



□ Thomas Rümmler

(thomas.ruemmler@aitag.com)

ist Senior Software Consultant und Projektleiter bei AIT sowie Mitglied des AIT-TeamSystemPro-Teams. Sein Arbeitsschwerpunkt liegt auf Application Lifecycle Management. Er hilft Unternehmen, ihre Software unter Einsatz des Microsoft Visual Studio Team Foundation Servers effizienter zu entwickeln. Seine Erfahrung gibt er als Autor des TFS-Blogs und Sprecher im Microsoft ALM-Umfeld weiter.

Agilität in der Produktentwicklung: Wie agile Softwareentwicklung und klassisches Produktmanagement zusammenpassen

Gegenwärtig kann man den Eindruck gewinnen, dass man mit einem nicht-agilen Vorgehen in der Softwareentwicklung nicht mehr wettbewerbsfähig arbeiten kann. Doch so schön sich die Welt der Product Backlogs, der ungestörten Sprints und selbstbestimmten Teams auch anhört, so gibt es in der praktischen Umsetzung doch erhebliche Unterschiede und Herausforderungen. Speziell in der Entwicklung von Produkten, bei denen Software lediglich eine Komponente darstellt, wie z. B. bei industriellen Maschinen, treffen hier Welten aufeinander. Produktlebenszyklen von 10 bis 20 oder mehr Jahren und lange Beschaffungszeiten stehen Sprints von wenigen Wochen gegenüber. Während das Produktmanagement auf hohem Abstraktionslevel plant, möchte die Softwareentwicklung auf Veränderungen reagieren können und sich erst sprintweise genau festlegen. Der Artikel zeigt, wie beide Welten zusammenspielen können. Außerdem wird dargestellt, wie man durch clevere Kombination die Vorteile von klassischem Produktmanagement und agiler Softwareentwicklung nutzen und sich dadurch einen Wettbewerbsvorteil verschaffen kann.

Problemstellung

Die HOCHSCHULE KOBLENZ hat jüngst den Verbreitungsgrad und die Auswirkungen von agilen Methoden in einer Studie untersucht. Eine grundlegende Erkenntnis ist, dass die Verbreitung agiler Methoden seit dem Jahre 2008 ein enormes Wachstum erfahren hat [Kom12, S. 28]. Insbesondere in der IT ist dies am Verbreitungsgrad zu sehen. So gaben in der Studie $\frac{3}{4}$ der befragten Unternehmen an, agile Methoden in Projekten mit IT-Fokus anzuwenden (vgl. [Kom12, S. 46]).

Während sich die agilen Methoden in der IT immer größerer Beliebtheit erfreuen, ist in der klassischen Produktentwicklung der Industrie häufig Produktmanagement nach Phasenmodellen und mit möglichst detaillierter „Upfront“-Planung vorzufinden. Das zumindest im deutsch-

sprachigen Raum wohl bekannteste klassische Modell ist das V-Modell (V-Modell 97 bzw. dessen Weiterentwicklung V-Modell XT). Doch unabhängig von diesem Modell sind Phasenmodelle, wie in [Abbildung 1](#) dargestellt, nicht ungewöhnlich.

Auf der einen Seite sind vom Produktmanagement eine sorgfältige Anforderungsanalyse sowie die Planung des Markteintrittes und die damit verbundenen Ziele von elementarer Bedeutung. Auf der anderen Seite ist insbesondere die Softwareentwicklung von einer enormen

Geschwindigkeit technischer Weiterentwicklungen getrieben. Ein User Interface, welches einst entworfen wurde, kann bei seiner Umsetzung bereits wieder optisch veraltet wirken. In der Entwicklung muss man auf solche Veränderungen reagieren können, um im Wettbewerb zu bestehen. Diese Flexibilität muss der Prozess hergeben.

Produkt- vs. Projektentwicklung

In der Produktentwicklung, insbesondere im industriellen Umfeld von Maschinen, sprechen wir über einen Produktlebenszy-

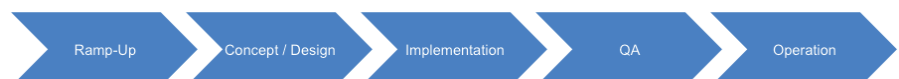


Abb. 1: Klassisches Phasenmodell

klus von mehreren Jahren, teils von Jahrzehnten. Hier ist ein diametraler Trend zu beobachten. Technologische Rahmenbedingungen ändern sich immer schneller. Der technische Fortschritt, aber auch der Druck durch innovative Startups, die frühzeitig am Markt auftreten, sind sicherlich nur einige Gründe. Dem gegenüber steht laut VDMA (VERBAND DEUTSCHER MASCHINEN- UND ANLAGENBAU E.V.) eine Trendwende in der Lebensdauer, die sich in der Zeit von 2007 bis 2012 von 13,1 auf 14,9 Jahre erhöht hat (vgl. [VDMA12, S. 13]).

Doch warum gibt es eigentlich diesen Widerspruch zwischen klassischer Produktplanung und agilem Projektmanagement? Um dieser Frage nachzugehen, ist ein Blick in einige Definitionen der agilen Welt hilfreich.

Nach SUTHERLAND und SCHWABER ist Scrum ein Framework bestehend aus den Elementen *Rollen*, *Events* und *Artefakten*. Die Rollen *Product Owner*, *Development Team* und *Scrum Master* sowie die Events *Sprint Planning Meeting*, *Sprint*, *Daily Scrum Meeting*, *Sprint Review* und *Sprint Retrospektive* zielen darauf, ein Arbeitsumfeld zu schaffen, in dem viel Transparenz und gute Kommunikation herrschen. Die Artefakte *Product Backlog*, *Sprint Backlog* und *Increment* repräsentieren die anstehende Arbeit bzw. den zu schaffenden Wert und ermöglichen eine Planung auf der Ebene einzelner Sprints [SaS11].

Wie bereits während des kurzen Ausflugs in den Scrum Guide zu sehen ist, bieten agile Vorgehensweisen Hilfsmittel für die Projektarbeit. Eine detaillierte Prozessbeschreibung ist dies jedoch nicht. So verwundert es auch nicht, dass man im Projektalltag verschiedenste Implementierungen agiler Modelle vorfindet.

Warum sollte man sich auch nicht die Bestandteile des Frameworks herausnehmen, die gut zum eigenen Unternehmen passen bzw. Teile davon anpassen und auf die eigenen Bedürfnisse zuschneiden? Prozess-Tailoring ist schließlich nicht erst mit agilen Vorgehensmodellen erfunden worden.

Doch dem stehen mitunter strenge Richtlinien im industriellen Umfeld gegenüber. Hier gibt es markt- und branchenspezifisch noch einige Unterschiede, jedoch gelten i. d. R. für eine Maschine strengere Vorschriften und Prozessdefinitionen, als bei einer Handy-App im Consumer-Bereich.

Dies ist unter anderem dadurch begründet, dass ein Update eines Maschinenparks so aufwendig sein kann, dass der ursprüngliche Deckungsbeitrag aufgebraucht wird. Während im letzten Fall die Time-to-Market überlebensnotwendig sein kann, hat ein intensives Risikomanagement im Beispiel der Maschine meist Vorrang. Deshalb sind insbesondere in Industrieunternehmen Referenzmodelle vertreten, wie sie CMMI bietet, die speziell zur Prozess- oder allgemeiner zur Unternehmensverbesserung beitragen [CMMI13].

Noch strenger sind die Auflagen im medizinischen Bereich. Dort haben Behörden, wie die amerikanische FDA (U.S. Food and Drug Administration) [FDA13], eine nicht zu unterschätzende Macht. Die Einhaltung der zu erfüllenden Richtlinien kann über die Zulassung auf einem Markt bestimmen und letztlich von existenzieller Bedeutung sein.

Es wird deutlich, dass die Spannung zwischen klassischer Produktentwicklung und agilem Projektmanagement von mehreren Seiten aufrechterhalten wird. Die Produktplanung, welche eine Vorausschau auf kommende Produktversionen mit einem größeren zeitlichen Horizont benötigt, ist hierbei ein wesentlicher Faktor. Ein weiterer wichtiger Aspekt sind vorgeschriebene Verordnungen und Verbindlichkeiten, die in streng regulierten

Branchen ein durchaus enges Korsett darstellen können.

Lösungsansatz

Doch wie sind die beiden Welten miteinander kompatibel? Ein erster Schritt ist sicherlich, zu akzeptieren, dass es verschiedene Ansichten und Ansprüche gibt. Wenn man nun agil Software entwickeln möchte, die Software jedoch „nur“ Bestandteil des eigentlichen Produkts ist, hilft es, die Modelle zu kombinieren.

Zunächst sei erwähnt, dass trotz der oben beschriebenen Spannungen auch durchaus Gemeinsamkeiten bestehen. So ist Requirements Engineering im weiteren Sinne erforderlich, unabhängig davon, ob man einen Maschinenpark beliefert oder eine Webseite erstellt. In beiden Fällen sollten Wunsch, Erwartungshaltung und Bedürfnisse des Kunden wohl verstanden sein. Auch die Dokumentation der Arbeitsergebnisse ist stets hilfreich, wenngleich es dabei eine qualitativ große Spannbreite gibt.

So findet man in Scrum auf oberster Ebene Elemente im Product Backlog. Durch das Scrum Framework werden diese nicht detailliert beschrieben. Deshalb findet man den allgemeineren Begriff *Product Backlog Item* ebenso vor, wie den Terminus *User Story*, der bereits eine gewisse Art der Beschreibung eines solchen Elements vorgibt [Pic10, S. 48–53].

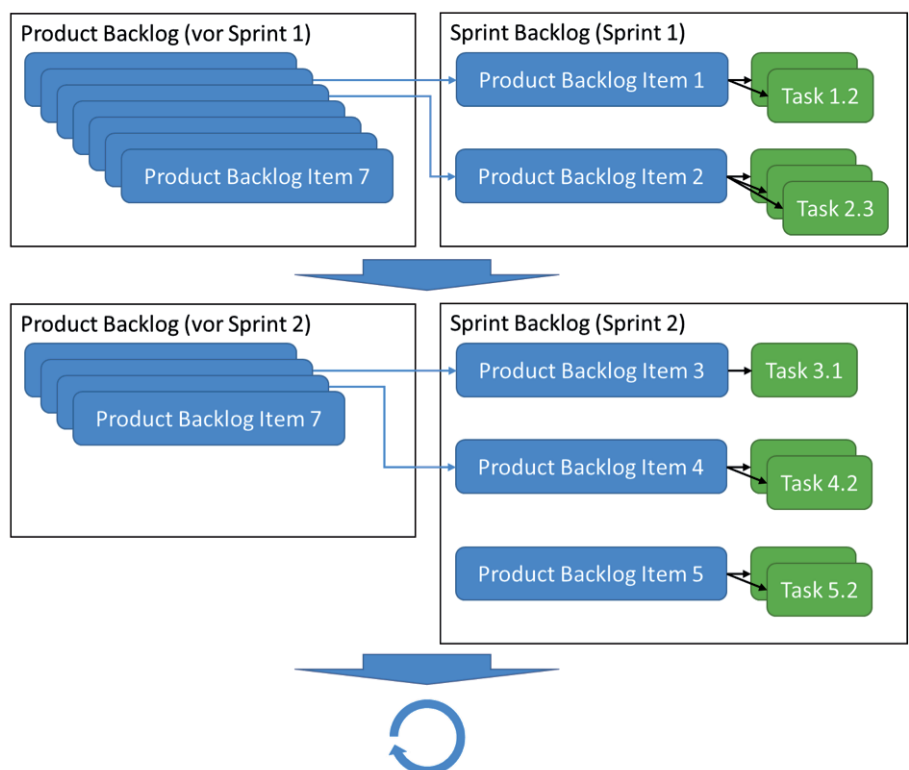


Abb. 2: Einige Geschäftsobjekte im Scrum-Prozess



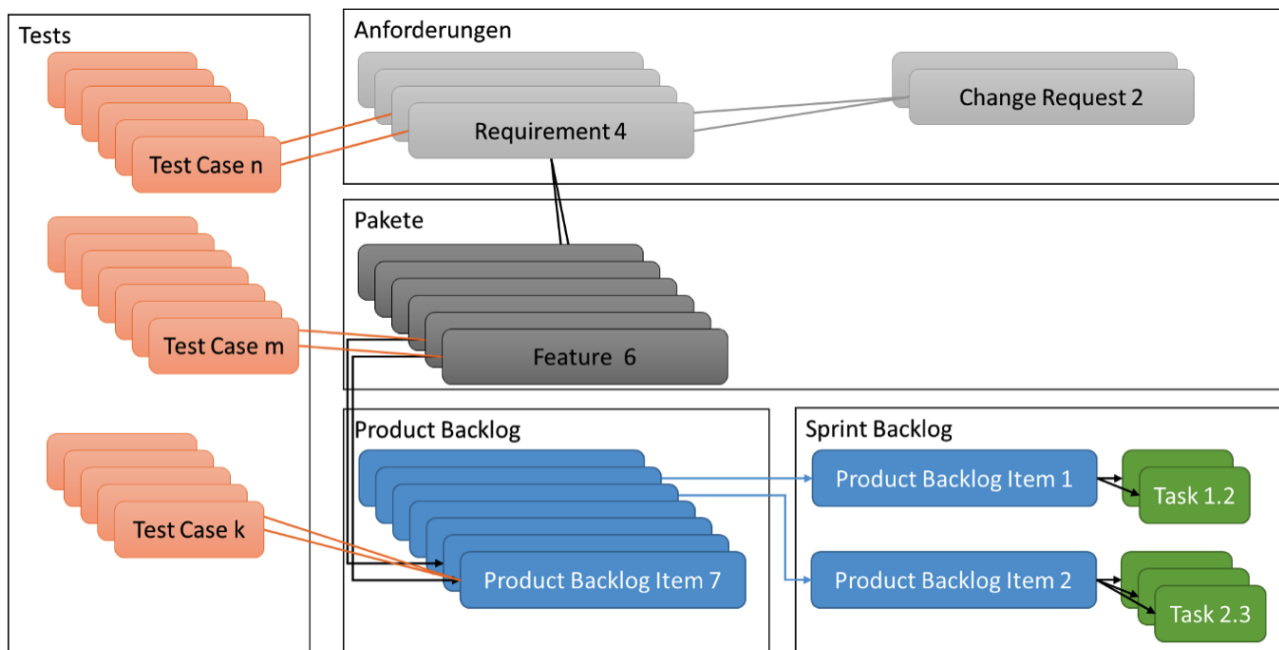


Abb. 4: Geschäftsobjekte des Scrum-Prozesses im Produktkontext

und Features (auch bekannt als magisches Dreieck) zu vermeiden, ist in diesem Falle der Umfang nicht fix vorgegeben.

Dadurch kann auf den unteren Ebenen in den einzelnen Teams agil gearbeitet werden. Auf der obersten Ebene, dem *Portfolio-Level*, wird die Richtung vorgegeben. Hier wird darüber entschieden, welche Systeme bzw. Anwendungen überhaupt zielführend sind [Lef11].

Wenn man diesen grundlegenden Gedanken aufgreift, wird deutlich, dass man in Richtung der langfristigen Planung (*Portfolio-Level*) die agilen Methoden in ihrer Reinform verlässt. LEFFINGWELL führt dafür die *Business Epics* sowie die *Architectural Epics* ein. Das agile Modell ist angereichert worden. Man benötigt also zusätzliche Geschäftsobjekte für diesen Teil der Planung.

LEFFINGWELL führt hierfür *Epics* auf oberster Ebene und *Features* auf *Program-Level* ein. Dies angewendet auf das Modell aus [Abbildung 2](#) in Verbindung mit Elementen des klassischen Requirements Engineerings lässt den Schluss auf ein erweitertes Objektmodell zu, welches in [Abbildung 4](#) dargestellt ist.

Dieses erweiterte Modell bietet die Möglichkeit, die Lücke zwischen Produkt und agilem Projekt zu schließen. Requirements Engineering kann auf oberster Ebene durchgeführt werden und ist zunächst nicht direkt von agilen Teams auf tieferen Levels beeinflusst. Jedoch sind hierbei verschiedene Varianten möglich.

Es obliegt wieder der konkreten Imple-

mentierung, ob man das Vorgehen in Phasen abbildet, in denen zunächst Requirements und dann Features entwickelt werden, um sie innerhalb der dritten Phase iterativ zu realisieren. Alternativ kann man auch zunächst die Must-Have-Requirements verfeinern, diese umsetzen und das Feedback in das weitere Requirements Engineering und das Herunterbrechen in Features einfließen lassen.

Dieses Modell bringt einen weiteren Vorteil mit sich. Man kann auf allen Ebenen, die man übrigens auch als Ebenen des V-Modells betrachten kann, Test Cases erfassen. Durch die Verwendung der Change Requests sowie der sorgfältigen Definition von Zustandsmodellen hinter den einzelnen Objekttypen kann sichergestellt werden, dass die Requirements stets den aktuellen Zustand des gesamten Produkts darstellen. Somit werden auch weitere Auswertungen, wie z. B. eine Impact-Analyse, ermöglicht.

Fazit

Die verschiedenen Paradigmen von Produktplanung und agiler Softwareentwicklung bringen durchaus Spannungspotenzial mit sich. Jedoch sind auf den ersten Blick die beiden schwer zusammenzubringenden Welten durchaus kompatibel. Durch geeignete Schnittstellen und Übergabepunkte in den Prozessen kann auf beide Vorgehensweisen Rücksicht genommen werden. Die damit verbundene Durchgängigkeit erlaubt eine sehr gute Nachvollziehbarkeit. In Abhängigkeit der ein-

gesetzten Werkzeuge kann man somit die Durchgängigkeit vom Requirement bis zum Code erreichen.

Des Weiteren kann man durch die vorher beschriebene Kombination weitere Vorteile ausnutzen. Da bei der Umsetzung in Sprints durchaus Entscheidungsspielraum für die Teams besteht, kann man von der Möglichkeit später Entscheidungen profitieren (Decide as late as possible, eines der Lean-Prinzipien). Solch einer späten Entscheidung liegen naturgemäß mehr Informationen vor, die wiederum zum Wettbewerbsvorteil führen können.

Jedoch verbleiben auch offene Punkte. Da bei Scrum die Schätzung des Aufwands in Stunden auf Task-Ebene erst relativ spät und nicht im Voraus für das gesamte Projekt geschieht, wird eine klassische Ressourcenplanung erschwert. Hier ist es erforderlich, die „Velocity“ von erfahrenen Teams herauszufinden, d. h. wie viele Story Points durchschnittlich in einem Sprint realistisch sind. Durch Bewertung der *Product Backlog Items* mit Story Points ist somit ein Forecast möglich, um abzuschätzen, nach welchem Sprint eine bestimmte Funktionalität vorhanden sein wird.

Auch die Toolseite trägt einen entscheidenden Anteil am Erfolg. Die Verwendung von Werkzeugen mit niedriger Eintrittsbarriere und intuitiver Bedienung verhilft zu Akzeptanz in den Teams. Über den gesamten Product Lifecycle hinweg entsteht jedoch meist eine Werkzeugkette, die wiederum integriert werden muss.

Im speziellen Beispiel der Softwareentwicklung im industriellen Produktkontext treffen typischerweise Werkzeuge aus dem PLM (Product Lifecycle Management) und ALM (Application Lifecycle Management) aufeinander. Diese zu integrieren ist eine technische wie auch eine organisatorische Aufgabe, die noch einiges Potenzial zur Optimierung und damit Effizienzgewinnen mitsichbringt.

Benötigen Sie Unterstützung bei der Implementierung der Entwicklungsprozesse in Ihrem Unternehmen? Verwenden Sie den Team Foundation Server als Application Lifecycle Management-Werkzeug? Dann sprechen Sie uns an: info@aitgmbh.de. ■

Literatur

[Kom12] A. Komus, "Ergebnisbericht (Langfassung) Studie: Status Quo Agile. Verbreitung und Nutzen agiler Methoden," 2012.

[VDMA12] VDMA, "VDMA-Kennzahlen vergleichen, verstehen, verändern 2012. Entwicklung und Konstruktion," Frankfurt a.M., 2012.

[Sa511] K. Schwaber and J. Sutherland, "The Scrum Guide," Scrum.org, no. October, p. 16, 2011.

[CMMI13] CMMI Institute, "CMMI Institute – CMMI Solutions," 2013. [Online]. Available: <http://cmmiinstitute.com/cmmi-solutions/>.

[FDA13] FDA, "U.S. Food and Drug Administration Home Page," 2013. [Online]. Available: <http://www.fda.gov/>.

[Pic10] R. Pichler, Agile Management with Scrum. Creating Products that Customers Love. Boston, 2010.

[Lef13] D. Leffingwell, "Scaled Agile Framework," 2013. [Online]. Available: <http://scaledagileframework.com/>.

[Lef11] D. Leffingwell, Agile Software Requirements. Lean Requirements Practices for Teams, Programs, and the Enterprise. Boston, 2011.