MoMuT - Eine Transfer-Geschichte über modellbasiertes Testen

Rupert Schlick[1]

Abstract: Forschungstransfer hat viele Gesichter, aber in den seltensten Fällen eine geradlinige Entwicklung. Am Beispiel modellbasierter, mutationsgetriebener Testfallgenerierung wird die Entwicklung einer Methode von der Konzeptionsphase zu (ersten) kommerziellen Verwertungen erzählt. Anhand dessen werden einige der wirkenden Kräfte und Einflussfaktoren betrachtet.

Keywords: Modellbasiertes Testen; Erfahrungsbericht Forschungs-Transfer; Reaktive Systeme

1 Einleitung und Hintergrund

AIT arbeitet an Verfahren für mutationsgetriebene, modellbasierte Testfallgenerierung. Die dabei entstandene Familie von Testfallgeneratoren *MoMuT*[2], wurde in mehreren Etappen um verschiedenste neue Module ergänzt. Im Folgenden betrachten wir deren Werdegang und wie sich Industrieanforderungen auf Entwicklung und Transfer ausgewirkt haben.

Prinzipiell gibt es zwei Ansätze für den Transfer von Forschungsergebnissen in die industrielle Anwendung: *Market-Pull*, wenn ein konkretes Problem von der Industrie an einen Forschungspartner herangetragen und von diesem (hoffentlich) gelöst wird, und *Technology-Push*, wenn eine als nützlich vermutete Technologie umgesetzt und dann dazu Nutzer gesucht werden. Ein reiner Market-Pull, der nicht innerhalb der Forschungsabteilung eines Unternehmens, sondern durch Zukauf der Forschungsleistung bei einem Transfer-Institut erfolgt, ist im Software-Engineering-Umfeld selten, da die Probleme nicht so kundenspezifisch sind.

Die Realität in geförderten, anwendungsorientierten Forschungsprojekten liegt zwischen den beiden Extremen. Um Mittel zu erhalten, muss das Vorhaben eine benennbare, generell nützliche Innovation als Lösung für eine Kategorie von Problemen versprechen. Üblicherweise sind, im Gegensatz zu einem reinen Technology-Push, Anwendungspartner mit an Bord, die Beispiel-Probleme mitbringen. Diese haben Interesse an der Lösung. Jedoch sind die Beispiel-Probleme womöglich gar nicht mit dem Ansatz lösbar oder nicht repräsentativ.

[1] AIT Austrian Institute of Technology GmbH, Center for Digital Safety and Security, Giefinggasse 4, 1210 Wien, Österreich, rupert.schlick@ait.ac.at

[2] `http://momut.org`

doi:10.18420/SE2020_56

Modellbasiertes Testen (MBT) generiert Testsequenzen aus Modellen. Hier verstehen wir darunter, aus einem Modell, das das Verhalten der Software/des Systems spezifiziert, Testfälle mit gegebenen, angestrebten Testzielen, z.B. einer Abdeckung des Modells, zu erstellen. Das Erstellen des (Test-)Modells ersetzt das Erstellen der einzelnen Testfälle. Beides erfordert ein Verständnis der Anforderungen - der Aufwand für die beiden Erstellungsschritte liegt in der gleichen Größenordnung, erfordert aber verschiedene Fähigkeiten und Fertigkeiten.

Ausgehend vom unveränderten Modell, kann die Tiefe/Qualität, bis zu der getestet werden soll, weitgehend frei erhöht werden. Bei direkt erstellten Tests erfordert dies zusätzlichen Arbeitsaufwand, mit MBT nur Rechenaufwand. Am meisten Vorteile bringt MBT in der Wartung und in der Entwicklung abgeleiteter Softwarevarianten. Modelländerungen für neue Funktionen bedeuten einen überschaubaren Aufwand. Dann kümmert sich der Testfallgenerator darum, welche Tests unverändert bleiben, wegfallen, adaptiert werden müssen oder neu hinzukommen. Auch in gut strukturierten Testsuiten gibt es eine Größenobergrenze, ab der es personell nicht mehr sinnvoll möglich ist, zu überprüfen, ob alle Tests nach wie vor ihren ursprünglichen Zweck erfüllen.

Mutationsgetriebenes Modellbasiertes Testen Ein mögliches Testziel im MBT ist Mutationsabdeckung. Dabei wird im Modell jeweils eine kleine Änderung vorgenommen, ein kleiner Fehler eingebracht. Anschließend werden Testsequenzen gesucht, in denen sich das außen (durch den Tester/Benutzer) beobachtbare Verhalten von verändertem und unverändertem Modell unterscheiden. Typische Kontrollflussabdeckungen als Testziel führen dagegen tendenziell zu schlechteren Tests. Ein erfahrener Tester baut, ebenso wie die Mutationsabdeckung, Tests, die ein Problem beobachtbar machen. Bei Strukturabdeckungen kann ein Test nach Erreichen der geforderten Stelle im Modell enden, obwohl ein aufgetretenes Problem nicht beobachtbar wird.

2 Initialzündung

Das FP7-Projekt MOGENTES[3] beschäftigte sich mit fehlerbasierten Methoden zum Test. Mutationsbasierte Testfallgenerierung war im Prinzip als Möglichkeit bekannt [AD06], aber sowohl akademische als auch kommerzielle Werkzeuge für mutationsgetriebenes MBT Testen waren nicht verfügbar. Im Projekt arbeiteten fünf Forschungspartner und ein Hersteller von Entwicklungswerkzeugen an fehlergetriebenen Verifikationsansätzen für Beispielanwendungen von vier Industriepartnern. TU Graz und AIT entwickelten einen ersten mutationsgetriebenen Testfallgenerator für UML [Ai11]. UML wird dazu in die Zwischensprache OOAS (Object Oriented Action Systems) übersetzt. Implementiert wurde mit VIATRA, ANTLR, C# und Prolog.

Der Ansatz funktionierte, hatte aber erhebliche Skalierungsprobleme. Für das Beispiel eines Eisenbahnstellwerks konnte nur eine minimalistische Schienentopologie verwendet werden

[3] Model-based Generation of Tests for Dependable Embedded Systems, `http://mogentes.eu/`, 2008-2011

und die Steuerung einer Baggerschaufel erforderte Hinweise zu Äquivalenzklassen im Input, da der enumerative Ansatz mit großen Datenbereichen überfordert war. Ein einfaches, aber realistisches Beispiel einer Automobilalarmanlage funktionierte hingegen sehr gut und ist repräsentativ für eine ganze Klasse von Aufgaben im Bereich Chassis-Steuerung.

3 Performance

Das national geförderte Projekt TruFal[4] widmete sich der Fragestellung, wie der Ansatz effizienter umzusetzen wäre. Ein wiederum durch die TU Graz in Prolog umgesetzter Generator unter Einsatz von symbolischen Techniken und SMT-Solvern (Z3) lieferte gute Ergebnisse für die Baggerschaufel-Steuerung aus MOGENTES und die Testsystem-Einbindung eines Abgas-Messgerätes [Ai14], scheiterte aber, wegen Datentyp-Einschränkungen des Solvers, wegen des großen Zustandsraums und wegen wenig optimierter Implementierung an größeren Topologien für die Stellwerkssteuerung und an einer Safety-Middleware.

Die Transformation von UML nach OOAS und der OOAS-Parser wurden nach Java migriert, um Erweiterungen und Änderungen zu erleichtern. Die Abkehr von VIATRA war motiviert von der leichteren Verfügbarkeit von JAVA-Programmierern. Gegen Ende von TruFal implementierte AIT einen neuen, wieder enumerativen Generator, der das hochgradig nebenläufig modellierte Stellwerksmodell bewältigen sollte. Er vermeidet die zuvor pro Mutant wiederholten Explorationen und führt Mutanten nur für die jeweils notwendigen Schritte aus. Das Explorations-Framework verwendet zudem einen (später als zu einfach erkannten) Ansatz für Partial Order Reduction, wie ihn Model Checker für nebenläufige Systeme verwenden. Die Architektur wurde gezielt performant gewählt, das Modell wird von OOAS nach LLVM-IR weitertransformiert und dann in Maschinencode übersetzt, der Generator selbst ist in C++ implementiert.

4 Babel der Modelle

Im Rahmen des Projekts MBAT[5] wurde die Anwendung des Ansatzes auf die Datenflusssprache von SCADE evaluiert. Ein Mutations-Mechanismus wurde implementiert, für die Testfallgenerierung wurde der zu SCADE verfügbare Model-Checker verwendet. Abgesehen von Limitierungen des Model-Checkers, konnte die Funktion zwar gezeigt werden - der Anwendungspartner hat jedoch seine Nutzung der Sprache im Zielumfeld zurückgefahren, der Toolhersteller war nicht interessiert, das Thema wurde aufgegeben.

Beide an TruFal beteiligten Industriepartner stellten fest, dass der Ansatz wert ist, weiterverfolgt zu werden, UML aber nicht der passende Modellierungsformalismus ist. AVL

[4] Trust via Failed Falsification of Complex Dependable Systems Using Automated Test Case Generation through Model Mutation, `https://trufal.wordpress.com/`, 2011-2014

[5] Combined Model-based Analysis and Testing of Embedded Systems,`http://www.mbat-artemis.eu/home/`,2011-2014

entwickelte in der Folge unter enger Einbindung der betroffenen Mitarbeiter eine eigene domänenspezifische Sprache (DSL), inklusive Editor-Unterstützung, Anbindung an die interne Geräte-Datenbank, Transformation nach OOAS und Transformation der Tests in das eigene nUnit-Framework. Diese Lösung ist nach etwas zusätzlicher Reifezeit mittlerweile im Produktiveinsatz [St18].

THALES Österreich ging in Richtung formaler Modellierung und begann Event-B und die zugehörige Umgebung Rodin[6] einzusetzen. Die Ziel-Anwendung (interne Projekte) änderte sich aus unternehmensinternen Gründen mehrfach. AIT entwickelte ein Plug-in für Rodin, das eine Untermenge von Event-B nach OOAS übersetzen kann. Rodin unterstützt (teil-)automatische Verifikation von Refinement, das heißt Modelle können, unter Erhalt von z.B. Safety-Zusicherungen, immer weiter detailliert werden, bis das Modell sehr nah an der Implementierung ist. Die aus dem letzten Refinement generierten Tests können dann verwendet werden, um den letzten Schritt zur Implementierung über Tests abzusichern.

5 Performance, die Zweite

Zur weiteren Verbesserung der Performance wurde in mehrere Richtungen gearbeitet. Rapidly exploring Random Trees (RRT) sind nun als Explorations-Strategie verfügbar [Fe19]. Will man (im Security-Umfeld) mittels Model-Checker Fragen zur Vertraulichkeit beantworten, lassen sich diese nicht als Aussagen über einzelne Traces ausdrücken, sondern nur als Aussagen über (mögliche) Paare von Traces, sogenannte Hyper-Properties. Das gilt auch für mutationsgetriebenes MBT, und ein verfügbarer symbolischer Model-Checker für Hyper-Properties wurde auf seine Anwendbarkeit für Mutation Testing untersucht [FBW19]. Die Verwendung von Partial Order Reduction (POR) für Testfallgenerierung ist komplexer als für Model Checking, da in der Exploration als identisch verworfene, alternative Interleavings für den Test rekonstruiert werden müssen. Die bisherige, naive Umsetzung von POR wurde überarbeitet und auf eine solide theoretische Basis gestellt.

6 Mehr als nur funktionales Testen

Initiiert von THALES Österreich und gemeinsam mit der Universität Southampton wurde ein Prozess erarbeitet, der Behavior Driven Development auf Modelle umlegt, die formale Modellierung unterstützt und über inkrementelle Testfallgenerierung einen indirekten Review des Modells durch den Review von Tests ermöglicht [Sn18]. Im Projekt EMBEET[7] wurde UML wieder aufgegriffen - zusammen mit zwei Partnern wurde eine integrierte Modellentwicklungsumgebung entwickelt. Sie unterstützt die Modellierung von Szenearien und Funktion, Rückverfolgbarkeit zu Anforderungen, inkrementelle Testfallgenerierung

[6] http://www.event-b.org/

[7] Environment for Model-Based Embedded Systems Engineering and Testing,http://embeet.com/, 2016-2019

und indirekten Modell-Review ebenso wie On-Target-Debugging auf Modellebene. Eine Produktveröffentlichung / erste Kundenprojekte werden 2020 erwartet.

Verfügbare funktionale Modelle können auch für andere Zwecke genutzt werden. Im neuesten MoMuT steht für Robustheits-Tests eine Smart-Fuzzing-Funktion zur Verfügung. Sie soll demnächst in Projekten zum Protokoll-Testen verwendet werden. Gemeinsam mit der TU Graz wurde ein Ansatz zum modellbasierten Performance-Test entwickelt. Ein erstelltes Verhaltensmodell und ein gelerntes Performance-Modell, das z.B. Reaktionszeiten vorhersagt, werden zur Ableitung von Performance-Aussagen und zur experimentellen, statistischen Prüfung der Vorhersagen verwendet [Ai19].

Im Projekt AQUAS[8] arbeitet AIT derzeit daran, die Ansätze für Performance-Tests, Robustheits- (und Security-)Tests und funktionale Tests zu verbinden. Beispiel ist eine von Integrasys entwickelte Infrastruktur zur Koordination zwischen Unmanned Aerial Vehicles (UAV).

7 Fortsetzung folgt ...

Die Erfahrungen der letzten Jahre auf dem Weg vom technischen Lösungsansatz zu industrieller Anwendung und wirtschaftlicher Verwertung lassen folgende Schlüsse zu:

Die hohe Spezialisierung auch in den Computerwissenschaften führt dazu, dass es schwierig ist, „Probleminhaber“ und die passenden Experten zusammenzuführen. Die Geschichte von MoMut ist doch auch die Geschichte einer Lösung auf der Suche nach einem Problem.

Mit einigen unserer Industriepartner hatten wir großes Glück, weil sie die Geduld hatten, auszuharren, weil sie Vertrauen in den Ansatz und AIT als Partner hatten. Viele begonnene Kooperationen sind trotzdem versandet, aus Gründen, die nichts mit der Anwendbarkeit der Lösung oder der Qualität der Zusammenarbeit zu tun hatten, sondern mit Firmenpolitik, internen Strategien, leeren Budget-Töpfen, oder Personalwechseln. Mitunter hatten einfach andere Probleme, als die mit automatisierter Testfallgenerierung zu lösenden, Vorrang.

Bewährt hat sich, für die Testfallgeneratoren mit OOAS[9] auf eine Zwischensprache unter eigener Kontrolle zu setzen und über Transformationen die benutzerseitigen Modellierungssprachen einzubinden. Das hat spätere Erweiterungen wesentlich vereinfacht.

Die Einführung neuer Lösungen in Unternehmen ist ein komplexes Unterfangen. Ein gelungenes Beispiel dafür ist die Einführung modellbasierter Ansätze und einer DSL bei AVL [St18]. Eine Rückmeldung aus der Industrie ist, dass die Forschungspartner auf fundamental neuen Ansätzen basierende Lösungen (Revolutionen) anbieten, die Industrie aber Evolution mit kalkulierbaren Risiken braucht.

[8] Aggregated Quality Assurance for Systems (AQUAS),http://aquas-project.eu/, 2017-2020

[9] OOAS wurde ursprünglich von der TU Graz entwickelt, Sprachdefinition und Parser-Quellen sind unter Open-Source-Lizenz verfügbar. https://www.momut.org/code/projects/ooastools/repository

Die Förderlandschaft fordert auch für anwendungsorientierte Forschung einen hohen Neuheitsgrad - das führt fast zwingend zu den genannten Revolutionen, während Forschungsgruppen immer neue Themen anreißen, um den neuen Ausschreibungsschwerpunkten zu folgen. Es bleibt eine große Lücke zur Produktreife - Benutzbarkeit und Anwenderakzeptanz, aber auch Dokumentation, finden hier keinen Platz. Auch nützliche Technologien sind oft nicht attraktiv genug für einen Investor, um das Schließen dieser Lücke zu finanzieren.

Im Projekt EMBEET war die Förderschiene EUROSTARS sehr hilfreich, deutlich näher an ein markttaugliches Produkt zu kommen.

Die Bewegung vom Einzelwerkzeug für funktionale Testfallgenerierung zu einerseits der Integration in Entwicklungsumgebungen und Gesamtlösungen (UML wie DSL) und die Ausweitung zum Test nicht-funktionaler Anforderungen unter synergetischer Nutzung von Modellierungsaufwänden zeigt eine Entwicklung von Technologieorientierung hin zur Nutzenorientierung. Es ist zu hoffen, dass dieser Lernschritt weiterhin durch zunehmenden produktiven Einsatz der Werkzeuge belohnt wird.

Literatur

[AD06] Aichernig, B. K.; Delgado, C. C.: From Faults Via Test Purposes to Test Cases: On the Fault-Based Testing of Concurrent Systems. In (Baresi, L.; Heckel, R., Hrsg.): FASE 2006. S. 324–338, 2006.

[Ai11] Aichernig, B. K.; Brandl, H.; Jöbstl, E.; Krenn, W.: Efficient Mutation Killers in Action. In: ICST 2011. S. 120–129, 2011.

[Ai14] Aichernig, B. K.; Auer, J.; Jöbstl, E.; Korosec, R.; Krenn, W.; Schlick, R.; Schmidt, B. V.: Model-Based Mutation Testing of an Industrial Measurement Device. In: TAP 2014. S. 1–19, 2014.

[Ai19] Aichernig, B. K.; Bauerstätter, P.; Jöbstl, E.; Kann, S.; Korosec, R.; Krenn, W.; Mateis, C.; Schlick, R.; Schumi, R.: Learning and statistical model checking of system response times. Software Quality Journal 27/2, S. 757–795, 2019.

[FBW19] Fellner, A.; Befrouei, M. T.; Weissenbacher, G.: Mutation Testing with Hyperproperties. In: SEFM 2019. S. 203–221, 2019.

[Fe19] Fellner, A.; Krenn, W.; Schlick, R.; Tarrach, T.; Weissenbacher, G.: Model-based, Mutation-driven Test-case Generation Via Heuristic-guided Branching Search. ACM Trans. Embed. Comput. Syst. 18/1, 4:1–4:28, Jan. 2019.

[Sn18] Snook, C. F.; Hoang, T. S.; Dghaym, D.; Butler, M. J.; Fischer, T.; Schlick, R.; Wang, K.: Behaviour-Driven Formal Model Development. In: ICFEM 2018. S. 21–36, 2018.

[St18] Stieglbauer, G.; Burghard, C.; Sobernig, S.; Korosec, R.: A Daily Dose of DSL - MDE Micro Injections in Practice. In: MODELSWARD 2018. S. 642–651, 2018.