



□ Clemens Reijnen

(E-Mail: Clemens.Reijnen@Sogeti.nl)

ist Berater bei Sogeti und erhielt die Auszeichnung Microsoft Most Valuable Professional (MVP) von Microsoft. Seine Spezialisierung liegt im Application Lifecycle Management. Als Trainer für Professional Developer Scrum verfügt er über ein tiefes Fachwissen in der Softwareentwicklung. Clemens Reijnen ist Mitautor des Buches „Collaboration in the Cloud“.

## TMap® mit Microsoft Visual Studio® 2010 – Scrum 1.0-Prozess-Template

Dieser Artikel erläutert, wie sich TMap-Testprozesse und -Verfahren in das Scrum-Prozess-Template integrieren lassen und wie sie innerhalb von Visual Studio 2010 durchgeführt werden.

### Testverfahren

So wie es Engineering-Verfahren in agilen Umgebungen gibt, wie Test Driven Development (TDD), Continuous Integration (CI) und Architekturverfahren wie aufkommende Architekturen ohne Big Design Up Front (BDUF), gibt es auch Testverfahren in agilen Umgebungen. Alle diese verschiedenen Verfahren helfen dem Scrum-Team, ein Produkt nach den Prioritäten und Abnahmekriterien des Product Owners und im Einklang mit der Definition des ‚done in the smartest way‘ zu entwickeln.

Ein Beispiel agiler Testverfahren ist das bekannte *frühe und häufige Testen*, ein häufig anzutreffendes Verfahren, das jedoch nicht immer leicht zu realisieren ist. Andere Verfahren sind: „Frühestmögliche Automatisierung“, „Kein Risiko, kein Test“, und weitere Verfahren, die im Testansatz von TMap zu finden sind. Aber wie lassen sich diese Testverfahren, zusammen mit einer Scrum- und risikobasierten Denkweise, am effizientesten durchführen, insbesondere mit Visual Studio 2010 als Toolset?

### Frühes und häufiges Testen

Das meistbekannte Testverfahren ist *frühes und häufiges Testen* – also der Beginn des Testens so schnell wie möglich. Meistens wird dieses Verfahren mit TDD in Ver-

bindung gebracht. Aber *frühes und häufiges Testen* berücksichtigt auch andere Testaktivitäten, also Tätigkeiten wie die Definition von Risiken, den Aufbau der Umgebungen und die Erstellung von Testplänen, Skripts und anderem.

*Frühes und häufiges Testen* deckt alle Testaktivitäten ab. Es bedeutet, dass man an Tests arbeiten soll, sobald man über ein System nachdenkt. Innerhalb eines Scrum-Teams bedeutet dies, dass man Testwissen bereits beim Produktplanungsmeeting und beim Sprint-Planungsmeeting verfügbar haben soll. Es ist von wesentlicher Bedeu-

tung, Testwissen durch alle Projektphasen hindurch zur Verfügung zu haben.

### Testen während des Release-Planungsmeetings

Testwissen während des Release-Planungsmeetings umfasst meistens Aufgaben, die den Product Owner unterstützen. Während der Release-Planung definiert, priorisiert und pflegt der Product Owner Backlog-Einträge und pflegt sie während des Projektes (tatsächlich ist diese Wissensunterstützung während des gesamten Projektes erforderlich). Die wichtigste

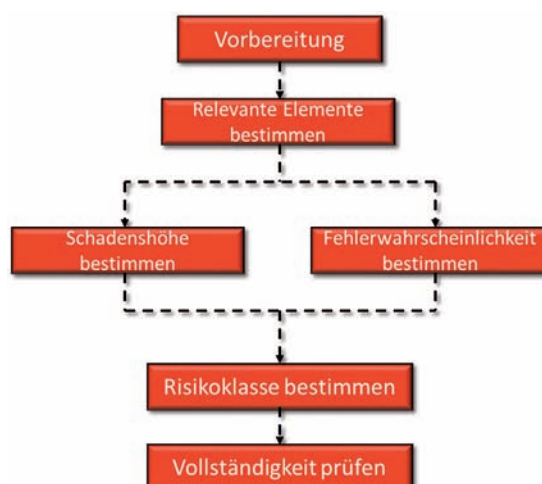


Abb. 1: TMap-Produktisikoanalyse

Wissensunterstützung betrifft das Risiko, da die wichtigsten Risiken für das Produkt definiert werden müssen.

Innerhalb des TMap-Testansatzes stellt die Produktrisikooanalyse eine wichtige Komponente dar. Bestimmte Risikoanalysen sind Bestandteil der empfohlenen Aktivitäten im Mastertestplan von TMap: „Analyse der Produktrisiken“. Sie unterstützen nicht nur den Product Owner dabei, die richtigen Entscheidungen zu treffen, sondern auch das Team in einer späteren Phase. Diese Informationen sind von unschätzbarem Wert bei der Definition der richtigen Test Case Design-Techniken für den Product Backlog-Eintrag. Die Bestimmung der Produktrisiken in dieser Phase liefert auch Input für die „Definition of Done List“.

Innerhalb des Visual Studio Scrum 1.0-Prozess-Templates werden Product Backlog-Einträge in der Arbeitsobjektart „Product Backlog-Eintrag“ aufgeschrieben. Diese Arbeitsobjektart hat keine spezifische Feld- oder Risikoklassifizierung. Das Hinzufügen eines Risikofeldes ist zu empfehlen (TFS Powertools erleichtern diese Aufgabe), sodass diese Eigenschaft abgefragt werden kann, oder die Risikoanalysen zu einer mehr produktgenerischen Eigenschaft werden.

Nicht nur die Risikoklassifizierung mit der Definition der größten Risiken ist eine Aufgabe, bei der Testwissen während des Release-Planungsmeetings hilfreich sein kann. Auch beim Definieren und Formulieren der Informationen zu den „Abnahmekriterien“ stellt Testwissen ein Muss dar.

### Testen während des Sprint-Planungsmeetings

Sprints beinhalten und bestehen aus dem Sprint-Planungsmeeting, der Entwicklungsarbeit, dem Sprint-Review und der Sprint-Retrospektive. Während des ersten Teils des Sprint-Planungsmeetings werden zu realisierende Product Backlog-Einträge ausgewählt, während des zweiten Teils werden diese in Aufgaben heruntergebrochen und geschätzt (falls noch nicht geschehen). Somit trifft das Team während des zweiten Teils Entscheidungen darüber, wie Backlog-Einträge zu realisieren sind und erstellt verpflichtende Aufgaben.

Im Rahmen des TMap-Lifecycle-Modells ist dies der erste Schritt – die Planungsphase. Dieser Schritt umfasst die Verantwortlichkeiten zur Beschreibung von Aufgaben, wie Funktionalitäten getestet

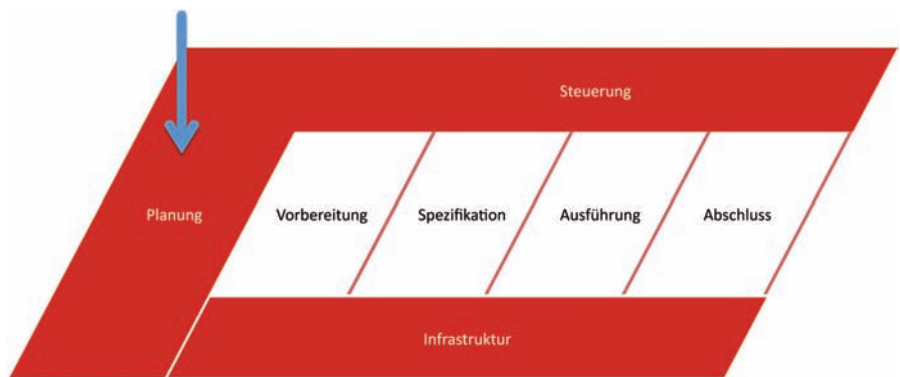


Abb. 2: TMap-Lifecycle

werden, die realisiert werden sollen. Dieses *wie* beinhaltet Entscheidungen auf der Grundlage der Risiken (diese Information findet man im Product Backlog) und umfasst Testdesign-Techniken, wie Testfälle erstellt werden sollten, und alle weiteren Aufgaben, die erforderlich sind, um mit der ‚Definition of Done‘ im Einklang zu stehen.

Somit beschreibt das Team, *wie* das System aussehen und *wie* es während des Sprint-Planungsmeetings getestet wird.

Während dieses Meetings werden Arbeitsobjekte vom Typ Task im Visual Studio erstellt. Die verbleibende Arbeit wird geschätzt, sodass der Sprint Burn-Down nachverfolgt werden kann. Diese Aufgaben können verschiedene Arten von Aktivitäten umfassen, wenn das Feld ‚Activity‘ dafür genutzt wird. Um diese Informationen nachzuvollziehen, können

Abfragen durchgeführt werden.

Visual Studio enthält das Wissen über verbundene Arbeitsobjekte, wodurch Aufgaben mit Product Backlog-Einträgen verbunden werden können. Diese Verbindung schafft die Fähigkeit, den Arbeitsfortschritt bei der Durchführung des Backlog-Eintrags nachzuverfolgen.

### Testbasisbestimmung

In der Vorbereitungsphase des TMap-Lifecycles wird ein weiteres Verfahren des *frühen und häufigen Testens* beschrieben. Während dieser Phase beginnt das Teammitglied, das sich zur Durchführung der Aufgabe ‚Testfall erstellen‘ verpflichtet hat, mit der Vorbereitung seiner Testaktivität. Eine von TMap vorgeschlagene Tätigkeit zu dieser Vorbereitung ist: Testbasis bewerten, prüfen, ob die für die Erstellung der Testfälle erforderlichen Informationen

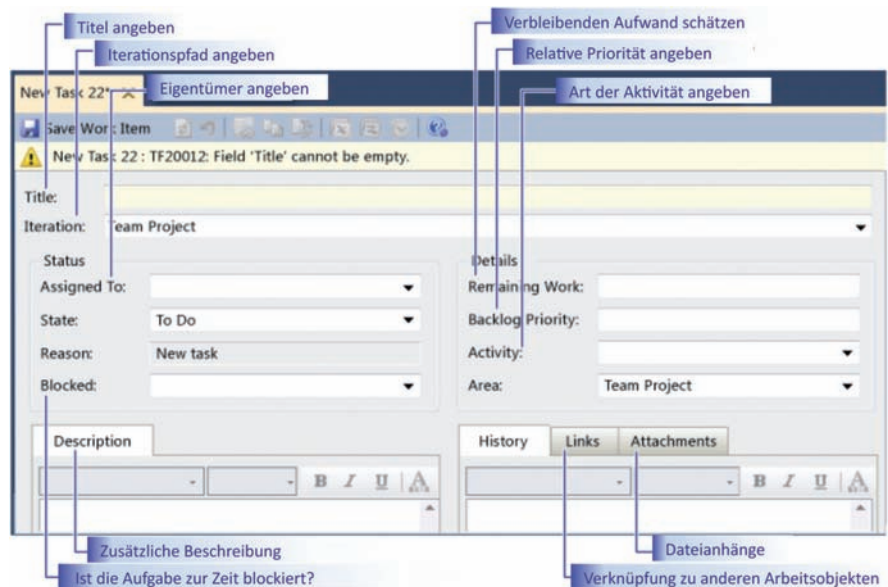


Abb. 3: Visual Studio 2010 Scrum 1.0 Task Work Item Type

(d. h. Dokumentation, Modelle und Beschreibung des Backlog-Eintrages) testbar sind. Testbarkeit bedeutet hier Vollständigkeit, Konsistenz, Erreichbarkeit und Übersetzbarkeit in Testfälle.

Dieses Testverfahren hilft, Fehler zu einem frühestmöglichen Zeitpunkt anzugehen (*frühes und häufiges Testen*), da die Teammitglieder, welche die Funktionalität realisieren, auch dieselben Informationen für ihre Aufgaben nutzen. Unvollständigkeit oder Inkonsistenzen können während des täglichen Scrum oder im Team erörtert werden.

Die Phasen Spezifizierung, Durchführung und Abschluss des TMap-Lifecycle-Modells werden während der Entwicklung des Sprints durchgeführt, da sie als Aufgaben definiert sind. Somit deckt das *frühe und häufige Testen* nicht nur die Unit-Tests ab, sondern den Prozess in jedem seiner Teile.

**Nur dann testen, wenn ein geschäftliches Risiko besteht**

Ein wichtiger Denkansatz des Scrum ist der, dass ,nur solche Aufgaben es wert sind, durchgeführt zu werden, die dem Produkt einen „Mehrwert“ verleihen. Dies gilt nicht nur für Entwicklungsaufgaben, sondern auch für Architekturaufgaben, aber in jedem Falle auch für Testaufgaben. Ein Grundgedanke von TMap ist, dass Tests ,geschäftsbasiert‘ sein sollen. TMap baut also auf einem geschäftsbasierten Ansatz auf (Business-Driven Test Management, BDTM). Das einfache Statement ,no risk, no test‘ sagt aus, dass das System nicht getestet werden muss, wenn kein geschäftliches Risiko besteht.

**Business-Driven Test Management (BDTM)**

Der gesamte Testaufwand steht in Bezug zu den Risiken des Systems, das für die Organisation zu testen ist. Die Schritte im BDTM-Prozess werden in **Abbildung 4** dargestellt.

Nicht jeder dieser Schritte passt zu einem Scrum-Projekt, aber die Denkweise von BDTM passt genau. Wenn Aufgaben für die Testaktivitäten definiert werden, lohnt es sich, einen genauen Blick auf BDTM zu werfen.

Innerhalb des Visual Studio 2010 Scrum-Prozess-Templates kann man einen Blick auf die Felder ,Geschäftswert‘ und ,Risiko‘ werfen (wenn hinzugefügt), um zu entscheiden, ob dieser Product Backlog-

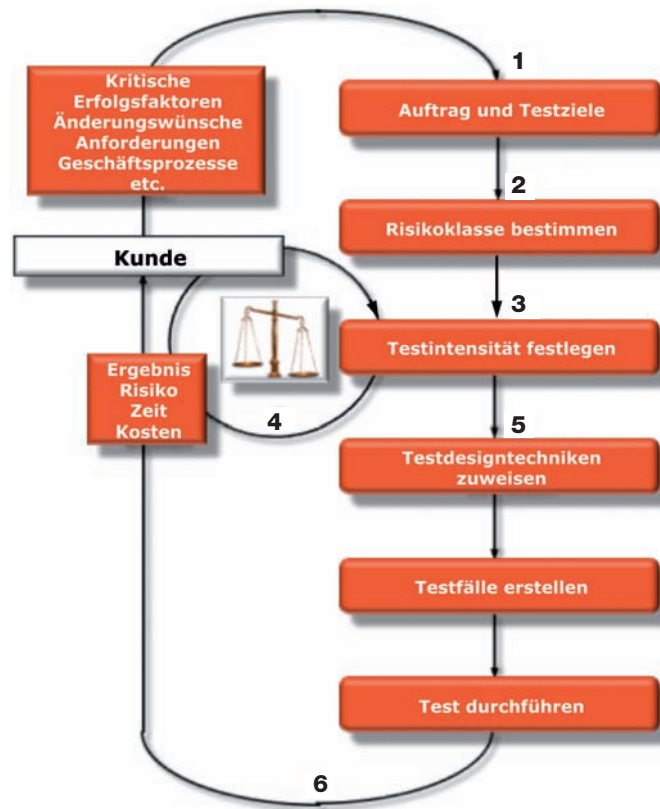


Abb. 4: Schritte im BDTM-Prozess

Eintrag gründlich oder oberflächlich zu prüfen ist. Beim Einfügen des Risikofeldes können Abfragen durchgeführt werden, um alle Product Backlog-Einträge mit einer hohen Risikoklassifizierung und mit Testfällen mit hoher Priorität zu erfassen. Dies sorgt für einen guten Überblick über die Qualität der Testaktivitäten.

BDTM-Testverfahren und Visual Studio 2010-Fähigkeiten zur individuellen Anpassung von Arbeitsobjektarten und zur Erstellung individueller Abfragen können Projekte mit dem richtigen Know-how versorgen, um Testaktivitäten mit einer Scrum-Denkweise durchzuführen.

**Frühestmögliches Automatisieren**

Eine frühestmögliche Automatisierung der Tests hat für das Team den Vorteil, dass es sich auf die Qualität fokussiert. Das Team weiß, dass das zu entwickelnde System noch wie abgenommen funktioniert, und wird benachrichtigt, wenn das nicht der Fall ist. Sie bietet auch einen Vorteil für den Product Owner, weil er auf einfache Weise sehen kann, welche Funktionalität bereits realisiert wurde und ob sie wie abgenommen funktioniert.

Visual Studio 2010 verfügt über einen

perfekten Mechanismus zur Automatisierung von Testfällen, nämlich die Coded UI-Tests. Zur Vorgehensweise bei der Erstellung von automatisierten Tests siehe auf MSDN „Testen der Anwenderschnittstelle mit automatisierten UI-Tests“ und das YouTube-Video „Create CodedUI“.

Jeden Testfall zu automatisieren wäre zu viel des Guten, da einige Testfälle nur sehr schwer zu automatisieren sind oder eine Automatisierung zu kostenintensiv oder sinnlos wäre. Jede Aufgabe innerhalb eines Scrum-Projektes sollte dem Produkt einen Mehrwert liefern. Es ist deshalb nicht damit getan, alles zu automatisieren. Die Aufgabe besteht darin, zu entscheiden, welche Tests automatisiert werden sollen. Sicherlich sind automatische Tests für die wichtigsten Szenarien, die bei jedem Sprint laufen sollen, erwünscht. Tatsächlich müssen die ausgewählten Tests für die Regressionstests automatisiert werden. Welche Tests für diesen Regressionstest auszuwählen sind, ist im TMap-Lifecycle beschrieben. Während der Abschlussphase werden Testfälle für die Regression ausgewählt.

Auch die Abfragemöglichkeiten von Visual Studio 2010 können verwendet wer-

den, um mögliche Testfälle für die Automatisierung auszuwählen. Beispielsweise stellt eine Abfrage wie „hochriskanter Product Backlog-Eintrag mit Test Cases mit hoher Priorität“ eine interessante Auswahl für die Automatisierung dar.

### **Qualitativ hochwertige Testfälle**

Kein wirklich agiles Testverfahren, sondern eher ein Testverfahren nach gesundem Menschenverstand – in agilen Umgebungen ist es jedoch ein Muss, qualitative hochwertige Testskripts erstellt und für jedes Teammitglied verfügbar zu haben.

Viele Testskripts kommen für die Automatisierung in Frage. Somit ist es für die Automatisierung wenig hilfreich, ein Testskript mit nur einem Schritt, z. B. ‚den Bildschirm testen‘ zu haben. Gut beschrie-

bene Tests, die in deutliche Schritte heruntergebrochen werden, sind da wesentlich besser. Gut beschriebene Testfälle erklären auch, welche Risiken sie abdecken.

Die Entwicklung von Testfällen in agilen Umgebungen gilt oft als eine große Herausforderung, da die Dokumentation fehlt und die Spezifikationen sich ändern. Meistens wird dies als Grund angeführt, weshalb das Testen in agilen Umgebungen nicht funktioniert. Ebenso wie aufkommende Architekturen sollten Testfälle gut designed sein, damit sie sich entwickeln können. Um die notwendige Flexibilität in den Testfällen zu erlangen, ist ein gutes Testfall-Managementsystem wie der Microsoft Test Manager ein Muss.

### **Schlussbetrachtung**

Die Integration von Testverfahren in einer

agilen Umgebung ist ein Muss – abgesehen davon, dass sie dem Scrum-Team das Leben leichter macht, steigert sie darüber hinaus die Qualität des Systems. In diesem Artikel wurden die gebräuchlichsten Testverfahren für Scrum-Teams dargestellt.

Das Visual Studio 2010 Scrum-Prozess-Template unterstützt agiles Arbeiten. Wenn jedoch Teams das Testen als eine getrennte Tätigkeit begreifen, wird sich der Erfolg nicht einstellen, selbst wenn man das Scrum-Prozess-Template verwendet. Daher müssen Testverfahren wie die des TMap-Testansatzes eingebettet werden. Dies ist umso wichtiger, wenn alle Möglichkeiten der Visual Studio 2010 Application Lifecycle Management-Toolsuite genutzt werden sollen, wo Testunterstützung ein First Class Object darstellt. ■

**Textübersetzung: Walter Löpsinger**

---

TMap® ist ein eingetragenes Warenzeichen der Sogeti Nederland B.V.