1 Einleitung und Motivation

1.1 Über dieses Buch

Der Fokus dieses Buches liegt darauf, gute Methoden und Techniken – egal welchen Alters oder welcher Ausrichtung – unvoreingenommen aufzugreifen und sie nicht von vornherein auszuschließen, nur weil es sich dabei um »klassische« oder »agile« Methoden oder Techniken handelt. Aussagen wie »nicht agile Methoden sind überfrachtet, ineffizient und schlecht« oder »agile Methoden sind was für Individualisten oder Dokumentationsfeinde« gehören nicht zu einer weitsichtigen und nachhaltigen Denkweise.

Jede der in diesem Buch genannten Methoden und Techniken stellt ein Werkzeug im Werkzeugkasten der Softwareentwicklung dar und soll hier mit ihren Vorteilen und Nachteilen in bestimmten Projektsituationen betrachtet werden. Es soll klar werden, wo und wie diese Methoden und Techniken in agilen Projekten eingesetzt werden können, um einen Nutzen für das Projekt zu stiften. Es werden daher auch sinnvolle Anwendungsmöglichkeiten und Fallstricke der einzelnen Methoden und Techniken dargestellt und im Kontext eines nachhaltigen, systematischen und praxisorientierten Requirements Engineering in der agilen Softwareentwicklung betrachtet.

Abgerundet wird das Buch durch die Behandlung von Qualitätsaspekten für Requirements, organisatorische Aspekte sowie durch rechtliche Aspekte bei der Anwendung von Requirements im agilen Umfeld.

Dieses Buch ist keine komplette Einführung in agile Methoden. Die Kapitel 1 und 2 geben zwar einen kurzen Überblick über das Agile Manifest, wesentliche agile Konzepte und Scrum, als am häufigsten eingesetzten Vertreter agiler Methoden. Der Hauptfokus des Buches liegt jedoch im Requirements Engineering und Requirements Management im agilen Kontext.

Hilfreich beim Lesen ist ein grundlegendes Verständnis des aktuellen Stands im Bereich Requirements Engineering und agiles Vorgehen sowie insbesondere die Kenntnis der allgemein anerkannten Begriffe, Techniken und Grundlagen aus diesen Themenbereichen. Zur vertiefenden Betrachtung wird z. B. auch der Lehrplan und die darin genannten Originalquellen für die Ausbildung zum »Certified

Professional for Requirements Engineering (CPRE)« [IREB CPRE FL 2017] des International Requirements Engineering Board [IREB] empfohlen, der den grundlegenden Stand der Technik zum Thema Requirements Engineering darstellt.

Das Buch soll Einführungslektüre sowie Praxisleitfaden sein und ist nicht als wissenschaftlich komplette Abhandlung über das Thema Requirements Engineering in der agilen Softwareentwicklung gedacht. Es kann sequenziell oder auch auszugsweise gelesen werden. Als Autor ist es mir natürlich am liebsten, wenn Sie dieses Buch ständig an Ihrem Arbeitsplatz griffbereit haben und bei Fragen oder Unklarheiten zu Requirements-Engineering-Techniken oder -Vorgehensweisen darin einen schnellen Rat und brauchbare Infos finden.

In der Praxis hat man oft nicht die Zeit, beim Auftauchen einer Frage ein ganzes Buch z. B. zu Use Cases, User Stories oder Behaviour Driven Development zu lesen. Ich habe daher auf eine möglichst große Unabhängigkeit der einzelnen Themen und Kapitel geachtet, sodass das Buch auch als Nachschlagewerk in der täglichen Arbeit verwendet werden kann. Mit entsprechendem Vorwissen in agilen Methoden und Requirements Engineering kann es auch auszugsweise gelesen und verstanden werden. Für Praktiker müssen das Wichtigste und die Zusammenhänge in Kürze ersichtlich sein. Daher gibt es in vielen Kapiteln eine tabellarische Übersicht über wesentliche Punkte und verschiedene Überblicksgrafiken, die die Zusammenhänge auf einen Blick darstellen.

Auch wenn in vielen Kapiteln Begriffe aus Scrum verwendet oder zitiert werden, so wurde darauf geachtet, die einzelnen Themenbereiche möglichst allgemeingültig zu halten. Das Buch adressiert nicht »Requirements Engineering in Scrum«, sondern behandelt generell das Thema Requirements Engineering im agilen Umfeld. Die beschriebenen Techniken und Methoden können auch in anderen agilen Vorgehensweisen angewendet werden.

Beispiele werden mit einem grauen Hintergrund versehen:

In diesem Buch werden verschiedene Beispiele und Formulierungen aus der Praxis und für die Praxis aufgeführt. Die meisten dieser Beispiele sind abgeleitet aus einem Projekt zum Thema »Zeiterfassung«. Es geht in diesem Beispielszenario darum, dass die Tagesarbeitszeiten inkl. der Pausen und Abwesenheitszeiten von Mitarbeitern eines Unternehmens erfasst, ausgewertet und verwaltet werden können und dass auch die Erfassung, Auswertung und Verwaltung von Projektarbeitszeiten durchgeführt werden können – also von einzelnen Zeitblöcken der Tagesarbeitszeit, die bestimmten Projekten zugeordnet werden.

1.1.1 Zielgruppen

Das Buch richtet sich an Product Owner, Produktmanager, Projektmanager, Softwareauftraggeber, Scrum Master, Entwicklungsleiter, Entwickler, Tester und alle anderen Personen, die sich mit nachhaltigem und systematischem Requirements Engineering und Requirements Management in der agilen Softwareentwicklung beschäftigen oder davon betroffen sind.

Folgende Zielgruppen werden primär durch den [IREB CPRE FL 2017] adressiert:

- Requirements Engineers, die sich mit agiler Entwicklung befassen und ihre Techniken in dieser Umgebung erfolgreich anwenden möchten.
- Requirements Engineers, die etablierte Konzepte und Techniken agiler Ansätze anwenden und ihre Requirements-Engineering-Prozesse verbessern möchten.
- Fachkräfte für agile Entwicklungsprozesse, die die Werte und Vorteile des Requirements Engineering in agilen Projekten verstehen möchten.
- Fachkräfte für agile Entwicklungsprozesse, die die agile Entwicklung durch bewährte Requirements-Engineering-Techniken und -Methoden verbessern möchten.
- Personen aus verwandten Disziplinen – IT-Manager, Tester, Entwickler, Architekten und andere Vertreter im Bereich der Entwicklung (überwiegend, aber nicht ausschließlich Softwareentwicklung) –, die verstehen möchten, wie sie Requirements-Engineering- und agile Ansätze in Entwicklungsprozessen erfolgreich kombinieren können.

1.1.2 Abbildung des Lehrplans IREB RE@Agile

[IREB RE@Agile FL 2017] LE 1.3

Seit 2017 gibt es einen neuen Lehrplan des International Requirements Engineering Board (IREB®), der die Lücke zwischen Requirements-Engineering-Methoden und agilen Methoden schließt.

Einerseits wird die Sicht des IREB-Standards auf agile Werte abgebildet und andererseits wird eine agile Sicht auf die Werte des Requirements Engineering dargestellt. Zum Inhalt des Lehrplans IREB RE@Agile gehören Klassifizierung und Beurteilung von Requirements-Engineering-Artefakten und -Techniken im Zusammenhang mit Agilität, agilen Artefakten und Techniken, Requirements Engineering und wesentlichen Prozesselementen in der agilen Produktentwicklung. RE@Agile zeigt die Motivation für die Verwendung agiler Methoden in Entwicklungsprozessen auf und betont die Synergie zwischen Requirements Engineering und Agilität.

Da dieses Buch viele ergänzende und vertiefende Inhalte umfasst, die nicht im Lehrplan IREB RE@Agile enthalten sind, ist in einigen Bereichen eine andere Strukturierung gegeben. Nachfolgend ist ein erster Überblick und einige Hinweise angeführt, wo in diesem Buch die wichtigsten Inhalte des Lehrplans zu finden sind:

Lehrplan RE@Agile	in diesem Buch
LE 1 Motivation und Mindset	■ Abschnitt 1.2 »Verbindung zwischen Requirements Engineering und agilem Vorgehen« und Abschnitt 2.1 »Methodenüberblick«
LE 2 Grundlagen von RE@Agile	■ Kapitel 2 »Grundlagen«
LE 3 Artefakte und Techniken in RE@Agile	■ Kapitel 3 »Requirements-Ermittlung und -Dokumentation« (enthält das Kapitel Artefakte LE 3.1 sowie die Techniken LE 3.2.1 und 3.2.2) ■ Kapitel 4 »Requirements-Validierung und -Abstimmung« (LE 3.2.3) ■ Kapitel 6 »Requirements Management« (LE 3.2.4) ■ Einige Aspekte aus LE3 sind auch in Kapitel 8 »Requirements-Engineering-Rollen« enthalten.
LE 4 Organisatorische Aspekte von RE@Agile	■ Kapitel 7 »Organisatorische Aspekte«
–	■ Die Kapitel 8 »Requirements-Engineering-Rollen« und 9 »Rechtliche Themen« sind Ergänzungen in diesem Buch, die detailliertere Aspekte beschreiben bzw. über den Lehrplan hinaus gehen.

Die für den Lehrplan [IREB RE@Agile FL 2017] relevanten Kapitel und Stellen werden mit »[IREB RE@Agile FL 2017]« sowie ggf. mit der Ergänzung »**LE xxx**« gekennzeichnet und sind prüfungsrelevant. LE xxx referenziert auf die jeweilige Lerneinheit (LE bzw. EU = »Educational Unit«) des Lehrplans. Alle anderen Kapitel sind ergänzende Informationen und Hinweise oder Anregungen für die Praxis.

Wenn die Bezeichnung »RE@Agile« als Kurzbegriff verwendet wird, so bezieht sich dies immer auf den Lehrplan [IREB RE@Agile FL 2017].

Weiterführende Literatur zum Thema Requirements und agile Vorgehensweisen ist im Literaturverzeichnis im Anhang des Buches zu finden.

1.1.3 Allgemeine Begriffseinordnung

[IREB RE@Agile FL 2017]

Das umfangreiche Wissen in der Softwarebranche wird in unterschiedlichen Abstraktionsebenen beschrieben: [Meyer 2014] unterscheidet zwischen Prinzipien, Praktiken und Techniken. Diese Ebenen werden durch die Begriffe »präskriptiv« (vorschreibend), »abstrakt« und »falsifizierbar« (widerlegbar) unterschieden (siehe auch das Glossar im Anhang des Buches).

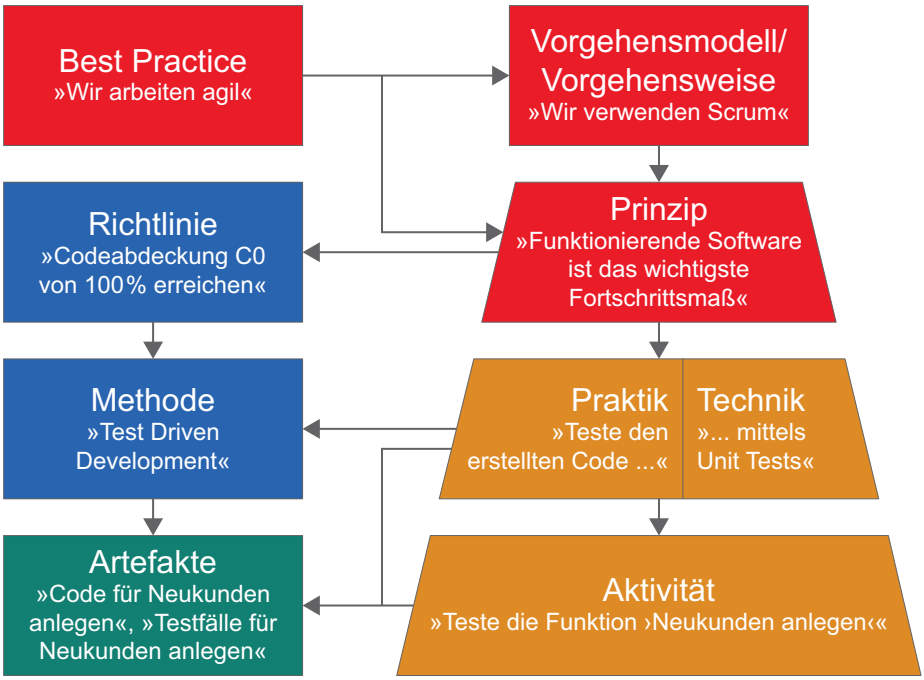

Abb. 1–1 *Abstraktionsebenen beim Wissen über die Softwareentwicklung*

- Ein **Prinzip** ist z. B. »Prüfe die Requirements vor der Implementierung auf ausreichende Qualität«. Dies ist präskriptiv, da eine Aktion vorgegeben wird. Und es ist abstrakt, da keine konkreten Praktiken oder Techniken, die verwendet werden sollen, vorgegeben werden. Diesem Prinzip kann in bestimmten Situationen auch widersprochen werden. Zum Beispiel könnte es in einer Forschungssituation sinnvoll sein, ohne vorherige Requirements-Spezifikation mehrere Prototypen zu erstellen und diese gegeneinander zu validieren. Dieses Prinzip ist somit auch falsifizierbar.

  Die Kenntnis der Prinzipien ist wichtig, um bewusste Entscheidungen treffen zu können. Die Falsifizierbarkeit ist wichtig, um Diskussionen und Nachdenken über Prinzipien und auch Praktiken in Gang zu bringen und zu entscheiden, ob diese in einem bestimmten Kontext angewendet werden sollen.

- Eine **Praktik** (in [IREB RE@Agile FL 2017] auch als »Verfahren« bezeichnet) ist ebenfalls präskriptiv, jedoch nicht so abstrakt, sondern eine konkretere »Instanz« des Prinzips in einem bestimmten Kontext, die Hinweise auf die Umsetzung des Prinzips gibt, ohne jedoch die tatsächliche Durchführung einer Aktivität zu beschreiben. Zum Beispiel »Prüfe jedes Backlog Item gegen die DoR (Definition of Ready), bevor es in den Sprint Backlog eingefügt wird« ist eine allgemeine Praktik, die schon konkret bestimmte Artefakte (Backlog Items, DoR und Sprint Backlog) zur Verwendung anspricht (die

Begriffe werden in den Kapiteln dieses Buches erläutert – siehe auch den Index im Anhang des Buches).

Abhängig von der jeweiligen Situation und Kontext können auch jeweils unterschiedliche Praktiken zur Erfüllung eines Prinzips angewendet werden.

- Eine **Aktivität** ist die konkrete Umsetzung einer Praktik bzw. Technik wie z.B. »Teste die Funktion ›Neukunden anlegen‹«.

Sowohl der Lehrplan [IREB RE@Agile FL 2017] als auch dieses Buch mit seinen erweiterten Ausführungen und Inhalten vermitteln Wissen von Requirements Engineering im Kontext der agilen Vorgehensweisen und stellen diese Abstraktionselemente in sachlicher Form vor bzw. regen zum Nachdenken und zur Diskussion über deren Anwendbarkeit an.

1.2 Verbindung zwischen Requirements Engineering und agilem Vorgehen

[IREB RE@Agile FL 2017] LE 1.1

Heute ist die Software und IT ein »Enabler« und Treiber in vielen Bereichen. Viele über lange Jahre etablierte plangetriebene Entwicklungsmethoden wurden nicht für ein sich schnell änderndes Umfeld ausgelegt. Agile Methoden (siehe auch Abschnitt 2.1) haben sich hier mittlerweile etabliert und schließen diese Lücke.

»Agile« oder »Agilität« wird in [Sheppard & Young 2006] wie folgt definiert: *»A rapid whole-body movement with change of velocity or direction in response to a stimulus.«*

Die Motivation für die Verwendung agiler Methoden ist hier definiert als Reaktion (z.B. eine schnelle Veränderung der Ausrichtung oder der Umsetzungsgeschwindigkeit) auf eine Stimulation (z.B. aus dem Projektumfeld).

Wenn man sich das Agile Manifest [Agile Manifesto] und die agilen Methodenbeschreibungen durchliest, dann geht es darin um mehr als nur Schnelligkeit. Im Grunde fokussieren alle agilen Prinzipien und Vorgehensweisen auf eine Lieferung von für den Kunden nützlichen Ergebnissen in kurzen Intervallen.

Agile Methoden oder Agilität sind jedoch kein Selbstzweck. Nicht in allen Situationen passt dieses Vorgehen gleich gut. Wenn z.B. längerfristige Planung, Vorhersagbarkeit oder Nachvollziehbarkeit erforderlich ist, müssen ggf. andere Vorgehensweisen oder Anpassungen der agilen Methoden angewendet werden.

Es ist daher wichtig, im Vorfeld darüber nachzudenken und ein passendes bzw. angepasstes Entwicklungsvorgehen zu wählen. Dies kann grundsätzlich auch dazu führen, dass in unterschiedlichen Teams entsprechend dem jeweiligen Bedarf unterschiedliche Vorgehensweisen oder auch Kombinationen aus agilen und plangetriebenen Vorgehensweisen angewendet werden (z.B. wenn im Maschinenbau

in der Hardwareentwicklung plangetriebenes Vorgehen und in der dazugehörenden Softwareentwicklung agiles Vorgehen in Kombination angewendet werden). Gartner hat dafür den Begriff »bimodale IT« geprägt. Dieses angepasste pragmatische Vorgehen ist auch einer der wesentlichen Erfolgsfaktoren der Softwareentwicklung und IT in dem heute sehr dynamischen Umfeld.

1.2.1 Mindsets und Werte im Requirements Engineering und agilem Vorgehen

[IREB RE@Agile FL 2017] LE 1.2

Requirements Engineering (RE)

In der IREB-Definition von Requirements Engineering im Lehrplan von 2017 sind Mindset und Werte von Requirements Engineering angegeben:

Requirements Engineering ist eine systematische und disziplinierte Herangehensweise zur Spezifikation und Management von Anforderungen mit den folgenden Zielen:

1. Die relevanten Anforderungen kennen, einen Konsens zwischen den Stakeholdern dieser Anforderungen erreichen, konform zu vorgegebenen Standards dokumentieren und die Anforderungen systematisch managen.
2. Die Wünsche und Bedürfnisse der Stakeholder verstehen und dokumentieren.
3. Die Anforderungen spezifizieren und managen, um ein System auszuliefern, das möglichst weitgehend den Wünschen und Bedürfnissen der Stakeholder entspricht.

Im IREB CPRE FL sind die vier Hauptaktivitäten des Requirements Engineering definiert:

- Ermittlung,
- Dokumentation,
- Prüfen und Abstimmen und
- Verwalten von Anforderungen.

Diese bilden keinen Prozess oder eine vorgegebene Folge von Aktivitäten, sondern stellen einen prozessunabhängigen Ansatz dar. Es sind Aktivitäten, die unabhängig von der gewählten Vorgehensweise durchgeführt werden sollen. In jedem dieser Aktivitäten des Requirements Engineering gibt es verschiedene Methoden und Techniken, die in allen Vorgehensweisen – sowohl in nicht agilen als auch in agilen – angewendet werden können. Dabei gibt es natürlich Unterschiede in den Detailausprägungen der angewandten Methoden und Techniken.

Zu beachten ist auch, dass im Requirements Engineering grundsätzlich von »System« gesprochen wird und nicht von Software, Produkt, Hardware, Prozess

oder sonstigen spezifischen Begriffen. »System« umfasst all diese Begriffe und bedeutet, dass dieses System eine Gruppe von Teilen oder Elementen umfasst, die in einer Umgebung (»Systemkontext«) zusammenwirken.

Agiles Umfeld

Der Ausdruck »agile Methoden« referenziert auf ein umfangreiches Set an Vorgehensweisen, die sich im Umfeld von »Agile« entwickelt haben (siehe auch Kap. 2 »Grundlagen«). Für andere Entwicklungsmethoden wird auch der Ausdruck »plangetrieben« oder »nicht agile Methoden« verwendet.

Die verschiedenen Methoden sind in unterschiedlichem Kontext unterschiedlich wirksam und haben jeweils Anwendungsbereiche, in denen sie sinnvoll und wertschöpfend anzuwenden sind.

Das Mindset von »Agile« ist im »Agilen Manifest« [Agile Manifesto] und den darauf basierenden zwölf Prinzipien definiert. Im Februar 2001 traf sich eine Gruppe von 17 Personen in den Bergen von Utah, um über die Gemeinsamkeiten von verschiedenen Ansätzen der Softwareentwicklung zu diskutieren. Diese Gruppe nannte sich »The Agile Alliance« [Agile Alliance] und das Ergebnis dieses Meetings waren der Begriff »agil« und das »Agile Manifest« [Agile Manifesto].

Das Agile Manifest besteht aus vier Leitsätzen und zwölf Prinzipien, die die Leitsätze erläutern. Es ist das Fundament für agiles Arbeiten und damit für alle agilen Methoden.

Manifest für agile Softwareentwicklung:

Wir erschließen bessere Wege, Software zu entwickeln, indem wir es selbst tun und anderen dabei helfen.

Durch diese Tätigkeit haben wir diese Werte zu schätzen gelernt:

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

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

Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas

→

Die zwölf Prinzipien hinter dem Agilen Manifest:

- Unsere höchste Priorität ist es, den Kunden durch frühe und kontinuierliche Auslieferung wertvoller Software zufrieden zu stellen.
- Heiße Anforderungsänderungen selbst spät in der Entwicklung willkommen. Agile Prozesse nutzen Veränderungen zum Wettbewerbsvorteil des Kunden.
- Liefere funktionierende Software regelmäßig innerhalb weniger Wochen oder Monate und bevorzuge dabei die kürzere Zeitspanne.
- Fachexperten und Entwickler müssen während des Projektes täglich zusammenarbeiten.
- Errichte Projekte rund um motivierte Individuen. Gib ihnen das Umfeld und die Unterstützung, die sie benötigen, und vertraue darauf, dass sie die Aufgabe erledigen.
- Die effizienteste und effektivste Methode, Informationen an und innerhalb eines Entwicklungsteams zu übermitteln, ist im Gespräch von Angesicht zu Angesicht.
- Funktionierende Software ist das wichtigste Fortschrittsmaß.
- Agile Prozesse fördern nachhaltige Entwicklung. Die Auftraggeber, Entwickler und Benutzer sollten ein gleichmäßiges Tempo auf unbegrenzte Zeit halten können.
- Ständiges Augenmerk auf technische Exzellenz und gutes Design fördert Agilität.
- Einfachheit – die Kunst, die Menge nicht getaner Arbeit zu maximieren – ist essenziell.
- Die besten Architekturen, Anforderungen und Entwürfe entstehen durch selbstorganisierte Teams.
- In regelmäßigen Abständen reflektiert das Team, wie es effektiver werden kann, und passt sein Verhalten entsprechend an.

Das Agile Manifest ist ein Meilenstein in der Geschichte der Softwareentwicklung. Es gab bisher nur wenig Literatur, die so große Auswirkungen auf die Softwareentwicklung hatte wie dieses kurze Dokument. Heute kann man sagen, dass das Agile Manifest und seine Prinzipien praktisch alle Bereiche der Softwareentwicklung erreicht haben. Auf Basis des Agilen Manifests haben sich verschiedene Vorgehensweisen etabliert wie z.B. Scrum, Kanban, Extreme Programming, Feature Driven Development oder Test Driven Development (TDD).

Bei den agilen Vorgehensweisen stehen das Ergebnis und der Wert für den Kunden im Vordergrund. Anstelle einer einmaligen umfassenden Vorausplanung wird eine ständig rollierende Planung in Verbindung mit schnellem Feedback durch kurze Iterationen zur Risikominimierung verwendet. Dies zielt darauf ab,

dass der Kunde möglichst rasch ein erstes Ergebnis ansehen und am besten auch gleich verwenden kann. Ein starker Fokus liegt auf Transparenz, Kommunikation und einem Team, das sich selbst steuert.

Neben diesen im Agilen Manifest und in Vorgehensmodellen wie Scrum explizit genannten Werten und Themen gibt es viele weitere Aspekte in der Softwareentwicklung, die in einem Projekt berücksichtigt werden müssen, auch wenn sie nicht durch die agilen Basiskonzepte explizit abgedeckt sind. Bereiche wie Risikomanagement, Testen und eben auch Requirements Engineering sind nach wie vor kritisch für den Erfolg von Softwareprojekten.

Für die erfolgreiche Einführung und den Einsatz agiler Vorgehensweisen ist es daher wichtig, dass einerseits eine Ausrichtung des Teams bzw. der Organisation an den agilen Werten und einer disziplinierten Anwendung der Regeln der von der Organisation gewählten agilen Vorgehensweise erfolgt, und andererseits auch alle anderen für den Erfolg wesentlichen Themen berücksichtigt werden, sodass diese in einer sinnvollen Art und Weise zu einem effektiven Framework für die Softwareentwicklung integriert werden.

Requirements Engineering im agilen Kontext

Die Aussagen, Methoden und Werte zwischen Requirements Engineering und agilen Vorgehensweisen stehen nicht im Widerspruch zueinander, auch wenn dies manchmal so vermutet wird oder dies aus manchen Grundprinzipien so interpretiert wird.

Requirements Engineering hat wie oben erwähnt den Fokus auf der systematischen Ermittlung, Spezifikation und dem Management von Anforderungen unabhängig vom Detaillierungsgrad und der Phase, in der man sich im Projekt befindet.

Agile Vorgehensweisen fokussieren primär auf ein lauffähiges Produkt und stellen Individuen und Kommunikation höher als einen Prozess und Tools. Sie bieten ein Mindset und Methoden in der Projektabwicklung und Teamsteuerung, ohne spezifisch auf das Thema Requirements einzugehen – so enthält z.B. der Scrum Guide [Schwaber & Sutherland 2017] nichts zum Thema Spezifikation von Anforderungen. Dies besagt nicht, dass in agilen Projekten keine Spezifikation erforderlich oder sinnvoll ist. Es wird ja im Scrum Guide auch nichts zu Risikomanagement, Versionsverwaltung, Konfigurationsmanagement, Continuous Integration und anderen Themen vorgegeben und doch ist es sinnvoll, auch in agilen Softwareprojekten Risikomanagement, Versionsverwaltung, Konfigurationsmanagement, Continuous Integration und auch noch weitere Themen durchzuführen.

Eine übertriebene bzw. dogmatische Betrachtung von Grundsätzen und Prinzipien führt typischerweise zu Konflikten. Es ist daher eine sehr eingeschränkte Sichtweise, wenn man meint, dass es durch Requirements Engineering möglich ist, ein vollständiges, konsistentes und abgestimmtes Requirements-Dokument zu

erstellen, das dann noch ohne künftige Änderungen umgesetzt wird. Dies ist bis auf ganz kleine Projekte in der Praxis nicht der Fall.

Eine ebenso eingeschränkte Sicht ist es, wenn man meint, dass durch agile Prinzipien ein Entwicklungsprojekt ohne irgendwelche vorbereitenden Arbeiten oder begleitenden Prozesse gestartet werden kann und ausschließlich durch regelmäßige Auslieferung von Software und Verbesserungen aufgrund des Feedbacks durch die Stakeholder erfolgreich implementiert werden kann. Auch hier gelingt dies nur unter »Laborbedingungen« und in sehr kleinen Projekten.

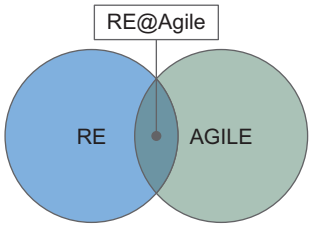

Abb. 1–2 *Mindset- und Werteüberschneidung zwischen Requirements Engineering und Agile (nach [IREB RE@Agile FL 2017])*

Requirements Engineering und die agilen Vorgehensweisen widersprechen sich nicht und überlappen sich nur in wenigen Punkten. Beiden gemeinsam ist, dass für den Kunden Wert in Form von lauffähiger, qualitativ guter Software geschaffen wird und der Kunde zufriedengestellt wird.

Agile Methoden helfen dabei, dass lauffähige Software effizient und schnell erstellt wird. Requirements Engineering stellt gute Techniken zur Verfügung, um die Bedürfnisse der Stakeholder zu verstehen, damit gute Software im Sinne der Stakeholder entwickelt wird.

Agilität macht die Softwareentwicklung effizient, Requirements-Entwicklung macht die Softwareentwicklung effektiv!

Unter diesem Gesichtspunkt liefert Requirements Engineering einen wesentlichen Beitrag für folgende agile Prinzipien:

- **1. Prinzip**
  Requirements Engineering erleichtert das Verstehen der Kundenwünsche und Bedürfnisse, um eine für den Kunden wertvolle Software zu entwickeln.
- **2. Prinzip**
  Requirements Engineering erleichtert durch passende Tools das Erkennen von Veränderungen beim Kunden bzw. im Markt, damit die Stakeholder einen Vorteil haben.

- **4. Prinzip**
  Requirements Engineering erleichtert durch passende Tools und Techniken die Zusammenarbeit zwischen den Stakeholdern und Entwicklern.
- **6. Prinzip**
  Requirements Engineering erleichtert durch passende Tools und Techniken die Kommunikation.
- **10. Prinzip**
  Requirements Engineering erleichtert das Verständnis der Stakeholder-Wünsche und -Bedürfnisse und hilft, die Entwicklung von nicht notwendiger Software zu minimieren.

Diese Überlappung und der Zusammenhang von Requirements Engineering und Agile ist der Bereich, der in diesem Buch und im Lehrplan [IREB RE@Agile FL 2017] adressiert wird.

Hinweis: Im Lehrplan RE@Agile wird der Begriff »RE@Agile« über den Begriff »agiles Requirements Engineering« gestellt, um damit klarzumachen, dass Requirements Engineering prozessunabhängig ist.

1.2.2 Requirements Engineering im Kontext des Agilen Manifests

Im Agilen Manifest selbst steht wenig über Requirements Engineering. Trotzdem haben seine Leitsätze großen Einfluss darauf, wie ein agiles Team mit Anforderungen umgeht:

- **Individuen und Interaktionen zählen mehr als Prozesse und Werkzeuge**
  Aus Sicht des Requirements Engineering bezieht sich diese Aussage auf die Interaktion und Kommunikation zur Ermittlung und Abstimmung von Anforderungen sowie auf den Prozess und die Tools zu deren Verwaltung und Dokumentation.

  Dass die Interaktion und Kommunikation der Beteiligten und Betroffenen der wichtigste Faktor im Requirements Engineering ist, ist seit Langem bekannt. Wichtig ist es aber auch, die Aussagen der Individuen und die Ergebnisse im Rahmen der Interaktion auf angemessene Art und Weise zu dokumentieren.

  Eine große Herausforderung in der Praxis ist die Definition des Requirements-Engineering-Prozesses. Dieser wird oft zu formal und sequenziell beschrieben. Um die Agilität zu gewährleisten, beschreibt man bei der Prozessdefinition nicht den Prozess insgesamt, sondern nur die einzelnen Teilprozesse und Techniken, wie z.B. Erhebungstechniken, Darstellungstechniken, einzelne Artefakte, Analysemethoden und Managementtechniken, und stellt dies dann zu einem modularen, flexibel einsetzbaren Methodenbaukasten zusammen. In Kapitel 4 ist ein Teil eines solchen Baukastens zu finden.

Flexible Requirements-Werkzeuge helfen, das Requirements Engineering agil umzusetzen. Vor allem auch die Integration mit anderen Werkzeugen und das Vermeiden von Brüchen im Entwicklungsprozess sind essenziell für das effiziente Funktionieren des Requirements-Prozesses. Microsoft Word und Excel als die am meisten verwendeten Requirements-Werkzeuge sind nicht gerade dazu geeignet, das Requirements Engineering nachhaltig und effizient zu gestalten. Aber auch »echte« Requirements-Management-Werkzeuge haben oft zu starre Prozesse implementiert oder ergehen sich in einer Attributierungsorgie. Die Integration mit anderen Werkzeugen im Entwicklungsprozess ist ebenfalls oft mangelhaft. Insofern ist die Werkzeugfrage auch im agilen Umfeld sehr wichtig.

Funktionierende Software zählt mehr als umfassende Dokumentation

[IREB RE@Agile FL 2017] LE 1.2

Diese Darstellung führt immer wieder zu der Meinung, dass Dokumentation in agilen Projekten generell nicht befürwortet wird. Das ist eine falsche Interpretation, denn Dokumentation, die einen klaren Zweck und Nutzen für den Kunden hat, wird auch in agilen Methoden befürwortet. Leider wurden und werden in vielen Projekten oft Dokumente erstellt, die keinen klaren Zweck oder Nutzen haben.

Die fertige Software ist natürlich wichtiger, als sich zu stark auf die Dokumentation zu fokussieren. »Dokumentation« kann jegliche Art von Anforderungsspezifikation, Architekturdokumenten, Designvorgaben, Prozessdefinitionen, Benutzerdokumentation, Entwicklerkommentaren im Sourcecode etc. sein.

Eine interessante Erkenntnis ergibt sich, wenn man die Kosten der Erstellung einer funktionierenden Software inkl. Sourcecode im Vergleich zu den Dokumentationskosten inkl. Sourcecode-Kommentare in Abhängigkeit vom Dokumentationsgrad betrachtet. Dabei kommt man zu folgendem Ergebnis: Eine umfassende Dokumentation vor der Erstellung der ersten Codezeile macht aus Zeit- und Kostengründen keinen Sinn. Auf die Dokumentation komplett zu verzichten, ist aber ebenfalls keine gute Lösung. Die Kommunikationskosten und die Kosten für zusätzliche unnötige Iterationen werden sonst wegen vergessener Informationen, Spätfolgen durch fehlendes Verständnis für Umsetzungsentscheidungen und fehlende Nachvollziehbarkeit ebenfalls sehr hoch sein (siehe Abb. 1–3).

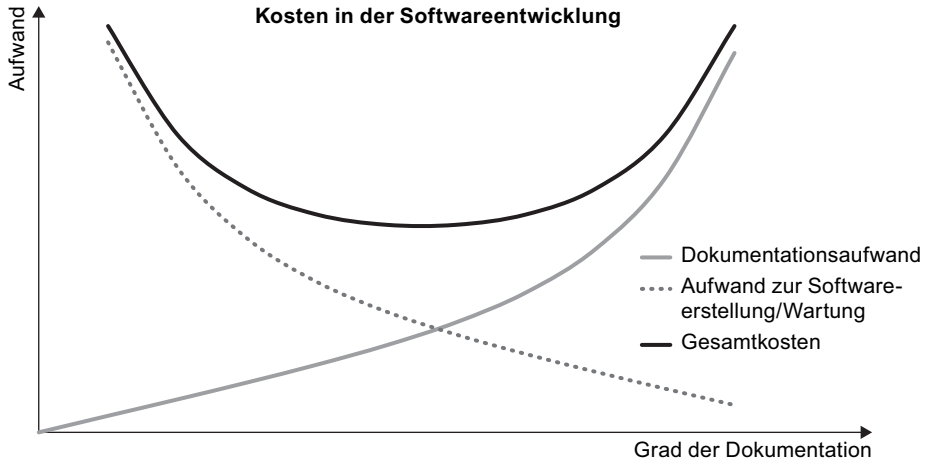

Abb. 1–3 *Kostenkurve in der Softwareentwicklung abhängig vom Dokumentationsgrad*

Jede Dokumentation unterstützt bis zu einem gewissen Grad die Flexibilität und die Effizienz und bremst sie ab einem gewissen (übertriebenen) Grad.

Es ist daher im Sinne der Flexibilität und Effizienz bei jeder Dokumentation zu überlegen, was der für das jeweilige Projekt optimale Grad ist. Nachfolgend sind beispielhaft einige zentrale Dokumentationsarten angeführt, für die dies gut überlegt werden sollte:

- Requirements
- Risikoanalysen
- Spikes[1] und Machbarkeitsanalysen
- Architekturspezifikation
- Codekommentare
- Testspezifikation und Testdokumentation
- Benutzerdokumentation
- Prozessdefinition
- Methodenbeschreibungen, Guidelines und Checklisten

Dokumentation, speziell Anforderungen, komplett wegzulassen, verursacht zusätzliche Kosten im Projekt! Den richtigen Grad an Dokumentation zu treffen, ist die Kunst guter agiler Projektabwicklung.

1. Eine Definition des Begriffs Spike ist in Abschnitt 3.8.1 zu finden.

- **Zusammenarbeit mit Kunden zählt mehr als Vertragsverhandlung**
  Hier geht es um externe Kunden-Lieferanten-Verhältnisse, die typischerweise auf einem Vertrag – welcher Art auch immer – basieren. Die Interpretation des Agilen Manifests ist in diesem Kontext sehr stark von der jeweiligen Projektsituation abhängig. In Projekten, in denen der Kunde selbst nicht genau weiß, was er will, muss die Zusammenarbeit mit dem Kunden sehr intensiv sein. In Projekten, in denen schon ziemlich klar ist, was der Kunde benötigt und dies auch schon vorab gemeinsam definiert wurde, kann die Zusammenarbeit weniger intensiv sein.

  Die Intensität der Zusammenarbeit wird auch von der Grundhaltung zwischen Kunde und Lieferant stark beeinflusst. Bei Kunden, die partnerschaftlich agieren und zu denen man als Lieferant aufgrund langjähriger Beziehungen großes Vertrauen hat, werden auch kaum intensive Vertragsverhandlungen nötig sein. Umgekehrt ist es genauso. Partnerschaftliche Lieferanten werden im Sinne des Kunden denken und handeln. Ist dies jedoch nicht der Fall, so sind auch in agilen Projekten gute Verträge im Sinne der Risikominimierung und wechselseitigen Klarheit sehr wichtig. Wichtig ist in allen Fällen, dass der Vertrag und die Spezifikation aufeinander abgestimmt sind (siehe auch Kap. 9).
- **Reagieren auf Veränderung zählt mehr als das Befolgen eines Plans**
  Änderungen sind die Normalität in Projekten. Wenn der Kundennutzen durch eine Änderung größer wird, ist eine Änderung auch zu befürworten. Es gibt jedoch Änderungen, die zwar vom Kunden gewünscht werden, die aber den Gesamtkundennutzen des Systems reduzieren. In solchen Fällen ist es besser, dem Kunden die bisher gewählte Umsetzung und die Konsequenzen der Änderung zu erklären und den ursprünglichen Plan weiterzuverfolgen.

Beispiel:

Ein Kundenvertreter wünscht sich nach Sichtung des ersten Prototyps, der von den Entwicklern selbstständig mit einfachen Bedienelementen umgesetzt wurde, nun in einer Webmaske eine Drag&Drop-Funktion. Dies wäre jedoch nur sehr aufwendig mit den zur Verfügung stehenden Technologien umzusetzen. Außerdem würde dadurch der Teil seiner Systembenutzer ausgeschlossen, der eine ältere Browserversion verwendet oder auf mobilen Geräten arbeitet. Die Änderung nicht umzusetzen ist in diesem Fall die bessere Alternative für den Kunden. Sie stellt in der Bedienung nur unwesentlich mehr Aufwand dar, spart jedoch die hohen Zusatzkosten der Implementierungsänderung und berücksichtigt außerdem alle potenziellen Benutzergruppen. Für den einzelnen Kundenvertreter wäre die Änderung aus seiner Sicht durchaus mit einem Nutzen verbunden. Für den Kunden insgesamt und für den längerfristigen Nutzen aus dem System hätte sie jedoch nachteilige Auswirkungen.

In allen Projekten sollen Änderungen systematisch analysiert und bewertet werden, bevor sie akzeptiert und in das Backlog eingefügt werden. Der Kunde sollte auch immer eine Rückmeldung erhalten, ob und warum die Änderung von den anderen Beteiligten (Entwickler, IT-Betrieb, Marketing etc.) als Nutzen steigernd oder Nutzen verringernd eingeschätzt wird. Er kann ja auch im nicht sinnvollen Fall die Änderung trotzdem durchführen lassen – es ist ja schließlich sein Geld, das er dann mehr ausgibt! Diese Tatsache sollte ihm aber bewusst sein.

In der Praxis gibt es kaum Projekte, bei denen Zeit und Geld keine Rolle spielen und die Entwickler ohne Vorgaben einfach drauflosentwickeln oder Änderungen durchführen können. Es besteht daher in jedem Projekt die Notwendigkeit, zu Beginn zumindest eine grobe Planung und Spezifikation zu erstellen. Die Releaseplanung und das Product Backlog sind genau solche Pläne (siehe Abschnitte 6.3.4 und 6.4.1). So kann man erkennen und nicht nur ahnen, ob bzw. wie weit man vom ursprünglich vorgesehenen Ziel und Rahmen abweicht.

Es gilt jedoch auch hier wieder das schon oben erwähnte Prinzip: Nur wenn die Kosten für die Erstellung und Pflege des Plans geringer sind als die Kosten, die sich durch eine unsystematische und schlecht abgestimmte Softwareentwicklung ohne Plan ergeben, sollte ein Plan erstellt und gewartet werden.

Dass Pläne sich ändern, ist ein Faktum. Gute Planer wissen dies und sehen in Änderungen auch nichts Negatives und schon gar kein Versagen ihrer planerischen Fähigkeiten.

Laufende Änderungen sind ein Naturgesetz in der Softwarewelt, das der gute Planer akzeptiert und berücksichtigt.

Ein Grundsatz der Planung lautet: Entweder gut oder gar nicht! Wobei hier mit guter Planung nicht gemeint ist, dass das System vorab bis in die letzte Zeile Quellcode vorauszuplanen ist, sondern es geht hier darum, die Wegpunkte auf der Reise zum Ziel festzulegen, an denen man sich immer wieder orientieren kann. Wenn aber ein Plan erstellt wird, dann muss er auch bis zum Ende laufend gepflegt, angepasst und konsistent gehalten werden. Wenn hier Nachlässigkeit aufkommt und der Plan von den Beteiligten als nicht konsistent angesehen wird, dann ist er nutzlos.

1.2.3 Nutzen von Requirements Engineering im agilen Vorgehen

[IREB RE@Agile FL 2017] LE 1.4.1

- **Übergabeaufwände im Team werden reduziert**
  Das agile Team soll alle Fähigkeiten besitzen, die für die Erlangung der definierten Ziele notwendig sind. Dazu gehören neben den Entwicklungs- und Qualitätssicherungskompetenzen auch Fähigkeiten zur Durchführung von Requirements-Engineering-Aufgaben. Durch die direkte Kommunikation im agilen Team erfolgen große Teile des Requirements Engineering und das Beseitigen von Unklarheiten auf direkter Ebene zwischen den Teammitgliedern, wodurch Übergabeaufwände reduziert werden, und es wird vermieden, im Voraus eine umfassende Spezifikation der Anforderungen zu erstellen.
- **Laufende Optimierung bestehender Anforderungen**
  Das iterativ inkrementelle Vorgehen im agilen Kontext mit seinen regelmäßigen Feedbackschleifen fördert die laufende Anpassung der Ideen und Anforderungen. Bei jeder Sprint-Planung oder jedem Refinement, bei jeder Sprint-Retrospektive und auch zwischendurch wird die Diskussion über die bestehenden Anforderungen durch die Kommunikation in den Iterationen gefördert. Dies kann z. B. nur als grobe Idee oder schon als detailliertere Ausarbeitung in Form von Artefakten stattfinden, wie z. B. einem Prozessmodell, einer User Story, einem Epic, einem Screendesign oder einer Ablaufbeschreibung (siehe auch Kap. 3). Dies führt einerseits zu einer passenden Detailliertheit der Requirements entsprechend dem jeweiligen Bedarf und andererseits auch zu besserer Qualität.
- **Anforderungen reifen und werden validiert**
  In Scrum (siehe Abschnitt 2.1.2) werden Requirements regelmäßig in den Refinement-Meetings weiterentwickelt und durch die Beteiligten hinterfragt und überprüft. Um in das Backlog einer Iteration aufgenommen zu werden, müssen die Requirements den vorab in der Definition of Ready (siehe Abschnitt 5.2) festgelegten Qualitätskriterien entsprechen.
- **Verändernde Anforderungen werden durch kontinuierliches Requirements Engineering erkannt und unterstützt**
  Sowohl im agilen Vorgehen als auch im Requirements Engineering stehen die Wünsche und Bedürfnisse der Stakeholder im Fokus. Das agile Vorgehen und dessen Prinzipien unterstützen, fördern und systematisieren die laufende Anpassung und Veränderung der Sichtweise der Stakeholder. Requirements Engineering ist kein einmaliger Prozess, sondern eine Aktivität, die sowohl im agilen als auch im nicht agilen Vorgehen kontinuierlich durchgeführt werden soll und dadurch diese verändernden Anforderungen und Bedürfnisse der Stakeholder erkennen und entsprechend abbilden soll.

- **Definition des initialen Product Backlogs**
  In agilen Projekten muss zu Beginn ein initiales Product Backlog erstellt werden (siehe auch Abschnitt 6.4.1). Dies erfolgt oft intuitiv. Die Praxis des Requirements Engineering bietet aber ein gutes Set an Techniken und Methoden, um hier effektiver und effizienter vorzugehen und die Erstellung des initialen Backlogs zu erleichtern. Dies kann für jeden einzelnen Eintrag bzw. jede Anforderung im Backlog auch auf unterschiedlichen Abstraktions- und Spezifikationsebenen passieren (siehe Abschnitte 3.2 bis 3.6). Es geht dabei nicht um eine komplett vollständige Spezifikation, sondern um ein ausreichendes Verständnis für das umzusetzende System, den Kontext und die Wünsche der Stakeholder.

1.2.4 Vorurteile und Probleme beim Requirements Engineering im agilen Umfeld

[IREB RE@Agile FL 2017] LE 1.4.2

Manchmal haben sich aufgrund der früher oft zu einseitigen Betrachtungen sowohl im Requirements Engineering als auch im agilen Umfeld einige Gedankenmuster und Vorurteile festgesetzt, die irreführend sind:

- **Requirements Engineering ist immer vorgelagert (upfront)**
  Es gibt Leute, die denken heute noch, dass Requirements Engineering *nur* als vorgelagerte Tätigkeit stattfindet und die ganzen Spezifikationstätigkeiten am Beginn eines Projektes erfolgen.

  Tatsächlich ist Requirements Engineering jedoch unabhängig vom gewählten Vorgehensmodell und unabhängig von den Phasen im Prozess eine Aktivität, die ständig stattfindet. Natürlich gibt es abhängig vom Vorgehensmodell unterschiedliche Schwerpunkte (siehe auch Abschnitt 2.4), aber es ist eine Tatsache, dass Requirements Engineering – ob nun explizit definiert oder implizit – in der Praxis vom Beginn bis zum Abschluss in allen Phasen und Zeiten eines Projektes geschieht.

  Requirements Engineering ist grundsätzlich eine prozessunabhängige Disziplin, die sowohl in nicht agilen als auch in agilen Vorgehensweisen gleichermaßen angewendet werden kann. Sie kann bzw. sollte auch jeweils an die Erfordernisse und den Bedarf angepasst werden, wie dies ja typischerweise auch beim Programmieren, Testen etc. gemacht wird.
- **Vorgelagert ist schlecht**
  Vor einem Vorhaben oder Projekt darüber nachzudenken, wie man es angehen möchte und was die Anforderungen und Randbedingungen sind, ist jedenfalls unabhängig von der gewählten Vorgehensweise zu empfehlen und hat sich bewährt.

Dies bedeutet nicht, dass dies sehr lange dauern muss oder bestimmte Dokumente in bestimmten Phasen in einem großen Umfang erstellt werden müssen.

Das Agile Manifest und die agilen Prinzipien sagen nicht, dass vorausschauendes Denken schlecht ist und sie sollten auch nicht so interpretiert werden. Es gibt genügend Methoden und Praktiken, die sich im agilen Umfeld entwickelt haben und vorausschauendes Denken unterstützen, wie z.B. Story Maps (siehe Abschnitt 6.4.2) oder Test Driven Development (siehe die Abschnitte 2.1.3 und A.2 im Anhang). Test Driven Development ist überhaupt eine der restriktivsten Vorabspezifikationstechniken, bei der zuerst die Testfälle spezifiziert bzw. sogar programmiert werden – was im Grunde nichts anderes ist als eine sehr detaillierte Requirements-Spezifikation – und danach erst der eigentliche Produktcode programmiert wird.

Gute und effizient durchgeführte Projekte nehmen bei der Wahl der Methoden und Techniken und deren sinnvollem Anwendungszeitpunkt immer Rücksicht auf den jeweiligen Projektkontext und benutzen vor allem auch den gesunden Menschenverstand, um zu beurteilen, welche Methode zu welchem Zeitpunkt gerade passend ist oder auch nicht.

Zumindest in diesem einen Punkt ist es im Sinne eines guten Projektes daher zwingend erforderlich, vorab nachzudenken, welche Methoden und Vorgehensweise man im Projektkontext anwenden soll.

- **Requirements Engineering = Dokumente bzw. Dokumentation**
  Leider wird unter dem Begriff Requirements Engineering oft nur das Erstellen von (umfangreichen) Anforderungsdokumenten verstanden. Diese Sichtweise greift deutlich zu kurz und macht nicht bewusst, was Requirements Engineering eigentlich bedeutet und umfasst. Durch die Aktivitäten des Requirements Engineering, die primär auf die Kommunikation mit dem Kunden und das Schaffen von Wissen fokussieren, *können* (nicht *müssen*) Dokumente als Ergebnis produziert werden. Dies ist dann sinnvoll, wenn es für das weitere Vorhaben nützlich ist (z.B. für Rechtskonformität, Erhalt von Informationen oder Unterstützung der Kommunikation und des Nachdenkens). Gutes Requirements Engineering wird in dem Wissen durchgeführt, dass auch bei noch so großem Bemühen und Aufwand die Dokumente nicht vollständig sein werden. Der notwendige oder auch noch akzeptable Grad der Unvollständigkeit wird wiederum durch den Projektkontext bestimmt.

- **User Stories sind ausreichend für das Requirements Engineering im agilen Umfeld**
  User Stories (siehe Abschnitt 3.4.2) sind im agilen Umfeld die beliebteste Spezifikationstechnik. Es ist jedoch unzureichend, wenn man sich beim Erfassen der Kundenanforderungen alleine auf User Stories fokussiert. User Stories zu erstellen ist nur eine von vielen Spezifikationsarten und deckt nur eine Ebene und Sichtweise ab. Um ein möglichst realistisches Verständnis von den Wünschen und der Sichtweise des Kunden zu bekommen und in weiterer Folge

auch eine passende Systemarchitektur und entsprechendes Systemdesign zu entwerfen und das passende lauffähige Produkt erstellen zu können, müssen noch weitere Abstraktions- und Beschreibungsebenen betrachtet werden (siehe Abschnitte 3.2 bis 3.6).

Vor allem die übergeordneten Zusammenhänge und die Prozesssicht gehen verloren, wenn man sich ausschließlich auf die Erstellung von User Stories konzentriert.

- **Dokumente sind wertlos, nur Code hat bleibenden Wert**
  In der Vergangenheit wurde durch verschiedene Projekte, in denen Spezifikationen in dicke unlesbare Textdokumente ausgeartet sind, ein negatives Image der Spezifikationsdokumente erzeugt. Das bedeutet aber nicht, dass alle Spezifikationsdokumente wertlos sind. Zum richtigen Zeitpunkt, im angemessenen Umfang und in der passenden Art erstellte Dokumente (z.B. Anforderungsspezifikationen, Testdokumente, Risikoanalysen, Schulungsunterlagen, Architektur- und Designdokumente) sind für eine nachhaltige Softwareentwicklung jedenfalls notwendig.
- **Anforderungen können nur durch lauffähige Software validiert werden**
  Auch wenn im Agilen Manifest lauffähige Software über die Dokumentation gestellt wird, so bedeutet dies nicht, dass man nur aufgrund einer lauffähigen Software beurteilen kann, dass die Anforderungen valide umgesetzt werden. Es gibt verschiedene andere Möglichkeiten im Requirements Engineering, um auch ohne lauffähige Software und ohne eine einzige Zeile Code die Anforderungen des Kunden zu validieren, wie z.B. Wireframes, Sketches oder szenariobasierte Spezifikation der Benutzeroberfläche und des Verhaltens des Systems (siehe auch Abschnitt 3.5).

1.2.5 Fallstricke bei RE@Agile

[IREB RE@Agile FL 2017] LE 1.4.3

- **Requirement = nur eine Art bzw. Ebene von Information**
  In der Praxis erlebt man es oft, dass unter dem Begriff »Requirement« genau eine Art oder Ebene von Information verstanden wird. Man merkt dies schnell an Aussagen wie z.B. »Eine User Story ist kein Requirement«, »Ein Requirement ist eine viel genauere Spezifikation und verfeinert eine User Story« oder umgekehrt: »Ein Requirement wird mit einer User Story verfeinert«. Diese und ähnliche Aussagen zeigen große Lücken im Verständnis des Requirements Engineering und bedeuten, dass mit dem Begriff Requirement genau nur eine Ebene der Spezifikation verbunden wird.

  Dabei wird übersehen oder negiert, dass die Vision und Ziele auf hoher Ebene, die User Stories auf einer mittleren Detaillierungsebene sowie Use-Case-Szenarien oder sogar ausführbare Prototypen auf einer detaillierten

Ebene allesamt Requirements darstellen können, jedoch mit einem jeweils unterschiedlichen Detail- bzw. Abstraktionsgrad.

Je höher die Abstraktion, desto langlebiger und desto weniger Änderungen unterworfen sind die Requirements und deren Artefakte. In diesem Zusammenhang sind auch die Dokumentationsmittel und Tools wichtig, die passend zur jeweiligen Ebene und Requirements-Art gewählt werden sollten, damit das Arbeiten mit den Requirements auch effizient möglich wird.

- **Der Blick auf »das große Ganze« fehlt**
  Wenn man die agilen Grundsätze oder den Scrum Guide liest, sind die Texte (scheinbar) recht einfach und schnell verständlich. Oft wird dies fälschlicherweise auch so interpretiert, dass nur die Themen, die in diesen Beschreibungen erwähnt werden, umgesetzt werden müssen, um eine erfolgreiche Softwareentwicklung zu betreiben. Aus Sicht mancher Entwickler ist dies möglicherweise so, da sie sich nicht mit dem Projektumfeld auseinandersetzen müssen (und wollen) und sich auf die reine Entwicklungstätigkeit konzentrieren.

  Aus Sicht des Projektes, des Unternehmens und des Kunden ist es jedoch notwendig, auch viele andere Themenbereiche im Kontext mit dem agilen Vorgehen zu betrachten. Wenn etwas nicht im Scrum Guide steht, bedeutet dies ja nicht, dass es nicht sinnvoll ist und nicht gemacht werden muss.[2]

  Nachhaltig und langfristig denkende und agierende agile Teams berücksichtigen daher auch weitere Aspekte und setzen diese in ihrem Vorgehen um mit Release- & Roadmap-Meetings, Visions-Workshops, Risikoanalyse-Workshops etc.

- **Informationsüberlastung**
  In einem agilen Sprint soll am Ende des Sprints ein für den Kunden brauchbares Ergebnisartefakt präsentiert werden.

  Wenn die Stakeholder nun am Ende eines Sprints nicht mit einem lauffähigen Produkt(prototyp), sondern nur mit evtl. auch recht umfangreichen Spezifikationsartefakten konfrontiert werden, so kann dies zu einer Informationsüberlastung der Stakeholder im Review-Meeting am Ende des Sprints führen, da die Stakeholder eine große Informationsmenge in recht kurzer Zeit durchsehen und beurteilen müssen. Dies passiert vor allem dann, wenn die Stakeholder im Laufe des Sprints nur unzureichend eingebunden werden. Es ist durchaus legitim, auch Sprints zu planen, die primär Dokumentationsartefakte erzeugen, jedoch sollte hier darauf geachtet werden, dass die Stakeholder sehr eng im gesamten Sprint-Verlauf in die Kommunikation und Erarbeitung eingebunden sind und dass vor allem nicht nur reine Textartefakte erzeugt werden, bei denen meist der Kontext und die Zusammenhänge

2. Der Scum Guide sagt z.B. nichts oder wenig aus über Requirements Engineering, Risikomanagement, Architektur & Design, Codequalität und Wartbarkeit, Testvorgehensweisen, Build-Automatisierung, Deployment, Support & Wartung, Security etc. All diese Themen müssen jedoch auch in agilen Projekten berücksichtigt werden.

schwerer erkennbar sind oder verloren gehen. Zu empfehlen sind Techniken und Artefakte wie Modelle, Prototyping (auch wenn es nur auf Papier ist), Bilder etc., die mehr Überblick schaffen und den Stakeholdern ein besseres Produktverständnis und einen Wert bieten.

- **Alles inkrementell entwickeln**
  Themen, die in ihrem Umfang und der Komplexität schrittweise wachsen können oder auch in voneinander unabhängige Teile oder Schritte aufgeteilt werden können oder die von vornherein unklar bezüglich des Ergebnisses sind, eignen sich typischerweise für inkrementelle Umsetzung (z.B. Forschungsprojekte, Website-Projekte, Projekte mit unklaren Anforderungen). Hier ist die inkrementelle Annäherung auch vorteilhaft, weil sich mit jeder Iteration neue Erkenntnisse ergeben können und die Kunden durch die ersten Ergebnisse zu neue Anforderungen animiert werden oder erst dann verstehen, was sie eigentlich wollen.

  Themen, die nur in sich geschlossen oder als gesamte Einheit funktionieren, sind hingegen für eine iterative Umsetzung schlecht geeignet. Beispiel ist hier eine komplexe Berechnungsfunktion, die ein korrektes Ergebnis liefern soll. Wenn hier zuerst nur Teile der Berechnung spezifiziert und umgesetzt werden, kann das Ergebnis nicht korrekt berechnet werden. Oder beispielsweise eine Maschinensteuerung, bei der die Maschine insgesamt auch nur dann korrekt arbeitet, wenn alle notwendigen Sensoren und Aktoren richtig abgefragt und angesteuert werden.

  Für das Verständnis und die korrekte Umsetzung der Kernfunktionen ist es in diesem Fall notwendig, zumindest diese Kernfunktionen auch vorab komplett durchzudenken und zu spezifizieren.

  Natürlich bezieht sich dies nur auf die im Kern für die Korrektheit notwendigen Anforderungen. Darstellungs-, Komfortfunktionen oder ähnliche »Wunsch-Anforderungen« können natürlich auch später inkrementell dazu entwickelt werden.

- **Inkrementelle Entwicklung behindert Innovation**
  Agile Vorgehensweisen verfolgen einen Prozess der ständigen inkrementellen Verbesserung. Damit wird eine kontinuierliche Innovation begünstigt. Disruptive oder radikale Innovationen fördert dies jedoch nicht primär. Diese entstehen durch Betrachtung, Bewertung und Rekombination verschiedener Ideen. Die Entwicklung alternativer Ideen wird bei agilen Vorgehensweisen typischerweise als »waste« betrachtet (vgl. 10. Prinzip »Menge nicht getaner Arbeit zu maximieren«). Für disruptive oder radikale Innovationen müssen daher zusätzliche Praktiken wie z.B. Design Thinking oder Lean Startup berücksichtigt werden (siehe auch [IREB RE@Agile FL 2017] LE 1.5).

- **Agile bedingt einen Kulturwandel (der wichtigste Punkt)**
  Die Art und Weise, wie in agilen Organisationen gearbeitet wird, fördert kollektive Verantwortung, bringt aber auch den Verlust an »Eigentum« mit sich (Arbeitsprodukte, Ergebnisse, Prozesse etc.).

  Durch kontinuierliche Retrospektiven wird die laufende Verbesserung der Arbeitsweisen und evtl. auch der gesamten Organisation angestoßen.

  Diese ständigen Veränderungen benötigen Zeit und auch qualifizierte Personen, die die Teams und die Organisation in diese Richtung lenken. Dies sind Rahmenbedingungen, die bei der Anwendung agiler Vorgehensweisen beachtet werden müssen.

1.2.6 Resümee

Die Requirements-Engineering-Aktivitäten wie Erhebung, Dokumentation, Validierung und Verhandlung sowie Anforderungsmanagement müssen normalerweise auch in der agilen Entwicklung durchgeführt werden. Welche Techniken man für die Ermittlung und Dokumentation anwendet, kann durchaus zwischen agiler und nicht agiler Entwicklung unterschiedlich sein.

Es sollte die Chance genutzt werden, voneinander zu lernen, um damit schlussendlich eine bessere Lösung zu erreichen. Damit können die Stärken aus Agile und aus Requirements Engineering kombiniert und Verschwendung oder Overhead verringert werden.

Damit werden auch die fünf Scrum-Werte unterstützt (Engagement, Mut, Fokus, Offenheit und Respekt) und ihre Repräsentation in den drei Säulen von Scrum (Transparenz, Inspektion und Anpassung) [Schwaber & Sutherland 2016]. Offenheit und Respekt sind nützlich, wenn man Requirements Engineering in die agile Welt bringt bzw. die agilen Ideen und Prinzipien in die Welt des Requirements Engineering einführt.

Zusammengefasst kann man sagen, dass das agile Requirements Engineering mehr ist, als nur User Stories im Backlog zu erstellen und zu verwalten, und dass die Kern-Requirements-Engineering-Aktivitäten auch in agilen Projekten nicht vernachlässigt werden dürfen, sondern basierend auf den agilen Prinzipien mit unterschiedlichen Schwerpunkten und Methoden durchgeführt werden müssen.