Evaluierung des Fehlererkennungspotentials modellbasierter Komponenten- und Integrationstestfälle

F. Pinte und F. Saglietti

Lehrstuhl für Software Engineering (Informatik 11)
Universität Erlangen-Nürnberg
Martensstraße 3
91058 Erlangen (D)
pinte@informatik.uni-erlangen.de
saglietti@informatik.uni-erlangen.de

Abstract: Der vorliegende Beitrag präsentiert neueste Ergebnisse einer Evaluierungsstudie über die Effizienz modellbasierter Verifikationsansätze im Hinblick auf Anzahl und Art erkennbarer Fehler. Ein seit kurzem einsatzbereites Werkzeug zur vollautomatischen modellbasierten Testfallerzeugung lieferte hierfür ein wertvolles Forschungsinstrument, das in Bezug auf die Aufdeckung von Modellierungsfehlern und Implementierungsfehlern zur Erlangung weiterer Einsichten auf diesem Gebiet verhelfen konnte.

1 Einführung

Die weitgehend automatische Umsetzung modellbasierter Testverfahren war Gegenstand des mehrjährigen Verbundprojekts UnITeD. Im Rahmen dieses kürzlich abgeschlossenen Vorhabens wurde ein gleichnamiges Werkzeug zur Testfallgenerierung entwickelt. Zu vorgegebenen Überdeckungskriterien [SOP07] erzeugt UnITeD vollautomatisch Testsequenzen mit zugehörigen Eingabedaten sowohl für den Komponenten- [OSS07] als auch für den Integrationstest [PSO08]. Darüber hinaus unterstützt das implementierte Werkzeug auch die Visualisierung der erzielten Testüberdeckung [PSN09] sowie die systematische Erkennung und Ergänzung wieder verwendbarer Testläufe im Regressionstest [PBS08]. Die zu diesen Zwecken entwickelten Verfahren sind vor allem durch multikriterielle Optimierung gekennzeichnet; hierdurch wird sowohl die Maximierung der Testobjektüberdeckung als auch die Minimierung der Testfallanzahl verfolgt.

Das Werkzeug wurde anschließend – neben dem Piloteinsatz in industriellen Umgebungen – auch als wissenschaftliches Forschungsinstrument zur Evaluierung des Fehlererkennungspotentials unterschiedlicher modellbasierter Testverfahren eingesetzt. Ziel der hier vorgestellten Evaluierungsstudie ist es, die Effizienz modellbasierter Verifikationsansätze im Hinblick auf Anzahl und Art dadurch aufgedeckter Fehler sowie den damit verbundenen Aufwand vergleichend zu bewerten. Das Papier ist folgendermaßen gegliedert: In Abschnitt 2 werden unterschiedliche Mutationsklassen zur Simulierung entsprechender Fehlerarten eingeführt. Die anschließend in Abschnitt 3

vorgestellte Mutationsanalyse erlaubt eine experimentelle Evaluierung des Fehlererkennungspotentials im Hinblick auf Modellierungsfehler. Darüber hinaus wird in Abschnitt 4 über die Möglichkeit der Aufdeckung von Implementierungsfehlern berichtet.

2 Mutationsklassen

Zur Untersuchung modellbasierter Testfallmengen hinsichtlich ihres Fehlererkennungspotentials am Modell wurden zunächst unterschiedliche Fehlerarten identifiziert und klassifiziert, wobei unterschieden wurde zwischen:
- intramodularen Fehlern im Entwurf einer Komponente und
- intermodularen Fehlern in der Interaktion zwischen zwei Komponenten.

2.1 Intramodulare Mutationsklassen

Zur Simulation intramodularer Fehlerarten wurden folgende Mutationsklassen für zustandsbasierte Komponentenmodelle betrachtet:
- Trigger-Mutationen durch Entfernen, Hinzufügen bzw. Verfälschen des Auslösers eines Zustandsübergangs,
- Guard-Mutationen durch Entfernen, Hinzufügen bzw. Verfälschen eines Wächters an einem Zustandsübergang (durch Änderung der Vergleichsoperatoren, Boolescher Operatoren bzw. zugrunde liegender Variablen),
- Effekt-Mutationen durch Entfernen, Hinzufügen bzw. Verfälschen des von einem Zustandsübergang ausgelösten Ereignisses (durch Änderung der auszulösenden Variablenzuweisungen),
- Pre-State-Mutationen durch Änderung des Anfangszustandes einer Transition,
- Post-State-Mutationen durch Änderung des Endzustandes einer Transition,
- Transition-Mutationen durch Entfernen bzw. Hinzufügen eines Zustandsübergangs,
- Zustand-Mutationen durch Entfernen bzw. Hinzufügen eines Zustands.

2.2 Intermodulare Mutationsklassen

Zur Simulation intermodularer Fehlerarten wurden folgende Mutationsklassen für zustandsbasierte Modelle aufrufender und aufgerufener Komponenten betrachtet:
- Trigger-Mutationen durch Entfernen, Hinzufügen bzw. Verfälschen des Auslösers eines Zustandsübergangs,
- Guard-Mutationen durch Entfernen, Hinzufügen bzw. Verfälschen eines Wächters an einem Zustandsübergang,
- Effekt-Mutationen durch Entfernen, Hinzufügen bzw. Verfälschen des von einem Zustandsübergang ausgelösten Ereignisses (durch Änderung der auszulösenden Operationsaufrufe),
- Pre-State-Mutationen durch Änderung des Anfangszustandes einer Transition,
- Post-State-Mutationen durch Änderung des Endzustandes einer Transition.

3 Testfallgenerierung und Mutationsanalyse

Zur Analyse des Fehlererkennungspotentials wurden einerseits Testüberdeckungskriterien auf Komponentenebene (Zustands-, Transitions-, Transitionspaarüberdeckung), andererseits zustandsbasierte Integrationstestkriterien ausgewählt. Letztere beruhen auf dem Konzept der Überdeckung interagierender Transitionen [SOP07], d.h. von Paaren (t, t') von Transitionen, so dass als Effekt der Transition t in einer aufrufenden Komponente die Transition t' in der aufgerufenen Komponente traversiert wird. In [SOP07] wurden auf der Basis dieses Konzeptes 8 unterschiedliche Überdeckungsmaße definiert, deren Subsumptionshierarchie in Abb. 1 dargestellt wird.

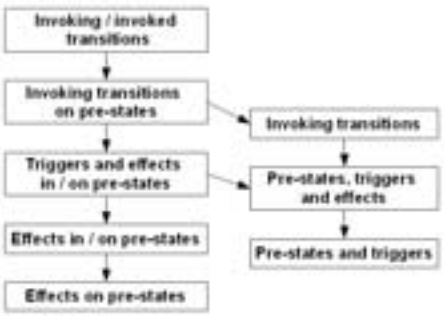

Abbildung 1: Subsumptionshierarchie der zustandsbasierten Integrationstestkriterien

Um das Fehlererkennungspotential eines Überdeckungskriteriums zu bewerten, wurde der Fehlererkennungsgrad von Testfallmengen betrachtet, die besagtes Kriterium erfüllen und mit dem im Projekt UnITeD entwickelten Werkzeug (s. [OSS07], [PSO08]) generiert wurden. Zur automatischen Ermittlung des Fehlererkennungsgrades wurden durch Anwendung unterschiedlicher Mutationsklassen (s. Abschnitt 2) zahlreiche Mutanten des ursprünglichen Modells generiert. Weicht nach Ausführung eines Mutanten mit den zu bewertenden Testfällen dessen von außen beobachtbares Verhalten von dem ursprünglichen Modellverhalten ab, so kann die zugrunde liegende Testfallmenge im Hinblick auf die Fehleraufdeckung der betrachteten Mutation als adäquat betrachtet werden. Die im Folgenden dargestellten Ergebnisse bzgl. der Anzahl erkannter Mutanten sind als konservativ zu betrachten, da sie auch Mutanten umfassen, die zum ursprünglichen Modell funktionsäquivalent und deshalb nicht erkennbar sind.

Die Mutationsanalyse wurde exemplarisch auf das Modell eines Softwaresystems angewendet, das die korrekte Positionierung einer Patientenliege beim Einsatz eines medizintechnischen Gerätes zu überprüfen hat. Zu diesem Zweck kontrolliert und verwaltet die Software einzelne Prüfschritte, die vor Auslieferung jeder Liege durchzuführen sind; darüber hinaus ist sie auch für die Generierung eines Protokolls über die durchgeführten Prüfschritte und deren Ergebnisse zuständig. Neben dem internen Verhalten der Software beschreibt das Modell auch die vom Benutzer wahrgenommene Mensch-Maschine Interaktion, also die Abfolge der dem Benutzer angebotenen Dialoge. Zur Realisierung dieses Verhaltens wurden 2 interagierende Software-Komponenten durch UML-Zustandsmaschinen modelliert.

3.1 Fehlererkennungspotential auf Komponentenebene

Zur Bewertung des Fehlererkennungspotentials auf Komponentenebene wurden sämtliche Mutationsklassen aus Abschnitt 2.1 wiederholt instanziiert und die dadurch erzielten Mutationen auf die komplexere der beiden Komponenten angewendet. Die generierten Testfallmengen erreichten jeweils folgende Modellüberdeckung:

- 100%ige Zustandsüberdeckung durch 8 Testfälle,
- 100%ige Transitionsüberdeckung durch 27 Testfälle,
- 97,8%ige Transitionspaarüberdeckung durch 80 Testfälle.

Abb. 2 zeigt die jeweiligen Anteile an erkannten Mutationsklassen.

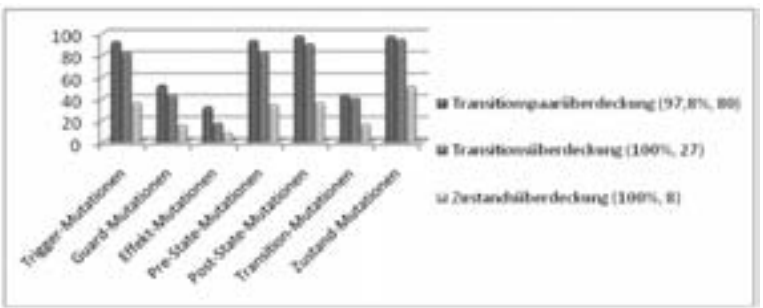

Abbildung 2: Ergebnisse der Mutationsanalyse für Komponententestkriterien

Erwartungsgemäß konnte das stärkste Kriterium, nämlich die Transitionspaar-überdeckung, trotz des höheren Aufwands (und der deshalb mit den vorhandenen Ressourcen nur unvollständig erzielbaren Überdeckung) hinsichtlich des Fehlererkennungsgrads am besten abschneiden. 63% aller Mutanten wurden identifiziert, während durch Transitionsüberdeckung nur 55% und durch Zustandsüberdeckung nur 23% aller Mutanten erkannt werden konnten.

Dabei muss der Fehlererkennungsgrad bzgl. der einzelnen intramodularen Mutationsklassen differenziert betrachtet werden. Bezogen auf Mutationen, die grundsätzliche Strukturänderungen enthalten, wie z.B. geänderte End- oder Anfangszustände von Transitionen, erreichte die Transitionspaarüberdeckung sehr gute Ergebnisse (über 90%). Mutationen, die geringfügige Änderungen im Datenfluss beinhalten, wie z.B. Effekt-Mutationen, wurden allerdings nur zu einem ca. 30%igen Anteil erkannt. Eine Erklärung hierfür liegt darin, dass die hier betrachteten Testkriterien auf strukturellen und nicht auf datenflussbasierten Überdeckungskonzepten basieren.

3.2 Fehlererkennungspotential der Integrationsteststrategien

Während im Komponententest die Funktionsweise innerhalb einzelner Komponenten getestet wird, prüft der Integrationstest speziell die Komponenteninteraktionen. Die implementierte Mutationsanalyse ermöglicht auch die Untersuchung der Fehlererkennung von Testfällen, die zustandsbasierten Interaktionsüberdeckungs-kriterien genügen. Die Ergebnisse sind in Abb. 3 illustriert.

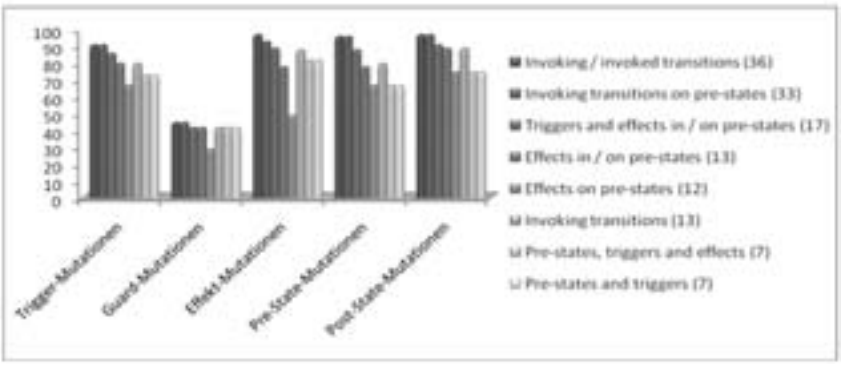

Abbildung 3: Ergebnisse der Mutationsanalyse für Integrationstestkriterien
(pro Kriterium Angabe der zu einer 100%igen Überdeckung benötigte Anzahl an Testfällen)

Wie bereits auf Komponentenebene, erkannte die Testfallmenge, die zum anspruchsvollsten Kriterium generiert wurde, die meisten Mutanten. In diesem Falle handelt es sich um das „Invoking / invoked transitions“ - Kriterium, das 84% der generierten Mutanten zu erkennen erlaubte. Mit Ausnahme der Guard-Mutationen erkannte diese Testfallmenge über 90% aller erzeugten Mutanten. Guard-Mutationen konnten allerdings nur zu einem 44%igen Anteil erkannt werden. Dies lässt sich dadurch erklären, dass ursprünglich generierte Testfälle nach Entfernung vorhandener Wächterbedingungen ihren Testablauf (und damit das Testverhalten) im Wesentlichen beibehalten. Nur geringfügig schlechtere Ergebnisse (83%) erzielte die Testfallmenge zum zweitstärksten Kriterium „Invoking transitions on pre-states“, die aus heuristischen Gründen zum Teil die gleiche Anzahl interagierender Transitionspaare erzielen konnte.

Abschließend sollte nicht unerwähnt bleiben, dass der so gestaltete Integrationstest - über die Erkennung intermodularer Fehler hinaus - auch die Aufdeckung intramodularer, einem vorausgegangenen Komponententest evtl. entgangener Fehler unterstützt. Um auch dieses Fehlererkennungspotential experimentell zu untersuchen, wurden sämtliche Mutationsarten aus Abschnitt 2 auf beide Systemkomponenten angewendet. Erwartungsgemäß konnte die gemäß höchster Integrationstestanforderungen generierte Testfallmenge mehr intermodulare Mutanten als intramodulare Mutanten aufdecken, da das zugrundeliegende Testkriterium ja auf die Überdeckung von Komponenten-interaktionen ausgerichtet war. Dennoch weist der relativ hohe Anteil (59%) an erkannten Komponentenfehlern darauf hin, dass ein automatisierter Integrationstest auch im Hinblick auf Schwächen vorausgegangener (bei vorgefertigten Komponenten u.U. ausgelagerten, kontextfremden) Verifikationsaktivitäten wertvolle Hilfe bieten kann.

4 Erkennungspotential hinsichtlich Implementierungsfehler

An der gleichen Anwendung wurde die beschriebene automatische Testfallerzeugung mit der manuellen Ermittlung (ohne Modellbeschreibung) verglichen. Zunächst wurde der Vergleich in Bezug auf den Aufwand beider Vorgehensweisen durchgeführt: die

Erstellung des Modells erforderte ca. 18 Stunden, während die manuelle Testfallermittlung etwa 12 Stunden in Anspruch nahm. Auf dieser Grundlage lohnt sich das modellbasierte Vorgehen nur unter der Annahme, dass es eine entsprechend höhere Qualität des Endprodukts in Aussicht stellt. Tatsächlich stellte sich beim anschließenden Vergleich beider Verfahren in Bezug auf die Testobjektüberdeckung heraus, dass die automatisch generierten Testfälle eine deutlich höhere Überdeckung interagierender Transitionen als die manuell erzeugten Testfälle erzielen konnten (100% vs. 58%); dementsprechend höher war auch deren Fehleraufdeckung, insbesondere in Bezug auf drei Implementierungsfehler, die die manuell gestalteten Tests nicht erkannten. Dabei handelte es sich im Wesentlichen um inkorrekte Reaktionen auf vom üblichen Nutzungsprofil abweichende Aktionen des Anwenders. Beispielsweise wurde ein anomales Verhalten seitens des Anwenders, wie etwa das Abbrechen eines Prüfschrittes vor Einstufung der Prüfergebnisse, nicht korrekt umgesetzt. Das höhere Fehlererkennungspotential modellbasierter Testfälle lässt sich durch die frühe Modellierung der Mensch-Maschine Interaktion unter Berücksichtigung entsprechender Ausnahmeszenarien erklären.

5 Zusammenfassung

In diesem Beitrag wurden Ergebnisse aus einer Evaluierungsstudie über die Effizienz modellbasierter Verifikationsansätze im Hinblick auf Anzahl und Art erkennbarer Fehler präsentiert. Die experimentelle Untersuchung erfolgte mit Hilfe eines seit kurzem einsatzbereiten Werkzeugs zur vollautomatischen modellbasierten Testfallerzeugung. Die Analyse umfasste die Erkennbarkeit von Modellierungsfehlern und von Implementierungsfehlern.

Literaturverzeichnis

[SOP07] Saglietti, F.; Oster, N.; Pinte, F.: Interface Coverage Criteria Supporting Model-Based Integration Testing. In M. Platzner, K.-E. Großpietsch, C. Hochberger, A. Koch (Eds.): ARCS '07 Workshop Proceedings, VDE Verlag, 2007.

[OSS07] Oster, N.; Schieber, C.; Saglietti, F.; Pinte, F.: Automatische, modellbasierte Testdatengenerierung durch Einsatz evolutionärer Verfahren. In Informatik 2007 - Informatik trifft Logistik, Lecture Notes in Informatics, Vol. 110, Gesellschaft für Informatik, 2007.

[PBS08] Pinte, F.; Baier, G.; Saglietti, F.; Oster, N.: Automatische Generierung optimaler modellbasierter Regressionstests. In Informatik 2008 - Beherrschbare Systeme dank Informatik, Lecture Notes in Informatics, Vol. 133, Gesellschaft für Informatik, 2008.

[PSN09] Pinte, F.; Saglietti, F.; Neubauer, A.: Visualisierung überdeckter sowie zu überdeckender Modellelemente im modellbasierten Test. In Informatik 2009 - Im Fokus das Leben, Lecture Notes in Informatics, Vol. 154, Gesellschaft für Informatik, 2009.

[PSO08] Pinte, F.; Saglietti, F.; Oster, N.: Automatic Generation of Optimized Integration Test Data by Genetic Algorithms. In Software Engineering 2008 - Workshopband, Lecture Notes in Informatics, Vol. 122, Gesellschaft für Informatik, 2008.