Replay-basiertes Testen verteilter Anwendungen

Beteiligte an der Hochschule

- Prof. Dr. Reinhold Kröger
- Dipl.-Inform. (FH) Holger Machens

Kooperationspartner

- Siemens AG CT, München

Laufzeit

Beginn: 01.12.2004
Ende: 30.10.2009

Finanzierung

- 100% Siemens AG CT

Veröffentlichungen

- Kröger, Reinhold; Machens, Holger: "Replay von nebenläufigen Anwendungen", Technischer Bericht, Fachhochschule Wiesbaden, Fachbereich Informatik, Januar 2006

- Kröger, Reinhold; Machens, Holger: "MOST Log Parser für Eclipse/TPTP", Technischer Bericht, Fachhochschule Wiesbaden, Fachbereich Design Informatik Medien (DCSM), Oktober 2006

- Machens, Holger; Kröger, Reinhold: "MOST QA Tools - Quality Assurance für MOST-Bus-Applikationen im TPTP Test Framework auf Eclipse", Labor für Verteilte Systeme, Fachhochschule Wiesbaden, Juli 2007

- H. Machens; R. Kröger; S. Jell; K. Grabenweger: "Replay-basiertes Testen von MOST-Bus-Anwendungen im Automotive-Umfeld", Gesellschaft für Informatik, Lecture Notes in Informatics, Bonn, Software Engineering 2008, Seite 85, Vol. 91, Lecture Notes in Informatics, ISBN: 978-3-88579-215-4, Februar 2008

Kurzbeschreibung

Das Testen ist seit jeher ein fester Bestandteil der Softwareentwicklung.

Zu den Testzyklen, die der Entwickler heute meist unmittelbar nach Fertigstellung einzelner Code-Fragmente manuell oder automatisch durchführt, wurden insbesondere für komplexere Anwendungen entsprechend aufwendige Testverfahren entwickelt, um das Zusammenspiel der verschiedenen Softwarekomponenten besser beurteilen zu können. Hauptsächliches Merkmal von Testkomponenten im Allgemeinen ist das Erzeugen von Stimuli am zu testenden System (System Under Test, SUT), wie z.B. das Drücken einer Schaltfläche auf einer grafischen Oberfläche oder Methodenaufrufe an Komponenten der Anwendung. Durch das automatisierte Erzeugen einer Folge von Stimuli will man das SUT dazu bringen, bestimmte Testfälle zu durchlaufen und das Verhalten in den verschiedenen Testfällen ebenso automatisiert prüfen zu können.

Unterstützt wird dieses Verfahren durch Aufnahme-Werkzeuge (Capture oder Recording Tools), die zur Aufnahme der Stimuli während der Eingabe durch den Tester dienen. Die Aufnahme dient dann als Vorlage für die Testkomponenten, von denen sie wieder abgespielt werden. Solche Tools sind z.B. aus dem Bereich GUI Testing (z.B. Abbot) und Web-Testing bekannt.

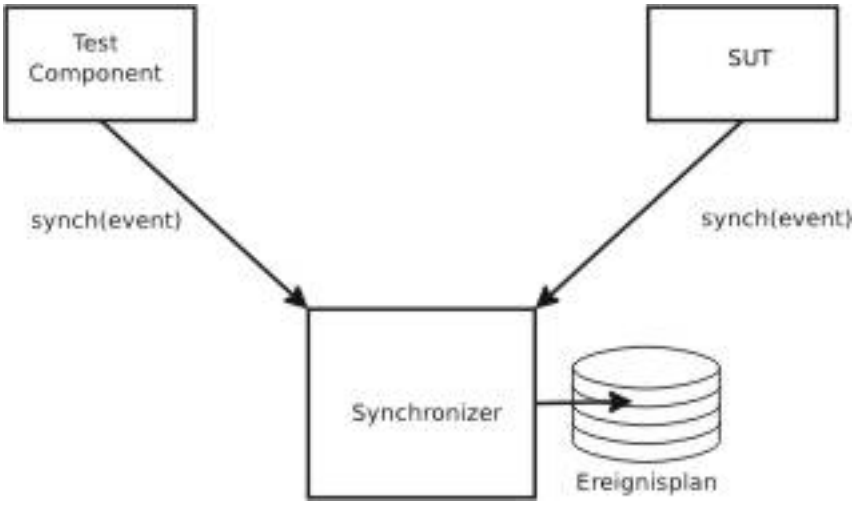
Das Replay von Aufnahmen aus nebenläufigen Anwendungen wirft eine Reihe spezifischer Probleme auf, die in dieser Arbeit eine besondere Beachtung finden. Für sequenzielle Anwendungen bestehen diese Probleme nicht bzw. sie bilden in Bezug auf die Replay-Fähigkeit nur eine Untergruppe der nebenläufigen Anwendungen und werden hier nicht weiter betrachtet.

In dem Projekt wurde ein Konzept für ein Testsystem erarbeitet, das ein Replay in verteilten, nebenläufigen Anwendungen erlaubt. Das erarbeitete Konzept wurde bereits für CORBA-Anwendungen realisiert, wobei auch die Integration in die Entwicklungsumgebung
http://www.eclipse.org/
berücksichtigt wurde, basierend auf der Testing and Profiling Tools Platform (
http://www.eclipse.org/tptp
, ehemals bekannt unter dem Namen Hyades) als Automated Software Quality Framework (ASQ).

Das Vorgehen des Testers orientiert sich an dem Prinzip des ASQ, wie es auch von TPTP vorgestellt wird:

- Aufnahme: Das Durchführen eines exemplarischen Laufs zur Erfassung des Verhaltens der Anwendung bildet die Vorbereitung eines Tests. Das Ergebnis dieses Schritts ist die Aufnahme (auch Record oder Trace), die die Basis für ein späteres Abspielen (Replay) darstellt.
- Import: Das Einlesen der Aufnahme dient zum einen zur Überführung in ein editierbares Format, und zum anderen kann in diesem Schritt das im Trace meist unvollständige Wissen über das Verhalten der Anwendung manuell ergänzt werden. Hier könnte alternativ auch ein UML-Modell oder ähnliches als Quelle für den Import dienen.

- Editieren: Die importierten Stimuli können nachträglich verändert werden, um Spezialfälle herbeizuführen. Editieren kann aber auch als Alternative für das Recording dienen, wenn der Tester den Testfall manuell eingeben will.
- Testgenerierung: Das Replay wird von einem Testsystem durchgeführt. Dieses Testsystem kann entweder die Aufnahme als Input bekommen und wie ein Skript dynamisch interpretieren oder die Aufnahme hart kodiert enthalten. Für die letztere Variante wird im allgemeinen ein Generator eingesetzt, der die Testkomponenten des Testsystems generiert.
- Replay: Durch das Replay wird die Folge der im Test enthaltenen Stimuli wieder abgespielt. Während des Replays werden die Reaktionen des SUT beobachtet und verifiziert. Das Ergebnis des Testlaufs ist ein Testurteil.



In der Abbildung ist schematisch die Architektur für ein Replay System dargestellt. Darin sind symbolisch die folgenden Komponenten enthalten:

- Die Testkomponenten, die ein oder mehrere Komponenten der Anwendung simulieren.
- Das System under Test (SUT), das sich aus einer oder mehreren zu testenden Komponenten der Anwendungen zusammensetzt.
- Ein Synchronizer als zentrale Komponente, die zur Synchronisation des Auftretens der Ereignisse im gesamten System (bestehend aus SUT und Testkomponenten) entsprechend dem konservierten Verhalten aus der Aufnahme, das in gefilterter und aufbereiteter Form als Ereignisplan bereitgestellt wird.

SUT und Testkomponenten werden für das Replay-basierte Testen um zwei Funktionalitäten ergänzt. Die auftretenden Ereignisse werden auch während des Testlaufs, also während des Replays, protokolliert. Desweiteren werden die auftretenden Ereignisse synchronisiert mit dem geplanten Ablauf im Ereignisplan, was durch synchrone Methodenaufrufen am Synchronizer realisiert ist.

Basierend auf einer eingehenden Analyse des Themas unter Einbeziehung vorhandener Arbeiten aus den Bereichen Replay und

Testing im Sinne von Automated Software Quality wurde ein allgemeines Konzept für ein entsprechendes Rahmenwerk erstellt, das beide Themengebiete umfasst. Hierbei wurde auch der gängige Standard UML2TP für die Modellierung von Tests und Testumgebungen berücksichtigt. Wichtiges Ergebnis der Analyse war u.a. die Sammlung von Anwendungsfällen für das Replay-basierte, automatisierte Testen in verteilten Anwendungen.

Besonderheit des Konzepts ist die ausführliche Betrachtung von Ereignisplänen für das Replay und deren verschiedene Ausprägungen in realen Anwendungen. Hier wurden maßgebliche Klassifizierungen vorgenommen, die dem Kommunizieren und Entwickeln von Replay-basierten Systemen eine professionelle Grundlage verleihen.

Der "Proof of Concept" wurde durch die Realisierung des Frameworks in einer CORBA-Umgebung auf Basis des Open Source Testing und Tracing Frameworks TPTP durchgeführt. Hier wurde in einer 1. Version ein global lineare Ordnung der Ereignisse während des Replays erzwungen, was bedeutet, dass alle Ereignisse nach ihrem Zeitstempel geordnet wieder auftreten.

In einem anschließenden Projekt wurde in Zusammenarbeit mit der VDO Automative AG als assoziierten Partner an einer Verwendung der entwickelten Konzepte für das Testen von MOST-Bus-Anwendungen (Media Oriented Systems Transport, siehe
http://www.mostcooperation.com/
) in der Automotive Domäne gearbeitet.
Der MOST-Bus ist ein Infotainment-Bus, der hohe Datenraten für den Transport von Multimedia-Daten im Fahrzeug bereitstellt, also beispielsweise für die Übertragung des Bildes einer Heckkamera auf ein Display im Fahrzeug. Die Kommunikation in einer MOST-Applikation weist einen hohen Anteil an zyklisch gesendeten Statusmeldungen auf, wie es beispielsweise auch in Feldbussystemen der Automatisierungstechnik der Fall ist. Dieser zyklische Nachrichtenverkehr ist auch im ``Ruhezustand'' des Systems als ständiges Hintergrundrauschen vorhanden und überlagert auch den eigentlichen Kontrollfluss der Anwendung, also solche Nachrichten, die einem bestimmten Use Case zuzuordnen wären.

Das Konzept für das Replay von klassischen, RPC-basierten Anwendungen, wie es für CORBA realisiert wurde, war wegen des nebenläufigen Hintergrundrauschens in MOST-Applikationen nicht ausreichend. Eine Berücksichtigung des Hintergrundrauschens als nebenläufiges Kommmunikationsverhalten war allerdings inakzeptabel, da es zu komplizierten Testbeschreibungen führen würde, die vom Benutzer nur noch sehr schwer zu handhaben wären. Die Anzahl der Nachrichten, die einem zu untersuchenden Anwendungsfall zuzuordnen sind, ist gegen das Hintergrundrauschen sehr gering und damit für den Benutzer leicht handhabbar.

Das Replay-Verfahren wurde daher durch zwei entscheidende Komponenten ergänzt.

- Die für den Anwendungsfall relevante Sequenz von Nachrichten wird händisch selektiert und in ein "Validierungsmuster" übernommen. Dieses Validierungsmuster dient während des Replays zur Überprüfung der Reaktionen des SUTs (Antwortnachrichten auf Stimuli).
- Das Hintergrundrauschen wird durch einen Rauschfilter ausgeblendet. Er wird gewöhnlich mit den Nachrichten gefüllt, die in der Aufnahme als Hintergrundrauschen enthalten sind und nicht dem Validierungsmuster zuzuordnen sind.

Zur Evaluierung wurde das gesamte Verfahren als Plugin-Sammlung auf Eclipse/TPTP realisiert und in einer Experimentalumgebung bei VDO eingesetzt.

Bei dem aktuellen Vorhaben im Rahmen des fortgesetzten Kooperationsprojektes steht nun die Untersuchung Performance-orientierter Test- und Analyse-Techniken basierend auf dem Replay von Kommunikations-Traces im Vordergrund. Als Anwendungsdomäne wurden hierfür moderne Service-Orientierte Architekturen auf Web-Services-Technologien gewählt. Im ersten Schritt wurde hier auf Basis von Eclipse/TPTP eine einfache Testumgebung für Web Services realisiert, die eine modellgetriebene Testentwicklung halbautomatisch unterstützt. Im weiteren Verlauf wird der Fokus stärker auf Workflow-orientierte Anwendung verlagert. Ziel ist hierbei die bestehenden Forschungsansätz zur automatischen modellgetriebenen Testentwicklung für Workflows mit dem Replay-basierten Regressionstesten zu verschmelzen und eine Bewertungen hinsichtlich verschiedenere Testabdeckungsgrade zu ermöglichen.