mehr zum thema:
https://www.xing.com/net/are
http://pbl.posterous.com

DAS PRODUKT-BACKLOG: RÜCKGRAT AGILER ANFORDERUNGSSTEUERUNG

Das Produkt-Backlog (PBL) ist Kernartefakt und Dreh- und Angelpunkt eines post-tayloristischen und dynamikrobusten Requirements-Engineerings. Aufbauend auf der nahezu trivialen Grundform des PBL strukturieren und organisieren Product Owner (POs) ihr PBL individuell entsprechend ihren Bedürfnissen und projektspezifischen Herausforderungen. Dieser Beitrag gibt Anregungen zu grundlegenden Strukturierungs- und Organisationsmöglichkeiten des PBL.

der autor

Bernd Oestereich
(E-Mail: E-Mail: bernd.oestereich@oose.de)
ist Gründer und Gesellschafter der oose Innovative Informatik GmbH und Autor zahlreicher Fachpublikationen. In Kürze erscheint sein Buch „Agiles Requirements-Engineering".

Grundsätzlicher Aufbau des PBL

Sehr vereinfacht ausgedrückt, ist das *Produkt-Backlog (PBL)* eine vom *Product Owner (PO)* gepflegte priorisierte Anforderungsliste, wobei der Begriff Anforderung sehr weit gefasst ist. Gewöhnlich handelt es sich dabei um eine geordnete Liste mit Einträgen, die beschreiben, welche Eigenschaften das zu entwickelnde System aufweisen soll.

Die Einträge sind meistens in einer eindeutigen Reihenfolge geordnet. Sie sind von *1* bis *n* durchnummeriert, wobei *1* die höchste Priorität hat. Henrik Kniberg (vgl. [Kni07]) ordnet die Elemente mit absoluten numerischen und nicht zwingend eindeutigen Werten. Ich präferiere jedoch die relative Ordnung, weil es eine Entscheidung erzwingt und weil das Wesen von Priorisierung auf dem Vergleich von Elementen beruht. Priorisieren heißt Entscheiden. Neben anderen Kriterien (siehe Kasten 1) ist der geschäftliche Wert der im Eintrag beschriebenen Produkteigenschaft für die Sortierung maßgeblich. Die Sortierung führt dann zu einer entsprechenden Realisierungsreihenfolge.

Die Elemente mit der höchsten Priorität werden in der Iterationsplanung von den Teams in Aufgaben (Tasks) überführt und kommen sukzessive in die Umsetzung. Die Priorisierung der Elemente ist nur ein markantes Merkmal des PBL. Ein anderes Merkmal ist der heterogene Abstraktions- und Verständnisgrad der Elemente. Das PBL enthält also sowohl sehr grobe, abstrakte und kaum verstandene Elemente, als auch fein-granulare, konkrete und sehr gut verstandene:

- Die inhaltliche Verständnistiefe reicht von *völlig unverstanden* bis *ausreichend verstanden*.
- Die Granularität reicht von *groß und umfassend* bis sehr *feingliedrig zerlegt*.

Welche Faktoren beeinflussen die Bearbeitungsreihenfolge?

- Direkter geschäftliche Nutzen und verhinderte Risiken/Schäden (funktionale und qualitative Anforderungen, Geschäftsgestaltung, Geschäftsprozessgestaltung, Anzahl der betroffenen Kunden/Anwender).
- Zeitlicher Nutzen, Zeitfenster (z. B. saisonal variierender geschäftlicher Nutzen).
- Organisatorischer Nutzen (indirekt, z. B. Zuliefer- oder funktionale Abhängigkeiten auflösen).
- Architektur- und Infrastrukturnutzen (indirekt, z. B. Investitionen in Robustheit, Stabilität, Nachhaltigkeit, Fehlerbehebung, Training, Entwicklungsproduktivität).

Hier kommen einige bewährte Techniken aus dem präskriptiven Requirements-Engineering (RE) ins Spiel, weil diese die Faktoren positiv beeinflussen können. Beispielsweise verfügen wir im RE über Techniken, Abhängigkeiten zwischen Anforderungen proaktiv zu antizipieren.

Kasten 1: Faktoren für Steuerung der Bearbeitungsreihenfolge (Priorität).

- Es sind alle möglichen Beschreibungsformen denkbar: Benutzergeschichten, Epen, Themen, Anwendungsfälle, Features, UML-Diagramme, DSL, sonstige formale Sprachen, Fachkonzepte, Skizzen der Benutzungsschnittstelle etc. Auch wenn Benutzergeschichten der Quasi-Standard sind, liegt gerade in der Vielfalt und Inhomogenität Potenzial zu möglichen Beschreibungsdetails (siehe auch Kasten 2).

Soweit der minimale und allgemeine Rahmen. Im Folgenden zeige ich nun weitere Strukturierungsmöglichkeiten, die ich persönlich hilfreich finde.

Nebenbereiche des PBL

Nicht alle Elemente sind priorisierbar oder als abgeschlossene Einheit umsetzbar. Vor allem Qualitätsanforderungen (auch nichtfunktionale Anforderungen genannt) – beispielsweise zur Benutzbarkeit, Administrierbarkeit, Wartbarkeit, Robustheit oder Sicherheit – eignen sich hierfür meistens nicht, da sie gewöhnlich grundsätzlich und übergreifend gelten sollen. Auch manche funktionale Anforderung ist übergreifend und damit in der Priorisierung unhandlich. Solche Anforderungen können separiert werden, wobei wiederum zwei Unterkategorien typisch sind (siehe Hinweise 1 und 2 in Abbildung 1):

- *Anforderungen, die projektweit gültig sind:* Diese gelten somit automatisch auch für jedes einzelne normale PBL-Element – es sei denn, dass einzelne Elemente explizit davon befreit werden. Anforderungen dieser Art gehören in die *Definition of Done*?[1])
- *Anforderungen, die für mehrere, aber längst nicht alle PBL-Elemente relevant sind:* Für diese Fälle ist explizit zu beschreiben, welche PBL-Elemente welche übergreifenden Anforderungen berücksichtigen sollen. Ein gängiger Weg ist es, alle von der übergreifenden

[1]) *Am Ende einer jeden Iteration wird das tatsächlich erreichte Ergebnis ermittelt, d. h. der aktuelle Entwicklungsstand anhand der fertigen Software begutachtet. Da stellt sich die Frage, was heißt eigentlich „fertig"? Bevor mit der Entwicklung begonnen wird, ist diese Frage verbindlich und für alle Beteiligten eindeutig verständlich zu klären. Der Scrum-Terminus hierfür lautet Definiton of Done (DoD) und meint eigentlich „PBL-Item-Done". Entsprechend gibt es auch „Definition of (PBL-Item-)Ready for Implementation", „Definition of Feature-Done" und „Definition of Release-Done".*

Als PO überlegen Sie sich zu Beginn des Projekts, ob und welche Mindestanforderungen Sie an den Inhalt und die Attribute der Einträge stellen – zusätzlich zu den durch die Art des Elements (Benutzergeschichte, Anwendungsfall, Entscheidungstabelle etc.) üblichen Eigenschaften. Häufiger anzutreffen sind folgende Attribute:

- ID
- Name oder Titel
- Priorität
- Hinweise auf besondere Chancen und Risiken
- relativer Aufwand
- fachliches Thema (um z. B. alle Einträge zu einem Thema gemeinsam zu repriorisieren)
- Subsystem, Komponente oder andere Architekturkategorien, um gegebenenfalls spezialisierte Teams treffend zu bedienen
- Akzeptanzkriterien (um festzustellen, ob die Anforderung fertig umgesetzt wurde, d. h. *was* am Iterationsende zu demonstrieren ist)
- Begutachtungs-/Demonstrationhinweise (grobe Beschreibung, *wie* diese Story zum Iterationsende demonstriert werden soll)
- Urheber und Liste relevanter Ansprechpartner, eventuell sogar ein Kommunikationsplan (wer sollte mit wem in welcher Weise welche Aspekte besprechen?)
- eventuell Lösungsideen und -empfehlungen

Es sind noch viele weitere Attribute denkbar. Hier gilt jedoch: Weniger ist mehr. Jeder PO muss also überlegen, welche Attribute er wirklich braucht.

***Kasten 2:** Attribute von PBL-Elementen.*

Anforderung betroffenen Einzelanforderungen entsprechend zu kennzeichnen und eine einfache Matrix über die Zusammenhänge zu pflegen.

In einem weiteren Bereich können ausgeschlossene Anforderungen gesammelt werden. Dort landen Anforderungen, die so unwichtig sind, dass sie gar nicht mehr an der Priorisierung teilnehmen sollen. Sie werden bis auf Weiteres aus dem PBL ausgeschlossen und ignoriert. Alternativ können sie auch einfach vernichtet werden, aber manchmal ist es hilfreich, hiermit an die explizite Ausschlussentscheidung zu erinnern.

Welche Faktoren haben Einfluss darauf, wann eine Anforderung genug verstanden wurde?

- Grad der inhaltlich-fachlichen Ähnlichkeiten (Affinität) mit anderen Anforderungen
- Grad der inhaltlich-fachlichen Abhängigkeiten von anderen Anforderungen
- Grad des offenen Erkenntnispotenzials (aufgrund möglicher Widersprüche, Lücken, Unsicherheiten und Abhängigkeiten)

Die im klassischen *Requirements-Engineering (RE)* geforderten Eigenschaften, Anforderungen sollen vollständig, konsistent und designfrei sein, greifen im agilen Vorgehen wegen der zeitlich verteilten und bestandsarmen Bearbeitung kaum noch.

***Kasten 3:** Faktoren für Steuerung der Verständnistiefe.*

Qualitätsanforderungen, die spezifisch für eine einzelne Anforderung gelten, können meistens über die Akzeptanzkriterien zu der Anforderung beschrieben werden[2]). In welcher Form und mit welchen Werkzeugen das PBL gepflegt wird, ist nachrangig. Am einfachsten und für die gemeinsame Arbeit und Kommunikation am besten ist eine große Wand, wie z. B. eine Pinnwand mit Karteikarten oder eine mit Haftnotizen beklebbare Wand. Es kann auch eine elektronische Liste sein oder eine Kombination von allem.

Die Dynamik des PBL und seiner Einträge

Das PBL wird kontinuierlich gepflegt, d. h. die Einträge ändern sich immer wieder während der gesamten Projektlaufzeit: Neue kommen hinzu, unmittelbar zur Realisierung anstehende werden entnommen, Prioritäten werden geändert, Inhalte werden geändert, Einträge weiter unterteilt oder wieder zusammengeführt – und in alle diese Aktivitäten wird das Projektteam soweit eingebunden, dass es die Inhalte und ihre Entwicklungsgeschichte zumindest grundsätzlich überblicken und verstehen kann.

Eine große Anforderung wird besser durchdrungen und verstanden, wenn sie in einzelne konkretere Elemente zerlegt wird. Regelmäßig werden also einzelne Elemente aus dem PBL herausgenommen und durch eine Menge fein-granularer Elemente ersetzt, die dann auch jeweils individuelle Prioritäten besitzen (**siehe Hinweis 3 in Abbildung 2**). So können aus einem Epos schließlich Benutzergeschichten werden bzw. aus Produkt-Features und Geschäftsprozessen schließlich Systemanwendungsfälle, elementare Anforderungen, detaillierte fachliche Regeln und Testfälle. Zuerst steht dabei immer die Entscheidung über die Priorität, dann muss das Verständnis für die Anforderungen mit hoher Priorität ausreichend vertieft werden (**siehe Kasten 3**). Der Lebenszyklus des PBL insgesamt erstreckt sich mindestens über die gesamte Projektlaufzeit, kann bedarfsweise aber auch früher beginnen oder länger dauern.

Im Prinzip kann jeder Projektbeteiligte zu neuen Einträgen beitragen, aber der Weg führt immer über den PO. An der Pflege des PBL sind verschiedene Personen beteiligt, aber der PO hat allein die Verantwortung und Entscheidungskompetenz für die Inhalte, Strukturierung und die Sortierung. Da der PO über die Priorität entscheidet, kann nur er den Eintrag aufnehmen und positionieren. Außerdem ist das PBL keine Anforderungssenke, in die einfach neue Anforderungen hineingeworfen werden, sondern das PBL und seine Elemente sind Dreh- und Angelpunkt für die stetige Kommunikation über die Anforderungen im Projekt.

Ich möchte an dieser Stelle drei grundsätzliche Phasen definieren, die jedes PBL-Element nacheinander durchläuft:

- *Unverstanden:* Die meisten Anforderungen sind zunächst gar nicht richtig verstanden. Das Verständnis des POs reicht vielleicht gerade aus, um die Anforderung überhaupt zu priorisieren. Das Entwicklungsteam muss zu diesem Zeitpunkt die Anforderungen nur ▶

[2]) *Ein Akzeptanzkriterium beschreibt, unter welchen Bedingungen (zusätzlich zu denen der Definition von „fertig") eine vom Entwicklungsteam umgesetztes PBL-Element als fertig entwickelt gelten soll. PBL-Elemente werden vor der Realisierung durch Akzeptanzkriterien konkretisiert und am Ende der Iteration an ihnen binär (fertig/nicht fertig) beurteilt.*

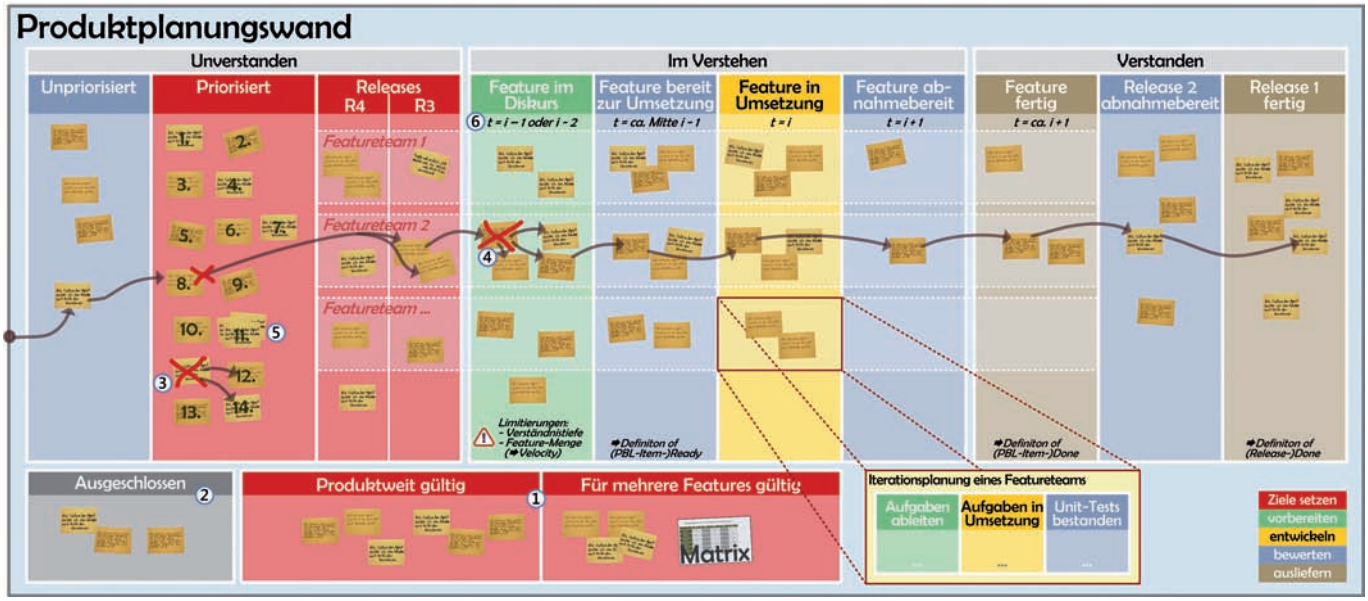

Abb. 1: Das PBL mit einem nach dem Verständniszustand, Release-Planung und Teampräferenzen der Elemente unterteiltem Hauptbereich.

soweit verstehen, um den PO bei der Priorisierung ausreichend zu unterstützen, beispielsweise um Aufwand und Risiken taxieren zu können. Die Anforderung ist recht roh. Abstrakte Anforderungen werden in dieser Phase vor allem deswegen in mehrere konkretere zerlegt, um sie zum Zwecke der Priorisierung und Planung gut genug zu verstehen – nicht jedoch um sie umsetzen zu können.

- *Im Verstehen:* Die Anforderung ist noch ungenügend verstanden. Irgendwann schnappen wir uns eine Anforderung und versuchen, sie gut genug zu verstehen, um sie umsetzen zu können. Der PO sorgt nun dafür, dass die relevanten Stakeholder – vom Benutzer bis zum Entwickler – über die ausgewählten Anforderungen sprechen, sodass alle gemeinsam Erkenntnisse gewinnen und die Anforderung näher kennenlernen (**in Abbildung 1 „Feature im Diskurs" genannt**). Jetzt geht es darum, nicht nur das Wissen der Anforderungsgeber und des POs zu vertiefen, sondern vor allem auch das Entwicklungsteam mit einzubeziehen. Anforderungen werden in mehrere elementarere Elemente zerlegt, sodass auch die Prioritäten spezifischer werden und das Entwicklungsteam Aufwand und Risiken nun spezifischer schätzt (**siehe Hinweis 4 in Abbildung 1**).

Klassisches RE	Agiles/dynamik-robustes RE
Anforderungen perfektionieren	Anforderungsprozess optimieren
Hohe/maximale/homogene Verständnistiefe	Je nach Anforderung individuell minimal notwendige Verständnistiefe
Anforderungsbestände aufbauen	Bestände minimieren (Lean-Prinzipien)
Homogenes Anwenden von RE-Techniken, einheitliche Werkzeuge	Je nach Anforderung individuelle Auswahl passender RE-Techniken
Formale Korrektheit und Nachvollziehbarkeit	Kollaboration und Kommunikation
Festlegungen und Abgrenzungen; Ruhe und Stabilität erzielen	Permanente Evolution/Weiterentwicklung; Flexibilität erzielen
Vollständigkeitsgesteuert; Befriedigung von Sicherheitsbedürfnissen	Prioritätsgesteuert; Weglassen von geschäftlich wertarmen Anforderungen
Wasserfall oder iteratives Vorgehen	Iteratives oder kontinuierliches Vorgehen
Auftragsorientiert	Produktorientiert
Anforderungen sind eigenständig existierende Artefakte/Objekte	Anforderungen sind Ausdruck/Objekte von Interessenshaltern
RE verwaltet und macht sich zum verantwortlichen Subjekt der Artefakte	PO führt, d. h. der PO vermittelt zwischen verantwortlichen Subjekten

Tabelle 1: Prinzipielle und zum besseren Verstehen zugespitzte Unterschiede zwischen klassischem und agilem RE.

- *Bereit zur Umsetzung:* Der Prozess des Verstehens wird gestoppt, sobald wir die Anforderung gut genug verstanden haben, um sie einem Team zur Umsetzung zu übergeben. Hierzu muss die Anforderung gut schätzbar und in klare prägnante Aufgaben zerlegbar sein und die Akzeptanzkriterien müssen geklärt und formuliert sein.

Dass grobe Anforderungen in mehrere detailliertere verfeinert werden, kommt daher praktisch nur in den Bereichen *Priorisiert und Release* und *im Diskurs* der **Abbildung 1** vor. Der Zweck der Verfeinerung in den Bereichen *Priorisiert, Release* ist es, priorisieren zu können. Der Zweck der Verfeinerung im Bereich *im Diskurs* ist es, realisieren zu können.

Für Anforderungen, die noch wenig verstanden sind, die aber möglicherweise schon weiter unterteilt wurden, kann es trotz Priorisierungsgebots angemessen sein, sie weiterhin als quasi einen Eintrag aufzufassen und ihnen somit zumindest temporär die gleiche Priorität zu geben wie zuvor dem nicht aufgeteilten Eintrag (**siehe Abbildung 1, Hinweis 5**). Sobald Einträge explizit in den Verstehensprozess wandern und in Kürze umgesetzt werden sollen, ist es wichtig, dass jeder Eintrag eigenständig ist und eine eigene Priorität hat.

Viele Erkenntnisse zu Anforderungen ergeben sich auch beiläufig beim Diskurs über andere Anforderungen. Aufgrund solcher Erkenntnisse eine Anforderung beispielsweise in mehrere Detailanforderungen zu unterteilen, ist angebracht. Wenn diese Anforderungen aber andererseits allesamt noch recht grob sind, kann die differenzierende Priorisierung meistens ohne besondere Risiken vertagt werden.

Anforderungen können prinzipiell von jedermann beigesteuert werden und der PO muss diese nicht immer gleich priorisieren. Manchmal handelt es sich gar nicht um echte Anforderungen, sondern mehr um Probleme, Ziele, mehr oder weniger unklare Wünsche oder irgendetwas anderes, was gar nicht oder kaum zu priorisieren ist. In **Kasten 1** werden wichtige Kriterien genannt, nach denen die Priorisierung erfolgen sollte. Grundsätzlich sind Nutzen und Chancen nur die eine Seite der Medaille – die andere betrifft die Kosten und Risiken. Vor einer Priorisierung sollten auch die mit der Anforderung verbundenen Kosten geschätzt werden.

Im Fall des PBL bedeutet Priorisieren, die Elemente in eine eindeutige relative Ordnung zu bringen. Hierzu müssen die Elemente miteinander verglichen und es muss entschieden werden, welches jeweils wichtiger ist. Dass die im PBL enthaltenen Elemente in ihrem Abstraktionsgrad und ihrer Granularität heterogen sind, macht die Vergleiche nicht immer einfach. Manchmal fühlt es sich so an, als würde man Äpfel mit Birnen vergleichen. In diesem Sinne sind Anforderungen anfangs manchmal noch gar nicht in einem Zustand, in dem sie sinnvoll an der Priorisierung teilnehmen können. Ein PBL enthält also typischerweise eine Menge von temporär nicht priorisierten Elementen.

Zu welchem typischen Zeitpunkt sollte eine Anforderung in welchem Bereich liegen (**siehe auch Hinweis 6 in Abbildung 1**)?

- In der erforderlichen Tiefe verstanden – und damit bereit zur Umsetzung – sollten Anforderungen möglichst kurzfristig sein, d. h. unmittelbar vor der Umsetzungsiteration.
- Der Verständnisprozess sollte, vielleicht abgesehen von sehr kleinen Projekten, wiederum spätestens eine Iteration vorher gestartet werden, denn Verstehen braucht Zeit. Je nach Anforderung kann das aber auch deutlich früher erfolgen.

Die eigentliche Priorisierung findet im Bereich *Unverstanden* statt. Die höchsten Anforderungen dieses Bereichs wandern dann möglichst zügig weiter nach rechts. Hier kann der PO bedarfsweise auch die Release-Planung visualisieren, wie es in **Abbildung 1** gezeigt wird.

In größeren Projekten mit mehreren Teams, in denen nicht mehr jedes Teammitglied für jedes PBL-Element in gleicher Intensität am Anforderungsdiskurs teilnehmen kann, sollten frühestens mit der Release-Planung, idealerweise aber mit dem Eintritt in den Bereich *im Diskurs*, die Verteilungspräferenzen (welches Team wird voraussichtlich welche Elemente umsetzen) festgelegt werden, wozu im PBL die Teamstruktur explizit aufgenommen werden kann. Die Verteilung auf die Teams sollte auch nicht zu spät erfolgen, da Schätzungen und Entwicklungsgeschwindigkeit (Velocity) teamspezifisch sind und Anforderungen möglichst von den Teams umgesetzt werden sollen, die sie geschätzt und im Anforderungsdiskurs inhaltlich gut durchdrungen haben.

Minimierung des Bestands

Während die Priorisierung letztendlich einfach nur eine jederzeit revidierbare Entscheidung ist, erzeugt das Durchdringen und Verstehen von Anforderungen gewöhnlich nachhaltige und unumkehrbare Erkenntnisse – zumindest solange unser Gedächtnis funktioniert und wir über Techniken verfügen, dieses Wissen zu speichern oder kurzfristig weiter zu verwenden.

Damit kommen wir zum nächsten entscheidenden Aspekt: Wir versuchen, das Maß der notwendigen Dokumentation und Wissensspeicherung so klein wie möglich zu halten. Eine generelle Strategie hierzu besteht darin, den Zeitraum vom Verstehen einer Anforderung bis hin zur Akzeptanz der Fertigstellung möglichst kurz zu halten. Das führt bei einer begrenzten Teamstärke – respektive Entwicklungskapazität – sinnvollerweise dazu, die Anzahl der gleichzeitig im Verstehensprozess befindlichen Anforderungen auf ein angemessenes Maß zu begrenzen (vgl. hierzu auch [And10] und das kürzlich erschienene Kanban-Heft des OBJEKTspektrum).

Wenn neue Elemente ins PBL kommen, sind sie gewöhnlich noch unverstanden, d. h. sie werden mit einer relativ großen Unsicherheit und eher grob granular priorisiert. Das ist genau so beabsichtigt. Es ist nicht das Ziel, alle Elemente möglichst schnell zu detaillieren – ein Punkt, an dem sich viele erfahrene Anforderungsanalytiker aufreiben, sind sie doch darauf konditioniert, Anforderungen stets umfassend, vollständig und detailliert verstehen zu wollen.

Stattdessen werden erst einmal die rohen, unbearbeiteten und wenig verstandenen Anforderungen priorisiert und von diesen nur die am höchsten priorisierten Elemente konkretisiert. Das bedeutet, dass ein kleiner Teil am oberen hochprioren Ende des in **Abbildung 2** *Unverstanden* genannten Bereichs einem Verständnis- und Detaillierungsprozess zugeführt wird. Das Verständnis wird dabei solange weiter vertieft (Bereich *Konversation*), bis es für die Umsetzung, also die Programmierung, ausreicht (dann landet es in Bereich *Bereit*). Das bedeutet: Die Elemente werden nicht solange konkretisiert, bis sie vollständig bis ins letzte Detail verstanden wurden, sondern nur

Der Übergang vom Wasserfall-Vorgehen zum iterativen Vorgehen bedeutet, für jede einzelne Anforderung zu entscheiden, wann der richtige Zeitpunkt für sie ist. Der Übergang vom iterativen zum agilen RE bedeutet zusätzlich, für jede einzelne Anforderung zu entscheiden, wie sehr sie überhaupt verstanden werden muss (Steuerung der Verständnistiefe) und welche RE-Techniken hierfür am besten sind. Agiles RE ist also die aktive Steuerung von Anforderungsunvollkommenheit und ein uneinheitliches RE.

Definition
Requirements-Engineering (RE) ist der Erwerb und die Anwendung von struktur- und sozialwissenschaftlichem Wissen[3]), um die Zusammenarbeit von Menschen im Kontext von Anforderungen für die konsistente Entwicklung von Systemen zu unterstützen.
Agiles RE ist RE mit der Strategie,

- genau die Menge an Anforderungen in der angemessenen Beschaffenheit zum passenden Zeitpunkt den relevanten Beteiligten verfügbar zu machen,
- in einer kontinuierlichen Begleitung immer die insgesamt effektivsten und effizientesten Kommunikationsmöglichkeiten zwischen den Beteiligten und den dafür notwendigen wissenschaftlichen Techniken einzusetzen,
- Anforderungsänderungen als Ausdruck eines reifenden Verständnisses und reifender Entscheidungen wertzuschätzen.

Interaktion ist wichtiger als Spezifikation
RE beschränkt sich nicht auf die Erhebung, Analyse, Spezifikation und Dokumentation von Anforderungen. Vielmehr müssen die Inhalte und Bedeutungen von Anforderungen zwischen den verschiedenen Beteiligten (vor allem zwischen fachlichen Vertretern und Entwicklern) proaktiv vermittelt, übersetzt, vertieft und geklärt werden.

Evolution ist wichtiger als Determination (Abgrenzung)
Die Aufgabe des RE ist nicht einmalig und findet keinen eigenen Abschluss, sondern ist eine die gesamte Entwicklungszeit andauernde begleitende Tätigkeit. Agiles RE versteht Spezifikationen, Definitionen und andere Abgrenzungen als sowohl volatile und jederzeit zu hinterfragende und weiter zu entwickelnde als auch als Struktur bildende und Kollaboration und Prozesse vereinfachende Artefakte an.

Lean
Diesen Prinzipien untergeordnet sind zusätzlich Wartezeiten im Entwicklungsprozess, Informationsbestände und Produktnachbesserungen zu minimieren (ein Bestand ist etwas, das aktuell nicht verwendet wird).

Querschnittsfunktion
RE ist vor allem eine indirekt wertschöpfende, querschnittliche und dienstleistende Disziplin. Sie unterstützt verschiedene andere Disziplinen, beispielsweise Entwicklung, Organisations-und Projektmanagement, Architektur, Betriebswirtschaft und die jeweiligen fachlichen Domänen.

Kasten 4: Was ist agiles RE?

soweit, wie es für die kurzfristige Umsetzung eben notwendig ist. Abbildung 1 verdeutlicht diese Zusammenhänge:

- Die Anzahl der Elemente, die gleichzeitig verstanden sein müssen, soll auf ein angemessenes Minimum reduziert werden. Hier ist der Begriff *Work Items in Progress (WIP)* üblich.
- Die mit diesem Prozess erreichte Verständnistiefe wird für jedes Element individuell auf ein angemessenes Minimum reduziert. Freunde von Anglizismen nennen dies *Minimal Depth Fit (MDF)*.

Die Anzahl der WIP für den Bereich *Bereit zur Umsetzung* ergibt sich aus der Umsetzungskapazität des Entwicklungsteams. Als Heuristik schlage ich vor: Wenn Sie keine Erfahrungswerte haben, setzen Sie für die Elemente in Bereich *Bereit zur Umsetzung* in Summe ca. 150–200 Prozent der Umsetzungskapazität an, um ausreichend Spielraum für Eventualitäten zu haben. Als Heuristik für den Bereich *im Diskurs* empfehle ich entsprechend, mit 200–300 Prozent der Umsetzungskapazität zu starten. Alle anderen Elemente sind im Bereich *Unverstanden*.

Die (auf synthetischer Zahlenbasis beruhende) Abbildung 2 zeigt die kontinuierliche Zunahme der fertigen Elemente, die sich hier durch eine relativ konstante Umsetzungskapazität ergibt. Die Bereiche *Genug verstanden* und *Im Verstehen* bilden den ebenfalls halbwegs kontinuierlichen Vorlauf hierfür. Alles andere bleibt unbearbeitet, solange es die Priorisierung nicht in den nächsten Bereich trägt. Was auch immer zum Projektende, also zur letzten Iteration, nicht umgesetzt wurde, bleibt dann zwangsläufig ausgeschlossen.

Der Anteil der letztendlich ausgeschlossenen bzw. nicht umgesetzten Anforderungen zeigt vor allem die Bereitschaft und Möglichkeit, jederzeit beliebige Anforderungen in den Prozess einsteuern zu können, und einen funktionierenden Filtermechanismus, nicht jeden Wunsch blindlings umzusetzen, sondern mit den verfügbaren Ressourcen verantwortungsvoll umzugehen.

Fazit

Der PO kann das PBL in sehr unterschiedlicher Weise ausprägen, von einer simplen Liste bis hin zu einer ausdifferenzierten Struktur. Es gibt keine Standardstruktur und kein Patentrezept – jeder PO muss in Zusammenarbeit mit dem übrigen Beteiligten die für das spezielle Projekt passende Struktur finden. Dieser Beitrag sollte einige Möglichkeiten hierzu skizzieren. Wann immer Sie Gelegenheit haben, sich in einem fremden Projekt das PBL anzusehen, nutzen Sie diese, sprechen Sie mit den

[3]) *Zu den Strukturwissenschaften zähle ich insbesondere die Mathematik, Informatik, Systemtheorie, Linguistik und Synergetik, zu den Sozialwissenschaften vor allem Wirtschafts-, Kommunikations-, Rechtswissenschaft, Soziologie und Psychologie.*

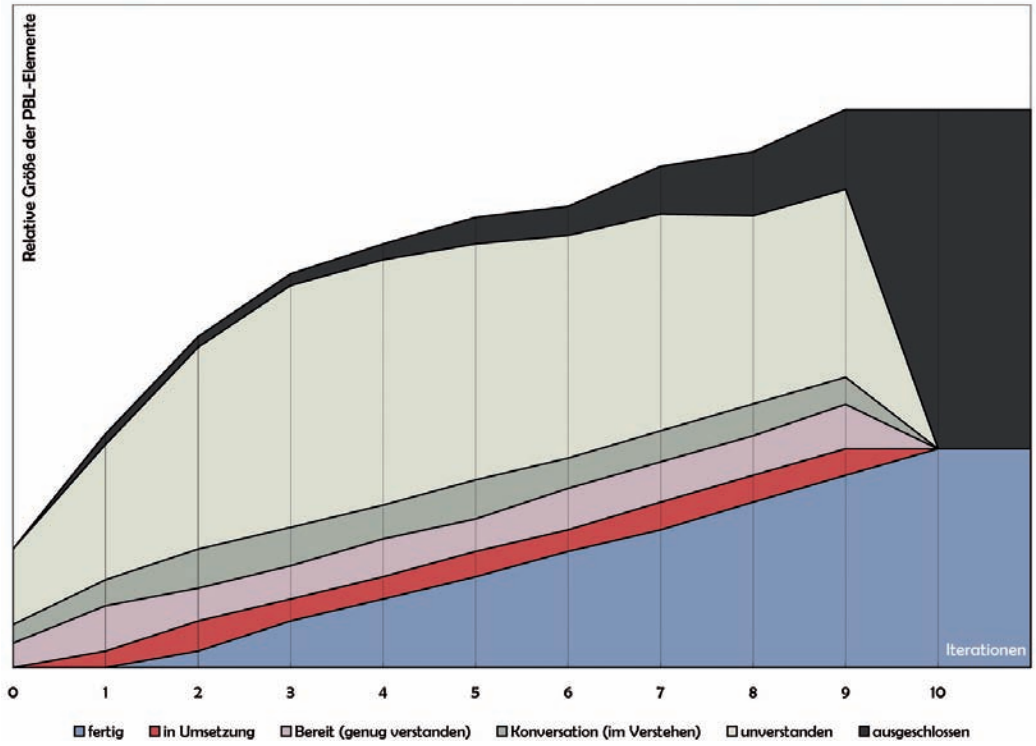

Abb. 2: Das kumultative Flussdigramm für PBL-Elemente zeigt die Veränderung der Anforderungszustände im Projektverlauf.

Beteiligten über die Besonderheiten und deren Erfahrungen, und erweitern Sie so Ihr persönliches Ideen-Repertoire. Falls es Ihnen erlaubt ist, senden Sie doch ein Foto von Ihrem PBL per E-Mail an post@pbl.posterous.com oder schauen Sie sich unter http://pbl.posterous.com die PBL anderer an. ■

Literatur

[And10] D. Anderson, Kanban – Successful Evolutionary Change for Your Technology Business, Blue Hole Press 2010

[Coh10] M. Cohn, User Stories, mitp-Verlag 2010

[Pic10] R. Pichler, Agile Product Management with Scrum, Addison-Wesley 2010

[Kni07] H. Kniberg, Scrum and XP from the Trenches, C4media 2007

[Oes] B. Oestereich, Agiles Requirements-Engineering, dpunkt.verlag (in Vorbereitung)