Fachbeitrag

Agiles Projektmanagement

Häufige Stolperfallen in Scrum

Scrum ist ein agiles Projektmanagement-Framework, das nur wenige Rollen Besprechungen und Berichte verpflichtend vorgibt. Anstatt viele Prozesse und Dokumente im Detail vorzuschreiben, setzen agile Methoden darauf, das richtige Maß an Prozess und Formalität zwischen allen Beteiligten iterativ auszuhandeln: Der Schwerpunkt liegt dabei auf der Kommunikation.

Bild 1 zeigt das Prozessbild von Scrum. Eine detaillierte Beschreibung der Rollen und Abläufe in Scrum finden Sie in dem Artikel "Agiles Projektmanagement. Scrum – eine Einführung" von Ralf Wirdemann (Projekt Magazin 21/2009).

Was ein erfolgreiches Scrum-Team ausmacht

Auf den ersten Blick scheint Scrum als Software-Entwicklungsprozess bestechend einfach: Es gibt nur einige wenige Rollen, eine überschaubare Menge an Regeln und so gut wie keine zu erstellenden Dokumente. In der Praxis zeigt sich jedoch immer wieder, dass eine erfolgreiche Umsetzung von Scrum nicht trivial ist. Gerade weil die Methode als solche wenig formell ist und damit kaum Vorgaben macht, hängt der Erfolg wesentlich von den beteiligten Personen ab.

Das perfekte Team ...

Ein erfolgreiches Team zeichnet sich dabei vor allem durch eine hohe Lernbereitschaft, das Bestreben kontinuierlich besser werden zu wollen und eine offene Feedback-Kultur aus. Einem solchen Team gelingt es, am Ende jeder Iteration eine potentiell releasefähige Version des Produkts auszuliefern. Dieses Zwischenergebnis, das sogenannte "Produkt-Inkrement", ist dabei implementiert, integriert, dokumentiert und erfolgreich getestet. Ein perfektes Team ist außerdem eingebettet in eine Umgebung, die den Prozess unterstützt, fördert und die damit verbundenen agilen Werte lebt.

Autor

Thomas Lieder

Diplom-Informatiker, Certified Srum Master, Senior Project Manager bei der Setzwein IT-Management GmbH

Kontakt: thomas.lieder@setzwein.com

Mehr Informationen unter:
› projektmagazin.de/autoren

Autor

Katja Roth

Dipl.-Inform., PM-Fachfrau IPMA Level D, Certiefied Scrum Master, Certified Scrum Product Owner, Senior Project Manager bei der Setzwein IT-Management GmbH

Kontakt: katja.roth@gmx.de

Mehr Informationen unter:
› projektmagazin.de/autoren

ähnliche Artikel

in der Rubrik:

› Agiles Projektmanagement

... und die Wirklichkeit

Doch leider ist nicht jedes Team erfolgreich und die Ursachen sind häufig auch erst auf den zweiten Blick zu erkennen. Bild 2 ergänzt das Scrum-Prozessbild um einige der häufigsten Probleme. Diese Stolperfallen werden im Folgenden beschrieben, die Ursachen analysiert und Wege zur Verbesserung der Anwendung von Scrum gezeigt. Natürlich ersetzt dies nicht die detaillierte Analyse der jeweils konkret vorliegenden Situation, kann aber erste Hinweise auf potentielle Probleme und ihre Lösung geben.

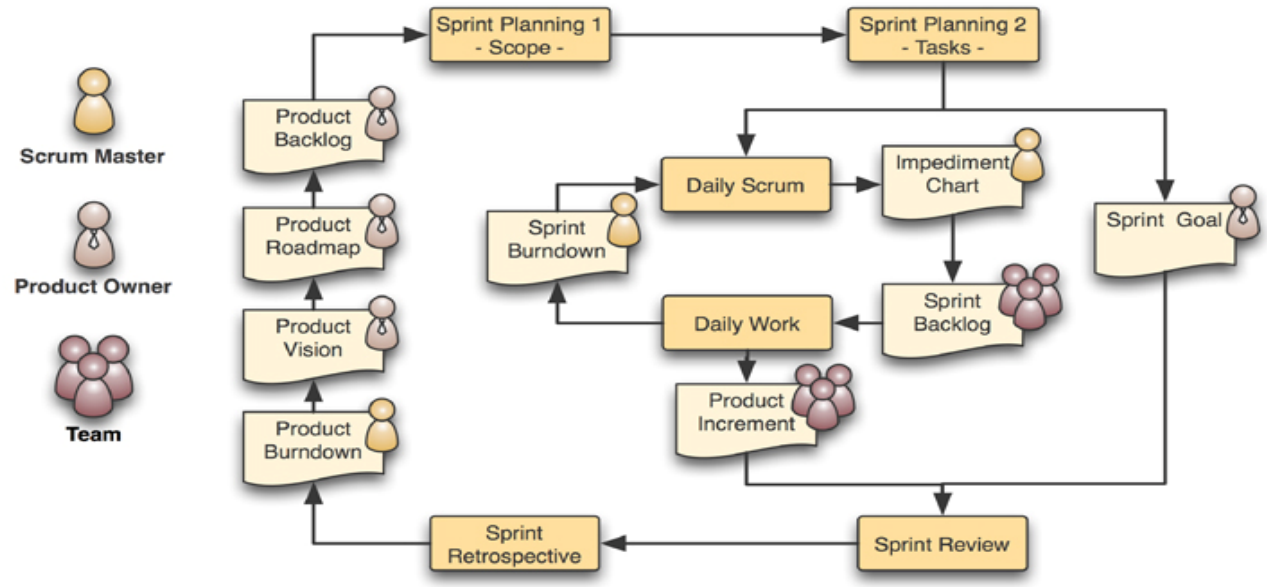


Bild 1: Scrum-Prozessbild.

Das Rollenverständnis des Scrum Masters

In einem Scrum-Team ist der Scrum Master für die Einhaltung des Prozesses verantwortlich. Er stellt sicher, dass sich das Team ungestört auf die Erstellung des Produkt-Inkrements konzentrieren kann. Nicht selten gibt es Probleme an dieser zentralen Stelle: Zum einen mit dem Rollenverständnis des Scrum Masters selbst, zum anderen mit den Aufgaben, die er aufgrund seiner Rolle durchführen muss.

Direkte Verstöße gegen die Rollendefinition – z.B. das Einmischen in den Sprint durch die Verteilung neuer, für den Sprint nicht eingeplanter Aufgaben – lassen sich in der Regel einfach feststellen und durch entsprechende Schulungen beheben. Es gibt aber auch Verhaltensweisen eines Scrum Masters, deren negative Wirkung sich eher indirekt zeigt, da er sich vordergründig entsprechend der Rollen-Definition verhält. Sie äußern sich z.B. darin, dass das Team in der Retrospektive wiederholt dieselben Probleme schildert oder dass aufgetretene Störungen, sog. "Impediments", vom Scrum Master nicht beseitigt werden. Das "Impediment Chart", d.h. die Störungsliste, wird infolge dessen immer länger.

Diese Symptome sind unter anderem dann zu beobachten, wenn der Scrum Master gegenüber der Organisation nicht durchsetzungsfähig ist. Sind die Führungsqualitäten des Scrum Masters nicht ausreichend ausgeprägt, kann er z.B. die in der Organisation liegenden Störungen nicht beseitigen. In einem solchen Fall hilft ein Coaching des Scrum Masters, sowohl bezüglich seiner Rolle als auch bezogen auf Mittel und Wege, Dritte zur Mitarbeit und damit zur Beseitigung der Impediments zu bewegen. Im einfachsten Fall mag es reichen, konstruktive Vorschläge zur Lösung eines Problems zu machen, z.B. feste Sprechzeiten vorzuschlagen, wenn ein Mitglied des Scrum-Teams auch als Know-how-Träger für andere benötigt wird. In anderen Fällen, z.B. wenn zugesagte Ressourcen nicht zur Verfügung gestellt werden, ist vielleicht eine konstruktive Eskalation im Lenkungsausschuss notwendig.

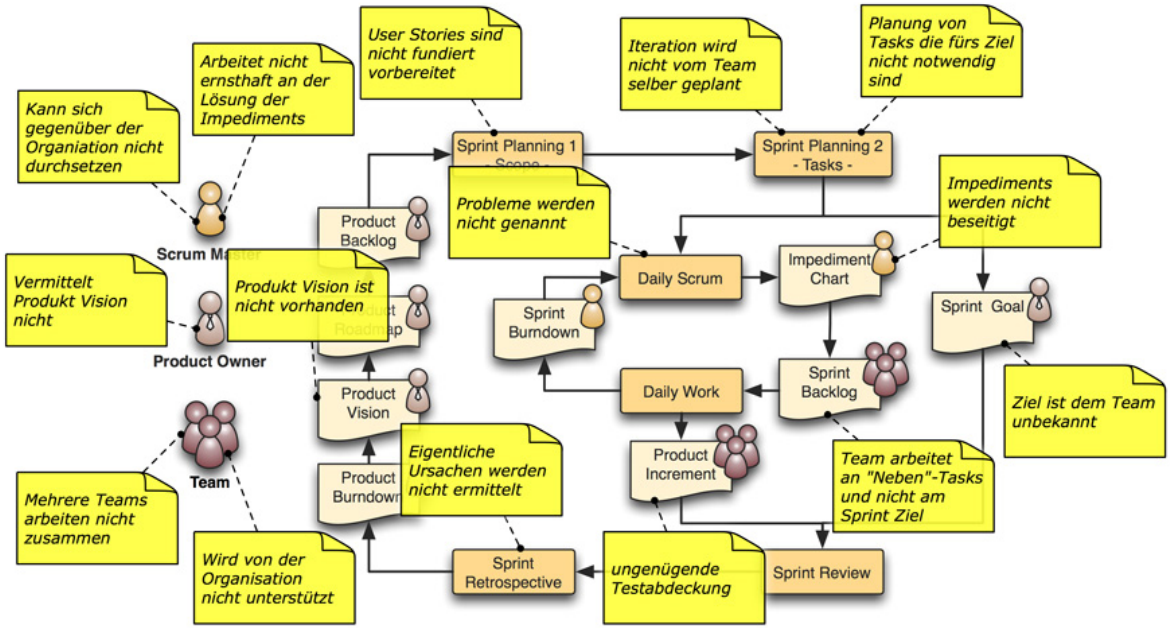


Bild 2: Einige der typischen Probleme bei der Umsetzung von Scrum.

Die ganze Organisation muss dahinter stehen

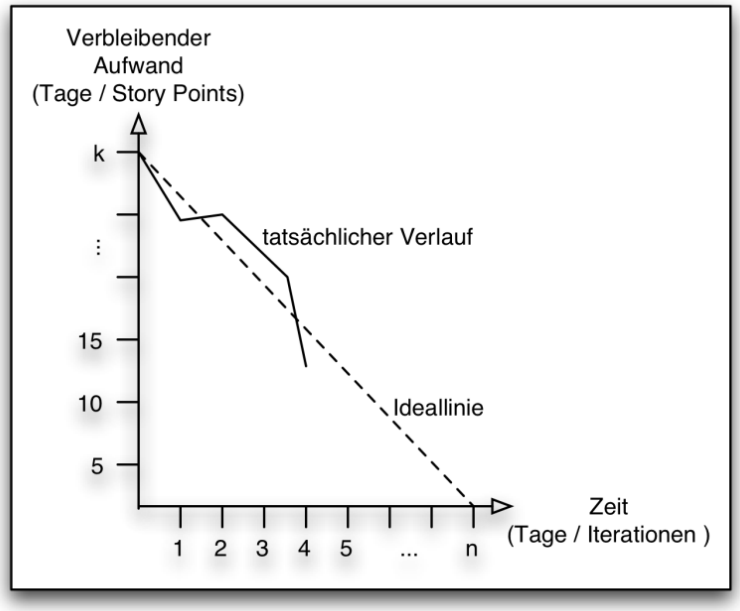
Sollte die Organisation, in der das Projekt durchgeführt wird, insgesamt jedoch nicht hinter der Methode Scrum stehen und die Einführung eher kritisch sehen, lassen sich die Impediments alleine durch den Scrum Master häufig nicht lösen. Hier gilt es, wie bei jedem Change-Projekt, Aufklärungsarbeit zu leisten, die Organisation von dem Vorgehen Scrum zu überzeugen und eine für Scrum funktionierende Umgebung zu schaffen.

Diese Aufklärungsarbeit sollte dabei mindestens von der Managementebene ausgehen, die das Vorgehen für das Projekt ausgewählt hat. Je höher in der Hierarchie die Förderer angesiedelt sind, desto höher ist die Wahrscheinlichkeit, Neuerungen in der Organisation anstoßen zu können. Es gilt natürlich wie bei jeder Änderung der Orga-

nisation, dass im Zweifelsfall auch Rückendeckung durch ein Mitglied des oberen Managements oder der Geschäftsleitung notwendig ist.

Die Charts im Auge behalten

Doch auch, wenn die Organisation dem Vorgehen positiv gegenübersteht, kann es zu einem Anwachsen des Impediment Charts kommen. Dies ist insbesondere dann der Fall, wenn der Scrum Master die in Retrospektive und Daily Scrum genannten Störungen für nicht so wichtig hält und sich nicht ernsthaft um deren Beseitigung kümmert. Dies geschieht häufig dann, wenn der Scrum Master auch noch andere Aufgaben wahrnimmt, etwa wenn er zur Hälfte Scrum Master und zur anderen Hälfte Entwickler ist. Hier gilt es, das Rollenverständnis des Scrum Masters zu schärfen. Nur so lassen sich Impediments auch wirklich beseitigen und das Vertrauen in den Scrum Master und die Methode aufrecht erhalten.



Burndown Charts

Burndown Charts zeigen den Restaufwand für einen Sprint (Sprint Burndown Chart) oder ein Release (Release Burndown Chart) im Zeitverlauf.

Sprint Burndown Chart

Die vertikale Achse zeigt den Restaufwand gemessen in Tagen oder Stunden im aktuellen Sprint. Die horizontale Achse stellt die Tage des Sprints dar.

Release Burndown Chart

Die vertikale Achse stellt den Restaufwand gemessen in Story Points zu Beginn jedes Sprints dar. Die horizontale Achse zeigt die Sprints eines Release.

Jeder Scrum Master sollte als höchstes Ziel verfolgen, eine ideale Arbeitsumgebung für das Team herzustellen. Ein anwachsendes Impediment Chart oder wiederkehrende Impediments sind ein deutliches Zeichen dafür, dass der Scrum Master seiner Rolle nicht gerecht wird. Ob dies am Scrum Master selbst oder an äußeren Einflussfaktoren liegt, lässt sich nur in direkten Gesprächen oder durch einen externen Scrum Coach in einem Prozess-Review ermitteln. Um solche Probleme zu erkennen, ist es sinnvoll, im Statusbericht eines agilen Projekts neben dem Burndown Chart (s. Kasten) auch immer das Impediment Chart und die gerade dringendsten Probleme zu betrachten.

Mit fundierten Retrospektiven Problemursachen beheben

Gleichermaßen problematisch ist eine nicht fundiert durchgeführte Retrospektive. In einer solchen schildert das Team Störungen, die nur Symptome des eigentlichen Problems sind. Die wahren Ursachen werden vom Scrum Master jedoch nicht herausgearbeitet. Ein Beispiel hierfür wäre das wiederholte Verfehlen des Sprint-Ziels aufgrund von nicht-geschätztem Mehraufwand: Zur Lösung dieses Problems nimmt sich das Team dann vor, beim nächsten Mal besser zu schätzen, um so ein realistisches Commitment abgeben zu können. Die wirklichen Ursachen für das Verfehlen des Sprint-Ziels, beispielsweise Probleme im Umgang mit der Architektur oder fehlendes Fachwissen, werden aber nicht erkannt und daher auch nicht gelöst.

Somit werden lediglich die Symptome kuriert. Die eigentlichen Probleme verlagern sich möglicherweise, werden aber nicht beseitigt. Zur Lösung dieser Situation bietet sich ein Coaching des Scrum Masters in Bezug auf Moderations- und Fragetechniken an. Durch den gezielten Aufbau der entsprechenden Soft Skills erkennt der Scrum Master die wirklichen Ursachen eher und kann die Probleme dann auch lösen. Ein Training in der Methode Scrum alleine hilft in diesem Fall hingegen nicht.

Herausforderung "Commitment im Team"

Scrum basiert auf der Idee, dass sich ein Team pro Iteration auf die Umsetzung einer Reihe von Anforderungen verpflichtet. Das Team arbeitet dabei selbstorganisiert: Es plant sein Arbeitsvolumen für jede Iteration selbst und ist daher auch für die Erreichung des selbstgesteckten Ziels verantwortlich.

Ein fehlendes Commitment im Team in Bezug auf das Sprint-Ziel äußert sich u.a. darin, dass sich das Team ablenken lässt und sich nicht auf die Bearbeitung der Aufgaben auf dem Taskboard fokussiert. "Nebenkriegsschauplätze" sind wichtiger als die Realisierung dessen, was angefordert wurde. Architektur- bzw. Technologiefragen haben in solchen Situationen zumindest für Teile des Teams eine größere Bedeutung als die Umsetzung der Anforderungen, auf die sich das Team im Sprint Planning verpflichtet hat. Das KISS-Prinzip ("Keep it small and simple") wird dabei gerne ignoriert, sodass folglich Zeit mit Lösungen verbracht wird, die so nicht angefordert waren. Dies alles führt in Summe dazu, dass das Sprint Burndown Chart keinen wirklichen Fortschritt zeigt und sich der Restaufwand nicht weiter verringert.

Die Produkt-Vision im Team kommunizieren

Die Ursachen für ein fehlendes Commitment im Team sind vielschichtig. Möglicherweise gelingt es dem Product Owner nicht, die Produktvision zu verkaufen. Das Team identifiziert sich dann nicht mit dem Produkt und arbeitet wenig begeistert an den anstehenden Aufgaben. Die Erfahrung zeigt, dass ein Team dann besonders motiviert ist, wenn ihm zum einen die Vision bekannt ist und zum anderen transparent wird, wie die anstehenden User Stories zum Erreichen der Visionsziele beitragen. Dies ist in erster Linie eine kommunikative Aufgabe des Product Owners. Sollte er sich nicht in der Lage sehen, die Vision zu vermitteln, etwa weil eine solche nicht explizit formuliert wurde, muss dies nachgeholt werden. Ansonsten wäre es denkbar, dass jedes Teammitglied seine eigene Vorstellung von dem entwickelt, was umgesetzt werden soll.

Eine gute Produktvision grenzt dabei das Produkt ab, ohne es zu spezifizieren. Ein Beispiel: "Das zu entwickelnde Produkt ist eine Plattform für den Online-Aktienhandel, die den An- und Verkauf von Aktien ermöglicht. Mindestens 100 Benutzer sollen gleichzeitig über die Plattform arbeiten können." Mit dieser Vision weiß jedes Teammitglied, worum es geht. Jeder kann abschätzen, was für die Entwicklung relevant ist, im Beispiel also der An- und Verkauf von Aktien. Gleichermaßen gibt die Vision vor, was nicht wichtig ist, im Beispiel etwa der An- und Verkauf von Fonds. Ohne eine solche Vision und damit ohne ein konkretes Ziel vor Augen sollte kein Projekt – und damit auch kein agiles Projekt – durchgeführt werden.

Iterations-Planung – nicht ohne das Team

In der Iterations-Planung definieren die Beteiligten den Umfang der Iterationen und gliedern die User Stories in Tasks. Geben Scrum Master oder Product Owner die Planung vor, ist eine Identifikation des Teams mit der Iteration ebenfalls unwahrscheinlich. Die Motivation des Teams leidet in diesem Fall, da es nicht in die Planung mit einbezogen wurde. Infolge dessen wird sich das Team als vorrangiges Ziel die Bearbeitung der vorgegebenen Tasks setzen, sich jedoch nicht einer qualitativ akzeptablen Umsetzung der zugrunde liegenden User Stories verbunden fühlen. Im schlimmsten Fall sind dem Team die eigentlichen Ziele unbekannt, so dass es Fehler oder Auslassungen in den Tasks nicht einmal erkennen könnte.

In einer solchen Konstellation lässt man das Wissen und das Engagement des Teams mindestens teilweise ungenutzt. Vorrangig ist deshalb dafür zu sorgen, dass das Team die Iterations-Planung eigenverantwortlich durchführt. Ist dies nicht der Fall, weil sich Scrum Master oder der Product Owner nicht an ihre Rollenvorgabe gehalten haben, bieten sich hier entsprechende Schulungen mit hohem Praxisanteil an, in denen insbesondere die Scrum-Rollen und deren Aufgaben geübt werden.

Das Team als Ganzes einschwören

Es kommt auch vor, dass einzelne Team-Mitglieder ihre eigenen Interessen verfolgen und sich daher nicht der Umsetzung der vom Product Owner vorgestellten Anforderungen verpflichtet fühlen. Es besteht dann die große Gefahr, dass Aufgaben im Sprint Planning identifiziert werden, die zwar das Teammitglied, nicht aber das Projekt weiterbringen. Ein Beispiel hierfür wäre die Umsetzung einer Anforderung mit einer neuen Technologie, obwohl die Umsetzung auch mit den vorhandenen Bordmitteln möglich wäre. In einem solchen Fall kann es ratsam sein, Maßnahmen zur Teambildung durchzuführen, um das Team als Ganzes auf das Projekt und die Vision einzuschwören. Dabei empfiehlt es sich, die gewohnte Arbeitsumgebung zu verlassen.

In der Praxis hat es sich beispielsweise bewährt, das Team zunächst spielerisch auf die Vision einzustimmen, etwa durch das gemeinsame Malen eines Bildes zur Visualisierung der Vision. Gleichermaßen sollte ein solcher Workshop Elemente integrieren, die etwaigen Differenzen im Team entgegenwirken. Hier könnte man unter anderem mit einem paradoxen Ansatz arbeiten und dem Team die Frage stellen, was geschehen müsste, damit die Situation noch schlimmer wird. Auf diese Weise gelingt es die Atmosphäre zu lockern, um dann konstruktiv Ver-

besserungsmaßnahmen abzuleiten. Bleibt der Erfolg dieser teambildenden Maßnahmen aus, sollte man das Team verstärken oder einige Teammitglieder austauschen.

Unterstützung durch das Umfeld

Wird einem Team die erforderliche Unterstützung seitens der Organisation verweigert, fühlt es sich im Stich gelassen und es lässt sich ebenfalls kein echtes Commitment erzielen. Damit ein Team motiviert arbeiten kann, muss es die Unterstützung seines Umfeldes erfahren. Führt jedoch jeder Wunsch zu Blockaden und kann kein Impediment gelöst werden, entsteht beim Team der Eindruck, dass das Projekt und seine Ziele unwichtig sind. Wenn wir davon ausgehen, dass kein Unternehmen unwichtige Projekte durchführt, handelt es sich hierbei also in erster Linie um eine Aufgabe des Scrum Masters: Er muss gegenüber der Organisation verdeutlichen, wie wichtig es ist, die Impediments zu beseitigen.

Das Team muss außerdem die Unternehmensstrategie kennen und das Projekt als einen Teil davon betrachten, um motiviert arbeiten zu können. Es ist daher empfehlenswert, dem Team von vornherein die Bedeutung des Projekts auch im Hinblick auf die Unternehmensstrategie klar zu machen. Aus der Einbettung des Projekts in die Strategie folgt im Umkehrschluss, dass das Team entsprechende Unterstützung durch die Organisation tatsächlich erfahren muss: Wichtige Impediments müssen auch tatsächlich aufgelöst werden.

Fehlerhafte Schätzung mit Planning Poker

Eine besondere Herausforderung in Scrum sind die Schätz-Workshops, in denen das Team mit dem sogenannten Planning Poker die Größe von User Stories schätzt. Sicheres Zeichen für eine nicht-optimale Scrum-Umsetzung ist unter anderem eine lange Dauer solcher Planungssitzungen. Denn eine gut vorbereitete Sitzung zeichnet sich dadurch aus, dass die User Stories hinreichend detailliert vorgestellt werden und dem Team die Gesamtfachlichkeit und die Systemarchitektur bewusst sind. Nachfragen beziehen sich nur auf Details und können schnell geklärt werden. Entstehen aber in der Planungssitzung lang andauernde Diskussionen um Inhalt oder Umsetzung einer User Story, so ist die Story entweder schlecht vorbereitet oder das Wissen über das System im Team nicht gleichmäßig verteilt. Beides sollte in einem funktionierenden Scrum-Umfeld nicht vorkommen.

Fehlerhafte Schätzungen aufgrund falsch eingeschätzter Komplexität sind ebenso ein Symptom für eine schlecht durchgeführte Schätzklausur. Die Velocity, also die Geschwindigkeit des Teams, ist definiert durch die Anzahl der Story Points, die ein Team pro Sprint umsetzen kann (näheres zu Velocity und Story Points erfahren Sie in dem Artikel "Agile Softwareentwicklung mit Scrum und User Stories", Projekt Magazin 2/2010). Weichen erwartete und tatsächliche Velocity voneinander ab, ist das ein weiterer Hinweis darauf, dass das Team die Komplexität der Anforderungen nicht richtig eingeschätzt hat.

Planning Poker

Mit Planning Poker wird die Größe von User Stories relativ zueinander in Story Points geschätzt.

Die Regeln

Jedes Teammitglied bekommt einen Kartenstapel, wobei die Kartenwerte einer unreinen Fibonacci-Folge (0, 1, 2, 4, 8, 13, 20, 40, 100) entsprechen. Zu Beginn wählen die Spieler die User Story aus, die eine der kleinsten aus dem Product Backlog ist und weisen dieser 2 Story Points zu. Diese User Story ist fortan die Referenz-User-Story und ist bei jeder Schätzung zu berücksichtigen. Anschließend werden die übrigen User Stories nacheinander vorgestellt. Die Teilnehmer schätzen die Größe durch Auswahl der Karte, die dem Aufwand entspricht, der in dieser User Story relativ zur Referenz-Story vermutet wird. Die Karten werden dabei solange verdeckt gehalten bis sich jeder Spieler für eine Karte entschieden hat. Nach dem Offenlegen der Karten erläutern die Spieler mit der höchsten und der niedrigsten Schätzung ihre Wahl. Nach einer kurzen Diskussion findet eine erneute Schätzrunde statt. Es folgen weitere Diskussions- und Schätzrunden bis ein Konsens gefunden ist.

Story Points richtig verstehen

Die Schwierigkeiten mit dem Planning Poker haben einerseits technische Ursachen, können aber andererseits auch fachlicher Natur sein. So ist bei der Durchführung des Planning Pokers unbedingt darauf zu achten, dass die Maßeinheit Story Point richtig verstanden wird. Mit dieser Einheit schätzt man die Größe der User Stories und zwar nicht absolut, sondern relativ zueinander. Außerdem muss jeder Schätzende die gemeinsame Referenz-Story kennen und bei seiner Schätzung berücksichtigen.

Aus fachlicher Sicht scheitert das Planning Poker unter anderem dann, wenn es keine Teammitglieder gibt, die sich im Fachgebiet auskennen und somit die Komplexität der zu schätzenden Anforderungen zuverlässig beurteilen können. Infolge dessen wird die Schätzung zwangsläufig fehlerhaft sein.

Erfolgreiches Schätzen nur mit guter Vorbereitung

Aber auch wenn Fach-Experten im Team sind, hängt der Erfolg einer Planungssitzung von der Vorbereitung durch den Product Owner ab. Dieser muss die User Stories so formulieren, dass sie spezifisch, messbar, attraktiv, rational und testbar sind (in Anlehnung an das SMART-Prinzip). Zur Sicherstellung der Testbarkeit bietet sich die Formulierung von Akzeptanzkriterien an. Diese beschreiben anhand konkreter Beispiele, unter welchen Bedingungen eine User Story als erfolgreich umgesetzt gilt.

Es ist nicht zielführend, mit einer nur vagen Idee eines Features in diese Sitzung zu gehen und dort die Details gemeinsam zu erarbeiten. Das Team kann die Komplexität einer User Story besser einschätzen, wenn die Story vom Product Owner bereits zusammen mit Abnahmekriterien und Testfällen präsentiert wird. Dies hilft auch, um die Auswirkungen einer Änderung auf andere betroffene Systemteile zu analysieren. Workshops zur detaillierten Ausarbeitung von Features sind durchaus sinnvoll, sollten aber unabhängig von und vor der Planning-Poker-Sitzung durchgeführt werden.

Fachwissen ist unerlässlich

Um den Planungs-Workshop technisch korrekt durchzuführen, bietet sich ein Training sowohl des Product Owners als auch des Teams an. Hat ein neu zusammengestelltes Team noch kein Fachwissen aufgebaut und muss es eine Referenz-User-Story erst noch definieren, bietet es sich an, eine Exploration durchzuführen. In einer solchen Phase werden zur Klärung grundlegender Fragen einige wenige sehr kurze Iterationen durchgeführt. Die Exploration schließt mit einer ausführlichen Retrospektive, um eine Basis für dann folgende reguläre Iterationen zu bilden. Im Rahmen einer Exploration könnte z.B. eine Funktion, die vom neuen System bis in das anzubindende Altsystem reicht, implementiert werden.

In manchen Fällen, z.B. wenn das Team insgesamt unerfahren in der anstehenden Fachlichkeit ist, kann es auch hilfreich sein, externe Fachexperten hinzuzuziehen. Im Falle einer veränderten Team-Zusammensetzung sollte die Referenz-User-Story neu definiert werden. Hier kann es sinnvoll sein, diese von einem Kern-Team, das aus bereits erfahrenen Entwicklern besteht, bestimmen zu lassen. Sie kennen sich im Fachgebiet bereits aus und müssten die Komplexität der Anforderungen einschätzen können. Die so definierte User Story mit den daraus abgeleiteten Tasks muss aber allen Teammitgliedern bekannt und die damit verbundene Komplexität verstanden sein.

Fehlende Testabdeckung und Testautomatisierung

Will man Scrum erfolgreich anwenden, muss man auf eine ausreichend hohe Testabdeckung achten. Nur so lassen sich die kontinuierlichen Änderungen und Erweiterungen am System bei einer gleichbleibend hohen Qualität durchführen.

Es ist denkbar, dass eine bereits lauffähige Funktionalität durch Weiterentwicklungen wieder fehlerhaft wird. Bei ungenügender Testabdeckung kann es dann passieren, dass diese Fehler nicht rechtzeitig erkannt werden und in den produktiven Betrieb gehen. Um nach jeder Iteration eine releasefähige Version des Produkts bereitstellen zu können, sollte die Testabdeckung kontinuierlich überprüft und insbesondere die bereits lauffähige Funktionalität früherer Iterationen durch entsprechende Tests abgesichert werden.

Scheiternde Reviews als Warnung verstehen

Ein weiteres Signal dafür, dass in die Testabdeckung investiert werden sollte, sind regelmäßig scheiternde Reviews, in denen eigentlich die umgesetzten Anforderungen abgenommen werden sollen. Denn nur mit einer guten Testabdeckung lässt sich sicherstellen, dass auch kurz vor der Abnahme angepasster Code lauffähig ist. Dabei ist sowohl auf eine ausreichend hohe Abdeckung der durch die Entwickler erstellten Unit-Tests zu achten, als auch auf eine kontinuierliche Weiterentwicklung einer automatisierten Regressions-Testsuite. Unter einer Regressions-Testsuite versteht man in der Softwareentwicklung die regelmäßige Wiederholung aller oder einiger Testfälle, um durch Code-Korrekturen entstandene Fehler in bereits getesteten Softwareteilen zu entdecken.

Eine kontinuierlich abnehmende Velocity des Teams weist darauf hin, dass es Probleme mit der Wartbarkeit des produzierten Codes gibt. Um eine gute Velocity zu erhalten, ist es erforderlich, die Software kontinuierlich zu

refaktorisieren (Restrukturierung des Quellcodes). Ein Team wird sich vorzugsweise dann trauen, Überarbeitungen und Änderungen durchzuführen, wenn die Funktionalität mit automatisierten Tests abgesichert ist. Eine ungenügende Automatisierung der Tests kann auch Inkonsistenzen in der Versionsverwaltung nach sich ziehen, da Fehler nicht schnell genug auffallen. Oft ist es dann nicht mehr möglich, einen fehlerfreien Code aus der Versionsverwaltung zu exportieren und eine lauffähige Version der Software zu erstellen.

Automatische Tests anstelle überflüssiger Features

Die einfachste Ursache einer mangelnden Testabdeckung ist das Fehlen einer Verpflichtung zur Erstellung von Tests. Die Lösung ist in diesem Fall, dass das Scrum-Team die Definition of Done anpasst. Dabei handelt es sich in der agilen Softwareentwicklung um eine Checkliste, die definiert, wann fertig wirklich fertig ist. Dazu gehört typischerweise nicht nur die lauffähige Software, sondern auch die Erstellung der entsprechenden Dokumentation sowie die verpflichtende Erstellung von Tests. Diese Definition of Done erstellt das Team für sich selbst und wird daher auch vom Team gepflegt. Auch wenn mehrere Teams parallel arbeiten, sollten teamspezifische Definitionen die Ausnahme sein.

Eine Ursache dafür, dass automatische Tests vielfach nicht vorhanden sind, ist der in vielen Projekten gelebte hohe Featuredruck. Die Zeit für die Erstellung von Tests wird eingespart, um möglichst viel neue Funktionalität zu erstellen. Mit Scrum versucht man jedoch, sich auf das wirklich Wichtige zu konzentrieren und Verschwendung in Form unnötiger Funktionen zu vermeiden. Bezogen auf den hohen Featuredruck heißt das, dass die Notwendigkeit von Features geprüft und nur wirklich wichtige umgesetzt werden sollten. Eine Priorisierung der Anforderungen durch den Product Owner ist also notwendig, um neben der Umsetzung von Funktionen auch Zeit für die Erstellung von Tests zu haben und damit qualitativ hochwertige Software produzieren zu können.

Eigene Testsysteme lohnen sich

Können Tests nur auf einem Entwicklungssystem durchgeführt werden, kann häufig keine wirklich verlässliche Aussage über die Qualität der Software getroffen werden, da die verwendeten Testdaten nicht repräsentativ sind. Standardisierte Testdaten und dedizierte Testsysteme sind jedoch eine wichtige Voraussetzung für zuverlässige Tests. Die Erfahrung zeigt, dass eine vielleicht anfänglich abschreckend hohe Investition in den Aufbau von Testsystemen mit für die Testfälle ausreichenden Testdaten sich mittelfristig lohnt. Durch die Möglichkeit der Testautomatisierung werden mehr Tests erstellt und durchgeführt, was einen sehr positiven Einfluss auf die Qualität und Wartbarkeit der erstellten Software hat.

Mangelnde Zusammenarbeit mehrerer Teams

Die optimale Größe eines Scrum-Teams liegt bei ca. 7 bis 12 Personen. Werden mehr Personen zur Realisierung eines Projekts benötigt, ist es notwendig, mehr als ein Scrum-Team parallel arbeiten zu lassen. Modelle für die Skalierung von Scrum sind schon häufig beschrieben worden und sollen hier nicht im Fokus stehen. Neben dem auch bei Scrum höheren Verwaltungsaufwand für ein größeres Projekt, z.B. durch ein größeres Product Backlog und mehr Stakeholder, ergeben sich durch mehrere Teams auch typische Probleme im Prozess.

Direkt beobachtbar werden Unstimmigkeiten zwischen Teams spätestens dann, wenn sich Teams übereinander beschweren. Nicht so offensichtlich ist es hingegen, wenn nur die Qualität des Codes abnimmt. Häufig kommt es zu Inkompatibilitäten und Mehrfach-Entwicklungen und die teamübergreifende Integration des Quellcodes wird zunehmend schwieriger. Letztendlich nimmt die Velocity der einzelnen Teams ab. Dies kann so weit gehen, dass immer häufiger das Sprintziel nicht erreicht wird.

Architectual Board – regelmäßige Abstimmungstreffen der Teams

Begründet sind alle diese Probleme aber nicht in der fehlenden Skalierbarkeit von Scrum an sich, sondern in einer unzureichenden Zusammenarbeit der Teams. Wie diese verbessert werden kann, hängt wiederum von den jeweiligen Ursachen ab. Fehlt z.B. eine Plattform, in der sich die Teams über die Gesamtarchitektur austauschen können, sollte ein entsprechendes "Architectural Board" eingerichtet werden. Dies empfiehlt sich auch, um eine gegebenenfalls vorhandene Unternehmensarchitektur in das Projekt zu tragen.

Das Board dient der regelmäßigen Abstimmung der Teams untereinander auf einer technischen Ebene. Es geht also nicht in erster Linie darum, welche Fachlichkeit von welchem Team realisiert wird, sondern welche Änderungen am System vorgenommen werden und wie dies ggf. andere Teams beeinflusst. Ebenso kann das Board genutzt werden, um offene Architekturentscheidungen gemeinsam zu treffen.

Die Organisation eines Architectural Boards muss den Gegebenheiten des jeweiligen Projekts angepasst werden: Sind viele hochqualifizierte Entwickler in allen Teams vorhanden, kann jedes Team zu jedem Treffen jeweils einen anderen Entwickler ins Board entsenden. Will man auf der anderen Seite nicht immer wieder die gleichen Diskussionen führen, spricht vieles dafür, eine feste Zuordnung vorzunehmen. Dabei besteht wiederum die Gefahr, dass damit die Homogenität innerhalb eines Scrum-Teams aufgebrochen wird und sich implizite Hierarchien bilden.

Ebenso wie die Zusammensetzung muss auch die Häufigkeit der Abstimmungstreffen im Projektkontext entschieden werden. Sinnvoll ist sicherlich ein Treffen im Zusammenhang mit der Sprint-Planung, um die sich daraus ergebenden Fragen zu klären. Ob auch innerhalb einer Iteration Treffen sinnvoll sind, hängt sehr davon ab, wie stark die Arbeiten der Teams einander beeinflussen.

Scrum of Product Owner – Iterationen teamübergreifend planen

Probleme anderer Art treten auf, wenn nicht nur mehrere Teams sondern auch mehrere Product Owner eingesetzt werden. Es besteht dann die Gefahr, dass die Teams einander widersprechende Anforderungen umsetzen sollen und es zu Konflikten bei der Integration kommt. So kann es zum Beispiel vorkommen, dass durch die beauftragten User Stories Änderungen am zugrunde liegende Objektmodell vorgenommen werden, die untereinander nicht kompatibel sind.

Ebenso kommt es vor, dass der Product Owner Zuarbeiten des eigenen Teams für ein anderes als nicht so wichtig einstuft. Dies hat zur Folge, dass die Arbeit des betroffenen Teams behindert wird. Lösen lassen sich solche

Widersprüche nur durch eine integrierte, teamübergreifende Iterationsplanung. Hierbei werden die Inhalte der Backlogs der verschiedenen Teams und auch die jeweiligen Iterationsziele aufeinander abgestimmt. Diese Iterationsplanung wird am besten gepaart mit einem "Scrum of Product Owner"-Meeting, in dem die tägliche Arbeit aufeinander abgestimmt wird.

Scrum of Scrums – Tägliche Abstimmung der Scrum Master

Arbeiten mehrere Teams parallel, ist es nur natürlich, dass mehrere Teams durch die gleichen Hindernisse gestört werden. So werden alle Teams von denselben Infrastrukturproblemen betroffen sein, wenn eine Testumgebung für die kontinuierliche Integration fehlt. Um solche Störungen effizient zu beseitigen und den Teams eine Gelegenheit für die organisatorische Abstimmung zu geben, sollte ein "Scrum of Scrums", also ein tägliches Scrum-Meeting der Scrum Master eingesetzt werden.

Nicht eindeutig beantworten lässt sich hingegen die Frage, wie denn ein Projekt auf mehrere Teams aufgeteilt werden sollte. Nahe liegend ist sicherlich die Aufteilung entlang von Systemgrenzen, also z.B. getrennte Teams für Frontend und Backend. Ebenso legen standortübergreifende Projekte eine Aufteilung der Teams je Standort nahe. Eine solche Aufteilung führt aber in der Regel zur Verstärkung einer schon vorhandenen Spezialisierung mit dem Ergebnis, dass sich jedes Team nur noch in seinem Bereich auskennt.

Dies wiederum widerspricht fundamental der Idee der Collective Code Ownership, also dem Prinzip, dass jedes Teammitglied gleichermaßen für den gesamten Code verantwortlich ist und diesen auch jederzeit verändern kann und darf. Aus diesem Grund sollte bei standortübergreifenden Teams in die Kommunikationsinfrastruktur investiert werden. Dies führt mit hoher Wahrscheinlichkeit zu homogenen Teams und einer konsistenten Plattform.

Fazit

Auch wenn agile Vorgehensweisen auf den ersten Blick bestechend einfach sind, ist eine problemfreie Umsetzung nicht garantiert. Viele der geschilderten Probleme treten jedoch in fast allen Projekten – unabhängig von der verwendeten Methode – auf und auch ihre Lösungen sind nicht spezifisch für agile Vorgehensweisen. Durch die starke Fokussierung auf die Umsetzung einer Aufgabe innerhalb eines festen Zeitrahmens werden die geschilderten Probleme bei Verwendung agiler Methoden aber viel früher deutlich. Die einfachste Möglichkeit, die hier beschriebenen Probleme zu vermeiden, ist die Qualifikation der Mitarbeiter zu verbessern, in Scrum also die Qualifikationen des Product Owner, des Scrum Master und der Teammitglieder. Je vertrauter die Mitarbeiter mit agilen Methoden sind, desto eher fällt ihnen selbst auf, dass ein Problem vorliegt, und desto eher finden sie auch aus dem Projekt heraus eine passende Lösung.

Neben der reinen Lehre spielen im agilen Kontext aber auch die mit dem Vorgehen verbundenen Werte eine entscheidende Rolle. Agile Methoden setzen in besonderem Maße bestimmte Werte voraus wie Offenheit und Vertrauen, vor allem aber Kommunikation: Viele Elemente des Vorgehens beruhen im Kern darauf, dass man miteinander redet. Dieser hohe Anteil an Kommunikation ist nur in einer von Vertrauen und Offenheit geprägten Unternehmenskultur möglich.

Das agile Vorgehen versucht darüber hinaus, durch iteratives Vorgehen Komplexität zu beherrschen und Lernen zu ermöglichen. Lernen setzt aber die Möglichkeit voraus, trotz Unsicherheit Entscheidungen zu fällen und Fehler machen zu dürfen. In einer Unternehmenskultur, die von Angst vor Fehlern geprägt ist, wird sich daher ein agiles Vorgehen nicht erfolgreich umsetzen lassen.

Bild 3: Agile Werte.

Werteunterschiede sind häufig die eigentliche Ursache von Problemen, dies gilt auch für agile Methoden. Viele der in diesem Artikel beschriebenen Probleme sind in ihrem Kern Probleme der Kommunikation: Kommunikation untereinander, innerhalb einer Organisation und übereinander. Eine Atmosphäre, in der eine offene und vertrauensvolle Kommunikation stattfinden kann, lässt sich jedoch nicht per Dekret herbeiführen. Sie setzt vielmehr ein Bekenntnis zu den agilen Werten voraus, wie in Bild 3 dargestellt. In diesem Sinne ist der agile Wertekanon auch der Schlüssel zur Lösung der hier geschilderten Stolperfallen: In dem Maße, wie ein Unternehmen bereit ist, sich ernsthaft auf die den agilen Methoden zugrunde liegenden Werte einzulassen, wird auch die Wahrscheinlichkeit einer erfolgreichen Implementierung agiler Methoden in dem Unternehmen steigen.