Certified Professional for Requirements Engineering

RE@Agile

Lehrplan

Stefan Gärtner, Peter Hruschka,
Markus Meuten, Gareth Rogers,
Hans-Jörg Steffe

2.3.0 | 1. Oktober 2025

Nutzungsbedingungen:

1. Einzelpersonen und Seminaranbieter dürfen den Lehrplan als Grundlage für Seminare verwenden, sofern die Inhaber der Urheberrechte als Quelle und Besitzer des Urheberrechts anerkannt und benannt werden. Des Weiteren darf der Lehrplan zu Werbezwecken nur mit Einwilligung des IREB e.V. verwendet werden.
2. Jede Einzelperson oder Gruppe von Einzelpersonen darf den Lehrplan als Grundlage für Artikel, Bücher oder andere abgeleitete Veröffentlichungen verwenden, sofern die Autoren und IREB e.V. als Quelle und Besitzer des Urheberrechts genannt werden.

Danksagung

Die aktuelle Version dieses Lehrplans wurde verfasst von: Stefan Gärtner, Peter Hruschka, Kim Lauenroth, Markus Meuten, Gareth Rogers und Hans-Jörg Steffe.

Review durch Rainer Grau. Mit Kommentaren von Jan Jaap Cannegieter, Andrea Herrmann, Uwe Valentini und Sven van der Zee. Deutsch-Review durch Raik Arndt, Sibylle Becker und Ruth Rossi.

Die Freigabe der englischen Version wurde am 25. Oktober 2024 durch das IREB Council auf Empfehlung von Xavier Franch genehmigt.

Allen sei für ihr Engagement gedankt.

Zweck des Dokuments

Dieser Lehrplan definiert die Zertifikate für den RE@AGILE Practitioner und für den RE@Agile Specialist des International Requirements Engineering Board (IREB). Der Lehrplan dient den Schulungsanbietern als Grundlage für die Erstellung ihrer Kursunterlagen. Studierende können sich anhand des Lehrplans einen Überblick über die vorgesehenen Inhalte und Lernziele verschaffen. Inhaltliche Details zur Vorbereitung des Kursmaterials und auf die Prüfung finden Sie im „Handbuch RE@Agile, Aus- und Weiterbildung zum IREB Certified Professional for Requirements Engineering RE@Agile Practitioner | Specialist".

Inhalt des Lehrplans

Das Modul RE@Agile richtet sich an Personen aus dem Requirements Engineering, der Business Analyse, dem Business Engineering, der Software- und Systementwicklung sowie an bestehende Rollen in der agilen Community (Product Owner, Scrum Master, Entwickler), die ihre Kenntnisse im Umgang mit Anforderungen vertiefen möchten.

Detaillierungsgrad

Der Detaillierungsgrad dieses Lehrplans erlaubt international konsistentes Lehren und Prüfen. Um dieses Ziel zu erreichen, beinhaltet dieser Lehrplan Folgendes:

- Allgemeine Lernziele
- Inhalte mit einer Beschreibung der Lernziele und
- Referenzen zu weiterführender Literatur (falls notwendig).

Lernziele/Kognitive Stufen des Wissens

Allen Modulen und Lernzielen in diesem Lehrplan ist eine kognitive Stufe zugeordnet. Die folgenden Stufen werden verwendet:

- **K1: Kennen** (identifizieren, erinnern, abrufen, erkennen, wissen) – Der Lernende erkennt und erinnert sich an einen Begriff oder ein Konzept.
- **K2: Verstehen** (zusammenfassen, generalisieren, abstrahieren, klassifizieren, vergleichen, übertragen, gegenüberstellen, erläutern, interpretieren, übersetzen, darstellen, schlussfolgern, schließen, kategorisieren, Modelle erstellen) – Der Lernende ist in der Lage, Gründe oder Erklärungen für Aussagen zum Thema auszuwählen, und kann zusammenfassen, vergleichen, klassifizieren, kategorisieren und Beispiele für den Lerngegenstand geben.
- **K3: Anwenden** (umsetzen, ausführen, anwenden, einem Vorgehen folgen, ein Vorgehen anwenden) – Der Lernende kann die korrekte Anwendung einer Vorgehensweise oder Technik auswählen und in einem gegebenen Kontext durchführen.
- **K4: Analysieren** (analysieren, organisieren, Zusammenhänge finden, integrieren, umreißen, überprüfen, strukturieren, schlussfolgern, zerlegen, unterscheiden, auseinanderhalten, gegenüberstellen, fokussieren, auswählen) – Der Lernende kann Informationen, die sich auf ein Vorgehen oder eine Technik beziehen, zum besseren Verständnis in seine Bestandteile zerlegen und kann zwischen Fakten und Schlussfolgerungen unterscheiden. Typische Anwendungsfälle sind, ein Dokument eine Software oder eine Projektsituation zu analysieren und angemessene Maßnahmen vorzuschlagen, die ein Problem oder eine Aufgabe lösen
- **K5: Beurteilen** (kritisieren, beurteilen) – Der Lernende kann eine gut begründete Kritik an einem gegebenen Artefakt äußern und ein fundiertes Urteil in einem gegebenen Fall abgeben.

Beachten Sie, dass ein Lernziel auf der kognitiven Wissensstufe Kn auch Elemente aller darunterliegenden kognitiven Wissensstufen (K1 bis Kn-1) enthält.

> Beispiel: Ein Lernziel der Art „Die RE-Technik xyz anwenden" ist auf der kognitiven Wissensstufe (K3). Die Fähigkeit zur Anwendung setzt aber voraus, dass die Lernenden die RE-Technik xyz kennen (K1) und dass sie verstehen, wozu diese Technik dient (K2).

Alle in diesem Lehrplan verwendeten Begriffe und Begriffe, die im Glossar genannt werden, sind zu kennen (K1), auch wenn sie in den Lernzielen nicht explizit genannt sind.

Das Glossar steht auf der IREB Webseite zum Download zur Verfügung: https://www.ireb.org/de/downloads/#cpre-glossary-2-0

Im Lehrplan sowie im dazugehörigen Handbuch wird die Abkürzung „RE" für Requirements Engineering verwendet.

Lehrplanaufbau

Der Lehrplan besteht aus 6 Hauptkapiteln. Ein Kapitel umfasst eine Lerneinheit (LE). Der Haupttitel eines Kapitels beinhaltet die kognitive Stufe des Kapitels, das ist die höchste Stufe der Teilkapitel. Die genannte Unterrichtszeit ist das Minimum, das in einem Kurs für dieses Kapitel aufgewendet werden sollte. Schulungsunternehmen steht es frei, mehr Zeit für die LEs und Übungen zu investieren. Sie sollten jedoch sicherstellen, dass der Zeitaufwand im Verhältnis zu den übrigen LEs beibehalten wird. Die für ein Kapitel wichtigen Begriffe werden zu Beginn jedes Kapitels aufgelistet.

> Beispiel: LE2 Projekte erfolgreich starten (K2)
> Dauer: 120 Minuten + 60 Minuten Übungszeit
> Begriffe: Produktvision, Produktziel, Stakeholder, Persona, Produktumfang, Systemgrenze

Das Beispiel zeigt, dass in Kapitel 2 die Lernziele der Stufe K2 enthalten sind und 180 Minuten für das Lehren des Materials in diesem Kapitel vorgesehen sind.

Jedes Kapitel kann Unterkapitel enthalten. In deren Titel findet sich ebenfalls die kognitive Stufe der betroffenen Teilinhalte.

Die Lernziele (LZ) sind gelistet. Die Nummerierung zeigt die Zugehörigkeit zu Unterkapiteln an.

> Beispiel: LZ 3.3.1 Bei der Erstellung von Anforderungen die INVEST Kriterien anwenden können (K3)

Dieses Beispiel verdeutlicht, dass das Lernziel LZ 3.3.1 im Unterkapitel 3.3 beschrieben wird und die kognitive Lernzielstufe K3 erwartet wird.

Die Prüfung

Dieser Lehrplan umfasst Lerneinheiten und Lernziele für die Zertifizierungsprüfungen zum

- CPRE RE@Agile – Practitioner
- CPRE RE@Agile – Specialist

Die Prüfung zum Erlangen des RE@Agile – Practitioner – Zertifikats besteht aus einer **Multiple-Choice-Prüfung**.

Die Prüfung zum Erlangen des RE@Agile – Specialist – Zertifikats besteht aus einer **schriftlichen Ausarbeitung**.

Beide Prüfungen umfassen Prüfungsfragen zu allen Lerneinheiten und allen Lernzielen des Lehrplans.

Jede Prüfungsfrage kann Stoff aus mehreren Kapiteln des Lehrplans sowie mehreren Lernzielen oder auch von Teilen eines Lernziels beinhalten.

Die Multiple-Choice-Prüfung für das Practitioner Zertifikat

- prüft alle Lernziele des Lehrplans. Bei den Lernzielen der kognitiven Wissensstufen K4 und K5 beschränken sich die Prüfungsfragen jedoch auf Elemente auf den kognitiven Stufen K1 bis K3.
- kann unmittelbar im Anschluss an einen Kurs aber auch unabhängig davon (z. B. remote oder in einem Prüfzentrum) abgelegt werden.

Die schriftliche Ausarbeitung für das Specialist Zertifikat

- prüft alle Lernziele des Lehrplans auf den für die jeweiligen Lernziele angegebenen kognitiven Wissensstufen.
- folgt der Aufgabenbeschreibung für RE@Agile – Specialist –, zu finden unter https://cpre.ireb.org/de/downloads-and-resources/downloads#cpre-re-agile-specialist-written-assignment.
- erfolgt in Eigenregie und wird bei einer lizenzierten Zertifizierungsstelle eingereicht.

Für die schriftliche Ausarbeitung für das Specialist Zertifikat gelten zudem die folgenden generischen Lernziele:

LZ G1: Analysieren und illustrieren von RE@Agile-Problemen in einem dem Lernenden bekannten oder sehr ähnlichen Kontext (L4).

LZ G2: Evaluieren und reflektieren der Anwendung von RE@Agile – Praktiken, Methoden, Prozessen und Werkzeugen in Projekten, an denen die Lernenden beteiligt waren (K5).

Eine Liste der von IREB lizenzierten Zertifizierungsstellen finden Sie auf der Website:

https://www.ireb.org

Versions-Historie

Version	Datum	Kommentar	Autor
1.0.0	10. September 2018	Erste Version	Bernd Aschauer, Lars Baumann, Peter Hruschka, Kim Lauenroth, Markus Meuten, Sacha Reis and Gareth Rogers
1.0.1	10. September 2018	Rechtschreibung korrigiert Einige LE's neu formuliert, um dem Standard zu entsprechen. Inhaltlich keine Änderung. Danksagung hinzugefügt	Stefan Sturm
1.0.2	17. Dezember 2019	Umbenennung des Begriffs „Verfeinerungssitzung" durch den englischen Begriff " Refinement Meeting".	Hans-Jörg Steffe
2.0.0	1. Juli 2022	Lernziele aktualisiert Neufassung von Kapitel 6 Korrekturen in allen Kapiteln Überarbeitung der vorgeschlagenen Unterrichts- und Übungszeiten für alle Kapitel. Information über Advanced Level Prüfungssplit hinzugefügt.	Peter Hruschka, Markus Meuten, Gareth Rogers, Stefan Gärtner, Hans-Jörg Steffe
2.1.0	1. Mai 2024	Neues Corporate Design implementiert, kognitive Stufen des Wissens synchronisiert, Begriff Advanced Level eliminiert, "Voraussetzungen" für das Modul RE@Agile entfernt.	Stan Bühne, Ruth Rossi
2.2.0	24. Mai 2024	Kognitive Stufen des Wissens erneut gefixt.	Stefan Sturm

2.3.0	1. Oktober 2025	Lernziele aktualisiert und ergänzt. Kurzbeschreibung der Kapitel überarbeitet. Illustrationen zu den Kapiteln hinzugefügt.	Hans-Jörg Steffe

Inhalt

1 Was ist RE@Agile (K2)

Dauer: 45 Minuten

Begriffe: Stakeholder, Product Owner, kooperativ, iterativ, inkrementell

Lernziele

LZ 1.1 Kenntnis der Definitionvon RE@Agile (L1)
LZ 1.2 Die Ziele von RE@Agile verstehen (K2)
LZ 1.3 Verstehen, dass die Verantwortung für gute Anforderungen beim Product Owner liegt (K2)

Inhalt

RE und agile Ansätze werden aufgrund ihrer unterschiedlichen Geschichte häufig separat und nicht gemeinsam in Betracht gezogen. Dies führt oft zu dem Missverständnis, dass es zwei Arten von RE gibt: klassisches RE und agiles RE. Nach Ansicht der Autoren gibt es nur ein gutes oder ein schlechtes RE – in einem nicht-agilen oder einem agilen Umfeld. Deshalb bezeichnen wir den Ansatz als RE@Agile.

Agilität und RE sind zwei Disziplinen mit unterschiedlicher Herkunft und klar abgegrenzten Zielen, die jedoch viel voneinander lernen können. Eine unserer Schlussfolgerungen im RE@Agile Primer [Prim2017] lautete: „Der wichtigste Wert ist dem RE und Agilität gemein: die Benutzer des Produkts durch eine Lösung zufriedenstellen, die ihren Bedürfnissen entspricht oder die größten Probleme behebt."

RE@Agile ist ein kooperativer, iterativer und inkrementeller Ansatz mit vier Zielen:

1. Die relevanten Anforderungen in einem angemessenen Detaillierungsgrad zu kennen (zu jedem Zeitpunkt während der Systementwicklung).
2. Eine ausreichende Einigung der relevanten Stakeholder über die Anforderungen zu erzielen.
3. Die Anforderungen gemäß den Rahmenbedingungen der Organisation zu erfassen (und zu dokumentieren).
4. Alle auf Anforderungen bezogenen Aktivitäten gemäß den Prinzipien des Agilen Manifests durchzuführen.

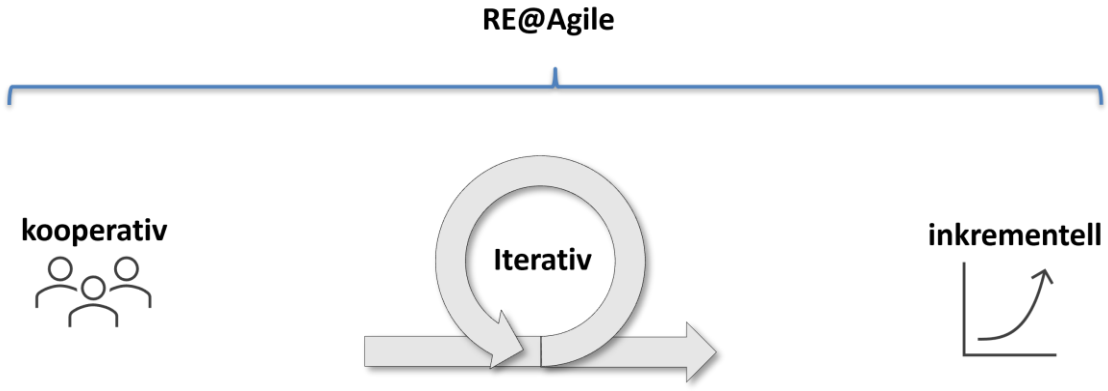

Abbildung 1: Kooperativer, iterativer und inkrementeller Ansatz

Bei agilen Ansätzen stellen verschiedene Rollen Anforderungen an das gewünschte System. Unabhängig davon wer diese Anforderungen liefert, wer sie strukturiert und wer sie detailliert, bleibt die Person mit der Rolle/Verantwortlichkeit des Product Owner für RE verantwortlich.

2 Projekte erfolgreich starten (K3)

Dauer: 120 Minuten + 60 Minuten Übungszeit

Begriffe: Vision, Ziel, Stakeholder, Systemgrenze, Kontextdiagramm, Use Case Diagramm

Inhalt

Selbst bei agilen Herangehensweisen müssen einige wichtige Voraussetzungen geschaffen werden, bevor mit einer erfolgreichen iterativen und inkrementellen Systementwicklung begonnen werden kann.

- Definition der Vision des Systems und/oder der Ziele, die damit erreicht werden sollen
- Identifizierung des derzeit bekannten Systemumfangs und der Systemgrenze
- Identifizierung der relevanten Stakeholder und anderer wichtiger Anforderungsquellen

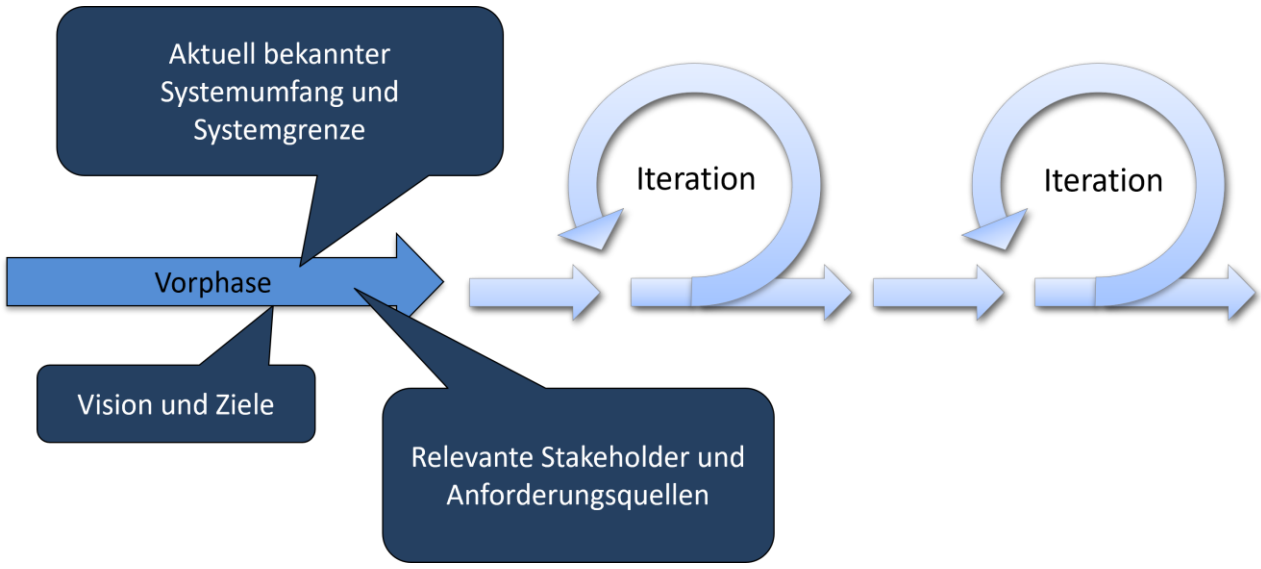

Abbildung 2: Projekte erfolgreich starten

2.1 Vision und Ziele (L3)

Lernziele

LZ 2.1.1 Die Spezifikation von Zielen und Visionen anwenden können (K3)

Inhalt

Anhand der System- oder Produktvision wird das Gesamtziel beschrieben, das mit dem System/Produkt erreicht werden soll. Die Vision ist für jede Entwicklungsaktivität von höchster Bedeutung. Sie legt den Grundstein und dient als allgemeine Richtschnur für sämtliche Entwicklungsaktivitäten. Jede Anforderung sollte zum Erreichen der Systemvision beitragen.

Es gibt dabei alternative Ansätze zur Formulierung von Zielen oder Visionen:

- SMART [Dora1981]

- PAM [Robe2003]
- Product Vision Box [High2001]
- Nachrichten aus der Zukunft [HeHe2011]
- Vision Boards
- Canvas Techniken

2.2 Festlegen der Systemgrenze (K3)

Lernziele

LZ 2.2.1 Systemgrenzen festlegen können (K3)

Inhalt

Ein gemeinsames Verständnis des Systemumfangs und des Kontexts eines Systems sind Voraussetzung für eine effektive und effiziente Entwicklung [Glin2024].

Der Systemumfang und die Systemgrenze lassen sich mithilfe verschiedener Techniken klären und dokumentieren wie:

- Kontextdiagramme
- Natürliche Sprache
- Use Case Diagramme
- Story Maps

2.3 Identifizierung und Management von Stakeholdern (K3)

Lernziele

LZ 2.3.1 Die Identifizierung und das Management von Stakeholdern anwenden können (K3)

Inhalt

Ähnlich wie bei traditionellen Ansätzen müssen auch zu Beginn des agilen Prozesses die wichtigsten Stakeholder identifiziert werden, damit ein Rahmen für die Anforderungserhebung gesetzt wird. Im Agilen können und werden sich die Stakeholder jedoch ständig ändern. Daher ist es wichtig, dass die Identifizierung und Definition der Stakeholder selbst als kooperativer, schrittweiser und vor allem iterativer Prozess verstanden wird (siehe auch Kapitel 1).

Es kann große Auswirkungen haben, wenn man versäumt, einen wichtigen Stakeholder zu identifizieren und in die Entwicklung einzubeziehen. Werden die Anforderungen eines solchen Stakeholders zu spät (oder gar nicht) erkannt, hat dies unter Umständen kostspielige Änderungen oder gar ein unbrauchbares System zur Folge [PoRu2021].

Mithilfe des Zwiebelschalenmodells von Ian Alexander [Alex2005] lassen sich Stakeholder einfach ermitteln und klassifizieren. Das Modell umfasst drei Arten von Stakeholdern („Onion Layers" oder Zwiebelschichten), die systematisch nach Stakeholdern durchsucht werden können:

- Stakeholder des Systems
- Stakeholder des umgebenden Kontexts
- Stakeholder aus dem weiteren Kontext

Als Faustregel gilt, dass bei der Identifizierung von Stakeholdern auf eine große Bandbreite von Quellen zurückgegriffen werden sollte.

Abhängig vom System und von der Domäne, können auch die bereits vorhandene Dokumentation, angrenzende Systeme mit Schnittstellen zum entwickelten System, Altsysteme oder sogar Systeme von Konkurrenten eine wichtige Quelle (zusätzlich zu den Stakeholdern) für Anforderungen darstellen.

2.4 Abhängigkeiten zwischen Visionen/Zielen, Stakeholdern und der Systemgrenze (K3)

Lernziele

LZ 2.4.1 Die dynamische Veränderung von Vision und Zielen, Stakeholdern sowie des Systemumfangs anwenden können (K3)

Inhalt

Die Definitionen von Vision und Zielen sowie Stakeholdern und Systemgrenzen sind dabei voneinander abhängig [PoRu2021]:

- Die relevanten Stakeholder formulieren die Vision und die Ziele. Aus diesem Grund kann sich die Identifizierung eines neuen relevanten Stakeholders auf Vision und Ziele auswirken.
- Mit Visionen und Zielen lässt sich die Identifizierung von neuen Stakeholdern lenken, indem man die Frage stellt: Welcher Stakeholder könnte am Erreichen der Vision und Ziele Interesse haben oder ist von deren Erreichen betroffen?
- Visionen und Ziele können zum Definieren eines initialen Systemumfangs verwendet werden. Dazu muss man folgende Frage beantworten: Welche Elemente sind zum Erreichen der Vision und der Ziele nötig?
- Eine Änderung der Systemgrenze (und dadurch des Systemumfangs) kann sich auf die Vision und die Ziele auswirken. Wird ein Aspekt aus dem Systemumfang entfernt, muss verifiziert werden, dass das System immer noch zum Erreichen der Vision und Ziele ausreicht.
- Stakeholder definieren die Systemgrenze. Deshalb kann sich die Identifizierung eines neuen relevanten Stakeholders auf den Systemumfang auswirken.
- Für Änderungen am Systemumfang (z. B. zur Erfüllung eines Ziels) ist die Zustimmung der relevanten Stakeholder erforderlich.

Diese wechselseitigen Abhängigkeiten sollten genutzt werden, um alle drei Elemente (Vision und Ziele, Stakeholder, Systemumfang) im Gleichgewicht zu halten und die Auswirkungen der Veränderung eines der drei Elemente auf die anderen zu untersuchen.

Aufgrund dieser Abhängigkeiten zwischen Vision und Zielen, Stakeholdern und Systemumfang empfehlen wir, diese Elemente zusammen und kohärent zu behandeln.

3 Umgang mit funktionalen Anforderungen (K4)

Dauer: 195 Minuten + 120 Minuten Übungszeit

Begriffe: Funktionale Anforderungen, Epic, Feature, Story, User Story, Definition of Ready (DoR), INVEST, Granularitätsstufen

Inhalt

In dieser grundlegenden Lerneinheit werden funktionale Anforderungen in erster Linie aus einer statischen Perspektive betrachtet, d. h. die Strukturierung eines großen Anforderungsumfangs in Abstraktionshierarchien.

Wenn einmal die Idee akzeptiert wurde, dass Anforderungen auf verschiedenen Granularitätsstufen existieren, wirft das in der Regel einige Fragen auf:

- Wie gehen wir mit verschiedenen Granularitätsstufen um?
- Welche Kriterien können und sollten angewendet werden, um große, abstrakte Topics in kleinere Blöcke zu untergliedern?
- Ist es manchmal erforderlich, viele kleine Anforderungen in größere Blöcke zusammenzufassen, sodass wir einen „größeren Zusammenhang" zur besseren Orientierung haben?
- Wie präzise müssen wir sein, bevor die Entwickler mit der Implementierung beginnen können?
- Ist es notwendig oder ratsam, verschiedene Anforderungsniveaus beizubehalten oder können wir abstrakte Aussagen verwerfen, sobald wir konkretere Anforderungen haben?
- Strukturieren wir das Backlog lieber nach funktionalen Zusammenhängen/Prozessen, oder lieber nach anderen Zusammenhängen wie z. B. technische Zusammenhänge?
- Müssen wir all das schriftlich festhalten oder genügt es, wenn wir einfach darüber sprechen?

3.1 Unterschiedliche Stufen der Anforderungsgranularität

Lernziele

LZ 3.1.1 Die Existenz von funktionalen Anforderungen auf unterschiedlichen Granularitätsstufen kennen (K1)

Inhalt

Stakeholder kommunizieren Anforderungen gewöhnlich auf unterschiedlichen Stufen der Granularität. Manchmal wünschen sie, dass große Funktionalitätsblöcke hinzugefügt oder geändert werden, manchmal nur geringfügige Details.

Epics (manchmal auch Themes genannt) oder große Storys (die potenziell komplexe Geschäftsprozesse darstellen) sind eine gute Möglichkeit, ein Gesamtbild über alle aktuell bekannten Anforderungen im Backlog zu erhalten, d. h. einen Überblick über all die Dinge, die Stakeholder von einem System oder einem Produkt erwarten. Epics werden in detailliertere

Elemente zerlegt, in der Regel Features und Storys. Storys bieten eine feinere Granularität und können Anforderungen aus verschiedenen Perspektiven kommunizieren, z. B. aus der Sicht des Benutzers oder aus der technischen Perspektive.

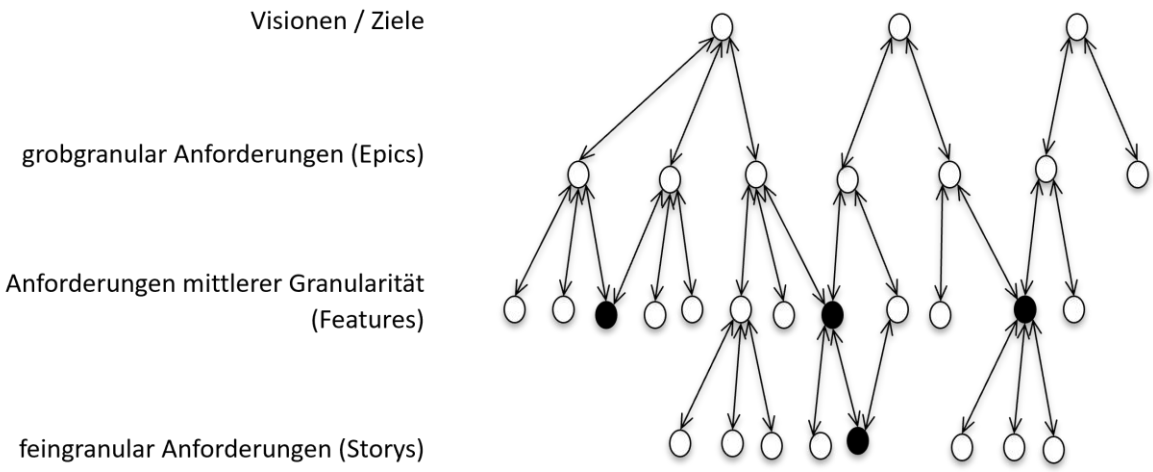

Abbildung 3: Granularität von Anforderungen

3.2 Kommunizieren und Dokumentieren auf unterschiedlichen Stufen

Lernziele

LZ 3.2.1 Die Dekomposition von Anforderungen auf oberster Ebene analysieren und anwenden können (K4)

LZ 3.2.2 Unterschiedliche Dekompositionsstrategien für eine sehr große Anzahl von Anforderungen analysieren und anwenden können (K4)

LZ 3.2.3 Identifizierung, Dokumentation und Kommunikation von funktionalen Anforderungen auf unterschiedlichen Granularitätsstufen analysieren und anwenden können (K4)

Inhalt

Auf Basis des Prinzips „teile und herrsche" müssen wir ein großes System oder Produkt in kleinere Teile zerlegen.

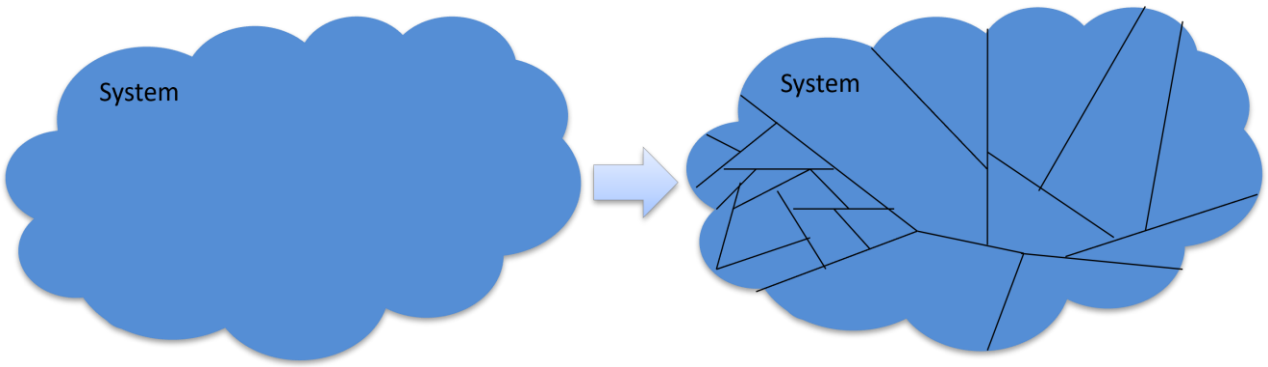

Abbildung 4: Zerlegung des Systems in kleinere Teile

Dabei gibt es unterschiedliche Herangehensweisen, um dieses Ziel zu erreichen (z. B. Aufteilung in logische Funktionen, Aufteilung nach organisatorischen Aspekten, Aufteilung nach Hardware, Aufteilung nach Daten, ...). Wir sehen uns dabei die Herangehensweise zur Aufteilung der Anforderungen in extern angestoßene, wertschaffende Prozesse an.

Neben der Zerlegung der Anforderungen in kleinere Teile müssen wir uns auch um das Kommunizieren und Dokumentieren der funktionalen Anforderungen kümmern. Die grundlegende Entscheidung liegt dabei zwischen Zeichnen (z. B. mit Modellen) und Schreiben (z. B. in natürlicher Sprache). Dabei ist es vor allem eine Entscheidung des gesamten Teams welche Form der Darstellung bevorzugt wird. So könnte die Übersicht z. B. mittels Use Case Diagramm erfolgen, während die Prozessschritte der Use Cases mittels Prosa (z. B. als User Story) beschrieben wird.

3.3 Arbeiten mit User Storys und Backlog Items

Lernziele

LZ 3.3.1 Anwendung der Idee des nutzerzentrierten Denkens auf die Anforderungen mit User Storys (L3)
LZ 3.3.2 Die Erstellung von Anforderungen auf Basis der User-Story Idee und mithilfe der User-Story Vorlage durchführen können (K3)
LZ 3.3.3 Die INVEST Kriterien bei der Erstellung von Anforderungen anwenden können (K3)

Inhalt

User Storys haben in der agilen Umgebung in den letzten Jahren eine enorme Bedeutung bekommen, weil sie auf die Wichtigkeit hindeuten, Anforderungen und daraus resultierende Produkte anwenderzentriert zu strukturieren und zu entwickeln. Also den Nutzer in den Vordergrund zu stellen. Diese Nutzersicht ist wichtig, weil wir primär agil entwickeln, um regelmäßig Feedback von den Benutzern zu erhalten.

Es ist wichtig anzumerken, dass der Begriff „User Story" im agilen Umfeld oft für drei verschiedene Zwecke verwendet wird:

- Als benutzerorientierte Methode zur Diskussion und Formulierung von Anforderungen.

- Als unterste (detaillierteste) Ebene der Anforderungsstrukturierung (Epics -> Features -> (User) Storys). In der Tat wird in der Anforderungsstruktur häufig der Begriff „User Story" anstelle von „Story" verwendet.
- Als Template zur Beschreibung von Anforderungen auf verschiedenen Hierarchieebenen aus der Sicht eines Benutzers.

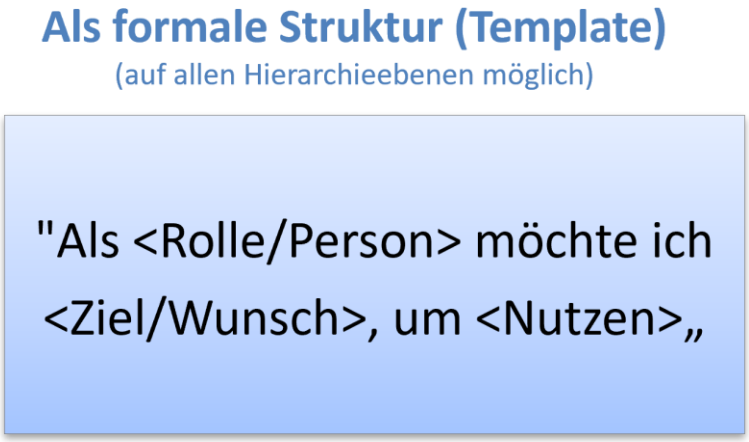

Abbildung 5: Vorlage für eine User Story

Um diese Mehrdeutigkeit zu entschärfen, verwenden wir in diesem Lehrplan den Begriff „Story" für die feinergranularen Elemente der Anforderungsstruktur; eine „User Story" ist also eine besondere Art von Story, d. h. eine benutzerzentrierte Story. Wenn wir uns auf die in Abbildung 5 dargestellte Vorlage beziehen, werden wir den Begriff „User Story Template" verwenden. Bitte beachten Sie, dass das User Story Template nicht nur zur Beschreibung von Storys, sondern auch von Epics und Features verwendet werden kann, wenn dies angemessen erscheint.

Generell ist bei User Storys darauf hinzuweisen, dass diese an sich noch keine vollständigen Anforderungen darstellen, User Storys sind vielmehr ein Kommunikationsversprechen. Um daraus vollständige Anforderungen und dementsprechend Backlog Items zu erstellen, wird eine weitere Detailierung (dokumentiert und/oder im Gespräch) stattfinden müssen.

Im Zuge der Einführung von User Storys [Cohn2010] hat Bill Wake [Wake2003] definiert, dass User Storys den INVEST (Independent, Negotiable, Valuable, Estimated, Small, Testable) Prinzipien entsprechen sollten.

Backlog Items werden zusätzlich zu IT Tools (z. B. Jira MS Devops) häufig auf Karteikarten oder Haftnotizzetteln erfasst und zur Erleichterung von Planungen und Diskussionen an Wänden oder in Tabellen angeordnet. Mittels 3C Modell (Card, Conversation, Confirmation) können diese Backlog Items dann detaillierter besprochen, und mit weiteren Anforderungsartefakten (z. B. Modelle, Dokumentationen) ergänzt werden.

3.4 Aufteilungs- und Gruppierungstechniken

Lernziele

LZ 3.4.1 Aufteilungstechniken für grobe funktionale Anforderungen analysieren und anwenden können (K4)

LZ 3.4.2 Das Gruppieren und Abstrahieren von detailgenauen funktionalen Anforderungen in gröbere Anforderungen analysieren und anwenden können (K4)

Inhalt

Um Backlog Items zu erzeugen, die klein genug für eine einzige Iteration sind, können Sie größere Backlog Items in detailliertere (Storys) aufteilen. Eine Reihe von Autoren haben Muster vorgestellt, die von der Verringerung der Feature-Liste bis zur Eingrenzung der geschäftlichen Variationen oder Informationskanäle reichen und für diesem Zweck verwendet werden können [Leffl2010]. Beachten Sie, dass selbst detaillierte Storys so definiert werden sollten, dass sie für mindestens einen Stakeholder einen gewissen Wert liefern.

Kleinere Backlog Items können für die Übersichtlichkeit in größere Blöcke oder Prozesse (z. B. Use Cases) zusammengefasst werden und mittels Story Map grafisch dargestellt werden. Das hilft dabei, den Überblick über die Anforderungen zu behalten und gleichzeitig die mittelfristige Strategie durch die Zuordnung der Anforderungen zu Sprints und Releases darzustellen.

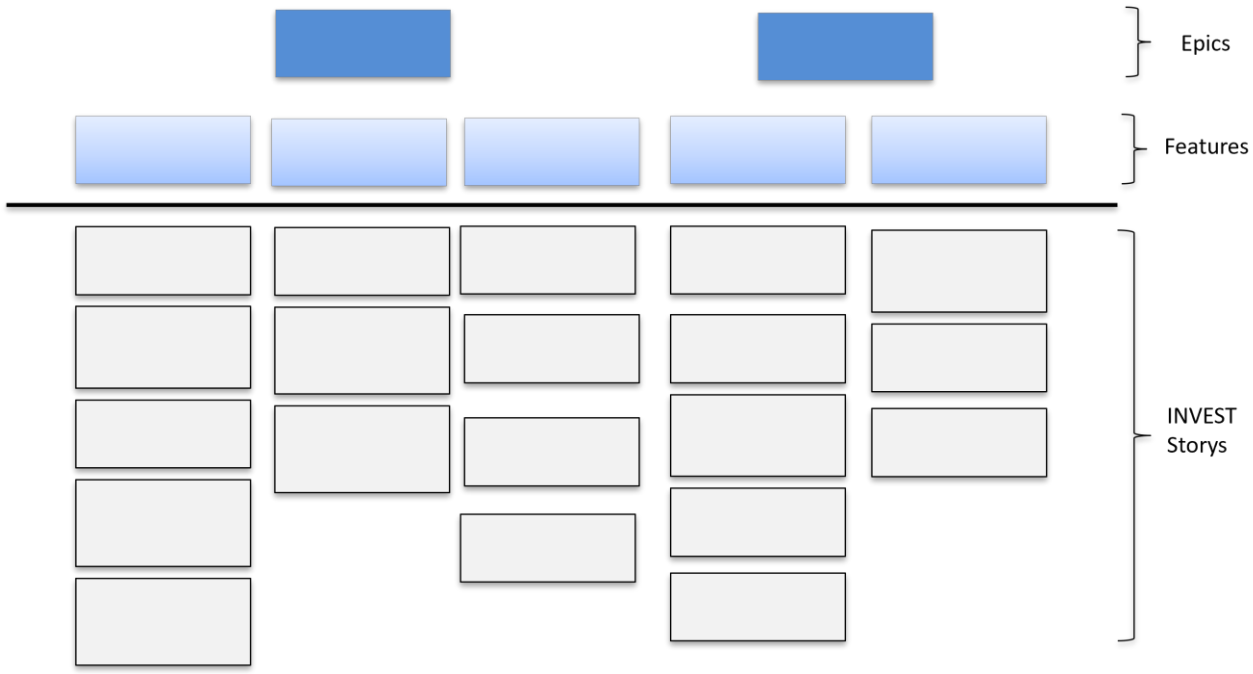

Abbildung 6: Aufteilung und Gruppierung

3.5 Wissen, wann man aufhören sollte

Lernziele

LZ 3.5.1 Die Verfeinerung von Anforderungen anwenden können (K4)

LZ 3.5.2 Qualitätssicherung von Storys im agilen Umfeld anwenden können (K4)

Inhalt

Der Product Owner ist für die Fortführung der Diskussionen mit Entwicklern verantwortlich, bis beide Seiten ein gemeinsames Verständnis von den Anforderungen erreicht haben Meye2014].

Für diese Stufe des gemeinsamen Verständnisses ist die Definition of Ready (DoR) definiert, welche für die Qualitätssicherung von Backlog Items verwendet werden kann, um sicherzustellen, dass ein gemeinsames Verständnis und ein ausreichender Detailgrad erreicht wurde.

3.6 Projekt- und Produktdokumentation von Anforderungen

Lernziele

LZ 3.6.1 Verstehen, wie zwischen Projekt- und Produktinformationen/Dokumentation unterschieden wird (K2)
LZ 3.6.2 Methoden und Techniken zur Aufbewahrung von Informationen für die künftige Verwendung kennen (K1)

Inhalt

Ein Product Backlog kann als Ersatz für das Anforderungsdokument eines traditionellen Projekts betrachtet werden. Es ist jedoch wichtig, daran zu denken, dass der schriftliche Teil eines Backlog Items (z. B. unter Verwendung des User Story Templates: „Als Benutzer möchte ich ...") unvollständig ist, bis die Diskussionen über dieses Backlog Item stattgefunden haben.

Es ist oft am besten, sich den schriftlichen Teil als Referenz auf eine präzisere Darstellung dieser Anforderung vorzustellen. Backlog Items (Epics, Features, Storys, ...) könnten auf ein Diagramm hinweisen, das einen Workflow darstellt, auf eine Tabellenkalkulation, in der die Durchführung einer Berechnung veranschaulicht wird, oder auf ein beliebiges anderes Artefakt, das der Product Owner oder das Team wünscht.

Zusätzlich zum dem Product Backlog kann es jedoch unterschiedliche Zwecke für eine detaillierte Anforderungsdokumentation, so für Kommunikation, Überlegungen, gesetzliche Vorgaben zur Bewahrung.

Das Definieren eines adäquaten Maßes an Dokumentation hängt von zahlreichen Faktoren ab wie vom Umfang des Projekts, der Anzahl der beteiligten Stakeholder, von rechtlichen Randbedingungen und/oder von den sicherheitskritischen Aspekten des Projekts. Basierend auf diesen Faktoren versuchen Teams im agilen Umfeld, ein Übermaß an Dokumentation zu vermeiden und den kleinsten Nenner für nützliche Dokumentation zu finden.

Das Arbeiten mit einem „lebenden" Product Backlog ist zwar eine effiziente Weise des Umgangs mit Dokumentation, doch sie ist nicht immer ausreichend. Eine strukturierte Dokumentation aller in einem Produkt implementierten Anforderungen, die auf dem neuesten Stand ist, ist in manchen Projekten nicht nur rechtlich verpflichtend, sondern sie

dient auch als idealer Ausgangspunkt für die schnellere Identifizierung von Änderungsanträgen basierend auf der vorhandenen Dokumentation.

4 Umgang mit Qualitätsanforderungen und Randbedingungen (K3)

Dauer: 90 Minuten + 30 Minuten Übungszeit

Begriffe: Qualitätsanforderungen, Randbedingungen, Qualitätsbaum, Definition of Done (DoD), Akzeptanzkriterium

Inhalt

In dieser Lerneinheit werden die Qualitätsanforderungen und Randbedingungen in agilen Entwicklungsprojekten beleuchtet. Auch wenn der Begriff „nicht-funktionale Anforderungen" in der Praxis immer noch häufig als Überbegriff zu finden ist, verwenden wir die konkreteren und präziseren Kategorien „Qualitätsanforderungen" und „Randbedingungen" nach [Glin2024].

4.1 Die Bedeutung von Qualitätsanforderungen und Randbedingungen verstehen (K2)

Lernziele

LZ 4.1.1 Die Bedeutung von Qualitätsanforderungen im agilen Umfeld verstehen (K2)
LZ 4.1.2 Kategorisierungsschemata für Qualitätsanforderungen und Randbedingungen verstehen (K2)

Inhalt

Viele agile Methoden sind ausschließlich auf funktionale Anforderungen ausgerichtet und legen nicht genügend Gewicht auf Qualitätsanforderungen und Randbedingungen Meye2014].

Die für ein System vorgesehenen zentralen Randbedingungen und Qualitäten sollten im Lebenszyklus eines Produkts frühzeitig explizit benannt werden, da sie für wichtige Entscheidungen hinsichtlich der Architektur ausschlaggebend sind (Infrastruktur, Softwarearchitektur und Softwaredesign). Werden sie in einem Projekt ignoriert oder zu spät erkannt, gefährdet dies unter Umständen die gesamte Entwicklungstätigkeit. Andere Qualitäten können iterativ erfasst werden, just in time, so wie es auch bei funktionalen Anforderungen der Fall ist Meye2014].

Kategorisierungsschemata für Qualitätsanforderungen und Randbedingungen (z. B. [RoRo2013], [ISO25010]) können als Checklisten verwendet werden, damit keine wichtigen Kategorien vergessen werden.

4.2 Qualitätsanforderungen präzisieren (K2)

Lernziele

LZ 4.2.1 Detaillierung und Zerlegung von Qualitätsanforderungen und Randbedingungen verstehen (K2)
LZ 4.2.2 Ableitung Funktionaler Anforderungen aus Qualitätsanforderungen verstehen (K2)

LZ 4.2.3 Spezifikation von Akzeptanzkriterien für Qualitätsanforderungen verstehen (K2)
LZ 4.2.4 Verstehen des Mehrwerts von Qualitätsbäumen (K2)

Inhalt

Zu Beginn sind Qualitätsanforderungen oftmals bewusst vage formuliert. Sie müssen als Ausgangsbasis in dieser vagen Form erfasst werden. Vage Qualitätsanforderungen und Randbedingungen können in präzisere Anforderungen transformiert werden. Dabei werden manchmal aus vagen Qualitätsanforderungen konkrete funktionale Anforderungen abgeleitet.

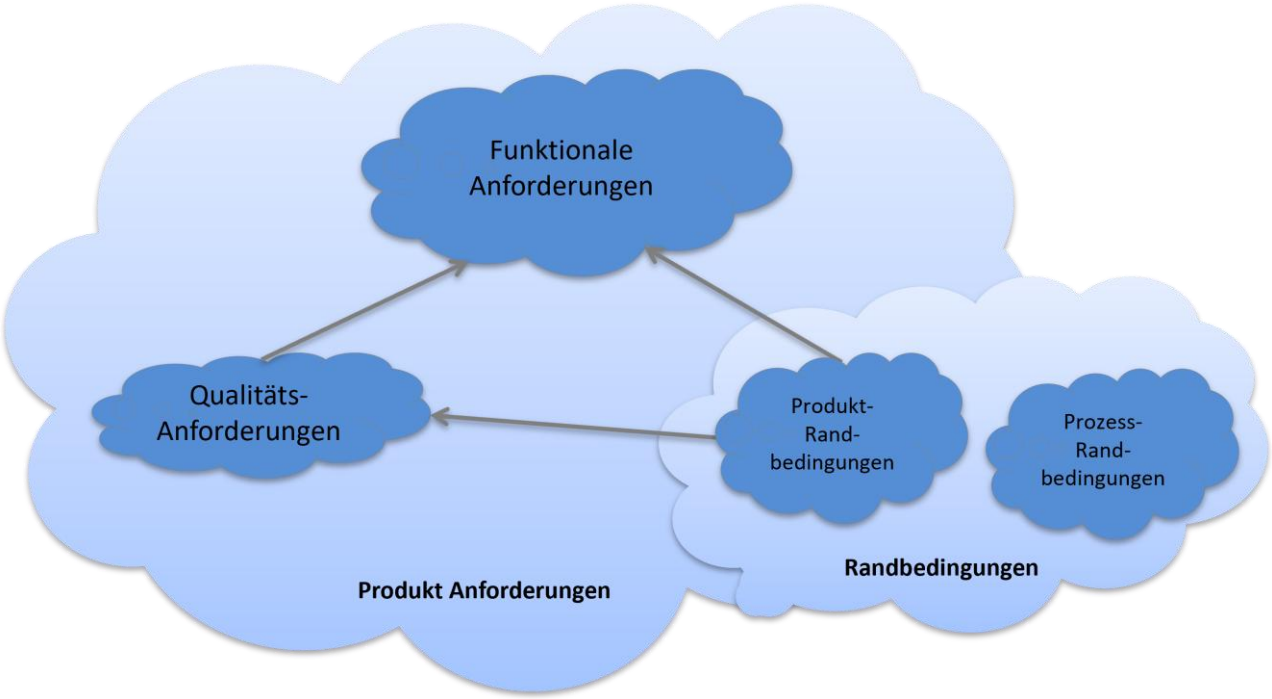

Abbildung 7: Qualitätsanforderungen und Randbedingungen zerlegen

Die Zerlegung (oder Detaillierung) von Qualitätsanforderungen bedeutet, dass diese auf einem niedrigeren Detaillierungsgrad spezifiziert werden, beispielsweise durch Generalisierung in Kategorisierungsschemata wie „Benutzbarkeit" und anschließend eine Präzisierung der Anforderungen erfolgt, beispielsweise für „einfache Benutzbarkeit" und „leichte Erlernbarkeit".

Ableiten bedeutet, dass man Qualitätsanforderungen durch das Definieren funktionaler Anforderungen erreichen kann, d.h. durch das Vorschlagen von Funktionen, die eine gewünschte Qualität oder Randbedingung bewirken. Ein Beispiel für die Verfeinerung einer Sicherheitsanforderung ist die Einführung eines Rollenkonzepts und Passwörtern.

Qualitätsbäume ([BOSS2022], [CleA2001]) sind ebenfalls ein bewährtes Verfahren, um Qualitätsanforderungen zu strukturieren.

Für Qualitätsanforderungen müssen ebenfalls Akzeptanzkriterien definiert werden um dieser in späterer Folge, genau wie andere Arten von Anforderungen, testbar zu machen

[PoRu2021]. Die Art der verwendeten Akzeptanzkriterien hängt von der Qualitätskategorie ab.

4.3 Qualitätsanforderungen und das Backlog (K3)

Lernziele

LZ 4.3.1 Die Zuordnung von Qualitätsanforderungen zu funktionalen Anforderungen anwenden können (K3)
LZ 4.3.2 Das separate Anlegen von Backlog Items für Qualitätsanforderungen anwenden können (K3)
LZ 4.3.3 Qualitätsanforderungen als Teil der DoD verstehen (K2)
LZ 4.3.4 Den Unterschied zwischen Qualitätsanforderungen und Akzeptanzkriterien verstehen (K2)

Inhalt

Generalisierte Qualitätsanforderungen müssen mit spezifischeren funktionalen Anforderungen verknüpft werden [PoRu2021], z. B. ein quantifizierbarer Durchsatz, der einer Story hinzugefügt wird oder spezifische Sicherheitsfunktionen, die ein Epic ergänzen.

Andere Qualitäten wie Skalierbarkeit, Wartbarkeit oder Robustheit sollten der Entwicklung mitgeteilt und in jeder einzelnen Iteration geprüft werden. Zu diesem Zweck werden diese Qualitäten üblicherweise in die Definition of Done aufgenommen. Unterstützt wird dies häufig durch automatisierte Tests [Leffl2010].

Ein weiterer Ansatz ist die separate Aufzeichnung (außerhalb des Product Backlogs) dieser Qualitäten, etwa in Form einer gemeinsamen Liste oder einer Checkliste, damit sie für die Teams immer präsent sind. Diese Anforderungen sind alle gleichrangig (da sie alle erfüllt werden müssen) [Leffl2010].

Es hat sich außerdem bewährt, die Beziehungen zwischen funktionalen und betroffenen Qualitätsanforderungen, etwa durch eine Matrix an einer Wand, sichtbar zu machen und die Beziehung „betroffen von" in den entsprechenden Zellen zu kennzeichnen.

Bei der Strukturierung eines Backlogs stellt sich für Product Owner öfter die Frage, ob eine erkannte/erhobene Qualitätsanforderung nun wirklich eine Qualitätsanforderung, ein Akzeptanzkriterium, oder vielleicht auch ein Akzeptanzkriterium für eine Qualitätsanforderung, ist.

- Qualitätsanforderungen beziehen sich auf Qualitätsaspekte, die nicht durch funktionale Anforderungen abgedeckt sind. Wie z. B. Leistung (Performance), Verfügbarkeit, Wartbarkeit, Sicherheit oder Zuverlässigkeit.
- Akzeptanzkriterien sind Kriterien, die eine Anforderung (dies kann eine funktionale Anforderung oder eine Qualitätsanforderung sein) erfüllen muss, um von den Stakeholdern akzeptiert zu werden.

Wir sehen also, dass sowohl funktionale Anforderungen als auch Qualitätsanforderungen Akzeptanzkriterien haben können und haben sollten.

4.4 Randbedingungen verdeutlichen (K2)

Lernziele

LZ 4.4.1 Verschiedene Arten von Randbedingungen im agilen Umfeld verstehen (K2)
LZ 4.4.2 Die Charakterisierung von Randbedingungen kennen (K1)

Inhalt

Randbedingungen sind eine wichtige Art von Anforderungen, welche die Designentscheidungen der Entwickler eingrenzt [Glin2024]. Randbedingungen lassen sich entweder als Produkt- oder als Prozessrandbedingungen kategorisieren. Produktrandbedingungen beinhalten die geforderte Verwendung von Technologien, die Wiederverwendung vorhandener Komponenten, Entscheidungen über „entwickeln oder kaufen", bzw. Ressourcen in Form von Material, Wissen und Kompetenzen. Die Prozessrandbedingungen geben hingegen organisatorische oder Entwicklungsprozesse vor. Dazu zählen organisatorische Richtlinien und Vorschriften, finanzielle Limits, Normen, Compliance-Vorschriften und Audits sowie rechtliche und behördliche Randbedingungen.

Es ist wichtig, diese Randbedingungen explizit zu machen, damit sich jeder im Team dieser bewusst ist. Randbedingungen, die für die größten Einschränkungen sorgen, sollten im Projektverlauf frühzeitig bekannt sein. Sonstige Randbedingungen sollten erfasst werden, sobald sie erkannt werden. Generell betreffen Randbedingungen meist mehrere funktionale Anforderungen. So stellt sich auch bei den Randbedingungen (genauso wie bei den Qualitätsanforderungen) die Frage, wie diese dokumentiert werden sollten. Die Möglichkeiten dazu haben wir bereits in Kapitel 4.3 erläutert. Das Gleiche gilt für die Randbedingungen: Zumindest sollte in die Definition of Done eine Prüfung aufgenommen werden, um sicherzustellen, dass sie eingehalten wurden.

Beachten Sie, dass Sie nicht zwingend alle Randbedingungen als Backlog Item verfassen müssen. Es ist gegebenenfalls ausreichend, das Team darüber zu informieren, dass z. B. C# und ORACLE nicht verhandelbare Randbedingungen sind.

Derartige Randbedingungen gelten normalerweise für eine Vielzahl von Projekten. Einmal aufgezeichnet, können sie problemlos in anderen Projekten oder Produkten wiederverwendet werden.

5 Priorisieren und Schätzen von Anforderungen (K3)

Dauer: 120 Minuten + 90 Minuten Übungszeit

Begriffe: Geschäftswert, MVP, MMP, Planning Poker, Cone of Uncertainty, Velocity, Sizing, Referenz-Storys, T-Shirt Größen, Fibonacci-Folge

Inhalt

Selbst in einer perfekten agilen Welt ist man auf Prognosen angewiesen, um zu ermitteln, wie viel Arbeit in einer zuvor festgelegten Iteration (Timebox) erledigt werden kann. Zudem sind Entwicklungsorganisationen mit mehr als einem Team gewöhnlich auf Prognosen angewiesen, um die Arbeit richtig zu priorisieren und zu planen.

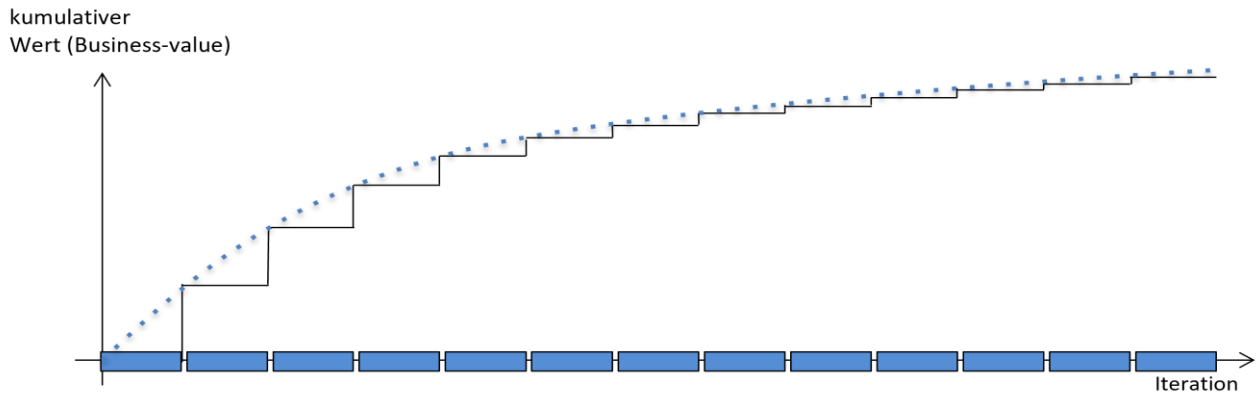

Abbildung 8: Wachsender Geschäftswert

5.1 Ermittlung des Geschäftswerts

Lernziele

LZ 5.1.1 Die Ermittlung des Geschäftswerts verstehen (K2)
LZ 5.1.2 Verstehen, wie man den Geschäftswert nutzt, um Backlog Items zu sortieren (K2)
LZ 5.1.3 Alternative Berechnungsmethoden für den Geschäftswert anwenden können (K3)
LZ 5.1.4 Verstehen, wie man Maßnahmen zur Bewertung des Geschäftswerts auf strategische Ziele der Organisation ausrichtet (K2)

Inhalt

Agile Herangehensweisen zielen darauf ab, den gesamten Geschäftswert zu maximieren und den Erstellungsprozess des Geschäftswerts insgesamt dauerhaft zu optimieren [Leffl2010]. In einem Product Backlog sollten sämtliche Anforderungen (ob grob- oder feingranular) in erster Linie nach dem Wert sortiert werden, den sie der Organisation bieten. Eine Voraussetzung dafür ist eine abgestimmte Definition dessen, was den Geschäftswert für das Produkt/Unternehmen ausmacht.

Der Geschäftswert wird nicht nur über den Profit definiert: Zu alternativen Berechnungsmethoden zählen u. a. die Gesamtkapitalrentabilität (Return on Investment, ROI), die Amortisationszeit, der Kapitalwert (Net Present Value), Weighted Shortest Job First (WSJF, „Wert/Aufwand-Verhältnis"), die Kosten für Verzögerungen (Cost of Delay) und die Balanced Scorecard. Der Marktwert, die Markteinführungszeiten (time to market) und die Reduzierung potenzieller Risiken stellen allesamt potenzielle Arten von Geschäftswert dar, ebenso wie betriebliche und organisatorische Exzellenz [Rein2009].

Die Definition des Geschäftswerts kann in der Tat für jede Organisation, jedes Projekt und aus der Perspektive unterschiedlicher Stakeholder anders aussehen. Experten sollten wissen, wie sie Maßnahmen zum Erzielen des Geschäftswerts auf die strategischen Ziele der Organisation ausrichten und sie sollten diese Ausrichtung bei einer Veränderung der Ziele anpassen können.

5.2 Geschäftswert, Risiken und Abhängigkeiten (K3)

Lernziele

LZ 5.2.1 Die Abhängigkeiten zwischen potenziellem Geschäftswert und zugehörigen Risiken verstehen (K2)

Inhalt

Es bestehen sehr häufig wechselseitige Abhängigkeiten zwischen potenziellem Geschäftswert und Risiken. Die Konzentration auf einen bestimmten Geschäftswert kann gewisse Risiken verursachen, wird der Fokus auf den Geschäftswert geändert, können sich auch die Risiken ändern [Rein2009].

In jedem Fall sollte die Sortierung von Anforderungen gemäß der gewählten Strategie angepasst und dabei die Abhängigkeiten zwischen den Anforderungen berücksichtigt werden.

5.3 Äußern von Prioritäten und Sortieren des Backlogs

Lernziele

LZ 5.3.1 Die Priorisierung von Backlogs anwenden können (K3)
LZ 5.3.2 Verschiedene grundlegende Priorisierungsstrategien anwenden können (K3)
LZ 5.3.3 Die Ermittlung von Abhängigkeiten zwischen Anforderungen anwenden können (K3)
LZ 5.3.4 Die Reihenfolge von Backlog Items durch Berücksichtigung der Abhängigkeiten verstehen (K2)

Inhalt

Sobald Sie festgelegt haben, was Wert für Sie bedeutet, müssen Sie diese Prioritäten zum Ausdruck bringen und das Backlog nach den Prioritäten der Backlog Items sortieren. Es gibt viele verschiedene Methoden, um Backlog Items einen Wert zuzuweisen. Einige davon sind sehr einfach, andere hochkomplex. Beispiele für Priorisierungsstrategien sind:

- MoSCoW

- High/Medium/Low
- Anhand einer Bandbreite an Zahlen die den Geschäftswert (Business-value) ausdrücken
- Anhand einer Wertermittlung des Geschäftswertes auf Basis mehrerer Kriterien

Backlog Items	Umsatz im 2. Quartal		Minimierung des technischen Risikos		Erhöhung der Benutzbarkeit		Gesamt-ergebnis
	Gewicht = 5	Umsatz Wert	Gewicht = 4	Risiko Wert	Gewicht = 2	Benutzbarkeit Wert	
Anforderung 1	3	15	0	0	0	0	**15**
Anforderung 2	0	0	3	12	1	2	**14**
Anforderung 3	4	20	2	8	2	4	**32**
Anforderung 4	2	10	2	8	3	6	**24**
Anforderung 5	0	0	2	8	5	10	**18**
...							
...							

Abbildung 9: Prioritäten und Sortieren des Backlogs

Bei der Priorisierung der Backlog Items sollte generell beachtet werden, dass je näher der Zeitpunkt der geplanten Umsetzung eines Backlog Items kommt, desto klarer sollten auch die Prioritäten der gewählten Einträge sein. Eine Priorisierung mit den Kriterien „hoch", „mittel", „niedrig" ergibt oftmals, dass viel zu viele Backlog Items den Wert „hoch" erhalten. Das Ergebnis, wenn z. B. 30 % aller Backlog Items die Priorität „hoch" erhalten ist, dass die Entwickler nicht wissen, was dem Product Owner am wichtigsten ist. Es weist auch darauf hin, dass der Product Owner keine klare Strategie für die kurz- bis mittelfristige Umsetzung hat. Das Ziel der Prioritätensetzung sollte immer sein, eine klare Aussage zu treffen, was die Stakeholder demnächst als Wert des Produktes erwarten dürfen.

5.4 Schätzen von Backlog Items (L3)

Lernziele

LZ 5.4.1 Prognosen und Schätzungen anwenden können (K3)
LZ 5.4.2 Verstehen, wie man eine mittelfristige Prognose ableitet (K2)
LZ 5.4.3 Die Vorteile von relativen, kategorisierenden und Gruppenschätzungen verstehen (K2)
LZ 5.4.4 Verstehen von Schätzungstechniken (K2)

Inhalt

Die initialen Projektschätzungen sind häufig ungenau. Sie werden jedoch zunehmend genauer, da die Aktivität iteriert wird (ein Prinzip, das als „Cone of Uncertainty" bekannt ist). Durch die Analyse, was in vorherigen Iterationen geliefert wurde, kann die Arbeitsgeschwindigkeit („velocity") des Teams berechnet werden. Damit lässt sich die Kapazität für künftige Iterationen besser schätzen.

Für bessere mittelfristige Schätzungen werden große Anforderungen wie Epics auf Features heruntergebrochen. So können Schätzungen auf Feature-Ebene durchgeführt und dann zum Epic summiert werden. Diese Schätzungen sind zwar immer noch nicht sehr präzise, aber für die Epic-Schätzung hilfreich. Außerdem dienen sie als eine Art Prüfung der Kenntnisse über das Epic (Annahmen usw.).

Agile Methoden empfehlen einige bewährte Verfahren, mit deren Hilfe bessere und präzisere Schätzungen möglich sind:

- Alle an der Schätzung Beteiligten müssen dasselbe Verständnis der zu erledigenden Arbeit haben.
- Schätzungen müssen von denjenigen vorgenommen werden, die die Arbeit ausführen, also dem cross-funktionalen Team (Entwickler in Scrum). Durch den Austausch von Wissen und Annahmen über die zu erledigende Arbeit werden alle beteiligten Personen auf denselben Wissensstand gebracht.
- Schätzungen sollten relativ zur bereits erledigten Arbeit vorgenommen werden (Schätzung durch Analogie), da die Wahrscheinlichkeit größer ist, dass sie präziser ausfallen als absolute Schätzungen.
- Schätzungen sollten in einer künstlichen Einheit erfolgen, die Aufwand, Komplexität und Risiko vereint.

Für relative Schätzungen stehen mehrere Techniken zur Verfügung. Die bekanntesten davon sind das sogenannte Planning Poker [Cohn2006] sowie Magic Estimation.

5.5 Auswählen einer Entwicklungsstrategie (K2)

Lernziele

LZ 5.5.1 Das Konzept des MVP (Minimum Viable Product) verstehen (K2)
LZ 5.5.2 Das Konzept des MMP (Minimum Marketable Product) verstehen (K2)

Inhalt

Beim Auswählen der Elemente für frühe Releases können basierend auf bekannten Werten, Risiken und dem für die Entwicklung eines Backlog Items benötigten Aufwand verschiedene Strategien zum Einsatz kommen. Es gibt bei agilen Entwicklungsprozessen zwei typische Konzepte: Die Entwicklung eines Minimum Viable Product (MVP) und die Entwicklung eines Minimum Marketable Product (MMP).

Mit Minimum Viable Product ist sinngemäß die Version eines neuen Produkts gemeint, die einem Team das Erfassen des maximal möglichen validierten Lernens über Kunden bei geringstem Aufwand ermöglicht. Das MVP ist eine zentrale Idee der von Eric Ries entwickelten Lean-Startup-Methodologie, die auf dem „Build-Measure-Learn"-Zyklus (bauen, messen, lernen) beruht. Das MVP ist dabei nicht notwendigerweise ein einsatzbares Softwareprodukt.

Das MMP beschreibt das Produkt mit dem kleinstmöglichen Feature-Set, das auf die Bedürfnisse der ersten Benutzer (Innovatoren und frühe Anwender) eingeht und somit vermarktet werden kann.

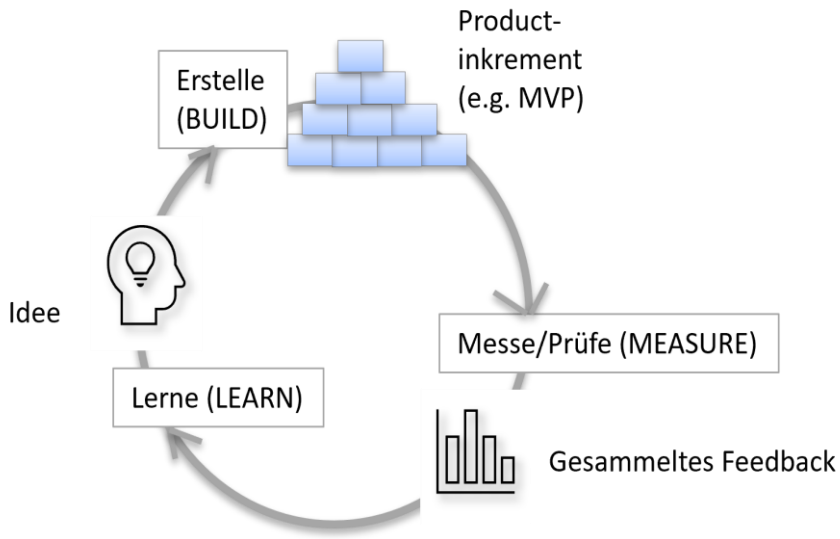

Abbildung 10: Der „Build-Measure-Learn"-Zyklus des Lean Development

6 Skalierung von RE@Agile (K2)

Dauer: 105 Minuten

Begriffe: Skalierungs-Frameworks, Skalierung von Anforderungen und Teams, Strukturierung von Anforderungen und Teams im Großen, Roadmaps und umfangreiche Planung, Produktvalidierung

Inhalt

RE ist einfacher für Produkte, die so klein sind, dass sie von einem einzigen Team an einem Standort bearbeitet werden können.

In dieser Lerneinheit wird erörtert, warum die Produktentwicklung manchmal skaliert werden muss und warum Produkte von mehr als einem Team entwickelt werden müssen, sei es am gleichen Standort oder geografisch verteilt. Wenn skaliert wird, ist der Product Owner des Gesamtprodukts (als die für das Anforderungsmanagement verantwortliche Rolle) wahrscheinlich mehr mit Managementaspekten als mit Anforderungsaspekten konfrontiert. Wir werden erörtern, dass die beiden Faktoren Markteinführungszeit und Komplexität (entweder funktionale Komplexität oder anspruchsvolle Qualitätsanforderungen) den Skalierungsprozess rechtfertigen und vorantreiben. Aber auch organisatorische und technische Randbedingungen beeinflussen die Art und Weise der Skalierung.

Wichtig sind die folgenden Fragestellungen:

- Was bedeutet Skalierung und wie wirkt sie sich auf Anforderungen und Teams aus?
- Wie (re)organisieren wir die Anforderungen und die Teams im Großen?
- Wie werden Releases und Roadmaps definiert und in der langfristigen Planung verwendet?
- Wie werden Anforderungen in skalierten Umgebungen validiert?

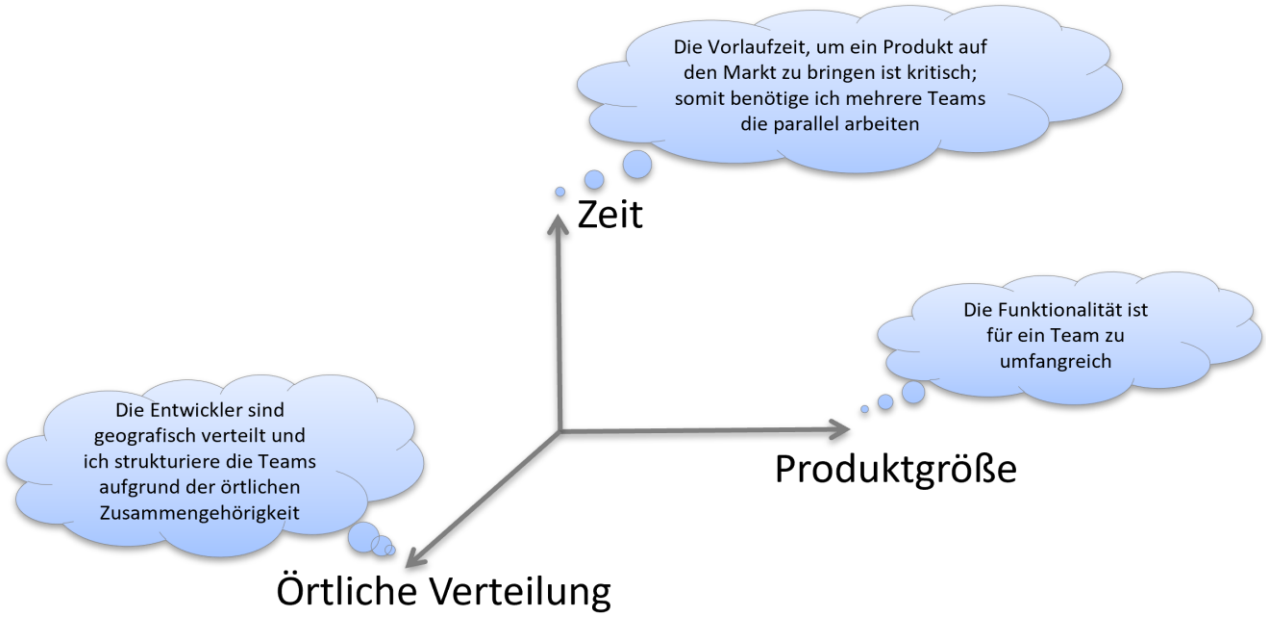

Abbildung 11: Gründe für die Skalierung

6.1 Skalierung von Anforderungen und Teams (K2)

Lernziele

LZ 6.1.1 Gängige Beispiele für Skalierungs-Frameworks kennen (K1)
LZ 6.1.2 Verstehen der Herausforderungen und Mechanismen zur Skalierung von Anforderungen im agilen Umfeld (K2)

Inhalt

Wir verwenden den Begriff „Skalierung", um eine Veränderung der Größe zu beschreiben, entweder des Systems oder des Produkts oder der Anzahl der beteiligten Personen.

Seit etwa 2010 wurde eine Reihe verschiedener agiler Skalierungs-Frameworks entwickelt, um diese Probleme zu lösen. Dazu gehören Nexus [Nexu2021], SAFe [SAFe2021a] [SAFe2021b], LeSS [LeSS], Scrum@Scale [S@SG2021], BOSSA Nova [BOSS2022], Scrum of Scrums [KnIvS2012], und Spotify[Spot2012], aber es gibt noch mehr. Skalierungs-Frameworks variieren hinsichtlich ihres Reifegrades, der Anzahl bewährter Verfahren, Richtlinien und Regeln sowie dem Grad der Anpassbarkeit des Frameworks an bestimmte Bedürfnisse einer Organisation.

Wenn Sie skalieren, werden zwei Dinge immer zutreffen: Sie werden gezwungen sein, die Anforderungen zu hierarchisieren und die Organisation zu hierarchisieren. Grobgranulare Anforderungen werden benötigt, wenn es um das Produkt als Ganzes geht; feingranulare Anforderungen werden in den Teams benötigt, die bestimmte Aspekte des Produkts umsetzen. Und die Teams selbst müssen ihre Zusammenarbeit so organisieren, dass sie innerhalb eines größeren Teams erfolgreich funktionieren.

6.2 Kriterien für die Strukturierung von Anforderungen und Teams im Großen (K2)

Lernziele

LZ 6.2.1 Kennen der Herausforderungen zur Organisation eines Backlogs und zur Kommunikation über Anforderungen in einem Netzwerk von Teams (K1)
LZ 6.2.2 Verstehen der Anforderungsaufteilung in verschiedenen (Projekt-)Umgebungen des agilen Umfelds (K2)

Inhalt

Bei der Produktentwicklung in großem Maßstab müssen meist mehrere Teams gemeinsam am gleichen Produkt arbeiten. In der Praxis entwickelt jedes Team einen spezifischen Teil des Produktes, das mit anderen Teilen in das Gesamtprodukt integriert werden muss, um eine funktionierende Lösung zu schaffen. Nur das integrierte Produkt hat einen Wert für die Stakeholder.

Bei der Skalierung der Produktentwicklung auf mehrere Teams reicht es nicht aus, dass sich alle Product Owner einfach treffen und irgendwie besprechen, welche Teams welchen Teil des Produkts entwickeln sollen und dann auf das Beste hoffen! Ausgefeilte Strukturen und Praktiken werden benötigt, um die Zusammenarbeit in Teams, Anforderungsänderungen und eine schnelle Produktlieferung zu ermöglichen. Andernfalls verschwenden Entwickler

möglicherweise Aufwand bei der Koordination mit anderen Teams, die für ihre Arbeit nicht relevant sind.

Aus der Anforderungsperspektive müssen wir den „Kreislauf" schließen: Von der anfänglichen (Business-)Anforderung der Stakeholder, über die Aufteilung komplexer Anforderungen in kleinere, von jeweils einem Team an Entwicklern handhabbare Teile, bis hin zur Sicherstellung, dass die zusammengesetzten Ergebnisse eine Lösung bilden, die für das Business freigegeben werden kann.

Um gemeinsam an den Anforderungen zu arbeiten und eigenständig vernünftige Entscheidungen zu treffen, brauchen Teams ein allgemeines Verständnis für die Anforderungen der anderen Teams, mit denen sie zusammenarbeiten müssen, ohne jedoch mit allen Details überfordert zu sein. Die Product Owner sollten daher einen angemessenen Detaillierungsgrad finden, der es den Teams ermöglicht, die Auswirkungen ihrer Entscheidungen auf andere Teams zu verstehen.

Um lieferbare Produktinkremente mit minimalen Abhängigkeiten zu anderen Teams zu liefern, sollten Teams im agilen Umfeld an lose gekoppelte End-to-End-Features arbeiten. In unserem Kontext bezieht sich der Begriff End-to-End-Feature auf eine Reihe von zusammenhängenden Funktionen, die eine bestimmte Aufgabe erfüllen und den Stakeholdern einen geschäftlichen Nutzen bieten. Use Cases sind ein Ansatz zur Strukturierung von Anforderungen, der nicht immer mit Agilität in Verbindung gebracht wird, aber dennoch von einer Reihe von Autoren empfohlen wird (z. B. [Jaco2011], Cockburn, [Leffl2010]).

Leider ist es in vielen Fällen nicht so einfach, Anforderungen basierend auf lose gekoppelte Einheiten von einer End-to-End-Funktionalität zu zerlegen. Aufgrund des architektonischen Designs (z. B. Technologie, Infrastruktur, Systemkomponenten, gemeinsame Plattform, Architekturschichten wie Front- und Backend) sowie organisatorischer Überlegungen (fachliche Kompetenzen, Standort des Teams, Sub-Auftragnehmer) können sich Funktionalitäten überschneiden. Das bedeutet, dass verschiedene Teams im agilen Umfeld zusammenarbeiten müssen, um bestimmte Features zu implementieren und, dass ihre jeweiligen Product Owner bei den Anforderungen enger zusammenarbeiten müssen. Alternativ kann ein spezielles Team eingerichtet werden, das sich mit der Überschneidung befasst und mit jedem der ursprünglichen Teams zusammenarbeitet, die sich auf eine Funktionseinheit konzentrieren.

6.3 Roadmaps und umfangreiche Planung (K2)

Lernziele

LZ 6.3.1 Den Unterschied zwischen einer Roadmap und einem Backlog im agilen Umfeld kennen (K1)

LZ 6.3.2 Verstehen der Erstellung und des Managements einer Roadmap im agilen Umfeld (K2)

Inhalt

In der groß angelegten Produktentwicklung verwalten Product Owner Anforderungen im produktfokussierten Backlog. Im Gegensatz zum Backlog wird eine Roadmap zur

inkrementellen Planung der Produktentwicklung verwendet. Eine Roadmap ist eine Vorhersage, wie das Produkt wachsen wird [Pich2016]. Roadmaps verändern nicht den Inhalt der Backlog Items, sondern ordnen sie auf einer Zeitachse an. Sie beantworten die Frage, wann wir welche Features ungefähr erwarten können.

Eine Roadmap ist ein nützliches Mittel, um (strategische) Ziele und Entscheidungen an die Entwickler und andere Stakeholder zu kommunizieren. Sie bricht ein langfristiges Ziel in überschaubare Iterationen herunter, stellt Abhängigkeiten zwischen den Teams dar und gibt den Stakeholdern Orientierung und Transparenz.

Eine Roadmap zeigt strategische Ziele, Meilensteine und grobe Anforderungen. Wichtige Meilensteine können entweder intern oder durch externe Ereignisse, wie z. B. eine Messe oder die Einführung einer neuen Vorschrift, bestimmt sein. Die Darstellung einer Roadmap ist abhängig von ihrem Zweck, ihrer Zielgruppe und ihrem Planungshorizont.

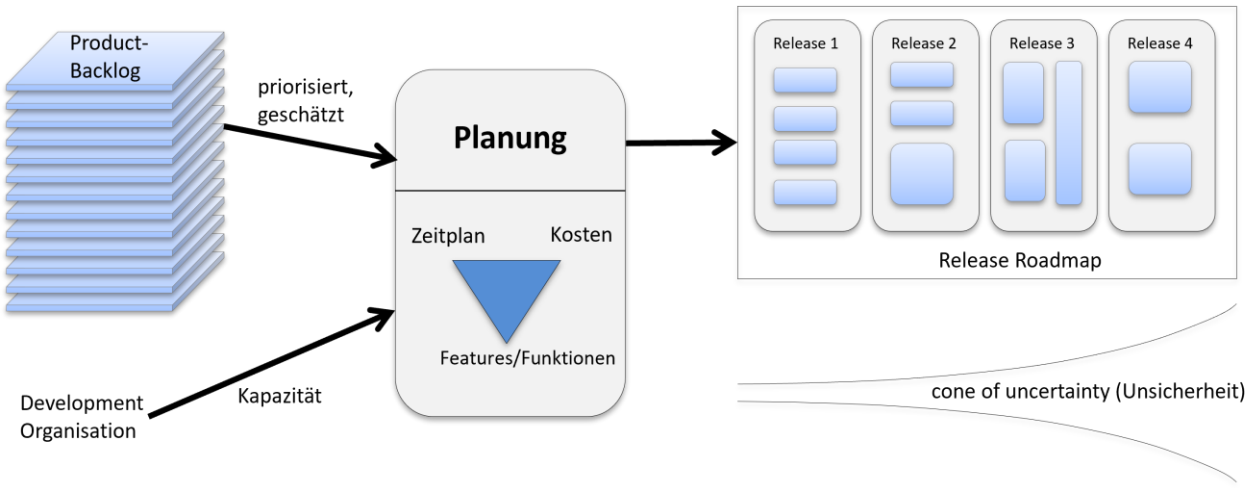

Abbildung 12: Roadmaps und umfangreiche Planung

Um eine langfristige Produkt-Roadmap zu entwickeln, muss ein Product Owner zunächst eine Produktvision und eine Strategie definieren. Danach müssen Product Owner die groben Anforderungen erheben, indem sie sich mit den notwendigen Stakeholdern auseinandersetzen. Es ist nicht nötig, an dieser Stelle Zeit in detaillierte Anforderungen zu investieren. Obwohl die Anforderungen in diesem Stadium mit einem hohen Maß an Unsicherheit behaftet sind, ist die Produkt-Roadmap als grober Iterationsplan für den ersten Schnitt gut genug, um die Planung und Synchronisation zu unterstützen.

6.4 Produkt-Validierung (K2)

Lernziele

LZ 6.4.1 Verstehen konkreter Methoden zur Validierung von Produktanforderungen im agilen Umfeld (K2)

Inhalt

Ein Grundgedanke der agilen Entwicklung ist es, einen kleinen Teil des Produkts zu entwickeln, durch die Einbindung von Stakeholdern Feedback zu generieren und die Produktentwicklung entsprechend den Erkenntnissen und Ergebnissen anzupassen. Nach dem Prinzip des Build-Measure-Learn-Zyklus [Ries2011] ist die Produktvalidierung ein wichtiger Schritt, um schnelles Feedback zu erhalten. Jedes Mal, wenn ein neues Produktinkrement freigegeben wird, verwenden die Product Owner dieses Produktinkrement, um seinen Geschäftswert zu prüfen und um festzustellen, ob die Produktanforderungen richtig verstanden wurden.

Die Validierung auf Produktebene (in Scrum z. B. Sprint-Review) ist eine wichtige Methode in der groß angelegten Produktentwicklung, da sie sicherstellt, dass die Product Owner gemeinsam die volle Verantwortung von den Geschäftsanforderungen bis zur Produktintegration tragen. Es ist das gesamte Produkt, das für die Stakeholder einen Wert hat, nicht nur kleine Teile des Produkts.

Ein weiterer Ansatz für die Produktvalidierung in der umfangreichen Produktentwicklung basiert auf Datenanalyse [MaeA2016]. Das integrierte Produktinkrement wird an Benutzer ausgeliefert und anhand deren Verhalten wird gemessen, ob die Produkt-Features einen positiven, neutralen oder negativen Einfluss haben.

7 Begriffsdefinitionen, Glossar

Für die Begriffsdefinitionen verweisen wir auf das IREB CPRE Requirements Engineering Glossar [Glin2024], das nicht nur ein umfassendes Glossar der Requirements Engineering Terminologie ist, sondern auch viele Begriffe aus dem Bereich der Agilität definiert. Für spezifische Agilitätsbegriffe kann der Leser den aktuellen Scrum Guide [ScSu2020] konsultieren.

8 Literaturverzeichnis

[Alex2005] Alexander, I. F.: A Taxonomy of Stakeholders – Human Roles in System Development. International Journal of Technology and Human Interaction, Vol 1, 1, 2005, pages 23–59.

[BOSS2022] https://www.agilebossanova.com/#bossanova. Zuletzt besucht im März 2025.

[CleA2001] P. Clements et al.: Evaluating Software Architectures, SEI Series in Software Engineering, 2001

[Cohn2010] Cohn, M.: User Storys für die agile Software-Entwicklung mit Scrum, XP u.a., mitp, 2010

[Cohn2006] Cohn, M.: Agile Estimation and Planning, Addison Wesley, 2006

[Dora1981]Doran, G. T: There's a S.M.A.R.T. way to write management's goals and objectives, Management Review. AMA FORUM. 70 (11): 35–36 1981.

[Glin2024] Glinz, M.: A Glossary of Requirements Engineering Terminology. Standard Glossary for the Certified Professional for Requirements Engineering (CPRE) Studies and Exam, Version 2.1.1, 2024. https://cpre.ireb.org/de/knowledge-and-resources/downloads#cpre-glossary. Zuletzt besucht im März 2025.

[HeHe2011] Heath, C., Heath, D.: Switch: Veränderungen wagen und dadurch gewinnen. Crown Business, 2010

[High2001] Highsmith, J.: Design the Box. Agile Project Management E-Mail Advisor 2001, http://www.joelonsoftware.com/articles/JimHighsmithonProductVisi.html. Zuletzt besucht im März 2025.

[Hrus2017] https://b-agile.de/downloads/articles/story_splitting.pdf. Zuletzt besucht im März 2025.

[ISO25010] ISO/IEC 25010:2023: Systems and software engineering – Systems and Software Quality Requirements and Evaluation (SQuaRE) – System and software quality models: https://www.iso.org/standard/78176.html. Zuletzt besucht im März 2025.

[Jaco2011] https://www.ivarjacobson.com/publications/white-papers/use-case-ebook. Zuletzt besucht im März 2025.

[Leffl2010] Leffingwell, D.: Agile Software Requirements – Lean Requirements Practices for Teams, Programs, and the Enterprise. Addison Wesley, 2010.

[LeSS] Large-Scale Scrum: https://less.works. Zuletzt besucht im März 2025.

[MaeA2016] Maalej, W., Nayebi, M., Johann T., Ruhe, G.: Toward Data-Driven Requirements Engineering. IEEE Software (Volume 33, Issue 1), 2016.

Meye2014] Meyer, B.: Agile! The Good, the Hype and the Ugly, Springer, 2014.

[Nexu2021] https://www.Scrum.org/resources/nexus-guide. Zuletzt besucht im März 2025.

[Pich2016] Pichler, R.: Strategize – Product Strategy and Product Roadmap Practices for the Digital Age, Pichler Consulting 2016.

[PoRu2021] Pohl, K., Rupp, C.: Basiswissen Requirements Engineering: Aus- und Weiterbildung nach IREB-Standard zum Certified Professional for Requirements Engineering Foundation Level, dpunkt.verlag, 5. Auflage, 2021.

[Prim2017] CPRE RE@Agile Primer https://cpre.ireb.org/de/downloads-and-resources/downloads#cpre-agile-primer-syllabus-and-study-guide. Zuletzt besucht im März 2025.

[Rein2009] Reinertsen, D.G.: The Principles of Product Development Flow – Second Generation Lean Product Development. Celeritas Publishing, 2009.

[Ries2011] Ries, E.: The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses, Crown Business, New York, NY 2011.

[RoRo2013] Robertson S. Robertson J.: Mastering the Requirements Process – Getting Requirements Right, 3rd edition, Addison Wesley, 2013.

[Robe2003] Robertson, S.: Stakeholders, Goals, Scope: The Foundation for Requirements and Business Models, 2003, https://www.volere.org/wp-content/uploads/2018/12/StkGoalsScope.pdf. Zuletzt besucht im März 2025.

[SAFe2021a] https://www.scaledagileframework.com/roadmap/. Zuletzt besucht im März 2025.

[SAFe2021b] https://www.scaledagileframework.com/pi-planning/. Zuletzt besucht im März 2025.

[SAFe2021] https://www.scaledagileframework.com/safe-requirements-model/. Zuletzt besucht im März 2025.

[S@SG2021] Sutherland, J. and Scrum, Inc: Scrum@Scale Guide: https://www.Scrumatscale.com/Scrum-at-scale-guide/. Zuletzt besucht im März 2025.

[KnIvS2012] Kniberg, H.; Ivarsson, A.: Scaling Agile @ Spotify with Tribes, Sqads, Chapters & Guilds. https://blog.crisp.se/wp-content/uploads/2012/11/SpotifyScaling.pdf. Zuletzt besucht im März 2025.

[Spot2012] https://blog.crisp.se/wp-content/uploads/2012/11/SpotifyScaling.pdf. Zuletzt besucht im März 2025.

[ScSu2020] Schwaber, K., Sutherland, J.: The Scrum Guide. 2020 https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf. Zuletzt besucht im März 2025.

[Wake2003] Wake, B: INVEST in Good Stories, and SMART Tasks, 2003, https://xp123.com/articles/invest-in-good-stories-and-smart-tasks/. Zuletzt besucht im März 2025.