Diplomarbeit

Automatisierung von Regressionstest eines Programms zur Halbleiter-Strukturanalyse

Johann Letzel Jens Hanisch

20.11.2002

Überblick

- Problemstellung
- Verwandte Arbeiten
- Vorgehen / Strategie

Das Testsystem

- Arbeitsweise des Testsystems
- Testdurchführung

Das Testsystem: RCParser

Aufgabe:

Extrahiert aus den Quelldateien (*RC*- und *H*-Dateien) des Testobjekts alle für den Test relevanten Informationen. Zusätzlich wird die Möglichkeit geboten zusätzliche Informationen einzufügen, die nicht in den Quelldateien beschrieben sind.

Zweck:

Die gesammelten Informationen werden im Programm *ATOS* verwendet um während des Testvorgangs das Testobjekt zu steuern.

Das Testsystem: RCParser (2)

Die wichtigsten Funktionen:

- Einlesen von Quelldateien (*RC*- und *H*-Dateien) aus unterschiedlichen Entwicklungsumgebungen.
- Hierarchische Darstellung der Informationen.
- Detaillierte Berichte über Fehler und Konflikte.
- Möglichkeit zum Hinzufügen und Verändern von Oberflächeninformationen.
- Alle Daten werden in einem Projekt verwaltet.
- Aktualisierung des Projekts nach der Veränderungen an den Quelldateien.
- Export der gesammelten Informationen in einer *URF*-Datei für das Programm *ATOS*.
- Verhalten von aussen veränder- und erweiterbar.

Das Testsystem: RCParser (3)

Projektverwaltung:

- Ein *Projekt* wird in einer Projektdatei gespeichert.
- In einem Projekt werden die verwendeten Quelldateien, ihre Lage und die vom Benutzer geänderten oder hinzugefügten Daten gespeichert.

URF-Dateien:

Enthalten alle relevanten Daten über die Oberfläche des Testobjekts:

- **U**niform **R**esource **F**ile
- Beziehungen der Elemente zueinander (z.B. Steuerelemente zu Dialogen)
- Alle Informationen die zur Identifikation eines Elements notwendig sind.
- Andere Daten (z.B. Position, Farbe, usw.) werden nicht gespeichert.

Das Testsystem: RCParser (4)

Erweiterbarkeit:

Viele Aspekte des Verhaltens lassen sich durch externe Dateien konfigurieren.

Folgende Eigenschaften sind erweiter- und veränderbar:

- Gültige Typen von Steuerelementen (*Controls*).
- Beschreibungen für den Import von Quelldateien anderer Entwicklungsumgebungen.
- Dialog-Vorlagen für die Standard-Dialoge des Betriebssystems.
- Identifikatoren von häufig vorkommenden Steuerelementen im Betriebssystem.

Das Testsystem: ATOS

Bedeutung:

Automatisierter **T**est **O**berflächenbasierter **S**ysteme

Aufgabe:

Unterstützung des Benutzers bei der Erstellung, Verwaltung, Durchführung und Auswertung automatisierter Testvorgänge. *ATOS* ist die Hauptkomponente des Testsystems.

Voraussetzung:

Gesammelte Oberflächeninformationen des Testobjekts durch das Programm *RCParser*.

Das Testsystem: ATOS (2)

Die wichtigsten Funktionen:

- Skript-basierte Steuerung des Testobjekts zur Durchführung des Testvorgangs.
- Unterstützung des Benutzers bei der Testsequenzerstellung. Unzulässige Testschritte sind nicht erstellbar.
- Hinweise an den Benutzer auf Konflikte und Fehler in den Testsequenzen.
- Import vorhandener Skript-Dateien als Testsequenzen.
- Import von Testsequenzen aus *CTE*-Diagrammen.
- Verwaltung aller Dateien in einem geschlossenen Projekt.
- Detaillierte Berichte während und nach einem Testvorgang.
- Detaillierte Anzeige des Projektzustands.
- Erweiterbarkeit des Programms ohne es neu übersetzen zu müssen.

Das Testsystem: ATOS (3)

Projektverwaltung:

- Ein *Projekt* bildet die Grundlage für einen Testvorgang.
- Es enthält die verwendeten *URF*-Dateien, Testsequenzen, Testpakete, Protokolldateien, sonst. Dateien (z.B. Referenzdateien) und die Lage des Testobjekts.
- Es liegt in einem eigenen Verzeichnisbaum, damit ist es leicht kopierbar.
- Eine *Testsequenz* ist eine Folge von Testschritten, die verschiedenste Aktionen durchführen können.
- Ein *Testpaket* ist eine Gruppe von Testsequenzen, die sich auf die/das gleiche Komponente/Testobjekt beziehen.

Das Testsystem: ATOS (4)

Erweiterbarkeit:

Das Programm kann mittels externer Dateien erweitert und an neue Anforderungen angepasst werden.

Folgende Aspekte lassen sich konfigurieren:

- Syntax der höheren Skriptsprache (nötig benutzerführende Kommandogenerierung).
- Umsetzung der höheren in die niedere Skriptsprache.
- Regeln für den Import der *CTE*-Diagramme.

Das Testsystem

- DataDiff

Das Testsystem: Design

Implementation ist noch nicht abgeschlossen !

Grundlage ist ein Schichtenmodell:

Eine große Klasse bietet alles, was das Programm später leisten soll. Die GUI greift nur noch auf ein Objekt zu und benutzt dessen Funktionalität.

Vorteil:

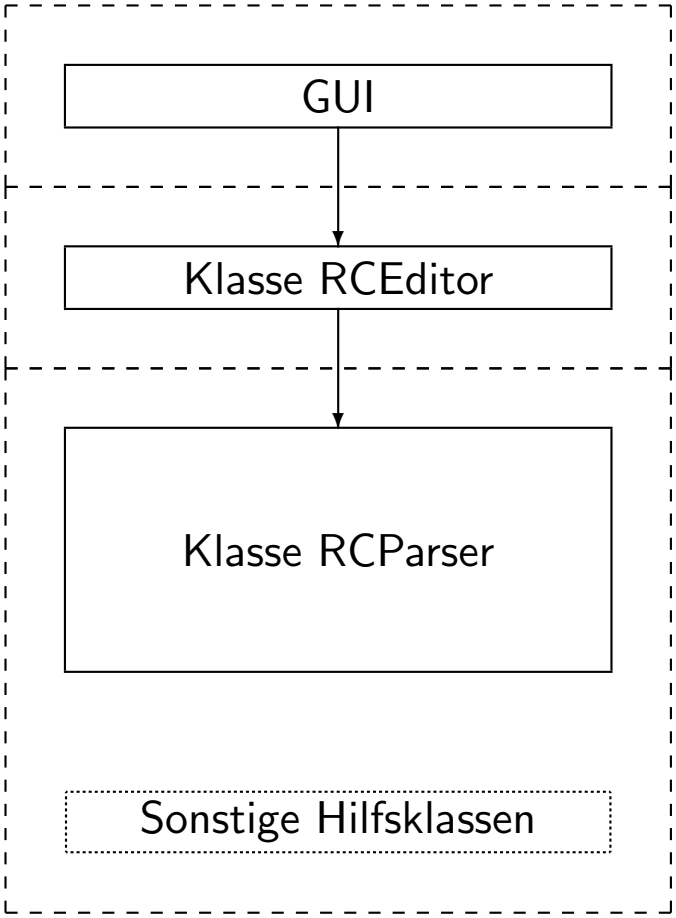
- Trennung von GUI und Funktionalität.
- Strikte Kapselung von Daten.
- GUI kann nur im Rahmen der gewünschten Funktionalität agieren.

Nachteil:

Die Hauptklasse muß sehr viele Methoden bereitstellen.

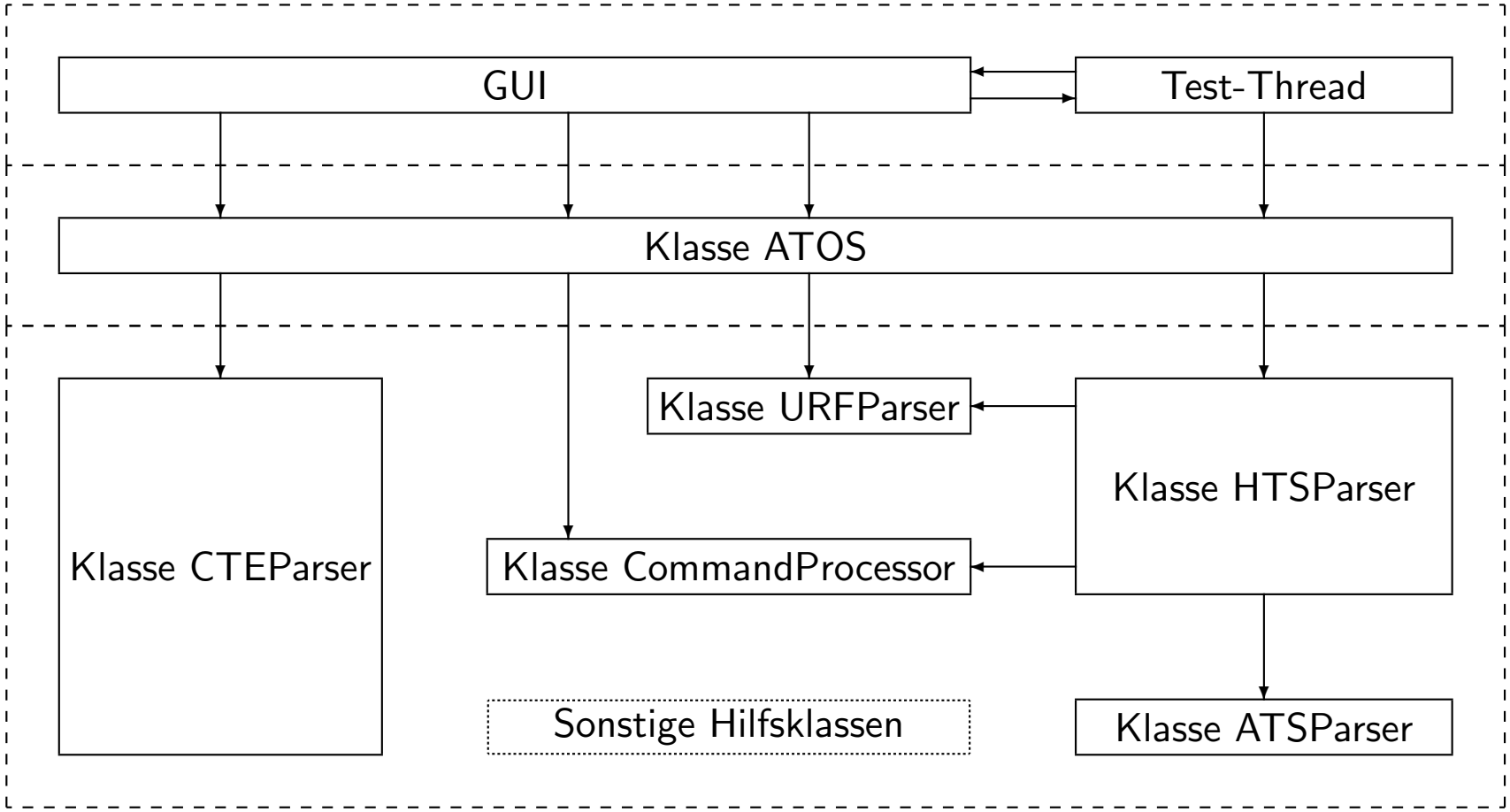
Das Testsystem: Design (2)

Schichtenmodell RCParser



Das Testsystem: Design (3)

Schichtenmodell ATOS



Das Testsystem: Design (4)

Aktueller Umfang:

Komponente		**Umfang in LOC**
ATOS	GUI-Komponenten	11000
	Klasse ATOS	7500
	Klasse ATSParser	6000
	Klasse HTSParser	3900
	Klasse CTEParser	1700
	Summe	**30100**
RCParser	GUI-Komponenten	5700
	Klasse RCEditor	6300
	Klasse RCParser	600
	Summe	**12600**
DataDiff		**600**
Hilfsklassen		**7700**
Summe		**51000**

Das Testsystem: Design (5)

Umfang des GUI-Anteils: etwa ein Drittel

Was ist noch zu tun ?

- *ATOS*: Abschlußarbeiten (Kommentare in Testsequenzen, Projektverwaltung, Feinarbeiten, GUI, Bugs)
- Optimierung/Verbesserung einiger Hilfsklassen
- *RCParser*: Optimierung, Anpassung an geänderte Hilfsklassen, Bugs
- Dokumentation einiger Klassen

Skriptsprachen: Zweischichtenmodell

Niedere Skriptsprache:

- *ATS* (**A**tomic **T**est **S**cript)
- Entstand bereits im Rahmen der Studienarbeit und wurde im Rahmen der Diplomarbeit erweitert.
- Agiert nahe am Betriebssystem.
- Nachteil: Die Sprache ist nicht sehr intuitiv, da zum Anwenden der Sprache Wissen über die Programmierung und interne Vorgänge des Betriebssystems notwendig sind.

Beispiel:

```
POST,"Allgemeine Einstellungen",1,245,0,0
```

Deshalb:

Entwicklung einer höheren intutivien Sprache.

Skriptsprachen: Zweischichtenmodell (2)

Höhere Skriptsprache:

- *HTS* (**H**igh-level **T**est **S**cript)
- Agiert nahe am Benutzer.
- Vorteil: Die Sprache ist intuitiver, da viele interne Aspekte (z.B. IDs) verborgen werden.

Beispiel:

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

Notwendig:

Übersetzung der höheren in die niedere Skriptsprache.

Skriptsprachen: Zweischichtenmodell (3)

Zusammenspiel:

Zur Laufzeit (während des Testvorgangs) werden die *HTS*-Testschritte mittels der Informationen aus den *URF*- und Regeldateien in kleinere *ATS*-Skripte übersetzt und ausgeführt.

Vorteil:

- In den Zwischenschritt kann eingegriffen werden.
- Somit kann die Übersetzung beliebig erweitert und verändert werden.
- *HTS* lässt sich erweitern und auf *ATS* abbilden.
- Theoretisch ist auch eine Übersetzung in eine andere Sprache als *ATS* möglich.

Nachteil:

Gewisser Overhead, der aber im Einsatz kaum in spürbar ist.

Skriptsprachen: Zweischichtenmodell (4)

Beispiel: Testfall AE.1

Im Webdokument, Schritt 4.10:

Button **Ok** anklicken

Benutzer schreibt in HTS:

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

Wird intern umgewandelt in ATS:

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

Skriptsprachen: Zweischichtenmodell (5)

Grundlage bildet die Übersetzungsregel:

```
RULE
   STATE    "NORMAL"
   PATTERN  "ACTION",<SAVE:Window>,"BUTTON","CLICK",<SAVE:Target>
   OUTPUT   "ISENABLED","CONTROL",<LOAD:Window>,<LOAD:Target>
   OUTPUT   "ERROR","14","Das Fenster, in dem der Button ..."
   OUTPUT   "ERROR","15","Der Button konnte nicht gefunden ..."
   OUTPUT   "ERROR","18","Der Button ist nicht aktiv !"
   OUTPUT   "SETFOCUS",<LOAD:Window>,<LOAD:Target>
   OUTPUT   "ERROR","20","Auf den Button konnte kein ..."
   OUTPUT   "POST",<LOAD:Window>,<LOAD:Target>,"245","0","0"
   OUTPUT   "WAIT","500"
   NEWSTATE "NORMAL"
ENDRULE
```

Skriptsprachen: Zweischichtenmodell (6)

Alle Regeln stehen in externen ASCII-Dateien !

Vorteil:

Änderungen an der höheren Skriptsprache sind bis zu einem gewissen Maß ohne direkte Eingriffe in *ATOS* möglich.

Skriptsprachen

- Unterstützung - Skriptentwurf

Bewertung

- Vollständigkeit der Testfälle
- Einsatz in der Praxis

Ausblick

- Weiterführende Arbeiten

Demonstration eines typischen Vorgangs

Am Beispiel der Testfälle der ,,Allgemeinen Einstellungen”