Case Study: BIT IT Consultancy

Autor und Projektverantwortlicher **Sven Hubert (AIT AG)**

Projektverantwortlicher auf Kundenseite **Ive Verstappen (BIT IT Consultancy)**

Stuttgart, den 28.11.2008

Inhalt

Referenzprojekt

Das Referenzprojekt läuft unter der Leitung eines unserer Kunden, der BIT IT Consultancy Belgien. BIT IT Consultancy übernimmt die Leitung und Entwicklung von Softwareentwicklungsprojekten und ist auf .NET-Technologien spezialisiert. 80% der Kunden von BIT IT Consultancy sind andere IT-Firmen.
BIT IT Consultancy arbeitet bereits seit mehreren Jahren nach Scrum. Seit Anfang diesen Jahres mit Unterstützung durch das Visual Studio Team System – der ALM-Lösung der Firma Microsoft. Die AIT AG unterstützt BIT IT Consultancy bei der Einführung und Anpassung des Visual Studio Team Systems sowie bei der kontinuierlichen Prozessoptimierung. Für die Umsetzung von Scrum wurde eine Prozessvorlage von Conchango gewählt – dem SRUM for Team System[1].
Das Projekt auf welchem die hier beschriebenen Erfahrungen beruhen, ist ein Softwareentwicklungsprojekt für ein großes Unternehmen aus der Maschinenbaubranche. Das Entwicklungsteam besteht aus einem Scrum-Master und 2 Entwicklern in Belgien und 5 Entwicklern in Ägypten. Es handelt sich um ein gemeinsames Scrum-Team welches alle Meetings gemeinsam abhält.

Scrum/CMMI Einführung

Scrum ist ein Framework, welches sich am „Agile Manifesto“ (1) orientiert. Scrum dient zur Umsetzung eines agilen Softwareentwicklungsprozesses. Ein Ziel dieser Umsetzung ist die Verringerung des Verwaltungsaufwandes durch teilweise Selbstorganisation der Entwicklungsteams. Das bedeutet nicht Anarchie, sondern erfordert Disziplin und Verantwortung von allen Projektbeteiligten. Die Prozesstransparenz steht im Vordergrund. Diese ermöglicht unter anderem die Verfolgung der Sollerfüllung und sorgt dafür, dass auftretende Probleme frühzeitig sichtbar werden (vgl. (2) und (3)).
Hinter CMMI versteckt sich wiederum kein Vorgehens- sondern ein Bewertungsmodell für Entwicklungsprozesse. Es gibt also keine konkreten Arbeitsanweisungen oder Empfehlungen zur Durchführung eines Projekts an die Hand, sondern zeigt auf, welche Bereiche zu einem vollständigen Vorgehen mindestens gehören. Ein Entwicklungsprozess gilt als CMMI-konformen wenn er die Maßstäbe des Bewertungsmodells in dem jeweiligen Reifegrad erfüllt (vgl. (4)). Motivation für eine CMMI-Zertifizierung ist die Nachweisbarkeit dieser Prozessreife nach innen als auch nach außen. Prozessqualität wird transparenter bzw. soll messbar werden.

TFS Einsatz

Der alltägliche Einsatz des TFS soll anhand von Beispielen aus dem Referenzprojekt beleuchtet werden. Die dargestellten Beispiele sind dabei aus Gründen der Diskretion verfremdet.
Bevor das Team einen Sprint durchführen kann, sollte es wissen, was es überhaupt tun soll. Dazu werden einzelne Anforderungen aus dem Product-Backlog ausgewählt. Um sicherzugehen, dass diese auch vom Team verstanden und für umsetzbar befunden werden können, müssen diese vom Product-Owner und Scrum-Master verifiziert und gegebenenfalls

[1] http://www.scrumforteamsystem.com

aktualisiert werden. Abbildung 1 zeigt die Integration in das Visual Studio. Im rechten Teil der Abbildung befindet sich der Team Explorer, der Abfragen bereitstellt, mit denen beispielsweise das Product-Backlog geöffnet und nach verschiedensten Kriterien gefiltert werden kann. Das Product-Backlog präsentiert sich als Liste im linken oberen Teil. Ein selektiertes Product-Backlog Item ist in der Detailansicht im unteren Teil des Fensters zu sehen. Neben Titel und Beschreibung werden auch Informationen wie Priorität, geschätzter Aufwand und eine Teamzuordnung gespeichert. Alle Einträge besitzen zudem einen Status. Einträge im Product-Backlog können so je nach Fertigstellungsgrad den Status „Not Done“, „In Progress“ und „Done“ besitzen.

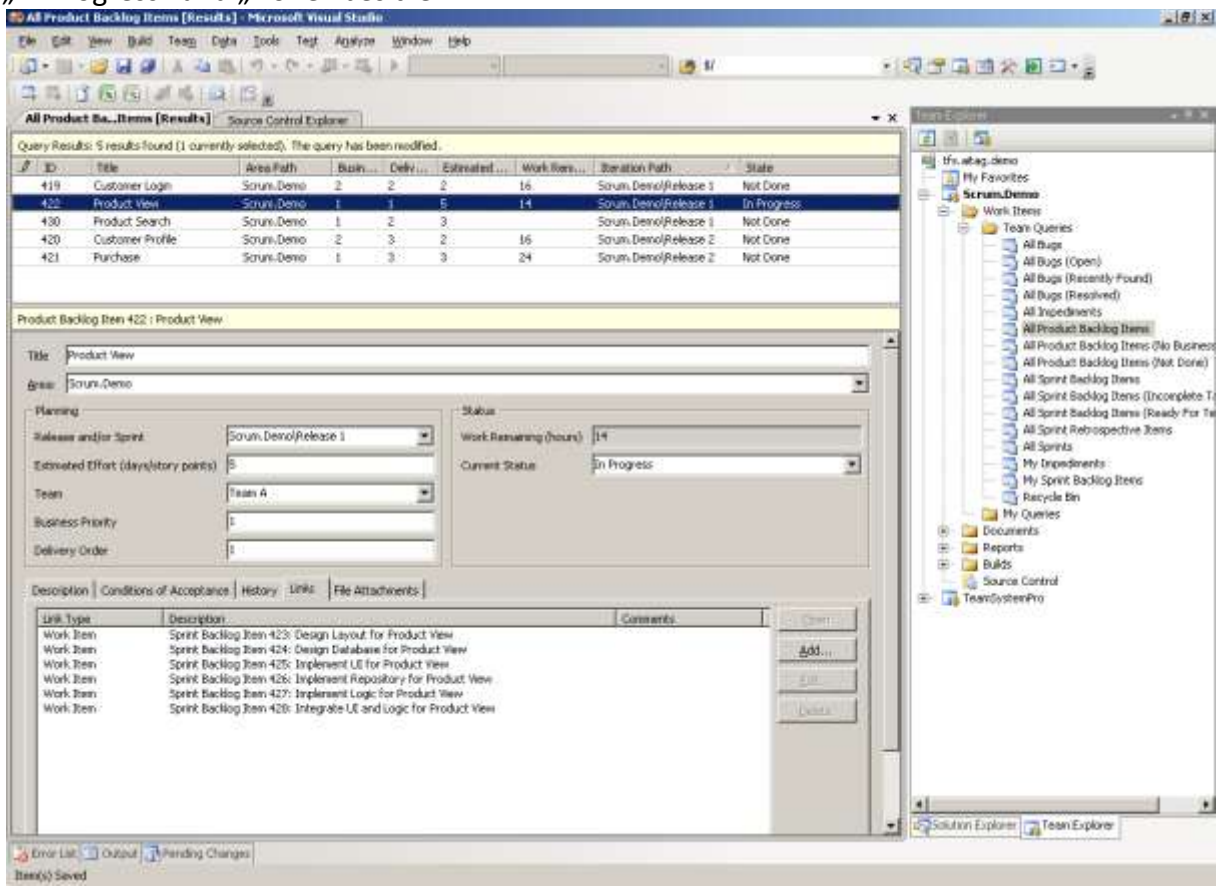

Abbildung 1 - Das Product-Backlog im Visual Studio

Die Liste der Product-Backlog Items wird nach Release in dem sie fertiggestellt werden sollen und nach Priorität sortiert. Sie wird demnach von oben nach unten abgearbeitet.

Neben der Integration in das Visual Studio, gibt es noch weitere Möglichkeiten, die Liste zu bearbeiten. Dies kann mit Microsoft Excel und Microsoft Project erfolgen. Zudem enthält das Visual Studio Team System ein Web-Frontend, das in Abbildung 2 gezeigt wird. Darin ist das nächste Level in der Hierarchie bereits dargestellt – das Sprint-Backlog (rote Markierung). Nach der Planung eines Sprints existieren zu den ausgewählten Product-Backlog Items verknüpfte Sprint-Backlog Items. In Abbildung 2 sind 6 Einträge für das Sprint 1 dargestellt. Durch diese Darstellung kann man einen schnellen Überblick über die Sprints erhalten.

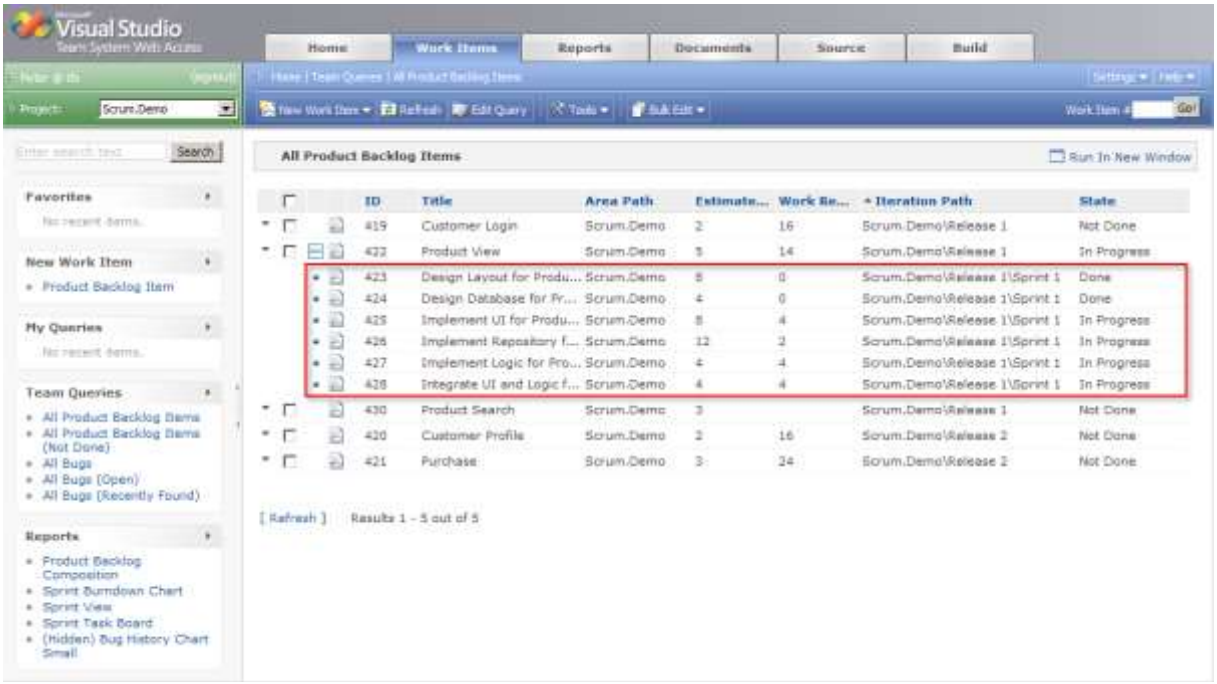

Abbildung 2 - Product-Backlog und das verknüpfte Sprint-Backlog im Web-Frontend

Die Integration im Visual Studio Team System gestaltet sich in der Praxis als sehr hilfreich. Die Daten befinden sich immer an einer zentralen Stelle in einem konsistenten Zustand. Durch die Versionierung und Erfassung von Veränderungen an allen projektrelevanten Daten werden die Anforderungen des CMMI für das Konfigurationsmanagement erfüllt. Diese verlangen, dass alle sogenannten „Work Products“ (engl. Arbeitsprodukte) – also Pläne, Entwürfe, Testergebnisse, Quellcode etc. – eines Projekts unter Konfigurationsmanagement stehen (vgl. (4)). Im Beispielprojekt wurden die Vorteile des neuen Werkzeugs gegenüber den früher verwendeten einfachen Listen deutlich. Zum einen verringerte sich der Pflegeaufwand, da nun konkurrierende Änderungen an Aufgaben keine Inkonsistenzen hervorrufen, wie das in den alten Listen häufig der Fall war. Der zweite Punkt der Verbesserung betrifft die Traceability – ebenfalls Thema im CMMI. Anforderungen lassen sich so über Aufgabendefinitionen bis hin zum Quellcode zurückverfolgen. Dies wird durch den Checkin-Prozess im Visual Studio Team System möglich und funktioniert genauso auch in die andere Richtung. Das wurde von den Entwicklern als größter Vorteil empfunden, da sie ohne größeren Aufwand frühere Designentscheidungen anhand dieser Verknüpfungen schneller nachvollziehen können.

Das Aufbrechen von Product-Backlog-Items in „kleinere“ aber konkretere Sprint-Backlog-Items stellt nur zwei Ebenen in einer Hierarchie dar, die der Komplexitätsbewältigung im Projekt dient. Weitere Stufen sind in beide Richtungen der Hierarchie anwendbar. So lässt sich die Hierarchie beispielsweise durch übergeordnete Abstraktionsebenen erweitern. So ist es zum Beispiel eine gängige Praxis, ein Top-Level Backlog auf Unternehmens- oder Produktfamilienebene zu pflegen. Dieses untergliedert sich in einzelne Produkte und evtl. weiter in einzelne Funktionen bis es ausreichend feingranular ist, um die Umsetzung in Form von Sprints zu planen. In (3) wird dieser Ansatz näher erläutert. Eine weitere Verfeinerung der Sprint-Backlog-Items ist ebenfalls nötig.

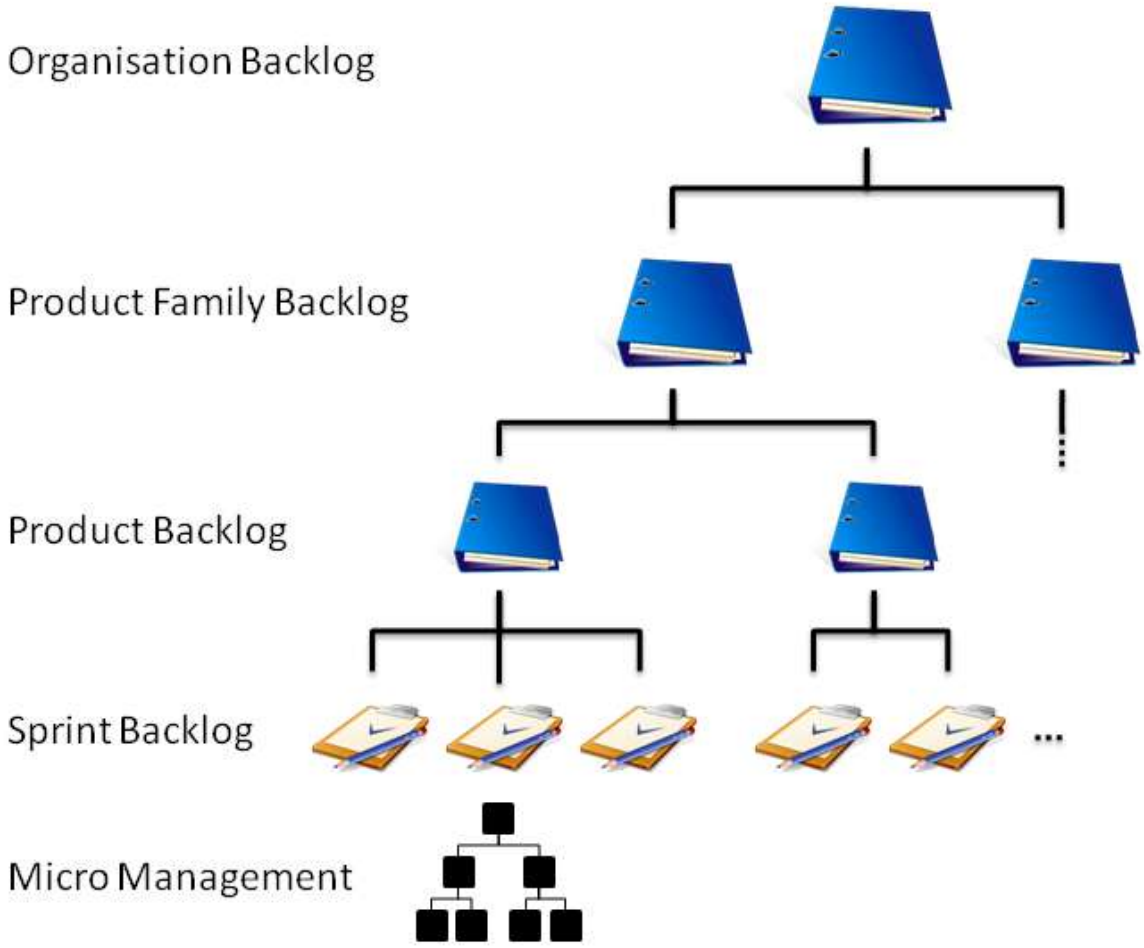

Abbildung 3 - Möglichkeiten zur Hierarchiebildung

Bei BIT IT wurde davon allerdings abgesehen. Man lief Gefahr, dem Mikromanagement von Aufgaben zu verfallen, was im Gegensatz zur gewünschten Selbstorganisation des Teams stünde. Stattdessen wurden Sprint-Backlog-Items von begrenztem Umfang angelegt. In der Planung stellen definierte Regeln sicher, dass Aufgaben mit einem Umsetzungsaufwand zwischen 1 Stunde und 2 Tagen definiert werden.
Diese Regel unterstützt zudem die Vorhersagbarkeit des Projektverlaufs, da kontinuierlich Fortschritte messbar sind bzw. blockierte und langwierige Aufgaben rechtzeitiger identifiziert werden. Scrum bietet zum Zwecke der Fortschrittsverfolgung das sogenannte „Burn-Down-Chart“. Es zeigt die geleisteten Aufwände und den Trend der Abarbeitung der geplanten offenen Aufgaben. In Abbildung 4 ist ein solcher Bericht für einen Beispiel-Sprint dargestellt. Es zeigt den Trend, wie er sich zum Zeitpunkt der Planung ergab und den aktuellen Trend anhand der bisher tatsächlich geleisteten Arbeit.
Diese Darstellung ist nicht auf einen Sprint beschränkt, sondern kann auch auf Product-Backlog-Ebene erfolgen und zum Beispiel in Releases aufgegliedert sein. So kann im Sinne der Hierarchie sowohl der Fertigstellungsgrad eines einzelnen Produkts als auch einer gesamten Produktfamilie beobachtet werden.

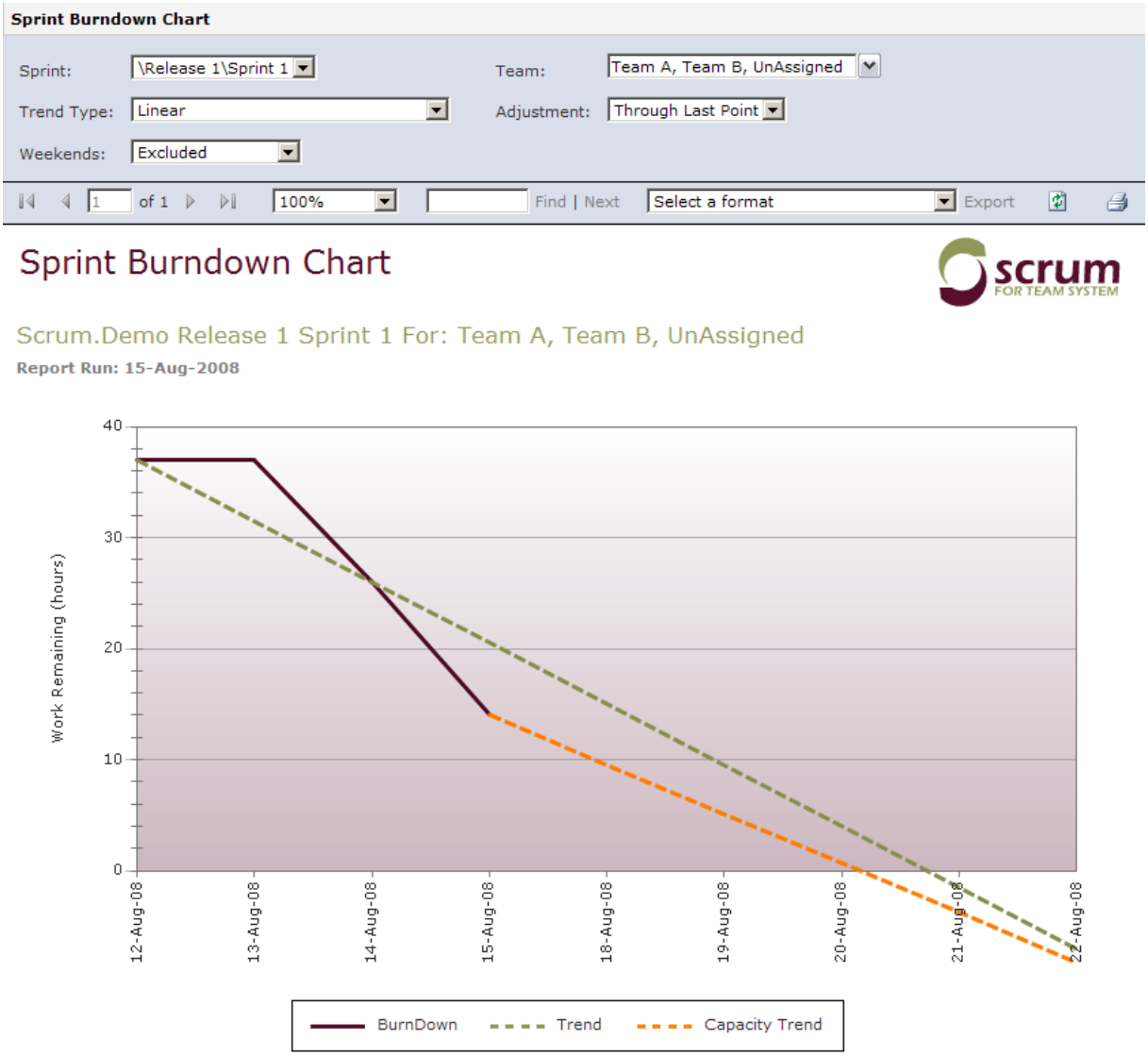

Abbildung 4 - Ein Sprint-Burn-Down-Chart

Erkenntnisse/Erfahrungen aus Projekt

Änderungen und Ergänzungen zu Scrum

Um mit Scrum CMMI-Kompatibilität zu erreichen, musste das Vorgehensmodell und deren Prozeduren angepasst bzw. erweitert werden.

So war die Dokumentation von Produkt und Prozessen nicht ausreichend. Entsprechend wurde diese Schwachstelle durch eine umfangreichere Prozessdokumentation, sowie durch formalisertere Produktdokumentation ausgeglichen. Die eingesetzte Prozessvorlage von Conchango bietet leider im Vergleich zu den Standardvorlagen von Microsoft nur wenig Prozessdokumentation. Hier muss man selbst nachbessern. Es hat sich zudem als Vorteil herausgestellt, die Prozesse in Form kurzer Videos zu dokumentieren. Die Motivation im Team sich diese „Dokumente näher anzusehen“ ist deutlich größer als bei seitenlangen Schritt-für-Schritt-Anleitungen, die parallel aber dennoch erstellt wurden.

Einige Meetings, die Scrum vorschreibt, haben kein formales Ergebnis und keine Dokumentation. Dies wurde bei der Prozessdokumentation berücksichtigt. Am Ende eines

jeden Meetings steht ein Ergebnis. Im Falle der täglichen Meetings z.B. steht ein Video bereit, um im Nachhinein wichtige Punkte zu verarbeiten. Das hat sich als großer Vorteil erwiesen, da in den Meetings mehrere kleinere Punkte, wie z.B. identifizierte Risiken, ins Gespräch kamen. Durch die Kürze des Meetings können diese nicht sofort protokolliert werden. Es sind zu viele Personen beteiligt. Dies wird im Anschluss an das Meeting nachgeholt.

Das Thema Risikomanagement wird in Scrum durch den iterativen Ansatz adressiert. Durch die kurzen Sprints und das ständige Feedback innerhalb des Teams durch die regelmäßigen Meetings, kommen Risiken schnell zur Sprache. Dennoch fehlt ein klar definierter Prozess, wie mit den vorhandenen Risiken verfahren werden soll. Hier wurde im Referenzprojekt das Riskikomanagement nach dem Microsoft Solutions Framework übernommen (vgl. (5)).

Der stark iterative Ansatz von Scrum lenkt den Fokus auf die Betrachtung einzelner Sprints. Es ist für den Überblick aber eine globalere Sicht notwendig. Das reicht von der Projektebene bis zur Unternehmensebene, also der Sicht auf eine Vielzahl von Projekten. In einem vorigen Abschnitt wurde die Einteiltung in eine Hierarchie von Product-Backlogs bereits dargestellt. Diese soll in Zukunft weiter verfolgt und erweitert werden. Derzeit fehlen noch entsprechende Reports, um die entstehenden Daten zu visualisieren.

Während im CMMI Produktqualität indirekt durch die Einhaltung formaler Prozesse erreicht werden soll, steht in den agilen Vorgehensmodellen die direkte Qualität im Vordergrund. Die Unit-Testing-Welle kam mit den agilen Vorgehensmodellen in Schwung. Bei Scrum ist es die Aufgabe des Scrum-Masters entsprechende Vorkehrungen zur Qualitätssicherung zu treffen und diese evtl. mit Unterstützung umzusetzen. Dies reicht von der Prozessdokumentation über Mitarbeiterschulungen bis zu Regressionstests während eines Sprints. Das macht den Erfolg der Maßnahmen und den Gesamterfolg des Projektes sehr stark abhängig von der Kompetenz des Scrum-Masters. Hier wird derzeit an einer besseren Rollenverteilung gearbeitet, so dass der Scrum-Master entlastet wird bzw. einzelne Themen auch an Spezialisten abgibt.

Neben dem Prozess nach Scrum muss es auch eine wohldefinierte Architektur sowie dokumentierte Entwicklungsrichtlinien geben. Anderenfalls würde die Selbstorganisation zwar auf Prozessseite funktionieren, allerdings auf der Produktseite chaotische Zustände hinterlassen. Man darf also trotz CMMI-Fokus nicht den Blick für das Wesentliche, nämlich das Produkt verlieren. Ein Ungleichgewicht zu Gunsten CMMIs würde zu viel Dokumentation zum Prozess ergeben und die Produktdokumentation unter Umständen außen vor lassen.

Dies reduziert sich den Erfahrungen zu Folge aber nicht nur auf die Softwarearchitektur.

Da im Referenzprojekt verschiedene Firmen beteiligt sind, wurde zu Beginn eine Projektcharta erstellt, die die verschiedenen Rollen und Verantwortlichkeiten klar regelt. So ist dort z.B. erfasst, dass BIT IT den kompletten Softwareentwurf übernimmt, Teile des Anforderungsmanagement aber an den Kunden abgibt.

Da sich die Projektcharta als nicht ausreichend erwiesen hat, wird zusätzlich an der Spezifikation einer Unternehmensarchitektur gearbeitet. Aus den bisherigen Erfahrungen zur Zusammenarbeit von 4 verschiedenen Unternehmen bzw. Abteilungen heraus, ergab sich aber der Bedarf einer sogenannten „Enterprise Architecture“. Diese beinhaltet definierte Verantwortlichkeiten über Infrastrukturfragen, die für die Projektabwicklung relevant sind. Das sind zum Beispiel das Projektcontrolling, die Netzwerkinfrastruktur oder die Ressourcenplanung. Um für CMMI konform zu bleiben, wurde das formale Zachmann-

Framework (vgl. (6)) als Basis für die Definition der Unternehmensarchitektur ausgewählt, da es wesentliche Punkte zur Organisation der Projektabwicklung zwischen Organisationen, wie z.B. Verantwortlichkeiten für Definition, Umsetzung und Tests adressiert.

Nicht formalisierbare Prozesse

Softwareentwicklung ohne Menschen wird wohl auch in Zukunft nicht machbar sein. Es handelt sich dabei um einen Prozess, der sowohl Disziplin als auch Kreativität abverlangt. Die agilen Methoden scheinen es mit der Disziplin nicht so genau zu nehmen. Doch auch dieser Schein trügt. Anstatt Organisation und Planung von außen, stellen die agilen Methoden lediglich die Selbstorganisation des Teams in den Vordergrund. Das bedeutet konkret, dass das Team für die Organisation und Berichterstattung während der Entwicklungsphase mitverantwortlich ist. Während bei Scrum die Priorisierung der Anforderungen in Form des Product-Backlogs vom Product-Owner vorgenommen wird, erfolgt die Planung der Aufgaben in Form des Sprint-Backlogs durch das Team. Selbstverständlich müssen die Bedürfnisse des Kunden, wie etwa Fertigstellungstermine einfließen. Dafür sorgen regelmäßige Meetings mit dem Team und dem Product-Owner, besonders während der Planung eines neuen Sprints.
Der Schritt in Richtung Selbstorganisation ist steinig. Es Bedarf in vielen Fällen einer Umstellung der gesamten Unternehmenskultur. Weg von gebrandmarkten Entwicklern, die gelernt haben, möglichst spät Probleme nach oben zu melden, so dass vielleicht doch ein anderer zuerst die schlechte Nachricht verbreitet. Weg von Projektmanagern, die auch dann noch auf frühere Commitments beharren, wenn sich die äußeren Umstände bereits mehrfach geändert haben. Hin zu höherer Verantwortungsbereitschaft und offenerem Umgang mit Konflikten und Problemen.
Ein Mehr an Verantwortung für den Einzelnen führt nur zum Ziel, wenn die Voraussetzungen auf beiden Seiten, also dem Management und dem Team stimmen. Die Entwickler müssen sich von einseitiger Technologieaffinität lösen und mehr im Sinne des späteren Anwenders ihrer Produkte denken. Das Projektmanagement wird sich in den täglichen Scrum-Meetings wohl auch auf technische Aspekte einlassen müssen.
Stimmen diese menschlichen Voraussetzungen in der Organisation nicht, wird das kein Tool und kein Prozess ausgleichen können. Ein Tool wie zum Beispiel das Visual Studio Team System kann lediglich dabei helfen, die Schwachstellen eindeutiger und früher sichtbar zu machen. Ein Prozessrahmenwerk wie Scrum kann dafür sorgen, die Anstrengungen besser zu kanalisieren.

Kosten und Nutzen des Visual Studio Team System

Mit dem Team Foundation Server sind alle Voraussetzungen für eine CMMI-Zertifizierung nach Level 2 und ein Großteil für Level 3 erfüllt. Für den Erfolg eines Projektes bzw. Teams ist das aber noch nicht hinreichend. Begründet sich dieser doch weiterhin auf Können und strukturiertem Vorgehen. Mit einer Zertifizierung nach CMMI lassen sich Lücken im Vorgehen identifizieren und schließen.
Mit dem Einsatz des Visual Studio Team System können viele Anforderungen mit geringen Startkosten im Softwareentwicklungsprozess implementiert werden. Im Referenzprojekt sind initiale Kosten von ca. 10000 € entstanden, für Lizenzierung und Einführung des TFS.

Diese haben sich durch die Ablösung umständlicher, manueller Prozeduren nach ungefähr einem halben Jahr bezahlt gemacht.

Referenzen

1. The Agile Manifesto. [Online] 2001. http://www.agilemanifesto.org.
2. **Schwaber, Ken.** *Agile Project Management with Scrum.* s.l. : Microsoft Press, 2004. ISBN:9780735619937.
3. —. *The Enterprise and SCRUM.* s.l. : Microsoft Press, 2007. ISBN:9780735623378.
4. **SEI Carnegie Mellon University.** *CMMI® for Development Version 1.2.* Pittsburgh : SEI Carnegie Mellon University, August 2006.
5. **Turner, Michael S. V.** *Microsoft Solutions Framework Essentials: Building Successful Technology Solutions.* s.l. : Microsoft Press, 2006. ISBN: 9780735623538.
6. **John A. Zachman, Zachman International.** The Zachman Framework: A Concise Definition. [Online] Zachman International. http://www.zachmaninternational.com/index.php/the-zachman-framework.
7. Microsoft Solutions Framework. *Microsoft Technet.* [Online] Microsoft Corp. http://www.microsoft.com/technet/solutionaccelerators/msf/default.mspx.