Generierung von Webanwendungstestskripten aus einem textbasierten Modell

Arne-Michael Törsel
Arne-Michael.Toersel@fh-stralsund.de

Gerold Blakowski
Gerold.Blakowski@fh-stralsund.de

Zusammenfassung Modellbasiertes Testen in der Domäne der Webanwendungen nutzbar zu machen, ist ein aktuelles Forschungsthema. Der Beitrag stellt ein System vor, das insbesondere die Übertragung von generierten, abstrakten Testfällen in ausführbare Testskripte für ein Testautomatisierungswerkzeug ohne zusätzlichen manuellen Aufwand demonstriert. Dazu wird ein textbasierter Modelltyp verwendet, der es ermöglicht, automatisiert mittels eines Generators ausführbare Testskripte mit einfachen, auf Textmatching basierenden Testorakeln zu generieren. Das Verfahren existiert derzeit als Prototyp auf Basis der openArchitectureWare-Plattform [1].

Motivation Zur Anwendung modellbasierter Testverfahren in der Domäne des Blackboxtests von Webanwendungen gibt es bereits mehrere Forschungsansätze. In (1) wurden hierarchische Finite State Machines zur Modellierung der Anwendung genutzt. Der in (2) vorgeschlagene Ansatz nutzt gerichtete Graphen (Event-Sequence-Graph). Die Modelle dienen in beiden Ansätzen als Grundlage zur systematischen, kriteriengesteuerten Testfallableitung mittels eines Algorithmus. Es entstehen abstrakte Testfälle die manuell ausgeführt werden können oder als automatisierte Testskripte durch den Tester implementiert werden müssen. Dabei ist auch ein Testorakel durch den Tester zu implementieren.

Beim Einsatz solcher modellbasierter Testverfahren entsteht zunächst zusätzlicher Aufwand zur Erstellung und Pflege des redundanten Modells der Anwendung. Sollen Testfälle zur mehrfachen Ausführung automatisiert werden, ist weiterer Aufwand erforderlich, um die aus dem Modell abgeleiteten abstrakten Testfälle zu konkretisieren und im Format des jeweiligen Testautomatisierungswerkzeugs zu implementieren. Ein Einsatz eines solchen Testverfahrens ist nur dann gerechtfertigt, wenn Vorteile gegenüber einer manuellen Testfallerstellung entweder durch die gewonnene Systematik oder durch einen geringeren Aufwand bestehen. Speziell in Szenarien in denen häufige Anpassungen zur Wartung am Modell anfallen, besteht also ein Bedarf an einer hohen Effizienz bei der Testfallableitung und -konkretisierung um die Praxistauglichkeit modellbasierter Testverfahren für den Webanwendungstest zu verbessern. Zur Effizienzsteigerung soll der Ablauf von der Testfallerzeugung bis hin zur Konkretisierung der abgeleiteten Testfälle mit dem vorgestellten Ansatz deshalb weitestgehend automatisiert werden.

[1] http://www.openarchitectureware.org

Anforderungen Testskripte für Testautomatisierungswerkzeuge auf der Nutzeroberfläche bestehen typischerweise aus Anweisungen die die Interaktion eines Nutzers mit der Anwendung simulieren. Dazu stehen Interaktionsweisungen - zum Beispiel zum Tätigen einer Texteingabe - und Prüfanweisungen, die die Reaktion der Anwendung auf Nutzereingaben mit einem erwarteten Ergebnis vergleichen, zur Verfügung. Um eine automatisierte Erzeugung ausführbarer Testfälle zu ermöglichen, müssen aus dem Modell sowohl Interaktionsanweisungen als auch Prüfanweisungen erzeugt werden können. Dazu werden Informationen zur Anwendungsstruktur als auch zum erwarteten "Verhalten" der Anwendung benötigt. Beim Blackbox-Testen von Webanwendungen beschränkt sich das beobachtbare Verhalten der Anwendung auf die Ausgaben im Browser des Anwenders. Testautomatisierungswerkzeuge arbeiten deshalb auf der Ebene des von der Webanwendung gelieferten HTML-Quellcodes und suchen zur Prüfung des Anwendungsverhaltens nach erwarteten Textfragmenten. Oft ist dabei die Testauswirkung nicht unmittelbar beobachtbar, sondern manifestiert sich erst zu einem späteren Zeitpunkt in anderen Anwendungsfällen. Sollen konkrete Testfälle aus dem Modell abgeleitet werden, muss der Generatoralgorithmus entsprechende anwendungsfallübergreifende Testorakel erzeugen können.

Das Modell Das vorgestellte System basiert auf einem textbasierten Modelltyp (das Metamodell findet sich im Anhang zu (3)), weil textbasierte Modelle mit geeigneten Editoren durch geübte Anwender in der Regel effektiver bearbeitbar sind als grafische Modelle. Mittels Direktiven werden die Elemente der Anwendung deklariert. Das Modell impliziert einen gerichteten Graphen, der die Struktur und den Kontrollfluss der Anwendung widerspiegelt. Gibt es einen durch den Nutzer verursachten Übergang zu einer anderen Sicht, entspricht dies einer gerichteten Kante im Graphen von der Sicht, auf der die Interaktion des Nutzers stattfindet, zur Zielsicht. Solch ein Übergang wird durch den Nutzer einer Webanwendung typi-

scherweise durch das Betätigen eines Links oder das Absenden eines Formulars ausgelöst.

Das Modell integriert Variablen mit dauerhafter Gültigkeit beziehungsweise einer Gültigkeit über eine simulierte Sitzung hinweg. Variablen mit Sitzungsgültigkeit erhalten mit Beginn einer neuen Sitzung erneut den deklarierten Startwert. Die Variablen werden im Modell zur Abbildung des Anwendungszustands genutzt. Zur Steuerung der Testfallgenerierung dienen logische Ausdrücke auf den Variablen die als "Guards" für die Übergänge wirken. An Übergängen können im Modell Zuweisungen an Variablen zum Ausdruck einer Zustandsänderung bei Ausführung des Übergangs vorgenommen werden. Testdaten werden durch einen eigenen Tupeldatentyp in das Modell integriert und werden als externe Ressource bereitgestellt.

Um bei der Testfallgenerierung aus dem Modell Prüfanweisungen erzeugen zu können, ist es möglich, in den Sichten des Modells erwartete Textausdrücke als Testorakel zu deklarieren. Die Textausdrücke sind dynamisch, das heißt, sie können Variablenwerte und Testdaten integrieren und werden bei der Testfallgenerierung aufgelöst. Ähnlich den Guards an Übergängen kann ein Textausdruck durch einen assoziierten Bedingungsausdruck deaktiviert werden, wenn der Textausdruck nur unter bestimmten Bedingungen in der Oberfläche sichtbar ist.

Prototyp Ein einfacher Testfallgeneratoralgorithmus wurde umgesetzt, der eine Abdeckung aller Sichten und eingabenausgelöster Übergänge durch Testsequenzen anstrebt. Dazu wird der Anwendungsgraph systematisch traversiert. Eingaben für den Generator sind das Modell und Testdaten. Die erzeugten Testfälle bauen aufeinander auf, das heißt, Vorbedingungen für spätere Testfälle werden durch die Ausführung früher Testfälle erfüllt. Dies hat den Vorteil, dass kein separater, vorfabrizierter Zustand für jeden Testfall notwendig ist. Als Nachteil ist festzuhalten, dass durch den Verlust der Unabhängigkeit der Testfälle das Risiko steigt, dass Fehler nicht bemerkt werden beziehungsweise die Identifikation der Fehlerursache erschwert wird. Die Ausführungsreihenfolge der Testfälle wird durch den Generatoralgorithmus durch Auswertung der Guardausdrücke an den Übergängen und Sortierung sowie durch Permutation/Simulation der "Kandidaten" ermittelt. Der Generator erzeugt zunächst Instanzen eines abstrakten Testfallmodells, die dann für das jeweilige Testwerkzeug konkretisiert werden.

Mit Hilfe der openArchitectureWare-Plattform, kurz "oAW", wurde ein Prototyp für die Werkzeugkette in der Programmiersprache Java erstellt. Aus dem Metamodell erzeugt oAW einen Java-Parser und einen Eclipse-basierten Modelleditor. Eine grafische Darstellung und Bearbeitung ist prinzipiell ebenfalls möglich. Sichten der Webanwendung entsprechen Knoten im implizierten Graph. Die Variablenverwaltung und die Auswertung von Ausdrücken erfolgt intern durch die Javascript-Engine "Rhino" [2]. Die Zielplattform für den Testskripterstellung ist das Open Source Werkzeug Canoo Webtest [3]. Durch die Abstraktion der Testfälle durch ein Testfallmodell können Konkretisierungen in Form von Testskripten auch für weitere Webanwendungstestwerkzeuge umgesetzt werden. Dazu ist ein entsprechendes Template für das oAW-Werkzeug "Xpand" zu implementieren, das die Transformation steuert.

Testdaten und implementationsspezifische Informationen für die Transformation zu konkreten Testfällen (zum Beispiel Eingabeelementbezeichner) liegen in externen Ressourcendateien. Die automatisierte Konkretisierung der abstrakten Testfälle stellt Anforderungen an die Testbarkeit der Anwendung. Es ist erforderlich, dass alle Steuerelemente und Links durch eindeutige Bezeichner identifizierbar sind.

Ausblick Nach dem gegenwärtigen Stand können Testsuiten für den automatisierten Smoketest aus dem Modell der Anwendung erzeugt werden. Die vollständige Automatisierung der Testfallerzeugung ist vor allem bei einer häufigen Wartung des Modells, zum Beispiel zur Integration geänderter Anforderungen, vorteilhaft. Das System wird fortlaufend weiterentwickelt. Ein Ziel ist dabei die Erhöhung der Testtiefe um Testsuiten für den Regressionstest erzeugen zu können. Dazu sind Erweiterungen am Metamodell und Generator notwendig, um zum Beispiel automatisiert Negativtests erzeugen zu können. Weiterhin soll die Komposition von mehreren Teilmodellen zu einem komplexen Anwendungsmodell zur Verbesserung der Handhabbarkeit ermöglicht werden.

Referenzen

[1] Andrews, Anneliese A. ; Offutt, Jeff ; Alexander, Roger T.: Testing Web applications by modeling with FSMs. In: *Software and System Modeling* 4 (2005), Nr. 3, S. 326–345

[2] Belli, Fevzi ; Linschulte, Michael ; Zirnsak, Ralf ; Hofmann, Günter: 'Negativ'-Tests interaktiver Systeme und ihre Automatisierung. In: Bleek, Wolf-Gideon (Hrsg.) ; Schwentner, Henning (Hrsg.) ; Züllighoven, Heinz (Hrsg.): *Software Engineering (Workshops)* Bd. 106, GI, 2007 (LNI). – ISBN 978–3–88579–200–0, S. 35–44

[3] Törsel, Arne-Michael ; Blakowski, Gerold: Automatisierte Erzeugung konkreter Testfälle für Webanwendungen aus einem textbasierten Modell. In: *INFORMATIK 2009, Im Focus das Leben, Beiträge der 39. Jahrestagung der Gesellschaft für Informatik e.V* (2009)

[2] http://www.mozilla.org/rhino
[3] http://webtest.canoo.com