Erfahrungen mit modellzentriertem Testen in der Validierung komplexer, sicherheitskritischer Systeme

Dr. Armin Metzger
sepp.med gmbh
armin.metzger@seppmed.de

25. September 2009

1 Abstract

Klassische dokumentenbasierte Ansätze zur Systemvalidierung zeigen sich immer weniger geeignet den heutigen technischen und ökonomischen Anforderungen in der medizinischen IT Domäne gerecht zu werden. IT Systeme werden kontinuierlich komplexer, integrierter und vernetzter. Effizienz, Qualität und Transparenz sinken bei den traditionellen Methoden in aktuellen Szenarien und die Automatisierung im Test schlägt in den meisten Fällen fehl.

In dieser Ausarbeitung wird die Notwendigkeit einer Paradigmenänderung in der Qualitätssicherung gezeigt und eine Lösung vorgestellt: modellzentriertes Testen.

2 Einführung

IT Hersteller müssen im Bereich Softwareentwicklung hohen Anforderungen genügen, um auf dem Markt konkurrenzfähig zu sein und, z.B. in Domänen wie der Medizintechnik, die strengen Regularien einzuhalten. Heutzutage dominieren dokumentenbasierte Methoden den Testprozess. Diese können aber nicht mit den steigenden Anforderungen an die Wartbarkeit, Wiederverwendbarkeit und niedrige Kosten im Test in Einklang gebracht werden.

Der modellzentrierte Test (.mzT) ist eine bahnbrechende Methode um den Einschränkungen dokumentenbasierter Tests zu begegnen. Dabei wird das Testerverhalten visuell als Testmodell beschrieben. Die Modellierung ist dabei auf die Anwendung in der Qualitätssicherung hin optimiert um bessere Testszenarien, basierend auf den Geschäftsprozessen, zu erhalten. UML spezifisch für Tester und Testprozess: .mzT [sep07].

3 modellzentrierter Test

Im klassischen Test Design verwendet der Designer seine Ideen und Erfahrungen um das System im Hinblick auf ein bestimmtes Requirement in einem beliebigen kurzen Workflow zu validieren.

Ein solches Vorgehen ist jedoch äußerst unstrukturiert und weit entfernt von der Abdeckung aller existierender Workflows. Somit können diese Testfälle als eine willkürliche Auswahl angesehen werden, aus denen keine Rückschlüsse über die bestehende Abdeckung der User Workflows -und damit der Testqualität- gezogen werden können. Die Konsequenz ist, dass eine Vielzahl an möglichen und relevanten Workflows ungetestet bleibt. Dies bedeutet auch, dass die so gewonnenen Abdeckungsmaße nicht aussagekräftig sind.

Eine weitaus strukturiertere Herangehensweise bietet das vor einigen Jahren eingeführte modellbasierte/modellgetriebene Testen. Nachteilig an diesem Verfahren ist allerdings, dass die Tests hierbei von Entwicklungsmodellen abgeleitet werden, welche typischerweise auf dem Systemverhalten basieren und nicht auf dem testrelevanten Nutzungsverhalten des Systems. Diese Entwicklungsmodelle sind in den allermeisten Fällen weder vollständig noch an "real life" Workflows orientiert.

Im Gegensatz zu modellbasierten Techniken stützt sich das modellzentrierte Vorgehen nicht allein auf Entwicklungsmodelle, sondern lässt darüber hinaus noch Requirements, Use Cases, bekannte Workflows und den Tester's Mindset für Schlecht- oder Ausnahmeszenarien einfließen (siehe Abb. 1). Dadurch kann eine volle Abdeckung mit gleichzeitiger Unabhängigkeit vom Systemmodell erreicht werden um eine echte Validierung des Systems zu gewährleisten [sep07]. Anforderungen aus dem Enwicklungs- und Testprozess hinsichtlich Testqualität, Abdeckungen oder Zeit-/Kostenaufwand müssen dabei berücksichtigt werden. Durch die Integration von Testmanagement Informationen im Testmodell sind alle Testrelevanten Informationen an einer zentralen Stelle übersichtlich abgelegt und können schon bei der Generierung der Testfälle aus dem Modell berücksichtigt werden.

Abstraktionsebenen in den Modellen erlauben dabei sowohl abstrakte, Anwender-Workflow nahe Sichten auf den Testablauf als auch eine beliebige Detaillierung. So lassen sich einerseits übersichtliche, allgemein verständliche Testmodelle erstellen, die aber ausreichend detailliert und formal sind, um dar-

aus automatisiert manuelle Testfälle und Testautomatisierungscode abzuleiten. Damit werden auch die hohen Automatisierungsgrade modellgetriebener Ansätze erreicht ohne das Ziel der Systemvalidierung (im Gegensatz zur Verifikation) zu vernachlässigen.

4 Testmodell

Testmodelle im modellzentrierten Test bestehen aus hierarchisch aufgebauten Diagrammen. Dabei repräsentieren die High-Level Diagramme die abstrakte Sicht auf die Testworkflows, während die unteren Diagrammschichten den detaillierten Testablauf abbilden um daraus automatisiert manuelle Testfälle oder Testautomatisierungsskripten zu generieren. Während die Struktur des Testmodells die möglichen Workflows bestimmt, können Testprozeß- und Testmanagementinformationen (z.B. Prioritäten) an Modelle und Modellelemente annotiert werden. Für das Testdesign werden Standard UML Modellierungstools verwendet [omg08]. Die Schnittstelle zwischen den Testmodellen und dem Testausführungswerkzeug wird vom Testfallgenerierungswerkzeug .getmore bereitgestellt.

.getmore ermöglicht es, UML-basierte Modelle aus einer Vielzahl an Modellierungswerkzeugen einzulesen. Durch die Verwendung von unterschiedlichsten Algorithmen können daraus Testfälle erzeugt und in das Testmanagement- oder Testausführungswerkzeug exportiert werden [sep08].

5 Testmanagement- und Prozessschnittstellen

Die Testmodelle im modellzentrierten Testvorgehen beinhalten die Informationen für die Integration in den Entwicklungsprozess. So können z.B. Systemrequirements im Testmodell referenziert werden, was ein konsistentes Tracing vom Requirementmanagement bis zum implementierten Testfall und von den Testergebnissen wieder zurück zum Requirementtool ermöglicht.

Testmanagementinformationen im Testmodell führen durch gezielte Reduktion der generierten Testfälle mittels Selektion, Priorisierung und Optimierung zu einer weiteren Effizienzsteigerung. Das Testmanagement steuert die Auswahl der Testfälle und deren Ausführung, deshalb werden bei der Generierung alle relevanten Informationen aus dem Modell (z.B. Prioritäten, Hazards, funktionale Cluster, etc.) auf einzelne Testfälle abgebildet.

Die modellierten Workflows können auch als Basis für Impact- oder Risikoanalysen verwendet werden. Das Testmodell gibt der verantwortlichen Person dabei den nötigen Überblick über die betroffenen Workflows.

6 Testmodelle zur Zertifizierung

In der Entwicklungsphase des Software Lifecycles werden immer stärker Test-First Ansätze verfolgt. Die Implementierungen werden dabei zwar noch anhand des Systemdesigns erstellt, die Bestätigung, das die Implementierung den Vorgaben entspricht findet nicht mehr über Reviews oder der Entwicklung nachgeschaltete Tests statt, sondern entwicklungsbegleitend mit im Vorfeld anhand des SW-Designs erstellten Testfällen.

Analog läßt sich auch auf Systemebene vorgehen. Werden die Systemanforderungen (Verhalten, Schnittstellen, Daten und Datenformate) als Testmodelle im Sinne des .mzT modelliert, so kann über einen erfolgreichen Test die korrekte Funktionalität des sich in der Implementierung befindlichen Systems nachgewiesen werden. Über ein solches Vorgehen können also Testmodelle der funktionalen Anforderungen, Schnittstellen und erwartete Workflows als Basis für die Implementierung dienen und im Test zur Generierung der Validierungstestfälle dienen.

Sind Standards die Grundlage für eine Implementierung (z.B. bei Kommunikationsstandards, Formatstandards) lassen sich auch diese meist textuellen Beschreibungen als Testmodell abbilden. Diese Testmodelle lassen sich dann für die Zertifizierung einer Implementierung, d.h. dem Nachweis der Standardkonformität verwenden. Ein solches Vorgehen, den Nachweis der korrekten Implementierung über den erfolgreichen Test mit einer Standard-Testsuite zu erbringen, ist in verschiedenen Domänen bereits Realität. Das im nächsten Abschnitt vorgestellte TestNGMed Testbett bringt diesen Ansatz in die Domäne Medizintechnik.

7 Erfahrungen in Industrieprojekten

Das modellzentrierte Vorgehen .mzT wird erfolgreich in verschiedenen Projekten eingesetzt. Verfolgt werden dabei unterschiedlichste Testziele in einer großen Bandbreite an Testphasen. Vier Beispiele sollen hier kurz herausgestellt werden.

Zunächst das **TestNGMed Testbett**, das zur Validierung von Systemen erstellt wurde, die auf Medizintechnik-Standards (IHE/HL7/DICOM) aufbauen [ihe08, hl708]. Im Testdesign werden UML Modelle verwendet, aus denen mittels des Testfallgenerators .getmore TTCN-3 Automatisierungsskripten generiert werden. Anwendungsgebiete für das Testbett sind die Validierung von auf oben genannten Standards basierenden Systemen (Bildarchive, intensivmedizinische Diagnosegeräte) und die Unterstützung der Integration solcher Applikationen in bestehende Klinikumgebungen. Die Vision für das TestNGMed Test-

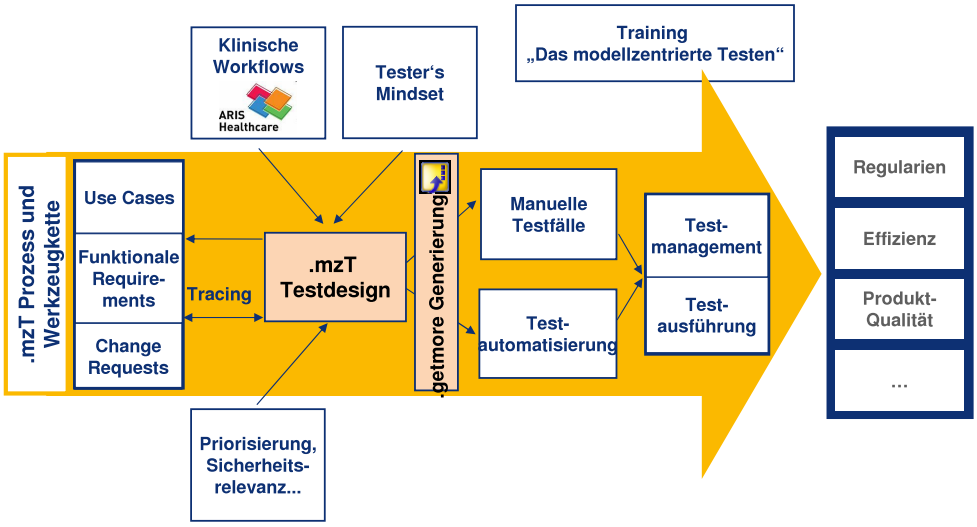

Abbildung 1. Darstellung eines .mzT Testprozesses. Aus den im Zentrum stehenden Testmodellen und den dort abgelegten Testmanagement Informationen werden mittels .getmore manuelle/automatisierte Testfälle abgeleitet.

bett ist, als Quasi-Standard für die Zertifizierung von Medizintechnikprodukten hinsichtlich der verwendeten Standards zu dienen.

Eine weitere Implementierung von .mzT ist der Test der **Verbindungsaufnahme-Funktionalität eines RFID Moduls**. Hier zeigten sich sehr früh die Vorteile des systematischen und visuellen Testdesigns mit Modellen. Die Testabdeckung konnte optimiert werden und die Modelle vermittelten allen Beteiligten ein besseres Systemverständnis. Grundlage für die Testmodellierung waren hier Entwicklungsmodelle und textuelle Beschreibungen des Systems und der Anforderungen. Im Testmodell wurden diese verteilten, nicht verknüpften Informationen abgebildet und so ein ganzheitliches Bild des Systems und der vorhandenen Workflows vermittelt.

Effiziente Anpassbarkeit und Wartbarkeit der Testmodelle ist ein wesentlicher Vorteil der Testmodellierung im Vergleich zum Dokumenten-basiertem Vorgehen. Bei der Validierung eines **hochkonfigurierbaren Warenwirtschaftssystems** konnte gezeigt werden, dass die Kombination aus der Übersichtlichkeit der visuellen Testmodelle und der leichten Änderbarkeit eine große Flexibilität im Test bietet und auf Anforderungsänderungen schnell und effektiv reagiert werden kann, um so in ihrer Funktionalität sehr dynamische Systeme zu testen.

Als letztes Beispiel sei hier noch ein **Workflow-orientierter Feature Test** bei medizinischen Diagnosesystemen genannt. Durch die Verwendung von Testmanagement Informationen innerhalb des Modells und intelligenten Algorithmen zur Testfallgenerierung wurden einerseits eine systematische Überdeckung erreicht und zugleich auch erfahrungsbasierte Testszenarien aus dem Modell abgeleitet. Eine zentrale Aufgabe in diesem Projekt war die intelligente, auf die Testziele ausgerichtete Einschränkung des Testumfangs. Die sytematische Abdeckung der relevanten Inhalte in den abgeleiteten Testfällen wurde durch die Wahl eines geeigneten Generierungsalgorithmus erreicht, die Auswahl spezieller Szenarien erfolgte durch Testdesigner, erfahrene Tester und Domänenexperten direkt im Modell.

8 Zusammenfassung

Kosteneffiziente Validierung von IT Systemen unter Berücksichtigung von Kunden- und Marktanforderungen sowie der regulatorischen Vorgaben ist essenziell in den heterogenen Infrastrukturen aktueller IT Systeme. Interoperabilität und Standardkonformität sicherzustellen ist dabei von höchster Kritikalität.

Hersteller müssen diese Integrationsaspekte mit Methoden angehen, die auch die Anwender mit einbinden, um so Interoperabilität bereits in frühen Phasen der Entwicklung zu erreichen. So wird eine signifikante Ersparnis in späteren Integrations- und Wartungsphasen erreicht.

.mzT nutzt visuelles Testdesign basierend auf konkreten Anwenderworkflows und dem Tester's Mindset. Testmanagement Informationen können dabei direkt im Modell hinterlegt werden. Unterstützt durch eine automatisierte Generierung von Testfällen ermöglicht .mzT einen Validierungs- und Integrationsprozess, der alle Stakeholder integriert. Durch die Möglichkeit, Prozessschritte zu automatisieren ist .mzT ein wichtiger Schritt dahin, die Anforderungen an Entwicklungsprozesse in der Medizintechnik zu erfüllen.

Literatur

[hl708] HL7: HL7 Standard. http://www.hl7.org/, 2008.

[ihe08] IHE International: Integrating the Healthcare Enterprise. http://www.ihe.net/, 2008.

[omg08] OMG: Unified Modeling Language. http://www.uml.org/, 2008.

[sep07] sepp.med: .modellzentrierter Test. http://www.seppmed.de/modellzentrierterTest.187.0.html, 2007.

[sep08] sepp.med: .getmore: UML based Test Generator. http://www.seppmed.de/getmore.65.0.html, 2008.

[tes09] Test Next Generation Medical Systems (TestNGMed): EUREKA-Project Σ! 4053 RETEMES, 2007-2009.