mehr zum thema:
www.v-arc.at

AGILITÄT UND REQUIREMENTS-ENGINEERING: ERGÄNZUNG ODER WIDERSPRUCH?

Die Vorteile agiler Vorgehensweisen sind heute breit publiziert. Agile Projekte geraten aber schnell auch in Probleme der „Unterdokumentiertheit", was sich in schlechter Nachvollziehbarkeit und Wartbarkeit der Software äußert. Viele Unternehmen, die auf agile Methoden umsteigen, haben langjährige Erfahrungen mit Konzepten aus dem klassischen Requirements-Engineering (RE), nutzen diese aber beim Umstieg nicht. Dieser Artikel diskutiert, welche konzeptionellen Überschneidungen zwischen agilem und klassischem RE existieren und wie agile Projekte von klassischen RE-Konzepten, wie z. B. Use-Cases, profitieren können.

die autoren

Dr. Wolfgang Göbl
(wolfgang.goebl@r-solution.at)
ist IT-Unternehmensarchitekt bei Raiffeisen Solution. Das Zusammenspiel zwischen Scrum und Requirements-Management eröffnet Chancen und Fragen, denen er als Moderator des „Vienna Agile Requirements Circles" (www.v-arc.at) nachgeht.

Dr. André Köhler
(koehler@softwareforen.de)
ist Geschäftsführer der Softwareforen Leipzig GmbH. Dort ist er fachlicher Leiter der User Groups „Agile Methoden in der Softwareentwicklung" sowie „Requirements Engineering".

Status quo: Klassisches RE

In klassischen Wasserfall-ähnlichen Vorgehensmodellen fokussiert sich das *Requirements-Engineering (RE)* auf die Erstellung und Verwaltung einer überwiegend aus einer funktionalen Sicht geprägten vollständigen und korrekten Anforderungsspezifikation. Das Gebiet hat sowohl in der Forschung als auch in der Praxis weit verbreitete Methoden hervorgebracht, beispielsweise Use-Cases und Objektmodelle (vgl. [Jac92]), Prototypen, Geschäftsregel-Modellierung sowie professionelle Ermittlungs- und Konsolidierungstechniken.

Im RE spielt der Analytiker eine zentrale, gestaltende Rolle bei der systematischen Erarbeitung der Anforderungsspezifikation aus oft diffusen, widersprüchlichen und nur unterbewusst vorhandenen Wünschen der relevanten Stakeholder. RE hat insbesondere in großen Organisationen erhebliche Bedeutung im Hinblick auf gesetzliche Dokumentationspflichten, die Nachverfolgbarkeit von Änderungen, die Auflösung von Anforderungskonflikten und andere Aspekte. Zentrales Konzept im klassischen RE ist der *Use-Case*. Er beschreibt eine grob-granulare Funktionalität des betrachteten IT-Systems aus einer Nutzungssicht. Gemäß Jacobson hat ein mittleres IT-System vielleicht 20 Anwendungsfälle, ein riesiges bis zu 100. Ein Use-Case hilft dabei, dass „ein Benutzer genau ein vollständiges Benutzerziel erreichen kann" (vgl. [Jac92]).

Durchaus bemerkenswert ist, dass Ivar Jacobson das Use-Case-Konzept derzeit weiterentwickelt. Vor Kurzem hat er in „Use Cases 2.0" den Ansatz „Use Case Slices" präsentiert: „One or more stories selected from a use case to form a work item that is of value to the customer" (vgl. [Jac11]). Dieser Ansatz versucht, eine Brücke zur agilen RE-Welt zu bauen.

Status quo: Agiles RE

Agile Vorgehensmodelle ändern die Sicht auf RE radikal – die schriftliche Spezifikation tritt in den Hintergrund, die möglichst direkte Kommunikation zwischen dem Kunden (z. B. dem Fachbereich in Form des *Product Owners, PO*) und dem Entwicklungsteam wird zu einem zentralen Element. Anforderungen werden grob und unvollständig in Form von *User-Storys* (vgl. [Coh04]), *Features* (vgl. [Pal02]) oder *Testfällen* und *Beispielen* (vgl. [Adz09]) beschrieben und nach *Epics*, *Themes* (vgl. [Coh12]) und *Story Maps* (vgl. [Pat12]) strukturiert. Zentrales Konzept ist dabei häufig die User-Story. Sie ist eine aus Sicht des Systembenutzers formuliertes „Versprechen, um später eine Konversation über ein bestimmtes Thema zu führen" (vgl. [Cob12]), oftmals in nur ein oder zwei Sätzen formuliert.

Aus dem klassischen RE bekannte Konzepte zum sukzessiven strukturierten Detaillieren der Anforderungen werden in der agilen Welt nur selten verwendet. Viele agile Projektteams verwenden User-Storys als alleinige Anforderungsspezifikation. Anstelle der zu recht kritisierten „Überdokumentiertheit" (und somit Trägheit) so mancher Wasserfall-Projekte entsteht eine „Unterdokumentiertheit" – die Projekte geraten schnell in Nachvollziehbarkeits- und Wartbarkeitsprobleme und verlieren leichter den Gesamtzusammenhang der User-Storys aus den Augen.

Die beiden Welten zusammenbringen

Verfolgt man die vielen Diskussionen der immer größer werdenden Scrum-Community, kann man leicht den Eindruck gewinnen, dass die über Jahrzehnte gereiften klassischen RE-Konzepte wie Use-Cases eher lästiges Beiwerk denn wertvolle Assets sind. Auf der größten deutschen RE-Konferenz (vgl. [REC12]) war 2012 der Bruch zwischen der agilen und der RE-Community spürbar. Aus Sicht der agilen Community ist das klassische RE zu unflexibel und zu wenig zielorientiert, weil eine „dicke Anforderungsspezifikation keinen Anwender glücklich macht". Aus Sicht des „klassischen RE" gibt es innerhalb der agilen Vorgehensweisen „zu schlichte Thesen zum Thema RE, die die Erfahrungen aus 20 Jahren RE ignorieren" (vgl. [Rup10]).

Ein erster Ansatzpunkt, um die beiden Welten zusammenzubringen, sollte die Zusammenführung der Begriffswelten sein.

Ansatz für ein gemeinsames Begriffsverständnis

Anforderungen aus dem Fachbereich

Ausgangspunkt unserer Überlegungen ist der Fachbereich bzw. das Produktmanagement, das die Erstellung oder

Veränderung eines *Softwareprodukts* wünscht. Das Softwareprodukt wird in der Regel in einem *Geschäftsprozess* eingesetzt, aus dem heraus sich die Wünsche an das Softwareprodukt ergeben. Solche Wünsche werden in Form von *fachlichen Anforderungen* beschrieben, zunächst meist ohne strukturelle Vorgaben.

Klassisches RE

In der RE-Praxis haben sich Uses-Cases in den letzten beiden Dekaden durchgesetzt. Sie entstammen dem klassischen Wasserfall-Denken und können bis ins letzte Detail ausspezifiziert werden. Die Erfahrung zeigt allerdings, dass das in der Regel nicht sinnvoll ist. Ein Use-Case wird durch ein oder mehrere *Szenarien* verfeinert, aus denen sich dann schließlich Detailanforderungen für die Entwicklung ableiten lassen. Daneben gibt es weitere wichtige Artefakte, die typischerweise begleitend dazu entstehen. Dazu gehören zum Beispiel *Geschäftsregeln, Geschäftsobjekte, User Interface Mockups* und *Datenmodelle*.

Agil organisiertes Softwareprojekt

Mit Blick auf ein agil organisiertes Projekt (in unserem Fall nach Scrum) lassen sich aus den fachlichen Anforderungen *Themes* ableiten. Ein Theme wird verfeinert durch *Epics* bzw. *User-Storys*. Eine User-Story wird später in fein-granulare Tasks für die Softwareentwicklung zerlegt. Diese Artefakte werden im *Sprint* bzw. im *Product Backlog* verwaltet.

Zusammenführung der Welten

Um professionelles agiles RE umzusetzen zu können, braucht man zumindest folgende Konzepte:

- Ein Konzept zur Systemabgrenzung und zum Clustering von Funktionalität. Hier gibt es mit Epics, Themes und Use-Cases gleich drei konkurrierende Konzepte.
- Ein Konzept zur agilen Priorisierung von Tasks. Hier sind User-Storys weit verbreitet, der neue Ansatz von „Use-Case-Slices“ (vgl. [Jac11]) könnte zukünftig eine echte Alternative werden.
- Ein Konzept, das dazu geeignet ist, durchgängig funktionale Anforderungen von grob bis zum für das jeweilige Projekt ausreichenden Detaillierungsgrad darzustellen. Hier sind iterative Use-Cases (von grob bis so detailliert wie nötig), wie von [Cob03] vorgestellt, ein viel versprechendes Mittel. Die in der Agilität verwendeten Konzepte „Themes“, „Epics“ und „User-Storys“ gehen in eine ähnliche Richtung, es fehlt jedoch ein Konzept zur detaillierten Beschreibung von Anforderungen.
- Begleitende Modelle (z. B. „Geschäftsobjekt-Modell“, „User Interface Mockups“), die sowohl für agile als auch für klassische RE-Projekte notwendig sind. Diese Modelle müssen übergeordneten funktionalen Blöcken zugeordnet werden können (Use-Cases, Themes).

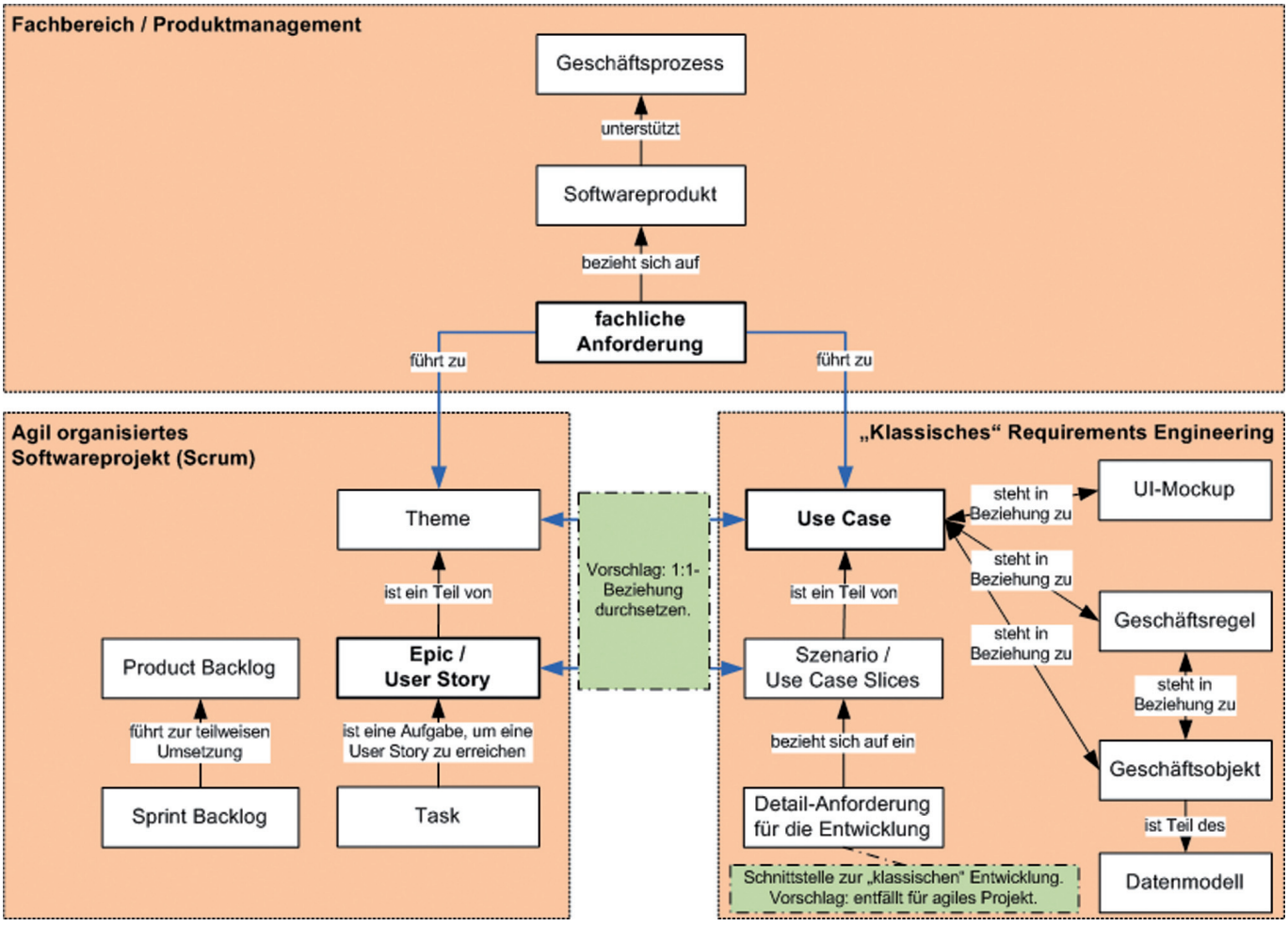

Abb. 1: Begriffe der agilen und der klassischen RE-Welt und deren Zusammenspiel.

In Bezug auf Use-Cases konnten wir in vielen Diskussionen den Eindruck gewinnen, dass diese in der agilen Community eher verpönt sind. Das Konzept wird hier offensichtlich so interpretiert, dass ein Use-Case bis zum letzten Alternativszenario ausspezifiziert werden muss, bevor mit der Implementierung begonnen werden darf. Aus unserer Sicht gibt es hier einiges an verbrannter Erde, die durch die häufig überspezifizierten Use-Cases der „alten Welt" entstanden ist, was von der agilen Community jedoch missverstanden zu sein scheint: In [Cob03] werden Use-Cases als „iterativ" und „multi-level" definiert (vom „ultralight" bis zum „detailed" Level). Für jede Ebene wird hier definiert, welche Attribute jeweils zu erfassen sind. Sie sind somit eigentlich das ideale Konstrukt für agiles Anforderungsmanagement und können spätestens seit [Jac11] sehr leichtgewichtig verwendet werden.

In [Göb12] wird dieses Konzept aufgegriffen und ein konkreter Ansatz beschrieben, wie Use-Cases als iteratives Strukturierungsmittel für agiles RE verwendet werden können: Zu Beginn eines Scrum-Projekts (Sprint 0) wird die Systemabgrenzung auf Basis von „ultralight" Use-Cases (nur Namen und Kurzbeschreibung) durchgeführt. Die funktionalen User-Storys im Product Backlog werden diesen Use-Cases zugeordnet. So behält der PO von Beginn an den Überblick über das Gesamtsystem.

User-Storys (vgl. [Coh04]) auf der anderen Seite sind heute das Herzstück in der agilen Projektplanung. Sie beschreiben Anforderungen des POs grob und ohne Anspruch auf Vollständigkeit. In der agilen Praxis kann man häufig beobachten, dass User-Storys (meist um User-Interface-Skizzen angereichert) als alleinige Anforderungsdokumentation verwendet werden. Dieser Ansatz hat jedoch einen entscheidenden Nachteil: Der Gesamtüberblick über das System geht verloren, da zumeist hunderte von User-Storys unterschiedlichster Granularität entstehen. Der PO tut sich dadurch schwer, Anforderungen zu priorisieren. Abhilfe schaffen Konzepte wie „Epics", „Themes" und „Story Mapping", die die User-Storys nach funktionalen Kriterien zusammenfassen.

Coburn vergleicht User-Storys und Use-Cases wie folgt: „Eine User Story ist das Versprechen, eine Konversation zu haben. Ein Use-Case ist die Aufzeichnung dieser Konversation (falls erforderlich)" (vgl. [Cob12]).

Um nun die Begrifflichkeiten aus beiden Welten in einen einigermaßen konsistenten Zustand zu bringen, schlagen wir eine Verknüpfung vor, wie sie in **Abbildung 1** dargestellt ist. Die Kern-Entitäten der Welten sind *fachliche Anforderungen* (Fachbereich bzw. Produktmanagement), *Use-Cases* (klassisches RE) und *User-Storys* (agil).

Aus einer fachlichen Anforderung lassen sich eine oder mehrere Themes bzw. Use-Cases ableiten. Zu Beginn der Analyse können diese Konzepte ähnlich verwendet werden (z. B könnte „Kunden verwalten" ein Theme und ein High-Level-Use-Case sein). Use-Cases werden jedoch später weiter genauer detailliert, während Themes meist nur als Begriff und Kurzbeschreibung stehen bleiben.

Die Konzepte der „Themes" und „Use-Cases" zielen anfangs auf die Lösung derselben Problemstellung ab: das System in überschaubare Blöcke aufzuteilen. Trotzdem gibt es einen wesentlichen Unterschied:

- *Themes*, definiert als „Sammlung miteinander in Beziehung stehender User-Storys" (vgl. [Amb10]), machen keine Vorgaben über ihre Granularität.
- Jacobson macht für Use-Cases striktere Vorgaben (vgl. [Jac92]): Use-Cases sollen von „mittlerer Granularität" sein (im Schnitt 20 bis 40 für ein durchschnittliches Softwaresystem) und ein „vollständiges Benutzerziel" haben. „Geschäftsregeln als Java-Klassen realisieren" könnte ein Theme, aber kein Use-Case sein (kein Ziel des Benutzers). In der Praxis werden diese Konzepte nach unserer Erfahrungen deshalb auch nicht identisch verwendet. Zum Beispiel umfasst ein Theme oft Funktionalität, die man im klassischen RE mehreren Use-Cases zuordnen würde.

Um jedoch beide Welten in Einklang zu bringen, schlagen wir vor, eine 1:1-Beziehung durchzusetzen, d. h. Themes so zu schneiden, wie man es auch für Use-Cases tun würde, und dadurch ein eindeutiges Mapping zu ermöglichen. Beide Welten können dadurch mit ihren gewohnten Begriffen arbeiten, aber es ist trotzdem eine eindeutige Zuordnung zwischen den Artefakten möglich.

Themes bzw. Use-Cases werden weiter in detailliertere Funktionsblöcke heruntergebrochen: Ein Theme wird weiter in Epics/User-Storys unterteilt, ein Use-Case in Szenarien/Use-Case-Slices. Aus einer User-Story (bzw. Use-Case-Slice) werden schließlich fein-granulare Tasks für die Softwareentwicklung abgeleitet. Epics, User-Storys/Use-Case-Slices und Tasks werden wie gewohnt im Product- bzw. Sprint-Backlog dokumentiert.

Fazit und Dank

Der Begriffswirrwarr, der durch die Vermischung agiler Konzepte und RE-Konzepte entstanden ist, macht das Thema „Agiles RE" derzeit unübersichtlich. Wir sind davon überzeugt, dass beide Ansätze nicht nur eine Daseinsberechtigung haben, sondern sich auch sinnvoll ergänzen können. Das von uns vorgestellte Modell veranschaulicht deshalb die Zusammenhänge zwischen diesen Konzeptwelten des RE und der agilen Community und will so zur Aufhellung beitragen.

In der agilen Welt haben sich mit Themes, Epics und User-Storys schlanke Konzepte durchgesetzt. Schaut man sich die diesen agilen Methoden zu Grunde liegenden Konzepte genauer an, so erkennt man große Überschneidungen mit althergebrachten RE-Methoden. Eine Konsolidierung der Begriffe und letztlich der Methoden erscheint daher erstrebenswert.

Zwei wesentliche Unterschiede zwischen den agilen und den klassischen RE-Konzepten können herausgearbeitet werden:

- Use-Cases machen striktere Vorgaben für ihre Granularität als Themes und führen deshalb dazu, dass die Anforderungsdokumentation überschaubarer bleibt und so der Gesamtüberblick über das System besser gewahrt bleibt. Wir schlagen deshalb vor, für Themes dieselben strikten Vorgaben wie für Use-Cases zu verwenden.
- Use-Cases können so tief detailliert werden, wie es für die konkrete Problemstellung sinnvoll ist, User-Storys zielen eher auf den Planungsaspekt und sind keine detaillierte Anforderungsdokumentation.

Wir sind der Überzeugung, dass agile Projekte von den klassischen RE-Konzepten profitieren können. Die agilen RE-Konzepte ersetzen das klassische RE nicht vollständig. Es fehlt ein Konzept zur detaillierten Beschreibung von Anforderungen, die für komplexe Fälle auch in

agilen Projekten sinnvoll ist. Der richtige Einsatz bewährter RE-Konzepte kann helfen, die Probleme der „Unterdokumentiertheit“, wie Nachvollziehbarkeit und Wartbarkeit, zu vermeiden. Die Kunst ist es, zur richtigen Zeit den richtigen Detaillierungsgrad der Anforderungen zu dokumentieren („so wenig wie möglich und so viel als nötig“). Das klassische RE bietet bewährte Konzepte, um dies tun zu können.

Mit dem Use-Case-Slice-Ansatz in „Use-Cases 2.0“ existiert seit Kurzem eine zweite agile RE-Konzeptwelt, die aus dem klassischen RE stammt (vgl. [Jac11]) und die einen Versuch unternimmt, die beiden Welten zu verbinden. Aus unserer Sicht ist das ein Schritt in die richtige Richtung.

Der hier vorgestellte Ansatz ist bei den Softwareforen Leipzig im Zuge der User-Group „Agile Methoden in der Softwareentwicklung“ interaktiv unter Mitwirkung von rund 30 Teilnehmern entstanden. Das Schwerpunktthema des Arbeitstreffens lautete „Agiles Vorgehen und Requirements Engineering – Freunde oder Feinde?“. In dieser Gruppe treffen sich halbjährlich Fachleute aus softwareintensiven Unternehmen, um anhand von Impulsvorträgen und Workshops über die aktuellen Herausforderungen bei der Einführung und Umsetzung agiler Methoden zu diskutieren und konkrete Handlungsempfehlungen für ihre eigene Projektpraxis zu erarbeiten.

Wir danken Dr. Thorsten Weyer von der Universität Duisburg-Essen, einem anerkannten Experten für das Thema RE, für hilfreiche Anmerkungen im Rahmen des Reviews dieses Artikels. ■

Literatur & Links

[Adz09] G. Adzic, Bridging the Communication Gap: Specification by Example and Agile Acceptance Testing, Neuri Limited 2009

[Amb10] S.W. Ambler, siehe: agilemodeling.com/artifacts/userStory.htm#Themes

[Cob03] A. Coburn, Use Cases effektiv erstellen, Mitp-Verlag 2003

[Cob12] Blog von A. Coburn, siehe: alistair.cockburn.us/Why+I+still+use+use+cases

[Coh04] M. Cohn, User Stories Applied, Addison Wesley 2004

[Coh12] Blog von M. Cohn, siehe: blog.mountaingoatsoftware.com/stories-epics-and-themes

[Göb12] W. Göbl, Use Cases in agilen Projekten, in: Proc. of ReConf 2012, siehe: 2012.reconf.de/vortragsdownload/

[Jac11] I. Jacobson, I. Spencer, K. Bittner, Use-Cases 2.0 – The Guide to Succeeding with Use Cases, 2011, siehe: ivarjacobson.com/use_case2.0_ebook/

[Jac92] I. Jacobson, Object- Oriented Software Engineering – A Use Case Driven Approach, Addison Wesley 1992

[Pal02] S.R. Palmer, M. Felsing, A Practical Guide to Feature-Driven Development, Prentice Hall 2002

[Pat12] Blog von J. Patton, siehe: agileproductdesign.com/blog/the_new_backlog.html

[Poh09] K. Pohl, C. Rupp, Basiswissen Requirements Engineering, dpunkt.verlag 2009

[REC12] Website der REConf 2012 mit allen Vorträgen, siehe: 2012.reconf.de/

[Rup10] C. Rupp, Eine Streitschrift – eine Provokation – jenseits von Lösungen, in: SQ Magazin Dezember 2010