Denn sie wissen nicht was sie tun – Den Überblick über agile Backlogs behalten

Agile Teams organisieren sich heutzutage mittels Company-, Product-, Sprint-, Release-, und persönlichen Backlogs. Diese Menge von unterschiedlichen Backlogs wirft Fragen auf: Was ist Inhalt der einzelnen Backlogs? Wie werden die Backlogs von wem sinnvoll verwendet? Wie hängen die Backlogs zusammen? Eine Herausforderung ist die Suche nach Lösungen vor allem für mittlere und große Organisationen, die mit verteilten Teams, in Projekten ihre Produkt(weiter)entwicklung agil planen wollen. In diesem Artikel beantworten wir diese Fragen exemplarisch. Wir liefern Ihnen wichtige Hinweise und Erfahrungen aus der Praxis, damit Sie für sich die richtigen Antworten finden.

Manifest für Agile Softwareentwicklung [Agi01]

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

- Individuen und Interaktionen mehr als Prozesse und Werkzeuge
- Funktionierende Software mehr als umfassende Dokumentation
- Zusammenarbeit mit dem Kunden mehr als Vertragsverhandlung
- Reagieren auf Veränderung mehr als das Befolgen eines Plans

Das heißt, obwohl wir die Werte auf der rechten Seite wichtig finden,
schätzen wir die Werte auf der linken Seite höher ein.

Kasten 1: Agile Werte

Theorie versus Praxis

In der Software-Entwicklung stehen Unternehmen täglich vor neuen Herausforderungen, ihre Produktentwicklung effizient und profitabel zu gestalten. Dabei sind auf der Suche nach Lösungen, die sie dabei geeignet unterstützen. In den letzten Jahren setzen viele Branchen und Unternehmen unterschiedlichster Größen vermehrt auf den Einsatz agiler Vorgehensweisen. Betrachten wir jedoch die Voraussetzungen, die idealerweise für die Verwendung agiler Methoden vorhanden sein sollten, treffen wir diese vor allem in der Wirklichkeit großer Unternehmen nicht oder nur selten an.

Ideale Rahmenbedingungen

Agile Frameworks wie beispielsweise Scrum wünschen sich kleine Teams (3-9 Personen), die zusammen an einem Ort sitzen. Das Team bleibt über die gesamte Projektlaufzeit in der gleichen Konstellation bestehen und hat keine anderen Aufgaben neben dem laufenden Projekt. Alle Anforderungen, die das Team umzusetzen hat, werden von genau einer Person, dem Product Owner, verantwortet. Die Anforderungen werden in Form eines Product Backlogs verwaltet und hauptsächlich durch persönliche Kommunikation zwischen dem Team und dem Product Owner vermittelt. Der Product Owner steht jederzeit für Fragen zur Verfügung und übernimmt alle erforderlichen Klärungen. Der Planungs- und Bearbeitungshorizont wird auf Entwicklungsiterationen (Sprint) von 2-4 Wochen beschränkt. Am Ende jedes Sprints steht ein potentiell einsetzbares Produktinkrement zur Verfügung, das heißt, der erstellte Teil der Software ist getestet und integriert lauffähig (siehe auch [Sut11]).

Die Realität

In der Praxis ist es häufig schwer, diese Rahmenbedingungen vorzufinden oder zu schaffen. Unternehmen müssen eine Vielzahl von Projekten und Produkten gleichzeitig durchführen und entwickeln bzw. warten. Es gibt eine Fülle von Teams, die koordiniert werden müssen und viele Organisationseinheiten, die involviert sind. Trotzdem ist der Einsatz agiler Methoden auch in einem nicht idealen Umfeld Erfolg versprechend.

Eine gemeinsame Basis für agile Software-Entwicklung wurde 2001 durch das agile Manifest geschaffen. Namhafte Vertreter agiler Methoden schrieben darin fest, welche Werte und Prinzipien aus ihrer Sicht für die Software-Entwicklung wichtig sind (siehe Kasten 1). Diese Werte und Prinzipiensind in jedem Umfeld erstrebenswert und einsetzbar, unabhängig davon, ob sich Frameworks wie Scrum nicht 1:1 anwenden lassen.

Was ist ein Backlog und welche Backlogs gibt es?

Der Begriff „Backlog“ wird zum Beispiel in der Auftragsverwaltung verwendet und beinhaltet alle eingegangenen Aufträge geordnet nach ihrer Bearbeitungsreihenfolge. Über diese Liste werden alle nachfolgenden Prozessschritte wie Beschaffung und Produktion geplant und gesteuert.

Das „Backlog“ findet nun in der agilen Software-Entwicklung analoge Anwendung. Es enthält alle potentiell umzusetzenden Backlog Items geordnet nach der Reihenfolge ihrer Bearbeitung. Auf Basis dieser Liste erfolgt die Planung und Steuerung der Umsetzung.

In der Literatur zu agilen Themen treffen wir auf verschiedenste Ausprägungen des Begriffes Backlog: Aus Scrum bekannt sind das Product Backlog, das Sprint Backlog und das Release Backlog [Sut10], [Sut07]. Ken Schwaber nennt weiterhin ein Infrastructure Product Backlog [Sch10]. In aktuellen Veröffentlichungen finden wir weitere Arten von Backlogs, wie zum Beispiel das Company Backlog [Ren11] oder das Impediment Backlog [Wik11]. Nicht alle dieser Begriffe sind eindeutig definiert und verwendet. So finden wir beispielsweise bei [Lef10] die Begriffe Portfolio Backlog als Entsprechung zum Company Backlog und Programm Backlog für das Release Backlog. In unserer Projektpraxis stoßen wir auf weitere Typen von Backlogs, wie z.B. Service, Deployment und Integration Backlog oder auch das persönliche Backlog. Im Folgenden beschreiben wir daher eine Auswahl dieser Backlogs im Einzelnen und versuchen, diese gemäß unserer Praxiserfahrungen zu definieren.

1. Das Product Backlog

Den Begriff Product Backlog treffen wir vor allem im Scrum Kontext an. Gemäß Scrum Guide ist das Product Backlog eine geordnete Liste aller Dinge die in dem Produkt benötigt werden könnten und die einzige Quelle für Anforderungen an jegliche Änderung des Produktes. Es listet alle Features, Funktionen, Anforderungen, Verbesserungen und Bug Fixes auf und hat die Attribute “Beschreibung“, „Reihenfolge“ und „Schätzung“. Die Reihenfolge der einzelnen Backlog Einträge ergibt sich meist aus Wert, Risiko, Priorität und Notwendigkeit. Das Product Backlog enthält also alle Aufgaben und Anforderungen, die zur erfolgreichen Realisierung eines Produktes notwendig sind. Das Product Backlog ist allerdings nicht vollständig, es entwickelt sich vielmehr im Projektverlauf weiter (evolving Backlog). Neue Einträge kommen hinzu, andere entfallen.

Die Einträge, die an oberster Stelle stehen, werden weiter verfeinert und detailliert. Arbeiten mehrere Teams gemeinsam an einem Produkt, wird ein gemeinsames Product Backlog geführt. Die Zuordnung der Backlog Einträge zu den Teams erfolgt dabei durch eine Gruppierung über ein Attribut.

2. Das Company Backlog

Der Begriff „Company Backlog“ wird häufig als Synonym für ein gemeinsames Backlog für mehrere Teams verwendet. Das Company Backlog beinhaltet jedoch mehr als das.

Ein Company Backlog erlaubt es, die unternehmensweiten strategischen Ziele einer Unternehmung mit den geplanten Projekten in Zusammenhang zu bringen.

Aus diesem Company Backlog werden die einzelnen Product Backlogs abgeleitet. Zudem kann es für die Synchronisation des Produkt- und Projektmanagements eingesetzt werden. Es dient dann der Priorisierung und Steuerung verschiedener parallel laufender Projekte oder parallel zu entwickelnder Produkte.

3. Das Release Backlog

Das Release Backlog dient ebenfalls übergreifenden Planungsaspekten. Es beinhaltet ein Subset des Product Backlogs, das im nächsten Release geliefert werden soll, typischerweise mit einem Planungshorizont von drei bis sechs Monaten [Coh09]. Hierfür wählt der Product Owner Backlog Items aus dem Product Backlog aus und ordnet sie einem Release im Release Backlog zu. Das Release Backlog enthält nur die Backlog Items, die aufgrund strategischer Ziele in einem der kommenden Releases realisiert werden sollen. Das Product Backlog hingegen enthält auch Items von niedriger Priorität, die gegebenenfalls nie zur Realisierung gelangen.

4. Das Sprint Backlog

Das Sprint Backlog enthält die Inhalte des Product Backlogs, die für den nächsten Sprint ausgewählt wurden und die Planung für deren Umsetzung in Form von Tasks. Es liefert einen Forecast zu den Funktionalitäten, die im nächsten Produktinkrement umgesetzt werden und enthält alle Aufgaben, die dafür notwendig sind.

5. Das Integration Backlog (Systemtest)

Gerade in der Entwicklung komplexer Systeme, die aus mehreren Teilsystemen bestehen, muss zu jeder Änderung an einem der Teilsysteme eine Integration mit den anderen Teilsystemen durchgeführt werden. Da dies nicht in jedem Umfeld (man denke zum Beispiel an medizinische Geräte oder den Steuergeräteverbund in einem Auto) innerhalb des laufenden Sprints möglich ist, müssen die Integrationsaktivitäten zur Durchführung von Systemtests geplant und gesteuert werden. Hierfür kommt ein Integration Backlog zum Einsatz.

6. Das Deployment Backlog

Ist die Auslieferung eines Releases an den Endanwender mit größeren Aufwänden verbunden, können diese über ein Deployment Backlog geplant werden. Die Inhalte gehen dabei von der Bereitstellung von Schulungsmaterial und Trainings über Datenmigration bis zum Austausch einzelner Komponenten oder der gesamten Hardware.

7. Das Impediment Backlog

Das Impediment Backlog enthält alle aktuell identifizierten Hindernisse, die das Team bei der Umsetzung seiner Arbeit behindert und steuert bzw. plant deren Beseitigung.

8. Das Service Backlog (Input für Kanban)

Das Service Backlog enthält typischerweise Backlog Items, die zur Planung und Steuerung von Wartungsaspekten und zur Bereitstellung von Services im Rahmen eines SLAs oder OLAs benötigt werden. Diese Backlog Items können über Kanban-Prozesse abgearbeitet werden.

9. Das persönliche Backlog

Das persönliche Backlog eines Entwicklers speist sich aus den Tasks, die er aus dem Sprint Backlog übernimmt und allen weiteren Aufgaben, die er zu erledigen hat, die aber nicht im Sprint Backlog enthalten sind. Das können Aufgaben aus anderen Projekten sein, aber auch die Planung persönlicher Weiterbildung oder generelle Tätigkeiten wie Stundennachweise oder Beantwortung von Emails.

Während wir in kleinen Projekten mit nur einem Team mit einem Product Backlog und einem Sprint Backlog auskommen können, sehen wir, dass in anderen Kontexten eine Vielzahl von Backlogs miteinander zusammenspielen.In einem Scrum Projekt nach der reinen Lehre sind die Verantwortlichkeiten für das Product Backlog (der Product Owner) und das Sprint Backlog (das Development Team) klar geregelt.

Die Zuordnungen bei einer Vielzahl von Backlogs, Teams, Projekten und Produkten ist aber nicht mehr so einfach möglich. Neben der Verantwortlichkeit für die Inhalte des Backlogs stellt sich dann auch die Frage, wer die Backlogs letztlich für Planungsaspekte verwendet. Diesem Aspekt werden wir uns im Folgenden detailliert widmen.

ID	Backlog Item	Beschreibung
1	Goal	Ein Goal beschreibt Visionen und Ziele die durch die Ableitung von Requirements konkretisiert werden. Es wird nicht vorausgesetzt, dass ein Produkt- oder Projektkontext besteht.
2	Stakeholder Request	Ein Stakeholder Request wird zur Aufnahme von Änderungen genutzt. Ein Stakeholder Request ist ab der Auslieferung des Produktes/Releases relevant und zielt auf Änderungen dieser Auslieferungs-Stände aus Sicht des Stakeholders.
3	Change Request	Ein Change Request wird zur Steuerung von Änderungen genutzt. Fachliche Inhalte werden durch einen Change Request gebündelt, um eine fundierte Investitionsentscheidung treffen zu können und deren Umsetzung zu verfolgen. Ein Change Request ist ab der Auslieferung des Produktes/Releases bzw. dem Erreichen einen Hauptmeilensteines relevant und zielt auf Änderungen dieser Auslieferungs-Stände.
4	Change Order	Eine Change Order ist ein Arbeitsauftrag, der durch einen Change Request legitimiert ist. Die Change Order bündelt fachliche Inhalte zur teilweisen oder vollständigen Realisierung eines Change Request. Ein Change Request kann eine oder mehrere Change Orders enthalten. Eine Change Order richtet sich an den Umsetzungsverantwortlichen im Sinne eines Auftragnehmers. Zu einer Change Order können ein bis mehrere Backlog-Items zugeordnet sein.
5	Requirement	Ein Requirement beschreibt eine geforderte Charakteristik an ein zu entwickelndes System.
6	Feature	Ein Feature beschreibt eine geforderte Charakteristik an ein zu entwickelndes System. Das Feature beschreibt die Lösung, die zur Realisierung von übergeordneten Requirements notwendig ist.
7	User Story	Eine User Story beschreibt eine geforderte Charakteristik an ein zu entwickelndes System. Die User Story wird mit dem Muster <"Als <Rolle>, möchte ich <Ziel/Wunsch>, um <Nutzen>"> formuliert.
8	Defect	Ein Defect wird verwendet, um Fehler an der Implementierung oder der Dokumentation zu kanalisieren. Der Defect wird verwendet, wenn Entwicklungslieferungen das Entwicklungsteam verlassen haben, also ein so genannter Hand-over stattgefunden hat. Beispielsweise ist das der Fall, wenn Systemtests außerhalb des Entwicklungsteams durchgeführt werden. Defects sind bis zur Auslieferung des Produktes/Releases bzw. dem Erreichen einen Hauptmeilensteines relevant.
9	Issue	Ein Issue ist ein offener Punkt, der während der Implementierung identifiziert wurde. Der Issue muss innerhalb des Implementierungsteams gelöst oder nach außen zur Bearbeitung kommuniziert werden. Issues können in Form von Bugs, Impediments, Mitigations, Clarifications oder Action Items auftreten.
10	Test Set	Ein Test Set bündelt verschiedene Test Cases, die für einen Testlauf relevant sind. Neben funktionaler Komposition sind auch projektphasenspezifische und organisatorische (nach Verantwortlichkeiten) Kriterien zur Definition von Test Sets möglich. Ein Test Set beschreibt zudem welches Test-Environment und welche Testdaten für den Testlauf relevant sind. Ein Test Set ist oft die Basis für einen Testreport, der die Ergebnisse des Testlaufes interpretiert.
11	Test Case	Ein Test Case beschreibt einen elementaren, funktionalen Test, der der Überprüfung einer zugesicherten Eigenschaft (funktionale und/oder nicht-funktionale Charakteristik) eines Testobjektes dient.
12	Ticket	Ein Ticket adressiert dienstleistungsrelevante Themen, wie beispielsweise Serviceanfragen und -aufträge sowie Service-Störungen und wiederkehrende Service-Probleme.
13	Service	Ein Service ist eine geforderte Charakteristik an eine Dienstleistung. Services beschreiben SLAs und OLAs.
14	Task	Ein Task ist die Operationalisierung eines Backlog-Items. Der Task beschreibt die durchzuführende Aufgabe, Budget, Termine und Personen für die Abarbeitung. Fachliche Inhalte (Produktsicht) zur Aufgabenbearbeitung sind in den übergeordneten Backlog-Items enthalten. Task stellen eine dynamische vom Produkt gekapselte Projektsicht dar.

Tabelle 1: Definition von Backlog Items

Wer plant mit welchen Backlogs?

Zur Klärung der Frage scheint die Analyse von Rollendefinitionen und deren Abbildung auf konkrete Personen in einem Unternehmen naheliegend. Unsere Erfahrungen zeigen aber, dass die Definitionen von Rollen je Organisation zum Teil sehr verschieden sind. Auch innerhalb einer Organisation werden diese Rollen mitunter sehr unterschiedlich verstanden und gelebt. Prozesse, Organisationsform, Branche und Projektgröße sind Dimensionen, die begründen, weshalb wir in der Praxis so vielfältige Instanziierungen von Rollen antreffen. Wir suchen aber Antworten, die eher allgemeingültig sind, damit sie auf unterschiedliche Entwicklungsprojekte übertragen werden können. Deshalb helfen uns Rollenmodelle an dieser Stelle nicht viel weiter.

Sichtenwechsel notwendig!

Zur Einordung der agilen Backlogs, benötigen wir eine andere Sicht. Diese Sicht soll Aspekte betrachten, die in möglichst vielen Produktentwicklungen relevant sind. Damit können wir Aussagen über die Verwaltung der agilen Backlogs machen, die sich unabhängig von den Projektrahmenbedingungen leicht adaptieren lassen. Deshalb untersuchen wir die Inhalte der Backlogs – die Backlog Items.

Backlog Items

Ein Backlog Item kann allgemein als Item definiert werden, das in der Prduktentwicklung geplant Ressourcen verbraucht. Typischerweise sind dies neben Requirement, Feature oder User Story auch Issue, Defect und Test Set. Damit ein gleiches Verständnis über die hier verwendeten Backlog Items möglich ist, sind diese Backlog Items in Tabelle 1 kurz definiert. Sollten Sie in Ihrem Umfeld andere Definitionen dieser Begriffe vorfinden, können sie diese entsprechend übertragen.
Backlog Items repräsentieren die Inhalte der Produktentwicklung, die planungsrelevant sind. Dabei können die Inhalte 1:1 von einem Backlog Item abgebildet werden, wie zum Beispiel ein Requirement oder ein Test Case. Aber auch durch Attribute in mehreren unterschiedlichen Backlog Items können Informationen dargestellt werden, wie beispielsweise geschätzte Aufwände, Nutzen und Risiken. Weitere Zusammenhänge werden durch Beziehungen zwischen Backlog Items dokumentiert. Zum Beispiel kann das Backlog Item Task vom Subtyp „Implementation“ als Child-Item zu einer User Story angelegt werden. Die Parent-Child Beziehung wird zur Planung und für Reports genutzt. Aussagen über den Stand der Implementierung einer User Story sowie die Darstellung in einem Burn-Down-Chart sind damit möglich.
Die Liste der oben aufgeführten Backlog Items könnte noch erweitert werden. Und für einige Zuordnungen müsste im Einzelfall konkretisiert werden, welche Attribute der Backlog Items für Themen wie beispielsweise Entwicklungskosten oder Projektrisiken genutzt werden.

Lebenszyklen von Backlog Items

Im folgenden führen wir noch den Begriff des Lebenszykus eines Backlog Items ein. Darunter verstehen wir die Zustände, die ein Backlog Item von seiner Entstehung bis zum Ende haben kann. Damit diese Zustandsmodelle nicht beliebig definiert werden, schlagen wir eine an der Wertschöpfung orientierte Modellierung vor, wie sie von [DON10] beschrieben wird. Das bedeutet, dass der Zustand eines Backlog Items nur dann wechselt, wenn die nächste Reifestufe im Sinne einer erfolgten Wertschöpfung erreicht wurde. Als ein Beispiel wird der Requirement-Lebenszyklus in Kasten 2 dargestellt. Die essentiellen Zustände sind als solche gekennzeichnet und haben folgende Bedeutung:

Defined
Eine Anforderung hat den Zustand "Defined", wenn die Spezifikation der Anforderung vollständig ist und ein gemeinsames Verständnis der Stakeholder vorliegt. Dazu muss sichergestellt werden, dass die relevanten Qualitätskriterien eingehalten wurden bzw. mit Abweichungen bewusst umgegangen wird. Die Anforderung ist für die Umsetzungsplanung bereit.
Die Wertschöfpung wird durch die Qualität der Anforderungsdefinition erreicht.

Committed

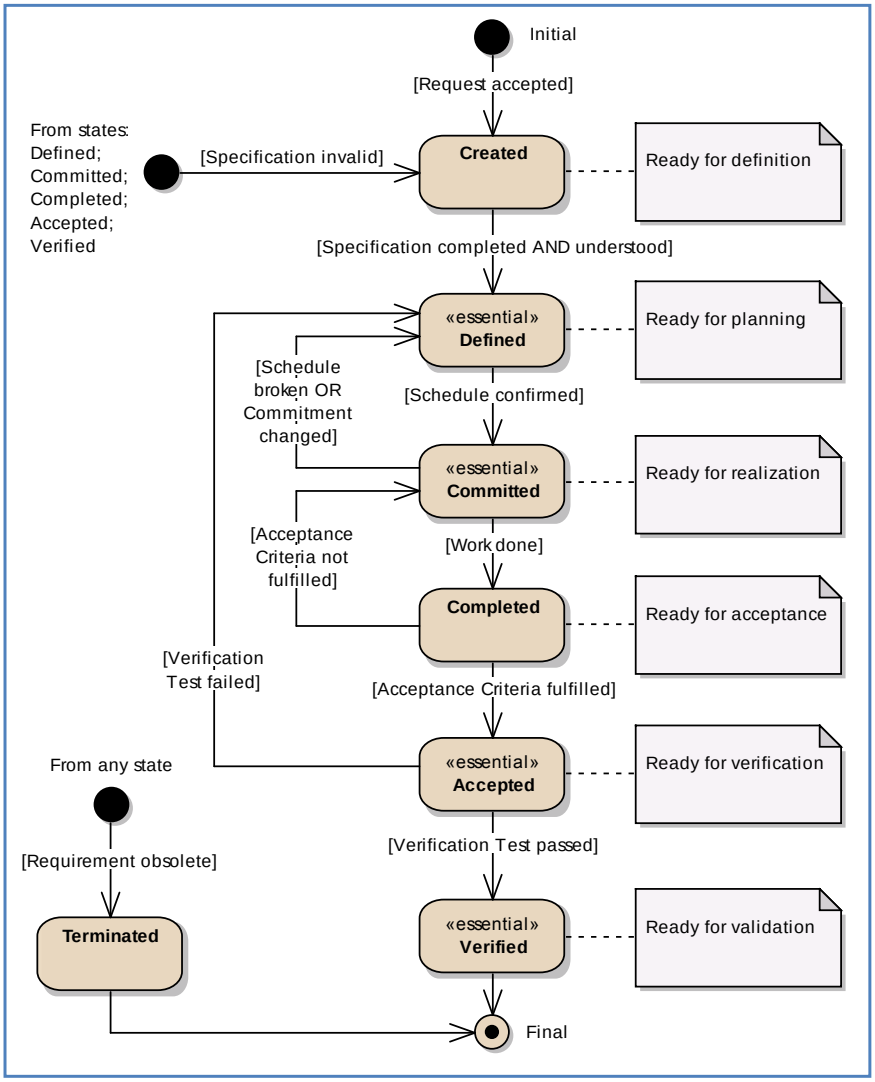
Die Anforderung wurde zur Realisierung durch ein Projektteam committed. Das Projektteam verpflichtet sich die Anforderung umzusetzen.
Die Wertschöfpung wird durch die abgeschlossenen Planungsaktivitäten für die Anforderung erreicht.

Accepted

Wenn die Acceptance Tests auf Basis der Anforderung erfolgreich waren, hat die Anforderung den Zustand "Accepted". Werden die Acceptance Tests nicht bestanden, kann eine weitere Umsetzung (Zustand "Committed") erfolgen. Die notwendigen Umsetzungen müssen noch im selben Entwicklungszyklus (z.B. Sprint oder Iteration) für die Anforderung realisierbar sein. Sonst muss die Umsetzung der Anforderung neu geplant und das Commitment neu eingeholt werden. Als Umsetzung wird auch ein konzeptionelles Lösungskonzept verstanden, zum Beispiel mit Prototyping. Die Acceptance Tests müssen gegen ein solches Lösungskonzept erfolgen.
Die Wertschöfpung wird durch eine vom Stakeholder (z.B. Product Owner) akzeptierte Umsetzung der Anforderung erreicht.

Verified

Wenn die Verification Tests auf Basis der Anforderung erfolgreich waren, ist der Zustand "Verified" erreicht. Eine Validierung des Entwicklungsergebnisses ist jetzt möglich. Werden die Verification Tests nicht bestanden, wird die Anforderung zur erneuten Realisierung in einem nachgelagerten Entwicklungszyklus für die Umsetzungsplanung bereitgestellt, die Anforderung ist dann im Zustand "Defined".
Die Wertschöpfung wird durch eine vom Stakeholder (z.B. Product Owner) akzeptierte Implementierung der Anforderung im auslieferbaren Produkt erreicht.



Kasten 2: Requirement-Lebenszyklus

Die essentiellen Zustände der Backlog Items sind stabil. Diese Eigenschaft wird durch die Orientierung an der Wertschöpfung erreicht. Backlog Items und deren Lebenszyklen sind gleichermaßen auf unterschiedliche Vorgehensweisen und Rahmenbedingungen für die Produktentwicklung übertragbar. Die Skalierung erfolgt meist mit der Auswahl bestimmter Items. Mit Hilfe der Lifecycle –Zustände ist es dann sehr gut möglich genau abzugrenzen, wer welchen Beitrag unter anderem in der Planungsphase leisten muss. Beispielsweise wird definiert, dass der Product Owner seine Features und User Storys definiert und priorisiert.
Der Product Owner setzt in dieser Konstellation den Zustand „Defined" für seine Backlog Items. Damit dokumentiert er den spätesten Beginn der Planung. Ein Scrum-Team schätzt die Efforts und muss ein Committment abgeben, wieviele User Storys im nächsten Sprint realisiert werden. Damit dieses und andere Szenarien für agile Planung transparent und nachvollziehbar ablaufen können, brauchen die Beteiligten gemeinsame Sichten auf Backlog Items.

Sichten auf Backlog Items - agile Backlogs

Über die Zuordnung der Backlog Items zu den am Anfang eingeführten Backlogs wollen wir weitere Aspekte im agilen Backlogmanagement aufzeigen.

Es werden folgende Verwendungsarten für ein Backlog Item in den Backlogs unterschieden:

• **temporary:**	das Backlog Item gehört für einen Abschnitt seines Lebeszyklus in ein Backlog und wird für weitere Abschnitte seines Lebeszyklus in andere Backlogs verschoben
• **partial:**	nur bestimmte Typen eines Backlogs Items gehören in ein Backlog
• **derived:**	das Backlog Item wird in andere Backlog Items anderer Backlogs abgeleitet
• **parallel:**	das Backlog Item gehört in einer Ausprägung zu mehreren Backlogs
• **exclusive:**	das Backlog Item gehört in einer Ausprägung ausschließlich nur in dieses Backlog

Eine vorgeschlagene Verwendung der Backlog Items in den Backlogs ist in Tabelle 2 aufgeführt.

Kritisch beim Zusammenwirken der Backlogs ist vor allem die parallele Verwendung derselben Backlog Items in mehreren Backlogs (siehe Ziffer 4 in Tabelle 2). Für dieses Szenario können die Lebenszyklus-Zustände keine klare Abgrenzung der Planungsaktivitäten sicherstellen. Möglich ist entweder eine Differenzierung der Planung auf Attribute-Ebene oder man entscheidet sich, das Backlog Item nur in einem Backlog zu editieren. Letzterer Vorschlag wird von uns favorisiert, da mögliche Inkonsistenzen oder aufwendige Abstimmungen durch eine doppelte Datenhaltung vermieden werden. Weiterhin kann ein Backlog Item, das in dem betrachteten Backlog keine Planungsrelevanz hat, dennoch zusätzlichen Kontext liefern. Beispielsweise kann bei der Bewertung der User Stories durch das Scrum-Team die Kenntnis über wichtige Issues – die das Team lösen muss – helfen, eine realistische Einschätzung über die tatsächliche Kapazität des Teams zu machen (siehe Zeile 9 (Issue) in Kombination mit Zeile 4(Sprint Backlog) in Tabelle 2).

Die aufgeführten Backlogs und Backlog Items haben nicht den Anspruch auf Vollständigkeit. Und doch sind aus Tabelle 2 zwei wichtige Zusammenhänge gut erkennbar:

- Die Backlogs sind nicht disjunkt, sie bilden Vereinigungsmengen, Schnittmengen und Teilmengen über Backlog Items
- Daraus folgt: Backlogs lassen sich sinnvoll als Sichten auf eine Menge von Backlog Items verstehen und handhaben. Sie liefern je nach Kontext einen passenden Überblick auf die relevanten Backlog Items.

Verwendung:
[1] = temporary
[2] = partial
[3] = derived
[4] = parallel
[5] = exclusive

	Backlog Items	Arten von Backlogs: Company Backlog (1)	Product Backlog (2)	Release Backlog (3)	Sprint Backlog (4)	Integration Backlog (Systemtest) (5)	Deployment Backlog (6)	Impediment Backlog (7)	Service Backlog (Input für Kanban) (8)	Personel Backlog (Developer) (9)
1	Goal	3	3							
2	Stakeholder Request	4	3	3					3	
3	Change Request		4	3					3	
4	Change Order			3					3	
5	Requirement	1	3	3	2	2	2		3	2
6	Feature		1	3	2	2	2		2	2
7	User Story				5					1
8	Defect				1	1	1			1
9	Issue				4	2	2	2	2	2
10	Test Set		1	1		2			1	1
11	Test Case			1	1	2			1	1
12	Ticket								5	
13	Service								5	
14	Task	2	2	2	4	4	4	4	2	4

Tabelle 2: Verwendung der Backlog Items in Backlogs

Daraus ziehen wir folgende Schlüsse:

1. Sind die relevanten Backlog Items identifiziert, lässt sich definieren, welche Backlogs (Sichten) tatsächlich benötigt werden.
2. Nachdem dann Backlog Items zu möglichen Backlogs zugeordnet wurden, kann je nach individuellem Rollenverständnis des Unternehmens adaptiert werden, wer welche Backlog Items / Backlogs bearbeitet und dafür verantwortlich ist.
3. Der Bürger „erster Klasse“ ist nicht das Backlog, sondern das Backlog Item
4. Das Zusammenwirken verschiedener Backlogs hängt vom Lebenszyklus und der Beziehungen zwischen Backlog Items ab.

Formen des Backlogs

Ein häufiger Diskussionspunkt ist die Frage, ob ein Backlog elektronisch geführt werden muss oder ob ein so genanntes Backlog Board eingesetzt wird. Es gibt für beide Ansätze ein Für und Wider. Für ein Backlog Board spricht die haptische Komponente und der unmittelbare Zugang zu den Informationen. Beim Konzept des Boards werden alle Backlog Items in Form von Karten auf einer Pinnwand angeordnet. Die Pinnwand ist für alle einsehbar und die Planung wird durch das Anordnen der Karten gemeinsam am Board durchgeführt. Diese Form der Backlog-Organisation funktioniert für Teams, die gemeinsam an einem Ort sitzen. Alle anderen Personen, die sich über die Inhalte eines Backlogs informieren wollen, müssen jedoch ebenfalls physisch anwesend sein.

Vorteil eines ein elektronischen Backlogs ist, dass die Beteiligten unabhängig von ihrem physischen Aufenthaltsort darauf zugreifen können. Weiterhin bietet es die Möglichkeit, unterschiedliche Sichten auf die Inhalte zu generieren und die Inhalte nach verschiedenen Aspekten zu gruppieren. Vor allem bei vielen Einträgen ist dies mit einem elektronischen Backlog weitaus einfacher machbar als durch das Umordnen von Karten an einem Board. Ein weiterer Vorteil ist die Versionierung des Backlogs und die Möglichkeit, die Historie des Backlogs und dessen Einträgen nachvollziehen zu können. Auch hier bietet ein elektronisches Backlog Vorteile gegenüber dem Abfotografieren eines Backlog Boards. In der Praxis sehen wir, dass sich eine Kombination der beiden Formen anbietet. Für Arbeiten in der Gruppe ist dem physischen Backlog Board Vorzug zu geben, für Dokumentation und Analysearbeiten ist das elektronische Backlog besser geeignet. Ebenso ist die Überlegung einzubeziehen, ob es sich um ein strategisches oder ein operatives Backlog handelt. So kann für das Sprint Backlog ein Board eher ausreichend sein, wohingegen für die unternehmensweite Planung ein Company Backlog besser in elektronischer Form vorliegen sollte.

Zusammenfassung

Agile Entwicklung ist in aller Munde und auch mittlere und große Organisationen wollen ihre Entwicklungsprojekte mit agilen Vorgehensweisen ausstatten.

Dies macht ein Umdenken erforderlich. Die Rahmenbedingungen, die zum Beispiel durch ein Framework wie Scrum definiert werden, trifft man im Alltag großer Entwicklungsorganisationen sehr selten an:

- Entwickler arbeiten zeitgleich in mehreren Projekten.
- Entwicklungsteam sind auf unterschiedliche Standorte und Zeitzonen verteilt.
- Ein auslieferbares Release muss durch Zulieferung vieler Teams erstellt werden.

Die unterschiedlichen Ausprägungen der Dimensionen Prozesse, Organisationsform, Branche und Projektgröße zeigen, wie vielfältig die Produktentwicklung gestaltet ist. Darauf basierende Rollenmodelle und deren Instanziierung sind daher keine guten Wege, mit dieser Vielfalt umzugehen.

Betrachten wir abschliessend nochmals den Lebenszyklus einer Anforderung:

Die Anforderung wird erhoben, wird dann angemessen definiert, dann für die Umsetzung geplant, umgesetzt und der Nachweis erbracht, dass die geforderte Charakteristik im Produkt enthalten ist. Das Backlog Item heißt „Requirement“. Und der Lifecycle liefert konstante Ergebniszustände. Jetzt muss einfach entschieden werden, in welchen Backlogs diese Zustände dokumentiert werden. Beispielsweise könnte die Erhebung im Company Backlog erfolgen, wird dann zur weiteren Detaillierung in ein Product Backlog verschoben. Die Umsetzungsplanung kann in unterschiedlichen Ebenen erfolgen: Produktplanung im Product Backlog, Releaseplanung im Release Backlog und für den Sprint ist diese Anforderung Input für die Selbstorganisation des Projetteams. Die Planung für den Nachweis der Realisierung kann dann entweder im Sprint selber durch Acceptance Tests oder im Integration Backlog erfolgen. Letzteres wird vor allem dann relevant, wenn ein auslieferbarer Stand nur durch Zulieferung vieler Teams erstellt werden kann.

Unsere Empfehlung ist, planungsrelevante Inhalte, also letztlich die Backlog Items, für die Entscheidung heranzuziehen, wie mit unterschiedlichen Backlogs gearbeitet werden soll. Die Backlog Items und ihre Lebenszyklen sind im Gegensatz zu Rollen leicht adaptierbar. Sie können sich darauf fokussieren, wie mit einem Backlog Item in der Entwicklung umgegangen wird und adaptieren die Ergebnisse auf beliebige Projektformen. So behalten Sie den Überblick und können klar trennen, wann welche Backlog Items (in welchem Zustand) geplant werden müssen. Welche Backlog -„Sichten“ Sie für Ihre Planung nutzen wollen, muss initial definiert werden – einschließlich der Zuordnung der relevanten Items. Daraus ergibt sich dann ein „natürliches“ Zusammenspiel der Backlogs, weil stabile und eindeutige Lebenszyklen der Backlog Items leicht verständlich sind und die agile Planung unterstützen.

Jens Donig

ist Senior Consultant für Systems Engineering bei der HOOD GmbH.

Die Schwerpunkte seiner Beratungstätigkeit liegen in den Bereichen Requirements Engineering (RE) und agile Softwareentwicklungsprozesse. Seit mehreren Jahren beschäftigt er sich intensiv mit der Analyse und dem Entwurf von ALM-Workflows. Mit modellbasierten Methoden schlägt er die Brücke vom abstrakten Makroprozess zum alltäglichen Arbeitsablauf.

Er ist Vater der Value-oriented Practices und Mitbegründer des ALM Solution Framework für TFS.

Susanne Mühlbauer

ist als Agile Coach für die HOOD GmbH tätig.

Sie arbeitet mit Leidenschaft und viel persönlichem Engagement mit Menschen, Teams und Organisationen auf ihrem Weg zu mehr Agilität. Aus ihrer Zeit als Consultant, Scrum Master, Projektleiter, Business Analyst und Requirements Engineer bringt sie langjährige Erfahrungen und die unterschiedlichsten Erlebnisse aus dem Projektgeschäft und in der Entwicklung komplexer Produkte und Systeme mit. Ein Schwerpunkt stellt hierbei sicherlich das Requirements Engineering dar.Sie hat zu diesen Themen Artikel veröffentlicht und ist gerne Referentin auf Fachkongressen.

Ihr Leitsatz: Agilität muss man erleben. Hierfür steht Agile-by-HOOD.

Über uns

Mit der Marke **Agile-by-HOOD** bündelt HOOD sein Angebot im agilen Umfeld zur zielgerichteten Unterstützung von Organisationen, die mit Scrum oder Kanban arbeiten wollen. HOOD kann dabei auf langjährige Erfahrungen im agilen Coaching und fundierte Expertise im Requirements Engineering zurückgreifen, um große Unternehmen beim Umstieg von klassischer auf agile Vorgehensweise zu begleiten.

Literaturhinweise & Links:

[Don10] *J. Donig, Dr. M. Künzle, Navigationsbesteck für Workflows – Teil 1: Arbeitsabläufe in der Softwareentwicklung optimieren, in: OBJEKTspektrum 06/2010*

[Sut11] *J. Sutherland, K. Schwaber, The Scrum Guide, www.scrum.org, 2011*

[Sut07] *J. Sutherland and K. Schwaber, The Scrum Papers: Nuts, Bolts, and Origins of an Agile Method. Boston: Scrum, Inc., 2007*

[Sch10] *K. Schwaber, Training Material Professional Scrum Master, ken.schwaber@scrum.org, 2010*

[Coh09] *M. Cohn, http://blog.mountaingoatsoftware.com/why-there-should-not-be-a-release-backlog/2009*

[Lef10] *D. Leffingwell, Agile Software Requirements, Addision Wesley, 2010*

[Agi01] *Agile Manifesto, http://agilemanifesto.org/iso/de/, 2001*

[Ren11] *Scrum of Scrums im Produktmanagement, http://www.renemt.de/2011/01/11/scrum-of-scrums-im-produktmanagement/, 2011*

[Wik11] *http://de.wikipedia.org/wiki/Scrum#Impediment_Backlog, 2011*