TRENDTHEMA „AGILITÄT”: WAS DIE EXPERTEN SAGEN

OBJEKTspektrum:
Agile Verfahren sind derzeit einer der ganz großen Hypes. Ist diese Entwicklung aus Ihrer Sicht substanziell oder lediglich eine Blase?

Gernot Starke: Von wegen Blase: Verfahren wie Scrum gibt es schon einige Jahre – und sind in vielen Unternehmen nachweislich produktiver als andere Entwicklungsansätze. Agile Verfahren sind daher eindeutig substanziell – außer vielleicht in Festpreis- oder Outsourcing-Projekten sowie in sehr großen und prozesslastigen Organisationen.

Johannes Mainusch: Insoweit es um die Verbesserung der Zusammenarbeit bei der Entwicklung und im Betrieb von Produkten geht, stehen wir hier aus meiner Sicht tatsächlich vor einer substanziellen Veränderung. Gerade das Arbeiten im IT-Umfeld ist voller kommunikativer Herausforderungen und Fallstricke. So gibt es z. B. bei einer „Erfolgsrate” klassischer IT-Projekte von 30 % noch viel Luft nach oben.

Wilhelm Schäfer: Wenn ich mit dem Flugzeug fliege, dann hoffe ich, dass die Steuerungssoftware nicht agil entwickelt worden ist. Ich bin in diesem Punkt eher Formalist. Als Passagier eines Flugzeugs bin ich über formale Change-Management-Verfahren sehr froh, die gewährleisten, dass Änderungen in der Software von 4 bis 5 Leuten überprüft werden und das diese Prüfung auch entsprechend dokumentiert wird. Mehr auf den Punkt gebracht: Viele Praktiken der agilen Verfahren sind natürlich sinnvoll: Das Vier-Augen-Prinzip z. B. ist ein starkes Qualitätsprinzip. Die Tatsache, dass Fachleute und Entwickler gemeinsam vor dem Rechner sitzen und entwickeln, ist aus meiner Sicht dagegen nicht wirklich förderlich. Ich vermisse in den Diskussionen um agile Verfahren die Bedeutung von Architekturen. Agilität darf auf keinen Fall zur Ausrede werden, dass wesentliche Architekturprinzipien nicht eingehalten werden müssen.

Arne Pott: Generell haben wir in den letzten Jahren sehr engagiert neue Entwicklungsprozesse auf CMMI-Basis in der Breite eingeführt. Hier müssen wir uns bestimmt erst einmal stabilisieren, bevor wir die nächste Veränderung in der Breite einführen. Bei bestimmten Aufgabenstellungen können uns agile Verfahren oder der Einsatz agiler Elemente aber helfen und besser mit der Kundenseite zusammenzuarbeiten. Gerade in Anwendungsgebieten, in denen wir rasch auf Marktveränderungen zu reagieren haben, sehe ich daher ein großes Potenzial für agile Verfahren. Sind auf der anderen Seite der Auftragsumfang, der Tätigkeitskatalog und der Einführungszeitpunkt klar festgelegt, bieten meines Erachtens unsere bisherigen Verfahren Vorteile. In diesem Sinne sehe ich agile Verfahren als neues, weiteres Instrument im Werkzeugkasten der Softwareentwicklung. Pilotprojekte durchzuführen ist einfach. Schwieriger ist es, hierbei alle Mitarbeiter und Projektleiter mitzunehmen.

mehr zum thema:
http://www.youtube.com/watch?v=ha8k2GiRdho

DAS BESTE AUS ZWEI WELTEN: AGILE ENTWICKLUNG UND NICHT-AGILE KUNDENPROJEKTE VERBINDEN

Scrum ist in der Softwareentwicklung „in”. Qualitätsnormen oder Prozesse wie das V-Modell XT stellen oft zusätzlich einzuhaltende Notwendigkeiten dar. Die Modelle setzen aber andere Akzente: direkte Kommunikation vs. viele Dokumente, ein flaches Product Backlog vs. eine hierarchische Spezifikation, viele Sprints vs. wenige Iterationen mit wasserfallartigen Meilensteinen. In diesem Artikel geht es um eine sinnvolle Kombination beider Modelle.

▶ der autor

Thomas Klingenberg
[E-Mail: T.Klingenberg@microTOOL.de]
arbeitete als Softwareentwickler, Projekt- und Entwicklungsleiter. Seine Erfahrung aus über 20 Jahren Beschäftigung mit Softwaretechnik und -management nutzt er für die Unterstützung von Organisationen bei der Optimierung ihrer Entwicklungsprozesse.

Agile Prozessmodelle – allen voran Scrum – sind für Entwicklungsorganisationen mittlerweile eher die Regel als die Ausnahme. Arbeitet eine Organisation, die Scrum einsetzt, aber in einem Geschäftsumfeld, das klassische Modelle vorgibt, muss man sich fragen, wie beide Modelle miteinander vereint werden können. In meiner Beratungstätigkeit für mehrere Kunden, die ganz unterschiedliche Produkte herstellen, tauchte in den letzten Monaten immer wieder genau diese Frage auf. Die Lösungen,

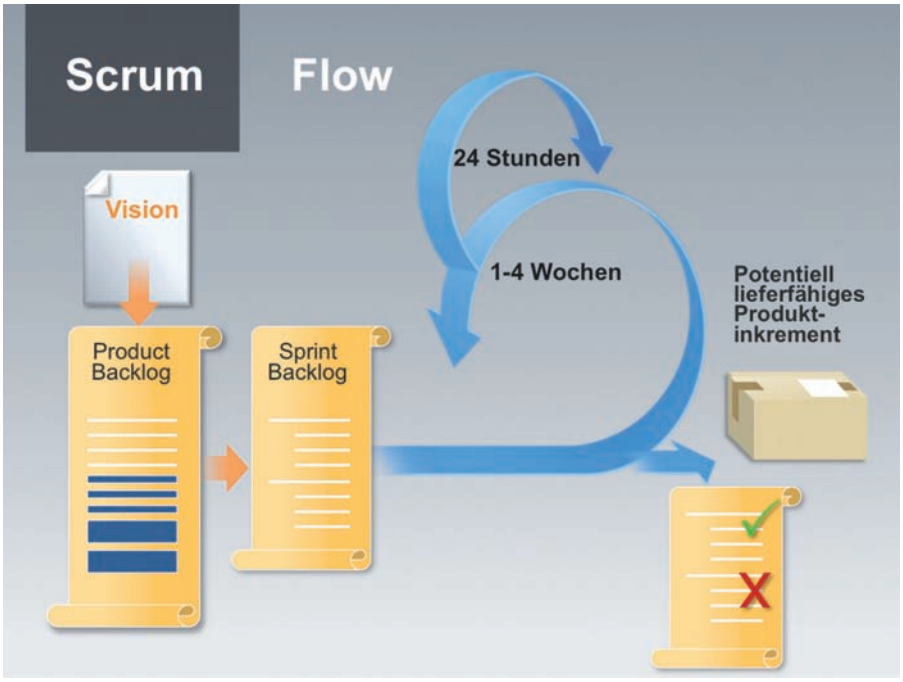

Abb. 1: Scrum-Flow.

die sich dabei ergeben haben, möchte ich hier vorstellen. Allen Kunden war gemeinsam, dass sie Produkte herstellen, die sowohl aus Geräten als auch aus Software bestehen.

Scrum aus Überzeugung

Scrum ist heute der am weitesten verbreitete Vertreter der agilen Prozessmodelle und kommt auf einen Marktanteil von 58 %. Die Kombination von Scrum mit XP, welches Scrum ideal um softwarespezifische Praktiken ergänzt, kommt auf weitere 17 % (vgl. [Ver10]). Der Trend ist zunehmend, wie ein Blick in ältere Erhebungen zeigt. Zur Popularität von Scrum tragen sicher zwei Faktoren bei, die dafür sorgen, dass viele Entwickler es aus echter Überzeugung einsetzen:

- Scrum ist ein sehr schlankes Modell.
- In einer geschickten Kombination verschiedener Techniken bietet Scrum für Entwicklungsprojekte eine wirklich gute Unterstützung.

Seine Schlankheit demonstriert das Modell bereits bei den Rollen: Hier werden lediglich der *Product Owner (PO)* als Ergebnis-Verantwortlicher, der *ScrumMaster* als Prozess-Verantwortlicher und das *Team* als Umsetzungs-Verantwortlicher definiert. Scrum erleichtert den Aufbau einer ergebnisorientierten und durch direkte Kommunikation geprägten Arbeitsweise. Statt vieler Dokumente, wie man sie in anderen Prozessmodellen findet, legt Scrum mit dem *Product Backlog* nur ein sehr einfaches Instrument zur Steuerung des Projekts fest.

Das Product Backlog stellt den Ersatz für den sonst üblichen Projektplan dar, da es die noch offene Arbeit beschreibt. Es ersetzt auch jegliche Art von Spezifikationsdokumenten, da es die einzige Stelle zur Beschreibung von Inhalten ist. Diese Beschreibung legt aber keinen Wert auf eine umfangreiche Dokumentation, sondern dient den Beteiligten primär als Gedächtnisstütze. Wichtig ist die intensive Kommunikation zwischen dem PO und dem Team über das gemeinsame Verständnis der einzelnen User-Storys, die das Backlog bilden. Das Team liefert die Aufwandsschätzungen, der PO definiert durch seine Priorisierung die Reihenfolge der Abarbeitung. Die Umsetzung der User-Storys nimmt das Team dann in den Sprints vor, die mit ihrer konstanten Länge den Grundrhythmus des Projekts angeben.

Jeder Sprint beginnt mit einem Planungs-Meeting, in dem das Team die Menge der Arbeit für den Sprint aus der durch den PO priorisierten Liste auswählt. Am Ende des Sprints demonstriert das Team dem PO die geleistete Arbeit. Die während des Sprints täglich gleichartig stattfindenden *Daily Scrums* dienen dem Team zur internen Abstimmung. Im Rahmen einer Sprint-Retrospektive wird mögliches Verbesserungspotenzial an der eigenen Arbeitsweise identifiziert. **Abbildung 1** zeigt den Scrum-Flow in der Übersicht.

Scrum ist ein kollaborationsorientiertes Modell: Die direkte Kommunikation steht im Vordergrund, Meetings sind definiert und Geschriebenes ist nur ein untergeordnetes Hilfsmittel.

V-Modell XT als Vorgabe

Einen ganz anderen Ansatz verfolgt das V-Modell XT (vgl. [Kuh09]). Hier geht es um Kommunikation über die Zeit, d. h. um das dauerhaft nachvollziehbare Aufschreiben von getroffenen Entscheidungen. In diesem Artikel verwende ich das V-Modell XT (**siehe Abbildung 2**) als typischen Stellvertreter dieser Art von Modellen. Die Aussagen gelten aber genauso für alle V-Modell-artigen Modelle, wie sie beispielsweise von den Reifegradmodellen *SPICE (Software Process Improvement and Determination Capability)* und *CMMI (Capability Maturity Model Integration)* impliziert werden, oder auch für Modelle, die Vorgaben von Qualitätsnormen erfüllen müssen, wie z. B. der IEC62304 in der Medizintechnik.

Das V-Modell XT ist ein dokumentationsorientiertes Modell. Das Geschriebene steht im Vordergrund, Vorgaben über Meetings und die Art der Zusammenarbeit werden kaum gemacht. Durch die entstehende Dokumentation können auch nach Jahren noch alle Projektentscheidungen nachvollzogen werden – auch von anderen Personen als den ursprünglichen Projektbeteiligten. Dieser Mehraufwand hat seinen Wert, insbesondere im Bereich der Dokumentation von Architektur, Funktionsumfang und Test des entwickelten Produkts, wenn dieses weiterentwickelt bzw. gepflegt werden muss. Die dauerhafte Nachvollziehbarkeit von Entscheidungen und die Archivierung von Informationen, die bei der Produktentwicklung entstehen, stellen eine wesentliche Motivation für den Einsatz des V-Modell XT oder vergleichbarer Modelle dar. So ist es kein Zufall, dass Auftraggeber häufig gerade diese Modelle einem Entwicklungsteam vorgeben, wenn diese ein großes Interesse an Dokumentation haben.

Im V-Modell XT erfolgt die Spezifikation des Systems über mehrere Ebenen. Auf

jeder Ebene werden die Anforderungen und die Architektur mit einer bestimmten Granularität beschrieben und durch die darunter liegende Ebene weiter verfeinert. Auf diese Weise wird das System auf der linken Seite des „V" hierarchisch herunter gebrochen und anschließend auf der rechten Seite entwickelt, erstellt und überprüft. Für Anforderungen bedeutet das: Aus einer ursprünglichen Kundenanforderung im Lastenheft werden eine oder mehrere Anforderungen im Pflichtenheft. Daraus werden dann auf den weiteren Ebenen technische Detailanforderungen an einzelne Komponenten des Systems. Die Nachvollziehbarkeit der Entstehung von Anforderungen muss dabei über alle Ebenen gewährleistet sein (Stichwort „Rückverfolgbarkeit" bzw. „Traceability"). Testfälle sichern auf jeder Ebene die Testbarkeit der jeweiligen Komponente bzw. des Subsystems. Während einer Iteration durch das „V" fordert das V-Modell XT, dass eine Reihe von Meilensteinen formal bestätigt wird. Zu diesen so genannten Entscheidungspunkten muss jeweils eine genau definierte Menge an Dokumenten bzw. Komponenten vorliegen, die auf der rechten Seite des „V" erstellt und getestet wurden.

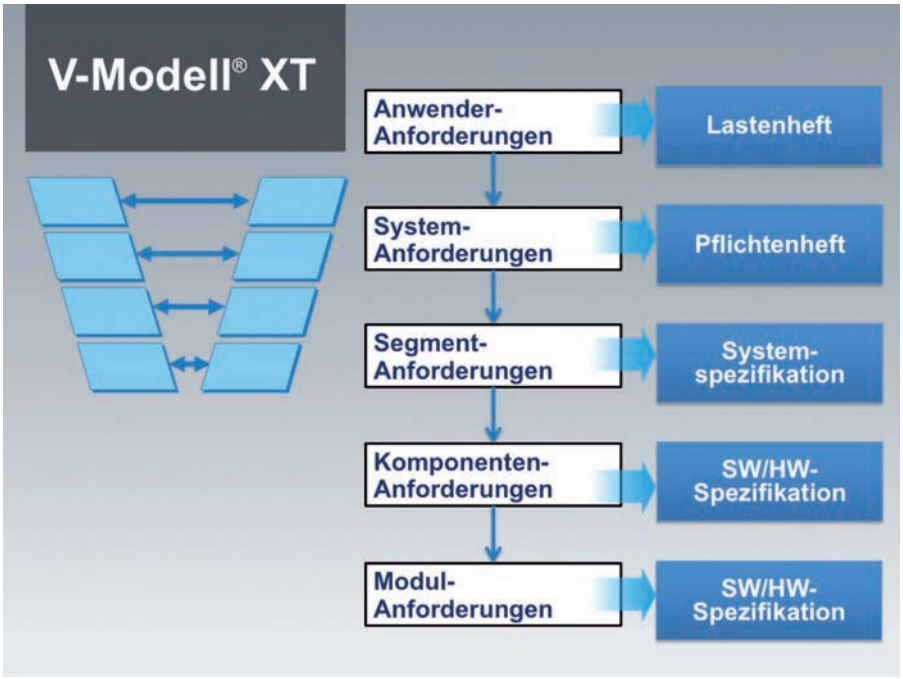

Abb. 2: V-Modell XT.

Ein Projekt mit zwei Prozessmodellen

Was ist nun zu tun, wenn die Entwicklung gerne nach Scrum arbeiten möchte, der Auftraggeber aber die Einhaltung des V-Modell XT oder eines ähnlichen Modells fordert? Wie beschrieben, dient der Einsatz der beiden Modelle sehr unterschiedlichen Zielen. Das eröffnet aber auch die Möglichkeit, durch eine sinnvolle Kombination beiden Seiten gerecht zu werden.

Für ein einzelnes Projekt bedeutet es zunächst, dass das Projekt die Auftraggeber/Auftragnehmer-Schnittstelle des V-Modell XT gegenüber seinem Auftraggeber erfüllen muss. Diese Schnittstelle regelt, wer im Projekt wann was zu liefern hat. Insbesondere gehören hierzu auch Dokumente, die das V-Modell XT zu bestimmten Entscheidungspunkten fordert. Das V-Modell XT bezeichnet einen Durchlauf durch das „V" der Systementwicklung als Iteration. Das Ergebnis ist eine Lieferung an den Kunden. Zwar sieht auch Scrum das Ergebnis eines Sprints als „potenziell lieferbares Produktinkrement" – also als eine Art Lieferung. Daraus sollte man aber besser nicht den Schluss ziehen, dass eine V-Modell-Iteration einem Sprint entsprechen könnte. Sprints sind recht kurz (z. B. zwei Wochen), die V-Modell-Iteration enthält jedoch eine Folge von Entscheidungspunkten, die einen kleinen Wasserfall bilden (Spezifikation – Realisierung –Test – Lieferung), der unmöglich so hoch-frequent durchlaufen werden kann. Besser ist es, eine V-Modell-Iteration als ein Release zu betrachten, das durch viele Sprints erreicht wird.

Die Erstellung der notwendigen Dokumente wird als zu leistende Arbeit mit in das Product Backlog aufgenommen. Für Managementdokumente, wie das Projekt- oder das QS-Handbuch, ist das recht einfach. Diese Dokumente werden ohnehin zu Beginn erstellt.

Spezifikationsdokumente beschreiben die Funktionalität der zu erstellenden Lösung. Sie bestehen typischerweise aus den Anforderungen der jeweiligen Ebene. Wir verwenden also nicht mehr nur das Product Backlog als Instrument zur Definition des Funktionsumfangs, sondern es wird ein Anforderungsmanagement-Werkzeug zur Erstellung einer dauerhaften Spezifikation eingesetzt. Das Product Backlog wird also reduziert auf die Rolle eines Instruments zur Projektplanung und -steuerung. Eine User-Story bezieht sich nun auf die Realisierung einer Anforderung. Im weiteren Verlauf kann eine weitere User-Story eine Änderung an dieser Anforderung realisieren.

Für Spezifikations- und Architekturdokumente stellt sich natürlich die Frage, zu welchem Zeitpunkt diese überhaupt erstellt werden können. Damit geht die Frage einher, wann die zugehörigen Entscheidungspunkte als erreicht betrachtet werden können. Generelle Antworten auf diese Fragen gibt es nicht, vielmehr hängen diese von der Stabilität der enthaltenen Informationen ab.

Solange Spezifikation und Architektur noch weitgehend unklar sind, kann man die passenden Dokumente schlecht erstellen und ergo auch den Meilenstein nicht erreichen. Es werden also zunächst weitere Sprints geplant und durchgeführt, bis sich das System hinreichend stabilisiert. Die extreme Variante, sich bis kurz vor Lieferung alle Optionen offen zu halten und auch die Spezifikation formal erst am Ende zu erstellen, ist aber sehr wahrscheinlich Unfug. Welches reale System ist so einfach umzubauen, dass man kurz vor der Lieferung noch alles ändern könnte? Die Dokumente werden also bei hinreichender Klarheit über die Anforderungen und die Architektur des Systems erstellt. Gegebenenfalls können sie bei späteren Änderungen noch aktualisiert werden. Ein Werkzeug mit einem guten Dokumentengenerator macht dem Projektteam hier das Leben leichter.

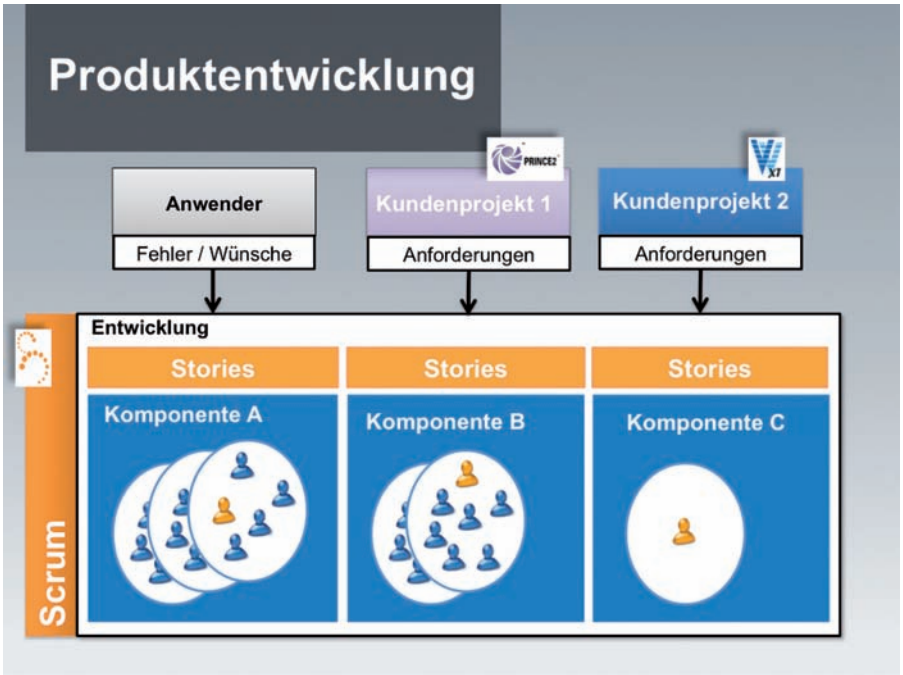

Abb. 3: Produktentwicklungsorganisation mit Kundenprojekten.

Agile Entwicklungsorganisation mit Kundenprojekten

Die Entwicklung eines komplexen Produkts oder Systems in einem einzelnen Projekt ist heute eher die Ausnahme. Zumindest bei vielen meiner Kunden geht es mittlerweile darum, bereits entwickelte Komponenten bzw. Lösungen in eine Produktentwicklung zu transformieren und für viele künftige Projekte nutzbar zu machen.

Daraus folgt die Zweiteilung einer Organisation in eine Entwicklungsgruppe, die für die langfristige Entwicklung und Pflege von Komponenten verantwortlich ist, und eine Projektgruppe, die die Kundenprojekte durchführt. In dieser Struktur ist Scrum das Modell für die Entwicklungsgruppe. Die Kundenprojekte können durchaus nach unterschiedlichen Modellen vorgehen, je nach Vorgaben des jeweiligen Kunden. Einen solchen Aufbau zeigt Abbildung 3.

Es geht also nicht mehr um eine Matrixorganisation, in der die Mitarbeiter aus der Linie in Projekte entsandt werden, sondern vielmehr um das geordnete Zusammenspiel von temporären Projektteams und dauerhaften Entwicklungsteams. Die Stabilität der Entwicklungsteams fördert die Teamarbeit und ist damit ganz im agilen Sinne von Scrum. Wie die Entwicklungsteams organisiert sind – ob als Feature- oder Komponenten-Teams oder als eine Mischform aus beiden – hängt stark von der Struktur des Produkts ab. In Abbildung 3 bin ich von reinen Komponententeams ausgegangen, die bei meinen Kunden derzeit noch überwiegen. Es gibt also pro Komponente bzw. Subsystem je ein Team, das dauerhaft für dessen Weiterentwicklung zuständig ist.

Die Entwicklungsteams führen jeweils ein eigenes Product Backlog mit den Storys, die sich auf ihre Komponente beziehen. Diese Storys werden wie gewohnt durch einen für diese Komponente verantwortlichen PO priorisiert und in Sprints abgearbeitet. Die POs aller Entwicklungsteams bilden gemeinsam ein Gremium, in dem übergeordnete Entscheidungen hinsichtlich Architektur, Techniken etc. getroffen werden. Komplexe Anforderungen, die sich auf mehrere Komponenten beziehen, müssen im Vorfeld analysiert und aufgeteilt werden. Dazu werden in den Kundenprojekten die Anforderungen je nach Art des verwendeten Prozessmodells detailliert analysiert. Wird das V-Modell XT als Modell für ein Kundenprojekt eingesetzt, so erfolgt die übliche hierarchische Zerlegung der Anforderungen. In der Regel wird die jeweils unterste Ebene der Anforderungen dann den Komponenten zugeordnet. Zu diesen Anforderungen werden User-Storys im Product Backlog des jeweiligen Entwicklungsteams angelegt. Abbildung 4 zeigt die verwendeten Konzepte auf der jeweiligen Ebene.

In einem Product Backlog mischen sich also User-Storys aus vielen Kundenprojekten, was eine ständige Abstimmung des PO dieser Komponente mit allen Projektleitern erfordert. Konflikte um die Prioritäten bleiben dabei nicht aus, sie müssen natürlich gelöst werden.

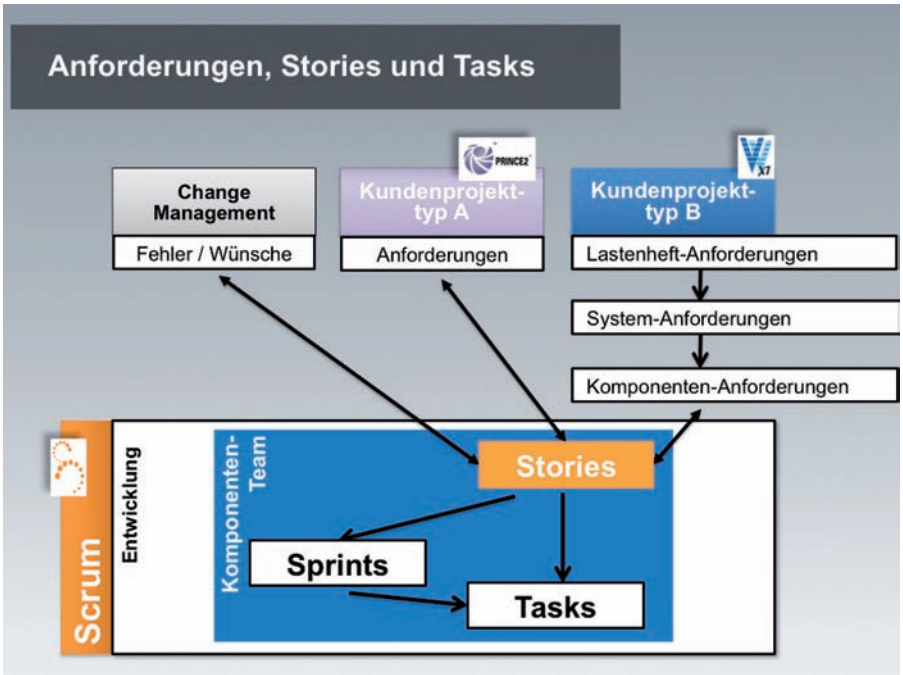

Abb. 4: Anforderungen, Storys und Tasks.

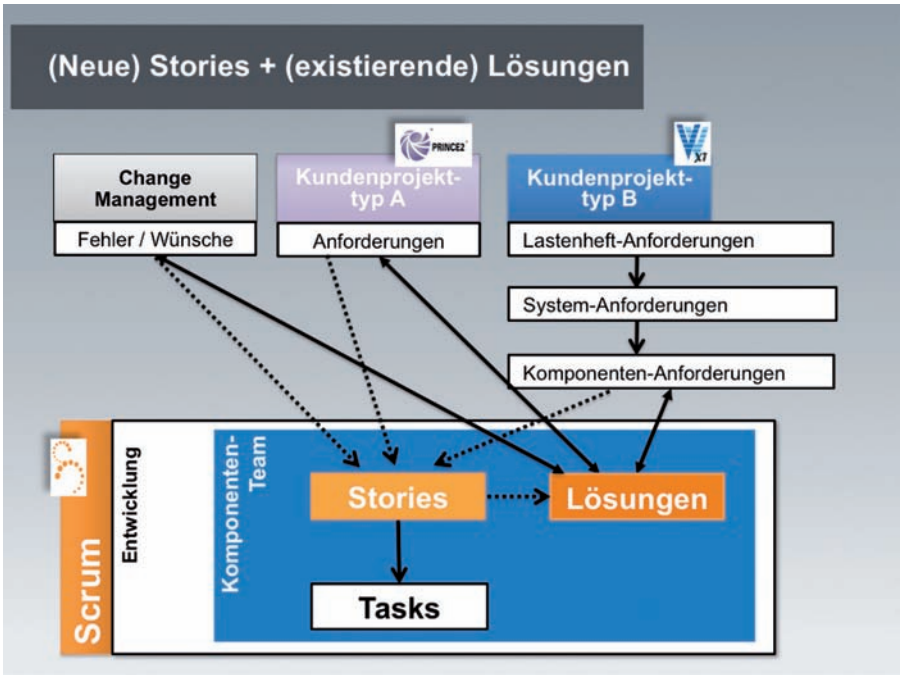

Abb. 5: Komponenten mit Features.

Zu jeder User-Story ist die Herkunft bekannt. Damit kann eine lückenlose Nachvollziehbarkeit über alle Ebenen erreicht werden: von der ursprünglichen Kundenanforderung bis zu den Storys und Tasks der Entwicklung. Sind diese mit einem Quellcode-Verwaltungssystem (wie z. B. „Subversion") verbunden, so reicht die Nachvollziehbarkeit (*Traceability*) bis in den Code.

Da zu jeder User-Story bekannt ist, aus welchem Kundenprojekt sie stammt, lassen sich Auswertungen über die Daten erstellen, die einem Projektleiter z. B. die Information liefern, wann – nach aktueller Planung – seine offenen Punkte erledigt sein werden. Die Transparenz, die das Product Backlog erzeugt, ist hier sehr hilfreich. Für alle Beteiligten ist erkennbar, welche anderen Arbeiten die Teams noch erledigen müssen, bevor man selbst an der Reihe ist. Dies erspart eine Menge an Nachfragen und vermeidet Hauruck-Aktionen. Die mittlere *Velocity* der Teams, also die Geschwindigkeit, mit der das Product Backlog abgearbeitet wird, bildet eine sehr gute Berechnungsgrundlage für die wahrscheinlichen Realisierungszeitpunkte der einzelnen Storys. Damit können für die Projektleiter dann die Zeitpunkte ihrer Storys ermittelt werden.

Wenn das entwickelte Produkt nicht völlig neu am Markt ist, gibt es bereits Anwender. Auch von diesen kommt Arbeit in die Product Backlogs der Entwicklungsteams in Form von Änderungswünschen oder Fehlermeldungen. In den Product Backlogs läuft also die gesamte offene Arbeit der Entwicklung zusammen. Diese Vollständigkeit bildet die Grundlage für eine bestmögliche Planung der Arbeit.

Komponenten mit Features und Anforderungen

Beim Aufbau einer Produktlinie aus Komponenten hat man es mit einer recht komplexen Situation zu tun. Die Komponenten wachsen über die Zeit in ihrem Funktionsumfang an. Nicht jede Anforderung aus einem Kundenprojekt ist wirklich neu und vieles kann bereits abgedeckt werden. Die bestehenden Fähigkeiten der Komponenten sind in ihrer Spezifikation dokumentiert. Ob man hierzu noch das Konzept der (mittlerweile umgesetzten) „Anforderung" verwendet oder ein eigenes Konzept wie z. B. das des „Features" oder der „Lösung" einführt, ist reine Geschmackssache. Wichtig ist, dass die Anforderungen des Kunden in den Projekten hinreichend analysiert werden und dann den bereits vorhandenen Features zugeordnet werden, sofern es sich um existente Fähigkeiten handelt. Nur das Delta der noch fehlenden Funktionalität bzw. Arbeit, die zur Integration von Komponenten nötig ist, wird dann zu neuen Storys. Abbildung 5 zeigt dies schematisch in der nun vollständigen Übersicht. Die mit den umgesetzten Storys erzeugten neuen Lösungen werden dann wieder dauerhaft mit den Anforderungen verknüpft.

Fazit

Der in diesem Artikel vorgestellte Ansatz einer Produktentwicklung aus Komponenten mit einer agilen Entwicklungsorganisation und damit verbundenen, nicht-agilen Kundenprojekten ist bei mehreren meiner Kunden mit nur leichten Abweichungen entstanden. Auch wenn diese kombinierte Vorgehensweise natürlich gegenüber einem rein agilen Ansatz einige Einschränkungen mit sich bringt, wie z. B. ein verzögertes Feedback durch weniger und dafür umfangreichere Releases, überwiegen die Vorteile deutlich. Die Kombination der besten Techniken aus agilen und dokumentenorientierten Prozessmodellen erfüllt externe Vorgaben und ermöglicht im Kern der Entwicklungsorganisation ein agiles Vorgehen. Die aus dem agilen Vorgehen entstehenden Vorteile hinsichtlich Effizienz, offener Kommunikation und Transparenz werden gut kombiniert mit den Vorteilen, die das Erstellen einer langfristig gültigen Dokumentation hinsichtlich Nachvollziehbarkeit über einen langen Zeitraum bietet. ■

Literatur & Links

[Kuh09] M. Kuhrmann et al., Das V-Modell XT: Für Projektleiter und QS-Verantwortliche, Springer 2009

[Ver10] VersionOne Inc., State of Agile Survey 2010, siehe www.versionone.com/pdf/2010_State_of_Agile_Development_Survey_Results.pdf