

BUSINESS AS USUAL?

SCRUM IN DER SAP-SOFTWAREENTWICKLUNG

Im November 2009 hat die zetVisions AG in der Softwareentwicklung Scrum eingeführt. In diesem Artikel beschreibt der SAP-Add-On-Hersteller die Erfahrungen und Ergebnisse dieser Einführung, bei der folgende Ziele im Fokus standen: Verbesserung der Effizienz in der Softwareentwicklung, realistischere und änderungsfreundlichere Gestaltung der Release-Planung und Verbesserung der Produktqualität. Besonderheiten der Softwareentwicklung in diesem Umfeld sind die hohe fachliche Spezialisierung der Software sowie die eingesetzte Technologie.

Die zetVisions AG ist ein mittelständisches Softwarehaus und entwickelt mit etwa 65 Mitarbeiterinnen und Mitarbeitern Softwarelösungen für das Beteiligungsmanagement. Dabei geht es um eine effektivere Verwaltung, ein besseres Controlling und letztendlich eine zielgerichtetere Steuerung von großen Beteiligungsportfolios bei teilweise mehr als 1.000 Tochterunternehmen bzw. Beteiligungen.

Das Angebot besteht aus drei verschiedenen Softwareprodukten, die teilweise über Akquisitionen zum Unternehmen gekommen sind und die alle in etwa den gleichen fachlichen Scope bedienen, aber auf unterschiedlichen Technologieplattformen basieren.

Aus Sicht des *Product Owners (PO)* gibt es ein für alle Produkte in großen Teilen thematisch ähnliches Product-Backlog sowie separate Entwicklungsteams, die sich meist produktspezifisch verteilen, aber auch themenzentriert zusammengesetzt werden. Zur Strategie des Unternehmens zählt es, alle Produkte gleichrangig weiterzuentwickeln und gleichzeitig Synergien über global verwendbare Komponenten herzustellen. Das ist in Teilbereichen gelungen, etwa in der visuellen Darstellung komplexer Unternehmensstrukturen, die für alle drei Produkte Verwendung findet und sukzessive integriert wurde (siehe **Abbildung 1**).

In der Organisationsstruktur gab es einen Gesamtentwicklungsleiter, drei Produktentwicklungsleiter und jeweils Komponentenverantwortliche innerhalb der Produkte. Dabei kam es zu Überschneidungen bei Spezialthemen. Beispielsweise ist ein wichtiges Gebiet der Bereich „Regulatorisches Meldewesen“, also eine fachliche Anforderung, die von einem Entwicklerteam für mindestens zwei

Produkte betreut wird. Technisch gesehen handelt es sich um:

- eine mit „Powerbuilder“, „Oracle DB“ und „BusinessObjects-Reporting“ betriebene Lösung
- eine auf „SAP NetWeaver 2004s“, „WebDynpros“ und „SAP BW“ basierende Lösung
- eine Java-Anwendung auf dem „MS SQL Server“ und „Actuate Reporting“

Mittlerweile werden alle drei Produkte im Scrum-Kontext weiterentwickelt. Aufgrund

- der singulären SAP-Architektur und -Technologie,
- der Teamgröße und der durch die Akquisition neuen Zusammensetzung der CIM-Teams,
- des geringeren Reifegrades des SAP Add-Ons CIM und
- des bei Weitem umfangreichsten Product-Backlog für CIM

stellt das SAP-basierte Produkt zetVisions CIM dabei eine besondere Herausforderung dar, weshalb sie im Folgenden im Mittelpunkt stehen wird.

Die beiden anderen Produkte unterscheiden sich in diesen Punkte deutlich. Dazu kommen auch noch abweichende organisatorische Aspekte, wie z. B. ein hoher Anteil an Arbeit im Home-Office. Deshalb lassen sich die Erfahrungen mit den drei Produkten auch nicht direkt miteinander vergleichen. Obwohl alle drei Produkte von derselben Firma weiterentwickelt und vertrieben werden, legt man großen Wert darauf, dass die gewachsenen Strukturen nicht mit Gewalt vereinheitlicht werden. Scrum bietet hier die Möglichkeit, dass unter-



Thorsten Deuter

[E-Mail: thorsten.deuter@zetvisions.com]

ist bei der zetVisions AG Leiter der Qualitätssicherung und der internen IT und seit Einführung von Scrum zusätzlich Scrum Master.



Jörg Brandeis

[E-Mail: joerg.brandeis@zetvisions.com]

ist bei der zetVisions AG Entwicklungsleiter für das SAP-basierte Produkt zetVisions CIM.

schiedliche Entwicklungskulturen unter einem gemeinsamen Prozessrahmenwerk bestehen bleiben.

Wie es vorher war

Die Lage vor der Scrum-Einführung in der Produktentwicklung ergab eine hohe Motivation, nach alternativen Prozessen zu suchen.

Release-Planung

Am Anfang des Release trafen sich Produktmanagement und Entwicklungsleiter, stimmten den Release-Umfang ab und nahmen eine grobe Kapazitätsplanung und Aufwandschätzung vor. Typischerweise war die Planung von vorneherein knapp kalkuliert, weil der Release-Termin durch die geplanten Kundenprojekte vorgegeben war und den Kunden viele Features versprochen worden waren.

Release-Entwicklung

Der Entwicklungsleiter verteilte die Aufgaben anschließend an einzelne Experten. Eine Abstimmung innerhalb der Entwicklungsteams erschien somit nicht besonders wichtig. Im Laufe des Entwicklungsprozesses wurden die Anforderungen mit dem zuständigen Produktmanager diskutiert. Als wesentliches Tool diente dabei eine Excel-Liste pro Thema, in der Verbesserungen und Fehler geführt wurden. Diese wurde vom einzelnen Entwickler abgearbeitet, während immer wie-

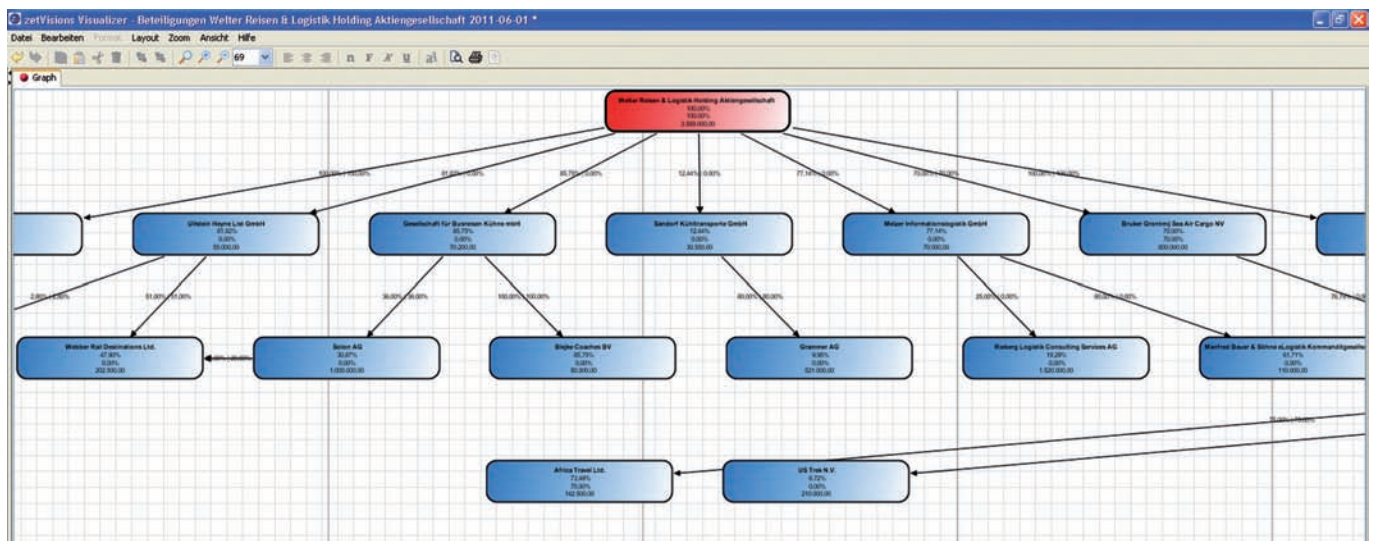


Abb. 1: Visualisierung von komplexen Konzernstrukturen mit dem OrgChartTool zetVisions Visualizer.

der neue Punkte aufgenommen wurden.

Nachdem ein Thema weitgehend fertiggestellt war, folgte für den Entwickler das nächste, wobei immer wieder Rückläufer der vorherigen Themen dazwischen kamen. Somit sammelte sich im Laufe der Release-Entwicklung eine immer größer werdende Anzahl von noch nicht abgeschlossenen Themen an.

Die einzelnen Entwickler waren in fachlicher und zum Teil auch in technischer Hinsicht stark spezialisiert, was einerseits eine sehr effiziente Entwicklung ermöglichte. Auf der anderen Seite gab es dadurch sehr schnell Engpässe in der Wartung, wenn ein Mitarbeiter wegen Krankheit oder Urlaub nicht verfügbar war.

Fertigstellung des Release und Auslieferung

Gegen Ende des Release standen mehrere Testwochen an, in denen neben der Qualitätssicherung (*Quality Assurance, QA*) auch die Mitarbeiter aus dem Support und dem Consulting den aktuellen Entwicklungsstand testeten. Dabei wurden ein große Anzahl an Verbesserungsvorschlägen gemacht und viele Fehler gefunden. In der heißen Phase der Fertigstellung wurden regelmäßig entweder der Release-Umfang reduziert und/oder der *Code-Freeze* verschoben.

Probleme mit dem Vorgehen

Aus der geschilderten Vorgehensweise ergab sich eine Reihe von Problemen:

- Die Entwickler standen häufig unter großem Arbeitsdruck, da die Excel-Listen teilweise schneller wuchsen, als sie abgearbeitet werden konnten.
- Die Abstimmung zwischen den Entwicklern war unzureichend und Abhängigkeiten wurden häufig zu spät erkannt.
- Wegen der Spezialisierung der Entwickler wirkten sich ein längerer Urlaub oder Krankheit bis in den Support negativ aus.
- Die Anforderungen wurden erst im Laufe des Entwicklungsprozesses angepasst und präzisiert.
- Eine klare Aussage über den Fertigstellungsgrad des Release war nicht möglich, da viele Themen nur zu 90 % erledigt waren. Bekanntlich benötigen die letzten 10 % in etwa die Hälfte der Zeit.
- Erst zu einem sehr späten Zeitpunkt wurde klar, welche Features tatsächlich fertiggestellt werden können.
- Engpässe und Probleme im Entwicklungsprozess früh erkennen.
- Änderungen aufgrund von Kundenwünschen ermöglichen.
- Bessere Planbarkeit in der laufenden Entwicklung.
- Den Informationsfluss innerhalb des Entwicklungsteams verbessern.
- Die Entwickler stärker in den Prozess – auch in Richtung Kunde – einbinden.
- Know-how in der Entwicklung streuen.
- Für das Management des Unternehmens eine höhere Transparenz des Entwicklungsprozesses herstellen.
- Ständige Verbesserung des Prozesses.
- Den Produktwert durch eine höhere Qualität steigern.

Unter diesen Problemen litt die Qualität des Produkts.

Wünsch Dir was

Im Herbst 2009 wuchs die Bereitschaft, sich mittels einer neuen Entwicklungsmethode neu aufzustellen. Wir begannen mit der Evaluierung von Alternativen und entschieden uns letztlich auf Grund der Empfehlung einer externen Unternehmensberatung für Scrum. Wesentliche Ziele der Einführung waren:

Um diese Ziele zu erreichen, war es zunächst wichtig, diese nicht nur klar zu formulieren und zu benennen, sondern auch die Unterstützung des gesamten Unternehmens – vor allem der Unternehmensleitung – einzuholen und einzufordern. Scrum sollte bewusst nicht zum Selbstzweck oder als interne Spielerei eingeführt werden, sondern sich konkret an der Leistung für das Unternehmen messen lassen. In einer Eingangsschulung durch externe Experten der Firma andrena objects (Karlsruhe) wurden daher ganz bewusst nicht nur die Entwickler und Produktmanager geschult, sondern das gesamte Managementteam mit Begrifflichkeiten, Methoden und vor allem den Zielen von Scrum vertraut gemacht.

Die wesentlichen *Herausforderungen an die Herstellung unserer komplexen Produkte*



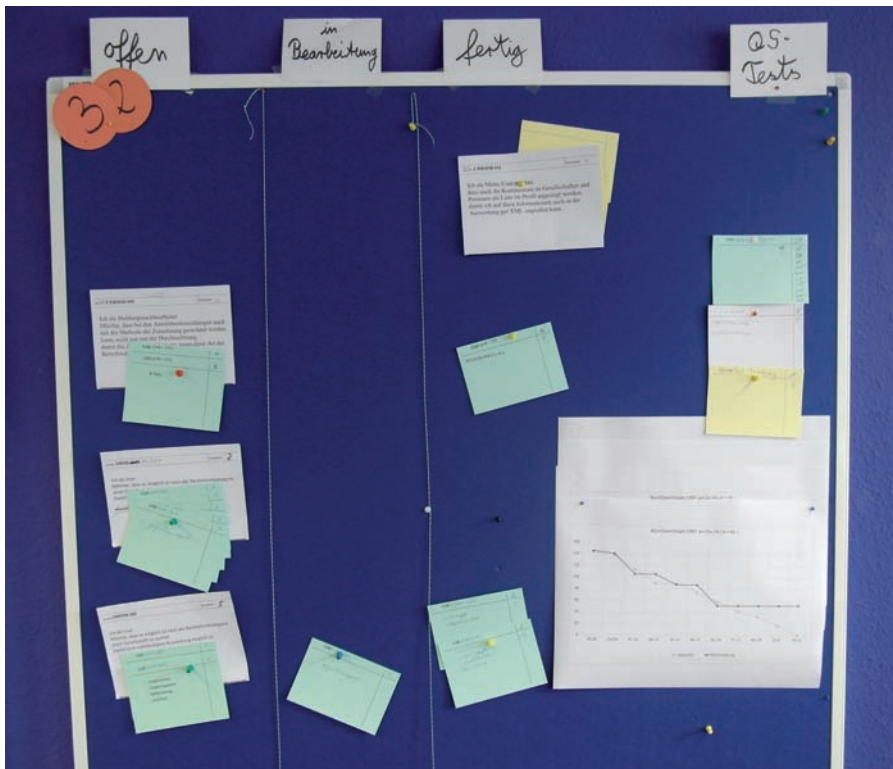


Abb. 2: Bearbeitungsphasen der User-Stories in den einzelnen Teams.

gleichen unserer Erfahrung nach denen vieler mittelständischer Unternehmen:

- Es herrscht ein hoher Erwartungsdruck hinsichtlich Funktionsumfang und zeitlicher Verfügbarkeit des nächsten Releases seitens der Kunden und des eigenen Managements.
- Die Entwicklung ist know-how-intensiv – mit Schlüssellressourcen sowohl auf Seiten des Produktmanagements als auch innerhalb des Entwicklungsteams.

Als Besonderheit kam bei uns hinzu, dass – bedingt durch die Akquisitions- und nachfolgende Fokussierungsstrategie – die Entwicklungsteams in den Jahren 2007 bis 2009 nicht nur neu zusammengesetzt werden mussten, sondern dass auch ein Umzug von zwei Standorten (Frankfurt/M., Heidelberg) an einen neuen gemeinsamen Standort in Heidelberg anstand. Eine weitere Besonderheit stellt sicherlich die Entwicklung des SAP-basierten Systems mit der Programmiersprache ABAP innerhalb der SAP Entwicklungsumgebung dar. Im Gegensatz zu anderen Entwicklungsumgebungen, wie beispielsweise Eclipse, steht dafür nur eine sehr begrenzte Anzahl von Werkzeugen zur Unterstützung des Entwicklungsprozesses mit Scrum zur Verfügung. Bei uns wurden die folgenden Werkzeuge eingesetzt:

- Der *CodeInspector* prüft die erstellten Entwicklungsobjekte nach vorgegebenen Metriken und weist auf problematischen Code hin.
- Mit dem *Unit Test Framework* der SAP können automatisierte Testfälle entwickelt werden, die nächtlich geprüft werden.

Der Build einer Anwendung erfolgt nach dem Aktivieren der jeweiligen Entwicklungsobjekte in SAP automatisch, weshalb hierfür kein weiteres Werkzeug notwendig war. Ebenfalls eingebaut ist auch eine Quellcode-Verwaltung, die eine einfache Versionierung der Entwicklungsobjekte erlaubt.

Ein Jahr später

Innerhalb kürzester Zeit konnten wir die ersten Effekte wahrnehmen. Andere Auswirkungen wurden erst nach einigen Sprints deutlich. In Bezug auf die Zielsetzung ergaben sich die im Folgenden dargestellten Verbesserungen:

Engpässe und Probleme im Entwicklungsprozess früh erkennen

Wir entschieden uns – auch auf Rat der einführenden Berater – für kurze, also 14-tägige, Sprints. Schon nach wenigen Sprints waren wir im Umgang mit dem Task-Board

(siehe Abbildung 2) und dem Sprint-Backlog so geübt, dass sich Störungen recht deutlich im Sprint-Burndown bemerkbar machten und wir gemeinsam den Ursachen früh gegensteuern konnten.

Änderungen aufgrund von Kundenwünschen ermöglichen

Jeden Änderungswunsch von Kundenseite münzte der PO in eine User-Story um. Diese Story nahm den gewohnten Scrum-Verlauf (schätzen, priorisieren, gegebenenfalls zur Planung mitbringen) – unabhängig davon, ob sie für das aktuelle Release oder eine kundenspezifische Anforderung für eine Produktversion in Wartung war. Wir waren erstaunt: Nicht nur etablierte sich ein strukturell besserer Prozess in der kundenspezifischen Entwicklung, sondern auch die Entwickler fühlten den Schutz vor dauerndem unkontrolliertem Eingriff in ihre Arbeitsabläufe sehr deutlich.

Bessere Planbarkeit in der laufenden Entwicklung

Aller Anfang ist schwer: Den Effekt der besseren Planbarkeit spürt man nur, wenn das Product-Backlog gefüllt und gepflegt ist. Angesichts der hohen Komplexität unseres Produkts wurde die Pflege des Product-Backlogs (inklusive Schätzungen) zum Wettlauf gegen die Zeit. Erst spät im Verlauf des Releases konnten wir Aussagen in Form eines Release-Burndowns treffen. Trotzdem landeten wir zum Ende des letzten Major-Release (3.0, April 2011) noch nie so gut. Hier spielte Scrum eine große Stärke aus: Wir erhielten eine Idee von der Geschwindigkeit der laufenden Entwicklung und somit ein effektives Steuerungsinstrument für den PO. Die vorher übliche letzte Phase vor Entwicklungsschluss mit Burnout-Syndromen in den Entwicklungsteams blieb aus.

Mit großem Optimismus gehen wir in das nächste Release, das sehr vom nun schon gut gepflegten Product-Backlog profitieren wird. Unsere Schätzungen haben sich als realistisch erwiesen und wir haben erlebt, wie stabil die Entwicklungsgeschwindigkeit in den Teams war (siehe Abbildung 3). Daher werden wir in Zukunft schon früh Prognosen über den Verlauf der Entwicklung eines Release treffen und somit auf und mit Änderungen reagieren können.

Den Informationsfluss innerhalb des Entwicklungsteams verbessern

Die nicht unübliche Skepsis gegenüber den Artefakten von Scrum wich nach einigen

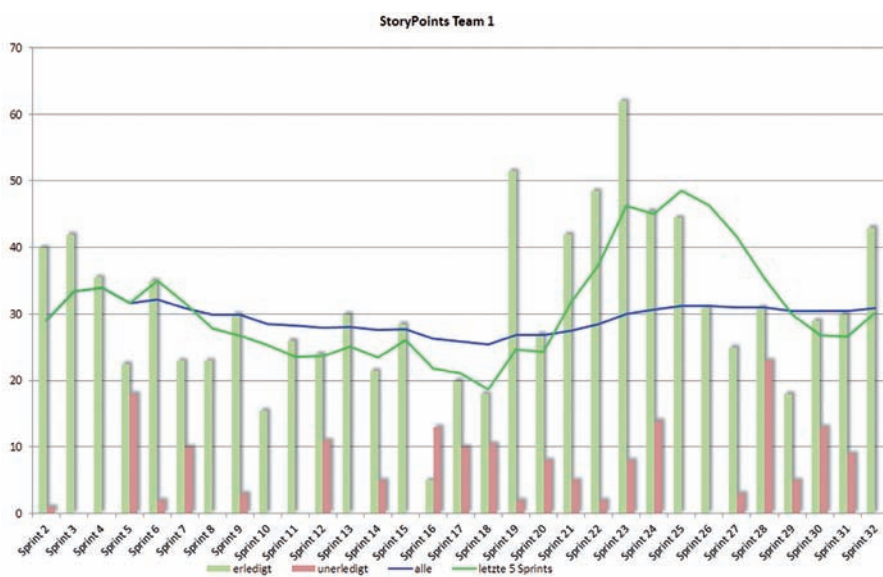


Abb. 3: Anzahl der abgenommenen Story-Points pro Sprint: Langfristig sehr konstant, kurzfristig stark schwankend.

Sprints und die ungewohnte Form des Stand-Ups im Daily Scrum wurde zur Selbstverständlichkeit. Scheuklappen und Tunnelblick gehören der Vergangenheit an und jeder Entwickler ist über die wesentlichen Vorgänge, Fortschritte und Hindernisse tagesaktuell informiert. Das machte sich z. B. in der sinkenden Rate von Missverständnissen bemerkbar. Aus unserer Sicht ist hier auch die soziale Komponente förderlich: In einer traditionell mit wenig zwischenmenschlicher Interaktion verbundenen Tätigkeit ist der häufige Austausch, den Scrum einfordert, hilfreich für das Zusammenwachsen der Teams und verbessert allgemein den Umgang miteinander.

Know-how in der Entwicklung streuen

Einige der Scrum-Mechanismen (Schätzen im Team, nicht nur durch Spezialisten, Planung durch das Team, nicht nur durch Teamleiter und Produktmanager) führten zu einer höheren Bindung der Entwickler an das Produkt und den Entwicklungsprozess. Gerade im Umfeld der SAP-Entwicklung ist eine sehr intensive Spezialisierung vorhanden, was gesteigert wird durch eine spezifische fachliche Kompetenz für einzelne Komponenten. Scrum hat uns hier geholfen, den Tunnelblick zu reduzieren, undurchsichtige, oft beklagte Einzelbaustellen aufzuräumen, durch die Mitwirkung jedes Entwicklers am Gesamtprozess die Identifikation mit

dem Produkt zu erhöhen sowie das Know-how zu streuen. Letzteres ist nicht nur wichtig, um Ausfälle zu kompensieren, sondern entlastet durch Verteilung auch den einzelnen, falls es in bei den Tätigkeiten an bestimmten Komponenten zu Auslastungsspitzen kommt. Die Teams äußern sich darüber hinaus positiv zum entstanden Team-Spirit.

Für das Management eine höhere Transparenz des Entwicklungsprozesses herstellen

Scrum entzieht dem Top-Management die Möglichkeit, durch Mikro-Management in den laufenden Entwicklungsprozess einzugreifen. An diese Stelle tritt der strukturierte Prozess, bei dem Storys eingebracht werden, deren Priorisierung, Abarbeitung und Fortschritt transparent sind. Neben dem gewünschten Effekt des Schutzes des Entwicklungsprozesses ist diese Transparenz für das Management – und generell natürlich für jeden interessierten Mitarbeiter – höchst willkommen. An die Stelle von Auswertungen, die immer rückwärts blicken, und Projektplänen, die schon nach wenigen Wochen nicht mehr akkurat waren und nur mit hohem Aufwand angepasst werden konnten, tritt der Blick auf das Jetzt in den Vordergrund. Im Rahmen von Sprint-Backlogs und Release-Burn-downs kann der Status quo begutachtet und eine Prognose über den Grad der Zielerreichung getroffen werden (siehe

Abbildung 4). So hat unser Management auch Sicherheit in den Aussagen gegenüber Kunden gewonnen. Wir haben jedoch auch festgestellt, dass die Offenheit der Reviews und Planungen nicht wie erwartet angenommen wurde. Der Scrum Master muss an dieser Stelle mehr Werbung machen, um die Anzahl der nicht in der Produktentwicklung Beteiligten (im Scrum-Jargon „Chickens“) in diesen Veranstaltungen zu erhöhen.

Ständige Verbesserung des Prozesses

Während des einen Jahres, in dem wir mit Scrum gearbeitet haben, haben wir unsere Scrum-spezifischen Regeln der Zusammenarbeit mehrfach angepasst. Retrospektiven, die wir nicht als Overhead ansehen, haben uns geholfen, einige Stolpersteine aus dem Weg zu räumen. Wir sehen daher auch weiterhin Potenzial, den Prozess zu verbessern – nicht zuletzt auch in technischer Hinsicht, wo uns das Umfeld nach wie vor keine schnellen Builds erlaubt und dies auch QA-seitig Raum für Optimierungen bietet. Große Themen werden in naher Zukunft testgetriebene Entwicklung (*Test Driven Development, TDD*), der Ausbau der Unit-Tests sowie Pair Programming sein. Scrum hat für uns den Weg in Richtung agiler Techniken geebnet.

Den Produktwert durch eine höhere Qualität steigern

Die Qualität der Neuentwicklungen ist erheblich gestiegen. Im Laufe der Zeit wurden von den Teams immer mehr Qualitätssicherungsmechanismen beschlossen, damit die Entwicklungen nach dem Review wirklich „potenziell auslieferungsfähig“ sind. Im ersten Schritt wurden ein Code-Review durch einen Kollegen und die Erstellung von Unit-Tests für den neuen Code vereinbart (siehe Abbildung 5). Später wurde beschlossen, dass für jede User-Story ein CCRU-Task (*Codereview, CodeInspector, Refactoring und Unit-Test*) geplant werden muss. Mit dem Umhängen dieser Task-Karte in die Spalte „Fertig“ bestätigt der Entwickler, dass diese vier Punkte durchgeführt wurden. Weitere qualitätssichernde Maßnahmen sind:

- Die Durchführung eines Pre-Review, in dem das Team zwei Tage vor dem Sprint-Ende unter realistischen Bedingungen (Testsystem, Test-User usw.) die User-Storys präsentiert.



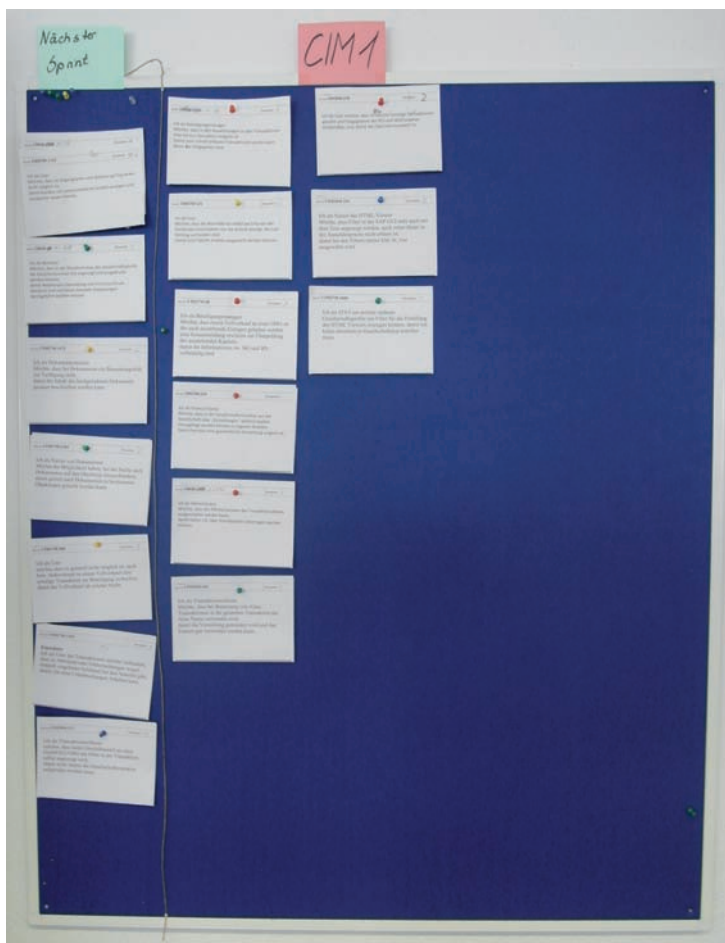


Abb. 4: Backlog für kommende Sprints.

- Die Einführung eines Tasks für Überkreuztests durch einen anderen Entwickler.
- Die klare Definition, wann eine User-Story nicht abgenommen wird.

De facto gab es vor Scrum nur eine informelle Abnahme von Produkt-Features. Die Formalisierung der Abnahme von Storys in Reviews zum Sprint-Ende war hier ein wesentlicher Fortschritt. Das Release ent-

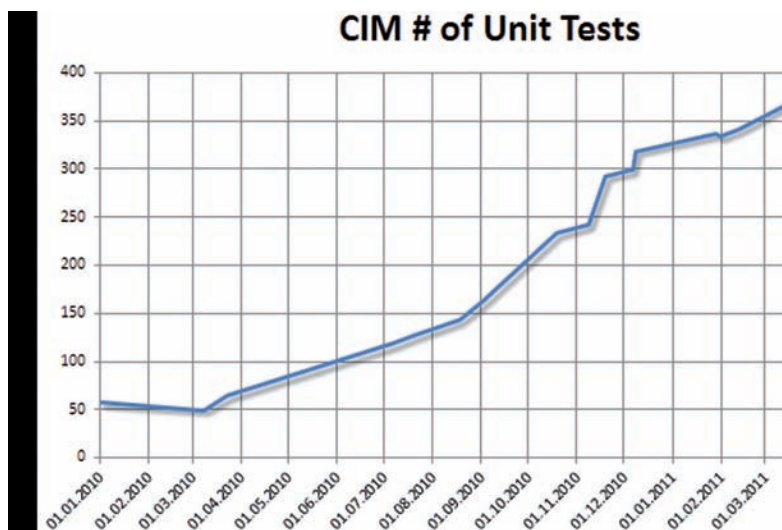


Abb. 5: Zunahme der Unit-Tests im Laufe der Release-Entwicklung.

hält nichts, was der PO nicht gesehen und für gut befunden hätte und unsere *Definition of Done* sichert den gewünschten Qualitätsstandard. Dass das den Wert des Produkts erhöht und den Support entlastet, ist offensichtlich.

Die genannten Erfahrungen sind teilweise spezifisch für das SAP-basierte Produkt. Insgesamt hat sich Scrum aber für alle drei Produkte ähnlich entwickelt. Unterschiede sind beispielsweise Sprint-Längen (zwei vs. drei Wochen), Nomenklaturen (Story-Points vs. Personentage) etc. Dies spiegelt aber mehr die Individualität der Beteiligten wider, als dass hier divergierende Entwicklungen sichtbar wurden.

Nur wenige Synergien gibt es im Bereich des Product-Backlogs. Hier kommen aufgrund der verschiedenen Zielgruppen der Produkte sehr unterschiedliche Storys zustande. Mehr als ein fachlicher Erfahrungsaustausch und wenige wiederverwendbare Komponenten sind nicht darstellbar.

Quo vadis, Scrum?

Die Einführung von Scrum war für uns alle ein voller Erfolg. Unsere Erwartungen und Ziele waren nicht zu hoch gesteckt, und die Befürchtung, Scrum ließe sich in einem „konservativen“ SAP-Umfeld nicht umsetzen, haben wir selbst widerlegt. Technologische Unterschiede des Entwicklungsprozesses hatten einen eher geringen Einfluss auf die Umsetzung des Scrum-Prozesses. Wir sind jedoch nach einem Jahr noch nicht am Ende des Weges angekommen und sehen durchaus weiteres Verbesserungspotenzial. Wir freuen uns, mit den gewonnen Erkenntnissen in das nächste Release zu starten und weitere, in Scrum quasi eingebaute Veränderungen vorzunehmen. Wie schon erwähnt, werden wir uns beispielweise in größerem Maße agile Techniken zueigen machen.

Wir haben auch gelernt, dass Scrum an manchen Stellen grundlegende strukturelle Probleme aufzeigen kann, aber keine Lösungen anbietet. Diese zu erarbeiten, bleibt in der Verantwortung aller Beteiligten.

Wir können jedem Hersteller, der Softwareentwicklung im SAP-Umfeld betreibt, Scrum als Prozess-Rahmenwerk uneingeschränkt empfehlen. Scrum setzt hier keine technologischen Elemente voraus und war für uns unabhängig von den unterschiedlichen Rahmenbedingungen erfolgreich umsetzbar. ■