Die Integration des Modellbasierten Systemtests in eine bestehende Prozess- und Werkzeugkette

David Kreische

david.kreische@imbus.de

Abstract: In diesem Artikel wird ausgeführt, wie der modellbasierte Systemtest aussehen sollte. Dabei wird auf die Integration in die bestehenden Prozesse und mit den vorhandenen Werkzeugen eingegangen: der modellbasierte Systemtest ergänzt den konventionellen Systemtest und ersetzt ihn nicht. Dabei liegt der Fokus auf einer Modellierung, die von Fachtestern ohne tiefere Modellierungskenntnisse geleistet werden kann.

1 Einleitung

Der zunehmende Erfolg der modellbasierten Softwareentwicklung (MDA/MDD) führt schnell zu der Überlegung, ob die dabei gewonnen Erfahrungen sowie die eingesetzten Werkzeuge nicht auch für den modellbasierten Test (MBT) verwendet werden können. Dabei ist insbesondere die Frage von Interesse, wie sich der Systemtest durch Modellierung unterstützen lässt. Bei der Spezifikation von Testfällen für den Systemtest kommt es zunächst darauf an, die korrekte Umsetzung der gestellten Anforderungen zu überprüfen. Dabei ist Hintergrundwissen über die Anwendungsdomäne von erheblichem Vorteil, da dann auch die impliziten Anforderungen und Annahmen in den Test mit einfließen können. Fachtester mit diesen Kenntnissen haben oft keinen Informatik-Hintergrund, sondern kommen aus den unterschiedlichsten Bereichen und haben sich im Verlauf ihres Berufslebens zu Testern qualifiziert. Sie verfügen nicht über Kenntnisse in der Software-Modellierung. Diesem Personenkreis soll eine einfache und verständliche Möglichkeit gegeben werden, Testabläufe und Testdaten übersichtlich in einem Modell zu beschreiben und daraus automatisch Testfälle zu generieren.

Allerdings kann das Modell nicht alle Aspekte des Systemtests abdecken, ohne sehr komplex zu werden, insbesondere wenn man sich vor Augen hält, dass im Systemtest nicht nur Funktionalität überprüft wird, sondern dass auch nichtfunktionale Aspekte wie Performanz und Useability zu beachten sind. Weiterhin gibt es immer wieder Testfälle, die aufgrund von Erfahrung und Intuition erstellt werden und die methodisch abgeleiteten Testfälle sinnvoll ergänzen. Aus diesen Gründen kann der Test nicht ausschließlich modellbasiert durchgeführt werden, sondern muss gemeinsam mit konventionell entworfenen Testfällen einsetzbar sein. Dazu muss der modellbasierte Test in die vorhandene Prozess- und Werkzeugkette integriert werden.

2 Prozesse und Werkzeuge

In Abb. 1 ist der an den Lehrplan des International Software Testing Qualifications Board (ISTQB) [Ger] angelehnte Testprozess mit seinen einzelnen Phasen, den dabei eingesetzten Werkzeugen und einigen erstellten Artefakten dargestellt.

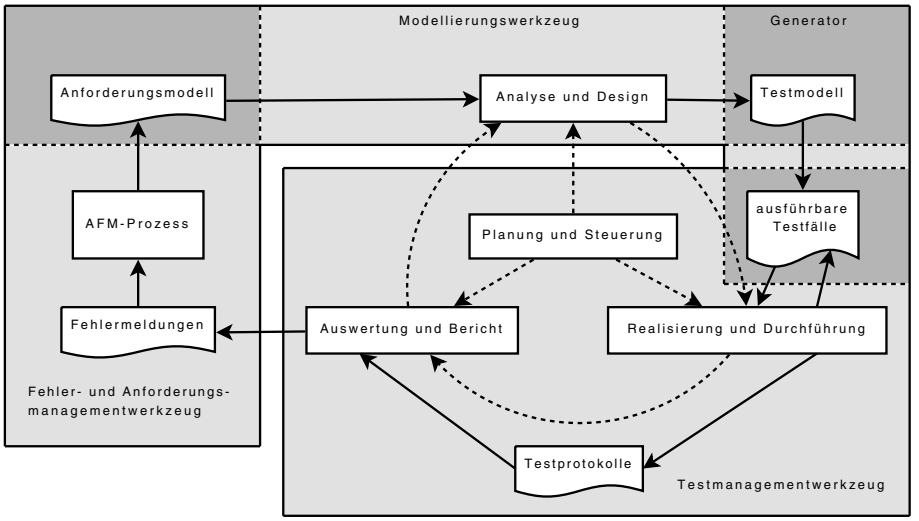

Abbildung 1: Testprozess angelehnt an den ISTQB-Lehrplan

In der Phase „Analyse und Design“ erfolgt die Analyse der Testbasis, insbesondere der Anforderungen auf ihre Testbarkeit und der Entwurf logischer Testfälle. Wenn MBT zum Einsatz kommt, werden in dieser Phase die Testmodelle erstellt. Daran schließt sich die „Realisierung und Durchführung“ an, bei der die Testfälle soweit konkretisiert werden, dass sie ausführbar sind. Dies beinhaltet insbesondere die Angabe von Sollwerten.

Nach der Durchführung der Testfälle folgt die Phase „Auswertung und Bericht“, in der die in den Testprotokollen festgehaltenen Ergebnisse ausgewertet und zusammengefasst werden. Für die gefundenen Fehler werden Fehlermeldungen erstellt und an das Fehler- und Anforderungsmanagement weitergeleitet.

Begleitend zu den technischen Phasen findet „Planung und Steuerung“ statt. Dabei werden Zeitpläne, Prioritäten, Verantwortliche, Meilensteine, Testendekriterien, usw. festgelegt und die korrekte Umsetzung laufend überwacht.

Um den vorgestellten Prozess effizient zu unterstützen, kommen Werkzeuge zum Einsatz. Dabei werden in verschieden Phasen unterschiedliche Werkzeuge verwendet, die miteinander interagieren müssen. Im Modellierungswerkzeug werden die Testmodelle erstellt und bearbeitet. Dieses Testmodell wird vom Generator in ausführbare Testfälle umgewandelt, die im Testmanagementtool weiter verfeinert werden. Dabei werden beispielsweise Sollwerte eingefügt, Prioritäten für die Testdurchführungen angegeben und Testautomati-

sierungsskripte angebunden. Die gefundenen Fehler werden im Fehler- und Anforderungsmanagementwerkzeug bewertet und priorisiert, so dass die Fehlerbehebungsmaßnahmen gezielt in die Anforderungen für die neuen Produktversionen einfließen.

Anforderungen können auch in Form von Modellen erstellt werden. Dabei kommt häufig die Unified Modeling Language (UML) zum Einsatz. Wenn sich ein konkretes UML-Werkzeug bereits in der Software-Entwicklung bewährt hat, dann ergibt sich für den Test oftmals die Forderung, für die Modellierung der Testfälle ebenfalls dieses Werkzeug zu verwenden. Bei dieser Entscheidung spielen Lizenzierungsmodelle und die mit dem Werkzeugeinsatz gemachten Erfahrungen ebenfalls eine Rolle. Das Anforderungsmodelle und Testmodelle stark zusammenhängen und sich aufeinander beziehen, ist ein weiteres Argument für den Einsatz eines gemeinsamen Modellierungswerkzeuges, weil sich diese Bezüge in einem gemeinsamen Werkzeug wesentlich leichter anlegen und verwalten lassen.

Wenn die Anforderungen bereits als Modelle vorliegen, ist die Vorstellung verlockend, aus diesen Modellen sowohl Quellcode als auch Testfälle dafür automatisch zu generieren. Allerdings würde ein solches Vorgehen keine Fehler im Modell selbst finden. Daher muss ein eigenes Testmodell erstellt werden, wobei die Anforderungs-Modelle durchaus als Ausgangsbasis und Ideengeber dienen können.

3 Modellierung in UML

Die UML stellt dem Modellierer eine Vielzahl an Möglichkeiten der Modellierung zur Verfügung. Für den modellbasierten Test müssen diese eingeschränkt werden. Dafür gibt es zwei Gründe. Erstens ist die in der Einleitung erwähnte Zielgruppe der Fachtester mit der kompletten UML überfordert, sondern benötigt Modellierungskonstrukte, die ihre Sichtweise auf den Systemtest unterstützen. Zweitens müssen die Freiheitsgrade, die die komplette UML bietet, soweit eingeschränkt und das Modell so präzisiert werden, dass die Generierung von sinnvollen, ausführbaren Testfällen möglich ist. Die UML bietet die Möglichkeit, solche Einschränkungen und Präzisierungen vorzunehmen. Ein einfacher Weg dazu ist die Verwendung eines Profils. In [BDKK08] ist ein solches Profil beschrieben, mit dem der Fachtester ihm bekannte und eingängige Konzepte wie Test- und Prüfschritte, Äquivalenzklassen und Repräsentanten anlegen und bearbeiten kann.

Er verwendet dazu die Aktivitäts-, Klassen und Objektdiagramme der UML. Die Testabläufe werden mittels Aktivitätsdiagrammen beschrieben. Einzelne Testschritte werden durch Call-Operation-Actions modelliert. Die aufgerufenen Operationen werden gemeinsam mit den Datentypen im Klassenmodell definiert. Beim Aufruf der Operationen hat der Tester die Möglichkeit, Testdatenvariationen anzugeben. Um die Testdatentypen mittels der Äquivalenzklassenmethode zu strukturieren, bietet sich das Klassendiagramm an. Dort können Datentypen angelegt und bis zu finalen Äquivalenzklassen verfeinert werden. Ebenso ist die Bildung zusammengesetzter Datentypen möglich. Im Objektdiagram wird schließlich festgelegt, welche Repräsentanten für die Testläufe konkret verfügbar sind.

4 Generierung der Testfälle und Nachbearbeitung im Testmanagementwerkzeug

Die Grundidee bei der Testfallgenerierung besteht darin, jeden möglichen Pfad vom Start- zum Endknoten zu identifizieren und als Testfall zu im Testmanagementwerkzeug abzulegen. Da die Menge der Testfälle sehr groß werden kann, sollen später Generator-Varianten zum Einsatz kommen, bei denen der Tester die Generierung unerwünschter Testfälle auf einfache Weise unterbinden kann. Solche Generator-Varianten lassen sich leicht erzeugen, weil die technische Umsetzung auf dem MDA-Framework openArchitectureWare [E$^+$] beruht. Damit können auf einfache Weise Transformationen zwischen Modellen durchgeführt werden, die auf verschiedenen Meta-Modellen basieren.

Vor der Durchführung der generierten Testfälle müssen im Testmanagementwerkzeug noch fehlende Datenwerte ergänzt werden. Dazu gehören typischerweise die Sollwerte, da es im Modell keine Möglichkeit gibt, diese abhängig von den Eingabewerten berechnen zu lassen. Es können aber auch noch andere Ergänzungen und Erweiterungen an den Testfällen vorgenommen werden. Zur Automatisierung der generierten Testfälle ist es lediglich nötig, die Operationen separat mit Automatisierungsskripten auszustatten. Nach der Testdurchführung werden die Testergebnisse gemeinsam mit den Testfällen versioniert, damit sie stets nachvollziehbar bleiben.

5 Zusammenfassung und Ausblick

Es wurde dargelegt, wie der modellbasierte Systemtest in eine bestehende Werkzeug- und Prozesslandschaft integriert werden kann. Insbesondere ist es durch die Verwendung eines UML-Profils leicht möglich, dieses Vorgehen in verschiedenen UML-Werkzeugen umzusetzen. Schwieriger ist der Austausch des Testmanagementwerkzeuges, da es auf dieser Seite keinen Standard gibt.

Bisher ist es nur möglich aus dem Testmodell Testfälle in das Testmanangementwerkzeug zu generieren. Es ist aber wünschenswert, gewisse Elemente aus dem Testmanagementwerkzeug ins Testmodell zu übernehmen: Testdatentypen, Äquivalenzklassen, Repräsentanten und Operationen, die schon bei anderen Tests eingesetzt wurden, sollen auch dem Modellierer im modellbasierten Test zur Verfügung stehen.

Literatur

[BDKK08] Christian Brandes, Andreas Ditze, Christian Kollee und David Kreische. Modellbasiertes Testen praxisgerecht realisiert: Vom UML2-Profil zum Testmanagement-Werkzeug. *OBJEKTspektrum*, 3:80–86, May 2008.

[E$^+$] Sven Efftinge et al. openArchitectureWare User Guide. Version 4.3.

[Ger] German Testing Board. Certified Tester Foundation Level Syllabus, http://german-testing-board.info/downloads/pdf/lehrplan/.