Jan Ebert

(E-Mail: Jan.Ebert@HOOD-Group.com)
ist Principal Consultant und Leiter der technischen Strategie der HOOD Group.
Seine langjährige Erfahrung im Software- und Systems-Engineering, insbesondere in den Disziplinen Requirements Management und Engineering (RM&E), Change, Configuration & Version Management (CC&VM), Safety-, Risk- und Test-Management hat er durch Entwicklungs- und Beratungstätigkeit in unterschiedlichen Branchen gesammelt. Zu seinen Aufgabenschwerpunkten gehört die Analyse von Verbesserungspotenzialen und darauf aufbauend die bedarfsgerechte Weiterentwicklung von Methoden und Prozessen des RM&E sowie angrenzender Management-Disziplinen bei seinen Kunden und für das Portfolio der HOOD Group. Aktuell interessiert ihn besonders das Spannungsfeld zwischen agilen Entwicklungsansätzen und formalen Dokumentationspflichten in sicherheitskritischem und reguliertem Umfeld. Er ist zertifizierter Scrum-Master. Zusätzlich zu seiner Beratertätigkeit ist Herr Ebert für die HOOD Group auf Konferenzen und in der branchenübergreifenden Gremienarbeit tätig. In diesem Rahmen hat er unter anderem maßgeblich das gültige Standard-Format für den Austausch von Anforderungsdaten geprägt.

Susanne Mühlbauer

(E-Mail: Susanne.Muehlbauer@HOOD-Group.com)
ist Senior Consultant und Trainer bei der HOOD Group.
Ihre langjährige Erfahrung in den Bereichen Requirements Management und Engineering (RM&E), Change, Configuration & Version Management (CC&VM) sowie Projektmanagement konnte sie in zahlreichen Projekten in unterschiedlichen Branchen einbringen und vertiefen.
Zu ihren Aufgabenschwerpunkten gehört die Einführung von Methoden und Prozessen im Requirements Management und Engineering (RM&E) Umfeld sowie die Umsetzung von Methoden und Prozessen in Anforderungsmanagement-Werkzeugen und deren Einführung beim Kunden. Themenschwerpunkte ihrer Tätigkeit bilden weiterhin Software-Entwicklungsprozesse und agile Vorgehensweisen. Sie ist zertifizierter Scrum Master und Product Owner. Zusätzlich zu Beratungs- und Optimierungsprojekten im Software Engineering-Umfeld hatte sie in der Vergangenheit die Projektleitung für Implementierungsprojekte (unter anderem mit Scrum) im Bereich kundenspezifischer Software-Entwicklung und ERP-Systeme. Sie hat verschiedene Artikel und Studien zum Thema Collaboration Software und Anforderungsmanagement veröffentlicht und unterstützt die HOOD Group durch Beiträge auf Konferenzen.

Hirn einschalten und miteinander reden – Anforderungsmanagement und Scrum

Wer Scrum einsetzt, braucht kein klassisches Anforderungsmanagement mehr. So der Tenor von Scrum Protagonisten. Wir als Experten im Requirements Engineering stimmen dieser Ansicht voll und ganz zu, wenn Sie unter Anforderungsmanagement verstehen, dass zu Anfang des Projekts die Spezifikation geschrieben und eingefroren wird. Oder wenn Anforderungsmanagement für Sie darin besteht, umfangreiche Lastenhefte zu erstellen, die Sie einem Lieferanten kommentarlos übergeben. Oder wenn Anforderungsmanagement für Sie heißt, sich mit Strukturen von Dokumenten, mit Daten und Attributen zu beschäftigen, statt mit dem Inhalt der Anforderungen.
Wenn Sie allerdings den Wunsch haben, Anforderungsmanagement so zu verstehen, wie wir es einsetzen, nämlich um zu „verstehen, was der Kunde wünscht und sicherzustellen, dass er genau das auch bekommt“, dann wird Ihnen dieser Artikel einen Einblick geben, wie Scrum Sie dabei unterstützt, Ihr Anforderungsmanagement zu verbessern und umgekehrt.

Requirements Management und Engineering (RM&E), agiles Vorgehen und agile Methoden wie Scrum sind keine Gegensätze. RM&E betrifft und berührt vielmehr alle Aspekte des Systems- bzw. des Software-Engineering, denn RM&E ist keine isolierte Aufgabenstellung. Agiles RM&E räumt mit dem Missverständnis auf, dass es das Ziel von Anforderungsmanagement-Methoden sei, ein System bis ins letzte Detail zu spezifizieren, bevor mit der Implementierung begonnen werden darf. Agiles RM&E integriert sich in iterative und inkrementelle Entwicklungsprozesse und zwar von der ersten bis zur letzten Iteration oder eben vom ersten bis zum letzten Sprint. Somit können Scrum-Projekte von Anforderungsmanagement-Methoden profitieren und umgekehrt.

Wir stoßen immer wieder auf Befürchtungen und Vorurteile gegenüber „klassischen“ RM&E-Methoden: diese seien schwerfällig, die Prozesse zu schwergewichtig, nicht geeignet für agile Entwicklung. Das muss nicht sein! RM&E hat zum Ziel, Systeme zu entwickeln, die den Anforderungen der Kunden, Nutzer und weiterer Stakeholder entsprechen. Nicht mehr, aber auch nicht weniger.

Mit agilen Projektmanagement-Methoden wie Scrum wird nun versucht, die Zeitspanne von der Formulierung der Kundenvision bis zur ersten Implementierung stark zu verkürzen, um das Risiko einer Fehlentwicklung zu minimieren. Trotzdem geht es noch immer darum, den Wunsch des Kunden zu verstehen, eine Lösung zu finden, die technisch, finanziell und zeitlich realisierbar ist und die Anforderungen des Kunden bestmöglich erfüllt.

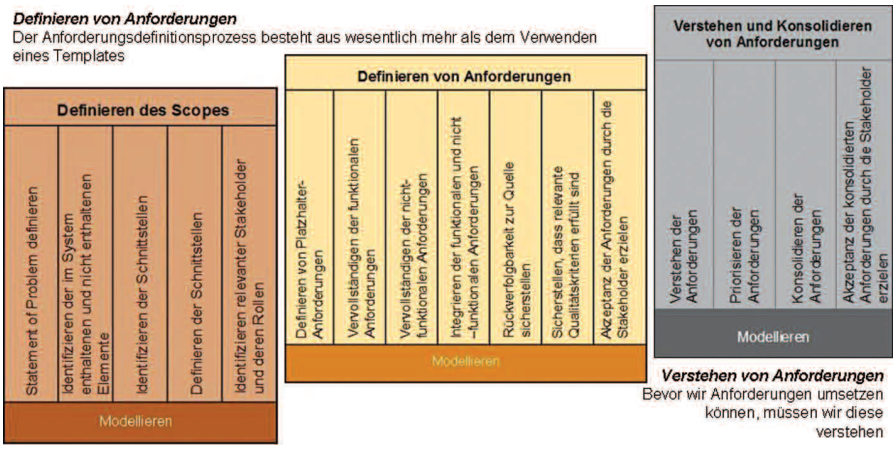

Abbildung 1: HOOD INSIDE Generischer RM&E-Prozess: Anforderungsdefinition [HIN]

Schließen sich RM&E und Scrum gegenseitig aus oder profitieren die Konzepte voneinander? Diese Fragestellung wird im vorliegenden Beitrag detaillierter untersucht.

Zentrale These dieses Artikels ist, dass sich sowohl die Methoden, Prozesse und Praktiken des RM&E als auch diejenigen von Scrum letztlich auf zwei grundlegende Aussagen reduzieren lassen: Wir müssen miteinander reden und das Hirn einschalten.

Im Kontext dieser beiden Aussagen betrachten wir mit Hilfe eines beispielhaften generischen RM&E-Prozesses für die Anforderungsdefinition (siehe **Abbildung 1**) typische Methoden und Vorgehensweisen des Anforderungsmanagements. Wir untersuchen, warum diese Ansätze in der Praxis oft nicht richtig umgesetzt werden und zeigen auf, wie Scrum dabei helfen kann, die Umsetzung zu verbessern.

Auf Bewährtes zurückgreifen

Der im Folgenden näher zu betrachtende generische RM&E Prozess für die Anforderungsdefinition ist in drei einfache und selbst wiederum generische Teilprozesse unterteilt (**siehe Abbildung 1**):

- (DS) Definieren des Scopes
- (VA) Verstehen und Konsolidieren von Anforderungen
- (DA) Definieren von Anforderungen

Diese Teilprozesse werden dazu verwendet, Anforderungen zu verstehen und sie schrittweise zu verbessern und zu vervollständigen. Sowohl klassische als auch iterative oder evolutionäre Vorgehensweisen werden unterstützt. Die Wahl des passendsten Vorgehensmodells hängt vom Entwicklungsgegenstand und dem jeweiligen Umfeld ab. Weiterhin gibt es keinerlei Reihenfolge für die Durchführung der Teilprozesse. Die Teilprozesse beeinflussen sich gegenseitig und können auch parallel zueinander durchgeführt werden. Die Ergebnisse des einen Schrittes liefern Input für die anderen Schritte und umgekehrt. Die Teilprozesse sind somit für iterative Vorgehensweisen prädestiniert.

Die Teilprozesse „Definieren des Scopes (DS)“ und „Verstehen und Konsolidieren von Anforderungen (VA)“ betrachten wir in den nächsten Abschnitten. Eine angemessene Diskussion des Teilprozesses „Definieren von Anforderungen (DA)“ würde den Rahmen dieses Artikels sprengen, ist aber für die Argumentationskette auch nicht erforderlich.

Hirn einschalten

Durch das Erstellen detaillierter und umfangreicher Vorabspezifikationen soll Menschen das Denken abgenommen werden. Ursprünglich natürlich nicht einmal im negativen Sinne, sondern vielmehr um Fehler zu vermeiden.

Leider kann das dazu führen, dass sich alle Beteiligten auf die Spezifikation zurückziehen und gute Ideen gar nicht erst zum Tragen kommen (siehe auch „miteinander reden“). Beispielsweise würde die Entwicklung eine einfachere, vielleicht sogar kostengünstigere Lösung vorschlagen; da der Kunde aber eine bestimmte Lösung bereits spezifiziert hat, wird auch diese umgesetzt.

Ein anderes Argument für Vorabspezifikationen ist die Angst davor, dass eine Lösung erstellt wird, die das Problem nicht löst oder die Features umfasst, die nicht gefragt sind. Hirn einschalten heißt eben nicht, dass sich Entwickler nun selbst verwirklichen können und das implementieren, was sie schon immer entwickeln wollten. Hirn einschalten heißt nicht, Zusatzfunktionalität zu implementieren, bei der nicht klar ist, wer diese zahlt, testet, dokumentiert, wartet, ...

An dieser Stelle wollen wir den Teilprozess „Definieren des Scopes (DS)“ näher betrachten, der aus mehreren Aktivitäten besteht (**siehe Abbildung 1**).

- Statement of Problem definieren
- Identifizieren der im System enthaltenen und nicht enthaltenen Elemente
- Identifizieren der Schnittstellen
- Definieren der Schnittstellen
- Identifizieren relevanter Stakeholder und deren Rollen

Im Folgenden gehen wir auf einzelne Punkte näher ein und zeigen auf, wie diese dabei unterstützen, die Angst vor Fehlern zu verringern.

Statement of Problem definieren

Im Statement of Problem steht, was das Ziel des Entwicklungsprojektes ist und welches Problem dadurch gelöst werden soll. Das hilft den Projektbeteiligten zu verstehen, warum ein Produkt bzw. System entwickelt werden soll. Das Statement of Problem darf kurz sein – ein paar Sätze reichen.

Identifizieren der im System enthaltenen und nicht enthaltenen Elemente

Es muss ganz klar definiert sein, was Teil des Systems bzw. Projektes ist und was nicht. Damit werden Fehlentwicklungen und die Entwicklung nicht geforderter Funktionalität vermieden und es wird sichergestellt, dass nichts Wichtiges vergessen wird. Hierfür eignen sich einfache und prägnante Darstellungen, wie zum Beispiel ein Kontextdiagramm. **Abbildung 2** zeigt auf der linken Seite, welche Use Cases im Scope des zu entwickelnden Hybrid SUVs liegen, auf der rechten Seite ist zu sehen, was sich außerhalb des Systems befindet, aber berücksichtigt werden muss.

Identifizieren relevanter Stakeholder und deren Rollen

Es muss selbstverständlich sein, auch die tatsächlichen Entwickler als Stakeholder des Systems zu sehen, ebenso wie den Systemarchitekten. Die Anforderungen aller Beteiligten sollten integriert werden und alle Experten sollten gemeinsam die Lösung

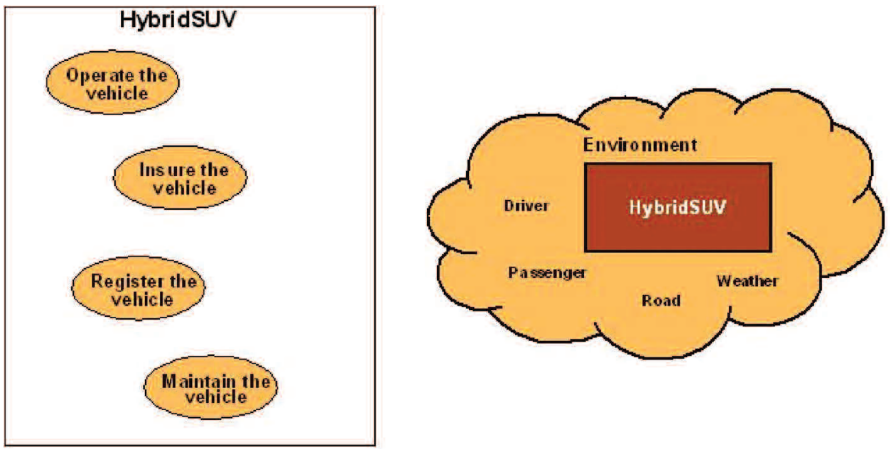

Abbildung 2: Beispiele für Kontextdiagramme

spezifizieren. Das bedeutet aber auch für Kunden, nicht zu viel Lösung vorzugeben, sondern vielmehr das eigentliche Problem zu beschreiben. Dies ist eines der Grundprinzipien des Anforderungsmanagements. Die Lösungsexperten müssen die Chance haben, das Problem zu verstehen, das durch die Lösung beseitigt werden soll. Es ist nicht das Ziel, ihnen das Denken abzunehmen.

Bewährtes durch Scrum ergänzen

Statement of Problem definieren

Scrum fordert eine Product Vision. In dieser wird das Ziel des Projektes festgehalten. Der Weg dorthin wird nicht beschrieben, er wird evolutionär entwickelt. Hilfsmittel hierfür ist der Release Plan, der auf dem Product Backlog basiert.

Scrum-Experten empfehlen die Verwendung von Epics für die Produktvision bzw. von User Stories zum Füllen des Product Backlogs. Aus unserer Sicht ist dies ein sehr geschicktes Mittel, den Kunden dazu zu bewegen, das Problem zu benennen und nicht eine Lösung zu beschreiben (**siehe Abbildung 3**).

Als	<Art von Benutzer>
möchte ich	<Anforderung/ Funktion>
damit	<Grund>

Abbildung 3: Standardisierte Form der User Story nach Mike Cohn

Identifizieren der im System enthaltenen und nicht enthaltenen Elemente

Kernelement eines jeden Eintrags im Product Backlog sind Abnahmekriterien. Der Product Owner beschreibt hier, welche Anforderungen erfüllt sein müssen, damit er das Produkt abnimmt.

Damit stellt er klar, was für ihn zur Realisierung eines Features dazugehört und was nicht.

Ein weiteres zentrales Element, um festzulegen, was für die Abnahme eines Produktinkrements notwendig ist, ist die Definition of Done.

Identifizieren relevanter Stakeholder und deren Rollen

Scrum bindet neben den Stakeholdern auf Kundenseite (repräsentiert durch den Product Owner) auch das Entwicklungsteam als Stakeholder von Anfang an in den Abstimm- und Verstehensprozess ein.

Das Entwicklungsteam verfügt dabei über alle Expertenkenntnisse, die notwendig sind, um die Software zu erstellen, zu testen und zu dokumentieren (bzw. die Definition of Done zu erfüllen).

Das Team wird dabei vom Kunden als Berater wahrgenommen. Die Lösung wird vom Team erarbeitet und der Kunde kann am Entstehungsprozess der Lösung teilhaben, bzw. Scrum fordert seine Anwesenheit für die Klärung von Fragen während der Lösungsfindung regelrecht ein.

Miteinander reden

Warum überhaupt miteinander reden? Es ist doch viel wichtiger, alles genau aufzuschreiben, z.B. in Form von Lasten- und Pflichtenheften oder Fachkonzepten und IT-Konzepten: Als Vertragsbasis, um nichts zu vergessen, um den geforderten Prozess einzuhalten, um die Anforderungen weitergeben zu können, etc. Geht das eine ohne das andere?

Um zu verstehen, was der Kunde wünscht, muss miteinander geredet werden. Leider erleben wir in der Praxis noch sehr oft, dass das „Miteinander reden" durch ein „Gegeneinander schreiben" ersetzt wird. In mühevoller, wochenlanger oft monatelanger Arbeit werden umfangreiche Spezifikationen für Software-Systeme erstellt, die alle Unklarheiten im Vorfeld beseitigen sollen.

Das kann nicht gelingen, denn es handelt sich in der Regel um komplexe Systeme, für die nicht alle Eventualitäten vorausgedacht und alle Unsicherheiten ausgeschlossen werden können.

Warum ist das Miteinander reden in der Praxis scheinbar so schwer umzusetzen und mit welchen Hindernissen haben wir in Softwareprojekten zu kämpfen? Welche bewährten Methoden und welche agilen Ergänzungen können hier Abhilfe schaffen?

Szenario

Wir nehmen an, der Entwicklungsabteilung gehen Anforderungen zur Implementierung eines Software-Systems in schriftlicher Form zu, z.B. als Lastenheft oder Fachkonzept. Dabei ist es unerheblich, ob der Verfasser ein interner Kunde (z.B. die Fachabteilung) oder ein externer Kunde ist. Der erste Schritt auf Seite des Empfängers sollte nun sein, die Anforderungen, die er erhalten hat, zu verstehen. Zuerst führen wir anhand des Teilprozesses „Anforderungen verstehen und konsolidieren (AV)" aus (**siehe Abbildung 1**), welche Ursachen es dafür geben kann, dass sich die Beteiligten nicht um ein gemeinsames Verständnis bemühen. Anschließend beleuchten wir, wie agiles RM&E diesen Ursachen entgegenwirken kann. Zuletzt betrachten wir, inwieweit eine agile Projektmanagement-Methode wie Scrum die positiven Effekte von agilem RM&E noch verstärken kann.

Anforderungen verstehen und konsolidieren

Im Folgenden befassen wir uns hauptsächlich mit Anforderungen, die bereits in schriftlicher Form (zum Beispiel als Spezifikation oder Lastenheft eines Kunden) vorliegen. Die Gewinnung dieser Anforderungen – d.h. der Themenkomplex der Anforderungserhebung – wird in diesem Beitrag nicht betrachtet (siehe hierzu z.B.

[HO1]). Ziel dieses Schrittes ist es, diese Kundenanforderungen zu verstehen, zu priorisieren, zu konsolidieren und mit den Stakeholdern abzustimmen.

Um Anforderungen zu verstehen, empfehlen sich die in **Kasten 1** dargestellten „Good Practices“ und Methoden:

Good Practices:

- Tragen Sie die Anforderungen zusammen und lesen Sie diese.
- Diskutieren Sie die Anforderungen mit dem Eigentümer der Anforderungen.
- Lassen Sie Ihr Verständnis der Anforderungen durch ein Review überprüfen (zumindest durch den Eigentümer der Anforderungen).
- Verwenden Sie Modellierungstechniken, um Ihr Verständnis der Anforderungen in einer anderen Repräsentationsform darzustellen (zum Beispiel als Grafik, als Bild, als Diagramm oder auch mittels eines Prototypen). Wählen Sie eine Repräsentationsform, die der Reviewer verstehen kann.
- Stellen Sie Fragen und dokumentieren Sie die Antworten der Stakeholder.
- Stellen Sie klar, wo Sie Annahmen getroffen haben.

Methoden:

- Diskussion
- Modellierung (inklusive Protoypenbildung)
- Iterative Verwendung von „Early Releases“

All diese „Good Practices“ und Methoden für sich genommen sind einfach und entsprechen dem „Common Sense“ – also unserem gesunden Menschenverstand. Warum tun wir uns aber in der Praxis offensichtlich so schwer mit der Umsetzung dieser Vorgehensweisen? Ursachen sind in den Unternehmensstrukturen zu finden, im Zeitdruck, in der Notwendigkeit, sich schriftlich abzusichern. Aber auch in der Angst, sich festzulegen, da Änderungen an der Tagesordnung sind oder Entscheidungen unter großer Unsicherheit getroffen werden müssen.

Bewährtes agil einsetzen

Abhilfe schafft es, wenn nicht im Vorfeld versucht wird, eine vollständige Spezifikation des Systems zu erstellen, sondern, wie in der Entwicklung auch, in der Erstellung der Anforderungsdokumente iterativ vorgegangen und mit den wichtigsten Anforderungen begonnen wird.

Die Teile der Anwendung, die besonders wichtig sind, sollten zuerst beschrieben werden, ebenso wie besonders risikobehaftete Anforderungen, die für das Gelingen des Entwicklungsvorhabens essenziell wichtig sind. Es empfiehlt sich, Teile in der Spezifikation, die noch ungenau sind und an denen noch Änderungen erwartet werden, zu kennzeichnen. Weiterhin ist es sinnvoll Platzhalter für Anforderungen zu verwenden, die am Anfang des Projektes nicht so wichtig sind, die aber nicht vergessen werden sollen oder die für die Einschätzung des Gesamtsystems relevant sind.

Diskussionen und Verhandlungen sollten mit den wichtigsten und gut beschriebenen Anforderungen begonnen werden. Diese sollten frühzeitig mit dem Lieferanten bzw. der Entwicklung abgestimmt werden. Ziel muss sein, dass die Entwicklung das Problem versteht, das gelöst werden soll. Geschieht das in kleineren Häppchen, also in Iterationen, werden die Beteiligten weniger schnell die Lust verlieren, da sie sich nicht vor einem riesigen Dokument wiederfinden und auch eher ein Zeitfenster für die Lektüre und die Diskussion der Anforderungen bereitstellen können.

... und Bewährtes durch Scrum ergänzen

Agile Frameworks, wie beispielsweise Scrum bieten mit einfachen, aber schlagkräftigen Regeln eine große Unterstützung dabei, Anforderungsmanagement zu verbessern und Hindernisse zu beseitigen (**siehe auch Kasten 1**).

Lesen der Anforderungen

Bei Scrum bekommt das Entwicklungsteam die Anforderungen vom Kunden quasi „vorgelesen“! Dies erfolgt in einem so genannten Sprint-Planungsmeeting an dem der Kunde(nrepräsentant) – also der Product Owner - und die Entwicklung - also das Team - teilnehmen.

Der Product Owner stellt auf Basis des Product Backlogs in diesem Meeting die wichtigsten und hochprioren Anforderungen vor. Es werden nur so viele Anforderungen besprochen, wie im nächsten Sprint umgesetzt werden sollen.

Und auch nur diese Anforderungen müssen ausreichend – also so, dass sie vom Team verstanden und umgesetzt werden können – detailliert und beschrieben sein. Das reduziert den Umfang der „Spezifikation“ und deren Änderungsanfälligkeit.

Diskutieren der Anforderungen

Damit zwischen Product Owner ein Gespräch bzw. eine Diskussion forciert wird, wird im Vorfeld möglichst wenig aufgeschrieben. Scrum empfiehlt, für die Befüllung des Backlogs, User Stories als Spezifikationsmittel zu verwenden. Die User Stories enthalten nur so viel Information wie nötig ist, um als Platzhalter oder Leitfaden für ein Gespräch dienen zu können (**siehe Abbildung 3**). Vor dem Sprint-Planungsmeeting sollten die Product Backlog Items jedoch so weit analysiert und detailliert sein, dass das Team mit der Implementierung beginnen könnte.

Stellen Sie Fragen und dokumentieren Sie die Antworten der Stakeholder

Eine aus Anforderungsmanagement-Sicht ganz zentrale Rolle in Scrum-Projekten ist der Product Owner. Der Product Owner ist nicht nur der Repräsentant des Kunden – im Idealfall ist es der Kunde selbst, der Teil des Entwicklungsprojektes ist und nicht nur am Anfang (Spezifikation und Vertrag) und Ende (Abnahme) des Projektes in Erscheinung tritt. Scrum fordert die Anwesenheit des Product Owners bei den Planungsmeetings und bei Fragen, die während eines Sprints auftreten. Am Ende jeder Iteration muss der Product Owner das Produkt abnehmen. So fordert Scrum eine extrem hohe Verfügbarkeit des Kunden und minimiert Verzögerungen während der Implementierung.

Stellen Sie klar, wo Sie Annahmen getroffen haben

Eine Ursache für schlechte Spezifikationen ist die Angst davor, sich festzulegen bzw. das Unvermögen, Entscheidungen unter Unsicherheit zu treffen. Das wiederum führt dazu, dass Annahmen getroffen werden müssen.

Hier unterstützt die Idee von Scrum, möglichst wenig vorab fest zu schreiben bzw. nur das im Detail zu spezifizieren, was als nächstes umgesetzt werden muss und was bereits ausreichend verstanden und geklärt wurde.

Gegen den Aspekt der Angst wirkt weiterhin ein iteratives Vorgehen. Scrum fordert extrem kurze Entwicklungsiterationen von 2 bis 4 Wochen, mit deren Ablauf potenziell lieferbare und lauffähige Software vorzuliegen hat. Eine weitere Vorgabe ist

es, hochpriore und risikobehaftete Anforderungen zuerst zu implementieren.

Dadurch erreicht Scrum, mit jeder Iteration dazu zu lernen, die Unsicherheit zu reduzieren, aus Erfahrung heraus zu entscheiden und mit weniger Angst in den nächsten Sprint zu gehen.

Fazit

Eine der zentralen Herausforderungen in Entwicklungsprojekten ist der Umgang mit Anforderungen. Das Risiko einer Fehlentwicklung kann durch RM&E verringert werden. RM&E liefert bewährte Praktiken und Methoden, um Anforderungen zu erheben, zu verstehen und in Lösungen umzuwandeln. Eine Auswahl davon haben wir in vorliegendem Artikel exemplarisch dargestellt. Der Einsatz dieser Methoden muss nicht langwierig und schwerfällig erfolgen, im Gegenteil. Wenn Sie jedoch in Ihrer Organisation Schwierigkeiten haben, RM&E agil zu betreiben und Anforderungen iterativ zu entwickeln, dann können Projektmanagement-Methoden wie Scrum Ihnen dabei helfen, ein Umdenken in der Organisation und in der Vorgehensweise in Entwicklungsprojekten herbeizuführen. Die Ideen und Leitsätze von agilem RM&E und Scrum harmonieren und unterstützen sich wechselseitig. ■

www.HOOD-Group.com

Literatur

[PIC] Pichler, Roman. (2008). *Scrum. Agiles Projektmanagement erfolgreich einsetzen* (1. Auflage). Heidelberg: dpunkt.verlag GmbH.

[COH] Cohn, Mike. (2009). *User Stories Applied. For Agile Software Development* (13th Printing). Boston, MA: Pearson Education Inc.

[SCH] Ken Schwaber. (2009). *Scrum Guide, May 2009, www.scrumalliance.org/resources*

[SC1] Schwaber, Ken. (2007). *Agiles Projektmanagement mit Scrum*. Unterschleißheim: Microsoft Press Deutschland.

[HIN] HOOD INSIDE, Bastian Gollwitzer, Markus Hoppe, 2009, http://www.hood-group.com/de/produkte/hood-inside/

[HO1] Hood, Colin & Wiebel, Rupert. (2005). *Optimieren von Requirements Management & Engineering. Mit dem HOOD Capability Model.* Heidelberg: Springer-Verlag Berlin.

[HO2] Hood, Colin, Wiedemann, Simon, Fichtinger, Stefan & Pautz, Urte. (2008). *Requirements Management. The Interface Between Requirements Development and All Other Systems Engineering Processes* . Berlin, Heidelberg: Springer-Verlag.