



□ Stefan Mieth

(Stefan.Mieth@aitgmbh.de)

arbeitet als Senior Software Consultant und Projektleiter bei AIT und ist Microsoft Certified Solution Developer für Visual Studio ALM. Er hat sich auf die Bereiche Projektmanagement, Entwicklungsprozesse und Anforderungsmanagement spezialisiert. Stefan Mieth unterstützt Unternehmen bei der effizienteren Softwareentwicklung und berät bei der Einführung und Anpassung des Visual Studio Team Foundation Server (TFS). Seine Erfahrungen vermittelt er als Autor des TFS-Blogs (<http://www.tfsblog.de>), in Magazinen und in Vorträgen.



□ Thomas Rümmler

(Thomas.Ruemmler@aitgmbh.de)

arbeitet als Senior Software Consultant und Projektleiter bei AIT und ist Microsoft Most Valuable Professional (MVP) für Visual Studio ALM. Sein Arbeitsschwerpunkt liegt auf Application Lifecycle Management (ALM). Thomas Rümmler hilft Unternehmen, ihre Software unter Einsatz des Microsoft Visual Studio Team Foundation Server (TFS) effizienter zu entwickeln. Seine Erfahrung gibt er u. a. als Autor des TFS-Blogs (<http://www.tfsblog.de>) und Sprecher im Microsoft-ALM-Umfeld weiter.

# Agilität ohne Product Backlog Items oder User Stories? – Agile Entwicklung mit dem TFS und dem CMMI-Template

Man kann den Eindruck bekommen, dass die Welt der Softwareentwicklung ausschließlich agil arbeitet – oder zumindest gerne so arbeiten würde. Alle sprechen von iterativer Entwicklung, Sprint-Planung, selbstbestimmten Teams und Product Backlogs. Die klassischen Produktverantwortlichen verschwinden mehr und mehr und werden durch Product Owner abgelöst. Scrum Master kümmern sich unter anderem um die Einhaltung der Spielregeln sowie darum, dass eventuell vorhandene Stolpersteine auf dem Weg des Teams zum Erfolg bzw. Sprint-Ende aus dem Weg geräumt werden.

## Problemstellung

In einigen Industriezweigen ist dies auch ohne Weiteres genauso möglich. Doch immer wieder gibt es spannende Projekte, in denen die Betrachtung von *Remaining Work* nicht ausreichend ist. Auch kann es erforderlich sein, Risiken explizit zu betrachten und zu bewerten. Kurzum, es gibt in manchen Industriezweigen die Anforderung, innerhalb des Entwicklungsprozesses mehr Informationen zu verwalten, als es die populären agilen Prozesse wie Scrum [SchSu11] vorsehen.

## Die Wegrichtung – eine fundamentale Entscheidung

In vielen Softwareentwicklungsteams wird irgendwann die Frage der richtigen Vorgehensweise gestellt. Dies kann ganz unterschiedlich ablaufen, als Diskussion in kleiner Runde, per Order di Mufti oder auch aus einem Beratungsprojekt heraus nach

eingehender Analyse der Bedürfnisse. Hierbei haben sich aus unserer Projekterfahrung heraus zwei grundsätzliche Herangehensweisen herauskristallisiert. Die Prozessdiskussion beginnt demnach entweder komplett ohne vorgegebenen Rahmen oder entlang eines Leitfadens.

Beide Varianten haben gewisse Vor- und Nachteile. Während man bei der ersten Variante mit größerer gedanklicher Freiheit ans Werk geht und gemäß gutem Anforderungsmanagement zunächst vollständig lösungsfrei diskutiert, besteht auch die Gefahr der Aufwandsunterschätzung.

	Visual Studio Scrum	MSF for Agile Software Development	MSF for CMMI Process Improvement
Portfolio Backlog			
Product Backlog			
Work Item Types	<ul style="list-style-type: none"> <li>• Feature</li> <li>• PBI</li> <li>• Bug</li> </ul>	<ul style="list-style-type: none"> <li>• Feature</li> <li>• User Story</li> </ul>	<ul style="list-style-type: none"> <li>• Feature</li> <li>• Requirement</li> <li>• Change Request</li> </ul>
Task / Iteration Planning	<ul style="list-style-type: none"> <li>• Task</li> </ul>	<ul style="list-style-type: none"> <li>• Task</li> </ul>	<ul style="list-style-type: none"> <li>• Task</li> </ul>
Bug Handling	<ul style="list-style-type: none"> <li>• Impediment</li> </ul>	<ul style="list-style-type: none"> <li>• Bug</li> <li>• Issue</li> </ul>	<ul style="list-style-type: none"> <li>• Bug</li> <li>• Issue</li> <li>• Risk</li> <li>• Review</li> </ul>
Non-Requirements			
Test Management	<ul style="list-style-type: none"> <li>• Test Case</li> <li>• Shared Steps</li> </ul>	<ul style="list-style-type: none"> <li>• Test Case</li> <li>• Shared Steps</li> </ul>	<ul style="list-style-type: none"> <li>• Test Case</li> <li>• Shared Steps</li> </ul>

Abb. 1: Work Item Typen im Vergleich, eigene Darstellung basierend auf [Microsoft14]

Häufig ist in unseren Kundenprojekten Microsofts ALM-Plattform Team Foundation Server (TFS) bereits im Einsatz oder soll aus unterschiedlichen Gründen eingeführt werden. Vor diesem Hintergrund kommt die zweite Variante in Betracht, nämlich sich an einem der vom TFS out-of-the-box bereitgestellten Prozessmodelle zu orientieren und somit einen roten Faden für die Prozessdiskussion zu haben. Dieses Vorgehen schränkt den Gestaltungsspielraum etwas ein, sorgt jedoch für schnelle Erfolge.

Ein solches Prozessmodell, in der TFS-Terminologie auch Process Template bzw. Prozessvorlage genannt, ist der Bauplan eines Projektes im Team Foundation Server. Um nun eine Entscheidung zu treffen, ist es zunächst wichtig, einen guten Überblick über die Prozessvorlagen zu haben, die out-of-the-box mit dem Team Foundation Server geliefert werden. Ein wesentlicher Bestandteil sind die *Work Item Types*, welche eine Art Datensatzbeschreibung für Geschäftsobjekte in der zentralen Datenbank sind. Diese können in verschiedenen Ausprägungen auftreten und bestimmen signifikant die Abläufe im Projekt.

**Abbildung 1** zeigt die vom TFS bereitgestellten Work Item Types, welche nachfolgend noch näher erläutert werden. Es ist deutlich zu sehen, dass sich die drei Prozessvorlagen sowohl namentlich als auch im Umfang der Work Item Types unterscheiden.

Folgende wesentliche Abgrenzung ist dabei zu beachten: Es gibt Work Item Types mit ähnlicher Semantik, die sich jedoch in Namen, Attributen, Layout sowie Zustandsmaschine unterscheiden. Ein Vertreter hierfür sind die Elemente Product Backlog Item (PBI), User Story und Requirement. Gemeinsam ist den drei Typen, dass sie Hauptbestandteil des Product Backlogs sind und die Basis für die Iterationsplanung in der Entwicklung darstellen.

Darüber hinaus gibt es den Fall, dass Work Item Types in einer der Prozessvorlagen vorhanden sind, welche in einem anderen Template gar nicht vorgesehen sind. Ein Vertreter dieser Kategorie ist der Work Item Type *Risk* der CMMI-Prozessvorlage, wobei CMMI für *Capability Maturity Model Integration* steht. Dieser Typ dient der expliziten Abbildung eines Risikos im Projekt.

Daraus leitet sich ein weiteres Unterscheidungsmerkmal der verschiedenen Templates ab. Es gibt Work Item Types, die namentlich in mehreren Prozessvorlagen vorkommen. Beispielhaft hierfür wird nun der *Task* betrachtet. Dieser Work Item Type ist in allen drei Prozessvorlagen vertreten. Es gibt Attribute, die solch ein Task in allen Templates bereitstellt. Hierzu zählen unter anderem *Title*, *Description*, *Assigned To*, *State* und *Reason*. Jedoch gibt es auch wesentliche inhaltliche Unterschiede. So gibt es beispielsweise im Task der CMMI-Prozessvorlage die Möglichkeit, diesen als *Task Type Mitigation Action* zu klassifizieren und somit zu verdeutlichen, dass er der Risikominimierung dient. Dies ist in diesem Falle auch sinnvoll, da die CMMI-Prozessvorlage auch das zugehörige Risiko Work Item zur Verfügung stellt.

Neben den inhaltlichen Unterschieden in Form der Attribute der einzelnen Typen ist auch die Benutzerführung unterschiedlich. Dies zeigt sich zum einen in der Gestaltung der Bedieneroberfläche. Während das Scrum Template ein Layout mit zwei gleichgroßen Spalten definiert, präsentieren sich die Agile und die CMMI-Prozessvorlage mit einem größeren zentralen Bereich und zusätzlichen Feldern in einer kleineren Spalte rechts daneben.

Zum anderen unterscheiden sich die Workflows zwischen den verschiedenen Prozessvorlagen sehr stark.

**Tabelle 1** zeigt die einzelnen Zustände aus den Zustandsmaschinen der verschie-

denen Prozessvorlagen. *New* und *Proposed* sind dabei jeweils die Initialzustände. Auffallend ist, dass es im Scrum Template mit *Done* und *Removed* sowie im Agile Template mit *Closed* und *Removed* jeweils zwei finale Zustände gibt. Dies ist in der CMMI-Prozessvorlage anders gelöst. Dort ist *Closed* stets der finale Zustand, unabhängig davon, ob etwas wieder aus der Planung genommen wurde oder nicht. Diese Information erhält man bei der CMMI-Prozessvorlage über das Attribut *Reason*. Neben den Unterschieden in der Bezeichnung sind auch die möglichen Zustandsübergänge und die zugehörigen Begründungen (Attribut *Reason*) jeweils unterschiedlich festgelegt.

Es lässt sich festhalten, dass die drei Prozessvorlagen unterschiedliche Ansätze im Umgang mit dem Entwicklungsprozess verfolgen. Während *Scrum* die leichtgewichtige Variante ist, bietet das *Agile* Template bereits ein paar mehr mögliche Eingabefelder. Das *Scrum Process Template* lehnt sich hingegen in seiner gesamten Terminologie sehr stark an den Scrum-Prozess an. Eine häufig unterschätzte Prozessvorlage ist das *CMMI Template*. Auch wenn die Begriffe auf den ersten Blick nicht auf die typischen agilen Prozesse schließen lassen, so eignet sich auch dieses Template, trotz einer höheren Anzahl an Work Item Types und verfügbaren Feldern, zur Durchführung agiler Projekte.

**Beispielprojekt**

Nachdem wir nun gesehen haben, welche Prozessvorlagen im Standard bereits von Microsoft im TFS mitgeliefert werden, stellt sich nun die zentrale Frage: Wie fühlt es sich an, mit den einzelnen Templates zu arbeiten? Die Statusübergänge und auch die Feldbezeichnungen werden jeweils aus der Terminologie der etablierten Prozesse übernommen. Eine Anforderung im Scrum Template heißt Product Backlog Item. In der CMMI-Prozessvorla-

	Visual Studio Scrum	MSF for Agile Software Development	MSF for CMMI Process Improvement
Workflow States	<ul style="list-style-type: none"> <li>• New</li> <li>• Approved</li> <li>• Committed</li> <li>• Done</li> <li>• Removed</li> </ul>	<ul style="list-style-type: none"> <li>• New</li> <li>• Active</li> <li>• Resolved</li> <li>• Closed</li> <li>• Removed</li> </ul>	<ul style="list-style-type: none"> <li>• Proposed</li> <li>• Active</li> <li>• Resolved</li> <li>• Closed</li> </ul>

Tab. 1: Workflow States, eigene Darstellung basierend auf [Microsoft14]

ge ist nach Fertigstellung einer Anforderung die Rede von Closed und nicht von Done. Doch die Unterschiede sind in der Handhabung marginal.

Als Beispiel verwenden wir ein Team-Projekt, das auf dem „MSF for CMMI Process Improvement“ Template basiert, und wenden den Scrum-Prozess darauf an. Wie bereits in [Abbildung 1](#) zu sehen, fehlt der CMMI-Prozessvorlage der Work Item Type „Impediment“ – ein elementarer Bestandteil des Scrum-Prozesses. Ein Impediment stellt ein Hindernis dar, das einen Entwickler von der Weiterführung seiner Arbeit abhält. Das können fehlende Informationen, ein Defekt in einer Hardware oder unvorhergesehene Schwierigkeiten bei der Implementierung sein, also grundsätzlich alle Punkte, die das Projekt maßgeblich negativ beeinflussen, wenn sie nicht behoben werden. Wir können in der CMMI-Prozessvorlage statt dessen das Work Item des Typs Risk finden und dieses als Substitut für das Impediment verwenden. Darüber hinaus setzen wir mit dem Risk Work Item auch die klassische Risikobetrachtung an.

In [Abbildung 2](#) sehen wir ein Requirement (CMMI), das mit Hilfe einer Anforderungsschablone eine Anforderung an das Produkt formuliert. Diese Schreibweise haben wir uns aus dem agilen Werkzeugkasten abgeschaut. Als Werkzeug haben wir hier die kostenfreie Erweiterung WordToTFS [AIT01] verwendet. Die klas-

AIT GmbH & Co. KG  
www.aitgmbh.de | info@aitgmbh.de



### 1 Functional Requirements

Requirement	Functional	
Customer Data Import		
As a Customer I want to import my CSV based data so that I can start working with the new tool.		

Abb. 2: Mit WordToTFS Anforderungen ganz klassisch in einem Word-Dokument

sische Aufnahme der Anforderung in einem Word-Dokument bietet die Möglichkeit, noch vor dem Übertragen der Anforderung in den TFS die Daten mit dem Kunden zusammen offline und weitestgehend vom Tool unabhängig zu besprechen. Auch kann ein Auditierungsprozess oder eine Abnahme durchgeführt und ein Angebot, das auf diesen Anforderungen beruht, erstellt werden.

Nachdem das Dokument für alle Seiten verabschiedet ist, kann es in den TFS übertragen werden. In [Abbildung 3](#) sehen wir dieselbe Anforderung im TFS Web Access im Product Backlog. Eine wichtige Frage ist sicherlich immer, welche Anforderung zuerst umgesetzt wird. Hierfür gibt es unterschiedliche Methoden, wobei *High Risk First* sicherlich eine der bekanntesten ist. Am Ende entscheidet aber der Projektleiter, was nach seiner Bewertung das beste Kosten-/Aufwand-Verhältnis bietet.

Was als Nächstes implementiert werden soll, steht im Backlog ganz oben. Egal in welchem Prozess – egal wie wir es nennen. Der nächste Schritt liegt nun in der Aufspaltung der Anforderung in umsetzbare Aufgaben. Eine Aufteilung in Implementierung, Dokumentation und Testen ist sicher eine mögliche Variante. Im Scrum-Prozess werden diese drei Teile zusammengefasst und in der „Definition of Done“ manifestiert: Eine Aufgabe ist erst dann abgeschlossen – also „Done“ –, wenn sie implementiert, dokumentiert und ausreichend getestet wurde.

Was im Scrum der Sprint ist und in einem agilen Umfeld eine Iteration als zeitlicher Bezug, ist in vielen klassischen Projekten eine Phase oder ein Meilenstein. Es handelt sich um die Projektkoordinate, wann etwas gemacht werden soll.

Unabhängig von Iteration, Sprint oder Phase stellt sich die Frage: Wie können die einzelnen Anforderungen auf die Zeit-

The screenshot shows a web interface for a TFS Backlog. On the left, there is a navigation menu with 'Features' and 'Requirements' selected. Below it, a list of iterations is shown: 'Aktuell' (Iteration 0), 'Zukunft' (Iteration 1, 2). The main area displays a 'Requirements' backlog. A modal window is open for adding a new requirement. The modal has fields for 'Typ' (set to 'Requirement'), 'Title' (empty), and 'Requirement Type' (set to 'Functional'). Below the modal is a table of existing requirements:

Rei...	Work Item Type	Title	State	Size	Requirement Type
1	Requirement	Customer Data Import	Proposed		Functional
2	Requirement	UI for Customer Management	Proposed		Functional
3	Requirement	Modify Customer Data	Proposed		Functional
4	Requirement	Save new Customer Data	Proposed		Functional
5	Requirement	Customer Data Backup	Proposed		Functional
6	Requirement	Customer Data Export	Proposed		Functional

Abb. 3: Das Backlog hilft dabei, den Überblick zu behalten

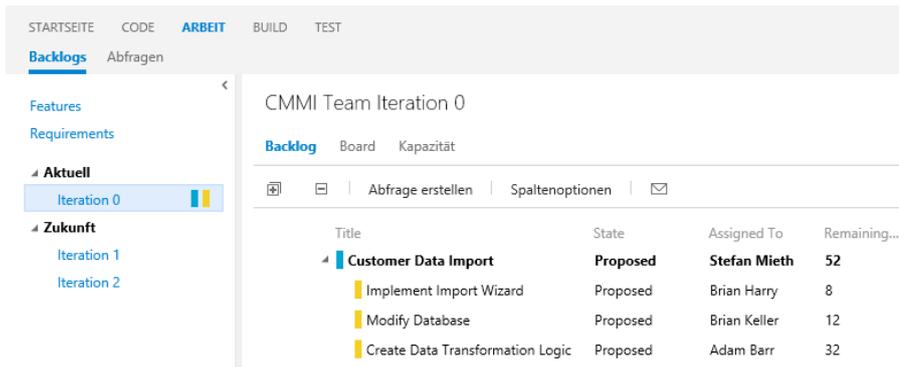


Abb. 4: Die aktuellen Aufgaben im Fokus: das Iteration Backlog

Slots verteilt werden? Hier bietet der TFS die einfache Möglichkeit via Drag'n'Drop die Work Items aus dem Product Backlog in die einzelnen Sprint- oder Iterations-Backlogs zu verschieben. Wie viel Arbeit in einer solchen Zeiteinheit zu erledigen ist, bestimmt der Prozess. Wo im Scrum das Team selbst entscheidet, wie viel machbar ist, und hierzu eine Abschätzung abgibt, wird der Umfang in Phasenbestimmten Projekten von außen vorgegeben. Und hierin liegt die Krux: Werden die Time-Box und der Umfang von außen vorgegeben, müssen Abstriche an der Qualität, Dokumentation oder anderen empfindlichen Stellen in Kauf genommen werden.

Spätestens, wenn die Antwort auf das Wann „jetzt“ lautet, müssen die konkreten Schritte festgelegt sein: die Tasks.

In **Abbildung 4** sehen wir die Anforderung mit der Aufschlüsselung und Verwaltung der Aufgaben. Auch hier können wir wieder auf die Features des WebAccess zurückgreifen. Das Backlog Board bietet die perfekte Grundlage sowohl für ein Daily Scrum also auch für ein wöchentliches Status-Meeting. Wer eine Liste bevorzugt,

für den gibt es zusätzlich die hierarchische Tabellenansicht.

Die Aufgaben sind somit verteilt und die Implementierung kann beginnen. Wir haben den Weg von einer Idee über eine Anforderung bis hin zur ersten Zeile Code hinter uns. An diesem Punkt heißt es nun: `while(RemainingWork > 0){ workOnIt; }`

Wir haben uns aus dem vielfältigen Werkzeugkasten der möglichen Methoden bedient: Gelebtes Scrum, Methoden aus dem allgemeinen agilen Umfeld, der Projektleiter hat sich an die „High Risk First“-Regel gehalten und CMMI „Style“ Requirements in eine WBS (Work Break Down Structure) gebracht.

**Fazit**

Abschließend lässt sich festhalten, dass auch oder gerade bei größerem Informationsbedarf agile Projekte ein Ansatz sind. Bei der Auswahl einer Prozessvorlage im Team Foundation Server sollte man sich also nicht von den Begriffen *Scrum*, *Agile* und *CMMI* leiten lassen, sondern vielmehr die einzelnen Prozessvorlagen inhaltlich gegenüberstellen. Mit der Prozessvorlage, welche die größten Überschneidungen zum eigenen Softwareentwicklungsprozess hat, kann man dann die Arbeit beginnen.

Auch wenn wir im Artikel von der CMMI-Prozessvorlage in unangepasster Form ausgegangen sind, so ist es absolut üblich, auch noch Änderungen an der Prozessvorlage vorzunehmen. Hierbei hat man durch geschicktes Wählen der Ausgangsvorlage jedoch die Chance, sich „nur noch“ mit den Abweichungen zu befassen. Die Vorlage kann also eine Art Rahmen bzw. einen roten Faden vorgeben, den man nutzen kann, um den Implementierungsaufwand für die Abbildung des eigenen Prozesses möglichst gering zu halten.

Wir müssen uns also keineswegs von der lieb gewonnenen Agilität verabschieden. Wie der Artikel gezeigt hat, lassen sich agile Projekte auch oder eben gerade in den informationsintensiveren Szenarien durchführen.

**Literatur & Links**

[AIT01] AIT WordToTFS, Team Foundation Server Tools, <http://aitgmbh.de/index.php?id=222>

[Microsoft14] Work with team project artifacts, choose a process template, MSDN, siehe: <http://msdn.microsoft.com/en-us/library/ms400752.aspx>

[SchSu11] K. Schwaber, J. Sutherland, The Scrum Guide. Scrum.org, siehe: <http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-US.pdf#zoom=100>



Das AIT-Team ist eine Gruppe von Experten für individuelle Softwarelösungen sowie Application Lifecycle Management (ALM) auf Basis der Microsoft-Technologieplattform. <http://www.aitgmbh.de>