Modellbasiertes Testen auf Basis des fundamentalen Testprozesses

Tobias Eckardt, Michael Spijkerman
Software Quality Lab (s-lab)
Universität Paderborn
Warburger Str. 100, D-33098 Paderborn, Deutschland
[tobie|spijk]@upb.de

ZUSAMMENFASSUNG

Modellbasiertes Testen (MBT) spielt eine immer größere Rolle, um den Testentwurf während eines Testprozesses zu systematisieren. Vorgehensmodelle im Hinblick auf Modellbildung, Testfallspezifikation, Generierung der Testfälle und Testdurchführung sind vorhanden (vgl. [2], [4], [5], [7]). Der fundamentale Testprozess (FTP) von Spillner und Linz [6] bietet eine gesamtheitliche Sicht, von der Planung bis zum Abschluss der Testaktivitäten. Er nimmt jedoch nicht explizit Bezug auf MBT. Es ist nicht klar welche Aspekte zu berücksichtigen sind, wenn modellbasiertes Testen in Kombination mit dem FTP angewendet werden soll. In diesem Artikel wird ein erster Schritt hin zu einem modellbasierten Testprozess vorgestellt, der sich an dem FTP orientiert.

1. EINFÜHRUNG

Das Testen von Softwaresystemen ist ein elementarer Teil der Qualitätssicherung während eines Softwareentwicklungsprozesses. Um die Qualität des Testprozesses und somit des zu entwickelnden Produkts gewährleisten zu können, muss dieser systematisch und strukturiert geplant und durchgeführt werden. Eine solche systematische Vorgehensweise für einen allgemeinen Testprozess liefert der FTP [6], der den folgenden Ausführungen zu Grunde liegt und als bekannt vorausgesetzt wird.

Ein wichtiger Teil des Testprozesses ist der Entwurf von Testfällen. Die quantitativ und qualitativ richtige Auswahl an Testfällen entscheidet darüber, ob vorhandene Fehler im Testgegenstand gefunden werden und welche Kosten das Finden dieser Fehler verursacht. MBT systematisiert den Testentwurf und ermöglicht die automatisierte Generierung von Testfällen.

Der FTP berücksichtigt nicht die besonderen Merkmale von MBT, da er die allgemeine Herangehensweise beschreibt. Um die besonderen Merkmale von MBT mit Bezug auf den FTP zu berücksichtigen, bedarf es eines allgemeinen modellbasierten Testprozesses, welcher die Aktivitäten und Artefakte des FTP für MBT konkreter definiert. Um dies zu erreichen, müssen die Aktivitäten und Artefakte des FTP mit denen der existierenden Vorgehensmodelle für MBT einander gegenübergestellt werden (Abb. 1, (1)).

Um erste Ergebnisse für einem allgemeinen modellbasierten Testprozess zu erhalten, wurde im Rahmen dieser Arbeit zunächst an einem Fallbeispiel ein vom fundamentalen Testprozess abgeleiteter Testprozess mit einem modellbasierten verglichen. Auf Basis von El-Far und Whittaker [2] sowie Utting, Pretschner und Legeard [7] (Abb. 1, (2)) wurde ein Modellierungswerkzeug mit einem Zustandsdiagramm beschrieben und die Testfälle mit Hilfe eines Algorithmus zur Knotenüberdeckung generiert. Der daraus resultierende konkrete modellbasierte Testprozess (Abb. 1, (3)) beschreibt hauptsächlich die Aktivitäten und Artefakte die im FTP den Phasen Analyse und Design sowie Realisierung und Durchführung zugeordnet werden können. Die FTP-Phasen Planung, Steuerung, Auswertung und Bericht und Abschluss wurden für den konkreten modellbasierten Testprozess aus dem FTP abgeleitet. Dies geschah mit Hinblick auf ein gesamtheitliches modellbasiertes Vorgehen. Anschließend wurde der konkrete modellbasierte Testprozess einem manuellen Testprozess (Abb. 1, (4)) gegenübergestellt, der ausschließlich vom fundamentalen Testprozess (Abb. 1, (5)) abgeleitet wurde. Der Vergleich (Abb. 1, (6)) beschreibt Unterschiede bezüglich der durchgeführten Aktivitäten und entstandenen Artefakten zwischen (Abb. 1, (3)) und (Abb. 1, (4)). Unterschiede auf konkreter Ebene bedeuten, dass sie jeweils in der einen oder anderen Ausprägung vom FTP abgleitet werden können. Es wurden auch Unterschiede auf abstrakter Ebene festgestellt, was bedeutet, dass nicht alle spezifisch modellbasierten Aktivitäten und Artefakte vom FTP abgeleitet werden können. Diese Ergebnisse der Gegenüberstellung werden in Abbildung 2 dargestellt. Das hier dargelegte Ergebnis ist ein erster Ansatzpunkt, um mittels weiterer Arbeiten festzustellen, ob ein allgemeiner modellbasierter Testprozess (Abb. 1, (7)) definiert werden sollte.

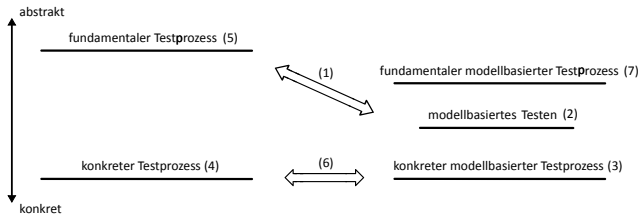

Abbildung 1: Vorgehensweise

2. GEGENÜBERSTELLUNG

Im Folgenden wird die Gegenüberstellung der Artefakte und Aktivitäten beider Testprozesse beschrieben (Abb. 2) und an Beispielen aus den durchgeführten Testprozessen näher erläutert. Dabei wird unterschieden zwischen Änderungen, die auf der abstrakten Ebene des FTPs stattfanden („Neu“ und „fällt weg“ in Abb. 2), und solchen, bei denen MBT den Prozess lediglich auf konkreter Ebene einschränkt

(„Änderung“ in Abb. 2).

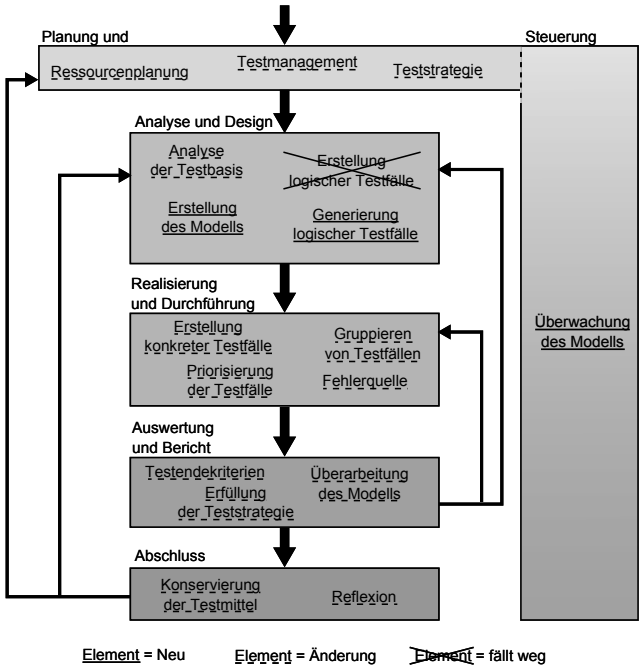

Abbildung 2: Beobachtete Änderungen

In der **Planungsphase** ergeben sich Änderungen auf konkreter Ebene sowohl bei der Ressourcenplanung als auch bei der Erstellung der Teststrategie (vgl. Abb. 1). Bei der *Ressourcenplanung* muss mehr personeller und zeitlicher Aufwand für das eventuelle (Neu-)Erstellen und Pflegen des benötigten Modells und weniger für das Erstellen der Testfälle eingeplant werden. Während im nicht-modellbasierten Beispieltestprozess die Erstellung der Testfälle als weitere Aufgabe für die Entwickler eingeplant wurde, musste im modellbasierten Testprozess ein dedizierter Mitarbeiter für die Erstellung und Pflege des Modells eingeplant werden. Beim Erstellen der *Teststrategie* ist zu überlegen, ob Prioritäten und Testintensitäten, die bei Testdurchführung berücksichtigt werden müssen, bereits im Modell oder erst in den Testfällen angegeben werden. Beim *Testmanagement* muss auf konkretem Level die Verwaltung des Modells in den Prozess integriert werden. Dies umfasst primär die Protokollierung von Modellerstellung und Änderungen am Modell. Weitergehend muss eine spezielle *Testinfrastruktur* und angepasste Testwerkzeuge zur Modellerstellung und Testfallgenerierung bereitgestellt werden. Für die Verwaltung des Modells wurde sich im modellbasierten Beispieltestprozess für den Einsatz des Versionsverwaltungssystems Subversion[1] entschieden. Als Werkzeug zur Modellerstellung wurde das Eclipse[2]-Plugin UMLet[3] gewählt.

Bei der **Steuerung** muss als neue Aktivität die *Überwachung des Modells* stattfinden. Es muss während des gesamten Prozesses kontrolliert und protokolliert werden, welche Änderungen am Modell vollzogen werden. So kann sichergestellt werden, dass die durchgeführten Tests immer zum jeweiligen Stand des Modells passen. Im modellbasierten Beispieltestprozess konnte dies ebenfalls durch den Einsatz des Versionsverwaltungssystems Subversion mitabgedeckt werden.

In **Analyse und Design** sind die wesentlichsten Anpassungen des Testprozesses zu beobachten. Die *Analyse der Testbasis* findet wie im FTP beschrieben statt. Es wird festgelegt, was für eine Art von Testfällen erforderlich ist. An dieser Stelle muss aber nun zusätzlich ein passender Modelltyp gefunden werden, der die Generierung der geforderten Art von Testfällen durch einen Algorithmus zulässt. In unserem Fall sollten die Testfälle Tests auf Basis der Benutzungsschnittstelle des Testobjekts beschreiben. Dazu passend wurde ein auf einem Zustandsautomaten basierender Modelltyp gewählt. In diesem stand eine Transition für die Ausführungen einer bestimmten Nutzeraktionen und ein Zustand für den Zustand des Systems nach einer bestimmten Nutzeraktion. Es kann auch ein Testmodell vorhanden sein, an dem sich die Testfälle orientieren müssen. Die *Erstellung der logischen Testfälle* wird ersetzt durch die *Erstellung des Modells* aus der diese im weiteren Verlauf generiert werden. Hierbei werden alle verfügbaren Informationen herangezogen, um das erwartete Systemverhalten so genau wir möglich in dem Modell beschreiben zu können. Bei Modellerstellung muss abgewogen werden, in welchem Maße das Testorakel im Modell integriert werden soll, was den Soll-Ist-Vergleich bei Testdurchführung beeinflusst. Bezüglich konkreter Testfälle muss abgewogen werden, ob Testdaten bereits im Modell integriert werden, wodurch logische Testfälle komplett entfallen und direkt konkrete Testfälle generiert werden können.

Die abschließende Aktivität in dieser Phase ist die Generierung logischer Testfälle, welche die *Erstellung logischer Testfälle* ersetzt. Hier bedarf es der Entwicklung eines geeigneten Algorithmus, der aus dem Modell logische Testfälle automatisch ableitet. Im modellbasierten Beispieltestprozess war dies ein Algorithmus zur Knotenüberdeckung, der folglich Testfälle generierte, in welchen jede Benutzeraktion mindestens einmal ausgeführt wird.

In der **Realisierung und Durchführung** müssen lediglich die abstrakten Aktivitäten und Artefakte des fundamentalen Testprozesses konkretisiert werden. Zur *Priorisierung der Testfälle* wurden nach Möglichkeit bereits Prioritäten im Modell integriert, sodass diese bei Generierung in die Testfälle übertragen werden konnten. War dies nicht möglich, muss die Priorisierung in dieser Phase des Prozesses stattfinden. Die separate *Erstellung konkreter Testfälle* ist nur dann noch notwendig, wenn Testdaten nicht bereits im Modell integriert wurden, das heißt, wenn überhaupt logische Testfälle existieren. Wurden direkt konkrete Testfälle aus dem Modell generiert, entfällt dieser Schritt. Nach Durchführung der Tests ist bei einem Unterschied zwischen tatsächlichem und erwartetem Ergebnis zusätzlich zum Testobjekt das Modell und der Testfallgenerierungsalgorithmus als mögliche *Fehlerquelle* in Betracht zu ziehen. Eine weitere Optimierung kann beim *Gruppieren der Testfälle* zu Testsequenzen stattfinden. Dieses kann bereits während der Testfallgenerierung durchgeführt werden, wenn entsprechende Techniken im Testfallgenerierungsalgorithmus integriert werden.

In **Auswertung und Bericht** sind auf abstrakter Ebene keine Änderungen am Testprozess notwendig. Es müssen auch mehrere *Testzyklen* eingeplant werden und nach jeder Testdurchführungsphase muss eine Aufwands-/ Nutzenabschätzung stattfinden, ob weitere Testfälle erstellt werden

[1] http://subversion.tigris.org/
[2] http://www.eclipse.org/
[3] http://www.umlet.com/

müssen. Genauso muss der *Testbericht* am Ende der Phase verfasst werden. Es muss ebenfalls überlegt werden, ob die *Testendekriterien* erfüllt sind. Mit Bezug auf die Testfallgenerierung muss an dieser Stelle beurteilt werden, ob der Generierungsalgorithmus tatsächlich entsprechend der Teststrategie viele Testfälle abdeckt. Modellüberdeckung kann hier eine angebrachte Metrik sein. Andernfalls muss er angepasst werden, damit die *Erfüllung der Teststrategie* erreicht werden kann. Auf Basis des Modells sind weitergehend neue Testendekriterien möglich. Wenn die Ursache einer größeren Menge von Fehlern im Modell liegt, muss bezüglich der Fehlerwirkung in dieser Phase geprüft werden, ob eine *Änderung des Modells* für einen weiteren Testzyklus über die **Steuerung** initiiert werden muss.

In der **Abschlussphase** müssen die Testmodelle als Interpretation der Anforderungen bei der *Konservierung der Testware* berücksichtigt werden, damit bei der Wartung auch diese genutzt werden können. Die *Reflexion* muss jegliche Prozesse, die mit der Modellerstellung und Testfallgenerierung in Verbindung stehen, miteinbeziehen.

3. FAZIT UND AUSBLICK

Die identifizierten Unterschiede zwischen Aktivitäten und Artefakten bei modellbasiertem Testen und denen des FTPs lassen schließen, dass ein fundamentaler Testprozess für MBT notwendig ist. Als gravierendste Unterschiede sind die neuen oder geänderten Aktivitäten und Artefakte zu nennen, die in direkter Verbindung mit der Analyse der Testbasis und dem Entwurf logischer Testfälle stehen. Unterstützt wird diese Meinung durch die große Anzahl der Elemente, die für MBT konkreter definiert werden können, als es auf der abstrakten Ebene des fundamentalen Testprozesses der Fall ist. Aber auch die vorbereitende Planungsphase sowie die Durchführungsphase werden von modellbasiertem Testen stark beeinflusst. Die gesamte Analyse fand auf Basis eines konkreten Fallbeispiels statt. Sie bildet so die Basis für weitere Untersuchungen, möglicherweise an industrienahen Projekten, bei denen MBT eingesetzt wurde, beispielsweise [1]. Weiterhin sollten unterschiedlichste modellbasierte Herangehensweisen [3] berücksichtigt werden. Zudem muss untersucht werden, wie der allgemeine modellbasierte Testprozess optimal in den Entwicklungsprozess eingegliedert werden kann. Mit dem Ergebnis dieser und den Ergebnissen zukünftiger Arbeiten kann anschließend ein allgemeiner Testprozess für MBT formuliert werden. Dieser kann dann die Systematik bei der Durchführung von zukünftigen modellbasierten Testprozessen sicherstellen.

4. LITERATUR

[1] Dalal, S. R., A. Jain, N. Karunanithi, J. M. Leaton, C. M. Lott und B. Bellcore: *Model-based testing in practice*. In: *Proc. Intl. Conf. on SE (ICSE '99*, S. 285–294, 1999.

[2] El-Far, I. K. und J. A. Whittaker: *Model-based Software Testing*. Encyclopedia on Software Engineering, 2001.

[3] Neto, A. C. D., R. Subramanyan, M. Vieira und G. H. Travassos: *A survey on model-based testing approaches: a systematic review*. In: *WEASELTech '07: Proc. of the 1st ACM Intl. workshop on Empirical assessment of software engineering languages and technologies*, S. 31–36, New York, NY, USA, 2007. ACM.

[4] Pretschner, A. und J. Philipps: *Methodological Issues in Model-Based Testing*. In: *Model-Based Testing of Reactive Systems*, S. 281–291, 2004.

[5] Robinson, H.: *Intelligent Test Automation*. Software Testing & Quality Engineering (STQE), S. 24–32, 2000.

[6] Spillner, A. und T. Linz: *Basiswissen Softwaretest: Aus- und Weiterbildung zum Certified Tester; Foundation Level nach ISTQB-Standard*. dpunkt-Verl., Heidelberg, 2005.

[7] Utting, M., A. Pretschner und B. Legeard: *A taxonomy of model-based testing*. Working Paper, April 2006.