üblichen Vorgehen darstellen, dieses aber nicht dauerhaft ersetzen. Früher oder später werden die Mitarbeiter Forderungen wie Entschädigungen äußern, wenn sie immer länger arbeiten.

Damit sich das Vorgehen nicht erschöpft, nutze ich es nie für zwei aufeinanderfolgende Sprints, sondern warte und versuche zu erkennen, wann das Team eine Abwechslung oder eine Extra-Motivation braucht. Wenn Sie auf Ihr Team eingehen und diesem zuhören, werden Sie z.B. spüren, wann es Zeit für eine Abwechslung ist. In einem Projekt habe ich immer nach vier Wochen Pause einen zweiwöchigen Blind-Sprint durchgeführt, in anderen Projekten kam ich ganz ohne aus.

Mein Fazit

Wir beendeten das Projekt nur leicht verspätet und hatten damit mehr erreicht, als aufgrund der

beschränkten Möglichkeiten realistisch erschien. Ich schätze, dass Neugierde, Spaß und neuentfachte Begeisterung eine Dynamik auslösten, dank derer wir die Aufgaben-Durchlaufzeiten verkürzen konnten. Ich verbuchte dies als persönlichen Erfolg.

Diese Erfahrung bestärke mich in der Überzeugung, dass ich als Projektleiter mit Kreativität und Mut zu Neuem viel erreichen kann. Es bestehen noch viele ungenutzte, ressourcengünstige Möglichkeiten, um das tägliche Arbeiten interessanter zu gestalten und wenn man die anfängliche Skepsis aushält, kann man als Projektleiter so das Arbeitsklima positiv beeinflussen.

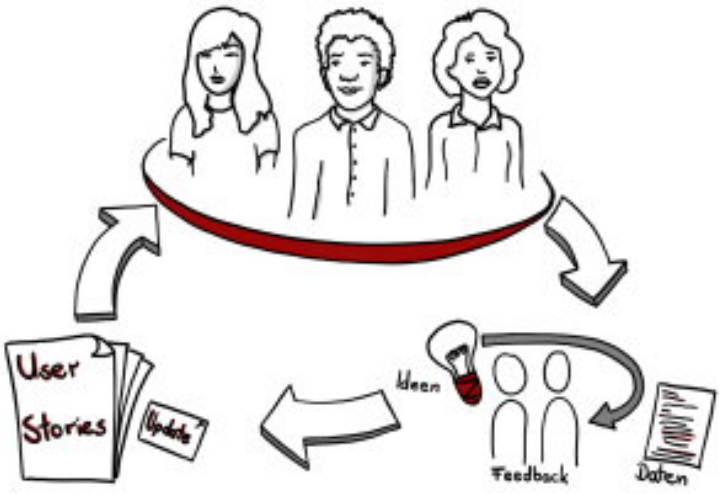
Da mein Team nur aus Männern bestand, kann ich nicht sagen, wie das Vorgehen auf Frauen wirkt, ich würde es aber auch in einem gemischten Team ausprobieren, denn Humor und Abwechslung sprechen meiner Einschätzung nach jeden an, außerdem wirkt die dadurch entfachte Begeisterung ansteckend, was sich bei meinem Team darin äußerte, dass selbst anfängliche Skeptiker nach einigen Tagen Spaß mit dem Blind Sprint Backlog hatten.

Das Vorgehen hat sich im Unternehmen mittlerweile ausgebreitet. Weil meine Teammitglieder in ihrer Begeisterung sogar in der Kantine darüber sprachen, wurden einige andere Projektleiter neugierig und erkundigten sich danach bei mir. Mittlerweile nutzen mehrere Kollegen das Vorgehen von Zeit zu Zeit ebenfalls.

Literatur

- Heckhausen, Jutta; Heckhausen, Heinz (Hrsg.): Motivation und Handeln, Springer 2010
- Sailer, Michael: Die Wirkung von Gamification auf Motivation und Leistung. Empirische Studien im Kontext manueller Arbeitsprozesse, Springer Fachmedien, Wiesbaden 2016

User Storys erstellen



Kurzdefinition User Storys sind kurze, einfach gehaltene Beschreibungen einer Funktionalität oder eines Gegenstands aus der Perspektive der Anwender oder Kunden. Die Beschreibung erfolgt zumeist in einem einfachen Schema:

Englisch: As a <type of user>, I want <some goal> so that <some reason>.

Deutsch: Als <Rolle der beschreibenden Person>, möchte ich <Funktion/Gegenstand>, damit ich <Nutzen>.

Beispiel: Als Anwender möchte ich, dass beim Anklicken der "Schnelldruck"-Schaltfläche die aktuelle Dokumentauswahl an den Standarddrucker gesendet wird, damit ich Zeit spare.

Im Gegensatz zum herkömmlichen Anforderungsmanagement werden User Storys lösungsoffen formuliert. Diese Form ermöglicht Raum für spätere Änderungen und erleichtert Detaillierungen, die mitunter zu Beginn eines Projekts nicht möglich sind. User Storys werden während der gesamten Laufzeit eines agilen Projektes erstellt und dienen primär als Input für das Product Backlog.

Einsatzmöglichkeiten

User Storys sind ein verbreitetes Konzept zur Beschreibung von Anforderungen in agilen Projekten. Sie dienen als Gesprächs- und Diskussionsgrundlage zur Vereinbarung des Leistungsumfangs und werden im Laufe des Projekts im Team weiterentwickelt.

Vorteile

- User Storys fungieren als Kommunikationsmittel zwischen Product Owner und Development Team und ermöglichen es, die Anforderungen sowohl aus Business- als auch aus technischer Sicht zu verstehen.
- User Storys reduzieren die Unsicherheit über die zu implementierende Funktionalität.
- Übersetzungsfehler von natürlicher Sprache (des Kunden) in abstrakte Modelle (z.B. Use-Case-Diagramme) werden eliminiert.
- Auf Grund Ihres hohen Abstraktionsniveaus verringern User Storys den Druck, unsichere Detailaussagen zu einem frühen Zeitpunkt treffen zu müssen.
- Die Verwendung von User Storys entlastet die Formulierung der Anforderungen von technischen Detailfragen. Deren Lösung obliegt vollständig dem Entwicklerteam oder anderen involvierten Fachabteilungen.

Grenzen, Risiken, Nachteile

- Wenn keine Abstimmung mit dem Development Team möglich ist, sollten User Storys nicht verwendet werden. In diesem Fall empfiehlt es sich, auf herkömmliche Techniken des Requirements Engineerings wie z.B. Use Cases auszuweichen.
- User Storys sind kein Ersatz für detaillierte Spezifikationen!
- Mangelnde Kommunikation bzw. Abstimmung hinsichtlich des Inhalts der User Story im Team kann zu unzureichenden Ergebnissen der Implementierung führen.

Ergebnisse

- Ggf. Epics als übergeordnete, grobe User Storys
- Vollständig beschriebene User Storys als Input für das Product Backlog
- Konsolidierte Akzeptanzkriterien für die User Storys
- Optional: weitere ergänzende Informationen für die weitere Arbeit mit den User Storys, z.B. Fact-Sheets zu bestehenden Software-Komponenten

Voraussetzungen

- Die Kunden oder Benutzer der Anforderung / Funktionalität müssen bekannt sein, da User Storys aus der Kundensicht erstellt werden.
- Aktive Beteiligung der Kunden bzw. Benutzer an der Erstellung der User Storys
- Das Team muss bereit sein, aktiv und engagiert die User Storys zu diskutieren.

- Zeit und Wille, die User Storys fortlaufend einem kritischen Blick zu unterziehen

Qualifizierung

- Die Beteiligten müssen mit der Methodik vertraut sein.
- Methoden-Neulinge sollten sich den grundsätzlichen Aufbau von User Storys vor Beginn ansehen und das Ausfüllen mit fiktiven Beispielen trainieren.
- Der Besuch eines Storytelling-Workshops bzw. die inhaltliche Auseinandersetzung mit dem Thema Storytelling ist empfehlenswert.

Benötigte Informationen

- Erwartete Leistung einer Funktion / eines Produkts aus Kundensicht (anfordernde Rolle).
- Akzeptanzkriterien, die zur Zielführung beitragen (werden im späteren Verlauf detailliert).

Benötigte Hilfsmittel

Ein digitales (z.B. Textverarbeitung, spezielle Software) oder analoges Tool (z.B. Moderationskarten) zur Erfassung der User Storys

Durchführung

"A User Story is a brief statement of intent that describes something the system needs to do for the user." (Dean Leffingwell: A User Story Primer, 2009)

Die folgenden Schritte beschreiben die Erstellung einer User Story. Die Erstellung kann durch den Kunden, den Product Owner, ein Teammitglied oder einen anderen Stakeholder erfolgen. Grundsätzlich kann jede Rolle eine User Story erstellen.

Wenn eine Person außerhalb Ihres Teams (z.B. der Kunde) die User Storys anfertigt, achten

Sie darauf, diese Person auch im weiteren Projektverlauf eng einzubeziehen. Dies ist notwendig, um adäquat auf Änderungen bei den entsprechenden User Storys reagieren zu können.

Um leichter in die Kundenrolle zu schlüpfen, sollten User Storys mit Hilfe von sog. "Personas", also fiktiven Charakteren der Zielgruppe erstellt werden (s. Schritt 2).

Schritt 1: Analyse und Recherche

Wenn Sie die User Storys erstmalig anfertigen, ist es sinnvoll, die betroffenen Rollen ausgiebig zu analysieren und rollenspezifische Daten zu erheben. Dies kann z.B. mittels Interviews oder durch eine Webrecherche erfolgen. Im Gespräch mit Stakeholdern sind folgende Informationen für die Ermittlung von Zielen relevant:

- Arbeitsweise der Rolle (z.B. elektronisch, analog, eingesetzte Tools)
- Haltung des Stakeholders gegenüber der geplanten Änderung (z.B. Initiator der Weiterentwicklung, Festhalten an bestehender Lösung usw.)
- Unzufriedenheit mit bestehender Lösung (z.B. lange Reaktionszeit, keine Vorauswahl bei Eingaben usw.)
- Wünsche und Anforderungen an neue Lösung (z.B. Möglichkeit der Spracheingabe, automatische Fehlererkennung usw.)

Im Rahmen einer Webrecherche tragen Sie bei Bedarf weitere Informationen, z.B. Studien, Marktanalysen oder Dokumentationen zu bestehenden Komponenten zusammen.

Schritt 2: Erstellen Sie Personas!

Die Anfertigung von Personas dient der Vertiefung des Kundenverständnisses. Erstellen Sie Personas, die von der Zielsetzung Ihres Projekts betroffen sind. Personas haben in der Regel einen Namen, ein Bild, Charaktereigenschaften, Verhaltensweisen und ein Ziel. Das Ziel ist der Vorteil, den die Person mit Hilfe des Produkts erzielen will oder das Problem, welches das Produkt lösen soll. Weitere Elemente der Beschreibung einer Persona können z.B. sein:

- Beruf, z.B.: "Informatiker", "Abteilungsleiter"
- Alter, z.B.: "40 Jahre", "Berufsanfänger", "kurz vor Renteneintritt"
- Bildungsstand, z.B. "Studium", "kaufmännische Lehre"
- Abteilung, z.B. "zentrale IT"
- Business-Ziele, z.B. "Kostenreduktion"
- Einstellung, z.B. "offen für neue Wege"

- Mögliche Anwendungsfälle für die zu entwickelnde Lösung, z.B. "Fokus Automatisierung von Regeltätigkeiten"
- Kontext der Benutzung, z.B. "im Rahmen des monatlichen Geschäftsberichts"
- Aufgaben, die für den Abschluss relevanter Szenarien notwendig sind, z.B. "grafische Aufbereitung in Form von Entwicklungstrends"

Kürzen oder erweitern Sie diese Liste entsprechend Ihrer Anforderung bzw. Einschätzung.

Je nach Projektumfang wird eine hohe Anzahl an Personas notwendig sein. Rechnen Sie daher mit ca. 30 Minuten Aufwand für die Erstellung einer Persona.

Schritt 3: Starten Sie mit Epics!

Gerade zu Beginn eines Projekts ist es schwierig, User Storys mit einer entsprechenden Detailtiefe zu erstellen, wenn noch keine dokumentierten Anforderungen vorliegen. Epics sind sehr allgemeine User Storys, die es erlauben, den groben Umfang einer Kundenanforderung festzuhalten. Man kann Epics mit einer Überschrift oder einem Platzhalter in einem Text verstehen, oder auch als Container für User Storys zu einem bestimmten Themenbereich, die im Laufe des Projekts mit Details befüllt werden. Eine Möglichkeit, Epics zu visualisieren, sind sog. "Storyboards" (auch Drehbuch genannt). Storyboards sind vergleichbar mit Szenarios, sie illustrieren die Interaktion, die benötigt wird, um ein Ziel zu erreichen. Im Unterschied zu einer Aufzählungsliste visualisiert ein Storyboard diese Interaktion in einer Art Comic-Darstellung (vgl. Bild 1).

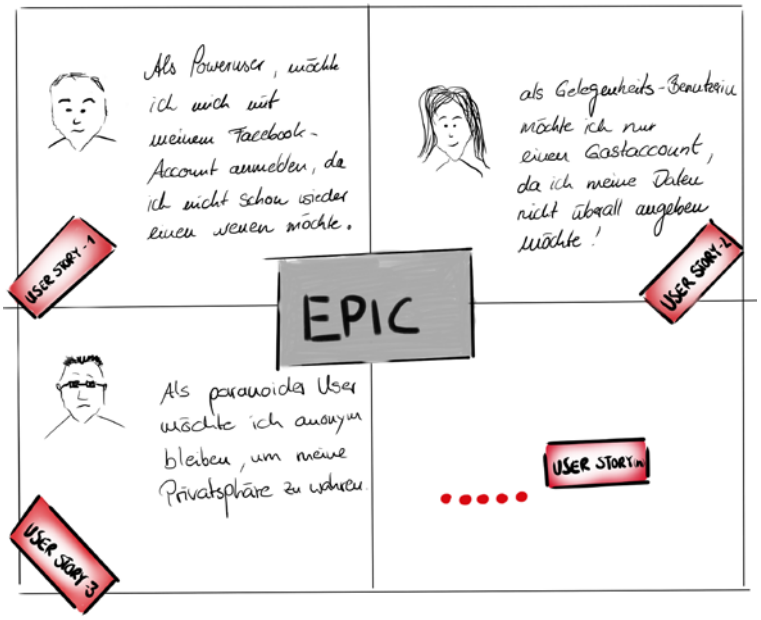


Bild 1: Visualisierung des Epics "Login-Varianten und Account-Typen" in Form eines Storyboards.

Der Aufwand zur Erstellung der Epics ist stark von der Vision des Projekts (Umfang, Komplexität) abhängig.

Schritt 4: Verfeinern Sie Ihre Epics zu User Storys!

Die User Story an sich ist eine implementierbare kleine Geschichte, die innerhalb eines Sprints abgearbeitet werden kann. Jedes Teammitglied muss ein klares Verständnis von der User Story besitzen, die Story muss klar, umsetzbar und testfähig sein.

Achten Sie hierbei darauf, dass die User Storys einfach und verständlich formuliert sind. Fokussieren Sie sich auf das Wesentliche und verzichten Sie auf Details. Lassen Sie Unwichtiges weg und formulieren Sie die User Storys immer aktiv. Sie können hierfür das populäre Template von Rachel Davies (Davies, Rachel: " Non-Functional Requirements: Do User Stories Really Help?", 2010, http://www.methodsandtools.com/archive/archive.php?id=113) verwenden und die Benutzerrolle durch Ihre jeweilige Persona nach dem in Bild 2 dargestellten ersetzen.

Als < persona>,

möchte ich <was?>

damit ich < warum?>

Bild 2: Schema für User Story.

Dieses Template kann hilfreich sein, ist allerdings kein Muss. Finden Sie mit Ihrem Team gemeinsam heraus, welche Formulierungen am besten zu Ihnen und Ihrem Umfeld passen.

Damit User Storys ihren Zweck im Projekt erfüllen können, muss überprüfbar sein, ob das zu entwickelnde Produkt sie tatsächlich erfüllt, d.h. sie müssen testbar sein. Dies bedeutet, dass jede User Story Akzeptanzkriterien benötigt. Dies hat den Vorteil, dass die Anforderungen gemäß den Wünschen und Vorgaben der Kunden umgesetzt werden. In der Regel empfiehlt es sich zwischen drei und fünf Kriterien pro User Story aufzustellen.

Szenario <Titel>

Gegeben <Kontext>

Und < noch mehr Kontext>

Wenn < Event>

Dann < Ergebnis>

Und <weiteres Ergebnis>

Bild 3: Schema für Epic.

Auch für die Erstellung von Akzeptanzkriterien können Sie ein Template verwenden, die sog. "Acceptance Story", die wie in Bild 3 gezeigt aufgebaut ist.

Bild 4 zeigt ein Beispiel für ein ausformuliertes Epic.

Neben den inhaltlichen Qualitätskriterien müssen User Storys noch eine weitere Eigenschaft besitzen, um für die Projektarbeit (z.B. für die Planung der Sprints) tauglich zu sein: Sie müssen so beschrieben sein, dass der Arbeitsaufwand für ihre Umsetzung zumindest relativ schätzbar ist.

Szenario: Kontostand ist negativ

Gegeben: Der Kontostand ist unter 0

Und es ist kein Dispo eingerichtet

Wenn der Kunde versucht, Geld abzuheben

Wird die Bank das Ablehnen

Und die Bank wird per E-Mail über den Vorgang informiert.

Bild 4: Beispiel für ein Epic.

Wenn ausreichend Zeit zur Verfügung steht und das Team die notwendige Kompetenz hat, kann es anschließend an die Erstellung der User Storys mit geeigneten Methoden eine Aufwandsschätzung durchführen, z.B. mit Planning Poker oder dem Team Estimation Game.

Am Ende dieses Schritts haben Sie die Anforderungen an das Projekt in Form von User Storys formuliert. Jede User Story enthält dabei folgende Informationen:

- "Story Name": eindeutiger und griffiger Titel der Story
- Zuordnungen zu Funktionalität und Anforderer
- "Value Statement": der eigentliche Inhalt der User Story gemäß dem oben dargestellten Schema
- Akzeptanzkriterien, die eine definierte Qualität sicherstellen
- Informationsanhänge, die die weitere Bearbeitung erleichtern (optional)
- Aufwandsschätzung in Story Points (optional)
- Zuordnung zu einem Epic (optional)

Schritt 5: Prüfen Sie die User Storys!

Ein häufiges Problem bei User Storys ist, dass Kundenfokus und Business-Wert bei der Erstellung verloren gehen. So kommt es nicht selten vor, dass User Storys für den Product Owner oder das Team erstellt werden, um deren Arbeit zu dokumentieren.

Die Überprüfung erfolgt im Team. Achten Sie beim gemeinsamen Review darauf, dass der Kundenfokus klar erkennbar ist. Gleichermaßen überprüfen Sie, ob die User Story zur Vision beiträgt, also dem allgemeinen Projektauftrag. Reine "Nice-to-have"-User Storys sollten nicht umgesetzt werden.

Eine mögliche Checkliste für das Review von User Storys sind die "INVEST"- Eigenschaften nach Bill Wake (Wake, Bill: INVEST in Good Stories, and SMART Tasks, 2003, http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/):

- "Independent" – Überlappt nicht mit anderen User Storys und sollte auch nicht von ihnen abhängig sein.
- "Negotiable ... and Negotiated" – Sie soll in der Verhandlung zwischen Kunde und Ersteller entstehen. Ihre Details sollen ebenfalls in der Zusammenarbeit der Stakeholder ausgearbeitet werden.
- "Valuable" – Aus Sicht des Kunden geschrieben und einen klaren Kundenwert enthalten.
- "Estimable" – Ihr Aufwand kann geschätzt werden.
- "Small" – Sie sind kurz und prägnant. Die Anforderung kann in wenigen Tagen abgearbeitet werden.
- "Testable" – Sie muss über Akzeptanzkriterien verfügen und testbar sein.

Investieren Sie nicht mehr als 15 Minuten für den Review einer User Story. Überschreitet die Diskussion die Zeit, sollten Sie die User Story erneut mit den vorhandenen Informationen abgleichen und/oder mit Ihren betroffenen Stakeholdern in die Diskussion gehen.

Wiederholen Sie den Review für bestehende und neue User Storys in regelmäßigen Abständen

(z.B. eine Woche) in sog. "Estimation Meetings". Dies führt zu einem nachhaltigen Verständnis im Team und einer qualitativ hochwertigen Einschätzung während des Projektverlaufs.

Schritt 6: Priorisieren Sie die User Storys!

User Storys sollten nach Mehrwert priorisiert werden. Diese Priorisierung erfolgt durch den Kunden oder den Product Owner in Absprache mit dem Kunden.

Ergänzende / ähnliche Methoden

- Planning Poker – Aufwandsschätzung
- Team Estimation Game – Aufwandsschätzung
- Sprint Review – Präsentation der Arbeitsergebnisse (Increments)
- Sprint Retrospektive – Bewertung des vergangenen Sprints (Maßnahmen zur Optimierung)
- Sprint Planning – Einteilung und Übernahme in den Bearbeitungsstatus
- World Café – Methode zur Informations- und Ideensammlung (Input für User Storys)
- Kanban Light – Methode zur Arbeitsorganisation mit User Storys

Praxistipps

- User Storys werden zumeist im Rahmen agiler Vorgehen erstellt. In Scrum beschreibt die Summe der User Storys die vollständige Funktionalität des Projektziels.
- Verwenden Sie Papier. Schreiben Sie die User Storys auf physische Moderationskarten, da dies die Zusammenarbeit im Team fördert. Zudem sind sie sind leichter verschiebbar oder austauschbar und jeder kann mit dieser Technik sofort arbeiten.
- Machen Sie Ihre User Storys sichtbar. Hängen Sie die User Storys z.B. an eine große Wand oder an eine Tafel. Dies schafft Transparenz und ermöglicht einen leichten und einfachen Zugang für alle Beteiligten.
- Den richtigen Detaillierungsgrad einer User-Story zu treffen, ist gerade für unerfahrene Scrum-Teams sehr schwer. Eine Hilfestellung dafür ist das Grundprinzip: "Kann eine User-Story in weitere User-Storys unterteilt werden, ist sie wahrscheinlich noch zu umfangreich oder zu allgemein beschrieben. Nehmen Sie eine Unterteilung vor."

Varianten

Kombination mit Sketches und Story Maps

Sie können User Storys mit anderen Methoden kombinieren. So können Sie z.B. Sketches und

Story Maps einsetzen, um die sog. Customer Journey ganzheitlich abzubilden. Bild 5 zeigt eine einfache Customer Journey, die so entstanden ist: Sie zeigt, wie sich ein Mitarbeiter sicher mit einem Account von allen Geräten in sein Firmennetz einwählen kann.

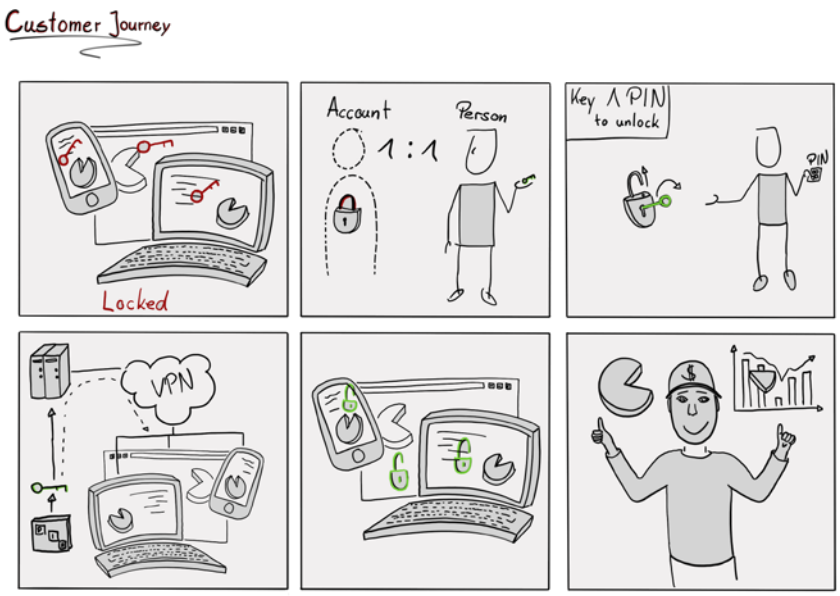


Bild 5: Customer Journey, die einzelne User Storys zu einem Ablauf verbindet.

Herkunft

Das Konzept der User Story wurde unter anderem von Dr. Ivar Jacobson (Erfinder der Use Cases) erstellt. Ivar Jacobsen erweiterte das System im späteren Verlauf unter dem Namen "Use Case 2.0" (Jacobson, Ivar; Spence Ian; Bittner, Kurt: USE-CASE 2.0. The Guide to Succeeding with Use Cases, 2011, https://www.ivarjacobson.com/publications/white-papers/use-case-ebook) als Ergänzung im Rahmen des Requirements Engineerings.

Auch Ron Jeffreys gilt als ein Erfinder der User Story, da er diese bereits 2001 in einem Blogbeitrag diskutiert (Jeffreys, Ron: Essential XP: Card, Conversation, Confirmation, 2001, http://ronjeffries.com/xprog/articles/expcardconversationconfirmation/).

Autoren

Daniel Reinold und Christian Botta

Erstellt am: 03.07.2016

Anforderungsmanagement in IT-Projekten

So vermeiden Sie Stolpersteine bei User Stories

Steffen Thols
Senior Consultant, codecentric AG, Scrum Master

Mit der Verbreitung agiler Vorgehensweisen in IT-Projekten stellt sich immer häufiger auch die Frage, auf welche Weise Anforderungen am besten beschrieben werden sollten. Vor allem die sog. "User Stories", in welchen Anforderungen aus Sicht des Nutzers beschrieben werden, erfreuen sich dabei steigender Beliebtheit: Sie haben einen klaren, einfachen Aufbau und konzentrieren sich auf die wesentlichen Elemente einer Anforderung.

Eine User Story darf inhaltlich nur so komplex sein, dass sie innerhalb eines Sprints umgesetzt werden kann und am Ende des Sprints dem Kunden – oder dem Product Owner, der die Kundensicht einnimmt – ein abnahmefähiges Teilprodukt gezeigt werden kann.

Im Artikel "Agile Softwareentwicklung mit Scrum und User Stories" (projektmagazin 02/2010) haben Sie erfahren, wie User Stories grundsätzlich aufgebaut sind und wie sich die Anforderungen eines Scrum-Projekts durch User Stories beschreiben, priorisieren und abarbeiten lassen.

Ergänzend hierzu sollten Sie bei der Konzeption von User Stories einiges beachten. Auch müssen die entsprechenden Rahmenbedingungen im Unternehmen geschaffen werden, damit die Software-Entwicklung mit dieser Vorgehensweise erfolgreich ist.

In diesem Artikel erfahren Sie,

- was eine User Story ist,
- warum es sinnvoll ist, mit User Stories zu arbeiten,
- welche formalen und strukturellen "Stolpersteine" es bei der Konzeption von User Stories gibt und
- wie Sie diese umgehen können.

Was ist eine User Story?

Ende der 90er Jahre arbeiteten die Software-Entwickler Ron Jeffries und Kent Beck in einem Projekt zur Entwicklung einer Gehaltsabrechnungssoftware bei Chrysler. Für die Softwareentwicklung nutzten sie die Methode XP (Extreme Programming) und verwendeten erstmals für

die Beschreibung der Anforderungen User Stories, die Funktionalitäten der Software aus Anwendersicht und in einfachen Worten darstellten. So stellten sie sicher, dass die entwickelte Software auch wirklich den Anforderungen der Anwender entsprach.

Die Realisierung einer User Story besteht aus folgenden drei Elementen:

1. Die Anforderung wird in einfacher Form auf einer Indexkarte festgehalten (Card):

Als <Nutzerrolle> will ich <das Ziel>, sodass <Grund für das Ziel>.

Beispiel:

"Als Kunde eines Online-Shops kann ich meine Bankverbindung in meinem Profil abspeichern, sodass ich diese bei einer weiteren Bestellung nicht erneut eingeben muss."

2. Das Entwicklungsteam stimmt sich direkt mit dem Anforderungssteller – oder dem Vertreter des Anforderungsstellers, d.h. dem Product Owner – ab. Die Details einer User Story werden in einer gemeinsamen Diskussion kurz vor Beginn der nächsten Iteration, d.h. des nächsten Entwicklungsabschnitts, geklärt (Conversation).
3. Jede User Story beinhaltet Testfälle, die sog. "Akzeptanzkriterien", die genau beschreiben, wie die Funktionalität nach der Umsetzung der User Story durch die Entwickler funktionieren soll (Confirmation).

Beispiel:

"Wenn die Bankleitzahl weniger als neun Ziffern enthält, führende Nullen oder Buchstaben eingegeben werden, gibt das System eine Fehlermeldung aus.

Die Kontonummer kann bis zu zehn Stellen umfassen und darf nur Ziffern beinhalten; führende Nullen sind erlaubt …"

Warum User Stories sinnvoll sind

Die Firma cyclespeed24 konnte in der Vergangenheit ihre Software-Entwicklungsprojekte nicht immer erfolgreich abschließen. Sie wurden zwar häufig "in time" und "in budget" beendet, aber die entwickelten Features entsprachen nicht den Erwartungen der Anwender in den Fachbereichen. Dies erschwerte die Akzeptanz dieser Anwendungen durch die Fachbereiche und mit dem Image der IT-Abteilung war es innerhalb des Unternehmens auch nicht zum Besten bestellt.

Da die interne Entwicklungsmannschaft die größeren Projekte häufig nicht alleine umsetzen konnte, wurden mit einem externen Dienstleister Werkverträge zu Festpreisen geschlossen. Die Anforderungen so zu beschreiben, dass sie eine juristisch verwertbare Grundlage für den Werkvertrag bildeten, bedeutete erhebliche Aufwände für die anfordernden Fachbereiche, war aber Voraussetzung dafür, dass ein Projekt überhaupt genehmigt und ausgeschrieben werden konnte.

Die erstellten Lastenhefte nahmen schon einmal 200 Seiten und mehr ein. Sie enthielten alle denkbaren Anforderungen an das System, ohne dass jeweils eine Aufwand-Nutzen-Betrachtung erfolgt wäre, und damit auch jede Menge unnötige Funktionalitäten. Die Begeisterung (und in der Folge das Engagement) der Mitarbeiter, die sich in der Vor-Projekt-Phase in der korrekten Abwicklung ihres Tagesgeschäfts beeinträchtigt sahen, fiel entsprechend gering aus.

Hinzu kam, dass im Falle der Vergabe einzelner Projektphasen an den externen Dienstleister eine firmenübergreifende Informationsweitergabe erforderlich war. Diese führte, wie bei dem bekannten Kinderspiel "Flüsterpost", zum Verlust von Informationen und zu Fehlinterpretationen, wenn eine Anforderung nicht exakt genug beschrieben worden war.

Aus diesem Grund entschied die Geschäftsführung, dass künftig jede Software mit Hilfe von Scrum entwickelt werden sollte und zur Beschreibung der Anforderungen User Stories verwendet werden sollten. Die User Stories sollten in einem Collaboration Tool erfasst werden, um die Transparenz zu verbessern.

Mit dieser Änderung im Softwareentwicklungsprozess waren folgende Hoffnungen verbunden:

- Da die Anforderungen nun aus Sicht des Anwenders formuliert wurden, erwartete die Geschäftsführung, dass die Hemmschwelle für nicht-IT-affine Mitarbeiter aus den Fachbereichen, die jedoch Experten in ihrem Fachgebiet waren, wesentlich geringer sein würde, sich im Projekt mit ihrer Expertise einzubringen. Dabei sollten passgenauere Funktionalitäten herauskommen.
- In jedem Fachbereich sollte künftig derjenige Verantwortliche die User Stories erfassen und aktualisieren, der auch bisher bei Projekten als wichtiger Ansprechpartner fungierte.
- Die Visualisierung aller User Stories in einem Collaboration Tool sollte den Entwicklungsstand der Features transparent machen. Alle Mitarbeiter der Firma erhielten Lesezugriff auf dieses Tool, so dass sie sich jederzeit auch einen Überblick darüber verschaffen konnten, welche User Stories für die nächsten Entwicklungsiterationen vorgesehen waren.
- Durch den direkten Austausch wollte man zwischen Vertretern der anfordernden Fachbereiche und Mitarbeitern des Entwicklungsteams Fehlinterpretationen bei Anforderungen vermeiden. Dies sollte aber auch dazu führen, dass Anforderungen zu einem frühen Zeitpunkt abgelehnt würden, wenn die Aufwand-Nutzen-Betrachtung entsprechend negativ ausfiele.
- Die Aufwände für die Vertreter der Fachbereiche würden sich verringern, da nicht mehr alle Anforderungen "bis ins Letzte" spezifiziert wurden, sondern nur jeweils diejenigen, bei denen die Aufwand-Nutzen-Betrachtung positiv ausfiel und die entsprechend hoch priorisiert wurden.

Soweit die Theorie. Bei der Arbeit mit den User Stories stellte sich jedoch heraus, dass es doch noch einige formale und strukturelle "Stolpersteine" gab, die es aus dem Weg zu räumen galt.

Formale "Stolpersteine"

Fachliche Abhängigkeiten nicht bedacht

Problem

Da die Firma cyclespeed24 durch den Einsatz neuer Technologien die bestehende Software um neue Funktionalitäten erweitern und neue Schnittstellen zu Fremdsystemen implementieren wollte, um die Attraktivität der Software für die Kunden zu steigern, starteten viele Produktideen gleichzeitig. Dies erschwerte es dem Product Owner, der die Anforderungen verwaltete und priorisierte, den Überblick über die wachsende Menge an Ideen und damit User Stories zu behalten.

Die Folge war, dass das Entwicklungsteam zwar mit konstanter Geschwindigkeit User Stories in guter Qualität in den Sprints umsetzte, die User Stories aber nicht gemeinsam nutzbar waren. Der Product Owner hatte die fachliche Abhängigkeiten nicht erkannt und deshalb die User Stories nicht immer richtig priorisiert.

Lösung

In den Retrospektiven nach den ersten Sprints diskutierten der Product Owner, Scrum Master und die Entwickler diese Schwierigkeiten und suchten gemeinsam nach einer Lösung. Nach einigen Überlegungen entschlossen sie sich, die Erfahrungen der Anforderungsspezialisten in den Fachabteilungen hinsichtlich UML-Modellierung zu nutzen und für die Darstellung des gesamten zu entwickelnden Systems mehrere Kontextdiagramme zu verwenden. In diesen Kontextdiagrammen sind die Anforderungen, ihre Beziehungen zueinander und zu den Nutzern des Systems in grober Granularität dargestellt (Bild 1).

Durch dieses Vorgehen gewannen Anforderer und Softwareentwickler ein gemeinsames Verständnis von der gewünschten Funktionalität und möglichen Abhängigkeiten von anderen Funktionalitäten. Auch erlaubte es bereits Rückschlüsse auf die erforderliche Systemarchitektur. So sollte es z.B. Herstellern künftig möglich sein, ihre Produktinformationen für das Portal so bereitzustellen, dass ein Administrator von cyclespeed24 diese würde importieren und ggf. ergänzen können. Für die Importschnittstelle konnten daher die benötigten Felder definiert und eine sinnvolle Technik ausgewählt werden.

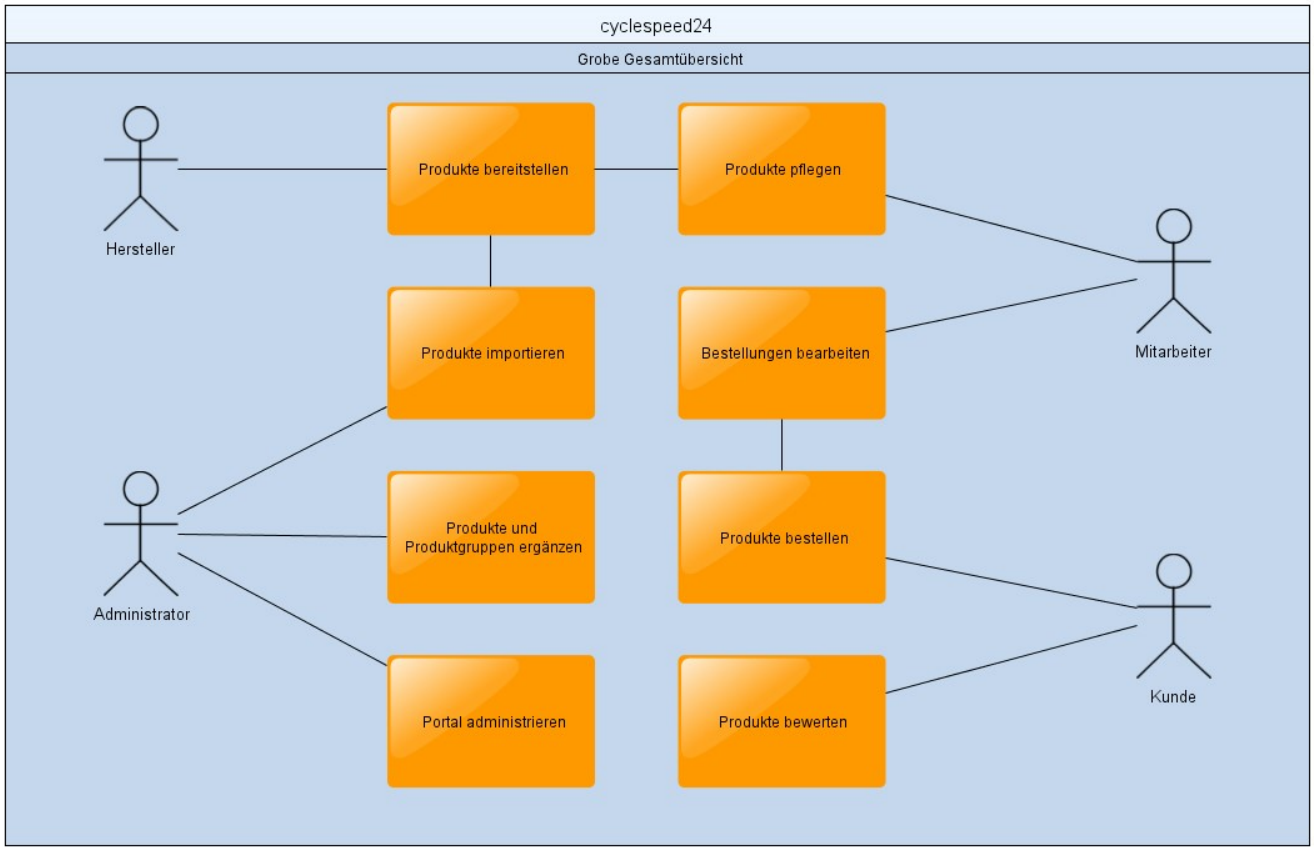


Bild 1: Grobe Gesamtübersicht der Anforderungen, ihrer Beziehungen zueinander und der jeweiligen Nutzer (Kontextdiagramm).

User Stories zu groß gewählt

Problem

Die Entwicklung nach Scrum mit dem regelmäßigen Feedback im Sprint Review war für alle Softwareentwickler von cyclespeed24 neu. Um eine lösungsorientierte Projektkultur mit häufigen Plan-do-check-act-Zyklen zu etablieren, entschied sich das Scrum-Team für eine Sprintlänge von zwei Wochen.

Ganz im Sinne des Agilen Manifests hatte sich das Entwicklungsteam gleich zu Projektbeginn darauf geeinigt, dass der Projektfortschritt nur anhand der lauffähigen Software gemessen werden sollte. Nach den ersten Sprints stellte das Team allerdings fest, dass es in einem Sprint nur ein bis drei Stories vollständig umsetzen konnte. Der Product Owner musste häufig unvollendete User Stories wieder in das Product Backlog zurücknehmen. Es wurde deutlich, dass die User Stories kleiner werden mussten, damit das Entwicklungsteam in der Lage war, sechs bis zehn Stories in einem Sprint fertigzustellen.

Lösung

Bei cyclespeed24 wendeten die Anforderer aus den Fachbereichen in Zusammenarbeit mit den Entwicklungs-teams verschiedene Methoden an, um die User Stories auf eine handhabbare Größe zuzuschneiden:

- Aufteilen in Workflow-Schritte
- Festlegen von Durchführungsregeln
- Aufteilen nach Eingabearten
- Untergliedern nach Datenoperationen

Diese Methoden werden im Folgenden näher ausgeführt.

Aufteilen in Workflow-Schritte

Folgende User Story scheint auf den ersten Blick recht einfach umsetzbar zu sein:

"Als Administrator kann ich Produktinformationen auf der Homepage veröffentlichen, damit diese den Interessenten und Kunden für eine Kaufentscheidung zur Verfügung stehen."

Im Gespräch zwischen dem Anforderungssteller und dem Entwicklungsteam stellte sich jedoch heraus, dass diese Produktinformationen nicht ohne vorherige Prüfung veröffentlicht werden sollten. Es bedurfte eines Freigabeprozesses, während dessen es möglich sein sollte, die Produktinformationen durch Mitarbeiter des Fachbereichs prüfen, Änderungen vornehmen und die Freigabe nach dem Vier-Augen-Prinzip genehmigen zu lassen.

Die Umsetzung aller damit verbundenen Anforderungen hätte nicht in einen Sprint gepasst. Die Anforderung wurde daraufhin anhand der Schritte im Workflow in mehrere User Stories aufgeteilt:

User Story 1:

"Als Administrator
... kann ich Produktinformationen direkt auf der Homepage veröffentlichen."

Dies entspricht der absoluten Mindestfunktionalität, die erforderlich wäre, damit die formulierte Anforderung als erfüllt gelten kann. In diesem Fall reichte dies natürlich nicht aus. Deshalb muss diese User Story noch ergänzt werden um:

... kann ich Produktinformationen zum Review durch den Fachbereich ablegen."

User Story 2:

"Als Mitarbeiter des Fachbereiches
... kann ich Produktinformationen ändern.

... kann ich zum Review abgelegte Produktinformationen nach dem Vier-Augen-Prinzip freigeben."

User Story 3:

"Als Administrator
... kann ich freigegebene Produktinformationen für die produktive Seite publizieren, damit diese den Interessenten und Kunden für eine Kaufentscheidung zur Verfügung stehen."

Festlegen von Durchführungsregeln

Häufig wird eine Anforderung so formuliert, dass für den Entwickler ein Interpretationsspielraum bleibt, wie das Ziel erreicht werden kann:

"Als Interessent kann ich ein Produkt mit Hilfe einer Suchfunktion finden."

In der Diskussion der Anforderung und durch das Ausformulieren der Akzeptanzkriterien, anhand derer die Anforderung als erfüllt gilt, wird meist schnell klar, wie dieses Ziel, hier das Auffinden eines Produkts über eine Suchfunktion, erreicht werden kann:

"Als Interessent
... kann ich ein Produkt durch die Eingabe eines einzigen Suchbegriffs finden." (User Story 1)
... kann ich ein Produkt durch Eingabe eines alternativen Suchbegriffs finden." (User Story 2)
... kann ich ein Produkt durch die gleichzeitige Eingabe von mehreren Suchbegriffen finden." (User Story 3)

In diesem Fall kann also die Suchfunktion in mehreren Ausprägungen entwickelt werden: von einer eher einfachen Suche bis hin zu komplexen Suchanfragen inklusive alternativer Suchbegriffe. Jede Ausprägung stellt dann eine eigene User Story dar.

Aufteilen nach Eingabearten

Bei Anforderungen, welche die Benutzeroberfläche betreffen, kann eine Aufteilung in verschiedene User Stories über die verschiedenen Eingabearten erfolgen. In der Regel wird in einer ersten User Story eine einfache Art der Eingabe entwickelt, welche im späteren Verlauf in einer weiteren User Story komplexer gestaltet werden kann.

User Story 1:

"Als Kunde kann ich ein Wunschdatum für die Auslieferung des von mir bestellten Artikels eingeben."

Akzeptanzkriterium: System bietet einfaches Eingabefeld an.

User Story 2:

"Als Kunde kann ich ein Wunschdatum für die Auslieferung des von mir bestellten Artikels per Kalenderauswahl auswählen. Die Feiertage sind im Kalender ausgegraut dargestellt."

Akzeptanzkriterien: System bietet Kalenderauswahl an. System zeigt in der Kalenderauswahl die Feiertage an. Die Feiertage sind nicht auswählbar.

Untergliedern in Datenoperationen

Die vierte Möglichkeit, Anforderungen in unterschiedliche User Stories zu unterteilen, stellt die Aufteilung nach den Datenoperationen "Create", "Read", "Update" und "Delete" (CRUD) dar. Wenn man in Anforderungen die Schlüsselwörter "pflegen" oder "verwalten" antrifft, verbergen sich dahinter meistens viele separat umsetzbare User Stories.

"Als Kunde kann ich meinen Account pflegen, um meine Daten immer aktuell zu halten."

Wird z.B. zu:

"Als Interessent
... kann ich einen Account anlegen, um Kunde zu werden."

"Als Kunde
... kann ich die Daten in meinem Account bearbeiten, um meine Daten immer aktuell zu halten."
... kann ich meinen Account löschen." (oder: "... kann ich beantragen, dass mein Account gelöscht wird.")

Strukturelle "Stolpersteine"

Feste Ansprechpartner in den Fachbereichen fehlen

Problem

In den bisherigen Projekten der Firma cyclespeed24 wurden die unterschiedlichen Projektphasen immer von verschiedenen Fachbereichen durchgeführt. So waren die Anforderungsspezialisten verantwortlich für die Phase der Lastenhefterstellung, die nach erfolgreichem Erreichen des Meilensteins "Lastenheft erstellt" das fertige Lastenheft an die Entwicklung übergaben. Eine Zusammenarbeit zwischen den Anforderungsspezialisten und den Entwicklern fand nur bei kritischen Punkten statt, z.B. wenn die Gefahr bestand, dass ein Meilenstein nicht erreicht werden konnte, da die Anforderungsspezialisten im Sinne einer bestmöglichen Auslastung bereits für ein neues Projekt verplant waren.

Agile Vorgehensweisen betonen jedoch das regelmäßige und enge Zusammenarbeiten aller Personen, die für die Realisierung des Vorhabens notwendig sind. Kollaborative Elemente waren aber zu Projektstart nicht in der Organisation der Firma verankert. So war z.B. die regelmäßige Mitarbeit der Fachbereiche an IT-Vorhaben nicht etabliert.

Aus diesen Gründen traten im Projektverlauf folgende, aufeinander aufbauende negative Effekte auf:

- Für die Mitarbeiter der anfordernden Fachbereiche hatte es gemäß ihrer persönlichen Zielvereinbarungen höchste Priorität, ihr Tagesgeschäft abzuwickeln. Deshalb meinten sie, keine Zeit dafür zu haben, auf Abruf für das Projekt als Ansprechpartner zur Klärung von Anforderungen zur Verfügung zu stehen.
- Dadurch hatte das Entwicklungsteam keine festen Ansprechpartner für die jeweiligen Fachbereiche, welche befugt gewesen wären, über Anforderungen zu entscheiden.
- Für die Klärung offener Fragen mit den Fachbereichen musste das Entwicklungsteam einen Mitarbeiter in Vollzeit abstellen, der für die eigentliche Entwicklungsarbeit nicht mehr zur Verfügung stand.
- Der aktuelle Stand der Anforderungen war nicht für alle Mitarbeiter transparent, da keine etablierten Kommunikationswege zur Verbreitung dieser Informationen innerhalb der Fachbereiche existierten und auch keine fachbereichsübergreifenden.
- Dadurch hegten manche Mitarbeiter in den Fachbereichen falsche Erwartungen bezüglich der umgesetzten Anforderungen.
- Da ihre Erwartungen nicht erfüllt wurden, kam bei diesen Mitarbeitern Frust auf.
- Diese Frustrationen führten zur Verstimmungen: Die Fachbereiche fühlten sich nicht abgeholt; die IT hat wieder einmal das Falsche entwickelt, also hatte sich also im Grunde mit der Einführung agiler Vorgehensweisen nichts verbessert.

Lösung

Damit die Situation, die das Entwicklungsteam und der Scrum Master dem Lenkungsausschuss berichtete, nicht weiter eskalierte, vereinbarte die Unternehmensführung mit den Führungskräften der Mitarbeiter, die für das Projekt benötigt wurden, dass deren Zielvereinbarungen entsprechend anzupassen seien. Des Weiteren wurden im Rahmen einer (nachgeholten) umfangreichen Umfeld- und Stakeholderanalyse die Schwachpunkte in der derzeitigen Kommunikationsstruktur innerhalb des Unternehmens ermittelt und entsprechende Gegenmaßnahmen ergriffen:

- eine einmalige Informationsveranstaltung, die von der Geschäftsführung für alle Mitarbeiter zum Projektstatus und zum aktuellen Stand der Anforderungen initiiert wurde
- regelmäßige Updates des Product Owners zum Projektstand für alle Mitarbeiter im Intranet
- regelmäßige persönliche Information der betroffenen Fachbereiche über den Projektverlauf durch die Führungskräfte
- Ernennung eines Ansprechpartners aus jedem Fachbereich und aus dem Management, der für Fragen zum Vorhaben allen Mitarbeitern zur Verfügung stand

Unterschiedliches Verständnis vom Inhalt einer Anforderung

Problem

Bei der Zusammenarbeit zwischen den Entwicklern und den Mitarbeitern aus den anfordernden Fachbereichen (bzw. dem Product Owner) bestand immer wieder eine grundsätzliche Herausforderung darin, dass alle Beteiligten dasselbe Verständnis von einer Anforderung und ihrer Realisierbarkeit erhielten (Bild 2).

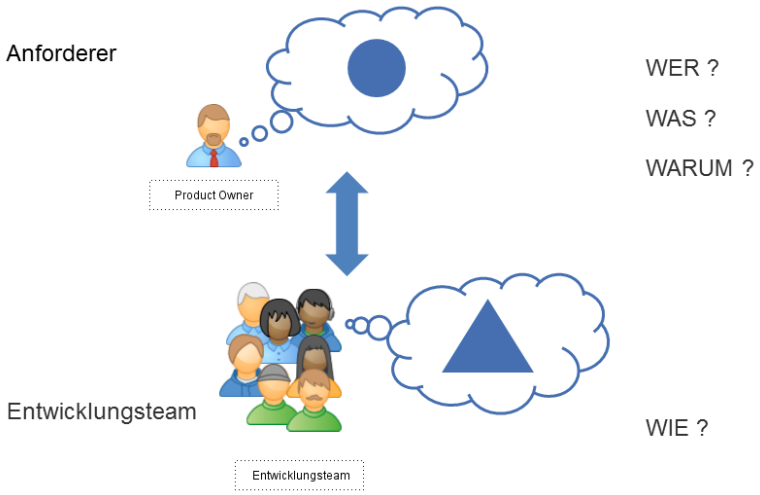


Bild 2: Zuständigkeiten von Anforderer und Entwicklungsteam bei der Anforderungsdefinition.

Warum die regelmäßige persönliche Abstimmung zwischen beiden Seiten so wichtig ist, lässt sich anhand eines Alltagsbeispiels veranschaulichen:

"Wie der Bundespräsident der Ukraine erklären lässt, dass er nicht nach Jalta fahren wird."

Angenommen, ein derartiger Satz – dieses Beispiel ist aus einer Tageszeitung entnommen – wäre Inhalt einer Anforderungsbeschreibung. Er ist nur teilweise verständlich und der logische Zusammenhang zwischen den Teilsätzen ist für den Leser nicht ersichtlich. Eine Klärung dieses Sachverhalts könnte natürlich in einem E-Mail-Austausch erfolgen. Wenn es sich aber um einen inhaltlich noch komplexeren Sachverhalt handelt, ist eine persönliche Abstimmung zwischen den Beteiligten effektiver, da dort weniger Raum für Interpretationen bleibt.

Lösung

Aufgrund schlechter Erfahrungen aus früheren Projekten wurde bei cyclespeed24 eine kontinuierliche persönliche Zusammenarbeit zwischen dem Product Owner, der die Anforderungen aus den Fachabteilungen bündelte, und dem Entwicklungsteam fest im Projektvorgehen verankert. Im sog. "Grooming", das alle zwei Wochen stattfand, wurde der Anforderungskatalog (Product Backlog) gemeinsam gepflegt (Bild 3).

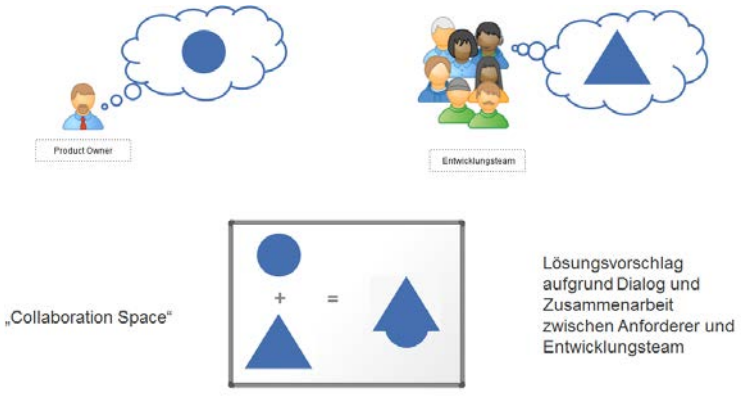


Bild 3: Gemeinsame Anforderungsklärung zwischen Product Owner und Entwicklungsteam.

Anforderer gibt technische Lösung vor

Problem

Ein wichtiges Merkmal einer User Story ist, wie bereits erwähnt, die Formulierung der Anforderung aus Anwendersicht. Der Anforderer macht sich also über das Wer, Was und Warum Gedanken. Das Entwicklungsteam ist gefordert, die geeignetste technische Lösung für die gestellte Anforderung auszuwählen und umzusetzen.

Allerdings kam es immer wieder vor, dass in der Anforderung bereits eine technische Lösung vorweg genommen wurde. Der Anforderer schrieb dem Entwicklungsteam also das Wie vor und nahm ihm den Entscheidungsspielraum, wie die User Story technisch umgesetzt werden könnte.

Vor allem bei Features, die architektonisch aufeinander aufbauen, kann es große Probleme verursachen, wenn der Anforderer nicht gewillt ist, der Argumentation des Entwicklungsteams zu folgen und die technische Lösung zu akzeptieren, die das Entwicklungsteam aufgrund seiner Fachkenntnis für die beste hält.

Lösung

Um zu vermeiden, dass der Anwender bei der Formulierung einer User Story die technische Lösung bereits vorwegnimmt, bietet es sich an, das Format "Als <Anwender> will ich … <Ziel>, um … <Nutzen> zu erzielen" zu verwenden. Dadurch wird der Fokus des Anwenders auf die für eine User Story wichtigen Aspekte gerichtet und das Abschweifen in technische Lösungsbeschreibungen vermieden.

Der letzte Teil der User Story, also der eigentliche Grund für diese Anforderung, wird bei der Formulierung der User Story gerne weggelassen. Um dies zu vermeiden, kann eine User Story auch dieses Format haben: "Um … <Nutzen> zu erzielen, will ich als <Anwender> <Ziel> … " Diese Art der Formulierung ist zugegebenermaßen etwas holprig, erfüllt aber den Zweck, den Nutzen einer User Story zu betonen, und erleichtert es, die User Stories nach Geschäftswert zu sortieren.

Ineffiziente Sprint Reviews

Problem

Die ersten Sprint Reviews von cyclespeed24 liefen meistens nach einem ähnlichen Muster ab: Das Entwicklungsteam brannte darauf, dem Product Owner auch bereits zu 90% fertiggestellte Funktionalitäten vorzuführen, die auf Basis der User Stories erstellt worden waren. Von den eingeladenen Mitarbeitern aus den Fachbereichen kam nur ein Teil zum Review, da sich das Projekt noch in der Entwicklung befand und die Führungskräfte der jeweiligen Mitarbeiter nicht wollten, dass diese regelmäßig zu diesen Terminen im Projekt gebunden waren.

Dies führte bei den ersten Sprint Reviews zu folgendem Ergebnis:

Der Product Owner konnte die vorgeführten Features nicht als "fertig" kennzeichnen, da

- sie nur in der lokalen Entwicklungsumgebung einwandfrei funktionierten.
- nicht alle notwendigen Vertreter aus den Fachbereichen anwesend waren, um die Features abzunehmen.
- die anwesenden Mitarbeiter unsicher waren, ob die Features die in den User Stories beschriebenen Anforderungen auch vollständig erfüllten.

So verursachten die ersten Sprint Reviews viel Enttäuschung im Entwicklungsteam, aber auch auf Seiten der anfordernden Fachbereiche wurden erste Stimmen laut, dass die IT "mal wieder am Bedarf vorbei" entwickeln würde.

Lösung

Um diesem Negativtrend entgegenzuwirken, wurden folgende Maßnahmen ergriffen:

1. Jede User Story erhielt genaue Akzeptanzkriterien, anhand derer möglichst automatisiert getestet werden konnte, ob die in der jeweiligen User Story enthaltenen Anforderungen auch erfüllt wurden.
2. Die Entwicklungsteams vereinbarten darüber hinaus mit den Fachbereichen eine "Definition of Done", also eine Definition, wann eine User Story als fertig gilt. Darin waren u.a. folgende Punkte enthalten:

- Lauffähigkeit der User Story auf einer Integrationsumgebung (nicht der lokalen Entwicklungsumgebung)
- Erfüllen der vorab definierten Akzeptanzkriterien
- Fortlaufendes Schreiben eines Benutzerhandbuchs für Anwender und Administratoren
- Erfüllen der übergreifend geltenden sog. "Nicht-Funktionalen-Anforderungen", z.B. Sicherheitsaspekte, Usability oder Performance

3. Die involvierten Mitarbeiter aus den Fachbereichen erhielten von ihren Vorgesetzten die Möglichkeit, an den Sprint Reviews teilzunehmen und sich vorher dafür vorzubereiten.
4. Alle beteiligten Mitarbeiter bekamen vor dem Sprint Review eine Information über die im Sprint enthaltenen User Stories inklusive der Akzeptanzkriterien.

Um die reibungslose Umsetzung der User Stories in einem Sprint zu unterstützen, wurde ein weiteres Instrument genutzt: die "Definition of Ready". Mit der Definition of Ready gibt das Entwicklungsteam dem Product Owner das Signal, dass eine User Story die erforderliche Reife besitzt, dass sie im Sprint Planning berücksichtigt werden kann.

Die Merkmale der Definition of Ready sind:

- Die User Story ist für alle Mitglieder des Entwicklungsteams klar verständlich.
- Die User Story ist testbar, d.h. es wurden entsprechende Akzeptanzkriterien festgelegt.
- Die User Story ist klein genug, dass sie neben weiteren User Stories in einem Sprint umgesetzt werden kann.

Nach den ersten Sprints wurde die Definition of Ready derartig erweitert, dass die Anforderung möglichst automatisiert testbar sein musste.

Unvorhersehbare Veränderungen im Projektumfang

Problem

Bei der Formulierung von Anforderungen an ein neues Softwaresystem wird vom Anforderer ein großes Abstraktionsvermögen abverlangt. Eine Software kann ja nicht angefasst werden; bestenfalls kann mittels Prototypen ein Überblick gegeben werden, wie die Software aussehen und sich verhalten könnte. Aber auch nur, wenn Geld für derartige "Wegwerfprodukte" ausgegeben werden darf.

In dem Projekt von cyclespeed24 wurden neue Features erstellt, welche aufgrund der veralteten Systemarchitektur bislang gar nicht entwickelt werden konnten. Die Anforderungen an das neue System waren daher auch für einen längeren Zeitraum in die Zukunft nicht immer bekannt, sondern entstanden quasi "auf der Strecke". Dies hatte zur Folge, dass Anforderungen an das Entwicklungsteam herangetragen wurden, die im Widerspruch zu bereits umgesetzten User Stories standen oder vorhandene Features wieder veränderten.

Lösung

Bei der Verwendung von agilen Vorgehensweisen müssen sich alle Beteiligten klar darüber sein, dass die Möglichkeit zum nahezu ständigen Einbringen von Anforderungen eine große Disziplin innerhalb der Fachbereiche erfordert, damit kein Anforderungschaos entsteht. Die gängigen Vorurteile gegenüber agilen Vorgehensweisen ("Ich bekomme immer weniger zu einem späteren Zeitpunkt und das zu einem höheren Preis") können nur dann abgebaut werden, wenn ein starker Product Owner ein klares Bild vom Produkt-/ Projektvorhaben hat und nicht-zielführende Anforderungen konsequent ablehnt.

Dazu muss er neben dem weitreichenden Überblick über die Systemlandschaft aus fachlicher Sicht auch ein technisches Grundverständnis mitbringen. Auch sollte der Product Owner in der Lage sein, aufgrund der vorhandenen Entwicklungsgeschwindigkeit eine längerfristige Projektplanung durchzuführen. Da bei agilen Vorgehensweisen die Details zu Anforderungen zu einem möglichst späten Zeitpunkt festgelegt werden, muss die Geschäftsleitung den Product Owner mit den erforderlichen Befugnissen ausstatten, um auch entsprechend vorgehen zu können, ohne dafür vorher mit allen Stakeholdern Rücksprache halten zu müssen.

Fazit

Die Verwendung von User Stories im Rahmen von agilen Vorgehensweisen ist kein "Selbstläufer". Auf den ersten Blick scheint das Vorgehen sehr einfach zu sein. Doch die Krux liegt hier im Detail.

Das Umdenken aller am Projekt beteiligten Abteilungen (Fachbereiche und IT) von einem "Lagerdenken" hin zu einer zielorientierten, pragmatischen und vor allem kollaborativen Zusammenarbeit erfordert zudem einen Kulturwandel und stellt für manche Unternehmen sicherlich eine große Herausforderung dar.

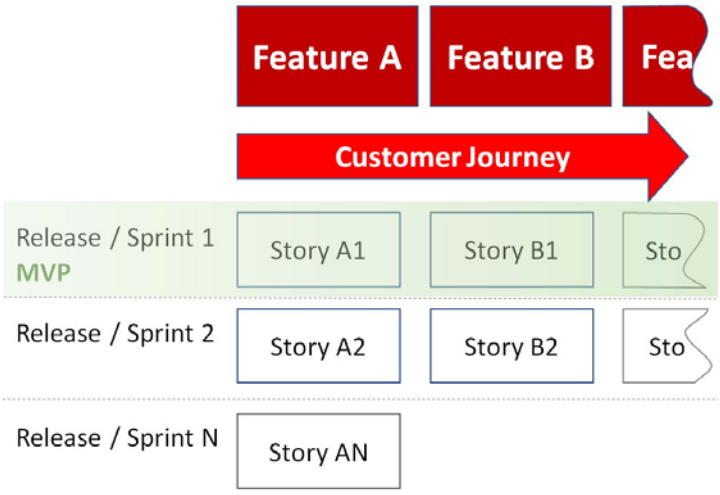
Dabei ist es zwingend notwendig, dass alle Führungskräfte der relevanten Fachbereiche diese Vorgehensweise unterstützen und für ihre Mitarbeiter die erforderlichen zeitlichen Freiräume zur regelmäßigen Zusammenarbeit mit dem Product Owner und dem Entwicklungsteam schaffen. Beschränkt sich die Arbeit mit User Stories auf die IT-Abteilung, ist dieses Vorgehen von vornherein zum Scheitern verurteilt, da die Entwickler im Projektverlauf immer wieder auf den Input der Fachbereiche angewiesen sind.

Letztendlich bietet die Arbeit mit User Stories neben der verbesserten fachübergreifenden Zusammenarbeit eine gleichmäßigere Auslastung der Fachbereiche, da hohe Arbeitsspitzen zur Erreichung von Meilensteinen vermieden werden. Dies wiederum ermöglicht eine bessere Einplanbarkeit der Mitarbeiter in den Fachbereichen und somit eine stärkere Vereinbarkeit der Projektarbeit mit dem Tagesgeschäft.

Literatur

- Cohn, Mike: Agile Estimating and Planning, Prentice Hall International, 2005 (ISBN-13: 978-0131479418)
- Cohn, Mike: User Stories Applied. For agile Software Development, Addison-Wesley Long-man, Amsterdam 2004 (ISBN-13: 978-0321205681)
- Gottesdiener, Ellen: Requirements by Collaboration. Workshops for defining Needs, Addison-Wesley Long-man, Amsterdam 2002 (ISBN-13: 978-0201786064)
- Leffingwell, Dean: Agile Software Requirements. Lean Requirements Practices for Teams, Programs, and the Enterprise, Addison Wesley, 2010 (ISBN-13: 978-0321635846)

Story Mapping



Story Mapping stellt die auf ein Produkt bezogene Kundenreise ("Customer Journey") strukturiert nach. Die so entstandene Story Map erleichtert es den Stakeholdern, die Entwicklung zusammengehöriger Bausteine (z.B. Features, User Storys) zu verstehen, diese in eine Reihenfolge zu bringen und sie in bearbeitbare Pakete zu fassen.

Einsatzmöglichkeiten

- strukturiertes Vorgespräch zur Auftragsklärung
- Erarbeitung von Lösungsansätzen im Team
- Justierung eines bestehenden Lösungsansatzes
- Einarbeitung neuen Personals hinsichtlich Kunden und Produkte
- Strukturierung und Priorisierung eines bestehenden Backlogs

Vorteile

- Strukturierte Analyse der Kundenbedürfnisse
- Praxisorientierte Zusammenfassung und Sortierung der Kundeninteraktionen
- Vollständige und übersichtliche Darstellung der bekannten Themen und Aufgaben
- Für alle Beteiligten leicht verständliche Darstellung

- Erzielen eines gemeinsamen Verständnisses durch die Erstellung im Team
- Einfache Erstellung
- Leichter Abgleich der Anforderungen mit exemplarischen Kunden
- Intuitive Methode, die kein Vorwissen erfordert

Grenzen, Risiken, Nachteile

- Fehlende Kundeninteraktion oder fehlendes Kundenfeedback führen mitunter zu Spekulationen. Die Methode ersetzt nicht den Austausch mit dem Kunden, sondern erleichtert diesen und veranschaulicht die Ergebnisse.
- Die Bearbeitung mit Story Mapping als Einzelperson ist möglich, aber riskant. Ein tiefes Verständnis aller produktrelevanten Aspekte (technisch und fachlich) ist für ein realistisches Ergebnis zwingend nötig. Dieses erreicht im Rahmen einer komplexen Produktentwicklung ein Team leichter als eine Einzelperson.
- Story Mapping liefert kein finales Ergebnis. Wiederholungen inklusive kritischer Prüfungen dienen der Aktualität und dem Einplanen von Veränderungen.

Ergebnis

- Story Map, deren Hauptelemente entlang der Kundenreise geordnet sind. Die Story Map ist ein "lebendes" Übersichtsdokument, welches im Laufe der Lösungserstellung weiter verfeinert werden sollte.
- Die Hauptelemente der Story Map können z.B. Epics, Storys / Features eines Produkts oder Aufgabenbereiche eines Projekts sein.
- Die Hauptelemente sind weiter strukturiert, evtl. bis zur Ebene von Arbeitspakten oder Aufgaben.
- Sammlung von Detailinformationen, die für die spätere Projektplanung oder zur Strukturierung eines Product Backlogs verwendet werden können
- Evtl. Liste von Elementen für andere Story Maps

Voraussetzungen

- Es muss bereits eine klare Vision des geplanten oder aktuellen Produkts geben, mit welchem die Kunden interagieren können.
- Bereitschaft der Beteiligten, sich auf diese visuelle, kundenorientierte Herangehensweise einzulassen und sie diszipliniert anzuwenden

Qualifizierung

Für die Durchführung der Methode bedarf es keiner speziellen Qualifikation. Unter den Mitwirkenden müssen Personen mit Expertise in Produktentwicklung sowie fachlichem Knowhow für den behandelten Gegenstand sein, sodass die Arbeits-gruppe für die Lösungsfindung relevante technische Abhängigkeiten erkennen kann.

Benötigte Informationen

- Bisher vorhandene Informationen über das geplante Produkt (z.B. Beschreibung der Vision) und die Rahmenbedingungen seines Einsatzes
- Fachwissen über die relevanten Kundengruppen (sowohl aktuelle als auch potentielle Kunden)
- Rahmenbedingungen für die Produktentwicklung
- Fachliche Anforderungen und Abhängigkeiten
- Informationen über Stakeholder hinsichtlich Einflussmöglichkeiten auf das Vorhaben
- Informationen über die Machbarkeit des Produkt- oder Anpassungswunsches

Benötigte Hilfsmittel

- Räumlichkeit: Der Raum muss ausreichend Platz für die Arbeitsgruppe und ausreichend Wandfläche, Moderationstafeln oder Flipcharts zur Erstellung der Story Maps bieten.
- Haftnotizen bzw. Karten und ausreichend Stifte in mehreren Farben und Dicken passend für die Arbeitsfläche, auf denen die Teilnehmer Ihre Ideen festhalten
- Sorgen Sie für ein angenehmes, das Arbeitsklima förderndes Ambiente. Eine ruhige, ungestörte Umgebung und ein einfaches Catering (z.B. Getränke) erleichtern die konstruktive Zusammenarbeit in Gruppen.

Durchführung

Schritt 1: Bereiten Sie das Story Mapping vor!

"Story Mapping" bezieht sich auf die sog. "Kundenreise" ("Customer Journey"), d.h. auf die Berührungspunkte ("Touch Points"), die Benutzer im zeitlichen Verlauf mit dem betrachteten Produkt haben. Führen Sie sich zunächst vor Augen, welche Inhaltsebene bzw. welche Elemente dieser Kundenreise Sie bearbeiten möchten. Im Projektmanagement (unabhängig von der Vorgehensweise oder dem Framework) gibt es mehrere Detaillierungsgrade, die zu unterschiedlichen Zeitpunkten oder spezifischen Situationen relevant sind.

Detaillierungsgrad Features / User Storys

Ist das Produkt bereits in seinem Funktionsumfang definiert, dann geht es darum, für die Umsetzung konkrete User Storys zu strukturieren. Aus der Story Map können dann Sprint-Backlogs erstellt werden.

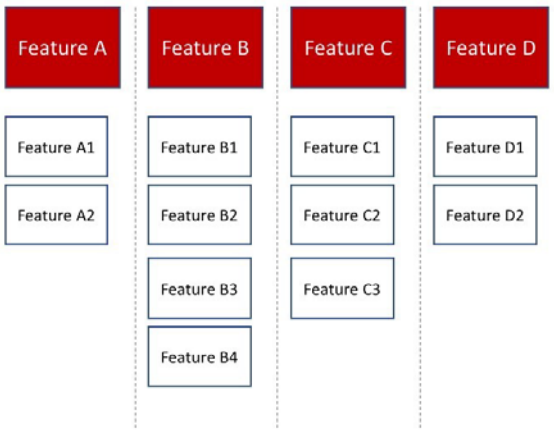


Bild 1: Features in User Storys strukturieren

Detaillierungsgrad Aufgabenbereiche

Bei einer traditionellen Projektplanung werden Arbeitspakete erstellt. Die Planer strukturieren hierzu das Vorhaben in Themenbereiche (oder Domains / Teilprojekte), die nach unterschiedlichen Kriterien gebildet werden können, z.B. nach Funktionen des Produkts. Diese werden dann in ausführbare und delegierbare Arbeitspakete gegliedert. Umgekehrt können auch bereits Arbeitspakete vorliegen, die dann entlang der Customer Journey geordnet und zu Gruppen zusammengefasst werden. Auf Basis der Story Map können dann Aufwände geschätzt sowie der Projektplan (Ablauf, Ressourcen usw.) erstellt werden. Ggf. können auch direkt Aufgaben im Team verteilt werden.

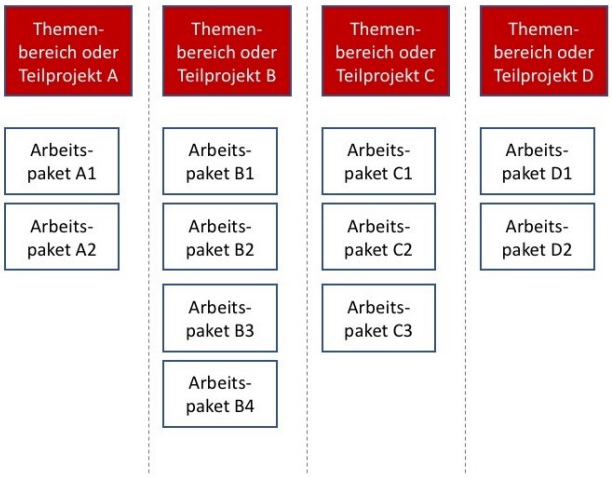


Bild 2: Gruppierung und Strukturierung von Arbeitspaketen

Bei der folgenden Beschreibung der Durchführung arbeiten wir auf der Ebene der Features und User Storys weiter, um Übersichtlichkeit zu gewährleisten. Wenn Sie stattdessen mit Themenbereichen bzw. Teilprojekten und Arbeitspaketen arbeiten, tauschen Sie die Begrifflichkeiten einfach aus.

Das Story Mapping benötigt lediglich eine freie Fläche wie z.B. Wand, Fenster, Moderationswand oder Tischfläche. Liegt bereits eine ähnliche Aufbereitung oder Visualisierung vor, berücksichtigen Sie entsprechend zusätzliche freie Fläche für weitere Sortierungen nach der Kundenreise. Wenn Sie z.B. bereits erste Inhalte vorbereitet haben, kann es sich in der Gruppenarbeit herausstellen, dass diese aufgrund weiterer Abhängigkeiten anders gruppiert werden müssen. Erfahrungsgemäß ist es umständlich, die Elemente innerhalb des bestehenden Aufbaus umzusortieren. Auf einer weiteren freien Fläche können diese leicht Schritt für Schritt umgezogen und neu strukturiert werden.

Sehen Sie auch Platz vor zur Sammlung thematisch verwandter, aber nicht im laufenden Vorhaben zur berücksichtigenden Ideen. Wenn Sie z.B. für Produkt A eine Story Map aus User Storys und Aufgaben erstellen, identifiziert das Team Ideen für das ähnliche Produkt B. Diese Ideen passen zwar nicht zu Produkt A, sollen aber nicht verworfen werden. Sammeln Sie diese deshalb auf einer weiteren Fläche.

Stellen Sie die Arbeitsgruppe zusammen. Sorgen Sie dafür, dass alle benötigten Wissensträger teilnehmen.

Schritt 2: Erklären Sie die Aufgabenstellung und die Methode!

Wenn Sie das Story Mapping alleine durchführen, können Sie diesen Schritt überspringen.

Stellen Sie zunächst die Aufgabenstellung und das Thema vor, z.B. die Sammlung und Strukturierung von Features für eine App zum mobilen Online-Verkauf der Produkte Ihres Unternehmens. Falls Sie bereits vorgearbeitet haben, stellen Sie Ihre ersten Inhalte vor, die Sie auf Karten notiert und nach Ihrer Vorstellung angeordnet haben. Geben Sie Raum für Rückfragen zum aktuellen Arbeitsstand, so dass die Gruppe einen gemeinsamen Einstieg finden kann.

Erklären Sie dem Team die Methode, wie Schritt 3 sie beschreibt. Passen Sie dabei die Erklärung bei Bedarf an die von Ihnen gewählte Detaillierungsebene an. Erläutern Sie das Prinzip der Kundenreise, evtl. anhand eines kurzen Beispiels (vgl.: Reinold, Daniel: Die Customer Experience Journey – so "tickt" mein Kunde, projektmagazin Ausgabe 03/2017).

Erläutern Sie auch bei Bedarf die unten zusammengestellten Tipps zur Anwendung der Methode.

Wenn der Teilnehmerkreis größer ist (mehr als ca. 6 Personen) empfiehlt es sich, in Kleingruppen zu arbeiten (zu je drei bis fünf Personen). Das bewirkt für einen regen Austausch aller Teilnehmer bei gleichzeitig übersichtlich gehaltenen Diskussionen.

Zentral ist, die "richtigen" Wissensträger einzubinden. Fachliche und technische Expertise, Branchenwissen und Informationen über die Marktsituation tragen gleichermaßen zum Erfolg Ihrer Arbeit bei. Wenn Sie im Lauf des Story Mapping hier Defizite erkennen und nicht sofort

lösen können, klammern Sie den davon betroffenen Bereich aus und notieren Sie den Bedarf für eine zweite Runde des Story Mappings.

Schritt 3: Erstellen Sie die Story Map!

Planen Sie ausreichend, aber nicht zu viel Zeit ein. Wenn Sie komplett von Beginn der Produktentwicklung an starten, empfehle ich eine Stunde zur konzentrierten Bearbeitung. So können die Teilnehmenden die Ergebnisse erst einmal "sacken lassen". Diese haben dann die Möglichkeit, Dritte nach ihrer Meinung zu befragen und mit zeitlichem Abstand die Kundenreise im Kopf nochmals durchzuspielen. Die Weiterentwicklung der Story Map (Veränderungen, Ergänzungen, Detaillierung) findet dann in weiteren Sessions statt.

Sammeln Sie im Rahmen eines klassischen Brainstormings Produktfeatures und ordnen Sie diese auf dem Interaktionspfad (horizontal von links nach rechts) der vorgesehenen Zielgruppe. Ziel ist, sowohl relevante Inhalte herauszustellen, als auch diese in die Reihenfolge der Benutzererfahrung zu bringen. Gruppieren Sie untereinander zum selben Berührungspunkt gehörende Unteraufgaben. Sortieren Sie diese von oben nach unten nach ihrer Wichtigkeit für das Funktionieren des gesamten Produkts.

Sie können die Themen sowohl "Top-Down", also übergeordnete Funktionen detaillieren, als auch "Bottom-Up", d.h. ausgehend von Basis-Funktionalitäten hin zu Oberbegriffen, bearbeiten. Welches Vorgehen sich besser eignet, hängt ab von Ihrer Arbeitsweise, der Arbeitsweise des Teams, dem Thema an sich und Ihren Moderationsfähigkeiten. Grundsätzlich geht es darum, beide Denkansätze in Einklang – also auf die Fläche zu bringen. Das Identifizieren, Ordnen und Strukturieren der Elemente einer Story Map soll bewirken, dass das Team fortlaufend den aktuellen Arbeitsstand reflektiert.

Die so erarbeitete Story Map liefert eine strukturierte Übersicht der übergeordneten Storys und der einzelnen Details zur Umsetzung. Aus ihr können Sie ableiten, wann welcher Teil der Umsetzung notwendig ist. Auf diese Weise können Sie mit "den richtigen" Dingen (oder auch dem "MVP" – Minimum Viable Product, s.u.) beginnen und sich schrittweise der vollständigen Lösung annähern.

Beispiel App-Entwicklung für mobilen Vertrieb

Beginnen Sie mit der grundlegenden Menüführung für die Kernaspekte Ihrer Produktidee. Im weiteren Verlauf sehen Sie weitere Funktionen vor, die verschiedene Bedürfnisse der Benutzer abdecken. Die Umsetzung der Funktionen ist zu Beginn möglicherweise weniger relevant, da evtl. die Grundfunktionen noch nicht implementiert sind oder Sie bereits Spezialfälle / spätere Bedarfe bearbeiten möchten. Für die Umsetzung beginnen Sie also mit den allerersten Themen wie: Bezug der App, Kernfunktionen, Basis-Daten hinsichtlich Eingabe oder Ausgabe.

Arbeiten mit der Story Map: Das Minimum Viable Product

Die Story Map liefert zum einen den Projektbeteiligten eine strukturierte Visualisierung des Leistungsumfangs. Zum anderen unterstützt sie durch die Orientierung an der Kundenreise die Priorisierung von Storys sowie die Definition von möglichen Releases.

Anhand eines einfachen Beispiels einer App für den mobilen Online-Versandhandel wird deutlich, weshalb die Produkt- bzw. Feature-Orientierung sowie die Abbildung der Kundenreise so wichtig sind.

Beispiel: App für Online-Vertrieb

Sie haben für die Verkaufs-App erste Funktionen gesammelt und diese in drei Kategorien zusammengefasst: Eine Produktsuche, eine Produktdarstellung (Detailinhalte) und der Kauf über einen abgebildeten Bestellprozess. Als nächstes wollen Sie definieren, mit welchem minimalem Leistungsumfang der App Sie auf den Markt gehen können. Welche ist nun für Ihre App die Wichtigste? Ein Kauf wird nicht ohne Bestellprozess passieren. Ein Bestellprozess wird nicht ohne Produktdetails abgeschlossen und ohne Suche werden Benutzer Ihre App wahrscheinlich bald wieder verlassen. Alle drei Kategorien sind also unverzichtbar. Dennoch müssen Sie im ersten Wurf nicht den gesamten Umfang der Story Map realisieren. Um die Minimallösung zu finden, müssen Sie in die Details der einzelnen Storys gehen:

Bild 3: Story Map, entsprechend der Customer Journey geordnet

- Die Produktsuche z.B. lässt sich zunächst anhand Produkt-Tags gestalten, später durch nachträgliche Produktkategorien oder Eigenschaften wie zum Beispiel Farbe und Form. Das Minimum wäre hier der Produktname oder die genannten Tags.
- Bei der Produktdarstellung müssen Sie überlegen, welche Produktdetails für den Anfang unverzichtbar sind: Eine Produktbeschreibung, ein Preis und die Funktion zur Übertragung in den Einkaufswagen könnten ausreichen. Später wären Erweiterungen möglich wie: Bilder, Details zu den Eigenschaften oder Rezessionen früherer Käufer.
- Beim Bestellprozess könnte der Minimalumfang die Bezahlung gegen Rechnung sein. Später könnten weitere Zahlungsmethoden, Hinweise zu Versandkosten, weitere Angebote oder ein Aufruf zur Bewertung der Interaktion mit der App folgen.

Diese Sichtweise ist wichtig. Wir neigen dazu, "alles gleich rein zu packen". Versuchen Sie durch Moderation und Reflektion während dem Anwenden der Methode das Minimum im Blick zu behalten und gleichzeitig die komplette Kundeninteraktion abzudecken.

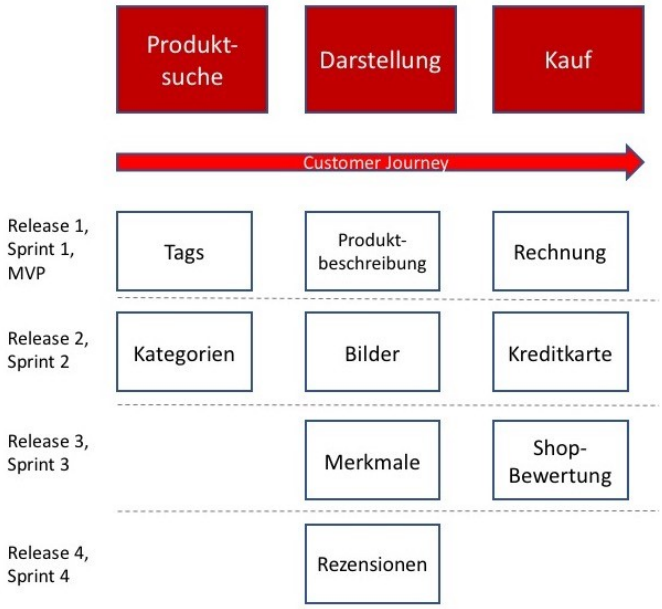


Bild 4: Das aus der Story Map abgeleitete Minimum Viable Product ("MVP")

Ergänzende / ähnliche Methoden

- Brainstorming – zur Sammlung der Elemente für die Story Map
- User Storys erstellen – zur vollständigen Ausformulierung der identifizierten User Storys
- Personas – zur Reflexion der Kundensicht bei Gestaltung der Kundenreise und der Auswahl der Features
- Projektstrukturplanung – ähnliche Methode aus dem traditionellen Projektmanagement. Die Story Map unterscheidet sich vom Projektstrukturplan durch die Abbildung der Kundenreise und die Priorisierung für das Minimum Viable Product.
- Produktbasierte Planung – ähnliche und ergänzende Methode aus PRINCE2®. Die Produktbasierte Planungstechnik hat nicht die Priorisierung gemäß der Kundenreise, bereitet aber mit dem Produktflussdiagramm die Umsetzung weiter vor. Die Story Map kann anstatt des Produktstrukturplans verwendet werden.

Praxistipps

- Verwenden Sie Haftnotizen. Sie lassen sich leicht neu platzieren und werden nicht beim geringsten Luftzug durcheinandergewirbelt.
- Halten Sie die Beschreibungen der einzelnen Karten so einfach wie möglich. Legen Sie die Umsetzung nicht zu detailliert fest. Technologische Gegebenheiten und Erkenntnisse ändern sich fortlaufend und werden erst relevant, wenn die tatsächliche Umsetzung ausgehandelt oder gestartet wird.
- Verlieren Sie sich nicht in Details. Wenn Sie sich bereits im Gebiet der Mutmaßung aufgrund fehlender Information oder Relevanz befinden, holen Sie sich entsprechende Expertise hinzu oder starten Sie mit der Umsetzung des Minimums und sammeln dann Erfahrungen aus den ersten Veröffentlichungen am Markt.

- Die Story Map ist ein "lebendes" Produkt, das Ihr Vorhaben begleitet. Das Story Mapping kann jederzeit wiederholt werden, insbesondere aufgrund neuer Erkenntnisse.
- Geben Sie die Möglichkeit für ein ausführliches Feedback. Die Story Map ist ein gutes Medium, um Dritten die Ideen zu erläutern, neue Expertise an Bord zu nehmen oder den Arbeitsstand zu überprüfen.

Varianten

Wie Sie die Ergebnisse aufbereiten oder später weiterverarbeiten, ist von Ihrem Vorgehen (oder dem Vorgehen Ihrer Auftraggeber) abhängig. Sie können die Ergebnisse im Rahmen von User Storys und Subtasks für Scrum aufarbeiten. Gleichermaßen ist es möglich, daraus Arbeitspakete für einen traditionellen Projektplan zu erstellen.

Herkunft

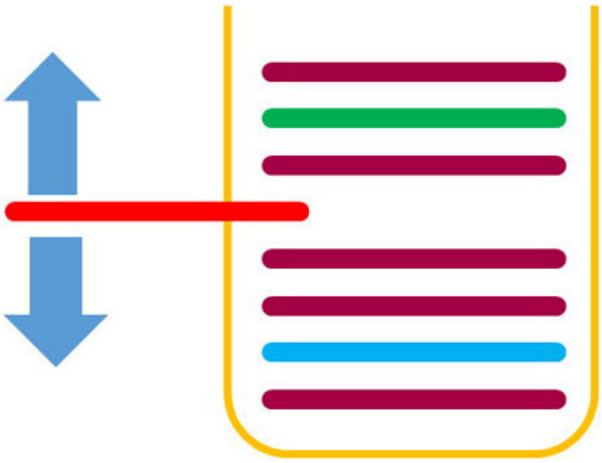
Story Mapping geht auf Jeff Patton (Patton, Jeff: User Story Mapping – Nutzerbedürfnisse besser verstehen als Schlüssel für erfolgreiche Produkte, 2015). Auf der Website von Patton and Associates (s. "Weitere Informationen") finden sich weitere Hintergrundinformationen zur Entwicklung der Story Map.

Autor

Daniel Reinold

Erstellt am: 23.09.2018

WSJF – Weighted Shortest Job First



WSJF (Weighted Shortest Job First) dient der Priorisierung von Aufgaben, insbesondere von Backlogs in agil gemanagten Projekten. WSJF berücksichtigt verschiedene Aspekte des geschäftlichen Nutzens einer Aufgabe und stellt dies in Relation zum Aufwand. So entsteht eine aus betriebswirtschaftlicher Sicht optimale Priorisierung der Aufgaben.

Einsatzmöglichkeiten

- Priorisierung von Backlog-Items (User Storys, Anforderungen, Aufgaben) eines agil gemanagten Projekts
- Priorisierung von Aufgabenlisten im geschäftlichen Kontext

Vorteile

- Betriebswirtschaftliche Optimierung von Backlogs hinsichtlich des Nutzen-Aufwand-Verhältnisses
- Neue Backlog-Items können schnell priorisiert und eingeordnet werden.
- Backlog-Priorisierungen erfolgen schneller und mit geringerem Aufwand, da es kaum Diskussionsbedarf gibt.

Grenzen, Risiken, Nachteile

Die Eingangsparameter beruhen auf subjektiven Schätzungen, sodass auch die Metriken einer Unsicherheit unterliegen.

Ergebnis

WSJF-Index, der die relative Priorisierung einer Aufgabe im Backlog definiert

Voraussetzungen

- Fertigstellungstermine müssen hinsichtlich ihrer Verbindlichkeit (z.B. Folgen einer Verzögerung) bewertet sein.
- Die Backlog-Items sind so definiert, dass sie unabhängig voneinander erledigt werden können.
- Verständnis über die zu bewertenden Backlog-Items muss vorhanden sein.

Qualifizierung

Die Methode selbst erfordert keine spezifischen Qualifikationen. Die Anwender sollten über fachliche Expertise im Projektgegenstand verfügen und in der Lage sein, die betriebswirtschaftlichen Auswirkungen zu beurteilen.

Benötigte Informationen

- Liste mit definierten Aufgaben bzw. ein Backlog
- falls vorhanden: Business Case des Projekts bzw. der Aufgabenstellung
- falls vorhanden: Terminplan mit verbindlichen Fertigstellungsterminen
- falls vorhanden: Aufwandsschätzungen für die zu bewertenden Elemente

Wenn verschiedene Personenkreise Aufwandsschätzung und Priorisierung durchführen, sollte die Aufwandsschätzung bereits im Vorfeld erfolgen. Ansonsten wird sie als erster Schritt der Durchführung (s.u.) vorgenommen.

Benötigte Hilfsmittel

- Bei moderierter, analoger Gruppenarbeit: Pinnwand oder große Tischfläche, Karteikarten, Klebepunkte oder ähnliches
- Bei rechnergestütztem Vorgehen: Tabellenkalkulationsprogramm, evtl. Beamer

Durchführung

- Schritt 1: Schätzen Sie die Größe der Backlog-Items ab!
- Schritt 2: Werte der Backlog-Items schätzen
- Schritt 3: Berechnen Sie die WSJF-Indexwerte!
- Schritt 4: Planen und steuern Sie Ihren Arbeitsvorrat gemäß der WSJF-Priorisierung!
- Ergänzende / ähnliche Methoden

Vorbemerkungen:

- Da in einem Backlog unterschiedliche Elemente wie Anforderungen, User Storys, Aufgaben, etc. enthalten sein können, werden diese Elemente im Folgenden ganz allgemein "Backlog-Items" bezeichnet.
- Aus Gründen der einfacheren Lesbarkeit wird im Folgenden nur die grammatikalisch männliche Form (der Product Owner, Teilnehmer) verwendet. Es sind dabei aber stets Personen jeden Geschlechts gemeint.

In agilen Vorgehensmodellen (z.B. Scrum) ist der Product Owner verantwortlich dafür, die Backlog-Items zu priorisieren. Allerdings definieren diese Vorgehensmodelle dafür keine eindeutige Metrik. In der Praxis priorisieren Product Owner deshalb die Backlog-Items nach ihrer subjektiven Einschätzung, wobei Themen wie Risiko, geschäftlicher Wert (Value) und geschätzter Aufwand mit einfließen. Dies verursacht insbesondere beim Einfügen neuer Items einen hohen Aufwand, da auch die bereits bewerteten Items noch einmal betrachtet werden müssen.

WSJF bietet hier eine Hilfestellung mit ökonomischem Fokus, indem es eine Kennzahl, den WSJF-Index, gemäß einem genau definierten Ermittlungsverfahren berechnet. Auf diese Weise liefert WSJF direkt die Position, an der das neue Item eingefügt werden muss.

Größe und Wert eines Backlog-Items

Die Priorität eines Backlog-Items ergibt sich aus zweien seiner Attribute: Größe und Wert.

Die **Größe** entspricht dem absolut oder relativ geschätzten Aufwand für die Erfüllung des Backlog-Items gemäß der für dieses Projekt gültigen Definition of Done. Die Größe kann in beliebigen Einheiten angegeben sein, typisch sind Story Points oder Arbeitsaufwand in Zeiteinheiten. Schritt 1 beschreibt die Ermittlung der Größe.

Wert bedeutet generell, Umsatz zu erhöhen oder Kosten zu reduzieren. Jeder Tag, der bis zur Umsetzung der geplanten Maßnahme vergeht, bedeutet also entgangenen Umsatz bzw. entgangene Kostenersparnis. Diese Kosten, die durch das Liegenlassen einer Aufgabe entstehen, werden als Verzögerungskosten (Cost of Delay, COD) bezeichnet. Verzögerungskosten werden zwar nur von wenigen Organisationen erhoben, können aber schnell die Kosten für die Abarbeitung

um ein Vielfaches übersteigen. WSJF hilft, die Aufgaben bzw. Backlog-Items unter Berücksichtigung von COD und deren Größe zu priorisieren. Schritt 2 beschreibt, wie der Wert einer Aufgabe in relativen Einheiten quantifiziert werden kann.

Der WSJF-Index

Mithilfe von Größe und Wert können die Backlog-Items nun in eine betriebswirtschaftlich optimierte Reihenfolge der Abarbeitung gebracht werden. Die Betrachtung zweier Extremfälle (gleicher Wert / gleiche Größe) liefert folgende Regeln:

- Haben zwei Aufgaben dieselbe Größe, aber einen unterschiedlichen Wert, sollte die wertvollere Aufgabe zuerst abgearbeitet werden
- Haben zwei Aufgaben denselben Wert, aber eine unterschiedliche Größe, sollte die kleinere Aufgabe zuerst abgearbeitet

Reinertsen (s.u. Herkunft) schlägt als Verallgemeinerung dieser Regeln vor, Wert und Größe zueinander ins Verhältnis zu setzen und somit eine Priorisierungsmetrik zu erhalten:

$$WSJF\text{-}Index = \frac{Wert}{Größe}$$

Gleichung 1: Definition des WSJF-Indexes

Schritte 3 und 4 erläutern, wie Sie die Priorisierung erstellen und mit ihr arbeiten.

Schritt 1: Schätzen Sie die Größe der Backlog-Items ab!

Für den Nenner des WSJF-Indexes benötigen Sie die Größe des Backlog-Items. Da der WSJF-Index dimensionslos berechnet wird, kann die Einheit für die Größe beliebig gewählt werden, solange sie für alle Backlog-Items identisch ist. Im agilen Projektmanagement wird die Größe meist in relativen Einheiten geschätzt, z.B. Story-Points. Genauso können Sie hier aber mit Aufwandsschätzungen in Zeiteinheiten (Arbeitsstunden, -tage usw.) arbeiten.

Wenn bereits eine Aufwandsschätzung für die Backlog-Items vorliegt, prüfen Sie diese auf Vollständigkeit und Konsistenz.

Andernfalls führen Sie eine Aufwandsschätzung für die Backlog-Items durch. Für den Einsatz von WSJF ist es nicht relevant, welches Schätzverfahren Sie benutzen. Wählen Sie deshalb ein dem vorgegebenen Projektmanagement-System entsprechendes Verfahren. Bei agilen Vorgehensweisen können Sie z.B. Planning Poker oder Team Estimation Game verwenden. Im traditionellen Projektmanagement können Sie z.B. die PERT-Drei-Punkt-Schätzung einsetzen.

Schritt 2: Werte der Backlog-Items schätzen

Da der WSJF-Index dimensionslos verwendet wird, kann auch der Wert in beliebigen Einheiten (Euro, Dollar, relative Schätzpunkte usw.) angegeben werden, solange dies für alle Items einheitlich geschieht. Allerdings ist der Wert einer Aufgabe selten konkret in monetären Einheiten bezifferbar. Falls dies bei Ihrem Projekt möglich ist, können Sie direkt diese Angaben verwenden, selbst wenn sie nur grob sind. Oft ist es jedoch schneller und einfacher wie bei agilen Größenschätzungen hier mit relativen Schätzpunkten zu arbeiten. Mögliche Vorgehensweisen sind unten beschrieben.

Reinertsen schlägt vor, den im Zähler stehenden Wert des Items in drei Wertbestandteile aufzuteilen: Geschäftswert, Wert der Zeitkritikalität und Risikowert. Tabelle 1 erläutert die Bedeutung dieser drei Werte.

Wert	Bedeutung	Beispiele
Geschäftswert	Steigerung von Umsatz oder Reduzierung von Kosten.	Marktwert einer Produktfunktionalität, Kosteneinsparungen durch Prozessverbesserungen oder Werkzeuge,
Wert der Zeitkritikalität	Einfluss einer bestimmten Deadline auf den Wert. Meist wird das Erledigen der Aufgabe nach dieser Deadline wertlos.	Deadlines in Verträgen, durch Gesetzesänderungen, bei termingebundenen Ereignissen wie z.B. Messen und Veranstaltungen
Risikowert	Riskantere Aufgaben sollten früher angegangen werden als weniger riskante.	Unklarheiten bzgl. Anforderungen, Markt und Technologien

Tabelle 1: Beschreibung der drei Wertbestandteile einer Aufgabe gemäß Reinertsen

Mit diesen Definitionen ergibt sich die Formel für den WSJF-Index nach Reinertsen wie folgt:

$$WSJF\text{-}Index = \frac{Geschäftswert + Wert\ der\ Zeitkritikalität + Risikowert}{Größe\ der\ Aufgabe}$$

Gleichung 2: WSJF-Index mit differenzierten Wertanteilen

Einzelwerte über Voting-Verfahren bestimmen

Zur Abstimmung des Werts zwischen mehreren Stakeholdern kann z.B. ein Pokerverfahren analog zum Planning Poker verwendet werden. Häufig anzutreffen sind Voting-Verfahren, z.B. mit Klebepunkten oder Spielgeld. Mit Voting-Verfahren wird in der Regel der Gesamtwert geschätzt und nicht die drei Unterkomponenten.

Für das Voting-Verfahren schreiben oder drucken Sie die Aufgaben auf Karten und verteilen Sie diese auf einem Tisch oder pinnen Sie sie an eine Moderationswand. Erklären Sie die Themen beim Auslegen der Karten, wenn sie den Beteiligten noch nicht bekannt sind. Jeder Teilnehmer bekommt dann dieselbe Anzahl an Klebepunkten (Faustformel: jeder bekommt halb so viele Klebepunkte ausgehändigt wie Karten zu bewerten sind). Die Teilnehmer können dann mit den Punkten Aufgaben "kaufen". Dabei können sie ihre Punkte verteilen oder mehrere bzw. alle Punkte auf eine

Karte kleben. Alternativ zu den Klebepunkten können auch Pokerchips oder Spielgeldmünzen eingesetzt werden. Bild 1 zeigt beispielhaft eine im Voting-Verfahren bewertete User Story.

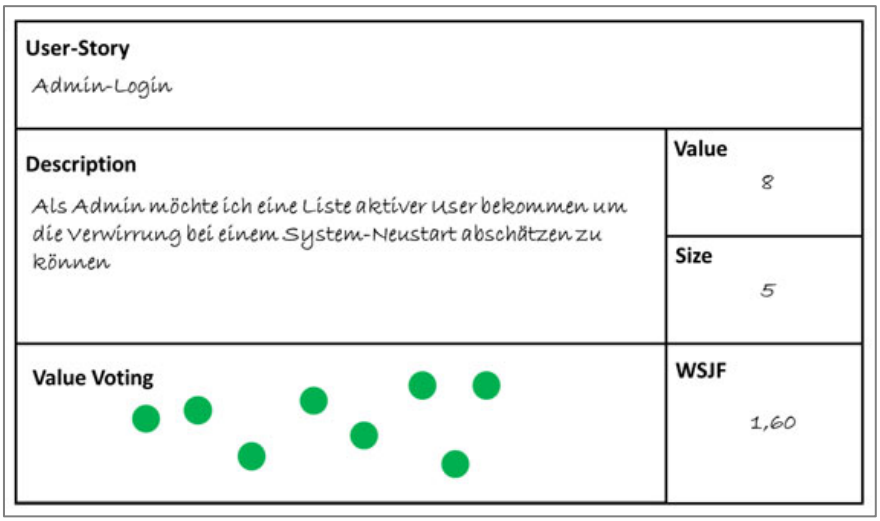


Bild 1: Beispiel User-Story-Karte mit Klebepunkt-Voting des Werts

Einzelwerte über parametrische Verfahren bestimmen

Transparenter wird die Abschätzung, wenn Sie versuchen, die drei Einzelwerte über weitere Formeln oder Berechnungen herzuleiten. Die Formel geht davon aus, dass alle drei Aspekte des Werts gleich gewichtet sind, also dieselbe Skala haben. Dies ist jedoch kein unumstößlicher Grundsatz. Wenn es für Sie sinnvoll erscheint, die drei Aspekte unterschiedlich zu gewichten, können Sie auch mit verschiedenen Skalen experimentieren.

Beispiel: Die Schätzung und Verrechnung von Einzelwerten könnte nach folgenden Regeln erfolgen:

- Jeder Einzelwert wird auf einer Skala von 1 bis 10 bewertet.
- Der **Geschäftswert** ergibt sich als Summe des Werts für die Aufgabenkategorie (Skala von 1 bis 5) und der Schätzung des Werts innerhalb dieser Kategorie (Skala von 1 bis 5).
- Der **Wert der Zeitkritikalität** beträgt 10 dividiert durch die Zahl der Wochen bis zur Deadline (gerundet auf eine ganze Zahl), minimal jedoch 1.
- Der **Risikowert** wird geschätzt auf einer Skala von 1 bis 10. Die Skalenwerte können noch genauer erläutert sein, z.B.: "3 bedeutet: Anforderungen sind klar, die eingesetzte Technik ist bekannt, wurde aber noch nicht für diesen Zweck eingesetzt".

Wenn Sie mit solchen, detaillierten Rechnungen arbeiten möchten, sollten Sie einige Aufgaben aus bereits abgeschlossenen Projekten im Team zu bewerten und den WSJF-Index berechnen. Vergleichen Sie die Ergebnisse mit den vorhandenen Erfahrungswerten, um ggf. die zugrunde gelegten Skalen und Regeln anzupassen.

Schritt 3: Berechnen Sie die WSJF-Indexwerte!

Berechnen Sie nun den Quotienten aus Wert und Größe und sortieren Sie die Backlog-Items absteigend nach den so erhaltenen WSJF-Indexwerten. Legen Sie die Anzahl der Nachkommastellen des WSFJ-Index so fest, dass Sie eine aussagekräftige Priorisierung erhalten Diese initiale Liste müssen Sie bei Bedarf noch umsortieren, falls z.B. Abhängigkeiten zwischen den Backlog-Items bestehen. Die Ergebnisse können Sie in einer Liste zusammenfassen, oder auf Karteikarten vermerken. Bild 1 skizziert ein mit WSJF priorisiertes Backlog.

Aufgabe	Wert	Größe	WSJF
Task 4711: xyz machen	17	3	5,66
Task 4843: abc machen	3	1	3,00
Task 4112: jkl machen	11	13	0,85

Bild 2: Tabelle mit Aufgaben und berechneten WSJF-Werten

Wenn Sie das Backlog initial auf diese Weise erstellt haben, können neu hinzukommende Items einfach eingefügt werden. Bewerten Sie Wert und Größe mit denselben Ansätzen die sie bei den anderen Items verwendet haben und berechnen Sie den WSJF-Index. Damit ist sofort die Position klar, ohne dass Sie explizit Vergleiche mit vorhandenen Backlog-Items vornehmen müssen.

Schritt 4: Planen und steuern Sie Ihren Arbeitsvorrat gemäß der WSJF-Priorisierung!

Bei der Arbeit mit WSJF wird allen Beteiligten eines schnell klar: Es reicht nicht, dass Aufgaben nur alleine nach Ihrem Wert sortiert werden. Um in naher Zeit abgearbeitet zu werden, muss die Aufgabe zudem klein sein, d.h. mit geringem Aufwand schnell umsetzbar. Denn nur so ergibt sich ein hoher WSJF-Index, da der Nenner gegenüber dem Zähler kleiner wird. Das gilt unabhängig davon, ob Sie agil (z.B. mit einem Backlog) entwickeln oder ob Sie Ihre traditionell geführte Aufgabenliste in eine eindeutige Reihenfolge überführen möchten. Durch diese nun entstehende eindeutige Reihenfolge, sprechen wir hier auch von Backlog-Items.

Auf Ihre konkrete Arbeit bezogen haben Sie durch den WSJF-Index immer die ökonomisch relevanten Backlog-Items als nächstes für den kommenden Schritt bestimmt. WSJF fördert ein gezieltes Aufspalten von großen Backlog-Items in wertvolle und weniger wertvolle, kleinere Backlog-Items, sodass die wertvolleren als nächstes bearbeitet werden. Weniger wertvolle und größere Backlog-Items rutschen in der Priorität nach unten, so dass sie u.U. nie umgesetzt werden. Über die Zeitkritikalität steigt ihre Priorität zwar im Lauf der Zeit, dennoch können u.U. andere Aufgaben immer noch einen besseren WSJF-Index haben. So hilft WSJF einen ökonomischen Fokus zu halten und Verschwendung zu vermeiden.

Mit zunehmender Laufzeit wird Zeitkritikalität und Risiko bei einzelnen Backlog-Items größer, sodass sie im Backlog weiter nach oben wandern. Dies verhindert, dass Backlog-Items immer verschoben werden und z.B. nur der Geschäftswert betrachtet wird. Damit das Backlog nicht immer weiter anwächst, können "Bodensatzaufgaben" über Altersanalysen identifiziert und entfernt werden (z.B. "alles was länger als 12 Monate im Backlog ist, wird gelöscht").

Unabhängig davon, ob Sie agil oder traditionell arbeiten, ob es sich um ein Projekt oder um Linientätigkeit handelt, ein eindeutig priorisiertes Backlog verschafft allen Beteiligten Vorteile. Es ist Basis für die im Projekt- oder Unternehmenssinn optimiert Arbeitsplanung und gewährleistet, dass mit den vorhandenen, begrenzten Ressourcen möglichst früh der höchste Nutzen erzielt wird.

Ergänzende / ähnliche Methoden

- Team Estimation Game – zur agilen Aufwandsschätzung
- Planning Poker – zur agilen Aufwandsschätzung
- PERT Dreipunkt-Schätzung – zur traditionellen Aufwandsschätzung
- Sprint Planning – zum Erstellen des Sprint Backlogs gemäß WSJF

Praxistipps

- Stecken Sie nicht zu viel Aufwand in die Schätzungen. Die Qualität von Schätzungen skaliert in der Regel nicht mit dem in sie gesteckten Aufwand, letztendlich ist jede Schätzung falsch. Oft ist es sinnvoller schnell (=preiswert) und grob zu schätzen.
- Zwei Aufgaben dürfen nicht dieselbe Priorität haben. Wenn der WSJF-Index für zwei Backlog-Items gleich groß ist, entscheiden Sie im direkten Vergleich, welches der beiden Items die höhere Priorität bekommt.

Herkunft

Die WSJF-Methode geht auf den Berater für Produktentwicklung Donald G. Reinertsen zurück. Er beschreibt sie im Buch Reinertsen, Donald G.: The Principles of Product Development Flow: Second Generation Lean Product Development, 2009, Celeritas Publishing. Reinerts ist Gründer und Inhaber von Reinerts & Associates, California, USA.

Autoren

Joachim Pfeffer und Sebastian Schneider

Erstellt am: 19.11.2017

Gesamtaufwände in agilen Projekten schätzen – darauf müssen Sie achten

Dorian Gloski
Dipl.-Physiker, freiberuflicher Software-Architekt

"Dieses Projekt war bei jeder Statusmeldung im Rahmen von Zeit und Budget. Als das Budget dann verbraucht war, stellten wir aber fest, dass noch jede Menge Projekt übrig war."
Aussage eines Controllers über ein agiles Projekt

Das scheinbare Paradoxon dieser Aussage mag dem einen oder anderen bekannt vorkommen, der in agilen Projekten tätig ist. Jede Iteration wurde, wie geplant, erfolgreich umgesetzt; doch am Ende des Projekts fehlen notwendige Anforderungen, die nicht mehr innerhalb des veranschlagten Gesamtbudgets umgesetzt werden können. Wird dies erst kurz vor dem ursprünglich geplanten Projektende deutlich, sind Konflikte zwischen Auftraggeber und Dienstleister vorprogrammiert.

Doch welche Missverständnisse führen zu solchen Konflikten und wie können Sie diese verhindern? Wie lässt sich der Gesamtaufwand in einem agilen Projekt bestimmen? Ist das überhaupt möglich? Diesen Fragen widmet sich der vorliegende Beitrag.

Zunächst stelle ich die Charakteristika des Schätzens einzelner Sprints und des Gesamtaufwands gegenüber. Anschließend folgen Handlungsanweisungen, wie sich der Gesamtaufwand in agilen Projekten in den Griff bekommen lässt. Als agile Vorgehensweise wird in diesem Beitrag "Scrum" betrachtet (für eine Einführung in Scrum s. auch Wirdemann, projektmagazin 21/2009).

Schätzen von Sprints

Die Schätzung, wie viele User Storys im anstehenden Sprint umgesetzt werden können, erfolgt in Scrum in der Regel mit Hilfe von Storypoints und dem sogenannten Planning Poker (für Storypoints s. Wirdemann, projektmagazin 02/2010, für Planning Poker s. Linssen, projektmagazin 10/2012).

Die wichtigsten Merkmale einer solchen Sprint-Schätzung sind:

- Die Anforderungen dürfen sich während eines Sprints nicht ändern.
- Allen Beteiligten ist klar, was zu tun ist, um eine User Story umzusetzen.
- Die Schätzung erfolgt durch die Entwickler, die die Anforderungen auch umsetzen.

- Durch die Verwendung von "Poker-Karten" geben die Entwickler ihre Schätzung unabhängig voneinander ab.
- Bei großen Abweichungen der Schätzungen einzelner User Storys werden deren Ursachen geklärt und die Entwickler schätzen erneut, bis ein Konsens erreicht wird.
- Das Team verständigt sich auf die Umsetzung der für einen Sprint besprochenen User Storys, d.h. es übernimmt die Verantwortung für deren Umsetzung.
- Durch die Verwendung der abstrakten Einheit "Storypoints" sind die Schätzungen unabhängig von Zeiteinheiten (wie z.B. Personentage) und lassen sich anhand von Erfahrungsdaten aus bisherigen Sprints auf Personentage umrechnen.

Lässt man Scrum-Spezifika wie Storypoints und Sprints weg, so ergibt sich daraus:

- Anforderungen müssen klar und unveränderlich sein.
- Die Entwickler, die die Anforderungen umsetzen, schätzen diese unabhängig voneiander.
- Im Falle stark abweichender Schätzungen wird ein Konsens gesucht.
- Es wird nicht in Tagen, sondern in einer abstrakten Größe geschätzt.
- Die geschätzten Anforderungen werden zeitnah umgesetzt.

Möchte man eine möglichst gute Expertenschätzung erreichen, so wird man bei ähnlichen Regeln landen (vgl. z.B. das Delphi-Verfahren, Schulz 2012).

Die Erfüllung dieser Punkte gewährleistet also – unabhängig vom Projektvorgehen – eine belastbare Aufwandsschätzung.

Schätzen des Gesamtaufwands

In einem Projekt, das mit einer traditionellen Vorgehensweise durchgeführt wird, existieren Pflichtenhefte oder Feinspezifikationen, die den Gesamtumfang des Projekts – idealerweise – genau dokumentieren. Damit ist es möglich, diese Dokumente zur Schätzung des Gesamtaufwands heranzuziehen.

Scrum hingegen fordert keine Dokumente, die zu Projektbeginn den Gesamtumfang beschreiben, sondern hier werden die Anforderungen im Product Backlog gesammelt. Dieses umfasst die noch nicht umgesetzten User Storys in einer priorisierten Reihenfolge. Im Gegensatz zu einem Pflichtenheft erhebt das Product Backlog allerdings nicht den Anspruch, initial alle Anforderungen eines Projekts zu enthalten. Es lässt sich jederzeit verändern, d. h. es können User Storys geändert, gelöscht oder hinzugefügt werden. Auch die Umpriorisierung von User Storys ist möglich und damit die Veränderung der Reihenfolge, in der diese abgearbeitet werden.

Für User Storys eines Product Backlogs gibt es, im Gegensatz zu den User Storys eines konkreten Sprints, keinerlei Qualitätsvorgaben. D. h. diese können von vagen Ideen ("Wir brauchen eine Nutzerverwaltung") bis zu sehr ausformulierten User Storys ("Die Anlegen-Maske der Nutzerverwaltung soll folgendes Layout haben") reichen. Mangels Alternativen ist das Product Backlog in der Regel das einzige Artefakt mit dessen Hilfe sich der Gesamtaufwand eines agilen Projekts schätzen lässt.

Das bedeutet: Während die Schätzung eines konkreten Sprints ein qualitativ hochwertiges Ergebnis ermöglicht, gilt dies für die Schätzung des Product Backlogs nicht. Und selbst wenn sich der Aufwand für die Umsetzung eines Backlogs einigermaßen genau schätzen lässt, so sagt dies nur dann etwas über den Gesamtaufwand des Projekts aus, wenn das Backlog bzgl. der Anforderungen vollständig ist und sich nicht mehr grundlegend ändert.

Beim Schätzen des Product Backlogs müssen die Projektverantwortlichen mit weitere Unsicherheitsfaktoren rechnen.

Unsicherheitsfaktor Velocity

Das Schätzen des Product Backlogs erfolgt in der Regel nach einem ähnlichen Vorgehen wie das Schätzen der User Storys für einen einzelnen Sprint. Meist übernimmt, aus Gründen der Zeitersparnis, die Schätzung nur ein kleiner Teil des Teams oder es werden vereinfachte Methoden wie z.B. das Team Estimation Game (Bjoersne et al., projektmagazin 02/2013) verwendet. Sind alle User Storys des Projekts geschätzt, erfolgt die Berechnung des Gesamtaufwands mit Hilfe der bisherigen Velocity des Teams. Da sich diese Velocity im Laufe des Projekts ändern kann (weil sich z.B. die Teamzusammensetzung ändert oder die Erfahrung des Teams wächst), führt dies zu einem Unsicherheitsfaktor bei der Schätzung.

Unsicherheitsfaktor Storypoints

Die mehrmalige Schätzung eines unveränderten Backlogs in Storypoints kann zu unterschiedlichen Ergebnissen führen. Storypoints sind eine Schätzgröße, die von vielen Faktoren abhängt. Beispielsweise wird ein Team mit zunehmender Erfahrung anders schätzen als zuvor. Auch Änderungen bei der Zusammensetzung des Teams können die Schätzungen beeinflussen. Ebenso kann der Zeitpunkt der Umsetzung einer User Story die dafür geschätzten Storypoints verändern; z.B. kostet die Unterstützung von Mehrsprachigkeit einer Software mehr Storypoints, je später sie im Projekt umgesetzt wird. Auch die Granularität von User Storys verändert deren Anzahl. So werden viele kleine User Storys tendenziell mit mehr Storypoints geschätzt als wenige große.

Damit gilt für das Schätzen des Product Backlogs:

- Aus den User Storys geht nicht unbedingt klar hervor, was zu tun ist.
- Der Inhalt eines Backlogs kann sich jederzeit ändern.
- User Storys, die für die erfolgreiche Umsetzung des Projekts unbedingt notwendig sind, sind zum Zeitpunkt der Schätzung möglicherweise noch nicht im Backlog enthalten.

- Ob und wann die geschätzten Storys umgesetzt werden, ist unklar.
- Die User Storys werden nicht unbedingt von den Entwicklern geschätzt, die für deren Umsetzung verantwortlich sind. Sei es, weil nur ein Teil des Teams die Schätzung übernimmt, sei es, weil sich bis zum Zeitpunkt ihrer Umsetzung die Zusammensetzung des Teams geändert hat.
- Während sich das Team durch sein explizites "commit" für die Umsetzung der User Storys innerhalb eines Sprints verantwortlich fühlt, gilt dies für das Backlog nicht; denn dieses kann sich jederzeit ändern und User Storys enthalten, bei denen noch unklar ist, was für deren Umsetzung zu tun ist.
- Die Berechnung des Aufwands für die Umsetzung der User Storys erfolgt mit der bisherigen Velocity des Teams. Da sich die Velocity über die Projektlaufzeit ändern kann, führt dies zu einem weiteren Unsicherheitsfaktor.

Jeder dieser Punkte kann sich negativ auf die Qualität der Schätzung auswirken. Insbesondere Veränderungen können die Schätzung des Product Backlogs innerhalb kurzer Zeit wertlos machen.

Doch worauf soll man achten, wenn die Schätzung des Gesamtaufwands unbedingt notwendig für weitere Entscheidungen ist?

Auf den "Charakter" kommt es an

Bild 1: Für die Erstellung eines Motorrollers wie diesem kann man den Gesamtaufwand und das Projektende nur schwer bestimmen. Man kann aber nach jeder Änderung damit losfahren und seine Freunde beeindrucken. Ähnliches gilt für eine Website, die nach dem Prinzip des "Continuous delivery" erstellt wird.

Die Erhöhung der Schätzgenauigkeit und der Verbindlichkeit für eine einzelne Iteration und eine damit einhergehende Ungenauigkeit bei der Schätzung des Gesamtaufwands ist ein Charakteristikum agiler Projekte, das ganz bewusst so gewählt wurde. Dahinter steckt die Annahme, dass sich Anforderungen mittel- und langfristig so stark ändern, dass eine Schätzung des Gesamtaufwands eigentlich nicht möglich ist. Im agilen Umfeld wird daher durchaus darüber diskutiert, ob das Schätzen des Gesamtaufwands für ein Projekt überhaupt sinnvoll ist oder ob man ganz darauf verzichten kann (siehe z.B. Jeffries). In der Regel hängt die Notwendigkeit einer belastbaren Schätzung des Gesamtaufwands und dessen Überwachung vom Charakter des konkreten Projekts ab. Dazu zwei Beispiele:

Betrachten wir zunächst ein Startup, das eine Webseite erstellen möchte, über die sich Haustierbesitzer das Futter für ihr Tier selbst zusammenstellen können. Das verfügbare Budget ist begrenzt. Ziel dieses Startups ist es, die Webseite möglichst frühzeitig produktiv zu schalten und schnell so attraktiv zu machen, dass eine maximale Steigerung der Kundenanzahl und des Umsatzes erreicht wird. Es geht darum die Gewinnzone zu erreichen bzw. neue Kapitalgeber zu finden, bevor

das aktuelle Budget verbraucht ist. Den Gesamtaufwand kann man in diesem Fall außer Acht lassen und sich auf die frühzeitige und regelmäßige Auslieferung der Software konzentrieren (vgl. Bild 1).

Als zweites Beispiel betrachten wir eine Zeitung, die ihren Internetauftritt überarbeiten möchte. Da dabei das bestehende Content Management System (CMS) ersetzt wird, soll der Umstieg vom alten auf den neuen Webauftritt zu einem festen Zeitpunkt erfolgen. Wenn zu diesem Zeitpunkt nicht alle notwendigen Anforderungen umgesetzt sind, also z.B. nicht alle Backend-Systeme angebunden sind, kann die Webseite nicht produktiv gehen. Da für die Umstellung außerdem ein festes Budget bereitgestellt wurde, gibt es ein Interesse den geschätzten Gesamtaufwand möglichst nicht zu überschreiten.

Bild 2: Eine Hochzeitstorte sollte am besten zum vereinbarten Preis, auf jeden Fall aber zum vereinbarten Termin geliefert werden. Ähnliches gilt für Projekte, die zu einem Stichtag fertig sein müssen.

Gesamtaufwände im Griff behalten

Die bisherigen Ausführungen haben gezeigt, dass es von der Art des jeweiligen Projekts abhängt, ob Gesamtaufwände geschätzt und überwacht werden müssen. Im Folgenden betrachten wir nun einige Projekte, bei denen ein bestimmter Aufwand sowie ein fester Releasetermin eingehalten werden müssen und sehen uns im Zuge dessen diejenigen Aspekte genauer an, die für eine valide Schätzung und Einhaltung des Gesamtaufwands in agilen Projekten wichtig sind.

Anfang, Ende und Scope festlegen

Eine Grundvoraussetzung für die Schätzung des Gesamtaufwands besteht darin, dass sich zu Beginn eines agilen Projekts – wie in eigentlich jedem Projekt – alle Beteiligten darüber einig sein sollten, was Anfang, Ende und Scope des Projekts sind. Auch wenn dies eine Selbstverständlichkeit sein mag, sollte geklärt (und natürlich schriftlich festgehalten) werden, was alle Beteiligten nach Verbrauch des Budgets zum geplanten Releasetermin erwarten.

Im oben angeführten Beispiel der Ablösung des CMS hatten Auftraggeber und Auftragnehmer unterschiedliche Erwartungen. Der Auftraggeber ging davon aus, dass er seine Webseite zum vereinbarten Zeitpunkt und Preis bekommt und dabei das Backlog jederzeit ändern und umpriorisieren kann. Der Dienstleister hingegen ging davon aus, dass sich aus der – über die Iterationen gewachsenen – Größe des Backlogs eine Erhöhung des Budgets und eine Verschiebung des Releasetermins automatisch ergeben würden. Da der Auftraggeber auf der Erfüllung seines Vertrags (Ablösung eines CMS zu einem festen Zeitpunkt und Budget) bestand, konnte der Dienstleister das Projekt schließlich nur mit hohen Verlusten umsetzen.

Rahmenbedingungen vor Entwicklungsbeginn klären

Müssen der Gesamtaufwand und ein fester Releasetermin in einem Projekt unbedingt eingehalten werden, bedeutet dies auch höhere Anforderungen an die Qualität sowie die Vollständigkeit des Product Backlogs zu Beginn des Projekts. Das Backlog muss schon zu diesem frühen Zeitpunkt möglichst alle notwendigen Anforderungen enthalten und deren prinzipielle Umsetzung sollte klar sein.

Fehlen die notwendigen Informationen, um Entscheidungen bezüglich der Durchführung treffen zu können, empfiehlt sich eine Analysephase vor dem Beginn des eigentlichen Projekts. Daraus sollte sich u.a. ergeben, welche Technologie zur Umsetzung verwendet werden soll und welche Schnittstellen und Systeme betroffen sind. Dabei muss nicht unbedingt jede einzelne Schnittstelle bis ins Detail ausspezifiziert sein, wie man das z.B. bei einem traditionellen Vorgehen machen würde; in der Regel sind Format und Technologie ausreichend.

Ein Ergebnis der Analysephase sollte auf jeden Fall auch eine Risikobewertung und Lückenanalyse sein, die aufzeigen, wo Unsicherheiten bei der Umsetzung des Projekts bestehen und welche Risiken sich daraus ergeben. Die Analysephase sollte – iterativ – so oft durchlaufen werden, bis Auftragnehmer und -geber die übrig gebliebenen Risiken akzeptieren.

Bild 3: Abreißen oder Renovieren? Wenn man vor Baubeginn nicht klärt, ob ein Haus abgerissen oder renoviert werden soll, läuft man Gefahr, Aufwände in die Renovierung zu stecken, die dann, falls sich später die Notwendigkeit eines Abrisses ergibt, umsonst waren. Ähnlich sollte man z.B. bei einem Softwareprojekt vor Projektbeginn klären, ob ein System ersetzt oder angepasst werden soll, um sich eventuell unnötige Anpassungsaufwände zu sparen.

Beispiel: Umsetzung von Geschäftsprozessen aufgrund gesetzlicher Bestimmungen

Aufgrund gesetzlicher Anforderungen müssen in einem Unternehmen einige Geschäftsprozesse bis zu einem bestimmten Stichtag umgesetzt werden. Für das Projekt wird ein Budget beantragt, das nicht überschritten werden darf. Um das Beispiel einfach zu halten, gehen wir davon aus, dass die Systemlandschaft des Auftraggebers aus einem Abrechnungs-, einem ERP- und zwei CRM-Systemen besteht (Bild 4).

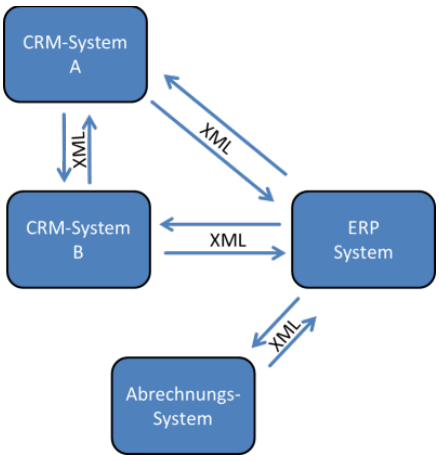


Bild 4: Systemdiagramm für das Beispiel "Geschäftsprozesse".

Das Ziel des Projekts und der späteste Releasetermin werden durch die gesetzlichen Vorgaben vorgegeben. Die konkreten Details zur Umsetzung der Geschäftsprozesse sind zum Zeitpunkt der Ausschreibung noch offen. Dies betrifft die zu verwendende Technologie sowie die Anzahl und Art der betroffenen Schnittstellen und Systeme. Auch liegt es im Ermessen des Auftragnehmers, ob für die Umsetzung bestehende Systeme angepasst oder teilweise

durch neue Systeme ersetzt werden sollen. Der Auftragnehmer muss ein Festpreisangebot abgeben. Die Staffelung des Angebots nach verschiedenen Szenarien (falls System A abgelöst werden muss, ist der Preis ein anderer, als wenn das System angepasst wird) ist nicht erlaubt.

Die Bandbreite der eingereichten Angebote verwunderte zunächst den Auftraggeber. Eine Analyse der Angebote ergab, dass der Aufwand für das Projekt nicht belastbar zu schätzen war. Die Anbieter konnten über die Anzahl der betroffenen Systeme und Schnittstellen nur Annahmen treffen und ihr Angebot dann um einen mehr oder weniger großen Risikoaufschlag erhöhen.

Analyse der Ausgangssituation

Da der Auftraggeber die konkreten Rahmenbedingungen nicht festgelegt hatte, waren folglich auch keine konkreten Schätzungen möglich. Wie also auch bei Projekten, die mit traditionellen PM-Methoden durchgeführt werden, ist für die Gesamtaufwandsschätzung agil durchgeführter Projekte die Kenntnis über die genauen Rahmenbedingungen unabdingbar. Die häufige Annahme, dass agile Projekte nur ein geringes Maß an Vorgaben benötigen, ist zumindest im Kontext der Gesamtaufwandsschätzung falsch.

Das Ergebnis dieser Analyse hätte z.B. so aussehen können, dass zur Umsetzung des Geschäftsprozesses die beiden CRM-Systeme A und B durch ein neues CRM-System ersetzt werden sollen. Das ERP-System bleibt unverändert und das Abrechnungssystem muss angepasst werden. Zudem muss der Auftragnehmer jeweils zwei Dateischnittstellen im XML-Format anpassen bzw. neu erstellen (Bild 5). Solche klar definierten Rahmenbedingungen in der Ausschreibung hätten es den Auftragnehmern ermöglicht, wesentlich konkretere Angebote abzugeben.

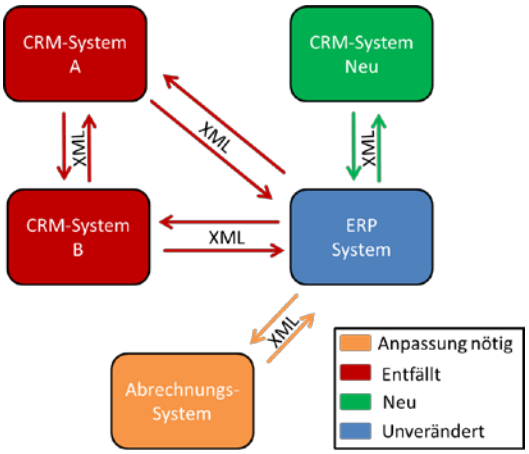


Bild 5: Mögliches Systemdiagramm nach Umsetzung des Geschäftsprozesses.

Änderungen begrenzen

Während der Laufzeit eines Projekts ändern sich in der Regel auch die Anforderungen. Die zunehmende Beliebtheit agiler Vorgehensweise lässt sich vor allem darauf begründen, dass diese auf den Umgang mit Änderungen optimiert sind.

Häufige Änderungen führen allerdings oft auch in agilen Projekten dazu, dass Aufwände steigen und schwerer kalkulierbar sind. Nehmen wir an, dass im obigen Beispiel "Geschäftsprozesse" die benötigten Anpassungen des Abrechnungssystems viel größer sind als erwartet und schließlich doch entschieden wird, das System zu ersetzen. Die Aufwände, die bis dahin in die Änderung des Systems geflossen sind, wären dann vergeblich gewesen – der Gesamtaufwand wird sich erhöhen.

Anforderungen richtig beschreiben

Es ist normal, dass Anforderungen geändert werden, und der adäquate Umgang damit trägt in agilen Projekten wesentlich dazu bei, dass Projekte zum größtmöglichen Nutzen der Auftraggeber umgesetzt werden. Kann man dennoch etwas tun, um ein unkontrolliertes Wachsen des Aufwands zu verhindern?

Zunächst sollten die Verantwortlichen bei der Analyse und Beschreibung der Anforderungen, die sich stark auf den Aufwand auswirken können, genauso sorgfältig vorgehen, als wären diese unveränderbar. Agile Vorgehensweisen verleiten leider zu der Meinung: "Das können wir ja jederzeit ändern, denn wir arbeiten agil". Nehmen wir z.B. an, dass Anforderungen durch die Erweiterungen eines bestehenden Systems zunächst erfüllt werden können. Stellt sich dann durch geänderte bzw. zusätzliche Anforderungen heraus, dass dieses System vollständig ersetzt werden muss, so waren die Aufwände, die in die Erweiterung des Systems flossen, überflüssig.

Große Änderungen rechtzeitig an alle kommunizieren

Sollten sich – bei aller Sorgfalt – doch Änderungen ergeben, die den Aufwand deutlich erhöhen, dies frühzeitig allen Beteiligten mitteilen. Auf diese Weise lassen sich rechtzeitig Gegenmaßnahmen treffen, indem z.B. andere Anforderungen gestrichen werden. Eine gute Möglichkeit die Auswirkungen von Änderungen auf den Aufwand darzustellen, ist das Scrum Release Burndown Chart (s. Cohn). Dieses Diagramm muss man natürlich fortwährend pflegen und allen Beteiligten in einer jeweils aktuellen Version zur Verfügung stellen.

Einführung von Meilensteinen

Das Erreichen von Sprintzielen besagt, dass vereinbarte Anforderungen in der vereinbarten Zeit umgesetzt wurden. Der Auftraggeber kann dadurch die Leistungsfähigkeit und Qualität der Arbeit des Teams erkennen. Das Erreichen von Sprintzielen sagt allerdings nicht unbedingt etwas über den Fertigstellungsgrad des Gesamtprojekts aus.

Beispiel: Ablösung eines Redaktionssystems

Bei einer Zeitung sollte das Redaktionssystem abgelöst werden. Dies betraf im Wesentlichen zwei Teile: Die Artikelverwaltung, eine Art Content-Management-System zur Erstellung der Artikel, sowie das Blattplanungssystem zur Planung und Erstellung der Zeitungsseiten.

Die Umsetzung der beiden Systeme erfolgte mit einem agilen Vorgehen. Die Artikelverwaltung konnte zügig und problemlos implementiert und produktiv gesetzt werden. Beim Blattplanungssystem kam es hingegen zu Verzögerungen, da es Unstimmigkeiten bei den Anforderungen gab und die notwendigen Ansprechpartner nur schwer erreichbar waren. Um die Auslastung des Entwicklungsteams zu gewährleisten wurden in den Sprints, bis zur Fertigstellung der Anforderungen für die Blattplanung, Anforderungen zur Erhöhung der Ergonomie der Artikelverwaltung umgesetzt, die sich aus dem Feedback der Anwender ergaben.

Schließlich stellte man fest, dass durch die Umsetzung der Ergonomie-Anforderungen so viel Budget verbraucht wurde, dass das restliche Budget für die Umsetzung der Blattplanung nicht mehr ausreichte. Da die neue Artikelverwaltung auch mit der bestehenden Blattplanung funktionierte, einigte man sich darauf, die Realisierung der neuen Blattplanung zunächst einmal zurückzustellen und das restliche Budget für die weitere Verbesserung der Artikelverwaltung zu verwenden.

Fertigstellungsgrad mit Meilensteinen im Blick

Im diesem Beispiel wurden nach der Umsetzung der Artikelverwaltung die weiteren Sprints dazu verwendet, die Ergonomie zu verbessern. Das Budget dafür war jedoch eigentlich für das Blattplanungssystem gedacht. Der Fertigstellungsgrad für das Gesamtprojekt erhöhte sich damit nicht: Sprintziele sind also keine Meilensteine!

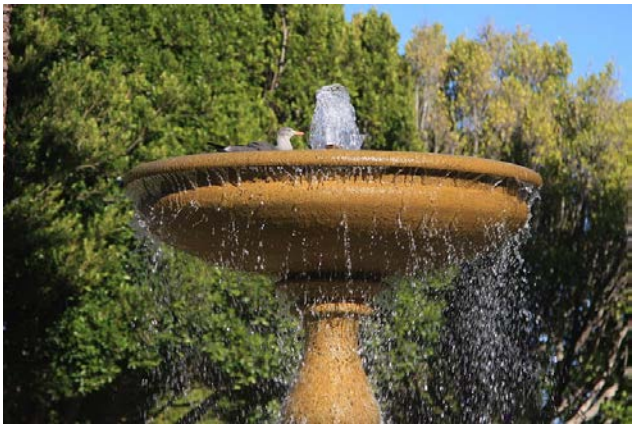


Bild 6: Stocken bei einem agilen Bauprojekt die Arbeiten, weil z.B. die Planungen für das Dach noch nicht fertig sind, so kann man die freien Kapazitäten z. B. nutzen um einen Brunnen zu bauen. Dies erhöht allerdings das kalkulierte Budget, falls der Brunnen darin nicht einkalkuliert war.

Um einen Überblick über den Fertigstellungsgrad zu erhalten, ist es ratsam, auch in agilen Projekten Meilensteine einzuführen. Im Beispiel könnten z.B. die Fertigstellung der Artikelverwaltung sowie die Erstellung der Blattplanung jeweils einen Meilenstein darstellen. Für jeden dieser Meilensteine sollte man den Aufwand schätzen und kontrollieren, ob der Meilenstein mit dem geplanten Aufwand erreicht werden kann. Darüber hinaus bietet es sich nach jedem Sprint an, auch den Fertigstellungsgrad des Projekts bzgl. der Meilensteine zu betrachten.

Änderungen verfolgen

Je stärker sich ein Product Backlog ändert, desto schwieriger ist es, valide Aussagen über den Restaufwand für die Umsetzung zu treffen. Es liegt in der Natur des Backlogs, dass die Hürde relativ niedrig ist, bestehende Anforderungen zu ändern; dadurch steigt die Gefahr eines Scope Creeps (s. Schleichender Funktionszuwachs). Änderungen sind bei agilen Projekten durchaus erwünscht, sollten aber explizit über einen Scope Change erfolgen (s. auch Änderungsmanagement). Im vorherigen Beispiel erfolgte die Änderung des Scopes (die Erweiterung der Artikelverwaltung statt der Implementierung der Blattplanung) schleichend, was jedoch glücklicherweise im Sinne aller Beteiligten war.

Um einen Scope Change frühzeitig zu erkennen, ist es sinnvoll, die Änderungen des Backlogs nachzuverfolgen. In der Regel bieten Tools, die zur Verwaltung eines Backlogs verwendet werden (wie z.B. JIRA, Mantis, Bugzilla, VersionOne o.ä.) Reporting-Funktionen an, die einen groben Überblick über die Änderungen des Backlogs liefern.

Eine andere Möglichkeit ist, das Product Backlog in regelmäßigen Abständen, z.B. nach jedem Sprint zu schätzen und so zumindest die Abweichungen für die Aufwände zu erhalten. Bei dieser Methode ist da-

rauf zu achten, dass sich der Aufwand für das Backlog u.U. auch durch Einflüsse wie die Zusammensetzung oder die zunehmende Erfahrung des Teams ändern kann (s. auch Abschnitt "Unsicherheitsfaktor Velocity"). Diese Einflüsse muss man bei der Bestimmung der Änderungsrate dann berücksichtigen.

Function Points

Die Änderungen eines Backlogs lassen sich auch mit Hilfe von Function Points bestimmen. Die Function-Point-Methode betrachtet Softwaresysteme aus Anwendersicht. Im Wesentlichen geht es darum, Software-Anwendungen als Systeme zu betrachten, in die Daten ein- und ausfließen und die Daten verwalten. Mit Hilfe weniger Regeln lässt sich damit der Umfang einer Software reproduzierbar messen. Wenn unterschiedliche Experten die Größe desselben Backlogs mit Function Points bestimmen, so weichen die Ergebnisse nur im Rahmen einer Messgenauigkeit von maximal 5% voneinander ab.

Da eine User Story mit Function Points nur dann bewertet werden kann, wenn sie ein Mindestmaß an Informationen bietet, lässt sich mit der Verwendung von Function Points als Nebeneffekt auch eine Mindestqualität der User Storys sicherstellen. Beispielsweise müssten User Storys für die Beschreibung einer Benutzerverwaltung alle Aktionen enthalten, die ein Anwender ausführen kann, also in etwa "Anlegen von Benutzern", "Zuordnen von Rechten" und "Löschen von Benutzern". Eine regelmäßige Vermessung des Backlogs mit der Function-Point-Methode liefert quantitative und – bei Verwendung entsprechender Tools – auch qualitative Aussagen über Veränderungen. So lässt sich feststellen, in welchem Bereich Anforderungen sich besonders häufig ändern und damit wahrscheinlich unklar sind (für nähere Informationen zur Function-Point-Analyse s. projektmagazin 20/2012 und 21/2012).

Änderungen einzeln betrachten

Die regelmäßige quantitative Vermessung des Backlogs kann nur einen Hinweis darüber liefern, ob sich das Backlog stärker ändert als erwartet. Um zu erkennen, ob und wie sich Änderungen auf den Gesamtaufwand auswirken, muss man sich die Änderungen im Einzelnen anschauen. Dabei sollten Fragen der folgenden Art beantwortet werden:

- Gibt es Änderungen bei Anforderungen, die bereits implementiert wurden?
  Die Umsetzung dieser Änderungen fällt in der Regel in den Bereich der Maintenance. In der Regel ist es sinnvoll, diese als solche zu kennzeichnen und dafür ein eigenes Budget bereitzustellen.
- Gibt es Anforderungen, die sich besonders häufig ändern?
  Bei diesen Anforderungen sollte geprüft werden, warum dies so ist; möglicherweise sind die Anforderungen noch unklar. Hier bietet es sich an, Workshops mit der Fachseite zu organisieren, um ein klares Bild zu erreichen.

Ab welcher Größe Änderungen kritisch für die Einhaltung des Gesamtaufwands sind, hängt sehr vom konkreten Projekt und der Projektphase ab. Zu Beginn des Projekts wird die Änderungsrate eher höher sein, mit zunehmender Projektdauer sollte sie abnehmen.

Transparenz

Die Hürde, einem Backlog eine neue User Story hinzuzufügen, ist in der Regel relativ gering. Die Anzahl der gewünschten Features einer Anwendung ist hingegen meistens groß. Daher wächst ein Backlog oft relativ schnell und erreicht eigentlich fast immer eine Größe, die sich mit dem geplanten Budget nicht mehr umsetzen lässt. Niedrig priorisierte User Storys können dann nicht mehr umgesetzt werden.

Um zu verhindern, dass das Projekt scheitert, weil sich notwendige User Storys aufgrund falscher Priorisierung nicht mehr im Rahmen des Budgets realisieren lassen, sollte allen Projektbeteiligten (insbesondere dem Auftraggeber), z.B. im Rahmen der Sprintreviews, klar gemacht werden, welche User Storys nach dem aktuellen Stand des Backlogs, dem verfügbaren Budget und der aktuellen Priorisierung wahrscheinlich nicht mehr umgesetzt werden können.

Dies sollte dadurch erfolgen, dass User Storys, die sich nicht mehr im Rahmen des aktuellen Budgets umsetzen lassen, deutlich als solche gekennzeichnet werden. Am einfachsten geschieht dies durch einen entsprechenden Status oder eine Releasenummer. Bei der Sprintplanung ist dann darauf zu achten, dass nur User Storys geplant werden, die dem aktuellen Release zugeordnet sind. Sollte eine Umpriorisierung ergeben, dass die User Story eines zukünftigen Releases in das aktuelle Release wandert, sind dabei die Auswirkungen auf den Gesamtaufwand zu berücksichtigen.

Fazit

In agilen Projekten spielt der Gesamtaufwand nicht unbedingt die Rolle wie in Projekten nach traditioneller Vorgehensweise. Daher muss zunächst geklärt werden, ob die Schätzung und Einhaltung des Gesamtaufwands notwendig für den Erfolg des Projekts sind. Ist dies der Fall, so sollte man auf Folgendes achten:

- Die Freiheit bei der Erstellung und Umpriorisierung des Backlogs ist eingeschränkt.
- Allen Beteiligten sollte bewusst sein, was innerhalb des vorgegebenen Aufwands umgesetzt werden soll.
- Man darf Sprintziele nicht mit Meilensteinen verwechseln, da deren Erreichung nicht unbedingt etwas über den Fertigstellungsgrad des Projekts aussagt.
- Agile Entwicklungsteams arbeiten – im Gegensatz zu traditionellen Projekten – bereits zu Beginn des Projekts mit und werden, falls zu wenig notwendige Anforderungen als User Storys verfügbar sind, mit der Umsetzung niedrig priorisierter User Storys ausgelastet. Dies kann den Gesamtaufwand erhöhen.
- Es ist wichtig zu wissen, wie stark sich das Backlog ändert und wie der aktuelle Fertigstellungsgrad des Projekts ist, um evtl. frühzeitig Gegenmaßnahmen zu treffen.

Literatur

- Bjoersne, Tord; Kostial, Ivan; Schmatz, Klaus-Dieter: Die Alternative zum Planning Poker: Das Team Estimation Game, projektmagazin 02/2013
- Gloski, Dorian; Schielein, Eva Maria: Function-Point-Analyse – die Methode und ihre Anwendung. Teil 1: Die Größe einer Software ermitteln, projektmagazin 20/2012
- Gloski, Dorian; Schielein, Eva Maria: Function-Point-Analyse – die Methode und ihre Anwendung. Teil 2: Kennzahlen, Effizienznachweis, Aufwandsschätzung, projektmagazin 21/2012
- Dr. Linssen, Oliver: Agile Aufwandsschätzung in Scrum. Planning Poker – Techniken, Erfahrungen und Empfehlungen, projektmagazin 10/2012
- Schulz, Kay: Aufwandsschätzung von IT-Projekten: die 11 wichtigsten Irrtümer, projektmagazin 13/2012
- Wirdemann, Ralf: Agiles Projektmanagement. Scrum – eine Einführung, projektmagazin 21/2009
- Wirdemann, Ralf: Agile Softwareentwicklung mit Scrum und User Stories, projektmagazin 02/2010
- Cohn, Mike: Alternative Scrum Release Burndown
- Jeffries, Ron: Estimation is Evil, http://pragprog.com/magazines/2013-02/estimation-is-evil

Bildnachweise

- Bild 1: Hochzeitstorte: commons.wikimedia.org/wiki/File:Pink_tiered_cake,_2009.jpg
- Bild 2: Vespa: commons.wikimedia.org/wiki/File:MODs_Vespa_Front.jpg
- Bild 3: Ruine: commons.wikimedia.org/wiki/File:Beelitz_Heilst%C3%A4tten_-jha-_590148686391.jpeg
- Bild 6: Springbrunnen: commons.wikimedia.org/wiki/File:Therefore_with_joy_you_will_draw_water_from_the_wells_of_salvation.JPG

Unternehmensweiter Wissenstransfer durch User Stories

5 Probleme bei agilen Aufwandsschätzungen

Management Summary

- Der Großteil der heutigen Softwareentwicklungsprojekte dauert länger und kostet mehr als ursprünglich geplant.
- Fehlerhafte Aufwandsschätzungen sind oft der Grund für diese Abweichungen. In der Praxis treten bei agilen Aufwandsschätzungen vorrangig fünf Probleme auf.
- Die schlechte Qualität von User Stories führt zu Missinterpretationen und Fehleinschätzungen.
- Mangelnde Disziplin bei der Ausführung agiler Methoden erschweren eine verlässliche Planung.
- Häufig sind agile Aufwandsschätzungen mehr ein "best guess" als erfahrungsbasiert.
- Die Abwesenheit von Kollegen und wechselnde Teamkonstellationen hemmen den Informationstransfer im Unternehmen.
- Ein fehlender teamübergreifender Austausch verhindert ein unternehmensweites Lernen. User Stories können jedoch als Medium zum Informationstransfer genutzt werden und Silos in großen Unternehmen aufbrechen.

Julian Mayer
Associate Management Consultant bei der diconium strategy GmbH

Stephan Ahrweiler
Senior Manager bei der diconium strategy GmbH

Seit Beginn der Softwareentwicklung und der Veröffentlichung des agilen Manifests 2001 findet die agile Bewegung immer mehr Anhänger. Das iterative Projektvorgehen, bei dem aus vorangegangenen Perioden gelernt wird, soll schnellere und kundenzentrierte Entwicklungszyklen ermöglichen. Laut einer Studie von McKinsey und der University of Oxford überschreitet jedoch ein Softwareentwicklungsprojekt (>15 Mio. USD) – trotz agiler Arbeitsweisen – das Budget um durchschnittlich 66% und den zeitlich definierten Projektrahmen um 33%. Bei den etwa 5400 untersuchten IT-Projekten wurde eine Überschreitung des geplanten Gesamtbudgets von insgesamt 66 Mrd. USD ermittelt. Dieser Wert ist höher als das luxemburgische Bruttoinlandsprodukt.

Gemäß dem "digitalen Darwinismus" überleben nur die Unternehmen, die es schaffen, sich schnell und effizient an technologische Veränderungen anzupassen. Werden Softwareprojekte zukünftig nicht effizienter geplant und durchgeführt, kann ein Unternehmen dadurch die Wettbewerbsfähigkeit verlieren.

Warum überschreiten so viele IT-Projekte Budget und Projektrahmen?

Die Ursachen hierfür sind vielfältig: Unrealistische Projektdimensionierung, größere ungeplante Entwicklungsprobleme, fehlende Ressourcen und Kompetenzen, Missmanagement oder die mangelhafte Ausführung agiler Methoden sind nur einige der möglichen Gründe. Letzteres stellt gerade in großen, bereichsübergreifenden agilen Projekten einen nicht zu vernachlässigenden Problemfaktor dar.

Bei vielen agilen Frameworks ist das **Schätzen von Aufwänden** ein integraler Bestandteil der iterativen Planung. Die Frameworks helfen, eine hohe Anzahl agiler Teams im Unternehmen zu koordinieren. Dabei setzen die Teams Aufgabenpakete, sog. **Backlog Items**, in zweiwöchigen Zyklen (Sprints) um. Um diese Sprints zu planen, benötigt es Schätzungen über den voraussichtlichen Arbeitsaufwand der einzelnen Backlog Items. In unserem Fall verwendeten wir **User Stories als Backlog Items**. Als übergreifendes Organisations- und Verwaltungstool werden häufig Tools wie Jira (Atlassian) verwendet. Dort werden alle Backlog Items inklusive Aufwandsschätzungen der einzelnen Teams digital erfasst.

Die Schätzmethoden Team Estimation Game und Planning Poker

Zur Schätzung von Aufwänden in agilen Teams gibt es verschiedene Methoden. Zwei sehr weit verbreitete sind das Estimation Game und Planning Poker. Bei beiden wird nicht in absoluten Einheiten, wie z.B. Personenstunden oder -tagen geschätzt, sondern in relativen Einheiten. Hierbei werden lediglich die Relationen zwischen den einzelnen Backlog Items betrachtet. Damit dies gelingt, muss das Team zunächst eine Referenzgröße definieren bzw. ein einheitliches Verständnis über relative Größen schaffen.

Die Vorteile von Team Estimation Game und Planning Poker …

Der Vorteil der beiden Methoden ist die geringe Komplexität der Schätzung, da keine exakten zeitlichen Werte festgelegt werden müssen. Zudem wird keine Scheingenauigkeit suggeriert, da Aufgabenumfänge häufig nicht stundengenau bezifferbar sind. Beim Estimation Game werden gerne T-Shirt-Größen oder Tierarten als Größeneinheit verwendet. Eine User Story der Größe M ist z.B. größer als eine Story der Größe S, jedoch kleiner als eine der Größe L. In der Regel formuliert der Product Owner (PO) die User Stories und spezifiziert die Anforderungen an das zu entwickelnde Inkrement. Die Aufwandsschätzung erfolgt im Team (i. d. R. Product Owner, Scrum Master, Entwicklungsteam), wobei lediglich die Personen schätzen sollten, die die User Stories schlussendlich umsetzen. Während der Schätzrunden werden alle User Stories diskutiert und geschätzt.

… und die Nachteile

Studien zufolge liegt die durchschnittliche Abweichung von geschätzten und tatsächlichen Aufwänden bei 20% bis 30% (Schweighöfer, 2016). Darüber hinaus werden 26% der Aufgaben erst während oder nach dem eigentlichen Sprint identifiziert (Grapenthin, 2014). Diese enormen Differenzen zeigen, warum gerade große Softwareentwicklungsprojekte, mit mehreren tausend Backlog Items pro Jahr, den finanziellen und zeitlichen Rahmen häufig sprengen.

Die Qualität einer Aufwandsschätzung ist abhängig von Kompetenz und Erfahrung der Teammitglieder und dem Austausch sowohl im Team als auch teamübergreifend. Außerdem spielt die disziplinierte und korrekte Anwendung der Methodik eine maßgebliche Rolle. In den meisten Fällen werden agile Aufwandsschätzungen nicht systematisch nach dem Lean-Gedanken optimiert, der u.a. SAFe (Scaled Agile Framework) zugrunde liegt. Der digital archivierte Datenschatz in Form von Backlog Items wird meist nicht genutzt, um den Planungsprozess bzw. das Planungsergebnis zu optimieren und aus vorherigen Sprints zu lernen.

Im Rahmen einer Projektstudie für einen führenden Automobilkonzern untersuchten wir, welche Probleme und Hindernisse bei agilen Aufwandsschätzungen auftreten und wie diesen auf Basis bestehender Daten und mittels Einsatzes von Technologie begegnet werden kann. Über 100 agil arbeitende Teams schätzten und archivierten zum Zeitpunkt der Untersuchungen jeweils mehrere hundert Backlog Items pro Jahr. Aus der Studie ergaben sich folgende fünf Hauptoptimierungspotenziale (Bild 1):

Bild 1: Die fünf häufigsten Probleme bei agilen Aufwandsschätzungen

Die 5 häufigsten Fehler bei agilen Aufwandsschätzungen

1. Geringe Qualität der Backlog Items

Ein häufiger Grund für Schätzdifferenzen ist die schlechte Qualität von Backlog Items. Diese kann zu Missinterpretationen und in der Folge zu Verzögerungen führen. Dabei gibt es vorrangig drei Faktoren schlechter Item-Qualität:

Unpräzise Formulierungen

User Stories sind das zentrale Instrument zur Vermittlung der Anforderungen eines Produkts aus Kundensicht an die Entwickler. Daher sollten sie ohne weitere Erläuterungen klar und deutlich darlegen, was das zu entwickelnde Feature leisten muss. Häufig enthalten diese jedoch Ausdrücke wie z.B. "alle, jeder, immer, nie, manchmal". Diese Ausdrücke machen User Stories unpräzise und lassen einen großen Interpretationsspielraum für den Leser. Lautet eine Formulierung in der User Story z.B.: "Integration aller 3rd Party Schnittstellen", weiß das verantwortliche Umsetzungsteam nicht genau, welche Schnittstellen gemeint sind und integriert möglicherweise zu viele oder die falschen. Besser hingegen sind spezifische und präzisierende Formulierungen wie: "Integration der in Partner Conference 3.4 definierten 3rd Party Schnittstellen für das MVP".

Uneinheitliche Syntax

Auch die Formulierung einer User Story erfolgt häufig nicht nach einem einheitlichen Schema. Vielmehr wird die Story je nach Schreibstil des Verfassers formuliert. Folgendes bewährtes Muster hat sich für User Stories etabliert:

"Als [Rolle] möchte ich [Anspruch], so dass [Grund für den Anspruch]".

Ein Beispiel dafür ist: "Als [Kunde] möchte ich [das erworbene Produkt bewerten können], so dass [ich Feedback und Kritik äußern kann]". Diese Formulierungsweise hilft dem Entwicklungsteam, sich besser in die entsprechende Rolle des Benutzers hineinzuversetzen und passgenauere Lösungen zu entwickeln. Zudem kann der PO das Backlog durch diese Struktur einfacher priorisieren, da diese einheitliche Syntax der User Stories den Kundenwunsch und dessen Beweggrund in den Vordergrund stellt.

Umfang und Spezifikationsgrad

Ein weiteres Problem stellt der Umfang und Detailgrad der User Stories dar. Häufig werden Anforderungen nicht trennscharf definiert, was die Aufwandsschätzung deutlich erschwert. Sind die Stories nicht in sich abgeschlossen oder enthalten Teile unterschiedlicher Stories, kann dies zu Fehleinschätzungen führen. Zudem werden Backlog Items in vielen Fällen nur sehr rudimentär beschrieben und viele Fragen offengelassen. Eine präzise Beschreibung inklusive Definition of Done, die definiert, welche Kriterien erfüllt sein müssen, damit eine Story als abgeschlossen gilt, sind deshalb Grundvoraussetzung für eine realistische Aufwandsschätzung.

2. Mangelnde Disziplin der Beteiligten

Eine weitere Herausforderung von Aufwandsschätzungen ist die teils fehlende Disziplin der Beteiligten. Oft wird nicht kontinuierlich bei jeder User Story geschätzt. Außerdem werden Schätzungen häufig nicht konsistent und unter methodisch korrekten Rahmenbedingungen durchgeführt. Dazu zählt z.B. die konstante Verwendung einer definierten Schätzmethode und deren fehlerfreie Durchführung (z.B.

Planning Poker mit objektivem Moderator). Geschieht das nicht, erschwert es zum einen die Vergleichbarkeit von Schätzungen und trägt zum anderen dazu bei, dass der Lerneffekt der schätzenden Mitarbeiter nur bedingt eintritt. Deshalb ist es **Aufgabe des Scrum Masters, die methodisch korrekte Durchführung der Aufwandsschätzungen bei jedem Schätztermin einzufordern**.

3. Fehlende Erfahrung

In der Projektstudie gab ein Großteil der Experten an, dass Aufwandsschätzungen häufig geraten werden. Es kommt durchaus vor, dass sich die schätzenden Beteiligten mehr auf ihr Bauchgefühl verlassen als auf tatsächliche Erfahrungen. Besonders bei unbekannten Technologien oder fehlender Expertise in einem neuen Themenbereich ist es oft nur ein "best guess" der Beteiligten. In diesem Fall sind die abgegebenen Prognosen tendenziell unpräziser.

Eine Lösung für dieses Problem sind **sog. Bottom-Up-Betrachtungen**. Dabei werden die Backlog Items in kleinstmögliche Subtasks unterteilt, da sich kleinere Aufgabenpakte leichter schätzen lassen. Anschließend werden alle Aufwände aufaddiert.

Zudem ist ein reger Austausch sowohl im Team als auch teamübergreifend notwendig, bei dem Erfahrungen unter den Kollegen weitergegeben werden können. Hierdurch entstehen gerade in großen Unternehmen mit vielen Entwicklerteams wertvolle Synergien.

4. Abwesenheit von Schlüsselpersonen und wechselnde Teammitglieder

Ein weiteres Hindernis im Rahmen von Aufwandsschätzungen ist, dass Schlüsselbeteiligte nicht an jedem Schätzprozess teilnehmen können. Dies kann zu Fehleinschätzungen führen. Gründe für die Abwesenheit sind meist persönliche oder betriebliche Anlässe wie Urlaub, Krankheit, zeitgleiche Termine oder Rollen in Doppelverantwortung. Hinzu kommen standortübergreifende Teambesetzungen, die den Austausch erschweren. Fehlt die einzige Person mit Expertise und Erfahrung in einem gewissen Themenfeld, werden die Schätzungen diesbezüglich weniger präzise.

Zusätzlich werden Teams je nach Situation und Thematik häufig neuformiert. Wechselnde Teammitglieder bedeuten auch sich verändernde Erfahrungen und Leistungsniveaus, die sich wiederum auf die Genauigkeit der Aufwandsschätzungen auswirken. Auch hier ist das Hauptproblem, dass kein umfänglicher und systematischer Informationsfluss von erfahrenen zu weniger erfahrenden Entwicklern und von alten hin zu neuen Teammitgliedern ermöglicht wird.

Umso wichtiger ist es, allen Beteiligten die **Relevanz und Wichtigkeit der Aufwandsschätzungen klar zu machen**. Eine präzisere Sprintplanung beutet eine weniger hektische Umsetzungsphase. Dies ist Aufgabe des Scrum Masters. Gelingt das nicht, sollte der Srum Master einen strukturierten Know-how-Transfer im Team zu den jeweiligen Themen aktiv fördern, z.B. durch **Brown Bag Sessions**. Bei einer Brown Bag Session treffen sich die Beteiligten zu einer informellen Mittagspause, bei der jeder sein Essen selbst (z.B. in einer braunen Tüte) mitbringt. Bei diesem zwanglosen, informellen Treffen tauschen sich die Mitarbeiter über beliebige Themen oder eben zu einem bestimmten Thema aus.

5. Mangelnder Wissenstransfer

Gerade in großen Organisationen bleibt der teamübergreifende Wissenstransfer, nicht nur in Bezug auf agile Aufwandsschätzungen, oft auf der Strecke. Lediglich innerhalb eines Teams (sofern konstant) kommt es im Rahmen der Sprint Retrospektiven oder Code Reviews unter Entwicklern zu einem strukturierten Austausch. In Konzernen mit dutzenden agilen Teams gestaltet sich ein systematisches Nutzbarmachen der Erfahrungen und Problemstellungen aus vorherigen User Stories äußerst schwierig.

Gerade bei einer hohen Anzahl an Entwicklerteams ist die Gefahr groß, dass sich Problemstellungen wiederholen, jedoch unterschiedliche Teams nichts voneinander wissen. Folglich fließen Erkenntnisse bereits abgeschlossener User Stories nur implizit in den Erfahrungsschatz der beteiligten Teammitglieder ein, statt aktiv und systematisch von allen genutzt zu werden. So steigt die Lernkurve in Bezug auf Aufwandsschätzungen unternehmensweit nur langsam und Synergien bleiben ungenutzt. Ein Großteil des Optimierungspotenzials liegt deshalb im teamübergreifenden Know-how-Transfer.

Wenn das Unternehmen wüsste, was es weiß – die User Story als Transfermedium

Dieses Transferproblem lässt sich mittels Natural Language Processing (NLP) lösen. Die Lösung basiert auf folgendem technischen Konzept (Bild 2):

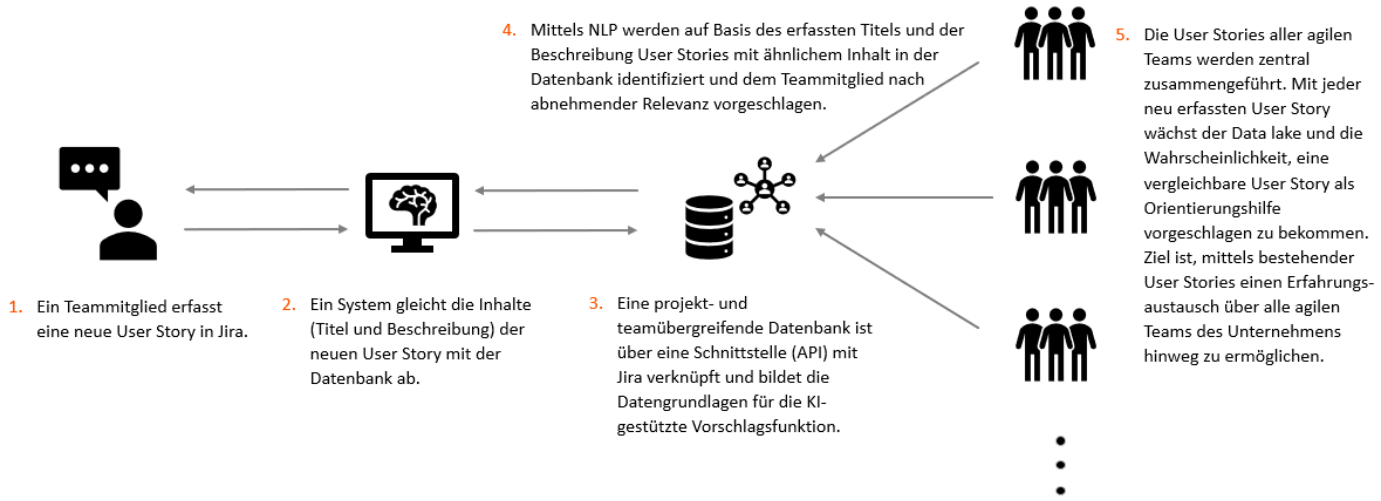


Bild 2: Das Konzept für ein KI-basiertes Vorschlagssystem für ähnliche User Stories

Bild 2 skizziert, wie man die schätzenden Personen bei agilen Aufwandsschätzungen unterstützen kann, indem bereits bestehendes Wissen schnell und systematisch nutzbar gemacht wird. Ziel der Lösung ist, durch Wissenstransfer sowohl den Schätzprozess als auch die Genauigkeit agiler Aufwandsschätzungen zu optimieren. Dafür verwende ich Erfahrungen aus früheren Projekten und Teams, vorhandenes Wissen zu kollektivieren und Silos zwischen den Entwicklungseinheiten aufzubrechen.

Vermeiden Sie den Silo-Tod!

Die Grundlage der skizzierten Lösung bilden die Backlog Items. In einem SAFe-Konstrukt mit einer Vielzahl an Entwicklerteams wiederholen sich gleichartige Items häufig. Ein Erfahrungsaustausch zwischen den einzelnen Teams und über Projekte hinweg findet in den meisten Fällen aber nicht statt. Jedoch werden alte Backlog Items der einzelnen Teams in Tools wie Jira archiviert. Nach Umsetzung der Items werden die Daten allerdings nicht mehr systematisch genutzt, um daraus zu lernen. **Die Idee ist, dem Anwender automatisiert lexikalisch und semantisch ähnliche Backlog Items vorzuschlagen.**

Der Fokus liegt dabei auf den User Stories (Bild 3). Nach Eingabe einer User Story durch den Anwender vergleicht das System den Titel und die Beschreibung der User Story mit allen anderen User Stories der Datenbank. Mittels NLP werden dem Anwender ähnliche User Stories nach abnehmender Relevanz vorgeschlagen. Das Informationsabrufsystem funktioniert in seiner Basisfunktion wie eine intelligente Suchmaschine. Die Suchanfrage ist im vorliegenden Fall jedoch keine Suche nach Inhalten im Internet, sondern eine neuverfasste User Story, die als Basis dient, ähnliche und verwandte User Stories in einer Datenbank zu identifizieren. Alle User Stories der unterschiedlichen Teams und Projekte werden hierfür zentral in einem Data Lake mittels Jira-Schnittstelle (API) gesammelt. Somit steigt der Nutzen des Systems mit jedem weiteren Backlog-Eintrag.

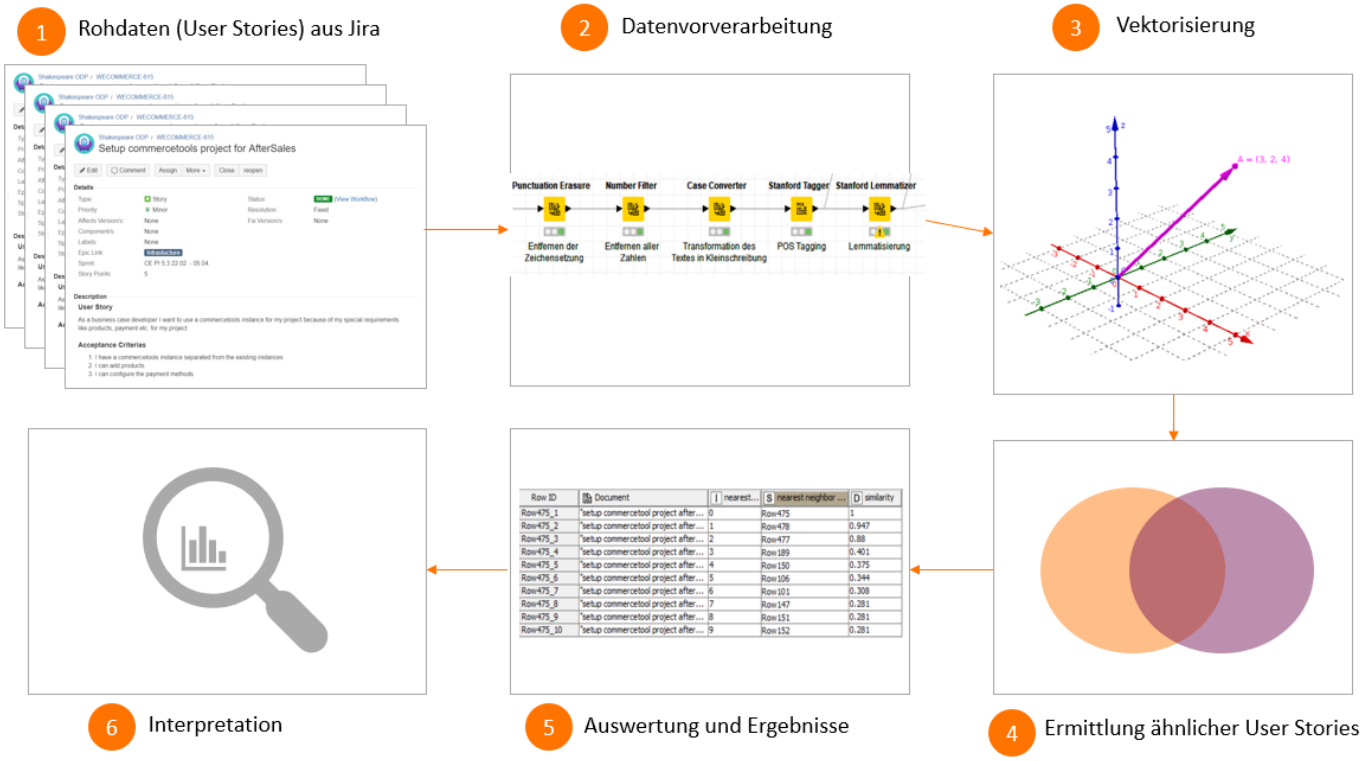


Bild 3: Durch systematischen Wissenstransfer mittels Natural Language Processing (NLP) soll der Schätzprozess und die Genauigkeit agiler Schätzungen verbessert werden

Werden einem Benutzer gleichartige User Stories vorgeschlagen, kann er sich daran orientieren und z.B. folgende Informationen nutzen:

- Geschätzte Aufwände anderer Teams für vergleichbare Tätigkeiten
- Umfang und Bestandteile vergleichbarer User-Story-Beschreibungen
- Aktivitätshistorie mit potenziellen Hinweisen auf Anpassungen und Probleme bei der Bearbeitung der Referenzstory. Diese Informationen kann der Anwender nutzen, um die Qualität seiner User Story zu steigern, potenzielle Schwächen und blinde Flecken zu identifizieren und so realistischere Aufwandsschätzungen im Team zu treffen.

Exaktere Aufwandsschätzungen durch unternehmensweiten Wissenstransfer

Für die Entwicklung dieser Lösung müssen Sie zunächst die Voraussetzungen für einen übergreifenden Datenpool für User Stories schaffen. Das bedeutet, dass falls noch nicht geschehen, alle User Stories digital und einheitlich erfasst werden müssen. Diese Daten müssen automatisch und regelmäßig über eine Schnittstelle in einem gemeinsamen Data Lake gesammelt werden. Dazu sind eine Datenfreigabe- und Zugriffsregelung im Unternehmen Grundvoraussetzung.

Hierin liegt auch die Hauptherausforderung des Vorhabens: Je nach Ansatz ist bei der Identifikation ähnlicher Textelemente mehr oder weniger Expertise im Bereich Data Science erforderlich. Ist der Algorithmus selbstlernend programmiert, verbessern sich die Ergebnisse mit jeder weiteren User Story. Um die User Experience zu verbessern, ist eine Integration eines solchen Tools in das Verwaltungstool der User Stories ratsam. Wird das Tool intern gehostet, muss mindestens ein Key User (unsere Empfehlung sind zwei bis drei) definiert werden, der bei Problemen und Fragen zur Verfügung steht.

Der Vorteil dieser Lösung liegt darin, dass unabhängig von Projekt und Team Erfahrungen über das Medium "User Story" geteilt werden können. Dies steigert die Effizienz von Aufwandsschätzungen und minimiert das nicht unerhebliche Planungsrisiko bei agilen Großprojekten.

Ausblick: KI und agile Arbeitsweisen in Kombination

In einer weiteren Ausbaustufe können mittels KI nicht nur ähnliche User Stories identifiziert werden, sondern auch Aufwände algorithmisch geschätzt werden. Eine vollständige Substitution von agilen Aufwandsschätzungen durch KI-Methoden ist allerdings aufgrund uneinheitlicher Messgrößen (z. B. T-Shirt-Größen vs. Story Points) mit Vorsicht zu genießen. Außerdem entfallen hierdurch die wichtigen Diskussionen im Team über Unklarheiten von Backlog Items. Hinzu kommt die demotivierende Komponente einer vollautomatischen Schätzung für die Entwickler, da sie so kein Mitspracherecht in Bezug auf ihre Aufwandsschätzungen mehr haben.

Beide Disziplinen, sowohl KI als auch agile Arbeitsweisen, bieten die Chance, in ihrer Kombination Unternehmen zu noch effektiveren und effizienteren Leistungen zu befähigen. Dabei sollte technologischen Veränderungen aus unserer Sicht nicht ängstlich und planlos begegnet werden, sondern strukturiert und anwenderorientiert. Denn eine schnelle Anpassungsfähigkeit bedeutet in der heutigen Zeit Überlebensfähigkeit.

Literatur

- Agiles Manifest: https://agilemanifesto.org/iso/de/manifesto.html, 2001
- Bloch, Michael; Blumberg, Sven; Laartz, Jürgen: Delivering large-scale IT projects on time, on budget, and on value, McKinsey Insights; 2012
- Grapenthin, S.; Poggel, S.; Book, M.; Gruhn, V.: Facilitating Task Breakdown in Sprint Planning Meeting 2 with an Interaction Room: An Experience Report; in: 40th EUROMICRO Conference; 2014
- Scaled Agile Inc.: SAFe® for Lean Enterprises, https://www.scaledagileframework.com/
- Schweighofer, T.; Kline, A.; Pavlic, L.; Hericko, M.: How is Effort Estimated in Agile Software Development Projects?; SQAMIA, 2016

Agile Aufwandsschätzung in Scrum

Planning Poker – Techniken, Erfahrungen und Empfehlungen

Dr. Oliver Linssen
Diplom-Wirtschaftswissenschaftler, Geschäftsführer der Liantis GmbH

Unzuverlässige Aufwandsschätzungen in der Softwareentwicklung sind eher der Normalfall als die Ausnahme. Um diesem Problem zu begegnen, sind im Lauf der letzten Jahrzehnte zahlreiche Verfahren zur Aufwandsschätzung in der Software-Entwicklung entstanden. Dabei ist in agilen Projekten das sog. Planungspoker (engl. "Planning Poker") weit verbreitet.

Dieser Beitrag stellt die Techniken und Verfahren vor, die für das Schätzen in Scrum mit Planning Poker erforderlich sind. Darüber hinaus erhalten Sie Empfehlungen, worauf Sie beim Einsatz von Planning Poker in der Praxis achten sollten.

Ziel des Planning Pokers

Der Product Owner erhält durch die Schätzung des Teams Aussagen darüber, wie "teuer" die Realisierung der einzelnen User Storys ist und verwendet diese Schätzungen, um die Einträge im Product Backlog zu priorisieren. Das Team erhält durch die Schätzung eine realistische Grundlage für die Planung der in den Sprints zu realisierenden Features.

Für beide Seiten ist es von Vorteil, dass in Scrum die Schätzung regelmäßig auf der Grundlage neuer Erkenntnisse überarbeitet wird. Dies ist eine wichtige Grundlage dafür, was der Betriebswirtschaftler auch als "rollierende Planung" bezeichnet: Eine gleitende Planung der Perioden in der Zukunft, wobei die nächste Periode konkret und die folgenden Perioden grob geplant werden.

Allgemeines zum Ablauf des Planungspokers

Aufwandsschätzungen finden in Scrum nicht nur zu Beginn des Projekts, sondern während des gesamten Projektverlaufs statt. Vor Beginn des ersten Sprints wird in einer oder mehreren Sitzungen der gesamte Inhalt des Product Backlogs geschätzt. Da sich das Product Backlog während des gesamten Projekts verändert, wird auch in jedem Sprint geschätzt. Dadurch fließen die aktuellen Erfahrungen über Produktivität des Teams, Komplexität der Technik und des Anwendungsgebiets und andere Randbedingungen in die Schätzung ein.

An der Schätzung sind der Product Owner, das Team und der Scrum Master beteiligt. Der Product Owner erläutert die Anforderungen und beantwortet Fragen des Teams in Bezug auf die Anforderungen. Das Team schätzt den Aufwand, da es später ja auch für die Realisierung verantwortlich ist. Der Scrum Master moderiert die Veranstaltung. Die Schätzung ist eine reine

Teamschätzung, d.h. der Product Owner und der Scrum Master schätzen selber nicht. Da alle Rollen bei der Schätzung beteiligt sind, ist das Verfahren sehr transparent.

Die Schätzrunden sind ermüdend und deshalb zeitlich begrenzt: Während man durchaus die ganze Nacht pokert, sollten die einzelnen "Runden" beim Planungspoker nur etwa zwei Stunden dauern (Pichler, 2008).

Dass die Entwickler den Aufwand für die Entwicklung schätzen, ist keinesfalls eine Erfindung der agilen Softwareentwicklung. Die Anwendung dieses Prinzips erfolgt auch in Projekten, die nicht nach einem agilen Vorgehen ablaufen – denn die ausführenden Stellen geben immer noch die präzisesten Werte bei der Aufwandsschätzung ab.

User Storys

Im Product Backlog werden die funktionalen und nicht-funktionalen Anforderungen in Form von sog. User Storys erfasst (vgl. Cohn, 2004 sowie "Agile Softwareentwicklung mit Scrum und User Stories", Projekt Magazin 02/2010). Eine User Story beschreibt ein Feature, das eine zukünftige Anwendung enthalten soll. Weit verbreitet ist die Verwendung von Schablonen bei der Formulierung von User Storys. Häufig wird hierbei die Form verwendet: Als <Rolle des Anwenders> möchte ich <die gewünschte Möglichkeit>, sodass <Geschäftswert>. Die User Story wird häufig um die Beschreibung von Akzeptanztests ergänzt, mit denen die Implementierung der User Story getestet werden soll.

Story Points

Beim Planning Poker wird nicht geschätzt, wie hoch der Aufwand zur Realisierung der User Storys (in Zeiteinheiten) sein wird. Stattdessen schätzt man die relative Größe von User Storys zueinander. Die hierfür verwendete Maßeinheit wird als "Story Point" bezeichnet. Schätzt man 1 Story Point für die Realisierung einer User Story A und 3 Story Points für die Realisierung einer User Story B, bedeutet dies, dass der Zeitaufwand für die Realisierung von B vermutlich drei mal so groß sein wird wie für die Realisierung von A.

Story Points sind eine relative Größe, die im Team durch Konsens festgelegt wird. Das bedeutet, dass unterschiedliche Teams mit unterschiedlich großen Story Points arbeiten. Über Teamgrenzen hinweg ist ein Vergleich von Story Points nicht möglich.

Fibonacci-Zahlen

Im Sprint Planning Meeting schätzt jedes Teammitglied den Aufwand so, als ob er das Feature selber realisieren würde. Zum Schätzen hat jeder Teilnehmer einen Satz Karten, auf denen eine Fibonacci-Zahl zwischen 1 und einem vorher festgelegten Höchstwert (z.B. 21) steht (1, 2, 3, 5, 8, 13, 21). Fibonacci-Zahlen werden gebildet, in dem man die Summe der beiden vorhergehenden Fibbonacci-Zahlen addiert. Sie werden deshalb gerne verwendet, weil die Abstände zwischen den einzelnen Werten immer größer werden. Damit soll den Schätzern deutlich bleiben,

dass der Aufwand zur Realisierung einer Story mit dem Wert 21 im Vergleich zu einer Story mit dem Wert 13 sehr viel größer ist als zwischen zwei Storys mit den Werten 3 und 5.

Die Verwendung von Fibonacci-Zahlen ist nur eine Möglichkeit, die Komplexität der User Storys zu schätzen (Cohn, 2005). Alternativ könnten z.B. auch die Werte 1, 2, 4, und 8 verwendet werden. Auch kleinste Anforderungen sollten aber immer den Wert 1 erhalten, da die Realisierung einer Anforderung nie ohne Aufwand erfolgen kann. Schätzungen zwischen den vereinbarten Werten einer Skala werden nicht zugelassen. Für weniger konkrete User Storys am unteren Ende des Product Backlog sind auch höhere Werte (z. B. 100) zugelassen. Diese User Storys sind häufig noch vage formuliert und ihre Realisierung steht erst in späteren Sprints an. Cohn schlägt für solche Storys den Begriff "Epos" vor.

Kalibrierung

Damit die Schätzungen der Teammitglieder vergleichbar sind, muss man die Schätzung justieren. Hierfür einigt man sich auf eine (oder mehrere) User Storys, deren Realisierungsaufwand man als "mittelgroß" einschätzt. Dieser Referenzstory ordnet man dann z.B. den Wert 5 zu. Die übrigen Größen werden im Vergleich zu diesem Wert festgelegt.

Empfehlung für die Praxis

Die Maßeinheit Story Point sollte im Team nicht zu groß gewählt werden. Wenn ein Story Point beispielsweise etwa drei Personentagen entspricht, dann benötigt die Realisierung einer User Story mittlerer Komplexität (die also den Wert 5 erhalten hat) schon 15 Personentage. Eine große User Story (die den Wert 13 erhalten hat) benötigt zur Realisierung bereits 39 Personentage. Abgesehen davon, dass bei solchen Größen der Arbeitsfortschritt (Fertigstellungsgrad) im weiteren Projektverlauf nicht mehr genau genug bestimmt werden kann, lässt bei solchen Größen die Schätzgenauigkeit erheblich nach. Eine – wenn auch pauschale – Empfehlung ist, dass eine User Story mit dem Wert 1 an einen Personentag realisiert wird. Eine "mittelgroße" User Story würde also einem Aufwand entsprechen, den man in 5 Personentagen realisieren kann.

Aufwandsschätzung mit Planning Poker

Das Team geht im Sprint Planning Meeting alle zu schätzenden User Storys durch. Am Anfang muss das Team alle vorhandenen Storys schätzen, in den folgenden Sprints werden in der Regel nur neue Storys geschätzt sowie Storys, bei denen das Team der Meinung ist, dass der bisher geschätzte Aufwand nicht mehr korrekt ist. Alle Teammitglieder halten gleichzeitig die Karte mit den geschätzten Aufwänden für die Realisierung hoch (oder legen sie hin) und vergleichen ihre Ergebnisse. Nun sind drei Fälle zu unterscheiden (Schwaber, 2010):

1. Wenn alle Teammitglieder die gleiche Schätzung abgegeben haben, übernimmt man diesen Wert für die User Story.

2. Wenn die Schätzwerte der Teammitglieder nahe beieinander liegen, übernimmt man den größeren Wert als Schätzung für die User Story. Beispiel: Vier Teilnehmer schätzen 3 Story Points, drei Teilnehmer schätzen 5 Story Points. Als Ergebnis werden 5 Story Points für die Schätzung übernommen. Damit ist man auf der sicheren Seite und hat auch die Schätzung von Teammitgliedern berücksichtigt, die nach eigener Einschätzung mehr Zeit für die Realisierung benötigen würden.
3. Liegen die Schätzwerte zu weit auseinander, diskutiert man die abweichenden Ergebnisse. Nach der Diskussion pokert man den Aufwand erneut und vergleicht wieder die Ergebnisse. Beispiel: Die Bandbreite der Schätzungen reicht von 2 Story Points bis 8 Story Points. Über den Aufwand muss diskutiert werden, und anschließend schätzen alle Team-Mitglieder mit dem neuen Hintergrundwissen die Story erneut. In der Regel nähern sich die Ergebnisse schon bei der zweiten Schätzung deutlich an.
4. Eine Diskussion bei stark voneinander abweichenden Werten ist absolut notwendig, um die verschiedenen Sichtweisen der Teilnehmer einzuholen und deren Erfahrungen in die Schätzung einfließen zu lassen. Es ist nicht zu empfehlen, automatisch den höheren Wert anzunehmen. Denn dies verschiebt bei einer größeren Anzahl von User Storys den geschätzten Gesamtaufwand u.U. zu sehr nach oben. Eher ist es sinnvoll, über alle großen Abweichungen zu diskutieren.

In der Regel erzielt man im Team eine Einigung über den geschätzten Aufwand. Ist dies nicht möglich, gibt es zwei Möglichkeiten: Entweder man zerlegt die Story in kleinere Storys, die sich besser schätzen lassen oder das Team muss sich mehr Wissen über den Inhalt der Story aneignen, damit es sie präziser schätzen kann. Diese Recherche wird als "Spike" (der Begriff stammt aus dem Extreme Programming) bezeichnet. Im Rahmen eines Spikes könnte z.B. ein Prototyp entwickelt werden.

Personenspezifische Schätzung

Bei Scrum schätzt jeder den Aufwand so, als ob er die User Story selber realisieren würde. Dies ist vernünftig, weil er möglicherweise im Sprint tatsächlich die Realisierung der User Story übernimmt. Die Produktivität und die Qualifikation der Entwickler in einem Team ist aber in der Regel unterschiedlich. Wenn man im Planungspoker dazu neigt, sich an den höheren Schätzwerten zu orientieren, wird der geschätzte Aufwand tendenziell wesentlich höher ausfallen als der tatsächlich realistische – dagegen gibt es kein Patentrezept. Im besten Fall kann das Team intern die Aufgaben so verteilen, dass die einzelnen Teammitglieder möglichst selten Aufgaben erhalten, mit denen sie sich nicht gut auskennen.

Dominante Personen

Sehr erfahrene Entwickler, Meinungsführer, Experten oder dominante Persönlichkeiten, können das Ergebnis der Schätzungen im Planungspoker wesentlich beeinflussen. Natürlich kann es sinnvoll sein, sich an der Schätzung von Kollegen zu orientieren, wenn diese z.B. eine große Erfahrung mitbringen. Negative Einflüsse entstehen allerdings, wenn ein Teammitglied statt hoher Kompetenz einfach nur die Meinungshoheit hat und dadurch die Schätzung in eine bestimmte Richtung (nach oben oder unten) beeinflusst. Dem Team sollte diese Gefahr deutlich sein bzw.

der Scrum Master muss hierauf hinweisen. Letztendlich muss das Team selber darauf achten, dass die Meinung aller Teammitglieder das gleiche Gewicht hat; die Schätzung des Aufwands ist eine Teamentscheidung, die auch vom gesamten Team getragen werden muss.

Magic Number

Der höchste der vereinbarten Werte (hier: 21) hat eine besondere Bedeutung. Schätzt jemand den Wert mit 21, bedeutet dies nicht, dass er den 21-fachen Aufwand von 1 vermutet. Vielmehr ist damit gemeint, dass er den Aufwand für nicht genau abschätzbar, sondern sehr groß hält. Solche Storys müssen zerlegt werden oder das Team führt einen Spike durch, wenn die Story sich nicht weiter zerlegen lässt. (s.o.)

Sprintkapazität

Die Umrechnung der Story Points in Personentage basiert auf der Erfahrung, wie viele Story Points im Durchschnitt in einem Sprint in bisherigen Projekten (am besten vom gleichen Team) umgesetzt wurden. Dies wird als "Sprintkapazität" (auch Team-Kapazität) bezeichnet. Bei der Berechnung der Sprintkapazität muss berücksichtigt werden, wie viel Arbeitszeit tatsächlich für die Arbeit im Projekt zur Verfügung steht. Wochenenden, Feiertage, Besprechungen sind in gleicher Weise abzuziehen wie Zeiten für krankheitsbedingten Ausfall oder Reisezeiten, die als Arbeitszeiten gelten. Die Sprintkapazität ist also die Netto-Arbeitszeit, die dem Team tatsächlich zur Realisierung der User Storys im Sprint zur Verfügung steht. Hier ist die Verwendung eines realistischen Werts absolut notwendig.

Beispiel: Bei einer Sprintlänge von 30 Kalendertagen (also etwa 21 Arbeitstagen) geht man davon aus, dass jedes Teammitglied ca. 14 Arbeitstage für die Realisierung der Storys Points zur Verfügung hat. Wenn ein Team aus fünf Entwicklern besteht, erhält man somit eine Sprintkapazität von 70 Personentagen.

Mit Hilfe der Sprintkapazität lässt sich grob abschätzen, wie lange das Team für die Realisierung eines Story Points benötigt. Hat das Team z.B. bisher im Mittel 35 Story Points je Sprint realisieren können, dann benötigt es für die Realisierung eines Story Points etwa 2 Personentage. Der Wert 35 wird als "Velocity" bezeichnet und bezeichnet den geschätzten "Output" je Sprint, gemessen in Story Points (s. nachfolgender Abschnitt).

Empfehlung für die Praxis

Die Sprintkapazität eines Teams sollte nicht geschätzt werden. Hier ist der Einsatz eines Zeiterfassungssystems sinnvoll, in dem feingranular erfasst wird, wofür die Arbeitszeiten aufgewendet werden. Auf diese Weise lässt sich ermitteln, wie viel Zeit tatsächlich für Besprechungen, "Tagesgeschäft", Systemarbeiten etc. aufgewendet wird. Realistische Werte enthalten durchaus "Sprengstoff" innerhalb einer Organisation: Bei einem Projekt mit einem hohen Overhead wurde schon festgestellt, dass die Sprintkapazität des Teams weniger als 50% der gesamten Arbeitszeit betragen hat.

Velocity

Die Velocity ist eine Angabe über den Output, den ein Team in einem Sprint produziert, also wie viele Story Points in einem Sprint umgesetzt werden konnten ("umgesetzt" bezeichnet hier lauffähigen, getesteten, dokumentierten und gelieferten Code. Siehe hierzu auch den Abschnitt "Definition of Done"). Cohn diskutiert drei Möglichkeiten, die Velocity eines Teams zu bestimmen (Cohn, 2005):

1. Verwendung historischer Daten
2. Durchführung einer oder mehrerer Iterationen, um die Velocity zu bestimmen
3. Prognose einer Velocity

Die beste Möglichkeit ist, einen oder mehrere Sprints durchzuführen und dadurch die Velocity zu bestimmen. Hierbei ist zu berücksichtigen, dass ein Team zu Beginn eines Projekts eine geringere Produktivität hat. Die Velocity eines Teams sollte also nur mit Vorsicht als konstante Größe angesehen werden – in der Regel schwankt dieser Wert (Cohn, 2009).

Gibt es keine Möglichkeit, einen oder mehrere Sprints durchzuführen, um die Velocity zu bestimmen, kann man ersatzweise einen ersten Wert aus historischen Daten bestimmen. Dieses Verfahren lässt sich wie folgt – stark vereinfacht – skizzieren:

- Zu Anforderungen, die man in der Vergangenheit in einem bestimmten Zeitfenster realisiert hat, formuliert man User Storys.
- Für diese User Storys führt man nachträglich einen Planungspoker durch.
- Schließlich dividiert man den Zeitaufwand, der für die Realisierung benötigt wurde, durch die Summe der im Planungspoker ermittelten Story Points. Wurden beispielsweise in 2959 Stunden 269 Story Points realisiert, hat man für einen Story Point 11 Stunden benötigt.

Berechnung der Sprintkapazität

Sprintkapazität (SK) = Netto-Arbeitszeit x Teammitglieder

Berechnung der Velocity

Velocity (V) = SK / Dauer 1 Story Point (SP)

Beispiel: Berechnung der Dauer eines Story Points

SK = 14 Personentage x 5 = 70 Personentage

V = SK / SP → SP = SK / V

SP = 70 Tage / 35 = 2 Tage

Der dadurch ermittelte Wert gibt an, wie viele Stunden (oder Tage) für die Realisierung eines Story Points benötigt wurden. Mit Hilfe der Sprintkapazität des Teams hat man auf diese Weise einen ersten Wert für die Velocity bestimmt.

Empfehlung für die Praxis

Man benötigt mehrere Sprints, bis ein brauchbarer Wert für die Velocity des Teams zur Verfügung steht. Insbesondere bei neu gebildeten Teams, neuer Technologie und einem bisher unbekannten Fachgebiet ist aktuell kein anderer Weg erkennbar, um die Velocity zu ermitteln. Aber auch hier ist zu bedenken, dass ein neues Team, welches u.U. mit einer wenig vertrauten Technologie arbeitet und sich möglicherweise auch noch mit der Anwendungsdomäne nicht auskennt, in den frühen Sprints eine erheblich geringere Velocity aufweisen wird als in darauffolgenden. Das Team sollte am Ende jedes Sprints die Velocity berechnen und mit den Werten aus früheren Sprints vergleichen. Anschließend muss das Team entscheiden, welche Velocity für die zukünftigen Sprints zugesagt wird. Hierbei ist auch die Sprintkapazität regelmäßig zu überprüfen.

Grobe Schätzung der Projektdauer

Kennt man die Sprintkapazität des Teams und ist man in der Lage, eine Schätzung für alle User Storys im Product Backlog vorzunehmen, kann man eine grobe Schätzung der Gesamtdauer des Projekts vornehmen, indem man die Gesamtzahl der Story Points durch die Velocity des Teams dividiert.

Beispiel: Schätzt man den Gesamtaufwand aller User Storys im Product Backlog mit 200 Story Points, dann benötigt man bei einer Velocity von 35 Story Points ungefähr 6 Sprints. Bei einer Sprintlänge von vier Wochen beträgt damit die Gesamtdauer etwa ein halbes Jahr. Hierbei ist allerdings äußerste Vorsicht geboten, und dieser Wert darf nur als grobe Schätzung verstanden werden, weil

- die User Storys am unteren Ende des Product Backlogs naturgegeben noch sehr vage sein können und
- die User Storys im Product Backlog, die nicht im aktuellen Sprint umgesetzt werden, vom Product Owner jederzeit verändert werden dürfen.

Definition of Done

Bei der Schätzung der Aufwände ist es absolut unerlässlich, dass sich das Team darauf einigt, was alles zur Realisierung einer User Story gehört. Dies wird als "Definition of Done" bezeichnet. Die Realisierung einer User Story bedeutet nicht nur, dass die geforderte Funktionalität programmiert ist. Es bedeutet außerdem (Bleek, Wolf, 2008):

- Die Funktionalität "läuft",
- sie ist getestet,
- sie ist dokumentiert,

- sie ist ausgeliefert und
- der Kunde hat die Lieferung akzeptiert.

Wird beim Planungspoker nur der Aufwand für die Programmierung geschätzt, kann man hilfsweise z.B. von folgender Verteilung der Arbeitszeit ausgehen:

- 1/3 für die Programmierung
- 1/3 für die Tests
- 1/6 für die Analyse
- 1/6 für die Dokumentation

Das bedeutet, dass der Gesamtaufwand ungefähr mit drei multipliziert werden müsste, wenn beim Planungspoker nur der Implementierungsaufwand geschätzt wurde.

Zuschläge für Dokumentation, Projektmanagement etc.

Schätzt man nur die Aufwände für die Realisierung, würde sich nach der genannten Faustformel der geschätzte Gesamtaufwand verdreifachen. Diesen Wert muss man jedoch stets kritisch hinterfragen. Bei kleinen Projekten ist der ermittelte Gesamtaufwand regelmäßig zu hoch, bei großen Projekten kann er durchaus zu niedrig sein. So wurde z.B. in einem großen Projekt festgestellt, dass der Aufwand zur reinen Implementierung nur ca. 20% des Gesamtaufwands betragen hat. Tendenziell ist hier davon auszugehen, dass der relative Anteil für die Programmierung mit zunehmender Größe des Projekts abnimmt.

Weitere Empfehlungen

Kleine und konkrete User Storys

Für die Zuverlässigkeit der Schätzung hat es sich als sinnvoll erwiesen, kleine und möglichst konkrete User Storys zu verwenden. Wir haben gute Erfahrungen damit gemacht, als Richtgröße für die User Storys einen Personentag anzustreben. Die Zuverlässigkeit der Schätzung nimmt ab, wenn User Storys zu groß oder so vage sind, dass sich die Entwickler keine konkrete Vorstellung vom notwendigen Aufwand machen können. Aus diesem Grund sollten User Storys mit dem Maximalwert (hier 21) nicht in die Schätzung für die Sprints eingehen. Dieser Wert signalisiert häufig, dass eigentlich eine Schätzung "aus dem Bauch" vorgenommen wurde. Wenn man nicht klar verstanden hat, worum es in einer User Story geht, sollte man hierfür den Aufwand nicht schätzen.

Viele User Storys

Große User Storys sollten folglich in mehrere kleine User Storys zerlegt werden. Eine gute Faustregel ist, dass die User Storys umso kleiner und konkreter sein sollten, je weiter sie oben im Product Backlog stehen.

Bei einer großen Anzahl von User Storys gleichen sich Fehleinschätzungen nach oben und nach unten gegeneinander aus (Pichler, 2008). Problematisch wird das Verfahren, wenn es nur wenige User Storys zu schätzen gibt. Hier ist die Schätzung des Gesamtaufwands mit einer hohen Unsicherheit behaftet.

Breit qualifizierte Entwickler im Team

Durch eine funktionale Arbeitsteilung entstehen hochgradig spezialisierte Entwickler. Dies kann sich nachteilig auf die Schätzung auswirken, weil die Entwickler sehr hohe Werte bei der Realisierung von User Storys angeben, die außerhalb ihrer Hauptqualifikation liegen. Ein GUI-Spezialist schätzt beispielsweise den Aufwand für die Realisierung von Datenbankoperationen hoch ein. Für den Datenbankbankspezialisten erscheint die Programmierung des Front-Ends extrem aufwändig, weil er sich mit den hierfür eingesetzten Werkzeugen nicht auskennt. Entwickler in Scrum-Teams sollten aus diesem Grund im Idealfall über eine umfassende Qualifikation verfügen (Linssen, 2009).

Trotzdem schätzt jeder im Team den Aufwand für sämtliche User Storys, da man nicht sicher ausschließen kann, dass immer das qualifizierteste/schnellste Teammitglied tatsächlich die Story im Sprint auch realisiert. Im Planning Poker spielt es eben keine Rolle, wie schnell der "schnellste" Entwickler ist, sondern wie schnell das Team ist – und zum Team gehören nun mal auch alle anderen, die weniger schnell sind. Dadurch lassen sich völlig unrealistische Schätzungen von "Assen" und "Experten" vermeiden, die dann hinterher (wenn "normale" Entwickler "am Werk" sind) überhaupt nicht eingehalten werden können.

Vergleich der Schätzungen untereinander

Story Points sind relative Größen. Unproblematisch ist, dass diese Größe vom Team und dem Projekt abhängt. Problematisch ist dagegen, wenn sich die Größe der Story Points im Zeitverlauf verändert oder die Story Points inkonsistent verwendet werden.

- Eine **Veränderung im Zeitverlauf** liegt vor, wenn eine User Story in zwei verschiedenen Schätzrunden geschätzt werden muss und in einem früheren Sprint ein anderer Wert zugeordnet wurde als in einem späteren Sprint. Diese Veränderung kann berechtigt sein, wenn man neue Erkenntnisse hinzugewonnen hat. Ist dies nicht der Fall, so ist der veränderte Schätzwert zu hinterfragen.
- Eine **inkonsistente Schätzung** liegt vor, wenn tatsächlich ähnlich komplexe User Storys unterschiedlich groß geschätzt werden oder wenn tatsächlich unterschiedlich komplexe User Storys den gleichen Wert erhalten.

Empfehlenswert ist hier der ständige Vergleich von Anforderungen untereinander, um eine in sich schlüssige Schätzung zu erhalten (Pichler, 2008).

Wenn es möglich ist, sollten für alle beim Planungspoker verwendeten Werte mehrere Referenzbeispiele vorhanden sein.

Keine Puffer, keine Scheinpräzision

Die Aufgabe, Entwicklungsaufwände zu schätzen, ist bei Entwicklern chronisch unbeliebt, weil sie dafür häufig hinterher zur Rechenschaft gezogen werden, wenn die Schätzung so nicht eintrat. Deshalb bauen Entwickler Puffer in ihre Schätzung ein. Vermeintlich ist man dadurch in der Lage, nun präzise den Aufwand zu "berechnen". Tatsächlich wird hier eine Präzision vorgegaukelt, die hauptsächlich aus "Luft" (dem Puffer) besteht.

Bei Scrum wird die Unsicherheit einer Schätzung toleriert. Es muss allen Beteiligten klar sein, dass keine präzisen Schätzwerte zu erwarten sind und dass Schätzungen in späteren Sprints revidiert werden können. Die Erfahrung zeigt, dass Schätzungen durch Erfahrung genauer werden. Aus diesem Grund sollten in die Schätzungen keine Puffer eingebaut werden.

Fazit

Planning Poker lässt sich schnell und ohne großen Aufwand einführen. Es schätzen die Personen den Aufwand, die hinterher auch für die Realisierung zuständig sind. Die Aufwandsschätzung ist eine Teamschätzung, das heißt, sie orientiert sich nicht an der Geschwindigkeit der schnellsten Mitarbeiter. Maßgebend ist die Geschwindigkeit, die das ganze Team glaubt, einhalten zu können, bzw. die es in der Vergangenheit hat realisieren können. Damit ist Planning Poker gegenüber anderen Verfahren, die mit idealtypischen Annahmen bzgl. Produktivität arbeiten, weit überlegen. Die im Projekt gesammelten Erfahrungen fließen in die Schätzungen ein, was sich positiv auf die Präzision der Schätzung auswirkt.

Planning Poker ist allerdings wenig geeignet, um zu Beginn eines Projekts den Gesamtaufwand "präzise" vorherzusagen, was für Manager eher unbefriedigend ist. Darüber hinaus verspricht Planning Poker keine "präzise" Schätzung. Wenn man nur wenige Storys schätzt, ist die Unsicherheit bei der Schätzung recht hoch. Sie nimmt tendenziell mit der Anzahl der zu schätzenden Storys ab. Bei der Anwendung des Verfahrens ist darauf zu achten, kleine User Storys (z.B. ein Personentag) zu verwenden. Außerdem ist eine realistische Einschätzung der Sprintkapazität wichtig, da diese tendenziell zu hoch angesetzt wird.

Literatur

- Beedle, M.; Schwaber K.: Agile Software Development with Scrum, Upper Saddle River, N.J., Prentice Hall, 2002
- Bleek, W.-G.; Wolf H.: Agile Softwareentwicklung: Werte, Konzepte und Methoden, 1. Aufl., Heidelberg, dpunkt Verlag, 2008
- Boehm, B. W.: Software Engineering Economics, Prentice Hall, 1981

- Bundschuh, M.; Dekkers C.: The IT Measurement Compendium: Estimating and Benchmarking Success with Functional Size Measurement, 1. Aufl., Berlin, Heidelberg, New York, Springer, 2008
- Cohn, M.: User Stories Applied: For Agile Software Development, Upper Saddle River, N.J., Addison-Wesley Longman, Amsterdam, 2004.
- Cohn, M.: Agile Estimating and Planning, Upper Saddle River, N.J., Prentice Hall International, 2005
- Cohn, M.: Succeeding with Agile: Software Development Using Scrum, 1. Aufl., Upper Saddle River, N.J., u.a.: Addison-Wesley Longman, 2009
- GPM Deutsche Gesellschaft für Projektmanagement, M. Gessler und SPM Swiss Project Management Association, Kompetenzbasiertes Projektmanagement (PM3): Handbuch für die Projektarbeit, Qualifizierung und Zertifizierung auf Basis der IPMA Competence Baseline Version 3.0, 2. Aufl., Nürnberg, GPM Deutsche Gesellschaft für Projektmanagement, 2009
- Koschek, H.: Geschichten vom Scrum: Von Sprints, Retrospektiven und agilen Werten, 1. Aufl., Heidelberg, dpunkt Verlag, 2009
- Linssen, O.: Agile Vorgehensweisen – Wandel im Management von IT-Projekten, in 26. Internationales Deutsches Projektmanagement Forum: Die Kunst des Projektmanagements. Inspiriert durch den Wandel: Tagungsband, 14./15. Oktober 2009, bcc Berliner Congress Center, Nürnberg, 2009, S. 330 - 339.
- McConnell, S.: Software Estimation: The Black Art Demystified, 1. Aufl., Redmond, Washington, Microsoft Press Corp., 2006
- Pichler, R.: Scrum – Agiles Projektmanagement erfolgreich einsetzen, 1. Aufl., Heidelberg, dpunkt Verlag, 2008
- Schwaber, K.: Agile Project Management with Scrum, Redmond, Washington, Microsoft Press Corp., 2004
- Schwaber, K.: Scrum im Unternehmen, 1. Aufl., Unterschleißheim, Microsoft Press Deutschland, 2008
- Schwaber, K.: Scrum-In-Depth. It Depends on Common Sense, Ver. 1.3. o.O., 2010
- Wirdemann, Ralf: Agile Softwareentwicklung mit Scrum und User Stories, projektmagazin 02/2010

Planning Poker

Planning Poker ist eine effektive Methode zur relativen Aufwandsschätzung von User Stories bzw. Aufgaben in kleinen Gruppen. Die agile Herangehensweise ermöglicht eine direkte Bestimmung des Aufwands durch die Mit-glieder des Umsetzungsteams. Die vorhandene Fachexpertise der einzelnen Personen wird eingesetzt, um ein zu schätzendes Thema in der Arbeitsgruppe aus unterschiedlichen Aspekten, Hintergründen und im Mehr-Augen-Prinzip zu bewerten.

Einsatzmöglichkeiten

- Initiale Aufwandsschätzung aller Anforderungen in Form von User Stories
- Regelmäßige (Neu-)Bewertung noch offener und/oder neuer Aufgaben, um gezielt auf Änderungen des Kontextes oder der Projektziele reagieren zu können

Vorteile

- Einschätzung aus unterschiedlichen Perspektiven, je nach Zusammenstellung des Teams
- einfaches Setup
- kann in verteilten Teams (z.B.: via Video-Web-Session) eingesetzt werden
- fördert die Teilnehmer-Identifikation mit den Anforderungen
- genauere Aufwandsschätzung durch die kollektive Betrachtung, im Vergleich zu individuellen Schätzungen
- unabhängig von der Art der vorliegenden Anforderungen

Grenzen, Risiken, Nachteile

- Die Bewertungsergebnisse sind nur durch eine entsprechende Übersetzung in absolute Maßzahlen (monetär, finanziell) oder in andere Managementsysteme (z.B. Risikoplanung) übertragbar.
- Unter fünf Personen verliert die Methode ihre kontextbezogenen Vorteile.
- Bei über elf Personen wird die Bewertung unübersichtlich und aufwendig.
- Ohne ausreichenden fachlichen Hintergrund der Teilnehmer wird die Schätzung spekulativ.

Ergebnisse

Story Points für jede geschätzte User Story

Die Story Points sind eine relative Maßzahl zur Abschätzung des Aufwands der User Stories und können nicht direkt mit Arbeitstagen oder monetären Werten gleichgestellt werden. Mit Hilfe von projektspezifischen Umrechnungsformeln können auf Basis der ersten Ist-Werte absolute Aufwandswerte geschätzt werden und als Input für andere Bereiche verwendet werden (z.B. Ressourcen und Einsatzplanung).

Hinweise auf Unsicherheiten und mangelnde Klarheiten bei den Anforderungen

Liegen die einzelnen Schätzwerte für eine User Story trotz ergänzender Informationen und mehrfacher Schätz-runden weit auseinander wird dieser mangelnde Konsens in der User Story dokumentiert.

Ergänzte Dokumentation der User Stories

Informationen, die in der Dokumentation der User Stories (Beiblatt, Excel-Liste, Software …) festgehalten wird.

Voraussetzungen

- grundsätzliche Akzeptanz der Methode bei den Teilnehmern
- ungestörter, ruhiger Arbeitsraum in für die Teilnehmerzahl ausreichender Größe

Qualifizierung

- Alle Teilnehmer benötigen Kenntnis der Grundregeln des Planning Pokers
- Neue Teilnehmer benötigen zwei bis drei Übungsrunden, um mit der Methodik vertraut zu werden und sie effizient einsetzen zu können.
- Der Leiter des Workshops sollte einschlägige Moderationserfahrung haben.

Benötigte Informationen

- User Stories, normalerweise in Form eines Product Backlogs bzw. Sprint Backlogs
- Erfahrungswissen der Teilnehmer (idealerweise aus ähnlichen Aufgabenstellungen)
- Expertenwissen der Teilnehmer über die einzusetzenden Realisierungstechniken
- Planning Poker Skala (z.B. Fibunacci-Zahlen, T-Shirt Größen) zur Sicherstellung der Vergleichbarkeit der Ergebnisse

Benötigte Hilfsmittel

- Pinnwand zur Sammlung und Sortierung einzelner Ergebnisse
- Moderationsmaterialien (Karten, Flip-Chart, Stifte) zur Bearbeitung und Darstellung der Ergebnisse
- Ausreichende Anzahl an Kartensets für Planning Poker. Auf den Karten sind die Zahlenwerte aufgedruckt, die zur Schätzung einer User Story dienen. Sonderkarten (z.B. Karte "Kaffeetasse") können zur Steuerung des Ablaufs eingesetzt werden.
- Moderationskarten mit den unter "Benötigte Informationen" genannten User Stories.
- Evtl. Material für Varianten bei der Bewertungstechnik (z.B.: Poker-Chips, T-Shirt in den Größen XS-XXXL)
- (Optional) Software, die Planning Poker unterstützt bzw. die Planning-Poker-Karten ersetzt (siehe: "Software") ...

Im Falle des Einsatzes einer Collaboration Plattform (Skype, Google Hang-out...) für verteilte Teams:

- Webcam, Headset
- Einen für Videokonferenzen geeigneten, ungestörten Arbeitsplatz
- Software, die internetbasierte Audio- und Video-Kommunikation unterstützt (z.B.: Skype, Webex, Sametime...) – inklusive eines entsprechenden Accounts.

Durchführung

- Schritt 1: Bereiten Sie den Planning-Poker-Workshop vor!
- Schritt 2: Workshop-Auftakt
- Schritt 3: Stellen Sie die Methode Planning Poker und die Kartenwerte vor!

- Schritt 4: Präsentieren der User Story
- Schritt 5: Klären Sie inhaltliche Fragen zur User Story!
- Schritt 6: Die verdeckte Schätzung
- Schritt 7: Begutachten Sie die Ergebnisse gemeinsam!
- Schritt 8: Übergeben Sie die Ergebnisse an die Schnittstellen!

Schritt 1: Bereiten Sie den Planning-Poker-Workshop vor!

Überprüfen Sie, ob die User Stories in der aktuellen Fassung und vollständig vorliegen. Stellen Sie sicher, dass alle zu behandelnden User Stories auf Moderationskarten gedruckt / geschrieben werden.

Achten Sie darauf, dass alle nachfolgenden Teamrollen besetzt sind:

- Product Owner: Er vertritt die Seite des Auftraggebers, priorisiert User Stories und steht den Schätzern zur Verfügung, um User Stories detailliert zu erläutern. Der Product Owner gibt keine Schätzungen ab.
- Scrum Master: Üblicherweise organisiert er das Planning Poker, achtet auf die Einhaltung der Spielregeln und dokumentiert die Ergebnisse.
- Entwickler: Sie realisieren später die User Stories. Durch ihr Fachwissen und ihre Erfahrungen aus anderen Projekten liefern sie den essenziellen Beitrag für das Planning Poker. Die Entwickler liefern die Schätzungen, ggf. geben auch Spezialisten Schätzungen ab (s.u.).
- Spezialisten (z.B. Analysten): Ist Ihnen vorab bekannt, dass bei den Entwicklern fachliche Wissenslücken vorhanden sind, können Sie die Runde durch Experten für diese Themen ergänzen. Diese unterstützen das Team indem sie fachliche Rückfragen beantworten und beteiligten sich bei denjenigen User Stories an der Schätzung, für die sie Experten sind.

Benennen Sie bei der Einladung zum Planning Poker das zu schätzende Vorhaben (Ihr Projekt).

Ergänzen Sie bei der erstmaligen Anwendung Hinweise zum Thema Planning Poker. Planen Sie bei einem mit Planning Poker unerfahrenen Team mehr Zeit ein, um Vorbereitungsrunden durchzuführen, bei denen die Teilnehmer die Methode lernen und einüben.

Schritt 2: Workshop-Auftakt

Der Product Owner präsentiert die Produkt/Projekt-Vision und stellt die thematischen Schwerpunkte des nächsten, zu schätzenden Sprints vor.

Der Scrum Master stellt neue Teammitglieder und hinzugezogene Experten vor. Ferner erläutert er, dass mit Hilfe von ein bis zwei Einstimmungsrunden mit dafür geeigneten User Stories die "Kalibrierung", sprich das gruppenweite Verständnis zu Kartenwerten, erreicht wird.

Schritt 3: Stellen Sie die Methode Planning Poker und die Kartenwerte vor!

Beschreiben Sie, je nach Vorkenntnissen der Teilnehmer, das Prinzip und die Funktionsweise von Planning Poker. Erklären Sie in jedem Fall die Aufgaben der einzelnen Rollen (s.o.) und erläutern Sie die Kartenwerte:

- Die Sonderkarte "0" bedeutet, dass die Aufgabe nach Meinung des Teilnehmers bereits erledigt wurde.
- Die Sonderkarte "?" sollte ein Teilnehmer dann wählen, wenn er während des Schätzens feststellt, dass er die Aufgabe nicht verstanden hat oder er weitere Informationen benötigt, um eine Schätzung abgeben zu können.
- Die Sonderkarte mit dem Symbol "Kaffeetasse" zeigt auf, dass der Teilnehmer eine Pause beantragt.
- Zahlenkarten: Der Zahlenwert gibt an, wie hoch der Teilnehmer den Aufwand zur Realisierung der Aufgabe hält im Vergleich zu einer Referenzaufgabe, die einen festgesetzten Zahlenwert erhält (z.B. 5).

Wenn die Teilnehmer noch keine Erfahrung mit Planning Poker haben, sollten Sie zwei bis drei Vorbereitungsrunden durchführen, in denen Sie die Schritte 3 bis 6 mit ausführlichen Erläuterungen und mehr Zeit für die Schätzungen durchführen. Evtl. können Sie die so erzielten Schätzwerte bereits verwenden, andernfalls legen Sie die User Stories wieder in den zu schätzenden Pool zurück.

Schritt 4: Präsentieren der User Story

Der Product Owner wählt eine User Story für die nächste Schätzrunde aus und stellt sie vor. Für die ersten beiden Schätzrunden sollte er User Stories aussuchen, die sich gut für die "Kalibrierung" eignen (s.o.).

Schritt 5: Klären Sie inhaltliche Fragen zur User Story!

Der Moderator (z.B. der Scrum Master) frägt die Teilnehmer, ob alle die User Story vollständig verstanden haben. Bei Bedarf moderiert er die Klärung der Fragen. Wenn es zu keiner Klärung kommt, wird die User Story auf den nächsten Workshop vertagt. Der Product Owner muss die User Story bis dahin mit den entsprechenden Anforderungen verfeinern, damit er sie erneut einbringen kann.

Schritt 6: Die verdeckte Schätzung

Der Moderator fordert die Teilnehmer auf, die Schätzung vorzunehmen. Jeder Teilnehmer schätzt für sich den Aufwand für die betrachtete User Story, nimmt die Karte mit dem entspre-

chenden Zahlenwert (oder eine Sonderkarte) aus seinem Kartendeck und legt diese verdeckt vor sich ab. Der Moderator bestimmt, wie viel Zeit hierfür zur Verfügung steht, dies hängt insbesondere davon ab, wie die Entwickler mit der Methode vertraut sind.

Schritt 7: Begutachten Sie die Ergebnisse gemeinsam!

Haben alle Teilnehmer den vorangegangenen Schritt ausgeführt, decken sie die Karten auf Kommando des Moderators auf. Begutachten Sie nun die Ergebnisse gemeinsam. Falls ein Teilnehmer eine der drei Sonderkarten "0", "?" oder "Kaffeetasse", gelegt hat, führen Sie die entsprechenden Schritte durch:

- "0": Überprüfen Sie, ob die User Story bereits erledigt wurde. Ist dies tatsächlich der Fall, vermerken Sie dies auf der Moderationskarte der User Story (Moderationskarte, evtl. auch bereits in der User Story selbst) und legen Sie die Karte für die anschließende Dokumentation zur Seite. Eine erneute Schätzung ist in diesem Fall nicht erforderlich. Falls die User Story noch nicht umgesetzt wurde, führen Sie die Schätzung nochmals durch.
- "?": Klären Sie die Rückfrage und führen Sie die Schätzung erneut durch.
- "Kaffeetasse": Klären Sie den allgemeinen Bedarf einer Pause. Führen Sie die Pause sofort durch, oder einigen Sie sich auf eine entsprechende Planung. Schätzen Sie anschließend die User Story erneut.
- Besteht das Schätzergebnis ausschließlich aus Zahlenwerten, analysieren Sie die Häufigkeitsverteilung der Schätzwerte und gehen Sie dann wie folgt vor:
- Bestehen offensichtlich große Abweichungen zwischen den Schätzungen (z.B.: vier Personen geben einmal 2, zweimal 3 und einmal 8 Punkte als Schätzwert ab), erklärt die Person mit der höchsten und die Person mit der niedrigsten Karte den Grund für die jeweilige Entscheidung. Danach wird die gleiche User Story erneut geschätzt (beginnen Sie erneut bei Schritt 4).
- Handelt es sich um ähnliche Ergebnisse (z.B.: vier Personen geben zweimal 2 und zweimal 3 Punkte als Schätzwert ab, lassen Sie in höchstens zwei bis drei Minuten Diskussion die Gruppe zu einem Konsenswert finden und schreiben Sie diesen auf die User-Story-Karte.

Kommt es bei einer User Story mehrfach zu starken Abweichungen, dokumentieren Sie den Sachverhalt im Protokoll. Anschließend empfiehlt es sich, die Mehrheitsentscheidung zu akzeptieren und auf der User-Story-Karte zu dokumentieren. Es steht Ihnen ebenfalls frei, die Schätzung der betroffenen User Story ein weiteres Mal durchzuführen.

Uneinigkeit bei der Vergabe der Story Points pro User Story ist ein klares Indiz für Unsicherheiten bezüglich der zugrunde liegenden Aufgabe. Kann diese durch weitere Informationen zur Aufgabe nicht aufgelöst werden, fließt der Hinweis zum mangelnden Konsens in die Dokumentation der User Story mit ein. Schätzen Sie die User Story im vorangeschrittenen Projekt neu (sofern verschiebbar) oder übergeben Sie die Unsicherheit bezüglich der Aufgabe an Ihr Risikomanagement.

Schritt 8: Übergeben Sie die Ergebnisse an die Schnittstellen!

- Überführen Sie die Ergebnisse (User Stories inklusive Konsens-Schätzungen in das Product Backlog.
- Übersetzen Sie Bewertungsergebnisse in die Ressourcen- und Einsatzplanung und / oder Ihr Risikomanagement.

Praxistipps

- Geben Sie neuen Teilnehmern die Möglichkeit, in einer Übungsrunde die Praktik des Planning Poker zu testen!
- Geben Sie den Teilnehmern die Möglichkeit, sich vorzustellen. Dies gibt den Planning-Poker-Runden eine persönlichere Note und erleichtert den Einstieg. Zudem geben Sie den Teilnehmern die Möglichkeit, sich gedanklich auf eine neue Rolle, bzw. Expertise einzurichten.
- Geben Sie nach jeder Runde die Möglichkeit, Fragen zu stellen. Auch ohne stark auseinanderdriftende Schätz-Ergebnisse können im Team nochmals nachgelagerte Fragen entstehen. Eine im Kontext stimmige Schätzung muss nicht zwangsläufig mit dem Verständnis der User-Story zusammenhängen.

Gehen Sie mit Skepsis konstruktiv um!

Wenn Sie Planning Poker in einem Team zum ersten mal einsetzen, können Sie auf eine gewisse Skepsis gegenüber der "neuen Methode" stoßen. Lassen Sie sich dadurch nicht beirren, sondern laden Sie die Teilnehmer dazu ein, diese neue Methode auszuprobieren und herauszufinden, ob sie für sie anwendbar ist. Bieten Sie einen Vergleich zu traditionellen Schätzmethoden an und stellen Sie die Vor- und Nachteile der beiden Vorgehensweisen gegenüber Planning Poker sollte nicht bei Schätzungen eingesetzt werden die immer wiederkehrende Aufgaben enthalten, hier kann auf Erfahrungswerte zurückgegriffen werden.

Testen Sie, ob Planning Poker für Ihr Team und Ihr Projekt geeignet ist!

Genauso wie die Teammitglieder können auch Sie beim erstmaligen Einsatz testen, ob die Methode für das zu Grunde liegenden Projekt und dessen Team zielführend anwendbar ist. Wägen Sie ab, ggf. in Rücksprache mit anderen Stakeholdern, welche Methode mit angemessenem Aufwand ein ausreichend genaues Schätzergebnis erzielt.

Setzen Sie Planning Poker als Führungsinstrument ein!

In manchen Projekten ist es möglich, dass im späteren Verlauf des Projekts Teilnehmer die Aufgaben des Moderators für das Planning Poker übernehmen. Diese Übertragung von Verantwortung können Sie als Führungsinstrument nutzen: Lernen Sie die Fähigkeiten der Teammitglieder kennen und unterstützen Sie die neuen Moderatoren nachhaltig, z.B. durch Fortbildungen für Moderation.

Varianten

Planning Poker mit alternativen Bewertungstechniken (z.B. Poker-Chips, T-Shirt-Größen)

Neben den klassischen Karten können auch andere Gegenstände, die entsprechende Werte symbolisieren, eingesetzt werden. Gut geeignet sind Poker-Chips, da diese starke Unterschiede in den jeweiligen Werten widerspiegeln (1, 5, 25, 100, 500). Dies ist wichtig, da bei nur leichten Unterschieden der Werte eine Diskrepanz der Meinungen nicht klar hervorgeht. Die ebenfalls verbreitete Variante mit Größenangaben von T-Shirts ist zwar einfach anzuwenden, zeigt aber die Diskrepanz unterschiedlicher Bewertungen nicht so klar auf.

Agile Planning Poker with remote teams

Bei dieser Variante findet der Workshop als virtuelles Treffen statt, d.h. über Videokonferenz oder softwaregestützt. Die "Spielregeln" sind die gleichen wie beim traditionellen Planning Poker vor Ort. Statt die Planning-Poker-Karten in einem Raum auf Kommando aufzudecken, werden diese in beim Einsatz einer Webcam in die Kamera gehalten. Softwaregestützte Collaboration-Tools ermöglichen z.B. eine für die anderen Teilnehmer verdeckte Vorab-Auswahl durch die einzelnen Benutzer auf ihrer individuellen Bedienungsoberfläche. Erst wenn alle Stimmen abgegeben sind, macht die eingesetzte Software die Karten für alle Teilnehmer sichtbar.

Herkunft

Die Methode wurde 2002 von James Grenning initiiert und von Mike Cohn (Eigentümer von "Mountain Goat Software") verbreitet (Lant, Michael: Estimate Story Size by Playing Agile Planning Poker, http://michaellant.com/2010/07/13/agile-planning-poker, zuletzt besucht am 10.8.2015.

Mountain Goat Software hält die eingetragene Marke auf Planning Poker Karten, sowie auf die Standard-Zahlenwerte: 0, 1, 2, 3, 5, 8, 13, 20, 40 und 100 (https://www.mountaingoatsoftware.com/agile/planning-poker/license, zuletzt besucht am 10. Juni 2015).

Die Methode wird primär dem agilen SCRUM-Vorgehen zugeordnet, kann aber auch in klassischen Projektvorgehensweisen angewandt werden. Aufgrund des einfachen Settings und dem direkten Feedback im Kreis der verantwortlichen Bearbeiter werden anspruchsvolle, nicht vollständig spezifizierte Aufgaben zu einem beliebigen Zeitpunkt des Projektverlaufs diskutiert und deren Aufwand bewertet. Planning Poker liefert nicht nur einen, dem Teilnehmerkreis entsprechend qualifizierten Erfahrungswert, sondern fördert durch die Stimmabgabe auch die inhaltliche Identifikation der Teilnehmer mit der Aufgabenstellung.

Autor

Christian Botta

Erstellt am: 17.09.2015

Die Alternative zum Planning Poker

Das Team Estimation Game

Tord Björsne
M.Sc. E.E bei Siemens Healthcare, Certified Scrum Product Owner

Ivan Kostial
Scrum Master bei der Siemens Healthcare

Klaus-Dieter Schmatz
Teamleiter und Program Manager bei Siemens Healthcare, Scrum Master

Üblicherweise erstellt der Projektmanager am Beginn eines neuen Projekts eine Aufwandsschätzung. Der Auftraggeber verwendet diese Abschätzung, um über die Freigabe oder den Abbruch des Projekts zu entscheiden. Im weiteren Projektverlauf dient sie als Ausgangspunkt für die Ressourcen- und Zeitplanung. Wegen der Tragweite dieser Entscheidungen neigen viele Organisationen dazu, sehr viel Zeit und Mühe in die Aufwandsschätzung zu investieren.

Trotz solch detaillierter Aufwandsschätzungen dauern viele Entwicklungsprojekte länger als geplant oder sie liefern nicht die erhofften Ergebnisse. Dieses Risiko liegt in die Natur von Entwicklungsprojekten. Schließlich sollen sie etwas Neues schaffen, das es vorher noch nicht gab. Da somit verlässliche Erfahrungswerte fehlen, lässt sich nicht mit Sicherheit vorhersagen, wie lange die Realisierung dauern wird.

Die agile Methodik zeigt einen Weg auf, das Bedürfnis des Auftraggebers nach vorhersagbaren Resultaten mit den Unwägbarkeiten in Einklang zu bringen, die sich unweigerlich während der Realisierungsphase einstellen. Die Erreichung der Projektergebnisse wird dabei in so genannte "Sprints" unterteilt (Iterationen). Jeder Sprint ist so ausgelegt, dass er neue Funktionen umfasst, die einen echten Kundennutzen haben. Die Funktionen mit dem höchsten Wert für den Kunden werden möglichst in den ersten Sprints des Projekts entwickelt. Auch wenn das Projektteam bis zum Projektabschluss nicht alle ursprünglich geplanten Funktionen fertigstellen kann, so entwickelt es doch mit hoher Wahrscheinlichkeit ein für den Kunden nützliches Produkt.

Voraussetzung für dieses Vorgehen ist, dass der gesamte Funktionsumfang in einzelne User Stories unterteilt wird, deren Umfang individuell abgeschätzt werden kann. Die wichtigste Messgröße dabei ist der relative Umfang dieser User Stories (vgl. Wirdemann, projektmagazin 2/2010), d.h. der Umfang einer User Story im Vergleich zu den anderen. Der Product Owner kann diese Information heranziehen, um das Product Backlog zu priorisieren, indem er den Umfang einer User Story gegen ihren Kundenwert abwägt und die User Stories mit dem besten Kosten/Nutzen-Verhältnis möglichst weit oben einsortiert.

Wir haben uns dafür entschieden, den Umfang der User Stories in der Einheit "Story Points" mit einer vorgegebenen Werteskala (s.u.) abzuschätzen. Nach unserer Auffassung ist es am besten, die Story Points nicht als absoluten Messwert, sondern als relative Größe zu verstehen. Das heißt, die Realisierung einer User Story mit 8 Story Points wird ungefähr dreimal solange dauern wie die Realisierung einer User Story mit einem Umfang von 3 Story Points, eine absolute Dauer für die Realisierung kann aber zunächst nicht angegeben werden.

Für die Abschätzung der Story Points gibt es neben dem relativ zeitaufwendigen "Planning Poker" (vgl. Linssen, projektmagazin 10/2012) das von Steve Blockmann entwickelte "Team Estimation Game" (Blockmann, 2007). Der Vorteil dieser Methode liegt darin, dass eine große Menge von User Stories innerhalb kurzer Zeit geschätzt werden kann.

Im Folgenden beschreiben wir unsere Erfahrungen mit dem Team Estimation Game, das wir in einem Teilprojekt mit einem Umfang von zehn Mannjahren verwendet haben. Wir konnten mit dieser Methode 250 User Stories innerhalb einer zweistündigen Sitzung analysieren. Die Ergebnisse halfen uns wesentlich dabei, das Projekt termingerecht abzuschließen und gleichzeitig die vorgesehene Funktionalität fast vollständig zu realisieren.

Das Projekt: Software-Entwicklung nach Scrum

Unser Teilprojekt hatte die Aufgabe, eine bestehende, webbasierte Software-Lösung als Desktop-Applikation neu zu implementieren und um einzelne Leistungsmerkmale zu erweitern. Diese Anwendung unterstützt die technische Qualitätssicherung, die während des Betriebs eines komplexen Medizinprodukts regelmäßig durchgeführt werden muss. Die Bedienungsoberfläche war neu zu entwickeln, während die Geschäftslogik auf eine neue technologische Basis übertragen werden sollte. Eine weitere, unverrückbare Randbedingung bestand darin, dass der Zeitrahmen vom Gesamtprojekt fest vorgegeben war.

Wir verfügten über drei Entwicklungsteams mit insgesamt 18 Entwicklern und Testern, einen Scrum Master und einen Product Owner. Die traditionelle Rolle des Projektmanagers ist in Scrum nicht vorgesehen; seine Aufgaben teilen sich im Wesentlichen der Scrum Master und der Product Owner. Die Entwicklung wurde in Sprints mit einer Dauer von jeweils vier Wochen untergliedert. Der Product Owner führte ein Product Backlog mit allen User Stories, die in der Reihenfolge ihrer Implementierung angeordnet waren.

Das Team Estimation Game

Ziel des Team Estimation Game ist, für jede User Story eine Maßzahl zu bestimmen, die den Umfang der User Story geeignet quantifiziert. Der Umfang einer User Story hängt sowohl von der Zahl der für ihre Umsetzung benötigten Funktionen als auch von der Komplexität der Anforderung ab. Gemessen wird der Umfang in der Einheit Story Point. Diese Story Points haben zunächst keinen bekannten Zusammenhang zu Aufwand und Projektdauer. Eine Abschätzung der Projektlaufzeit kann auf der Grundlage der Summe der Story Points aller User Stories im Product Backlog

und der Abarbeitungsgeschwindigkeit erfolgen. Die Abarbeitungsgeschwindigkeit drückt aus, wie viele Story Points innerhalb eines Sprints realisiert werden. Sobald die Abarbeitungsgeschwindigkeit (in der Terminologie von Scrum "Velocity" genannt) und die Umfänge aller User Stories bekannt sind, kann der Product Owner die Anzahl der Sprints für die Implementierung des gesamten Product Backlogs extrapolieren. Für die Abarbeitungsgeschwindigkeit kann er entweder Erfahrungswerte aus ähnlichen Projekten heranziehen, oder er wartet die Durchführung der ersten Sprints ab, um die Abarbeitungsgeschwindigkeit für das betrachtete Projekt zu ermitteln.

Die Motivation, den Umfang der User Stories in Story Points zu schätzen anstatt den Arbeitsaufwand für ihre Realisierung in Personentagen zu kalkulieren, liegt darin, die Illusion einer nicht vorhandenen Genauigkeit zu zerstören. Wir lassen als Maßzahlen für den Umfang einer User Story nur folgende Werte zu: 1, 2, 3, 5, 8, 13, 20, 40 und 100. Eine User Story kann also keinen Umfang von beispielsweise 3,47 Story Points haben. Damit ist allen Beteiligten klar, dass der Umfang nur eine ziemlich grobe Maßzahl darstellt und lange Diskussionen über den "exakten" Wert überflüssig sind.

Die vorgegebene Werteskala gilt auch dann, wenn man eine User Story weiter zergliedert, da ihre Einzelteile individuell leichter abgeschätzt werden können, als die User Story als Ganzes. Der Umfang der gesamten User Story ergibt sich dann als Summe der Story Points ihrer Einzelteile, z.B. für den Frontend- und den Backend-Teil. Diese Summe wird dann stets auf die nächst größere, zulässige Maßzahl aufgerundet. Wenn z.B. die Teile einer User Story zwei und fünf Story Points haben, dann beträgt die Größe der gesamten User Story nicht sieben, sondern acht Story Points, da dies der nächst größere zulässige Skalenwert ist.

Die normale Spanne für eine sinnvoll dimensionierte User Story liegt zwischen 1 und 13 Story Points. Einen höheren Schätzwert sollte der Product Owner als Feedback verstehen, dass die User Story zu umfangreich (episch) ist und besser in mehrere kleine User Stories aufgeteilt werden sollte.

Durchführung und "Spielregeln"

Zur Vorbereitung auf das Team Estimation Game stellt der Product Owner die zu bewertenden User Stories zusammen. In unserem Projekt waren das zum Zeitpunkt der Schätzung 250 Stück. Jede User Story wird auf ein eigenes Kärtchen mit einem gut lesbaren Titel in Fettschrift und einer kurzen Beschreibung in kleinerer Schrift gedruckt. Wir druckten dazu ganz einfach mehrere Kärtchen auf ein DIN A4 Blatt und schnitten sie mit einer Schere aus.

Der Moderator des Team Estimation Game, in unserem Fall der Scrum Master, erklärt den 15 Teilnehmern vorab die Methode. Er weist darauf hin, dass es nicht um die Ermittlung von exakten Zahlen geht, sondern darum, die Zuversicht in die Realisierung einer User Story auszudrücken, und zwar relativ zu den anderen User Stories. Mit dieser Zielsetzung ist es möglich, jede User Story innerhalb von wenigen Minuten abzuschätzen. Das Team benötigt ansonsten keine weitere Vorbereitung; die Team-Mitglieder sollten aber bereits mit dem Konzept der User Story vertraut sein und darüber hinaus ein Mindestmaß an Erfahrung mit dem Projektgegenstand mitbringen.

Das "Spiel" selbst läuft in den folgenden sechs Schritten ab:

Vorbereitung der Infrastruktur

Für das Team Estimation Game benötigt man Kärtchen mit den User Stories und eine ausreichend große Fläche, um die Kärtchen entsprechend ihres Umfangs in Gruppen zu sortieren. Wichtig ist, dass die Kärtchen leicht verschoben werden können. Wenn mit einer Wand als Arbeitsfläche gearbeitet wird, kann man mit Pinnwänden, Kärtchen und Pinns arbeiten oder man verwendet selbsthaftende Zettel z.B. auf einem Whiteboard. Genauso gut kann man das Spiel auf einer ebenen Fläche durchführen, z.B. auf zusammengestellten Tischen. Dies hat den Vorteil, dass man die Zettel einfach legen kann, allerdings muss man darauf achten, dass sie nicht durch Luftströmungen durcheinander gebracht werden, z.B. beim Lüften in einer Arbeitspause.

Wir stellten drei Schreibtische nebeneinander, um ausreichend Platz für das Arbeiten mit den Kärtchen zu haben. Diese Fläche wurde in gleich große Bereiche aufgeteilt, die wir mit Zetteln für die Skalenwerte 1, 2, 3, 5, 8, 13, 20, 40 und 100 markierten.

Vorstellung der User Stories

Als erstes präsentiert der für die User Stories verantwortliche Stakeholder, im Normalfall der Product Owner, den Teilnehmern kurz alle User Stories. Dabei ist die Aufnahmefähigkeit des Teams zu berücksichtigen: Wenn die User Stories zu abstrakt sind, kann es erforderlich sein, in geeigneten Abständen Pausen einzulegen. Der Moderator kann sich auch dafür entscheiden, besonders unklare User Stories aus dem Spiel zu nehmen. Wenn eine User Story nicht hinreichend klar ist, so schätzen die Teilnehmer ihren Umfang meist sehr hoch ein, um auf Nummer sicher zu gehen. In den meisten Fällen ist diese Vorsicht berechtigt und die hohe Schätzung spiegelt die Realität gut wider.

In unserem Beispiel stellte der Product Owner alle 250 User Stories innerhalb von 90 Minuten vor: Er benötigte dazu also durchschnittlich 22 Sekunden pro Story. Zu jeder Story stellten die Teilnehmer ein bis zwei Fragen, die der Product Owner kurz beantwortete. Es zeigte sich, dass eine detaillierte Besprechung der Funktionalität nicht erforderlich war.

Kalibrierung

Um allen Teilnehmern eine Orientierung für die Einschätzung der User Stories zu geben, sollten erfahrene Teilnehmer einige User Stories einschätzen, so dass nach Möglichkeit bei jedem Schätzwert ein oder zwei User Stories als Referenz liegen. Auf diese Weise wird die Bewertungsskala für das abzuschätzende Projekt gewissermaßen "kalibriert".

Auch bei uns wählten zwei vom Team bestimmte, erfahrene Teilnehmer einige repräsentative User Stories als Beispiele für die verschiedenen Skalenwerte aus: je ein oder zwei User Stories mit Komplexitätswert 1, 2, 3, 5 usw. Sie benötigten dafür nur fünf Minuten, da sie bereits umfangreiche Erfahrungen mit dem Produkt hatten. Die Gründe für die Auswahl der repräsentativen User Stories wurden kurz besprochen, sodass sich das Team daran orientieren konnte.

Diversifikation

Die User Stories wurden willkürlich auf die Teilnehmer aufgeteilt. Es ist nicht ausschlaggebend, wer die erste Schätzung abgibt, denn in der nächsten Phase kann die Schätzung von jedem modifiziert werden. Jeder Teilnehmer erhielt etwa dieselbe Anzahl von Kärtchen. Die Aufgabe war, alle Kärtchen auf den Tisch zu legen und jeweils dem passenden Skalenwert zuzuordnen. Dafür stand ein fester Zeitrahmen von 15 Minuten zur Verfügung. Die individuellen Entscheidungen durften von niemand kommentiert werden – es gab nur ein paar Versuche, gegen diese Regel zu verstoßen. Auch der Product Owner übernahm die Abschätzung von einigen User Stories, genau wie die anderen Teilnehmer.

Die ganze Phase soll bis auf vereinzelte Klärungen von User Stories mit dem Product Owner ziemlich ruhig ablaufen. Natürlich kamen Nachfragen zu einzelnen User Stories auf, weswegen der Moderator für einen kurzen Zeitraum Diskussionen zuließ. Ansonsten sorgte er für die Einhaltung der Grundregeln des Estimation Game und achtete darauf, dass die Teilnehmer stets über die verbleibende Zeit bis zum Abschluss der Phase informiert waren.

Konvergenz

In dieser Phase darf jeder Teilnehmer beliebige User Stories (von allen Teilnehmern) auf einen anderen Skalenwert verschieben, wenn er mit der aktuellen Einstufung nicht einverstanden ist. Die anderen Teilnehmer dürfen diese Aktion nicht kommentieren. Uns reichten für diesen Schritt 15 Minuten aus.

Während dieser Phase konnte man z.B. erkennen, wer ein natürlicher Anführer und wer unsicher war, ob eine User Story zu umfangreich war oder wie ausgeprägt die Technologie-Erfahrung eines Teilnehmers war. All dies war für den Scrum Master ein wertvoller Input, mit dem er in den folgenden Monaten arbeiten konnte. Beispielsweise wurden unsichere Team-Mitglieder eingeladen, am Pair Programming teilzunehmen, um Wissen von einem erfahreneren Kollegen aufnehmen zu können.

Konfliktlösung

In manchen Fällen konnten die Teammitglieder keinen Konsens erreichen. Wenn eine bestimmte User Story drei- oder viermal zwischen verschiedenen Skalenwerten hin und her wanderte, wurde sie vom Moderator aus dem Spiel genommen. Solche User Stories wurden nach dem Meeting mit dem Product Owner geklärt und anschließend mit der Methode "Planning Poker" separat abgeschätzt.

Nach dem Abschluss dieser Phase übernahm der Product Owner die in der Einheit Story Points ermittelten Umfänge der User Stories in das Product Backlog.

Beobachtungen während des Estimation Game

Während des Estimation Game konnten wir beobachten, wie die noch nicht so erfahrenen Teammitglieder von den Experten und Meinungsführern lernten. Wenn z.B. ein Experte eine bestimmte User Story in die 13-Story-Points-Klasse einteilt, dann erhalten die anderen Teammitglieder eine bessere Vorstellung, worum es bei dieser User Story geht. Das fördert das Verständnis und beeinflusst die weniger erfahrenen Teammitglieder, vergleichbare User Stories mit einem ähnlichen Komplexitätswert abzuschätzen.

Kann sich hingegen ein Teammitglied bei einer User Story überhaupt keinen Implementierungsansatz vorstellen, wird er wahrscheinlich einen sehr großen Umfang schätzen, der für seine Kollegen nicht plausibel ist – wodurch eine Wissenslücke offenkundig wird. Das hilft den Teammitgliedern, sich gegenseitig besser kennenzulernen, Sprints zu planen und Verbesserungsmaßnahmen im Team durchzuführen.

Eine offene, vertrauensvolle Atmosphäre im Team und der durch den Charakter eines Spiels erzeugte Spaßfaktor spielen eine wichtige Rolle. Die Möglichkeit, die Einstufung einer User Story eines Kollegen während der Konvergenzphase zu ändern, trägt viel dazu bei. In unserem Estimation Game hatte der Product Owner z.B. für eine bestimmte User Story eine sehr einfache Lösung im Kopf, weshalb er diese Story sehr niedrig einstufte. Der Architekt hatte jedoch eine ganz andere Sicht der Dinge, und so wanderte die Karte mehrmals zwischen den Kategorien 2 und 40 hin und her, bis schließlich der Scrum Master die User Story zur separaten Klärung aus dem Spiel nahm. Durch die gemeinsame, offene Diskussion der User Stories lernten sich das Team und der Product Owner besser kennen, wodurch sich der zuvor große Respektabstand verringerte.

Alle diese Aspekte führten dazu, dass die Teilnehmer ein sehr positives Feedback zum Estimation Game abgaben. Wir verbrachten zwei sehr produktive und auch unterhaltsame Stunden, mit denen wir uns zudem einige Wochen einsparten, in denen wir nach klassischer Vorgehensweise eine detaillierte Aufwandsschätzung erstellt hätten. Dabei ahnten wir damals noch gar nicht, wie genau unsere Schätzung letztendlich war.

Die Ergebnisse: Aussagekraft und Nutzen

Das Estimation Game liefert, wie jedes Schätzverfahren, keine exakten Größen. Vielmehr ist jeder einzelne Schätzwert für eine User Story mit einer gewissen Unsicherheit behaftet. Aufgrund unserer Erfahrung können wir zwar einigermaßen zuverlässig abschätzen, ob eine User Story größer oder kleiner als eine andere ist, aber jede Schätzung wird vom tatsächlichen Wert statistisch abweichen. Leider lässt sich diese Abweichung erst nach der Implementierung bestimmen, sobald der tatsächliche Wert bekannt ist. Die Gesamtgröße des Product Backlog ist also eine Summe von Werten mit zufälligen Abweichungen! Das klingt zunächst sehr ungenau, aber die Wahrscheinlichkeitsrechnung lehrt uns, dass sich die einzelnen Unsicherheiten nicht einfach zu einer großen Unsicherheit aufaddieren, sondern sich auch gegenseitig kompensieren können. Genauso wie wir eine User Story zu hoch geschätzt haben, werden wir eine andere User Story zu niedrig geschätzt haben. Intuitiv lässt sich also leicht nachvollziehen, dass die Unsicherheit für die Schätzung des Product Backlog insgesamt niedriger ist als die Summe aller Einzelunsicherheiten für die Schätzung der User Stories.

Die Statistik liefert uns hierfür sogar eine quantitative Aussage. Vielleicht wissen Sie noch aus der Wahrscheinlichkeitsrechnung, dass die Varianz der Summe von Zufallsvariablen gleich der Summe der Varianz der einzelnen Zufallsvariablen ist. Das hat zur Folge, dass die Standardabweichung der Backlog-Größe im Vergleich zu den einzelnen User Stories relativ klein ist. Sehen

wir uns unser Beispiel an: Am Ende des Projekts umfasste das Product Backlog 343 User Stories mit einer Gesamtgröße von insgesamt 770 Story Points. Die durchschnittliche User-Story-Größe betrug somit 2,2. Nehmen wir an, dass die Standardabweichung der Schätzfehler für die einzelnen User Stories bei 2 Story Points liegt (das sind immerhin 91%). Die Varianz für die Größe des gesamten Product Backlog ist somit $343 \bullet 2^2 = 1372$. Die Standardabweichung für das gesamte Backlog beträgt also $\sqrt{1372} = 37$. Das ist nur 5% der Backlog-Größe!

Die Story Points in der Projektplanung und -steuerung

Beim Estimation Game wird der relative Umfang der User Stories ermittelt. Der Produkt Owner kann mit diesen Informationen die Prioritäten der User Stories festlegen und die Freigabeplanung für die Sprints machen. Der Produkt Owner ordnet die User Stories mit dem besten Verhältnis zwischen Umfang und Kundennutzen oben im Backlog an. Umfangreiche User Stories mit geringem Kundennutzen landen weit unten. Unserer Erfahrung nach liefert das Estimation Game genügend Input, um das Backlog so ordnen zu können.

Um den Fertigstellungstermin für das Produkt vorherzusagen, benötigen wir noch die Abarbeitungsgeschwindigkeit, die in Story Points pro Sprint gemessen wird. Es ist normalerweise sehr schwierig, die Abarbeitungsgeschwindigkeit vor Projektbeginn einzuschätzen. Der früheste Zeitpunkt, an dem eine einigermaßen zuverlässige Prognose möglich ist (und dies auch nur bei besonders günstigen Bedingungen), liegt nach der ersten Sprintplanung.

Allerdings fordern die meisten Projektsponsoren und Kunden schon zu Projektbeginn eine klare Aussage von Ihnen, wie lange das Projekt dauern wird. Wir halten es aus unserer Erfahrung trotzdem für empfehlenswert, nur wenig Arbeit in Schätzungen des Arbeitsaufwands zu investieren. Setzen Sie stattdessen das Estimation Game ein, um die Umfänge ihrer Backlog-Einträge abzuschätzen. Versuchen Sie anschließend, die Abarbeitungsgeschwindigkeit zu schätzen, wobei Sie die Erfahrung des Teams berücksichtigen müssen: Wie hoch war die Geschwindigkeit des Teams in früheren Projekten? Ist das aktuelle Projekt ähnlich? Muss sich das Team in eine neue Technologie oder einen neuen Anwendungsbereich einarbeiten?

Wir vertreten die Meinung, dass man die Frage nach dem Fertigstellungstermin auch durch zeitintensive Aufwandsschätzungen nicht genauer beantworten kann. Es ist unserer Ansicht nach besser, die Zeit in den ersten Sprint zu investieren, denn so erhält man ab dem ersten Tag (Sprint-Planung) Hinweise bezüglich der Geschwindigkeit. Dabei halten wir es für sehr sinnvoll, dem Auftraggeber die Vorgehensweise zu erläutern, um Vertrauen zu schaffen. Die regelmäßigen Präsentationen, bei denen das Team dem Auftraggeber den jeweils aktuellen Entwicklungsstand des Produkts zeigt, steigern das Vertrauen des Auftraggebers in die Entwicklungsmannschaft von mal zu mal mehr.

Projekterfahrungen

Das übergreifende Burn-down-Diagramm (Bild 1) zeigt den Entwicklungsfortschritt über alle Teams in unserem Projekt. Neben der eigentlichen Burn-down-Kurve ("Remaining") enthält das Diagramm auch

den Burn-up ("Accepted") und die Größe des Backlog ("Planned"). Für die Zeitskala wurden Kalendermonate gewählt. Die vertikale Achse stellt den Leistungsumfang gemessen in Story Points dar.

Das übergreifende Burn-down-Diagramm ist das wichtigste Projektmanagement-Werkzeug für den Product Owner. Er kann damit extrapolieren, wann das Projekt beendet sein wird oder welcher Funktionsumfang implementiert sein wird, wenn die zur Verfügung stehende Zeit vorbei ist.

Das Diagramm unseres Projekts (Bild 1) zeigt einige interessante Punkte. Die Größe des Backlog (s. Kurve "Planned") bezieht sich auf die User Stories, die der Product Owner für das Projekt vorgesehen hatte. Zu Beginn des Projekts war die Abarbeitungsgeschwindigkeit sehr niedrig. In dieser Phase wurde ein neues Team aufgebaut, das mit der Vorentwicklung begann. Der Product Owner befürchtete deshalb, dass der Funktionsumfang nicht realisiert werden könnte und strich User Stories. Im fünften Monat kamen planmäßig zwei weitere Teams dazu und auch das erste Team konnte seine Geschwindigkeit steigern. Der Product Owner wurde wieder optimistischer und erweiterte das Backlog. Kurz vor Projektende wurde klar, dass der Kunde mit dem Produktumfang dennoch nicht zufrieden sein würde. Dank einer Umplanung der sehr umfangreichen Systemtestphase im Gesamtprojekt konnten in unserem Teilprojekt zwei zusätzliche Sprints durchgeführt werden. Somit konnte der Product Owner die vom Kunden gewünschten User Stories noch ins Backlog aufnehmen.

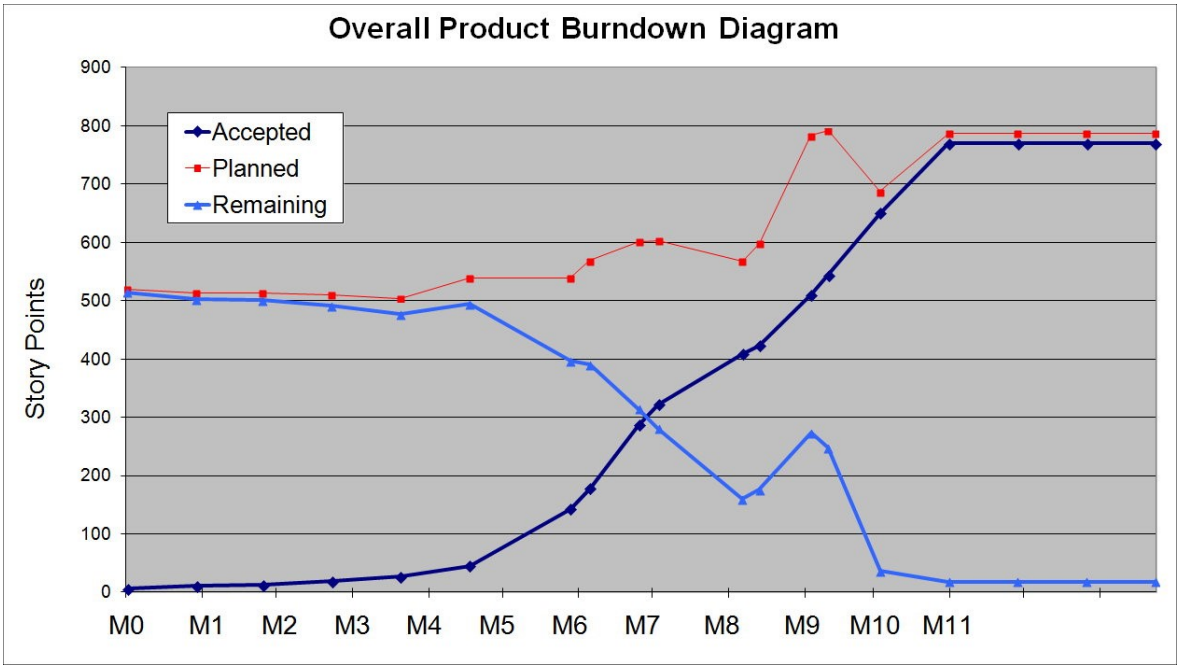


Bild 1: Burndown-Diagramm des Projekts.

Am eindrucksvollsten war für uns die Linearität des Burn-up (s. Kurve "Accepted" in Bild 1) ab dem Zeitpunkt, zu dem alle Teams am Projekt mitarbeiteten. Wir hatten eine sehr gleichbleibende Geschwindigkeit. Die einzige wirkliche Abweichung war im Monat M8: Dort hatten wir ein großes Refactoring durchzuführen, weil neue Anforderungen hinzukamen, die wir bei der Projektplanung nicht vorhergesehen hatten.

Fazit: Besser Spielen als Aufwände schätzen

Das Team Estimation Game kostet wenig Aufwand. Es liefert dem Product Owner die relative Größe seiner User Stories in hinreichender Genauigkeit, um Prioritäten definieren und die Freigabeplanung für die Sprints durchführen zu können. Das Estimation Game ist außerdem ein hervorragendes Team Building Event und fördert den Wissensaustausch innerhalb des Teams.

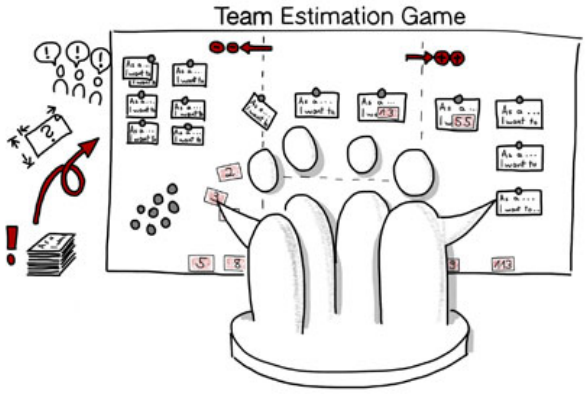
Den Umfang der User Stories in Story Points abzuschätzen schafft bei den Projektbeteiligten ein besseres Verständnis bezüglich der Abarbeitungsgeschwindigkeit als die traditionelle Schätzung des Arbeitsaufwands in Personentagen. Als Projektmanager oder Auftraggeber sollten Sie die Abarbeitungsgeschwindigkeit in Ihrem Projekt sorgfältig überwachen, da sie für die Projektplanung und die Prognose von Terminen essenziell ist. Für den Scrum Master ist diese Kennzahl für die Leistungsfähigkeit des Teams ebenfalls wichtig, da er für die Optimierung der Arbeitsabläufe verantwortlich ist.

Unserer Einschätzung nach ist es wertvoller, mit der Entwicklung so früh wie möglich zu beginnen, um die ersten Indikatoren für die Geschwindigkeit zu erhalten. Vergeuden Sie deshalb Ihre kostbare Zeit nicht damit, umfangreiche Aufwandsschätzungen durchzuführen.

Literatur

- Bockman, Steve: Team Estimation Game, Agile 2007 Conference, August 2007, zitiert nach: Blog "Agile Software Development, Training and Coaching", http://stevebockman.com/blog/about/
- Cohn, Mike: User Stories Applied: For Agile Software Development, Addison-Wesley Professional, 2004.
- Cohn, Mike: User Agile Estimating And Planning, Pearson Education, 2006.
- Linssen, Oliver: Agile Aufwandsschätzung in Scrum. Planning Poker – Techniken, Erfahrungen und Empfehlungen, projektmagazin, 10/2012.
- Larman, C.; Vodde, B.: Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum, Pearson Education, 2008.
- Reinertsen, D. G.: The Principles of Product Development Flow: Second Generation Lean Product Development, Celeritas Publishing, 2009.
- Wirdemann, Ralf: Agile Softwareentwicklung mit Scrum und User Stories, projektmagazin, 2/2010.

Team Estimation Game



Das Team Estimation Game dient der relativen Aufwandsschätzung von Product Backlog Items bzw. von Arbeitspaketen eines Projekts in kleinen Gruppen. Die Methode wird hauptsächlich in agilen Vorgehensmodellen angewendet (z.B. SCRUM) und ermöglicht eine direkte Schätzung durch die Mitglieder des Umsetzungsteams. Die Methode basiert auf der Annahme, dass detaillierte Aufwandsschätzungen für die einzelnen Aufgaben eines Product Backlog Items, bzw. eines Arbeitspakets, nicht praktikabel sind. Grund dafür ist, dass die Tätigkeit der Schätzung selbst sehr aufwendig ist, da die Aufgaben bzw. Backlog Items hierzu in schätzbare Einzeltätigkeiten zerlegt werden müssen. Die relative Schätzung der User Storys durch ein Team verspricht demgegenüber eine hohe Zeitersparnis bei vergleichbar genauem Ergebnis.

Einsatzmöglichkeiten

Das Team Estimation Game wird für die Aufwandsschätzung aus zwei Perspektiven eingesetzt:

- Bestimmung der in einem Sprint durchführbaren Arbeitspakete (nach dem Minimalprinzip: "Wie viele User Storys können wir im kommenden Sprint fertig stellen?")
- Bestimmung des Aufwands zur Abarbeitung eines vollständigen Product Backlogs (folgend dem Maximalprinzip: "Wie viel Aufwand bedeutet die Abarbeitung aller Product Backlog Items?")

Vorteile

- hohe Genauigkeit der Schätzung durch die kollektive Betrachtung gegenüber individuellen Schätzungen

- Einschätzung aus unterschiedlichen Perspektiven, je nach Zusammenstellung des Teams
- geringerer Aufwand als die Schätzung durch Zerteilung der User Storys, bzw. der Arbeitspakete in Teilaufgaben
- einfaches Setup
- kurzes, intuitiv verständliches Regelwerk
- Neue Teammitglieder mit Kenntnis der Methode können direkt in Schätzrunden eingebunden werden.
- Durch die Gestaltung als Spiel wird die aktive Mitarbeit in der Gruppe gefördert (aktives Feedback statt einseitiger Vortrag).

Grenzen, Risiken, Nachteile

- Einmal eingeführt kann die Methode schwer ersetzt werden. Andere Schätzmethoden verwenden andere Bezugsgrößen und Bewertungsmetriken.
- Ab einer Gruppengröße von mehr als neun Personen dauert eine Runde mehrere Minuten. Der Vorteil der schnellen Abarbeitung einzelner User Stories bzw. Arbeitspakete entfällt.
- Ohne ausreichenden fachlichen Hintergrund der Teilnehmer wird die Schätzung spekulativ.

Ergebnisse

- **Nach Aufwand geordnete User Storys.** Die User Storys werden anhand des geschätzten Aufwandes (niedrig – hoch) an einer Tafel oder auf einer anderen Fläche angeordnet.
- **Story-Points für jede geschätzte User Story.** Die einzelnen User Storys des Product Backlogs aus dem Eingangskanal wurden bezüglich Ihres Aufwands geschätzt.
- **Hinweise auf Unsicherheiten und mangelnde Klarheit bei den Anforderungen.** Während des ersten Durchgangs aufkommende Unklarheiten werden geklärt oder zur Klärung zurückgestellt.

Voraussetzungen

- ein Scrum Team, bzw. eine Gruppe von Personen, die für die Abarbeitung der User Storys in Frage kommt
- Besprechungsraum, in dem die Schätzungen ungestört durchgeführt werden können; ausreichend Fläche zur Ablage und Sortierung der einzelnen User Storys.

Benötigte Informationen

- vom Product Owner priorisiertes Product Backlog für die Schätzung eines Projekts

- Sprint Backlog für die Überprüfung eines Sprints
- Expertise der Teilnehmer für die zu schätzende Aufgabenstellung.

Benötigte Hilfsmittel

- Moderationskarten mit den User Storys des Product bzw. Sprint Backlogs
- Pinnwand zur Sammlung und Sortierung der geschätzten User Storys. Alternativ können die Karten auch an der Wand angebracht oder auf einem großen Tisch bzw. dem Boden ausgelegt werden.
- Moderationsmaterialien (Karten, Stifte, Pin-Nadeln, Klebestreifen, Flip-Chart) zur Bearbeitung und Darstellung der Ergebnisse
- Karten mit Zahlenwerten (z.B. Fibonacci-Zahlen) zur Bewertung der User Storys im späteren Verlauf. Die Zahlenwerte entsprechen der Messgröße Story Point und werden mit den User Storys als Ausgangsgröße übergeben

Durchführung

Schritt 1: Bereiten Sie das Team Estimation Game vor!

Überprüfen Sie als Moderator / Scrum Master mit dem Product Owner, ob alle User Storys in der aktuellen Fassung vollständig vorliegen. Die Methode umfasst mindestens die Schätzung eines Sprint Backlogs bis hin zur Schätzung eines Product Backlogs.

Stellen Sie sicher, dass alle zu behandelnden User Storys auf Moderationskarten gedruckt / geschrieben werden. Gesammelt in einem Kartenstapel bilden sie die Grundlage der nun folgenden Schätzung. Eine spezielle Sortierung der User Stories ist nicht erforderlich.

Benennen Sie bei der Einladung zum Team Estimation Game das zu schätzende Vorhaben (Ihr Projekt, optional auch die einzelnen User Storys). Ergänzen Sie bei der erstmaligen Anwendung Hinweise zur Methode Team Estimation Game. Planen Sie, sofern die Methode bei den Teilnehmern unbekannt ist oder erstmalig angewendet wird, Zeit für eine kurze Erklärung ein. Eine ausführliche Trainingsrunde ist aufgrund des einfachen Regelwerks nicht erforderlich.

Achten Sie darauf, dass das Projektteam vollständig vertreten ist. Die Teamrollen im Detail:

- Product Owner: Er vertritt die Seite des Auftraggebers, priorisiert das Product Backlog und steht den Schätzern zur Klärung von Fragen bzgl. der User Stories zur Verfügung. Der Product Owner nimmt keinerlei Einfluss auf die Schätzung: er gibt weder Schätzungen ab noch beeinflusst er die Teilnehmer.
- Scrum Master: Er organisiert den Termin, achtet auf die Einhaltung der Spielregeln und moderiert den Ablauf des Team Estimation Games. Der Scrum Master nimmt ebenfalls keinen Einfluss auf die Schätzung.
- Entwickler: Sie realisieren später die User Storys. Durch ihre Expertise aus anderen Projekten liefern sie den essentiellen Beitrag zum Team Estimation Game. Die Entwickler liefern die Schätzungen, ggf. geben auch Spezialisten Schätzungen ab (s.u.).
- Spezialisten: Ist Ihnen vorab bekannt, dass bei den Entwicklern fachliche Wissenslücken vorhanden sind, können Sie die Runde durch Experten für diese Themen ergänzen. Diese unterstützen das Team, indem sie fachliche Rückfragen beantworten und beteiligten sich bei denjenigen User Stories an der Schätzung, für die sie Experten sind.

Der Scrum-Master schlägt die Reihenfolge der Teilnehmer vor (z.B.: reihum oder freiwillige Meldung) und initiiert die erste Schätzrunde.

Schritt 2: Die erste User Story wird platziert

Die erste Person aus dem Team liest diese laut vor und positioniert sie danach mittig auf der vorgesehenen Fläche (Boden, Wand, Whiteboard, Flipchart...).

Schritt 3: Zwei weitere User Storys werden platziert

Die zweite Person aus dem Team liest die nächste Story laut vor und positioniert sie links oder rechts neben der ersten Story:

- LINKS neben einer Story bedeutet, dass der Spieler den Aufwand seiner Story geringer schätzt als den Aufwand der bereits positionierten Story.
- RECHTS neben einer Story bedeutet, dass der Spieler den Aufwand seiner Story höher einschätzt.

Hinweis: Üblich ist die Platzierung von links nach rechts für steigenden Aufwand. Sie können alternativ die Karten auch von unten (niedriger Aufwand) nach oben (hoher Aufwand) positio-

nieren. Welche Anordnung Sie wählen, ist für die Methode irrelevant, Sie sollten aber die Richtung während des Spiels unbedingt beibehalten, um Verwirrungen zu vermeiden.

Der am Zug befindliche Spieler erklärt den Grund für seine Einordnung in wenigen Sätzen. Alternativ kann auf eine Erklärung verzichtet werden. Die Entscheidung wird bei diesem Schritt nicht diskutiert. Klärende Verständnisfragen zwischen Entwicklern und Spezialisten sind erlaubt.

Der dritte Spieler platziert die dritte User Story in gleicher Weise.

Schritt 4: Erweitern Sie das Regelwerk!

Erweitern Sie jetzt die Spielregeln, ab der vierten User Story ist zusätzlich möglich:

- Der am Zug befindliche Spieler kann darauf verzichten, eine neue Story zu platzieren und stattdessen eine bereits abgelegte Story verschieben. Er kann z.B. die Story A, die zunächst links von Story B war, nach rechts verschieben, so dass sie sich jetzt rechts von Story B befindet. Der Spieler kann diese Aktion ggf. kurz erklären, aber es gibt hierzu keine Diskussion.
- Die Person am Zug kann aussetzen, wenn sie keine Änderung durchführen möchte und alle User Storys platziert sind.

In diesem Schritt werden auch User Storys mit ähnlichem Aufwand ausschließlich in die Reihe einsortiert, es gibt nicht die Möglichkeit, die Aufwände für zwei User Storys als gleich oder ähnlich zu kennzeichnen. Dies erfolgt erst in Schritt 6.

Schritt 5: Die Spieler platzieren die verbleibenden User Storys

Führen Sie nun die Aufwandsschätzung nach diesem Regelwerk mit allen verbliebenen User Storys durch. Kann eine User Story aufgrund fehlender Informationen nicht geschätzt werden, wird dies vom Scrum Master / Moderator dokumentiert und die User Story zurück an den Product Owner gegeben. Dies gilt auch für die Zuordnung der User Stories zu Zahlenwerten im folgenden Schritt.

Das Anordnen der User Storys und die damit verbundene Aufwandsschätzung sind abgeschlossen, wenn:

- alle User Storys auf der vorgesehen Fläche angebracht wurden.
- alle beteiligten Personen aussetzen, also alle Teilnehmer mit den Schätzungen zufrieden sind.

Schritt 6: Bewertung Sie die User Storys mit Zahlenwerten!

Nun tragen Sie entlang Ihrer sortierten User Storys die vorbereiteten Zahlenkarten auf, so dass eine Skala entsteht. Positionieren Sie die Karte mit dem kleinsten Zahlenwert bei der User Story, deren Aufwand am geringsten geschätzt wurde. Dementsprechend sollte der größte Zahlenwert bei der User Story mit der höchsten Aufwandsschätzung zu finden sein.

Zwischen diesen beiden Extremwerten gilt es nun, die einzelnen User Storys zu gruppieren und diese Gruppen jeweils einer Zahl zuzuordnen. Es ist hierbei Ihre Entscheidung, ob Sie die User Storys in der Anordnung aus Schritt 5 belassen, oder ob sie identische Aufwände neu sortieren (z.B. untereinander bei der üblichen Links-Rechts-Sortierung). Folgendes Beispiel zeigt die Zuordnung bei unveränderter Sortierung:

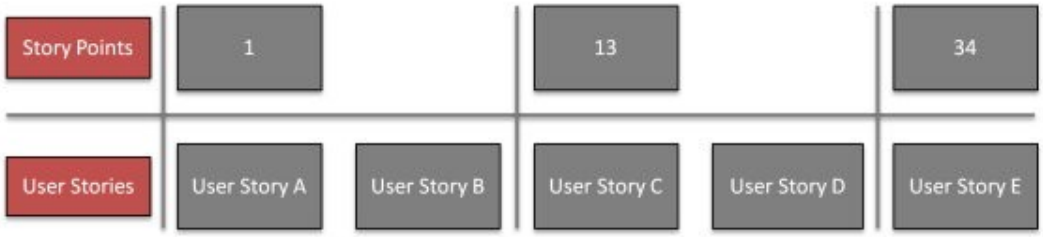


Bild 1: Ergebnis des Team Estimation Games

Hinweis: Als Zahlenskala bietet sich die Fibonacci-Folge (1, 1, 2, 3, 5, 8, 13 usw.) an, da diese nach oben starke Unterscheidungen zwischen den Zahlenwerten aufweist. Dies ermöglicht eine klare Unterscheidung zwischen subjektiv als aufwendig oder weniger aufwendig eingeschätzten User Storys.

Die Teammitglieder führen die Zuordnungen ebenfalls reihum durch. Pro Person und Schätzung (= eine Runde) ist jeweils entweder eine Zuordnung oder eine Umsortierung möglich (analog zu Schritt 5). Die Runden zur Schätzung enden, wenn:

- jeder User Story ein Wert zugeordnet ist,
- alle Teilnehmer passen, d.h. mit der Bewertung einverstanden sind.

Schritt 7: Übergeben Sie die Ergebnisse an die Schnittstellen!

- Überführen Sie die Ergebnisse (User Storys inklusive Konsens-Schätzungen) in das Product Backlog.
- Übersetzen Sie die Bewertungsergebnisse in die Ressourcen- und Einsatzplanung und / oder Ihr Risikomanagement.
- Geben Sie Ihre Dokumentation der Unklarheiten an den Product Owner weiter (fachlich zu den einzelnen User Storys und / oder die Anforderung eines entsprechenden Spezialisten), sofern dies nicht während der Durchführung geschehen ist.

Ergänzende / ähnliche Methoden

- Planning Poker
- Magic Estimation
- User Story
- Scrum

- Delphimethode
- Projektkalkulation

Praxistipps

Gehen Sie pragmatisch vor

Bei der Einordnung der User Storys auf der Spielfläche ist es nicht wichtig, ob eine User Story A "nur etwas" mehr bzw. weniger aufwendiger ist als eine User Story B. Machen Sie das den Teilnehmern klar und beenden Sie jede Diskussion darum. Wichtig ist die pure Einordnung in: größer, kleiner oder (in Schritt 6) gleich aufwendig.

Achten Sie auf Ihre Skala

Idealerweise können Sie die Einordnungen der User Storys in Schritt 6 der Methode in sieben bis neun Bereiche vornehmen. Dies hat sich in vielen Berichten als praktikabel erwiesen.

Auf jeden Fall müssen Sie damit rechnen, dass die Skala bei späteren Schätz-Workshops verbreitert wird, da neue Aufgaben über die Enden der bisherigen Skala hinaus eingeordnet werden. Gründe hierfür können z.B. sein:

- Es liegt bereits die Schätzung eines Product Backlogs vor. Während einer späteren Schätzung eines Sprint Backlogs wird erkannt, dass die bestehende Zahlenskala nicht ausreicht.
- Wenn mehrere Sprints nacheinander geschätzt werden, ergibt sich durch bewusste oder unbewusste Vergleiche mit den vergangenen Sprints die Notwendigkeit, die Skala zu erweitern.
- Ein Sprint wird abgebrochen und nach einem Review eine neue Schätzung durchgeführt. Eine oder mehrere User Storys werden aufgrund der neuen Erkenntnisse außerhalb der bisherigen Skala einsortiert.

Varianten

Dynamic Team Estimation

Dynamic Team Estimation ist für größere Teams geeignet, die bereits das Team Estimation Game beherrschen (bis drei mal neun Personen, neun entspricht der maximalen Größe einer Team Estimation Game-Gruppe). Im Gegensatz zur klassischen Variante werden beim Dynamic Team Estimation die User Storys des Product Backlogs nicht sequentiell, sondern parallel bearbeitet. Dies bedeutet, dass mehrere User Storys in kleinen Arbeitsgruppen geprüft und Argumente ausgetauscht werden.

Teilen Sie vor Beginn die zu behandelnden User Storys unter den Teams gleichmäßig auf. Die Abgabe der Schätzungen erfolgt an einer gemeinsamen Fläche. Dazu werden zu Beginn des Dynamic

Team Estimation Games aus jeder Gruppe drei Repräsentanten ausgewählt (entsprechend der maximalen Größe drei mal drei Personen). Diese sind für die Abgabe der Schätzungen verantwortlich.

Wichtig: Wenn ein Teilnehmer eine User Story auf der Arbeitsfläche platziert und die Gründe hierfür vorträgt, unterbrechen alle Arbeitsgruppen ihre Tätigkeit, um die Platzierung und die zugehörigen Argumente wahrzunehmen.

Die Abweichung vom stark sequentiellen Ansatz des Team Estimation Game ermöglicht einen höheren Durchsatz an User Storys. Der Nachteil am Dynamic Team Estimation ist der hohe Anspruch an die Disziplin der Teilnehmer. Die Unterbrechung der Tätigkeiten und Konzentration auf die Argumente und Einordnung einer einzelnen User Story muss schnell von statten gehen, um den Vorteil dieser Methode auszuspielen. Dies erfordert eine gezielte Moderation und notfalls einen Eingriff in das Geschehen bei Missachtung dieser zusätzlichen Spielregel.

Herkunft

Die Methode geht auf Steve Bockman zurück, der sie erstmals 2008 anwendete (Agile Unlimited, http://agileunlimited.com/whoweare.php, zuletzt besucht am 19.8.2015).

Autor

Daniel Reinold
Erstellt am: 18.09.2015