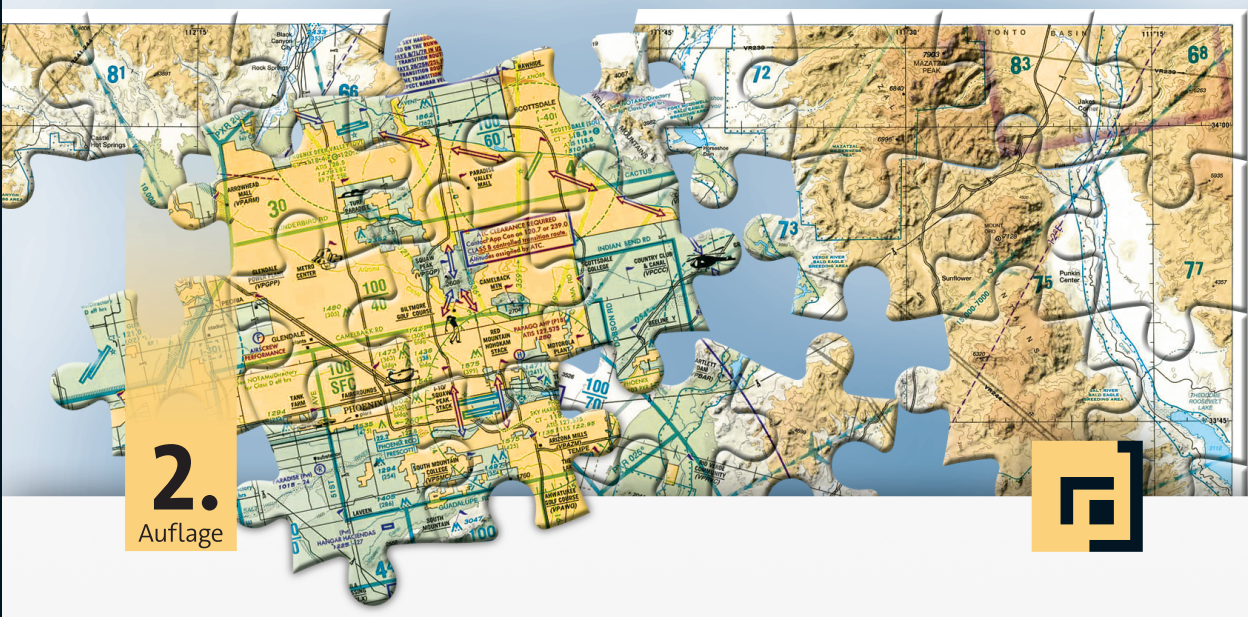


2. Auflage

Johannes Bergsmann

Requirements Engineering

für die agile Softwareentwicklung

Methoden, Techniken und Strategien

→ Unter Mitwirkung von Markus Unterauer

Optimal für den IREB® RE@Agile Primer

Openbook zu »Requirements Engineering für die agile Softwareentwicklung«

(ISBN 978-3-86490-485-1)

dpunkt.verlag

Johannes Bergsmann hat technische Informatik studiert und arbeitete ca. 11 Jahre als Softwareentwickler, Projektleiter, Technischer Leiter, Architekt, Produktmanager und Berater in einem internationalen Systemhaus und als selbstständiger Unternehmer. Im März 2003 gründete er »Software Quality Lab« und begleitet seither als Berater und Trainer viele Unternehmen im Bereich Requirements Engineering und Prozessgestaltung.

Johannes Bergsmann ist zertifizierter Scrum Master, Sachverständiger für Informatik bei Gerichten, als Lehrbeauftragter an Fachhochschulen im Bereich Softwarequalitätsmanagement tätig, ist Autor vieler Fachartikel und hält Fachvorträge bei verschiedenen Veranstaltungen und Konferenzen.

Johannes Bergsmann

Requirements Engineering für die agile Softwareentwicklung

Methoden, Techniken und Strategien

Ein Openbook zur 2., überarbeiteten und aktualisierten Auflage

dpunkt.verlag

Johannes Bergsmann
johannes.bergsmann@software-quality-lab.com

Lektorat: Christa Preisendanz
Copy-Editing: Ursula Zimpfer, Herrenberg
Satz: Birgit Bäuerlein
Herstellung: Stefanie Weidner
Umschlaggestaltung: Helmut Kraus, www.exclam.de

Dieses Openbook basiert auf dem Buch »Requirements Engineering für die agile Softwareentwicklung«, dpunkt.verlag, 2. Auflage 2018 (ISBN 978-3-86490-485-1).

Wieblinger Weg 17
69123 Heidelberg

Schreiben Sie uns:
Falls Sie Anregungen, Wünsche und Kommentare haben, lassen Sie es uns wissen: *hallo@dpunkt.de*.

Vorwort zum Openbook

Als Berater habe ich in meiner bisherigen beruflichen Tätigkeit mehr als 220 verschiedene Projekte bei Kunden begleitet oder auch selbst in unterschiedlichen Rollen (Entwickler, Tester, Projektleiter, Architekt, Produktmanager, Analytiker, Coach, Berater etc.) mitgearbeitet. Seit mehr als 15 Jahren waren viele dieser Projekte agile Projekte oder solche, in denen die Mitwirkenden zumindest versuchten, die agilen Prinzipien und Praktiken umzusetzen. Am erfolgreichsten und effizientesten waren fast immer diejenigen Projekte, bei denen agile Vorgehensweisen mit Elementen aus dem klassischen Requirements Engineering und Projektmanagement ergänzt und so die Stärken beider Bereiche genutzt wurden.

Requirements Engineering ist ein Aspekt, der in vielen agilen Vorgehensweisen nur sehr grob beschrieben wird. In der Begeisterung der ersten Stunde möchten viele Teams möglichst rasch starten und erste Erfolge zeigen und beginnen mit einem sehr intuitiven Ansatz, wie z.B. einer einfachen Liste von User Stories. Es gibt jedoch noch viele andere Techniken aus dem Requirements Engineering, die in agilen Vorgehensweisen angewendet werden können und sollen. In Kombination mit einigen passenden, langjährig bewährten Techniken des Requirements Engineering bieten sie gute Lösungen für die Requirements-Herausforderungen in agilen Projekten.

In diesem Openbook gebe ich Einblick in folgende Themen:

- **Qualität von Requirements**
  Definition of Ready und die Definition of Done
- **Requirements Management**
  Agile Requirements-Board
- **Rechtliche Themen**
  Das Vier-Stufen-Modell für agile Festpreisprojekte

Dieses Openbook gibt lediglich einen Überblick über diese Themen. Detailliertere Informationen dazu und weitere wichtige Konzepte und Anwendungen finden Sie in meinem Buch »**Requirements Engineering für die agile Softwareentwicklung**«.

Nun wünsche ich Ihnen viel Spaß bei der Lektüre!

Johannes Bergsmann
Linz, im Oktober 2019

Inhaltsverzeichnis

1 Qualität von Requirements

Anforderungen sind die Basis für die Planung (z. B. Sprint, Release), die Architektur, die Umsetzung und die Qualitätssicherung in agilen Projekten. Sie sind das Fundament, auf dem alles andere aufbaut. Daher müssen die Anforderungen eine dem Verwendungszweck angemessene Qualität aufweisen.

Jede Anforderung sollte vor Beginn des Sprints, in dem die geplante Umsetzung durchgeführt werden soll, anhand entsprechend definierter Qualitätskriterien geprüft und finalisiert werden, damit sie in das Backlog des folgenden Umsetzungs-Sprints aufgenommen werden kann (vgl. [IREB RE@Agile FL], LE 3.1.10).

Akzeptanz- und Abnahmekriterien ergänzen die Geschäftsanforderungen und sollten ebenfalls schon vor Beginn eines Sprints definiert werden.

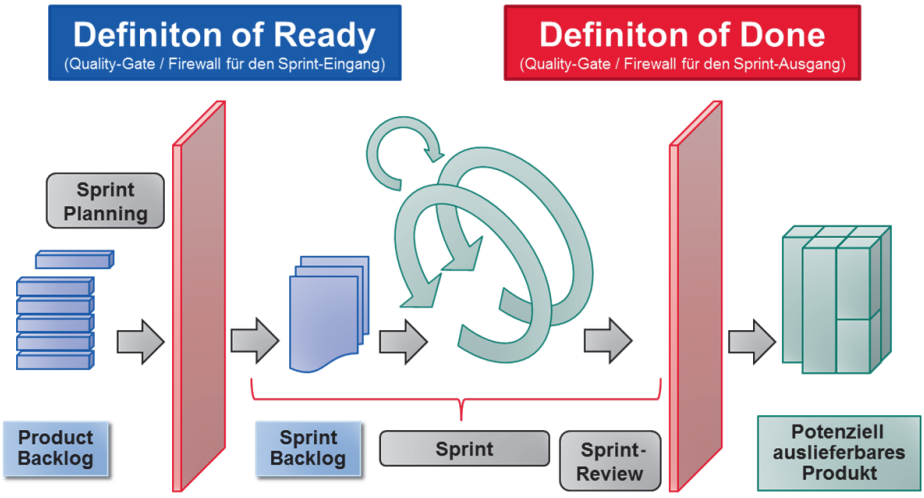


Abb. 1–1 *Definition of Ready (DoR) & Definition of Done (DoD)*

Die »**Definition of Ready**« (**DoR**) und »**Definition of Done**« (**DoD**) unterstützen den formalen Prozess der Entwicklung und sichern die Qualität der Anforderungen (DoR) und der Produktentwicklung am Ende eines Inkrements (DoD).

Die **DoD** ist ein offizielles Artefakt des Scrum Guide [Schwaber & Sutherland 2017] und stellt ein Quality-Gate für den »Ausgang« des agilen Entwicklungsprozesses dar. Nur Ergebnisse, die den definierten Qualitätskriterien entsprechen, werden zum Kunden gegeben.

Die **DoR** hat sich als »Good Practice« etabliert und hilft, gute Anforderungen zu erstellen und eine Überlastung des Teams durch schlechte Anforderungen zu vermeiden. Die DoR ist das Quality-Gate für den »Eingang« in den agilen Entwicklungsprozess. Nur Anforderungen, die die definierten Qualitätskriterien der DoR erfüllen, werden in das Sprint Backlog aufgenommen. Die DoR hilft dabei, die Anforderungen passend zu detaillieren und genügend Informationen für die Diskussionen zwischen dem Product Owner/Requirements Engineer und dem Entwicklungsteam zu liefern.

1.1 Qualitätskriterien für Requirements

Bevor die Anforderungen vom Team sinnvoll geschätzt und geplant werden können, muss sichergestellt sein, dass sie möglichst verständlich, vollständig und detailliert genug dargestellt sind. Dies ist Aufgabe des Product Owners. Neben der Verständlichkeit, Vollständigkeit und Detailliertheit gibt es im Requirements Engineering eine Reihe von weiteren Qualitätskriterien, die erfüllt sein müssen, sowie Techniken zu deren Überprüfung. Diese können helfen, Missverständnisse zwischen Team und Product Owner von vornherein zu vermeiden und die Effizienz weiter zu steigern.

Werden Qualitätskriterien für Requirements definiert, so sollten diese auch in die »Definition of Ready« (DoR) aufgenommen werden.

1.2 Definition of Ready (DoR)

Eine Voraussetzung für die Erstellung eines Produktinkrements ist, dass die dazu eingeplanten Anforderungen bereit für die Umsetzung sind.

Welche Qualitätskriterien eine Anforderung tatsächlich erfüllen muss und welche Analysen durchgeführt sein müssen, damit sie als »bereit für die Umsetzung« bezeichnet werden darf, wird in einer »Definition of Ready« (DoR) festgelegt. Analog zur »Definition of Done« (DoD) werden darin diejenigen Kriterien explizit sichtbar gemacht, die eine Anforderung erfüllen muss, bevor sie für die Umsetzung in der nächsten Iteration akzeptiert wird. Die Definition of Ready wird, wie die DoD, zwischen Product Owner und Team vereinbart.

Die DoR ist eine Liste mit Kriterien für die Aufnahme eines Product-Backlog-Elements in das Sprint Backlog. Sie dient der Freigabe der Spezifikation für die Umsetzung.

Wie bei der DoD kann es auch bei der DoR sein, dass einzelne Kriterien nicht für jede Anforderung angewendet werden müssen. So ist es möglicherweise sinnvoll, für unterschiedliche Risikoklassen von Anforderungen auch unterschiedliche DoR-Kriterien anzuwenden.

Tabelle 1–1 zeigt ein Beispiel für eine Definition of Ready.

Prüfpunkt	Risiko
Ist die Anforderung vollständig beschrieben? Sind UI-Masken ausreichend beschrieben? ■ Grafischer Maskenentwurf/Skizze ■ Mögliche Aktionen ■ Angezeigte Daten ■ Regeln ■ Berechtigungen zur Anzeige/Bearbeitung Sind die externen Systemschnittstellen ausreichend beschrieben? ■ Datenobjekte und Felder inkl. möglicher Feldwerte ■ Austauschformat ■ Austauschprozess ■ Erwartete Datenmengen ■ Fehlerbehandlung Sind Berichte ausreichend beschrieben? ■ Grafischer Entwurf/Skizze ■ Angezeigte Daten ■ Berechtigungen zur Anzeige	**H, M** **H, M, L** **H**
Sind nicht funktionale Anforderungen definiert und geprüft?	**H, M**
Ist die Story verständlich und eindeutig formuliert? Werden keine Universalquantoren, Konjunktive, Passivkonstruktionen und Relativierungen verwendet? Wird die Satzschablone für User Stories eingehalten (»Als ROLLE möchte ich AKTION, um NUTZEN«)?	**H, M, L**
Ist die Story bewertet? ■ Herrscht Übereinstimmung bei allen Stakeholdern, dass die Story wichtig für das Produkt ist? ■ Sind Priorität, Business Value und Risiko bestimmt?	**H, M, L**
Sind die relevanten Akzeptanzkriterien definiert? Manche Akzeptanzkriterien ergeben sich erst im Zuge der Umsetzung, die wesentlichen müssen aber schon vorher definiert sein.	**H, M, L**
Sind die Risiken für die Story identifiziert und bewertet? ■ Sind die Risiken in einer Risikoliste dokumentiert? ■ Sind für Risiken mit Wert > 15 Maßnahmen definiert? Wurden Risiken mit Wert > 20 an die Geschäftsführung gemeldet?	**H, M, L**
Wird die Story vom Tester als »testbar« eingestuft?	**H, M, L**
Sind Testfälle erstellt? Basierend auf den Akzeptanzkriterien muss ein Testplan für die Prüfung der Story erstellt werden.	**H, M**
Wurde überlegt, wer die Story abnimmt und wie das Team sie demonstrieren kann? Oft wird eine Story nicht vom Product Owner, sondern vom Kunden abgenommen. In diesem Fall muss dies vorher geplant werden.	**H, M, L**
Ist das Entwicklungssystem bereit für die Story? Kann aus personeller, technischer und Infrastruktursicht mit der Umsetzung der Story begonnen werden?	**H, M, L**
Erfüllt die Story die INVEST-Kriterien?	**H, M, L**

→

Prüfpunkt	Risiko
Sind alle Abhängigkeiten[a] identifiziert? Abhängigkeiten können die Umsetzung einer Story verzögern und die Prüfung erschweren.	**H**
…	

a. Dies widerspricht eigentlich dem INVEST-Kriterium »independent«. In der Praxis kann es jedoch vorkommen, dass Stories nicht ganz unabhängig sind. In diesem Fall sollten dann die Abhängigkeiten angegeben sein.

Tab. 1–1 *Beispiel für eine DoR-Checkliste (H = High, M = Middle, L = Low)*

Die DoR kann über den Projektverlauf angepasst und sogar kleiner werden, wenn sich das Team, der Product Owner und die Stakeholder »eingeschwungen« haben und die Zusammenarbeit so gut funktioniert, dass auch ohne laufendes Durchgehen einer Checkliste die Qualitätskriterien selbstständig eingehalten werden.

Oft genügt es, wenn die DoR als Checkliste beim Taskboard ausliegt. Die Stories werden dann in der Iterationsplanung gemeinsam bewertet und bei Erfüllung der Kriterien in das Sprint Backlog aufgenommen. Die DoR-Bewertungen werden in diesem Fall typischerweise nicht niedergeschrieben. Durch das Aufnehmen der Story in das Sprint Backlog ist implizit dokumentiert, dass sich das ganze Team einig war, dass die Kriterien auch erfüllt sind.

In manchen Fällen, wie z. B. bei sicherheitskritischer Software oder anderen haftungsrelevanten Umgebungen, kann es notwendig sein, die DoR-Durchführung nachvollziehbar zu dokumentieren. Um die DoR hier effizient und nachvollziehbar durchführen zu können, ist es sinnvoll, dies toolunterstützt zu machen, z. B. in einem entsprechenden Requirements-Management-Werkzeug oder einem elektronischen Taskboard.

Aufpassen muss man bei der DoR, dass diese nicht als Ausrede verwendet wird, um Diskussionen über User Stories vom Entwicklerteam fernzuhalten, was dann genau den agilen Grundsätzen widersprechen würde.

1.3 Definition of Done (DoD)

Immer wieder kommt es in Softwareprojekten zu Fehlern und Unmut, weil nicht klar geregelt ist, wann etwas wirklich »fertig« ist. Für einen Entwickler ist eine Anforderung vielleicht schon fertig, wenn er den Code geschrieben und erfolgreich kompiliert hat. Sieht sich dann der Product Owner die umgesetzte Funktion am Testsystem an, stellt er oft fest, dass noch viele Fehler vorhanden sind und Benutzbarkeit kaum gegeben ist. Für den Product Owner ist die Funktion erst fertig, wenn sie fehlerfrei einsetzbar und für den Anwender gut bedienbar ist. Aus solchen unterschiedlichen Sichtweisen resultieren in vielen Projekten Probleme und Unstimmigkeiten im Team. Ursache ist dabei eine fehlende explizite, für alle verbindliche und klare Definition, wann etwas »fertig« ist.

Die einfachste Lösung für dieses Problem ist eine explizite Definition, wann ein Backlog Item auf »fertig« gesetzt werden darf. Ken Schwaber und Jeff Sutherland schlagen dazu eine »Definition of Done« (DoD) vor, damit alle »ein gemeinsames Verständnis haben, was es bedeutet, wenn eine Arbeit abgeschlossen ist« [Schwaber & Sutherland 2017].

Die »Definition of Done« ist eine Checkliste mit zwei Sichten:

- Die Produktsicht umfasst alle Eigenschaften, die ein umgesetztes Backlog Item erfüllen muss.
- Die Prozesssicht definiert alle Tätigkeiten, die dabei erledigt sein müssen.

Bei der Planung der Iterationen und der Schätzung der Backlog Items muss das Team nicht nur die reine Programmierung berücksichtigen, sondern die gesamte Abarbeitung der DoD.

Eine DoD ist nicht nur zur Bewertung der Fertigstellung eines einzelnen Backlog-Eintrags sinnvoll, sondern kann auf mehreren Ebenen angewendet werden, wie beispielsweise:

- **DoD für eine Anforderung** (Story oder anderes Backlog Item)
- **DoD für einen Sprint**
  Diese ist anforderungsübergreifend und bezieht sich auf das gesamte Sprint Backlog und alle sonst relevanten Artefakte und Tätigkeiten. Die Bewertung erfolgt am Ende eines Sprints.
- **DoD für ein Release** (Auslieferung an den Kunden)
  Diese ist anforderungsübergreifend über mehrere Iterationen für eine tatsächlich ausgelieferte Version des Systems. Die Bewertung erfolgt nach der tatsächlichen Auslieferung an den Kunden.

Je nach Umfeld, Anwendungsbereich oder Zielgruppe können weitere DoD-Ebenen oder zusätzliche Granularität sinnvoll sein:

- **DoD für die Codierung**
  z.B. Kriterien, die definieren, wann ein Programmierer mit dem Code fertig ist und in die Quellcodeverwaltung einchecken darf, oder Forderungen bezüglich Unit-Test-Abdeckung, Codedokumentation, Codereview.
- **DoD für Design/Architektur**
  z.B. eine Checkliste für Systemdesign, Architekturkriterien, Testbarkeit, Wartbarkeit.
- **DoD für die Testdurchführung**
  z.B. Anforderungen an die Tests oder Testendekriterien, die definieren, wann man zu testen aufhört.
- **DoD für den Rollout**
  z.B. eine Checkliste für Verteilung, Inbetriebnahme, Konfiguration.

Der Vorteil bei der Aufteilung der DoD auf mehrere Checklisten besteht darin, dass bei der Durchführung nicht mehr eine sehr lange Checkliste herangezogen wird, unter der dann alle im Team stöhnen und die dann evtl. aufgrund der Länge nur oberflächlich oder lückenhaft durchgegangen wird. Der Aufwand wird vielmehr auf mehrere kleine Schritte aufgeteilt.

Bei der Aufnahme von Kriterien in die DoD sollte darauf geachtet werden, dass diese »SMART« sind:

- **Spezifisch**
  Die DoD-Kriterien sollen eindeutig und so präzise wie möglich definiert sein und nicht vage.
- **Messbar**
  Die Kriterien müssen messbar bzw. testbar sein.
- **Akzeptiert**
  Die DoD-Kriterien müssen angemessen sein und von den Betroffenen akzeptiert werden.
- **Realistisch**
  Die Einhaltung bzw. Erreichung muss möglich sein.
- **Terminierbar**
  Es muss klar sein, wie getestet und abgenommen wird.

Es kann auch sein, dass einzelne DoD-Elemente nicht für jede Anforderung angewendet werden. So ist es möglicherweise sinnvoll, je nach Risiko einer Anforderung unterschiedliche DoD-Kriterien zu haben.

Damit man in die DoD nicht zu wenig, aber auch nicht zu viele oder vielleicht sogar unsinnige Kriterien aufnimmt, sollte man sich vorab bei der Erstellung der DoD schon intensiv Gedanken machen. Unterstützt werden kann dies durch ein sogenanntes »Done Thinking Grid«[1]. Dies ist eine Technik, mit der oft auf einem Board Ideen zu sinnvollen Done-Kriterien gesammelt werden (siehe Abb. 1–2). Zur Strukturierung können evtl. schon Themenbereiche und Ebenen vorgegeben werden (z. B. als »Swimlanes«):

- Requirements/Tasks
- Architektur/Design/Coding
- Testing
- Sprint/Release/Rollout-Prozess/Tools

1. Quelle: *http://robinsonraju.info/agile/2014/10/07/agile-task-list-what-does-done-mean/*.

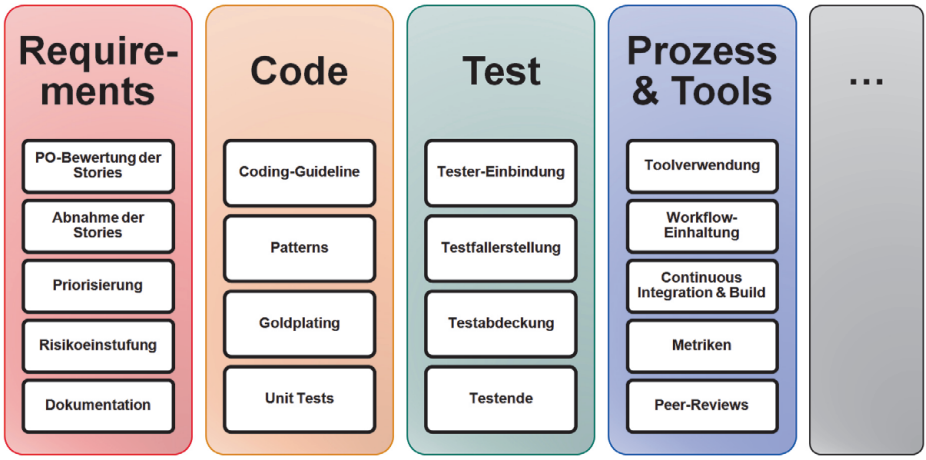


Abb. 1–2 *Beispiel für ein »Done Thinking Grid«*

Die in der Literatur und im Internet oft unklare Begriffsdefinition der DoD kann dazu verleiten, DoDs zu oberflächlich oder ungeeignet zu erstellen. Es werden oft zu einfache Beispiele für DoDs genannt (siehe Abb. 1–3).

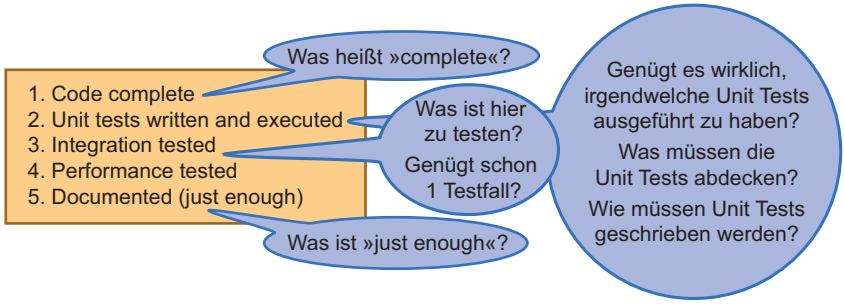


Abb. 1–3 *Schlechtes Beispiel für eine DoD*

Diese oberflächlichen Beispiele führen in der Praxis zu dem Problem, dass verschiedene Entwickler bzw. Teams und der Kunde dies unterschiedlich interpretieren. Das Team meint evtl., dass es fertig ist – tatsächlich sind dann aber noch viele Fragen offen, wenn der Kunde zur Abnahme hinzugezogen wird.

DoDs sollten nur Qualitätskriterien an die Prozessdurchführung und allgemeine Qualitätskriterien für die Produkterstellung (z. B. wichtige Qualitätsanforderungen, die für jedes Requirement zu berücksichtigen sind) enthalten. Sie sollen keine Testfallbeschreibungen oder inhaltlichen Abnahmekriterien beinhalten oder diese ersetzen. Ebenso sollten Kriterien, die spezifisch zu einer Anforderung gehören, in deren Akzeptanzkriterien aufgenommen werden und nur übergreifende Kriterien in die DoD. Tests und inhaltliche Anforderungen jeglicher Art sollten also in den jeweiligen Anforderungs- und Testartefakten definiert werden.

Für eine einfache und klare Anwendung der Definition of Done sollte vermieden werden, Testfälle und DoD-Kriterien zu vermischen! Die DoD soll ausschließlich qualitative Kriterien und keine inhaltlichen Anforderungskriterien an das zu erstellende System oder einzelne Artefakte enthalten!

Folgende Tipps helfen auch dabei, DoDs passend zu erstellen:

- Ein einziger DoD-Eintrag ist in den meisten Fällen zu wenig!
- Jedes Artefakt durchläuft verschiedene Phasen/Ebenen (z.B. einzelne Story ist fertig, der Sprint ist insgesamt fertig oder das Release ist insgesamt fertig). Es kann daher eine DoD-Checkliste für jede dieser Phasen/Ebenen geben.
- »Think twice«: Nach dem ersten DoD-Entwurf sollte man nach einer gewissen Zeit selbst noch einmal darüber nachdenken und die DoD von anderen Personen reviewen lassen.
- Keine universelle Themenbreite für die Kriterien in der DoD zulassen: Das Management könnte z.B. Zeit- und Kostenkriterien hineinreklamieren wie »Das Feature muss in 5 Tagen programmiert sein« oder »Das Release darf max. 18.000,- Euro kosten«. Der Product Owner könnte funktionale Kriterien mit aufnehmen wollen wie z.B. »Feature A muss in folgender Form realisiert sein: …«. Diese Kriterien sollten nicht in die DoD, sondern in Planungs- oder Spezifikationsdokumenten definiert werden.

Die Tabellen 1–2 bis 1–4 zeigen beispielhaft einige Kriterien als Anregungen für Inhalte von DoDs unterschiedlicher Phasen/Ebenen.

DoD-Kriterien für Story-Fertigstellung
■ Ist die Anforderung (aus Sicht des Product Owners) vollständig implementiert? ■ Sind die Bedürfnisse der Anwender abgedeckt? ■ Ist die Benutzbarkeit gegeben und als brauchbar bewertet?
Manuelle Abnahmetestfälle wurden mindestens einmal ausgeführt mit ■ Positivtests des gewünschten Verhaltens und ■ Negativtests des Verhaltens im Fehlerfall und bei Falscheingaben.
■ Sind Unit Tests gemäß Testkonzept erstellt und erfolgreich durchlaufen?
■ Wurden automatisierte Schnittstellen und GUI-Tests gemäß Testkonzept erstellt und erfolgreich durchlaufen?
■ Wurden Tests mit produktivnahen Testdaten durchgeführt?
…

Tab. 1–2 *Beispiel-Checkliste (Auszug) für eine DoD zur Story-Fertigstellung*

DoD-Kriterien für Sprint-Ende
☐ Alle für den Sprint festgelegten User Stories sind im Status »Done«.
☐ Alle für den Sprint festgelegten User Stories durch den Kunden wurden einem Abnahmetest unterzogen.
☐ Integrations- und Systemtests wurden gemäß DoD »Integrations- und Systemtestkriterien« erfolgreich durchgeführt.
☐ In den Tests wurden keine Fehler der Klasse 1 und 2 gefunden und max. 5 Fehler der Klasse 3.
☐ Alle Tests für User Stories der Kritikalität 1 wurden automatisiert.
☐ Gesamter Quellcode ist eingecheckt.
☐ Erfolgreicher Build des Gesamtsystems wurde durchgeführt. ☐ Die neuen Teile arbeiten fehlerfrei mit bestehenden Modulen zusammen.
☐ Codereview für die neuen Teile wurde positiv durchgeführt.
Die notwendigen Dokumentationen wurden erstellt bzw. angepasst: ☐ Benutzerhandbuch ☐ Konfigurations- und Parameterdokumentation ☐ Installations- und Administrationshandbuch ☐ Architekturdokumentation ☐ Quellcodedokumentation ☐ What's-new-Liste ☐ Behobene Fehlerliste ☐ Sonstige relevante Dokumentation
…

Tab. 1–3 *Beispiel-Checkliste (Auszug) für eine DoD zum Sprint-Ende*

DoD-Kriterien für Codereview
☐ Coding-Guidelines sind komplett eingehalten.
☐ Es wurden keine Anti-Patterns verwendet.
☐ Fehlerbehandlung wurde gemäß Richtlinie umgesetzt.
☐ Die angestrebte Kapselung bzw. Abhängigkeiten wurden erreicht.
☐ Code und Architektur wurden als »testbar« durch den Tester eingestuft.
☐ Es sind keine nicht geforderten Funktionen umgesetzt (»Goldplating«).
☐ Unit Tests und Testabdeckung gemäß Vorgaben wurden eingehalten.
☐ Inline-Codedokumentation für nicht selbsterklärenden Code wurde erstellt.
…

Tab. 1–4 *Beispiel-Checkliste (Auszug) für eine DoD zur Codereview-Durchführung*

DoDs sollten so in den Prozess integriert sein, dass sie eingehalten werden müssen, z. B. durch …

- Vier-Augen-Prinzip in verschiedenen Prozessphasen,
- Pair Programming (einer programmiert, der andere prüft),
- unternehmensweiten Prozessverantwortlichen, der die übergreifende DoD-Einhaltung prüft,
- projektbegleitenden Quality-Manager, der die DoD-Einhaltung in einem spezifischen Projekt prüft,
- Einbau von DoD-Checklisten in Tools,
- Verwendung eines Workflow-Tools, wodurch DoDs direkt in die Prozesssteuerung integriert werden.

Diese Prüfung und (Selbst-)Kontrolle bei der DoD-Umsetzung ist wichtig, da die Überprüfung viel Aufwand bedeuten kann und damit zu rechnen ist, dass die Durchführungsdisziplin bei fehlender Prozessabsicherung und Prüfung nachlässt. Damit DoDs auch gelebt werden, ist es wichtig, diese *vor* Projektbeginn unter Einbeziehung aller relevanten Beteiligten zu definieren. Im Laufe des Projektes hinterfragt das Team die DoDs in den Retrospektiven immer wieder kritisch und passt sie bei Bedarf weiter an. Der Agile Master sollte in jedem Fall ein Auge auf die Einhaltung und laufende Verbesserung der DoD haben.

Ziel ist es, kontinuierlich besser zu werden und die Qualitätsansprüche in den DoDs laufend anzuheben!

2 Requirements Management

Requirements Engineering fasst unter Anforderungsmanagement alle Aktivitäten zusammen, um Anforderungen über die Zeit zu handhaben. Dazu gehören Aktivitäten wie Versionsverwaltung, Änderungsmanagement, Konfigurationsmanagement, Rückverfolgbarkeit und das Hinzufügen von Attributen wie Status, Schätzungen, Prioritäten, Links zu widersprüchlichen Anforderungen und möglicherweise Aufzeichnungen über alle relevanten Personen, die an der Erfassung, Prüfung, Unterzeichnung, Implementierung oder dem Test dieser Anforderung beteiligt sind (vgl. [IREB RE@Agile FL], LE 3.2.4).

Viele dieser Aktivitäten werden im agilen Umfeld auf ein oder mehrere Backlogs reduziert. Hier wird oft nur die neueste und beste Version einer Anforderung im Backlog behalten. Der Backlog-Eintrag wird ggf. entfernt, sobald das Produkt, das diese Anforderungen erfüllt, fertiggestellt und ausgeliefert wird.

Agile Methoden umfassen in der Regel zwei wichtige Aktivitäten als Basis für das Anforderungsmanagement:

- **Priorisieren von Anforderungen**
  Basierend auf dem Geschäftswert oder anderen Faktoren wie z. B. dem Risiko wird entschieden, wann die Anforderung implementiert werden soll. Je höher der Wert (oder der Risikominderungsfaktor) für den Kunden ist, desto wichtiger wird die Anforderung eingestuft, da agile Projekte versuchen, höchsten Geschäftswert zuerst zu liefern.
- **Schätzen der Anforderungen**
  Dies ermittelt, wie viel Aufwand notwendig ist, um die Anforderung umzusetzen. Zu hohe bzw. zu weit auseinander liegende Schätzungen sind ein klares Signal an den Product Owner, dass noch mehr getan werden muss, um diese Anforderung »umsetzungsbereit« zu machen und durch die Definition of Ready (DoR) zu bringen.

Wenn hier nur zwei Aktivitäten angegeben sind, so bedeutet das nicht, dass nicht andere Aspekte auch für die agile Entwicklung sinnvoll und wichtig sind. Zusätzliche Aktivitäten können nützlich oder auch vorgeschrieben sein, wenn beispielsweise die entwickelte Lösung langfristig weiter gewartet und verbessert werden muss, wenn der Sourcecode an ein anderes Entwicklerteam (z. B. beim Kunden) zur Weiterentwicklung übergeben werden muss oder er eine formale Genehmigung durch eine externe Prüfstelle erfordert (z. B. Compliance). Dieses erweiterte

Anforderungsmanagement hängt daher auch primär vom Projektkontext ab und kann je nach Kontext und verwendeter Tools innerhalb oder außerhalb des Backlogs stattfinden.

Wenn über eine einfache Backlog-Verwaltung hinausgehend noch weitere Aktivitäten geplant oder gefordert sind, empfiehlt es sich, ein passendes Requirements-Management-Werkzeug zu verwenden, da das Handling umfangreicher Inhalte und die Nachvollziehbarkeit mit physischem Board und Karten schwierig ist und meist schnell inkonsistent wird.

2.1 Artefakte für das Requirements Management

2.1.1 Agiles Requirements-Board

Definition	Das agile Requirements-Board ist ein Visualisierungs- und Managementinstrument, auf dem der Zustand der Requirements in agilen Projekten dargestellt und gesteuert wird.
Anwendung	Durch das agile Requirements-Board wird der Prozess der Requirements-Validierung, Analyse, Selektion, Verfeinerung, Abstimmung und auch der Freigabeprozess der Requirements für den nachfolgenden Umsetzungs-Sprint systematisiert und für alle Beteiligten transparent gemacht.
Zuständig	■ Product Owner (hauptverantwortlich) ■ Team (mitwirkend)
Eigenschaften	■ Zeigt den Zustand der Requirements im Projekt an ■ Eine Spalte für jeden Zustand, in dem sich ein Requirement befinden kann (ähnlich einem Kanban-Board) ■ Im Besitz des Product Owners
Zeitpunkt	■ Laufende Arbeit am Board ist sinnvoll und möglich. ■ Schwerpunktarbeit am Board im Refinement-Meeting, Sprint-Planungs-Meeting und Sprint-Review-Meeting

Wozu noch ein Board? Typischerweise wird ja in agilen Projekten zur operativen Steuerung der Umsetzung der Requirements (zumeist als User Stories) das Taskboard verwendet. Als Eingangskriterium bzw. Filter in das Sprint Backlog und damit auch auf das Taskboard gibt es die Definition of Ready (DoR).

Aber wie kommen nun die Requirements systematisch in das Sprint Backlog bzw. auf das Taskboard? Dazu muss der Requirements-Entstehungs- und -Analyseprozess passend unterstützt bzw. abgebildet werden.

Dies kann durch passende Listen oder auch Requirements-Tools, die agile Ansätze unterstützen, oder eben durch ein weiteres vorgelagertes Management- und Visualisierungsinstrument wie das Requirements-Board (sei es als »papierbasiertes Tool« oder als Softwarelösung mit passenden Tools) umgesetzt werden, das das Requirements-Management ähnlich dem Taskboard unterstützt. Das Taskboard ist jedoch im Besitz des Entwicklungsteams, während das Requirements-Board im Besitz des Product Owners ist.

Eine Idee oder ein Epic wird nicht ohne entsprechende Requirements-Engineering-Arbeit zu einem passenden Backlog Item für den nächsten Umsetzungs-Sprint. Oft findet dieser Prozess recht intuitiv und schlecht managebar statt. Es liegt nahe, die agile Managementtechnik in Form des Taskboards zu adaptieren und für das Management der agilen Requirements zu verwenden.

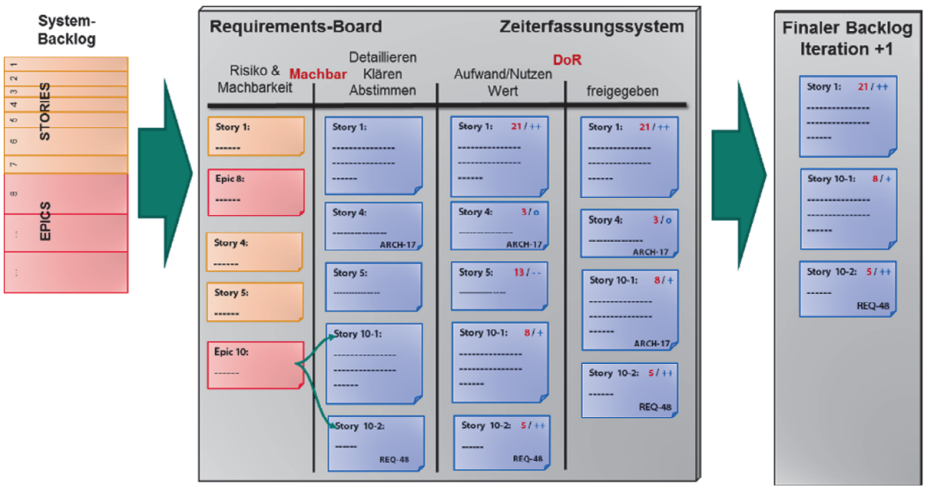


Abb. 2–1 *Das agile Requirements-Board*

Das agile Requirements-Board ist ein Visualisierungsinstrument, auf dem der Entwicklungszustand der Requirements eines Projektes dargestellt und gesteuert wird.

Dadurch kann die Selektion, Verfeinerung und auch der Freigabeprozess der Requirements für den nachfolgenden Umsetzungs-Sprint systematisiert und für alle Beteiligten transparent gemacht werden. Das agile Requirements-Board ist somit ein Entwicklungs- und Managementtool für Requirements in agilen Projekten.

Nachfolgend wird das agile Requirements-Board sowie seine Arbeitsweise und Anwendung näher beschrieben.

Grundelemente des agilen Requirements-Boards

Das agile Requirements-Board bildet die einzelnen Schritte bzw. den Prozess der Requirements-Validierung, -Analyse und -Abstimmung ab (siehe Abb. 2–2).

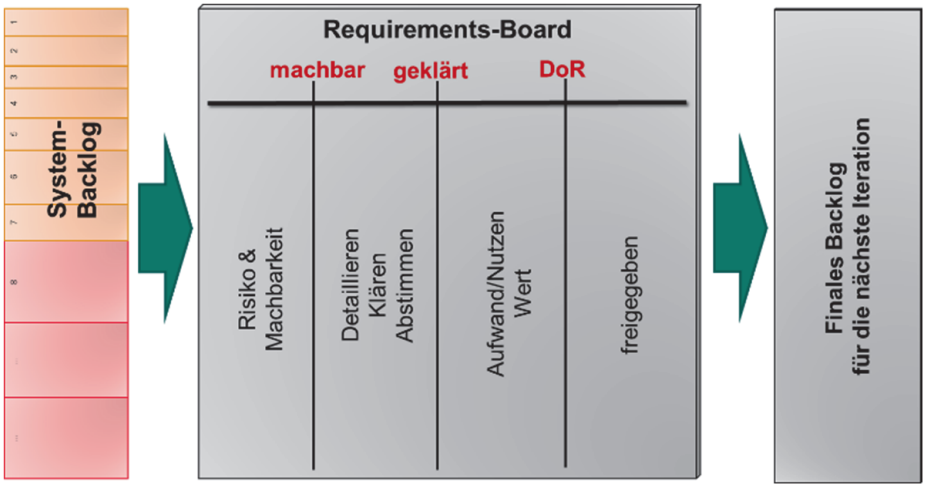


Abb. 2–2 *Grundelemente des agilen Requirements-Boards*

Das agile Requirements-Board hat als Input die Anforderungselemente (z.B. Epics, Stories oder andere Anforderungen) aus einem übergeordneten Backlog (z.B. Product Backlog, System-, Release-Backlog). Auf dem Requirements-Board werden diese Anforderungen systematisch von einem Zustand in den nächsten gebracht und zuletzt durch Check gegen die Definition of Ready (DoR) freigegeben.

Freigegebene Anforderungen können dann in das finale Backlog für die nächste Iteration übernommen werden. Dies ist der Output des agilen Requirements-Boards. Die Umsetzung der passend eingestuften Requirements wird dann vom Entwicklungsteam über das Taskboard in Scrum gemanagt.

Anwendung des agilen Requirements-Boards

Nachfolgend wird Schritt für Schritt erklärt, was im agilen Requirements-Board passiert und wie es angewendet werden kann.

- **Requirements-Board füllen**
  Aus dem initialen und unbewerteten Backlog (Product Backlog, System-Backlog etc.) werden Requirements entnommen, die mit großer Wahrscheinlichkeit in den nächsten 2–3 Sprints umgesetzt werden sollen. Diese Requirements sind typischerweise User Stories, können aber auch andere Requirements-Artefakte sein wie Epics, Use Cases, Prozessdarstellungen etc.
- **Machbarkeit und Risiko bewerten**
  Vom Entwicklungsteam und ggf. weiteren Stakeholdern wird jedes der initialen Requirements hinsichtlich Machbarkeit und Risiko bewertet. Wenn hier bezüglich Machbarkeit Unklarheiten vorhanden sind, sollte für dieses Requirement evtl. eine vom Aufwand angemessene Machbarkeitsuntersuchung (Spikes, Evaluation-Story) durchgeführt werden, die dann ein aussagekräftiges Ergebnis bringt, oder das Risiko für dieses Requirement muss entsprechend erhöht werden.

  Requirements, die als nicht machbar eingestuft werden, können in einen »Rejected-Container« verschoben werden, wo sie aufbewahrt werden, um später noch sagen zu können, welche Requirements bewusst entfernt wurden, oder um evtl. aus diesem Pool bei geänderten Rahmenbedingungen wieder einzelne Requirements herauszunehmen und erneut eine Machbarkeitsanalyse durchzuführen.

  Für jedes Requirement wird dann auch das Risiko eingestuft. Requirements, die ein zu hohes Risiko haben, können auch in den Rejected-Container verschoben werden. Alternativ kann natürlich auch entschieden werden, ein hohes Risiko bewusst einzugehen.
- **Verfeinerung des Requirements**
  Abhängig vom Risiko des Requirements empfiehlt es sich, die Detailliertheit bei der Spezifikation zu erhöhen. Requirements mit hohem Risiko sollten detailliert spezifiziert werden, sofern die initiale Formulierung nicht schon entsprechend detailliert vorgenommen wurde, und für Requirements mit niedrigem Risiko kann evtl. die initiale Formulierung ohne Änderung beibehalten werden. Je nach Bedarf ist auch eine weitere Besprechung, Klärung und Abstimmung mit den betroffenen Stakeholdern sinnvoll.
- **Aufwand abschätzen**
  Wenn das Requirement nun ausreichend »greifbar« für das Team ist, wird die Aufwandsschätzung durchgeführt. Diese erfolgt oft über die Story-Point-Schätzung. Es können aber auch andere Methoden angewendet werden.

  Anmerkung: Mit einiger Routine sollte für die Aufwandsschätzung eine Genauigkeit von ±20 % oder besser angestrebt werden. Wenn die Schätzwerte

hier deutlich schlechter sind, sollte dies in der Sprint-Retrospektive auffallen. Ein möglicher Grund dafür kann sein, dass die Requirements zu ungenau waren. Man kann dies dann zum Anlass nehmen, für den nächsten Sprint hier mehr Augenmerk auf die Requirements-Klärung und -Verfeinerung zu legen.

- **Nutzen und Wert bestimmen**
  Zusätzlich zum Aufwand wird nun der Wert bzw. Nutzen ermittelt. Dieser wird dem Aufwand gegenübergestellt. Der Nutzen kann ebenfalls in »Nutzenpunkten« (z.B. im Rahmen von Business-Value-Poker) oder im einfachsten Fall durch ein Bewertungsschema wie z.B. »++«, »+«, »o«, »-«, »--« abgebildet werden.

  Wenn sich hier ein schlechtes Aufwand-Nutzen-Verhältnis für den Auftraggeber abzeichnet (z.B. Requirements mit vielen Story Points und einer Nutzenbewertung von »-«), dann werden diese Requirements ebenfalls wieder in den Rejected-Container verschoben oder in das initiale Backlog an eine weiter hinten gelegene Stelle zurückgegeben.

- **DoR-Check**
  Requirements, die den Kosten-Nutzen-Check bestehen, werden nun dem finalen Qualitätscheck mit der Definition of Ready unterzogen. Diejenigen Requirements, die alle Prüfpunkte der DoR erfüllen, werden in den Status »freigegeben« übernommen.

- **Umsetzungs-Backlog füllen**
  Die »freigegeben«-Spalte am Requirements-Board ist nun der Sammelpunkt für alle »fertigen« Requirements, die dann nach Bedarf in die Umsetzung übernommen werden können. Es ist nicht zwingend notwendig, alle fertig spezifizierten Requirements in den nächsten Umsetzungs-Sprint zu übernehmen. Dieser Platz am Board ist auch der Puffer für das Umsetzungsteam für einige weitere Sprints. Eine gute Zahl an fertigen Requirements kann so gewählt werden, dass daraus der direkt folgende Sprint ausreichend gefüllt werden kann und dann noch genügend Requirements vorhanden sind, um zumindest noch einen weiteren Sprint komplett zu füllen.

  Mehr als drei Sprints im Voraus zu spezifizieren ist meist auch nicht sinnvoll, da dann die Gefahr erhöht wird, dass die fertigen Requirements »altern« und die Analyse und Spezifikationsarbeit umsonst war. Dies würde dem agilen Ansatz widersprechen. Wenn jedoch die Bearbeitung der Requirements im Projekt nicht kontinuierlich erfolgt, dann ist es evtl. auch notwendig, doch mehr Requirements in der »freigegeben«-Spalte vorzuhalten, damit nicht die Gefahr besteht, dass das Umsetzungsteam in einem Sprint keine Arbeit hat.

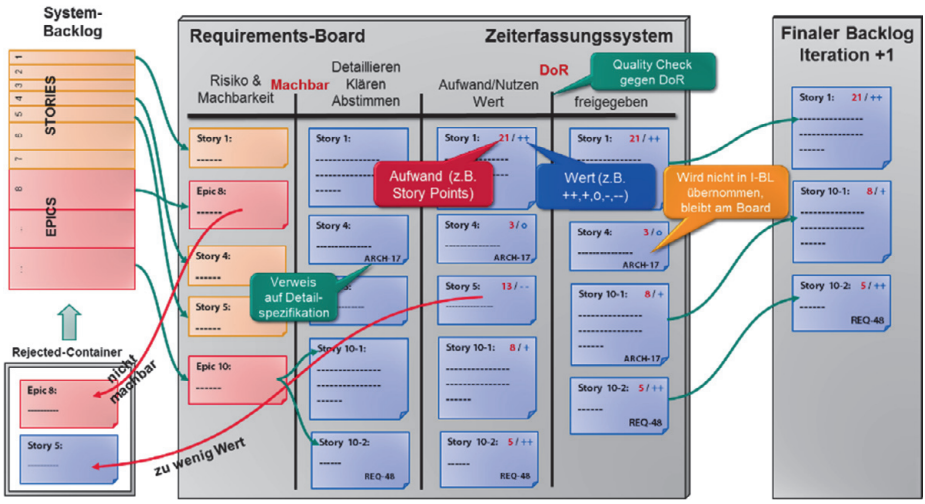


Abb. 2–3 *Gesamtüberblick über das agile Requirements-Board*

3 Rechtliche Themen[1]

Im Agilen Manifest wird betont, dass »Funktionierende Software höher gewertet wird als umfassende Dokumentation«. In jeder agilen Methodenbeschreibung steht, dass es besser ist, mehr zu kommunizieren, als alles im Detail niederzuschreiben. Dies ist einer der Grundgedanken der agilen Vorgehensweise und kann in einem guten und partnerschaftlich verlaufenden Projekt auch voll unterstützt werden.

In der Praxis gibt es jedoch viele Projektsituationen, bei denen verschiedene rechtliche Aspekte und Haftungsfragen betrachtet werden müssen, an die die Entwickler oder auch Kunden oft nicht denken. Erst im Schadensfall kommen dann die Probleme zutage, die durch schlechtes Requirements Engineering verursacht wurden. Dann ist es jedoch zu spät!

> Solange alles gut und in den erwarteten Bahnen verläuft, wird kaum jemand darauf drängen, dass alles aufgeschrieben werden muss, um sich abzusichern. Wenn dann jedoch das Projekt schlecht läuft, kommt meist die Aussage: »Hätten wir das doch vorher schriftlich festgehalten und detaillierter vereinbart!«

Bei den Juristen gibt es folgendes Sprichwort: »Wer schreibt, der bleibt und wer red't, der geht!« Dies kommt daher, dass in einem Streitfall – im schlimmsten Fall vor Gericht – alles Niedergeschriebene eine um vieles höhere Beweiskraft hat als eine mündliche Aussage.

Das ist scheinbar ein konträrer Standpunkt zum Agilen Manifest, das primär die direkte persönliche Kommunikation in den Vordergrund rückt. Es gibt die beiden Extreme: »viel reden und wenig schreiben« bis zu »möglichst alles aufschreiben, um abgesichert zu sein«. Funktionierende Software ist ein sehr hoher Wert in allen agilen Methoden, allerdings sind auch Dokumentation und Verträge wertvoll, wenn die Situation es erfordert! In der Praxis wird dies leider oft nicht ausreichend berücksichtigt und das Agile Manifest teilweise einseitig ausgelegt und als Vorwand genommen, um in Projekten auf schriftliche Spezifikationen und den damit verbundenen Aufwand zu verzichten!

---

1. Dieses Kapitel ersetzt keine rechtliche Beratung, sondern stellt die Erfahrungen des Autors als Unternehmensberater und als Gutachter in Gerichtsverfahren dar. Bei juristischen Fragen ist ein Rechtsanwalt zu konsultieren. Vom Autor wird keine Haftung übernommen.

Ein rechtliches Problem zeigt sich bei der Vereinbarung von Festpreisen in Projekten. Der Kunde möchte verständlicherweise möglichst bald wissen, was sein Vorhaben kosten wird. Zu frühen Projektzeitpunkten ist es jedoch unmöglich, eine einigermaßen haltbare Schätzung über den für ein bestimmtes Projektziel und eine grobe Feature-Menge zu erwartenden Preis abzugeben. Damit tritt in praktisch jedem Projekt, für das vor der Fertigstellung ein Festpreis vereinbart wird, das Problem auf, dass der geschätzte Preis und die Leistung nicht zusammenpassen und dass spätestens bei Überschreiten der anfangs prognostizierten Kosten die Diskussionen beginnen, wer nun den Mehraufwand bezahlt.

Es gibt mittlerweile auch diverse Abhandlungen zum Thema »agiler Festpreis«, in denen jedoch leider auch keine praktikablen Lösungen für dieses Problem angegeben werden. Möglicherweise ist dieses Thema auch nicht wirklich lösbar, sondern es wird immer ein Kompromiss sein. Durch agile Prinzipien und spezielle Vereinbarungen im Vertrag kann das Thema Festpreis zwar etwas entschärft werden, das eigentliche Problem wird dabei jedoch nicht gelöst.

Nachfolgend ist ein Vier-Stufen-Modell definiert, das angewendet werden kann, um das Risiko in »agilen Festpreisprojekten« etwas zu reduzieren.

3.1 Das Vier-Stufen-Modell für agile Festpreisprojekte

Die in Abbildung 3–1 dargestellte mehrstufige Vorgehensweise kann bei agilen Festpreisprojekten angewendet werden:

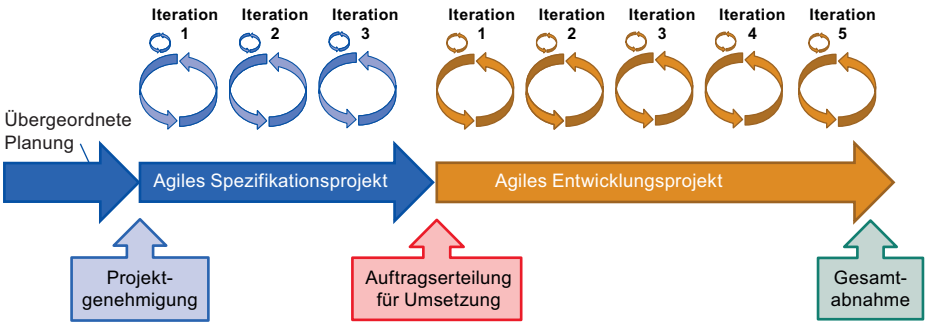


Abb. 3–1 *Mehrstufige Abwicklung eines agilen Festpreisprojekts*

3.1.1 Stufe 1: Definition der Projektziele und ersten Kundenanforderungen

Dies kann noch sehr grob geschehen und stellt keine Basis für den Festpreis dar. Diese Phase dient dazu, sich über die grundsätzlichen Projektinhalte klar zu werden und damit dann geeignete Partner und Anbieter zu selektieren.

In vielen Unternehmen wird diese Phase »Projekt-Feasibility-Phase« oder »Projektantragserstellung« genannt. Oft wird dafür auch ein bestimmtes Budget vorgesehen, das für die erste Recherche und Analyse verwendet werden kann.

Der Aufwand dieser Phase kann nicht vorab definiert werden, da dieser Schritt ja so lange durchgeführt werden muss, bis sich der Auftraggeber grundsätzlich klar ist, was er will bzw. ob er überhaupt ein Projekt starten will. Natürlich wird es auch Situationen geben, in denen der Auftraggeber dies nicht selbst herausfinden kann oder will. In diesen Fällen ist es hilfreich, einen fachkundigen Berater auf Auftraggeberseite hinzuzuziehen. Möglich ist auch eine Evaluierung von eventuell am Markt verfügbaren (Alternativ-)Lösungen, um sich ein erstes Bild zu machen. Auch die Möglichkeit, dass man in diesem Schritt zu keinem sinnvollen Ergebnis bezüglich der Projektvorstellung kommt, muss betrachtet werden. In diesem Fall endet das Projektvorhaben bereits hier.

3.1.2 Stufe 2: Agiles Erstellen der Vertragsbasis

Mit dem Ergebnis des ersten Schritts wird nun die Erstellung der Vertragsbasis (= Kundenanforderungen) begonnen.

Wenn für die Erstellung einer Vertragsbasis schon eine gute Grundlage aus dem ersten Schritt vorhanden ist, kann der Aufwand für die Spezifikationserstellung eventuell auch schon vorab genauer geschätzt bzw. können Angebote von Beratern oder potenziellen Lieferanten eingeholt werden, die dann mit dem Auftraggeber die Vertragsbasis erstellen.

Die Vertragsbasis kann durchaus iterativ und agil erstellt werden. Zum Beispiel könnte ein agiles Team für die Spezifikationserstellung gebildet werden, da es sowieso sinnvoll ist, alle relevanten Rollen (Kunde, Entwickler, Architekt, Tester, Betriebstechniker etc.) in die Spezifikationserstellung einzubinden. Es wird in diesem Fall dann kein lauffähiges Softwareartefakt geliefert, sondern eine vom Kunden akzeptierte Spezifikationsversion oder auch prototypische Softwareartefakte aus Spikes oder Masken- und Reportbeispiele zur Festlegung von Usability-Aspekten etc.

Für diesen Schritt der Spezifikationserstellung kann meist kein Festpreis abgegeben werden. Der Lieferant kann ja kaum abschätzen, was der Kunde noch für Ideen oder Wünsche hat und wann der Kunde mit der erstellten Spezifikation zufrieden ist. Dies würde eventuell funktionieren, wenn sich Lieferant und Kunde bereits länger kennen.

Bei der Spezifikationserstellung gibt es einige wichtige Punkte zu beachten:

- **Inwieweit soll ein potenzieller Lieferant in die Erstellung der Spezifikation eingebunden werden?**
  Wenn dieser eingebundene Lieferant der Favorit des Kunden für die Umsetzung ist, dann ist dies sicherlich eine gute Möglichkeit, da sich beide Partner dabei schon gut kennenlernen und vor der Beauftragung der Umsetzung für den Kunden ein definierter Ausstiegspunkt vorhanden ist, wenn es doch nicht passen sollte.

Problematisch wird es dann, wenn aufgrund von Compliance-Regeln, Organisationsrichtlinien oder Gesetzen ein neutraler Wettbewerb zwischen mehreren Lieferanten notwendig ist. Hier würde der eingebundene Lieferant klar bevorzugt werden. In diesem Fall ist es daher notwendig, den in die Spezifikationserstellung eingebundenen Lieferanten dann von der Umsetzungsphase auszuschließen.

- **Wie ist es mit den Kosten für die Spezifikation?**
  Manchmal sind Kunden der Ansicht, dass Lieferanten die Spezifikationsleistungen als kostenlose Vorleistung für die Umsetzungsphase erbringen müssen. Dies ist an sich ein unfairer Zugang, da im Rahmen der Spezifikation eine wesentliche Leistung erbracht wird, die auch honoriert werden soll.

  Wenn der Kunde die Spezifikationsleistungen extra beauftragt, ist das Risiko überschaubar. Es sollte auch vereinbart werden, dass der Kunde das Recht hat, in der Spezifikationsphase jederzeit abzubrechen, wenn die Spezifikation ausreichend für die Auftragsvergabe der Umsetzung ist oder er das Projekt eventuell auch nicht durchführen möchte. Der Berater oder eingebundene Lieferant bekommt dann die Leistungen bis zu diesem Zeitpunkt vergütet. Das Risiko ist daher jederzeit überschaubar für den Kunden.
- **K.-o.-Kriterien festlegen!**
  Es müssen vor der Beauftragung der Umsetzungsphase alle K.-o.-Kriterien definiert und auf Machbarkeit analysiert worden sein. Wenn dies nicht durch den Kunden selbst beurteilt werden kann, sollte der Berater oder eingebundene Lieferant dies mittels Spikes durchführen. Diese Spikes sollten schon im Rahmen der Spezifikationsiterationen stattfinden, sodass bis spätestens zum Start eines Umsetzungsprojekts die Unklarheiten für alle K.-o.-Kriterien beseitigt sind.
- **Nicht funktionale Anforderungen** (**NFA**) sollten berücksichtigt und möglichst klar (messbar) festgelegt werden!
- **Detailliertheit der Vorabspezifikation?**
  Die Spezifikation soll bewusst nicht bis ins letzte Detail gehen, da dies ja kontraproduktiv zur agilen Vorgehensweise wäre. Durch das erhöhte Kostenrisiko in einem Festpreisprojekt sollte jedoch insgesamt darauf geachtet werden, dass die Anforderungen so spezifiziert werden, dass die Projektkosten guten Gewissens mit einer Genauigkeit von etwa ≤20 % festgelegt werden können, sodass es kein großes Kostenrisiko mehr gibt. Für den Kunden und Lieferanten bleibt dadurch jedoch noch ausreichend Spielraum für eine variable Zielaussteuerung im agilen Umsetzungsprojekt.

  Mit einer hohen Detailliertheit sollten alle K.-o.-Kriterien und die High-Risk-Anforderungen beschrieben werden. Alle anderen Themen können je nach Priorität und Risiko auch nur grob dargestellt werden.

3.1.3 Stufe 3: Festpreisangebot durch den Lieferanten

Wichtig ist hier als »Eingangsvoraussetzung«, dass die zugrunde liegende Spezifikation eine akzeptable Schätzgenauigkeit (z. B. ±20 %) ermöglichen soll.

Der Kunde sollte dies dann auch bei einem Festpreisangebot als Projektpuffer zusätzlich budgetieren, da durch die Tatsache, dass die Spezifikation nicht zu 100 % alles beschreibt, zumindest in den noch unklar formulierten Bereichen Mehraufwände im Ausmaß der Schätzungenauigkeit zu erwarten sind, auch wenn an den zugrunde liegenden Requirements keine Änderungen mehr vorgenommen werden.

Beispiel: Zusatzaufwände trotz Festpreis und nicht geänderter Spezifikation

In der vom Kunden und Lieferanten mit Abgabe des Festpreises akzeptierten Spezifikation steht als Anforderung, dass ein »Report zur monatlichen Auswertung der Zeitdaten eines Mitarbeiters« erstellt werden soll. Eine genaue Abbildung und Beschreibung des Reports ist in der Spezifikation nicht enthalten.

Der erste Vorschlag des Lieferanten, der die Spezifikation grundsätzlich voll erfüllt, gefällt dem Kunden jedoch nicht.

Der Kunde möchte zusätzliche Felder angezeigt haben, möchte vorab noch eine Selektionsmöglichkeit in einer eigenen Maske, möchte im Report gruppieren und nach bestimmten Feldern sortieren etc.

Statt der vorab geschätzten Aufwände von zwei Tagen[a] für diesen Report sind nun vom Lieferanten fünf Tage notwendig. Die drei Tage Differenz werden dem Kunden als Mehraufwand für den besonderen Wunsch zusätzlich zum Festpreis verrechnet.

Da die Spezifikation in diesem Teilbereich keine Realisierungsdetails enthielt, hat der Kunde auch kein Recht, auf der kostenlosen Umsetzung seiner speziellen Wünsche zu bestehen.

Alternativ kann dem Kunden auch vorgeschlagen werden, diesen Mehraufwand zu kompensieren, indem ein anderes Feature aus dem Backlog, das etwa die gleiche Größe hat, dann nicht realisiert wird.

a. Die Einheit »Tage« wurde hier stellvertretend gewählt. Im Grunde könnte auch mit Story Points oder einer anderen Einheit gerechnet werden. Wichtig ist nur, dass dem Kunden klar ist, was eine Einheit kostet.

3.1.4 Stufe 4: Agile Projektabwicklung

Nach der Auswahl eines passenden Lieferanten und der Vereinbarung des vertraglichen und kaufmännischen Rahmens beginnt die Umsetzung in Form einer agilen Projektabwicklung.

Der Unterschied hier ist, dass durch die Vorabspezifikation die Bandbreite der Kosten- bzw. Anforderungsabweichungen etwas mehr eingeschränkt wird als bei einem agilen Projekt, bei dem am Anfang große Unklarheit herrscht, was der Kunde eigentlich möchte.

Diese Vorgehensweise ist eine gewisse Gratwanderung zwischen Vorabspezifikationsaufwand und Unschärferisiko bei der iterativen Umsetzung. Abbildung 3–2 zeigt, wie das Unschärferisiko durch die vorgeschlagene Vorgehensweise mit einem vorhergehenden agilen Spezifikationsprojekt entsprechend verringert wird.

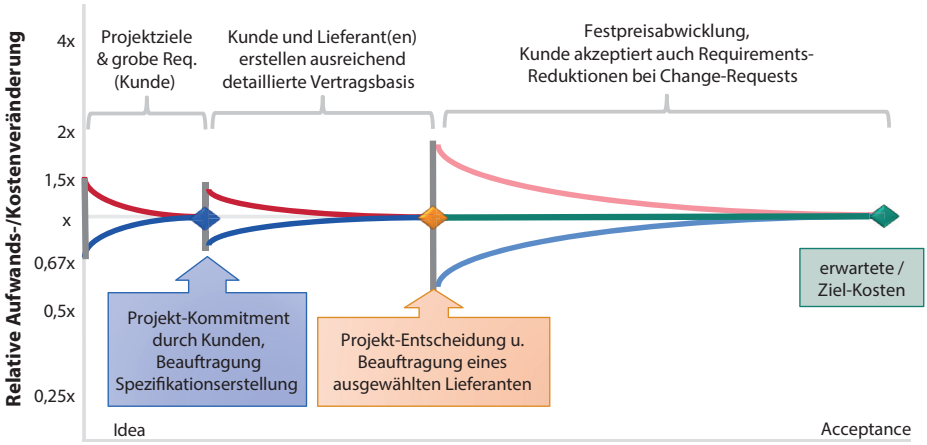


Abb. 3–2 *Risikoreduktion durch das Vier-Stufen-Modell für agile Festpreisprojekte*

In »klassisch agilen« Projekten wird von der ersten Iteration an schon der Fokus auf den lauffähigen Code gelegt. In diesen Anfangsiterationen herrscht aber typischerweise sowohl auf Kunden- als auch auf Lieferantenseite noch große Unsicherheit und es besteht ein Entscheidungsspielraum, der ein entsprechendes Preisrisiko bedeutet. Bei den in den ersten Iterationen zu erwartenden Änderungen geht der hier bereits in Detaildiskussionen und Implementierungen investierte Aufwand bei nachträglichen Änderungen dann teilweise wieder verloren.

Im Vier-Stufen-Modell für agile Festpreisprojekte wird dieser Aufwandsverlust der ersten Iterationen vermieden und stattdessen dieser Aufwand dazu verwendet, die Klärung der wichtigsten unklaren Anforderungen durchzuführen und damit das Preisrisiko in einen vertretbaren Bereich (z. B. ±20 %) zu bringen. Durch das Reduzieren der Codierung in den ersten Iterationen, wobei Prototypen auch im Rahmen der Spezifikationsiterationen sinnvoll sein können, konzentriert sich der Kunde stärker auf das, was er eigentlich will, und weniger auf die Implementierungsdetails, die sonst in den ersten Iterationen im Rahmen der Umsetzung diskutiert werden.

Quellenverzeichnis

[**IREB**] Interinational Requrements Engineering Board, *https://www.ireb.org/de/*.

[**IREB CPRE FL**] International Requirements Engineering Board (IREB): Lehrplan für den Certified Professional for Requirements Engineering – Foundation Level, Version 2.2.1, Juli 2017, *https://www.ireb.org/content/downloads/2-syllabus-foundation-level/ireb_cpre_syllabus_fl_de_v221.pdf*.

[**IREB Glossar**] International Requirements Engineering Board (IREB): A Glossary of Requirements Engineering Terminology, Version 1.7, Mai 2017, *https://www.ireb.org/content/downloads/1-cpre-glossary/ireb_cpre_glossary_17.pdf*.

[**IREB RE@Agile FL**] International Requirements Engineering Board (IREB): RE@Agile Primer – Lehrplan und Studienleitfaden, Version 1.0.2, November 2017, *https://www.ireb.org/content/downloads/27-cpre-re-agile-primer-syllabus/ireb_cpre_re%40agileprimersyllabusandstudyguide_de_v1.0.2.pdf*.

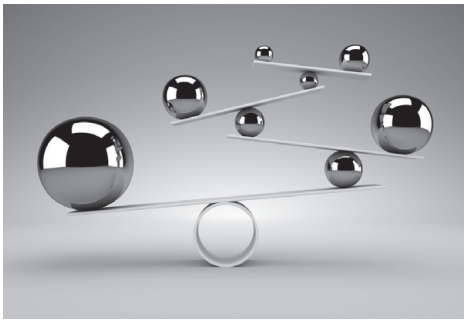
[**Schwaber & Sutherland 2017**] Schwaber, K.; Sutherland, J.: Der Scrum Guide™. Der gültige Leitfaden für Scrum: Die Spielregeln. November 2017, *https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-German.pdf*.

AGILE KOMPETENZ UND METHODEN

Effektiv und dynamisch in der System- und Software Entwicklung

Profitieren Sie von Wissen und Praxiserfahrung der Software Quality Lab Consultants aus zahlreichen agilen Projekten. Steigern Sie Effektivität, Flexibilität und Qualität in Ihren Entwicklungs-und Testprozessen.

- Sie erhalten Unterstützung bei der Einführung und Anwendung von agilen Vorgehensweisen.
- Das Optimierungspotential Ihrer agilen Prozesse wird identifiziert und ausgeschöpft.
- Sinnvolle Anwendungsmöglichkeiten agiler und pragmatischer Techniken und Methoden werden definiert.
- Sie und Ihre Mitarbeiter werden in der Umsetungsphase gecoacht und operativ unterstützt.



„Es ist nur natürlich, sich eine Methode vorzunehmen und sie auszuprobieren. Wenn es schief geht, gib es offen zu und versuche etwas anderes. Die Hauptsache ist, dass du überhaupt etwas versuchst."

Franklin D. Roosevelt

Ihr Nutzen

- **Qualitativ, Effizient und Effektiv**
  in agilen Prozessen
- **Ganzheitlich**
  Expertenwissen in agilen Methoden und darüber hinaus
- **Pragmatisch**
  keine dogmatischen Prediger sondern fundierte Praktiker
- **Umfassende Unterstützung**
  Beratung, Schulung, Coaching, operative Unterstützung
- **Blick über den Tellerrand**
  Wir kennen nicht nur die agilen Methoden, sondern haben auch eine Fokus auf alle anderen Themen im Entwicklungsprozess und verbinden dies mit den agilen Methoden zu einem sinnvollen Prozess- und Methoden-Framework.

Unsere Leistungen

- **Initialanalyse zur Bestimmung Ihrer „agilen Reife"**
- **Maßnahmen zur Prozessverbesserung**
  Consulting- und Coaching-Pakete nach Maß von der Einführung in die agile Entwicklung bis zum Team-Coaching zur effizienten Umsetzung von agilen Methoden
- **Operative Unterstützung durch agile Tester und Testautomatisierer**
- **Trainings und Seminare zu agilen Themen**
  - Agiles Requirements Engineering
  - Testen und Testautomatisierung in agilen Teams
  - Vorgehensmodelle wie Scrum und Kanban.

  Mehr Informationen finden Sie im aktuellen Seminarprogramm unter www.software-quality-lab.com/academy

REQUIREMENTS ENGINEERING

Gute Requirements sind der Grundstein für ein erfolgreiches Projekt!

Es gilt in plangetriebenen Projekten ebenso wie im agilen Vorgehen: Professionelles Requirements Engineering stellt sicher, dass Anforderungen passend ermittelt, ausreichend dokumentiert, geprüft, mit dem Kunden abgestimmt und angemessen verwaltet werden. Nur so entsteht am Ende ein Produkt, das die Erwartungen der Kunden erfüllt und vielleicht sogar übertrifft.
Nutzen Sie unsere langjährige Erfahrung im Requirements-Engineering!

- **Pragmatisch:**
  Wir suchen für Sie den passenden Mix zwischen Detailliertheit und Agilität im Requirements-Engineering.
- **Ganzheitlich:**
  Requirements-Engineering ist nicht nur das Schreiben von Anforderungen. Wir blicken „über den Tellerrand" und berücksichtigen auch Kommunikation, Kreativität, Nachhaltigkeit, Effizienz und andere Faktoren, die für eine erfolgreiche Anwendung wichtig sind.
- **Umfassend:**
  Wir helfen Ihnen mit Beratung, Coaching, Schulungen, passenden Tools und bei Bedarf auch operativer Unterstützung.

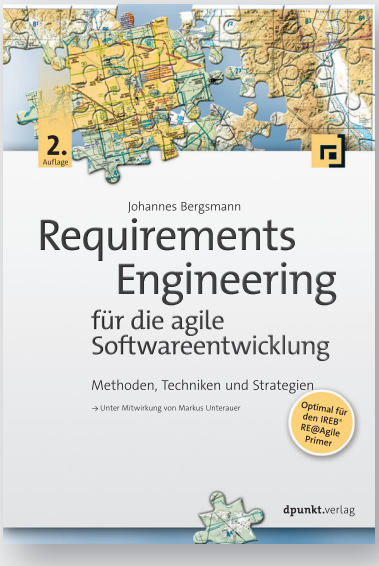
„Warum haben wir eigentlich nie Zeit, die Sache richtig zu machen, aber immer Zeit, sie nochmal zu machen?"
Weinberg, Systemdenken und Softwarequalität

Ihr Nutzen

- Requirements-Engineering wird optimiert
- Anwendungsmöglichkeiten von Requirements-Engineering in agilen Vorgehensweisen werden definiert
- Sie werden von erfahrenen Beratern gecoacht und auch in der Umsetzungsphase unterstützt
- Sie werden schneller und zielgerichteter in der Spezifikation und Umsetzung Ihrer Projekte
- Requirements Engineering Tools werden effektiver und effizienter genutzt
- Requirements Engineering Tools sind an moderne RE Methoden und Prozesse optimal angepasst

Unsere Leistungen

- Potenzialanalyse im Requirements Engineering
- Schnelle Überblicksanalyse und Workshops im RE
- Einführung Agiles Requirements Engineering (z. B. Product Owner Rolle, Team Aufgaben, Definition of Ready, etc.)
- Anforderungsreviews
- Requirements Engineering Prozessanalyse und Workshops zu Methoden und Prozessdefinition
- Operatives Erstellen von Anforderungsspezifikationen
- Spezifikationen für Ausschreibung erstellen/begleiten
- Auswahl und Einführung von Requirements-Tools
- Requirements Engineering Schulungen



Johannes Bergsmann

Requirements Engineering für die agile Softwareentwicklung

Methoden, Techniken und Strategien

→ Unter Mitwirkung von Markus Unterauer

2., überarb. und akt. Auflage 2018
386 Seiten, Festeinband
€ 36,90 (D)

ISBN:
Print 978-3-86490-485-1
PDF 978-3-96088-189-6
ePub 978-3-96088-190-2
mobi 978-3-96088-191-9

Dieses Buch gibt einen praxisorientierten Überblick über die am weitesten verbreiteten Techniken für die Anforderungsspezifikation und das Requirements Management in agilen Projekten. Es beschreibt sowohl sinnvolle Anwendungsmöglichkeiten als auch Fallstricke der einzelnen Techniken. Darüber hinaus werden Qualitätsaspekte für Anforderungen im agilen Umfeld vorgestellt sowie rechtliche und wirtschaftliche Aspekte und Zusammenhänge für größere Organisationen.

Das Buch ist Hilfestellung und Nachschlagewerk, um in der täglichen Praxis der agilen Projekte Requirements Engineering und Requirements Management professionell und mit nachhaltigem Nutzen umzusetzen.

Die 2. Auflage berücksichtigt den Lehrplan »RE@Agile Primer« des International Requirements Engineering Board (IREB).

dpunkt.verlag
www.dpunkt.de