Florian Otto

(E-Mail: otto@hs-coburg.de)
Im März 2010 beendete Herr Dipl.-Inf. (FH) Florian Otto (Email:XXX)erfolgreich den Diplomstudiengang der Informatik an der Hochschule Coburg. Seine Abschlussarbeit: „Optimierung des Clusteralgorithmus ROCK[...]" steht im Zusammenhang mit einem Projekt der HUK COBURG, zur Entwicklung intelligenter Schutzmechanismen im Netzwerkbereich. Direkt im Anschluss daran begann er mit dem Master-Studiengang: „Elektro- und Informationstechnik" der Hochschule Coburg. Das Thema des Masterprojekts ist die Entwicklung und Evaluierung modellbasierter Testmethodiken für domänenspezifische Anforderungen aus der Enterprise-IT.

Florian Prester

(E-Mail: florian.prester@seppmed.de)
studierte an der Friedrich-Alexander Universität Erlangen-Nürnberg Informatik und war nach seinem Abschluss als wissenschaftlicher Mitarbeiter im Regionalen RechenZentrum der Universität beschäftigt. Seit 2007 ist er bei der sepp.med gmbh beschäftigt und baute dort die Firmeneigenen F&E mit Schwerpunkt auf der Produktentwicklung von .getmore auf. Seit 2009 ist Florian Prester Geschäftsführer der sepp.med gmbh mit den Aufgabenbereichen F&E & Vertrieb - kurz das operative Geschäft. Sein besonderes Augenmerk liegt auf der Entwicklung der Methodik modellzentriertes Testen (.mzT) und der Produktentwicklung des Testfallgenerators .getmore.

.mzT@BPMN: modellbasiertes Testen für die Enterprise-IT

Mit der Veröffentlichung des BPMN-Standards in Version 2.0 existiert eine Methodik zur grafischen Modellierung von Geschäftsprozessen, von den Anforderungen bis hin zum ausführbaren Prozessmodell. Neben den reinen Prozessabläufen lässt sich mit der BPMN - als Alternative zur UML – auch die daraus resultierende Enterprise-IT darstellen und für Verifikationszwecke und Tests visualisieren.
Um die Korrektheit, Qualität und Zuverlässigkeit der in BPMN annotierten Prozessspezifikation und der resultierenden IT zu garantieren, bedarf es geeigneter Testverfahren, die trotz hoher Komplexität der geprüften Inhalte einfach zu spezifizieren sind. In diesem Artikel werden hierfür modellbasierte Ansätze zur Testspezifikation vorgestellt und gezeigt, wie sich diese mit Hilfe der BPMN realisieren lassen.
Weiterhin werden das Modellierungswerkzeug *Innovator for Business Analysts* von MID, die Best-Practice-Methode modellzentrierter Test (.mzT) und der Testfallgenerator *.getmore* von sepp.med sowie das Case Tool *Team Foundation Server* von Microsoft als geschlossene Werkzeugkette für alle Aktivitäten von der BPMN Modellierung bis hin zum abschließenden Test der Enterprise-IT dargestellt.

Einführung

Um qualitativ hochwertige Software und Systeme zu entwickeln und um die Qualität, Zuverlässigkeit und Korrektheit sicherzustellen, bedarf es geeigneter Testmethoden. Mit steigender Komplexität des zu entwickelnden Systems wird auch der Test zunehmend schwieriger und aufwändiger. Um mit möglichst geringem Aufwand zur bestmöglichen Testkonfiguration und lückenlosen Abdeckung des geplanten Testziels zu gelangen, bedarf es einer strukturierten Methodik, die über die reine Erstellung von manuellen oder automatisierten Testfällen auf Basis von Requirements hinausgeht.

Am Anfang eines jeden Entwicklungsprozesses stehen die Anforderungen an das zu entwickelnde System. Bereits in dieser Phase können leicht schwerwiegende Fehler durch Missverständnisse und Interpretationsfehler entstehen, die oft erst im Einsatz festgestellt werden. Fehler dieser Art gehören zu den kostenintensivsten. Es empfiehlt sich daher, Anforderungen in einer intuitiv verständlichen Form, die möglichst wenig Interpretationsspielraum bietet, zu dokumentieren. Die BPMN bietet für diesen Zweck eine grafische Notation, die durch ihren Fokus auf Geschäftsprozesse vor allem im Bereich Enterprise-IT und Workflow-Management eine schlanke und einfache Sprache bietet.

Weiterhin wird durch eine grafische Dokumentation der Anforderungen der modellgetriebene Ansatz zur Entwicklung des Systems sowie zur Testspezifikation wesentlich unterstützt, da sich die jeweiligen Modelle aus den Anforderungsmodellen ableiten lassen.

In diesem Artikel soll aufgezeigt werden, wie ein solches Vorgehen zur Entwicklung einer Testspezifikation im Detail aussehen kann.

BPMN

Die BPMN 2.0 [OMG10] als grafische Notation definiert einen überschaubaren Wortschatz, wodurch sie im Gegenteil zur menschlichen Sprache klar, präzise, einfach zu verstehen und vor allem leicht zu erlernen und anzuwenden ist.

Zu den grundlegenden Elementen der BPMN zählen:

- Aktivitäten: Handlungen oder Arbeitsschritte
- Sequenzflüsse: Pfeile, die den Verlauf eines Prozesses anzeigen
- Ereignisse: Zeitpunkte oder Zeitintervalle verbunden mit Funktionalität
- Gateways: Verzweigungspunkte

sepp.med gmbh

Mit 30 Jahren Erfahrung im gesamten Produkt- und System-Lifecycle von komplexen und sicherheitskritischen Systemen ist das IT-Unternehmen sepp.med gmbh ein verlässlicher und bewährter Partner, unter anderem in den Branchen Medizintechnik, Pharmazie und Automotive & Automation.

Die sepp.med gmbh ist ein Serviceanbieter mit Erfahrung in internationalen Großprojekten und Anlagenprojekten. Der Fokus liegt hierbei auf der Abwicklung kompletter Services, welche sowohl technischer Natur sein können (Requirementsengineering, Entwicklung, Beratung, QS und Test) als auch Querschnittscharakter haben können (Projekt-, Prozess- und Qualitäts-Management).

Eine daraus entwickelte Expertise der sepp.med GmbH ist der sichere Umgang mit regulatorischen Vorgaben in sicherheitskritischen Bereichen und Industrien.

Das sepp.med-Serviceportfolio umfasst unter anderem:

- Softwareentwicklung
- Softwarequalitätssicherung und Test
- Integration komplexer Systeme
- Prozessmanagement und -optimierung
- Durchführung und Management kompletter Test- und Entwicklungsprojekte
- .mzT (modellzentrierter Test) und automatische Testfallgenerierung (.getmore)
- Consulting und Projektbegleitung nach SPICE, CMMI oder ITIL
- Schulungen, z.B. zum ISTQB ® Certified Tester Foundation Level

Die besonderen Expertisen und die Innovationskraft von sepp.med werden in enger Kooperation mit Hochschulen und durch die regelmäßige Fort- und Weiterbildung unserer Mitarbeiter kontinuierlich erweitert.

Als Beispiel unserer erfolgreichen Arbeit in innovativen Bereichen und des produktiven Einsatzes dieser Forschungsarbeiten ist das modellzentrierte Testen (.mzT) zu sehen:

„Als kleines Forschungsvorhaben in Produktivprojekten gestartet ist diese Methodik mittlerweile ein wichtiger Bestandteil unseres Portfolios."

www.seppmed.de sepp.med Qualität sichert Erfolg info@seppmed.de

In der BPMN können Modelle anhand ihres Abstraktionsgrades unterschieden werden. Sogenannte **fachliche Modelle** sind sehr abstrakt formulierte Modelle, die den groben Ablauf eines Prozesses beschreiben und nicht jeden einzelnen Schritt spezifizieren. Diese Modelle sind für den Entwurf und die Abstimmung unter den diversen Stakeholdern, Testern, Anwendern und IT-Entwicklern bestens geeignet. Auch für die Beschreibung von Anforderungen an Workflow-orientierte Softwaresysteme sind fachliche Modelle ideal. Weiterhin existieren sogenannte ausführbare Modelle, die so weit ausspezifiziert sind, dass sie als direkter Input zur Implementierung und Integration von Workflow-Management-Systemen fungieren können.

Innovator for Business Analysts

Der *Innovator for Business Analysts* [MID10] von MID ist ein neues Modellierungswerkzeug, das speziell für die Rolle von Business Analysten, Anforderungsanalysten, Prozessmodellierern und auch IT-Architekten und Systemtestern entworfen wurde. Das Werkzeug implementiert die BPMN in vollem Umfang und bietet gleichzeitig viele weitere Elemente aus der UML, wie Zustandsdiagramme und textbasierte Funktionen, zum Beispiel zur Verwaltung von Anforderungen. Dabei orientiert sich das Werkzeug nicht daran, die OMG Standards in vollem Umfang zu realisieren, sondern dem jeweiligen Anwender die für seine Rolle notwendige Spezifikationsmethodik zur Verfügung zu stellen.

.getmore der Testfallgenerator

Die Firma sepp.med gmbh stellt mit dem Testfallgenerator *.getmore* [SE09-1] ein Programm für den modellzentrierten Test zur Verfügung, das eine Brücke zwischen Modellierungswerkzeug und Testmanagement- bzw. Testautomatisierungswerkzeug schlägt. Die in den Modellierungswerkzeugen spezifizierten .mzT-Modelle können in *.getmore* importiert und ausgewertet werden. Das Programm leitet aus den Modellen unter Einbeziehung von unterschiedlichen Strategien direkt Testfälle ab, die anschließend in Testmanagementwerkzeuge exportiert und dort weiter verarbeitet werden können. Die entstehenden Test Sets sind vollständig und können zur Vermeidung von Testfallexplosion anhand vorselektierter Modellattribute an die jeweils definierten Testziele angepasst werden. Durch die modulare Architektur und die offen gestalteten Schnittstellen lässt sich *.getmore* mit beliebigen Modellierungswerkzeugen und Testmanagementwerkzeugen kombinieren und so leicht in bestehende Werkzeugketten integrieren.

TFS die clevere Alternative

Der *Team Foundation Server* (TFS) [MIC10] von Microsoft bietet eine Umgebung, die Projektarbeiten in Software-Entwicklungsteams von Anfang bis Ende unterstützt. Angefangen bei den Anforderungen an ein System, über die Entwicklung bis zur Verwaltung der Tests bietet der TFS eine zentrale Plattform für alle anfallenden Schritte. Weiterhin bietet er Funktionalitäten zur Versionierung von Quellcode, automatische Builds, Testautomatisierung u.v.m. Dabei wird die Kommunikation zwischen allen Teammitgliedern unterstützt und er dient als Workflow-Management-System innerhalb eines Projekts.

Der modellzentrierte Test

Für die Spezifikation und die Entwicklung geeigneter Tests bedient man sich in der Softwareentwicklung der Methodik des **modellbasierten Testens** (MBT). Ist die Entwicklung von Systemen modellgetrieben, liegt es nahe, Testfälle aus den Entwicklungsmodellen automatisch zu generieren. So können Einsparungen geschaffen werden, da die Tests zusammen mit dem System konzipiert, entwickelt und verändert werden.

Es hat sich jedoch gezeigt, dass Entwicklungsmodelle allein nicht ausreichen, um adäquate Tests zu spezifizieren. Sie beschränken sich meist auf die geforderte Funktionalität und enthalten nicht alle möglichen Szenarien, die für den Test notwendig sind. So könnte lediglich verifiziert

werden, ob die aus dem Modell generierte Software dem Modell entsprechend funktioniert. Problematisch ist dabei, dass Fehler im Entwicklungsmodell auch zu fehlerhaften Testfällen führen.

Aufgrunddessen existieren verschiedene Ansätze, die diese Probleme über die Hinzunahme von testspezifischen Umgebungsmodellen, Usage-Modellen etc. zu vermeiden versuchen. In diesen fehlen jedoch die explizite Testersichtweise auf das System und die Möglichkeit, die Tests durch Testmanagement-Informationen und Teststrategien zu optimieren. Dies ist erforderlich, da eine rein kombinatorische Generierung von Testfällen aus Entwicklungs- oder auch Testmodellen üblicherweise zu einer nicht mehr handhabbaren Testfallexplosion führt.

Ein weiterentwickelter Ansatz, der diesen Problemen Rechnung trägt, ist die Methodik des **modellzentrierten Testens (.mzT)** [SE09-2]. Die hier spezifizierten Testmodelle werden eigens für den Test entwickelt und direkt aus den Anforderungen abgeleitet. Durch die daraus resultierende Unabhängigkeit der Testmodelle zu den Entwicklungsmodellen, führt eine fehlerhafte Systemmodellierung nicht automatisch zu fehlerhaften Tests. Weiterhin gehen Testmanagement-Informationen, das Know-how von Testdesignern (Tester's Mindset) und die Benutzersicht (Verwendung analog zur späteren Anwendung) mit ein. So konzentriert sich .mzT auf die Validierung des zu testenden Systems und auf die Möglichkeit, durch optimierte Testfallauswahl wirtschaftlich vertretbare Tests durchzuführen. Weiterhin werden Testfälle durch die Integration von Testmanagement-Informationen passend zu den jeweiligen Testzielen erzeugt, wodurch der Aufwand für die tatsächliche Ausführung wesentlich reduziert wird.

Liegen Anforderungen in Form eines Anforderungsmodells vor, ist es möglich, dieses durch sukzessives Erweitern zum Testmodell zu überführen. So ist eine durchgängige Methodik und Notation von den Anforderungen bis zum Test gegeben. Diese Methodik wird im Folgenden, mit den vorgestellten Werkzeugen exemplarisch demonstriert.

Modellzentrierter Test mit BPMN

Auf die grundlegenden Elemente reduziert, besteht ein Testmodell aus Testschritten (TestSteps), die mögliche Interaktionspunkte mit dem zu testenden System (SUT – System under Test) darstellen und Prüfschritten (VerificationPoints), für reine Prüfanweisungen. In der BPMN werden diese Elemente als Aktivitäten dargestellt. Über Sequenzflüsse und Gateways wird die Ablauflogik modelliert. Jeder Pfad des resultierenden Graphen beschreibt, abhängig von Testkonfiguration und Testdaten, einen oder mehrere mögliche Testfälle.

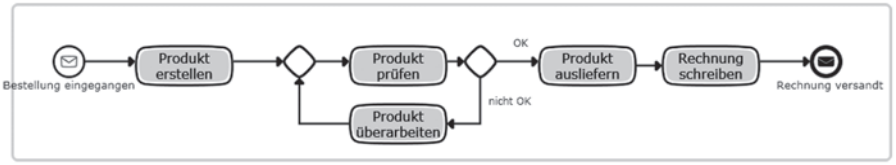


Abbildung 1: Anforderungsmodell

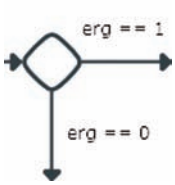
Vom Anforderungsmodell zum Testmodell

Wie kann nun ein Anforderungsmodell in ein solches Testmodell überführt werden? Als Beispiel soll folgendes, im *Innovator for Business Analysts* erstelltes, Anforderungsmodell dienen (siehe Abbildung 1:

In diesem stark vereinfachten und abstrakten Modell soll, nach dem Eingang einer Bestellung, ein Produkt erstellt, geprüft und versendet werden. Am Ende wird eine Rechnung versandt. So lange die Überprüfung fehlschlägt, wird das Produkt überarbeitet.

Die Aktivitäten des Prozesses müssen im Test geprüft werden, deshalb werden diese in Testschritte (im Diagramm: <<TS>>) überführt. Das anfängliche Ereignis definiert, dass der Prozess durch eine Bestellung gestartet wird. Diese muss der ausführende Tester durchführen, weshalb auch hier ein Testschritt notwendig wird. Der Prozess endet mit dem Versand der Rechnung, was der Tester überprüfen muss. Hier muss also ein Prüfschritt (im Diagramm <<VP>>) eingefügt werden. Weitere Prüfschritte können an sinnvollen Stellen eingefügt werden, wie z. B. nach der Überprüfung des Produkts. In den Aktivitäten werden detaillierte Informationen für die Durchführung der jeweiligen Schritte hinterlegt. Dies führt zu folgendem Modell (siehe Abbildung 2):

Der Testschritt „Produkt überarbeiten" ist nur dann sinnvoll, wenn die Prüfung fehlschlägt. Diese Abhängigkeit muss mit Hilfe einer maschinenlesbaren Syntax – im Falle von .getmore Python – am entsprechenden Gateway beschrieben werden. (**siehe nebenstehenden Detail-Ausschnitt**)



Danach müssen alle Testschritte, die Einfluss auf das Ergebnis der Prüfung haben, so aufgeteilt werden, dass die entsprechenden Äquivalenzklassen erzeugt werden. In „Produkt erstellen" muss also ein fehlerhaftes und ein korrektes Produkt erstellt werden (analog „Produkt überarbeiten"). Um die Formatierung des Diagramms zu erhalten, kann dies in einklappbaren Prozessen modelliert werden. Dabei werden in den Aktivitäten die Variablen, die im Gateway geprüft werden, in Skriptsprache definiert. Beispielsweise würde in „Produkt korrekt überarbeiten" die Anweisung „erg = 1" definiert. Dies führt zu folgendem Testmodell (siehe Abbildung 3):

Weiterhin können über Labels beispielsweise Prioritäten für die Sequenzflüsse vergeben werden, um die Testfälle nach Wichtigkeit zu ordnen. Denkbar wäre hier, den zu „Produkt fehlerhaft überarbeiten" führenden Sequenzfluss mit einer niedrigen Priorität zu versehen. So würde dieser, nur bei entsprechender Angabe im Testfallgenerator, mit aufgenommen werden. Das resultierende Diagramm kann nun, über ein Add-In direkt in den Testfallgenerator *.getmore* exportiert werden.

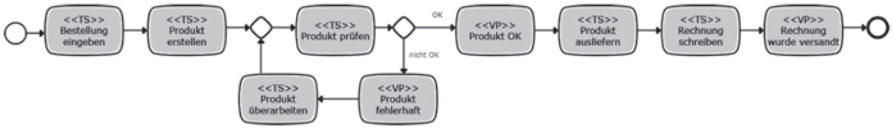


Abbildung 2: Testmodell Zwischenschritt

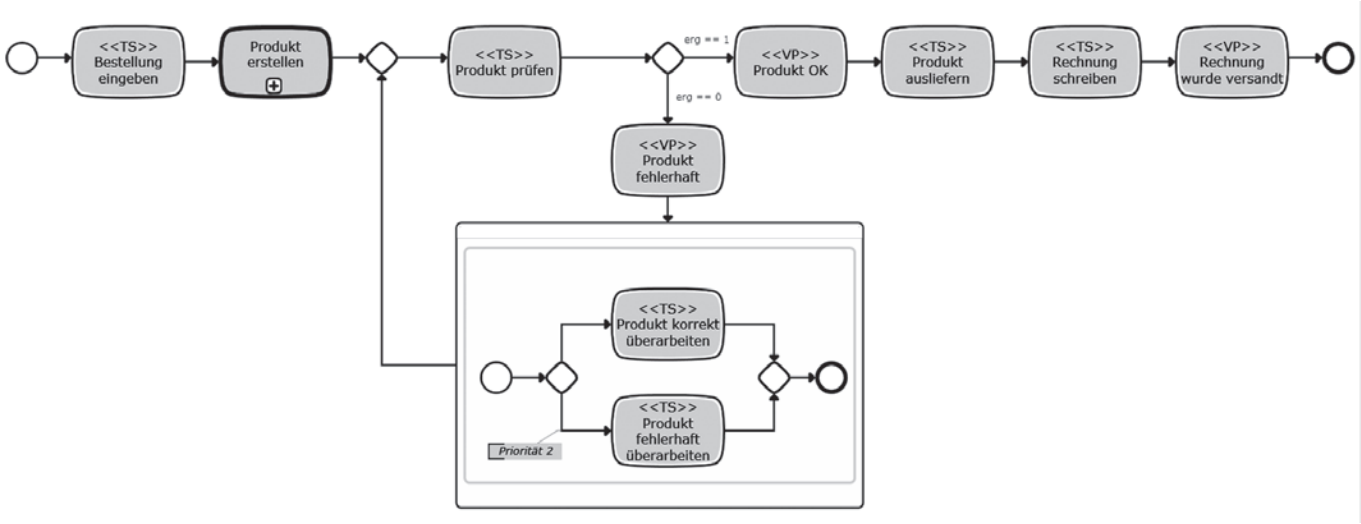


Abbildung 3: Fertiges Testmodell

Testfallgenerierung mit .getmore

Nach dem Export kann mit *.getmore* unter Verwendung konfigurierbarer Strategien ein Testfallbaum (TestCaseTree) erzeugt werden. Dieser enthält alle Testfälle, die unter Berücksichtigung der jeweiligen Generierungsstrategie und der zugehörigen Auswahlparameter, z. B. Schleifenbegrenzer, Prioritäten, etc., möglich sind. Eine mögliche Strategie ist u. a.: „Full Path Coverage“. Diese extrahiert ohne weitere Parameter theoretisch alle möglichen Pfade des Graphen. Um in endlicher Zeit zu einem Ergebnis zu kommen, können und müssen – abhängig von Modell und Strategie – spezifische Parameter gesetzt werden. Im Beispiel wurde „Full Path Coverage“ gewählt. Die maximale Pfadlänge wurde auf 50, die Schleifenbegrenzung auf 3 und die Priorität auf "hoch und mittel" gesetzt.

.getmore generiert mit den gegebenen Einstellungen vier Testfälle und erzeugt den Testfallbaum. Anschließend ist es möglich, diesen durch verschiedene Filter weiter zu optimieren. Mögliche Filter sind z. B. Knotenabdeckung (alle Knoten des Testfallbaums, mindestens einmal) oder Kantenabdeckung (analog für Sequenzflüsse). In Abbildung 4 ist der erzeugte Testfallbaum zu sehen. Dieser ist nach Knotenabdeckung gefiltert, wodurch nur die übrigen Testfälle selektiert sind.

Testmanagement mit dem Team Foundation Server

Die von *.getmore* extrahierten Testfälle können nun in die Test-Management-Komponenten des *Team Foundation Servers* von Microsoft importiert und dort verwaltet werden.

In Abbildung 5 ist die Detailansicht eines der beiden Testfälle zu sehen. Hier können die Testfälle ggf. angepasst oder mit zusätzlichen Informationen angereichert werden. Im Bereich „Aktion“ sind die Detailbeschreibungen der einzelnen Testschritte angezeigt, die dann Schritt für Schritt abgearbeitet werden müssen.

Schließlich können die Testfälle werkzeuggestützt z. B. manuell ausgeführt werden. Startet man einen Testfall, gelangt man zur Ansicht in Abbildung 6:

Hier können für jeden Testschritt eines Testfalls die Ergebnisse "passed" bzw. "failed" dokumentiert werden. Im Fehlerfall können Detailbeschreibungen protokolliert werden. Während und nach der Ausführung können zahlreiche statistische Informationen und Daten abgerufen werden.

Zusammenfassung

Für die Spezifikation möglichst optimaler Tests in der System- und Softwareentwicklung bedarf es einer geeigneten Methodik. Ziel ist es, unter Berück-

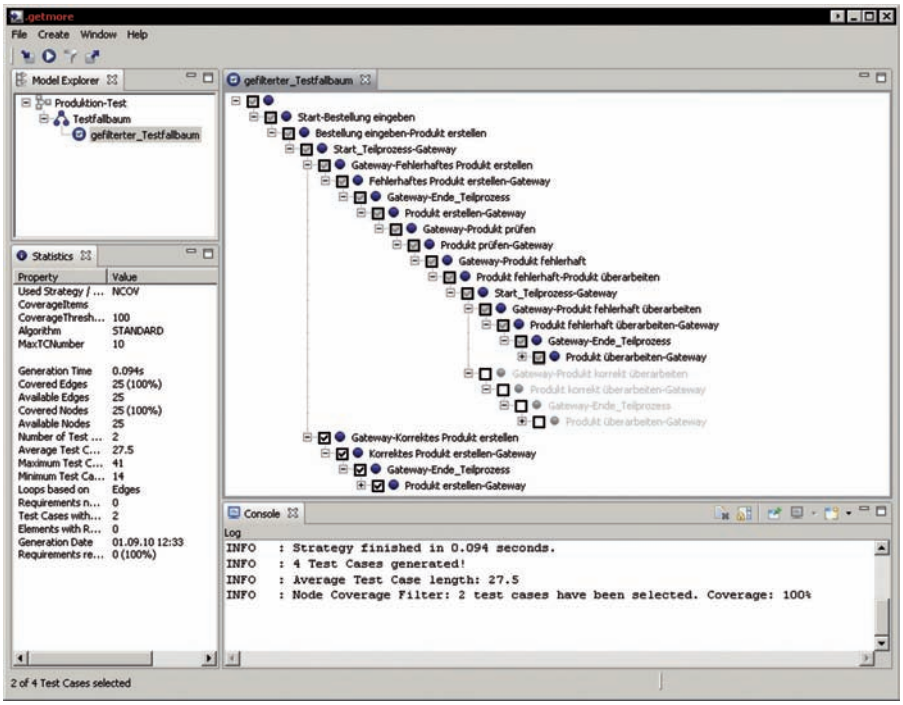


Abbildung 4: Testfallgenerierung mit .getmore

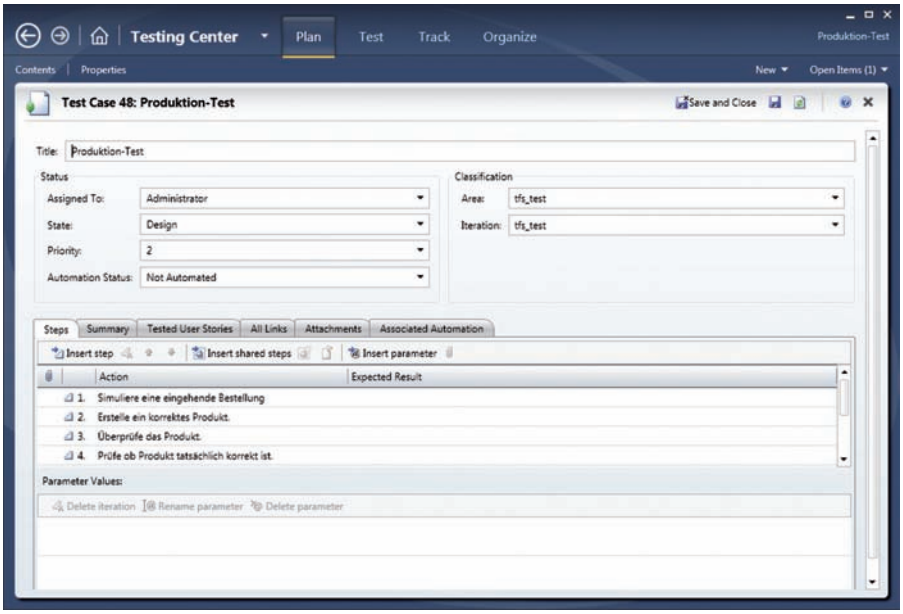


Abbildung 5: Detailansicht Testfall

sichtigung spezifischer Testziele, aufwandsarm möglichst gute Tests zu spezifizieren und den Aufwand der Testausführung so gering wie möglich zu halten.

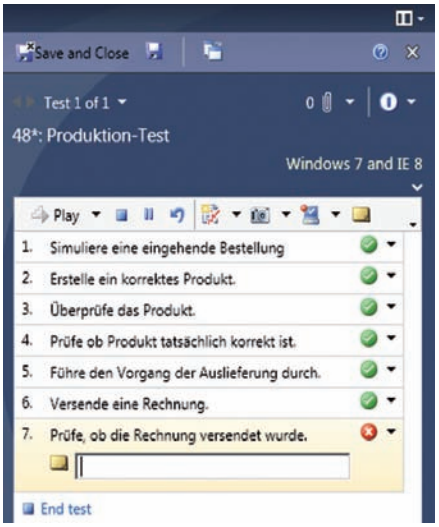


Abbildung 6: Ausführung mit TFS

Das modellzentrierte Testen, als Best-Practice-Methodik des modellbasierten Testens, eignet sich hervorragend zur Erreichung dieses Ziels (siehe auch [PS09], [BM08], [BM09]). Vorteilhaft ist es außerdem, wenn eine durchgängige und einheitliche Spezifikationssprache innerhalb des Systementwicklungs- und Testprozesses verwendet wird. Hierfür eignet sich die BPMN in der neuen Version 2.0. Als grafische Notation mit dem Fokus auf Geschäftsprozesse ist sie gut für die Dokumentation von Anforderungen an Systeme aus der Enterprise-IT geeignet. Durch ihren strengen Fokus stellt sie eine schlanke, leicht zu erlernende und intuitiv verständliche Sprache dar.

Das Modellierungswerkzeug *Innovator for Business Analysts* von MID, der Testfallgenerator *.getmore* von sepp.med und der *Team Foundation Server* von Microsoft als Test-Management-Werkzeug, bieten in Kombination eine gut geeignete Werkzeugkette, mit der die .mzT-Methodik unter Verwendung der BPMN angewandt werden kann. So können optimierte Testszenarien auf einfachem, durch Automatismen fehlerarmem und ressourcenschonendem Weg erreicht werden. ■

Referenzen

[OMG10] – OMG: „Business Process Model and Notation", dtc/2010-06-05 – http://www.omg.org/spec/BPMN/2.0 (letzter Aufruf: Sep 2010)

[MID10] – MID the modeling company: „Innovator for Business Analysts", http://www.mid.de/fileadmin/mid/PDF/Datasheets/Datasheet_Innovator_for_Business_Analysts.pdf (letzter Aufruf: Sep 2010)

[SEO9-1] – sepp.med GmbH: „.getmore (GEnerating Tests from MOdels to Reduce Effort)) – Der Testfallgenerator", http://www.seppmed.de/fileadmin/pdfs/metamethoden/Broschuere_getmore.pdf (letzter Aufruf: Sep 2010)

[MIC10] – Microsoft: „VISUAL STUDIO TEAM FOUNDATION SERVER 2010", http://www.microsoft.com/germany/visualstudio/products/team/visual-studio-team-foundation-server.aspx (letzter Aufruf: Sep 2010)

[SEO9-2] – sepp.med GmbH: „.mzT – modellzentriertes Testen – Mehr Effizienz bei Design & Spezifikation im Test", http://www.seppmed.de/fileadmin/pdfs/metamethoden/Broschuere_mzT.pdf (letzter Aufruf: Sep 2010)