Ein Werkzeugsystem zur Automatisierung von GUI-Tests

Jens Hanisch, Johann Letzel, Klaus Bothe
Humboldt-Universität zu Berlin
Institut für Informatik
bothe@informatik.hu-berlin.de

1 Überblick

1.1 Problemstellung

In einem mehrjährigen studentischen Reengineering-Projekt [1] wird ein reales sicherheitskritisches Softwaresystem in der Experimentalphysik bearbeitet, an dem regelmäßig zwischen 8 bis 12 Studierende teilnehmen. Dabei werden Modifikationen des Systems ausgeführt, die u. a. eine Fehlerbeseitigung, ein Rearchitecturing (u. a. ein Refactoring), Modifikationen des Systems ebenso einschließen wie Erweiterungen der Funktionalität des Systems. Da in mehreren Gruppen parallel gearbeitet wird, ist ein regelmäßiger Test unerlässlich, da hohe Qualitätsanforderungen an die sich im realen Einsatz im Institut für Physik der Humboldt-Universität zu Berlin befindliche Software gestellt werden.

Da eine vielfältige Funktionalität im Test abzudecken ist und die Testaktivitäten zeitaufwendig sind, musste eine weitgehende Automatisierung des Regressionstests in all seinen Phasen erreicht werden.

Der Betrieb des Anwendersystems erfolgt weitestgehend oberflächenorientiert, d. h. durch Nutzereingaben gesteuert. Obwohl für diese Anwendungsklasse kommerzielle Tools verfügbar sind [3], wurde eine Eigenentwicklung vorgenommen. Damit waren mehrere Vorteile verbunden: auf die Anwendung zugeschnitte Lösung, Verfügbarkeit der Quellen ermöglicht weiterführende Arbeiten, kostengünstig.

Trotz allem ist das von uns entwickelte Tool auch für andere oberflächenbasierte Anwendungen nutzbar, die in C++ entwickelt worden sind und unter Windows 9x, Windows ME / NT / 200 / XP laufen.

Dieser Artikel führt in die Grundzüge des Systems ein - Eine nähere Beschreibung findet sich in [2].

1.2 Vorgehen

Testfallentwurf

Im Laufe des Projektes sind zu den herausgearbeiteten Anwendungsfällen (UseCases) zahlreiche Dokumente zur Beschreibung der Programmfunktionen entstanden. Aus diesen Spezifikationsdokumenten heraus werden Testsequenzen entworfen, die typischen Arbeitsschritten eines potentiellen Benutzers entsprechen und bei jedem Regressionstest durchgeführt werden müssen. Als *Blackbox*-orientiertes Testverfahren wird das Testobjekt dabei über seine grafische Benutzerschnittstelle (GUI) angesprochen.

Automatisierung

Zur Abarbeitung einer Testsequenz über die GUI der Zielanwendung empfiehlt sich die Entwicklung eines eigenständigen (autarken) Testtreibers. Obwohl sich die Kommunikation zwischen dem Testtreiber und der Zielanwendung schwierig gestaltet, erhält man ein flexibles Testsystem, das auch auf andere Testobjekte mit grafischer Benutzerschnittstelle anwendbar ist.

Grundlage für eine Automatisierung bilden Dateien (Testskripte) mit Anweisungen zur Ausführung von Aktionen auf der GUI und zum Auslesen von Zuständen der GUI-Objekte wie Schaltknöpfe oder Eingabefelder.

Im Gegensatz zu vielen kommerziellen Testwerkzeugen [3] wurde kein *Capture&Replay*-Verfahren zur Aufzeichnung und Wiedergabe von Aktionen auf der Benutzeroberfläche mittels eines Makrorekorders implementiert. Da alle Testsequenzen bzw. Testskripte manuell entworfen werden müssen, wurde bei der Spezifikation einer geeigneten Skriptsprache besonderer Wert auf eine benutzerfreundliche Syntax gelegt.

Technologie

Zur Realisierung eines autarken Testtreibers wird das Nachrichtenkonzept des Betriebssystems Windows ausgenutzt. Jedes GUI-Objekt ist über Funktionen aus der Windows-API ansprechbar, sofern sein „Handle“ innerhalb der Windowsanwendung aus einer eindeutigen numerischen ID gewonnen werden kann. Die Zuordnung zwischen numerischen IDs und GUI-Objekten findet bei C++ Projekten in Ressourcen- und Header-Dateien statt.

Über dieselben Dateien werden häufig statische Menüs einer Anwendung definiert. Die Dateien zur Beschreibung einer Benutzerschnittstelle können hinsichtlich ihrer Syntax bei verschiedenen Entwicklungsumgebungen (Borland, Microsoft) voneinander abweichen. Mit der Komponente *RCParser* des Testsystems wird die Verwaltung von allen programminternen IDs vorgenommen. Der Entwickler von Testskripten kann zur Konstruktion von Kommandos umgängliche Bezeichnungen für GUI-Objekte

benutzen. Jedes Skriptkommando mit einer Aktion auf einem GUI-Objekt bildet ein logisches Ereignis, womit ein Testlauf vollkommen unabhängig von der Positionierung der GUI-Objekte ist.

2 Das Testsystem

2.1 Arbeitsweise

Abbildung 1 zeigt das Zusammenspiel zwischen der Testsuite *ATOS* (Automatisierter Test oberflächenbasierter Softwaresysteme) und ihrer Umgebung. Die Testsuite interpretiert Kommandos aus den Skriptdateien und bringt sie auf dem Testobjekt zur Ausführung. Jeder Testschritt wird dabei protokolliert. Wenn aufgrund eines Fehlers die Durchführung des aktuellen Skriptes abgebrochen werden muss, wird die Arbeit mit dem nächsten Testskript fortgesetzt.

ATOS ist in der Lage, durch ein entsprechendes Skriptkommando beliebige Programme zur Zeit des Testlaufs aufzurufen. Der Vergleich von Ausgabedateien der Steuersoftware mit Referenzdateien wird beispielsweise durch das Werkzeug *DataDiff* realisiert, welches im Rahmen der Diplomarbeit für diese Zwecke entworfen wurde.

Grundlage für die Skriptausführung bildet die vom *RCParser* verwaltete Datenbank mit der Zuordnung zwischen handlichen Bezeichnern und der zugrundeliegenden Benutzerschnittstelle des Testobjekts.

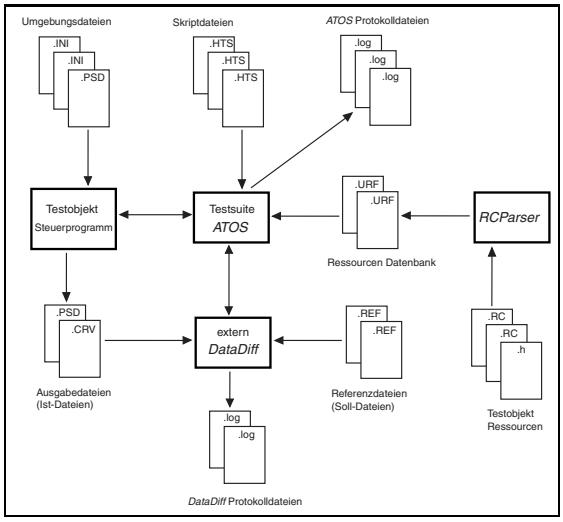

Abb. 1: Arbeitsweise des Testsystems

2.2 GUI-Verwaltung im *RCParser*

Der *RCParser* ist für die Zuordnung zwischen programminternen IDs und handlichen Bezeichnern für GUI-Objekte verantwortlich. Das Werkzeug hat sich zur Lösung der gestellten Aufgabe als besonders vorteilhaft herausgestellt. Im Zuge des Reengineering-Prozesses unterliegt die Benutzerschnittstelle ständigen Veränderungen. Weil der *RCParser* eine zugrundeliegende Datenbank mit den Ressourcen- und Header-Dateien eines Projektes der zu testenden Anwendung im konsistenten Zustand hält, kann die Lauffähigkeit der entworfenen Testskripte sichergestellt werden.

Das Werkzeug versucht soweit wie möglich aus den Projektdateien der Zielanwendung Informationen über die grafische Benutzerschnittstelle einzulesen. Einigen GUI-Objekten wird in diesem Zuge automatisch ein sinnvoller Bezeichner zugeordnet. Dialogboxen und Menüstrukturen, die erst zur Laufzeit der getesteten Anwendung dynamisch aufgebaut werden, können vom *RCParser* nicht erkannt werden und müssen der Datenbank manuell hinzugefügt werden. Neben beliebigen Menüs, Fenstern und Dialogboxen ist die Einrichtung von häufig benutzten standardisierten Windows-Dialogboxen (Öffnen und Schließen von Dateien) aus Vorlagen („Templates“) möglich.

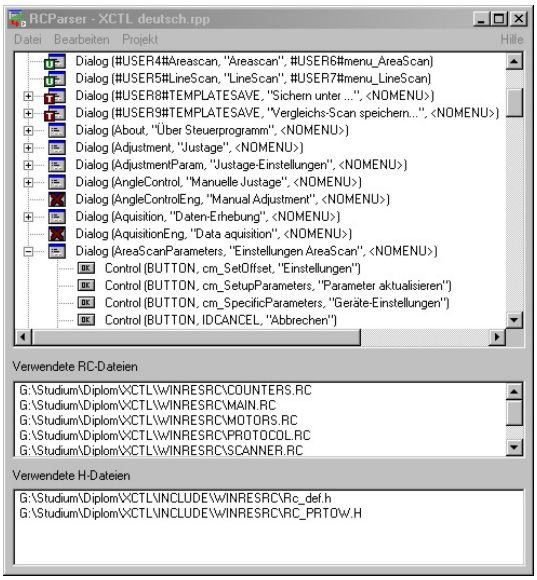

Abb. 2: Der Ressourcen-Parser *RCParser*

Abbildung 2 zeigt die Anwendung des *RCParsers* auf das physikalische Steuerprogramm. Eine Portierung von Anwendersoftware von einer Borland- auf eine Microsoft-Entwicklungsumgebung wird zu keinen Problemen führen, da der syntaktische Aufbau der Projekt-Ressourcen über *Regeldateien* beschrieben wird (siehe Abbildung 3). Solche Dateien können für beliebige Entwicklungsumgebungen entworfen werden und dem *RCParser* beim Einlesen von Projekt-Ressourcen behilflich sein.

2.3 Die Testsuite *ATOS*

Abbildung 4 stellt die wichtigste Komponente des entwickelten Werkzeugsystems dar. Mit der Konstruktion, Auswahl, Ausführung und Auswertung von Test-

```
DESCRIPTION "Borland C/C++"
BEGINSTATE  "NORMAL"                  # Anfanszustand des Matchers

...

RULE
   STATE    "NORMAL"
   PATTERN  <SAVE:ID>,"DIALOG",<CONTINUE>
   OUTPUT   <NONE>
   NEWSTATE "DIALOGBEGIN"
ENDRULE

RULE
   STATE    "DIALOGBEGIN"
   PATTERN  "CAPTION",<SAVE:CAPTION>
   OUTPUT   <NONE>
   NEWSTATE "DIALOGHEAD"
ENDRULE

RULE
   STATE    "DIALOGBEGIN"
   PATTERN  "{"
   OUTPUT   <NONE>
   NEWSTATE "NORMAL"
ENDRULE

...

RULE
   STATE    "DIALOGBLOCK"
   PATTERN  "CONTROL",<SAVE:CAPTION>,<SAVE:ID>,"\"BorBtn\"",<CONTINUE>
   OUTPUT   "CONTROL","BUTTON",<LOAD:ID>,<LOAD:CAPTION>
   NEWSTATE "DIALOGBLOCK"
ENDRULE

RULE
   STATE    "DIALOGBLOCK"
   PATTERN  "CONTROL",<SAVE:CAPTION>,<SAVE:ID>,"\"BUTTON\"",<CONTINUE>
   OUTPUT   "CONTROL","BUTTON",<LOAD:ID>,<LOAD:CAPTION>
   NEWSTATE "DIALOGBLOCK"
ENDRULE

...
```

Abb. 3: Regeldatei `borland.rul`

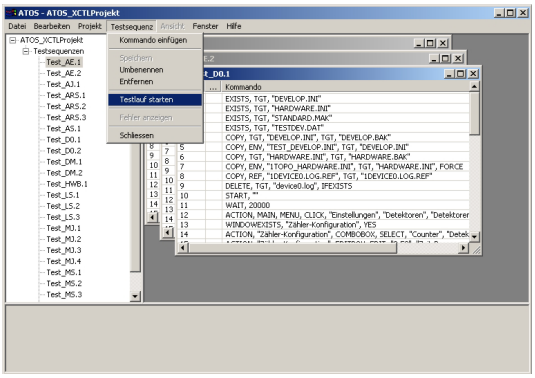

Abb. 4: Die Testsuite *ATOS*

skripten bietet sie die wesentlichsten Funktionen eines Testsystems. Das Programm ist bequem über eine grafische Benutzerschnittstelle bedienbar.

Für eine beliebige zu testende GUI-basierte Windows-Anwendung wird ein neues Projekt in *ATOS* angelegt. Der Verzeichnisbaum enthält neben Testskripten, Referenzdateien und testbegleitenden Programmen auch ein oder mehrere Datenbanken zur Beschreibung der GUI des Testobjektes. Die Datenbanken müssen zunächst mit dem *RCParser* erstellt werden.

Vor Ausführung eines Testskriptes wird mittels der zugrundeliegenden Datenbanken dessen Konsistenz sichergestellt. Weiterhin dient sie der Konstruktion zulässiger Skriptkommandos.

Testskripte können in Testpaketen zusammengefasst werden. Ein vollständiger Regressionstest erfordert die Ausführung eines Testpaketes, welches alle definierten Testskripte für vorhergehende Programmversionen enthält.

2.4 Die Skriptsprachen

Zweischichtenmodell

Die benutzerfreundlichen Skriptkommandos werden nicht direkt durch API-Funktionen des Betriebsystems ausgeführt. Als Zwischenstufe wird zunächst ein Kommando in ein oder mehrere Anweisungen einer tieferliegenden und betriebssystemnahen Skriptsprache übersetzt. In derselben Übersetzungsstufe findet der Übergang von Bezeichnern in programminterne IDs für GUI-Objekte statt.

Dieses Vorgehen hat sich als sehr vorteilhaft erwiesen. Da die Abbildung eines Skriptkommandos auf Kommandos der tieferen Skriptsprache über *Regeldateien* realisiert wird, ist dessen Verhalten jederzeit problemlos modifizierbar. Zudem können neue Skriptkommandos entworfen und in entsprechende Sequenzen der tieferen Sprache abgebildet werden.

Das folgende Beispiel demonstriert den beschriebenen Übergang für ein ausgewähltes Kommando. In einem Fenster bzw. in einer Dialogbox mit dem Titel „Allgemeine Einstellungen“ wird der Schaltknopf „Ok“ angeklickt:

```
ACTION,"Allgemeine Einstellungen",BUTTON,CLICK,"Ok"
```

Das Kommando wird zur Ausführungzeit in folgende Anweisungen der tieferen Skriptprache übersetzt:

```
ISENABLED,CONTROL,"Allgemeine Einstellungen",1
SETFOCUS,"Allgemeine Einstellungen",1
POST,"Allgemeine Einstellungen",1,245,0,0
WAIT,500
```

Neben den Skriptkommandos zur Realisierung von Aktionen auf der Benutzeroberfläche existiert eine Reihe weiterer nützlicher Kommandos. Zum Umgang mit Dateien gibt es Kommandos zum Kopieren, zum Löschen und zur Abfrage ihrer Existens. Da das Verhalten des Testobjekts im wesentlichen durch Konfigurationsdateien bestimmt wird, muss das Testsystem den Austausch dieser Dateien zur Laufzeit ermöglichen. Ferner ist somit der Zugriff auf Ein- und Augabedateien der Steuerungssoftware möglich.

Weitere Kommandos zur Definition von Schleifen, Wartezyklen und Tastatureingaben machen das Testsystem sehr flexibel.

Kommando-Konstruktion

Das Testsystem *ATOS* bietet Unterstützung bei der Konstruktion von gültigen und konsistenten Skriptkommandos. Dafür dient die mit dem *RCParser* verwaltete Ressourcen-Datenbank mit Informationen über grafische Oberflächenelemente des Testobjekts. Dem Entwickler von Testsequenzen werden diese Elemente

als Parameter für die jeweiligen Skriptkommandos zur Auswahl angeboten.

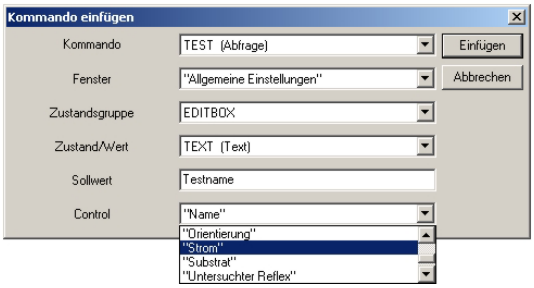

Abb. 5: Ein `TEST`-Kommando

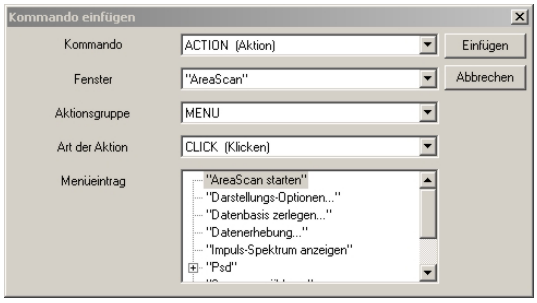

Abb. 6: Ein `ACTION`-Kommando

Abbildung 5 demonstriert die Konstruktion eines `TEST`-Kommandos zur Abfrage des Inhalts eines Eingabefeldes mit der Bezeichnung „Strom“ in der Dialogbox mit dem Titel „Allgemeine Einstellungen“.

Für den letzten Parameter bei der Konstruktion eines `ACTION`-Kommandos in Abbildung 6 ist nur die Auswahl eines gültigen Kontext-Menüeintrags des Fensters mit dem Titel „AreaScan“ möglich.

Die Konstruktion eines Skriptkommandos gestaltet sich dynamisch in Abhängigkeit von vorangegangenen Parametern. Als weitere Unterstützung dienen Kommentare für alle Kommandos und deren mögliche Parameter. Kommentierung und Beschreibung des Kommandosyntax finden wiederum in einer anpassbaren *Regeldatei* statt.

3 Fazit und Ausblick

Die Durchführung skriptbasierter Testsequenzen über die grafische Benutzerschnittstelle des Testobjektes ist ein gängiges Verfahren für BlackBox-Tests und wird bei modernen Testwerkzeugen häufig mit Hilfe eines Makrorekorders realisiert.

Zur Durchführung regelmäßiger Regressionstests des physikalischen Steuerprogrammes als zugrundeliegendes Testobjekt konnte ein flexibles Testsystem entwickelt werden, welches von den Teilnehmern des Projektes zur Softwaresanierung im Zuge weiterer Arbeiten eingesetzt wird.

Das zweistufige Skriptsprachen-Konzept und die konsequente Verwendung von *Regeldateien* machen das Werkzeugsystem sehr anpassungsfähig.

Einige Testschritte sind (derzeit) nicht automatisierbar, beispielsweise Maus-Aktionen auf der GUI, die eine pixelgetreue Positionierung über physische Bildschirmkoordinaten erfordern. Ebenso ist eine Überprüfung erzeugter Grafiken nicht einbezogen worden. Nicht automatisierbare Testschritte werden zur Zeit des Testlaufs interaktiv durchgeführt. Hier existieren Pläne für den Ausbau des Systems.

Literatur

[1] Klaus Bothe: *Reverse Engineering: the Challenge of Large-Scale Real-World Educational Projects*, CSEE&T, 14th Conference on Software Engineering Education and Training, Charlotte, USA 2001

[2] Johann Letzel, Jens Hanisch: *Diplomarbeit: Automatisierung von Regressionstests eines Programms zur Halbleiter-Strukturanalyse*, Humboldt-Universität zu Berlin, Institut für Informatik, November 2002

[3] Mercury Interactive: *Produktbeschreibung von WinRunner*, http://www.mercuryinteractive.de/products/winrunner/