Nico Orschel

(E-Mail: nico.orschel@aitag.com)
ist Microsoft Certified Trainer und Berater des AIT TeamSystemPro Teams. Er hat sich unter anderem auf das automatische Testen mit dem Visual Studio Lab Management spezialisiert. Er berät Unternehmen bei der Einführung und Anpassung des Visual Studio Team Foundation Server. Seine Erfahrungen vermittelt er zudem als Autor des TFS-Blogs (tfsblog.de)

Ausweg aus der Kommunikationskrise oder das Ende von „Bei mir funktioniert's"?

Die Auslieferung von hoch qualitativen Anwendungen ist das Ziel von allen Softwareentwicklungshäusern. Über die Zeit haben sich zwei Gruppen (Entwicklung und Qualitätssicherung) herausgebildet, welche für ihre Arbeit wiederum eigene Tools eingesetzt haben. Die eingeführten Werkzeuge bildeten Informationsinseln mit vielen Medienbrüchen und ohne große Möglichkeiten des Austauschs. Was folgt sind erhöhte Kommunikationsbarrieren zwischen Entwicklung und Test, die die Arbeit ineffektiv machen. Die Probleme spitzen sich im sogenannten „Bug Pingpong" zu, indem sich Qualitätssicherung und Entwicklung gegenseitig den Schwarzen Peter für fehlerhafte Software zuspielen. Im Artikel wird eine durchgängige Informations- und Kollaborationskette (vgl. (1)) unter Verwendung der Visual Studio 2010 Plattform (vgl. (2)) aufgezeigt. Es werden dabei zusätzlich Best Practices zum Testmanagement, manuellen und automatischen Testen mit der ALM-Plattform (Application Lifecycle Management) aus dem Hause Microsoft vorgestellt. Das Ziel des Artikels ist es, einen Weg aufzuzeigen, um die Kommunikationsbarrieren zwischen Entwicklung und Qualitätssicherung abzubauen.

Der Anfang ... Definition und Planung von Anforderungen

Zu Beginn jeder Softwareentwicklung werden in allen Softwarehäusern zunächst Anforderungen an die Software definiert. Art, Umfang und Begrifflichkeiten der Anforderungen können dabei zwischen den verschiedenen Vorgehensmodellen variieren. Im Kontext der Entwicklung mit dem Team Foundation Server (kurz: TFS) werden die Anforderungen mit bekannten Tools wie Visual Studio 2010, Excel oder Word erfasst. Für das zuletzt genannte Tool hat die AIT AG das Addon „WordToTFS" (vgl. (3)) zur Synchronisation von Anforderungen zwischen Word Dokument und Team Foundation Server erstellt (siehe Abbildung 1 – WordToTFS).

Die Definition und Erfassung führen dabei das Produktmanagement sowie die Projektmanager von Entwicklung und Qualitätssicherung gemeinsam durch. Alle im TFS gespeicherten Requirements werden dabei als sogenannte Work Items gespeichert. Für einen tieferen Einblick in das Thema Requirements Management mit dem TFS möchte ich Sie gerne auf den Artikel „Durchgängiges Requirements Management" (vgl. (4)) von meinem Kollegen Herrn Hubert verweisen.

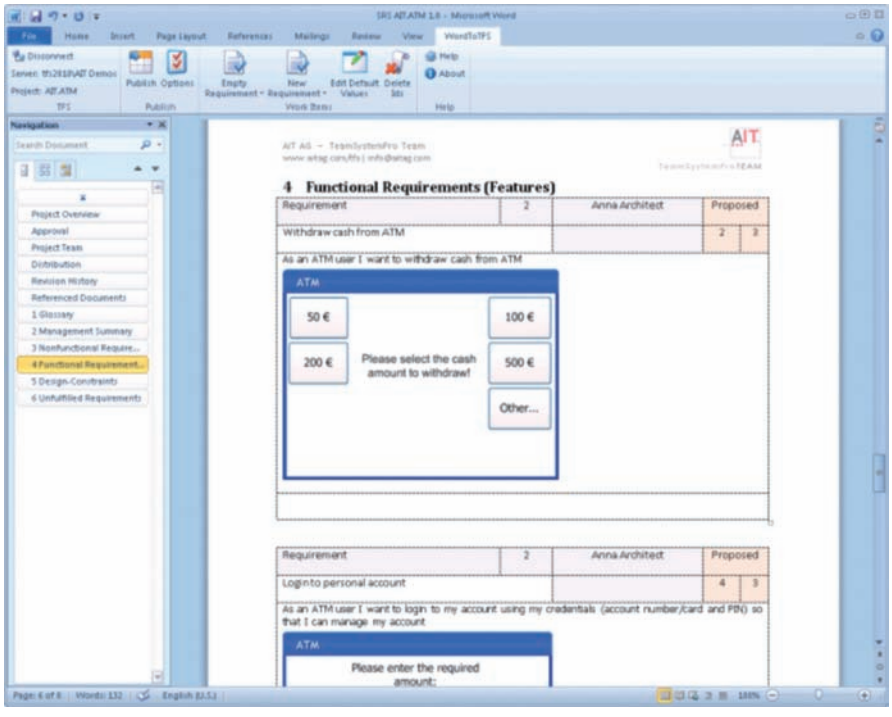

Abbildung 1 - WordToTFS

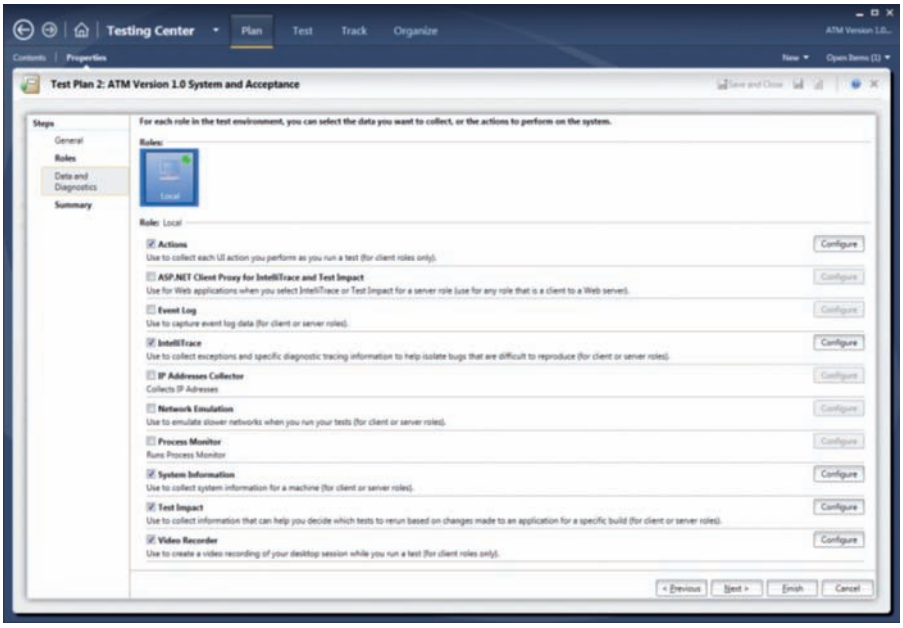

Abbildung 2 - MTM - Daten und Diagnose

Definition und Planung von Testfällen

Nachdem die Anforderungen im Team Foundation Server erfasst sind, beginnt die Arbeit der Qualitätssicherungsabteilung (kurz: QS) auf Seiten des Softwarehauses, aber auch je nach Projektsituation unter Einbeziehung des Kunden. Mit der Veröffentlichung der Visual Studio 2010 Produktfamilie bekommt unsere QS ein eigenes Werkzeug für das Testmanagement, dem Microsoft Test Manager 2010 (kurz: MTM). Der MTM besteht aus zwei Bereichen, Testing Center und Lab Center. Zunächst bewegen wir uns nur im Testing Center, denn dieser ist für das Management von Tests und das manuelle Testen verantwortlich. Das Lab Center hingegen ist für die Verwaltung von Testumgebungen für QS und Entwicklung zuständig.

Im MTM arbeitet die QS in sogenannten Testplänen. Der Testplan definiert den Rahmen für das Testen wie Team Projekt, Start- und Enddatum einer Testphase, Testplanverantwortlicher, Testkonfigurationen sowie die bei der Testausführung zu erfassenden Diagnosedaten.

Die Testeinstellungen definieren, welche Informationen bei der Ausführung von Tests automatisch im Hintergrund aufgezeichnet und im Fehlerfall reportet werden sollen. Es ist dabei möglich, z. B. alle Aktionen des Benutzers auf der Oberfläche aufzuzeichnen (ActionLog), ein Video des Testdurchlaufs zu erstellen, Daten aus dem EventLog einzusammeln, den aufgerufenen Code aufzuzeichnen (IntelliTrace) oder die Bandbreite der Netzwerkverbindung einzuschränken (siehe Abbildung 2).

Die aufgeführten Adapter sind dabei noch flexibel um eigene Adapter erweiterbar. Eine Beispielerweiterung ist der IP Address Collector vom Autor aus Abbildung 2. Je nach Produkt-, Test- und Projektstruktur definiert die QS ein oder mehre Testpläne pro Team Projekt. Ein Team Projekt definiert vereinfacht gesprochen einen Rahmen in Form von Work Item Typen, Reports, Source Control, Build-Prozessen, Testumgebungen sowie Testplänen für das gesamte Produktteam. In unserer Arbeit hat sich die Strukturierung und Benennung von Testplänen nach Produktversion und Testart etabliert.

Nachdem Testpläne erstellt wurden, geht es darum, die vielen Testfälle pro Testplan besser zu strukturieren. Pro Version unserer Software und Testart können schließlich mehre hunderte Testfälle existieren. Die Strukturierung innerhalb von Testplänen geschieht mit sogenannten Testsuiten. Der TFS kennt dabei drei Arten von Testsuiten: Testsuites, Query-based Testsuites und Requirements. Testsuites verhalten sich wie Ordner in Windows, d. h. sie ordnen Testfälle manuell zu bzw. entfernen sie manuell. In Query-based Testsuites werden Testfälle auf Basis von Abfragen (TFS: Work Item Queries) angezeigt. Erfüllt ein Testfall Work Item eine bestimmte Bedingung nicht mehr, so wird es nicht mehr in der Testsuite angezeigt. Eine Query-based Testsuite fungiert hier nur als eine dynamische Sichtweise auf eine große Anzahl an Testfällen. Eine Anwendung für die Query-based Testsuite wäre die Anzeige aller Testfälle mit hoher Priorität. Der dritte Testsuite-Typ ist ein Spezialfall. Die Anforderungen, welche wir bereits am Anfang unseres Entwicklungsprozesses im TFS erfasst haben, werden dabei als Ordner im Testplan abgebildet (siehe Abbildung 3).

Wenn jetzt die QS einen Testfall in diese spezielle Testsuite einordnet, so wird im Hintergrund automatisch eine Verknüpfung zwischen Anforderung und Testfall erstellt. Diese Verknüpfung steht in allen Tools, wie z. B. Visual Studio, Eclipse, Excel werkzeugunabhängig zur Verfügung. Durch die Verknüpfungen und Abfragen lassen sich jetzt auch Informationen gewinnen, wie „Welche Anforderungen haben Testfälle?“ oder noch interessanter „Für welche Anforderungen fehlen noch

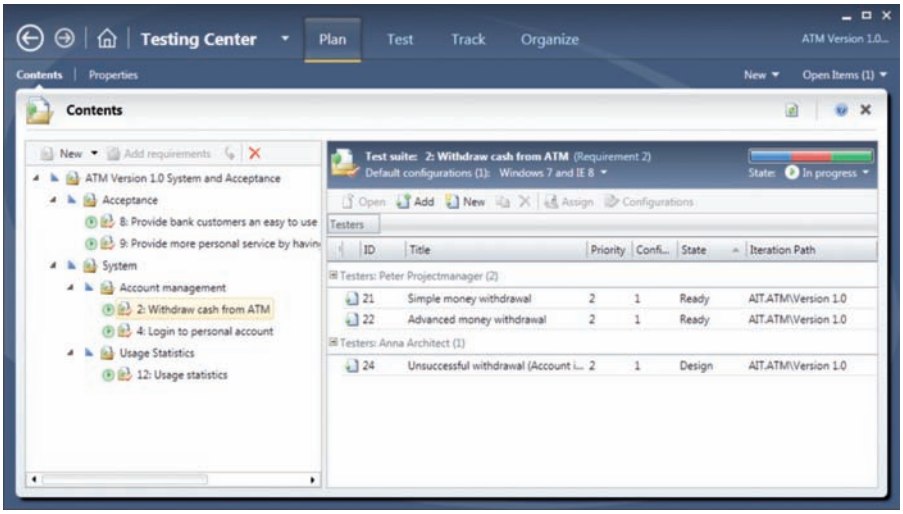

Abbildung 3 - Testpläne im MTM

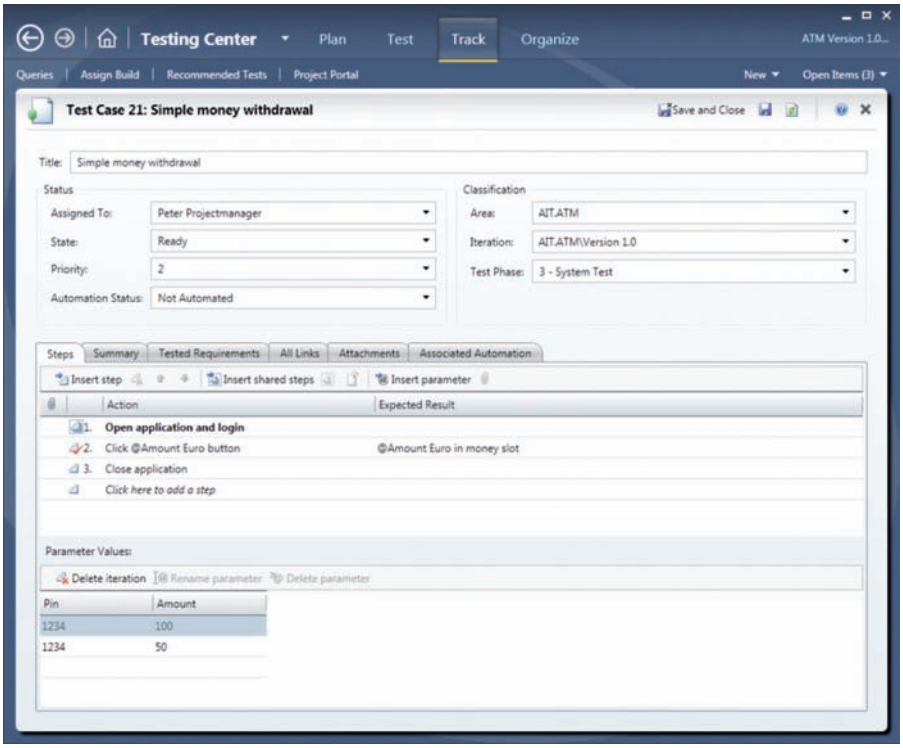

Abbildung 4 - Test Case Work Item

Testfälle?“. Die Navigation zwischen Anforderungen und Testfällen ermöglicht allen Beteiligten, den Kontext sowohl aus der Entwicklungs- und Testperspektive besser zu verstehen und so schlussendlich bessere Software durch bessere Kommunikation zu erstellen.

Die zuvor angesprochenen Testfälle sind im TFS auch nur Work Items. Ein Beispiel für ein Testcase Work Item zeigt Abbildung 4.

Ein Work Item setzt sich im TFS aus allgemeinen und spezifischen Informationen zusammen. Beispiele für allgemeine Informationen sind Bearbeiter, Titel und Zustand. In der Standarddefinition eines Testfalles sind die Zustände „Design“, „Ready“ und „Closed“ hinterlegt. Die wichtigsten spezifischen Informationen im Testfall Work Item sind Teststeps und Testautomation.

In den Teststeps werden Schritte beschrieben, welche ein Fachtester bei der Testdurchführung ausführen soll. Neben den Schritten kann dabei auch das erwartete Ergebnis angegeben werden. Wird ein solches Ergebnis für einen speziellen Testschritt angegeben, so wird von einem Validationstep gesprochen. Die eingegebenen Testschritte lassen sich noch weiter optimieren. Weil bei vielen Testfällen oft bestimmte Teilschritte immer identisch sind, lassen sich diese Schritte in einen speziellen Testfall Work Item Type mit dem Namen „Shared Step“ auslagern. Ein „Shared Step“ lässt sich in ein oder mehrere Testfälle einbinden. Ein typisches Anwendungsbeispiel ist der Login- Dialog bei einer Anwendung.

Der zweite Optimierungspunkt wäre die Parametrisierung unseres Testfalles. Durch das Anfügen eines @ - Zeichens können Parameter sowohl für den Testschritt als auch für das erwartete Ergebnis erzeugt werden. Durch Verwendung von Parametern muss nicht für jedes Testwertpaar ein neuer Testfall erzeugt werden (siehe Abbildung 4).

Der zweite wichtige Teil eines Testfall Work Items ist das Tab „Associated Automation“. Über dieses Tab Testautomation kann eine Beziehung zwischen einem automatisierten Test und einem Testfall im TFS hergestellt. Durch die Verknüpfung kann der automatisierte Testfall wie ein manueller Testfall verwaltet, ausgeführt und ausgewertet werden. Der Fachtester hat zudem den Vorteil, dass er bei Problemen mit dem automatisierten Test sofort auf das Visual Studio Projekt rückschließen kann, was gerade bei der Abstimmung mit dem verantwortlichen Entwickler eine Menge Zeit sparen kann.

Eine weitere Möglichkeit Ressourcen zu schonen, ist die Möglichkeit der Ermittlung von empfohlenen Tests auf Basis von geänderten Codezeilen (MTM: Recommended Tests).

Ausführung von Tests

Nachdem die Testfälle spezifiziert sind und sich im Zustand „Ready“ befinden, wechseln wir im MTM auf das Tab „Test“. Im Tab „Test“ finden wir erneut unsere Testsuiten aus der Planungsphase. In der aktuellen Ansicht haben Sie die Möglichkeit auf den Testernamen, sowie die zu testende Testkonfiguration zu filtern. Beispiele für Testkonfigurationen sind Betriebssystem oder Sprache. Ein Testfall kann einer oder

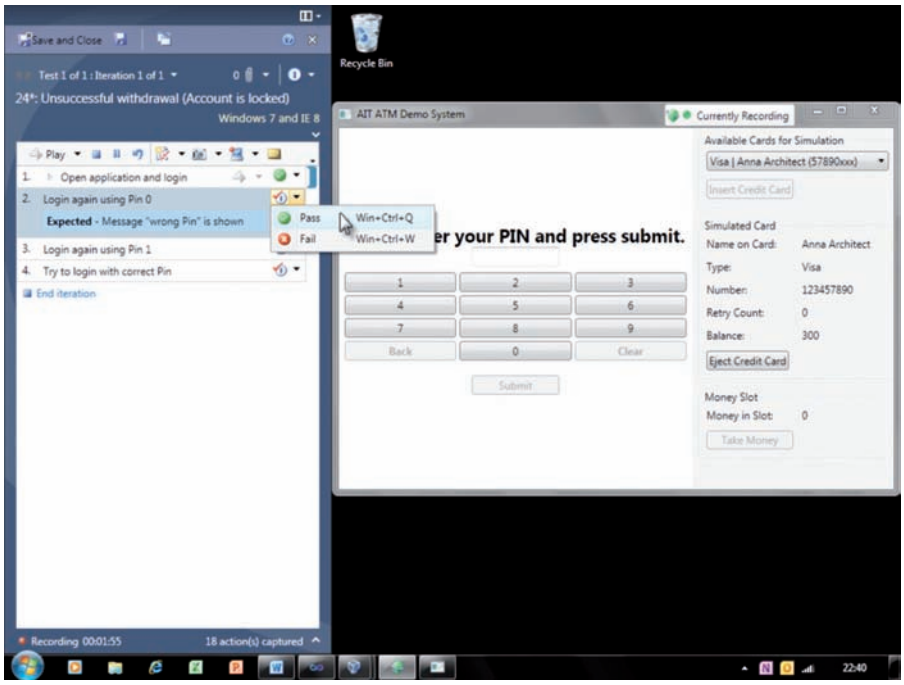

Abbildung 5 - Testrunner

mehreren Konfigurationen zugeordnet sein. Nachdem Tester und Konfiguration ausgewählt sind, können wir einen Testlauf (TFS: Testrun) starten. Im Rahmen eines Testruns kann ein spezieller Testfall oder alle Testfälle einer kompletten Testsuite ausgeführt werden. Bei der Ausführung einer Testsuite schaltet der MTM in den „Testrunner" Modus, dabei schaltet er erneut in eine optimierte Ansicht und verkleinert zusätzlich den Desktop des Testers um Übersicht zu bewahren (siehe Abbildung 5).

In diesem Modus führt der Tester alle Testschritte aus und meldet Erfolg oder Misserfolg mit Pass oder Fail. Aus dem Testrunner heraus kann der Tester im Fehlerfall direkt bei der Testausführung einen Bug erstellen. Wird ein Bug erzeugt, so beginnt der MTM sofort mit dem Einsammeln von Diagnosedaten. Ein Beispiel für einen solchen Richbug zeigt Abbildung 6.

Der mit umfangreichen Diagnosedaten, wie z. B. Video, EventLog, ActionLog und Testschritte mit Videozeitmarken und aktueller Testparameter angereicherte Bug wird als „Richbug" bezeichnet. In Kombination mit dem Lab Management geht der Prozess sogar soweit, dass an den Bug auch eine Verknüpfung zur passenden Testumgebung angefügt wird. Auf diese Weise erzeugte Bugs haben drastische Vorteile gegenüber Bugs aus klassischen Bug Tracking Systemen: Sie entlasten den Tester, indem der MTM das Einsammeln der technischen Diagnosedaten für die Testschritte automatisiert und so schlussendlich eine neue Art der Kommunikation zwischen Tester und Entwickler über Bugs ermöglicht. Der große Vorteil für den Entwickler wiederum besteht darin, dass er jetzt viele Informationen bekommt, um Fehler besser nachstellen zu können. Die Informationen außerhalb der rein textuellen Beschreibung wie Video, IntelliTrace Datei (vgl. (5)) und der mögliche Link auf verwendete Testumgebungen ermöglicht dem Entwickler, schneller Fehler zu finden. IntelliTrace Dateien stellen dabei eine Art Flugschreiber der Testausführung dar, d.h. der MTM oder Test Agent sammelt Informationen über den aufgerufenen Code während der Testausführung. Unter Verwendung des Visual Studio 2010 Ultimate kann der Entwickler die historische Aufzeichnung quasi schrittweise nachspielen und nachvollziehen, wie die Fehlersituation auf einem fremden Rechner entstanden ist, ohne das er am Testsystem angemeldet ist.

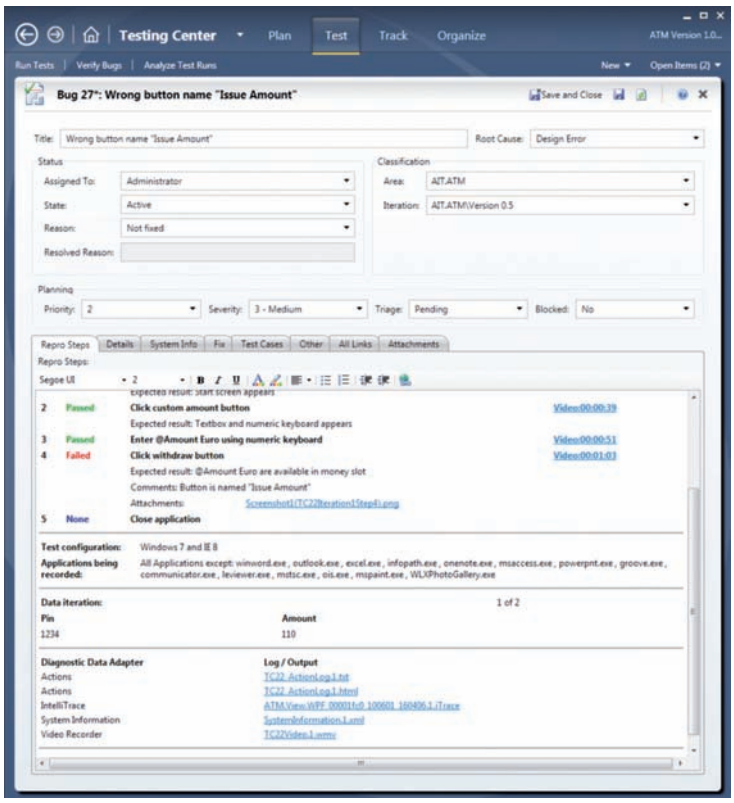

Abbildung 6 - Richbug

Neben der Erstellung von Bugs kann der Testrunner auch alle Aktionen auf der Oberfläche verfolgen und bei einem späteren Regressionstest erneut abspielen. Das erneute Nachspielen von Aktionen bis zu interessanten Testschritten wird als „Fast Forward Testing" (vgl. (6)) bezeichnet. Die so erzeugten Aufnahmen heißen im MTM ActionLogs und werden ebenfalls pro Testfall Work Item im TFS zentral gespei-

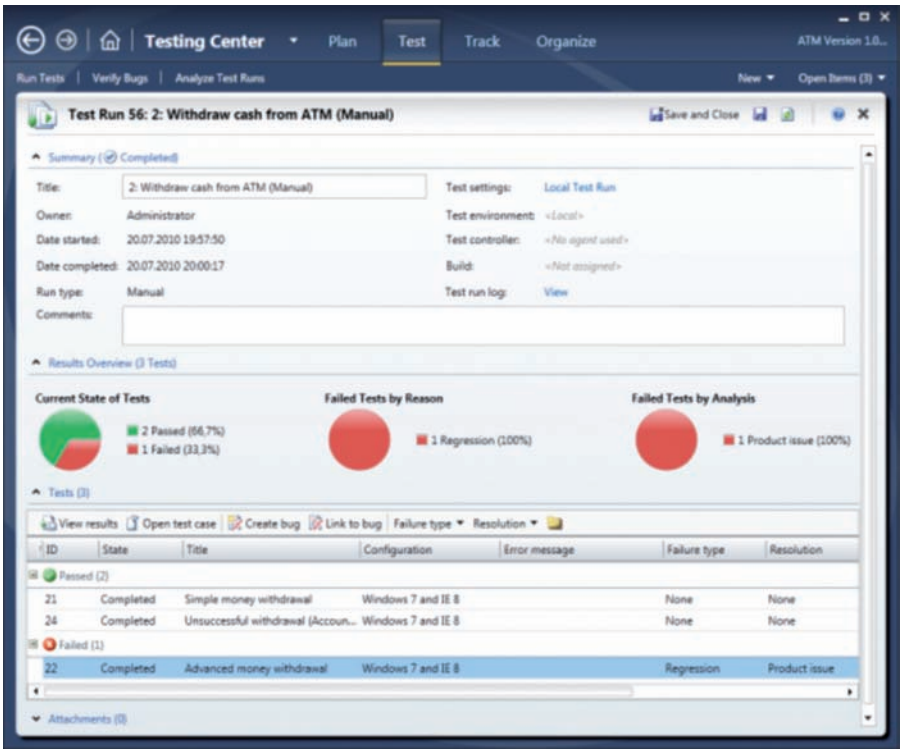

Abbildung 7 - Test Results

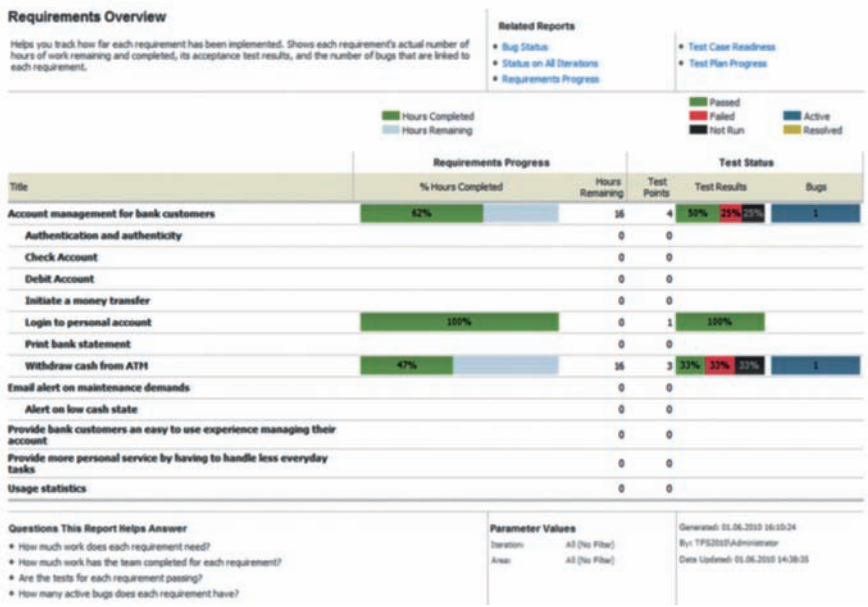

Abbildung 8 - Requirements Overview Report

chert. Damit der MTM Aktionen auf den Oberflächen verfolgen kann, müssen die Controls von Programmen entweder den Standard Microsoft Active Accessibility (kurz:MSAA) (vgl. (7))) oder Microsoft UI Automation (kurz: UIA (vgl. (8))) genügen. Aktuelle Microsoft .NET Frameworks Technologien wie z. B. WinForms 2.0 und neuer, WPF 3.0 und neuer und Webseiten (HTML/AJAX) im IE 8 werden bereits vollständig unterstützt (vgl. (9)). Weitere Programme und Technologien, wie Silverlight oder Webanwendungen im Firefox testen, werden in nächster Zeit ebenfalls möglich sein.

Auswertung von Testergebnissen

Alle Daten, welche sowohl bei der Planung und Ausführung von Tests entstanden sind, werden stets im TFS zentral abgelegt. Durch die zentrale Speicherung können alle Beteiligte die Ergebnisse von einzelnen Testläufen über den MTM einsehen (siehe Abbildung 7).

Aufgrund der Verknüpfung von Anforderungen und Testfällen ist des Weiteren die Auswertung von Testergebnissen im Kontext von Anforderungen möglich, wodurch sich Aussagen bzgl. der Qualität von einzelnen Anforderungen treffen lassen. Ein Beispiel für eine solche anforderungsbasierte Auswertung zeigt Abbildung 8. Sie haben über das Reporting aber auch die Möglichkeit, einfachere Auswertungen mit reinem Testfokus wie z. B. Testfortschrittsanalysen zu erstellen.

Fazit

Mit der neuen Visual Studio 2010 Familie hat es Microsoft geschafft, auch den Aspekt Testprozesse in die durchgängige Informationskette im Team Foundation Server zu integrieren. Testartefakte sind jetzt nicht mehr separat in eigenen Werkzeugen abgelegt, sondern stehen direkt in einer Beziehung zu Anforderungen, Bugs, Quellcode und Build-Prozessen. Diese tiefe Integration ermöglicht eine nie da gewesene direkte Zusammenarbeit und Kommunikation zwischen Entwicklung und QS.

Des Weiteren entlasten die neuen Testwerkzeuge den Tester durch einen hohen Automatisierungsgrad bei der Kommunikation mit dem Entwickler, indem die relevanten technischen Daten für den Entwickler im Hintergrund an den Entwickler extrahierbar sind. Diese sind auch für einen Fachanwender relativ einfach nachvollziehbar. Eine weitere Entlastung für alle Beteiligten lässt sich durch die Einführung von automatisierten Oberflächentests (TFS: CodedUI Tests) (vgl. (10)) um das Testen in virtuellen Testumgebungen (TFS: Lab Management) (vgl. (11)) erzielen. Der Einsatz vom Lab Management und Coded UI Tests ist nicht inhaltlicher Fokus dieses Artikels.

Zudem ist hervorzuheben, dass trotz der Integration jeder Anwender seine gewohnten bzw. speziellen Werkzeuge nutzen kann und alle dennoch über den zentralen TFS zusammenarbeiten können. ■

Referenzen

[1] Visual Studio 2010 Editions [Online] http://www.microsoft.com/visualstudio/en-us/products/2010-editions

[2] MSDN Webcast: Traceability mit Team Foundation Server 2010 - Die Möglichkeiten im Überblick [Online] http://www.microsoft.com/germany/events/eventdetail.aspx?EventID=1032456802

[3] AIT WordToTFS 2010 [Online] http://www.aitgmbh.de/word_to_tfs0.0.html?&no_cache=1

[4] Durchgängiges Requirements Management – Eine integrierte ALM-Werkzeugkette für das Requirements Management auf Basis des TFS 2010 [Online] http://www.sigs.de/publications/os/2010/RE/hubert_OS_RE_2010.pdf

[5] Debugging with IntelliTrace [Online] http://msdn.microsoft.com/en-us/library/dd264915.aspx

[6] Fast Forward Testing – Part 1 – The magic w(b)and. [Online] http://blogs.msdn.com/b/vstsqualitytools/archive/2010/01/07/fast-forward-testing-part-1-the-magic-w-b-and.aspx

[7] Microsoft Active Accessibility [Online] http://msdn.microsoft.com/en-us/library/ms971310.aspx

[8] Microsoft UI Automation [Online] http://msdn.microsoft.com/en-us/library/ms728097%28VS.85%29.aspx

[9] Platform Support for Coded UI Test (and Fast Forward feature of Test Runner) [Online], http://blogs.msdn.com/b/gautamg/archive/2010/01/07/platform-support-for-coded-ui-test-and-fast-forward-feature-of-test-runner.aspx

[10] Testing the User Interface with Automated UI Tests [Online] http://msdn.microsoft.com/en-us/library/dd286726.aspx

[11] Whitepaper: Lab Mangement [Online] http://www.aitgmbh.de/labmngwhitepaper (Veröffentlichungsdatum: 1.11.2010)