

Glassbox-Test zur Äquivalenzklassenbildung von Produktionsdaten

- ❑ Übersicht zum Glassbox-Test, Begriffe, Definitionen
- ❑ CodeCover
- ❑ Äquivalenzklassenbildung
- ❑ Praxisbeispiel mit Resultaten
- ❑ Zusammenfassung

Rainer Schmidberger

Universität Stuttgart

Institut für Softwaretechnologie, Abt. Software Engineering

rainer.schmidberger@informatik.uni-stuttgart.de

Glassbox-Test

© 2009, Rainer Schmidberger
06.05.2009

- ❑ Alternative Bezeichnungen: Coverage-Test, Whitebox-Test, Strukturtest.
- ❑ Der Glassbox-Test (GBT) beurteilt die Vollständigkeit eines Tests anhand der vollständigen Ausführung einzelner „Bausteine“ des Programms → Glassbox-Test-Entität (GBT-Entität)
- ❑ GBT-Entitäten repräsentieren ein „Stück“ Programmcode wie beispielsweise Anweisungen, Verzweigungen, Schleifen oder Bedingungsterme.
- ❑ Als Resultat des Glassbox-Tests entsteht das Glassbox-Test-Protokoll. Daraus kann u. a. der Anteil der im Programm ausgeführten GBT-Entitäten angegeben werden → Überdeckung, Coverage.
- ❑ Eine Übersicht zu GBT-Werkzeugen steht unter www.iste.uni-stuttgart.de/se/people/Schmidberger/Glassboxtools.html

WSR 2009

Beispiele für GBT-Entitäten

06.05.2009 © 2009, Rainer Schmidberger

| | | | | |
|-------------------------|---|---|------------------|---------------|
| Methoden/ Prozeduren | <code>void m() { ... }</code> | Ausführung | | |
| | | Ja | Nein | |
| Anweisungen | <code>a = a * 0.19;</code> | Ausführung | | |
| | | Ja | Nein | |
| Verzweigungen | <code>if(...) { ... }</code> <code>else { ... }</code> | Ausführung | | |
| | | If-Block | Else-Block | |
| Schleifenkörper | <code>while(...) { ... }</code> | Schleifenausführung | | |
| | | nicht | Genau ein mal | Mehr- fach |
| Prädikate | <code>if(a b) { ... }</code> | Jeder Teilterm soll zum Gesamtergebnis beitragen | | |

WSR 2009

Weitere Beispiele für GBT-Entitäten

06.05.2009 © 2009, Rainer Schmidberger

| | | | | |
|------------------------------|--|--|---|----|
| Synchronisierung | <code>synchronized(o) { ... }</code> | Anzahl wartender Threads | | |
| | | 0 | 1 | >1 |
| Aufzählungen | <code>enum E { e1, e2, e3 }</code> <code>...</code> <code>E e = ...</code> | Alle Aufzählungselemente sollen e zur Laufzeit mindestens einmal zugewiesen werden | | |
| Alternativzuweisungen | <code>e = f ? g : h;</code> | Beide Alternativausdrücke sollen wirksam werden | | |
| | <code>...</code> | | | |

WSR 2009



Äquivalenzklassenbildung von Produktionsdaten

Aufgabe: In der Wartung soll eine Programmschnittstelle getestet werden. Als Testdaten sollen Produktionsdaten verwendet werden.

Vorteile

- Eine große Menge an (aktuellen) Daten liegt vor
- Die Daten sind praxistypisch

Nachteile

- Immer gleiche Eingaben machen den Test aufwändig; die Chance, Fehler zu finden, ist aber gering

Abhilfe:

Äquivalenzklassenbildung der Produktionsdaten

Definitionen



- ❑ Das **Glassbox-Test-Protokoll** TP eines Testfalls T enthält die Häufigkeit der für T ausgeführten GBT-Entitäten
- ❑ Ein **Glassbox-Test-Kriterium** reduziert das Glassbox-Test-Protokoll auf die GBT-Entitäten eines bestimmten Typs: z. B. Anweisungen oder Zweige.

Äquivalenz

- ❑ Zwei Testfälle gelten als **stark äquivalent**, wenn sie bei der Programmausführung hinsichtlich ihrer Fehleraufdeckung als gleich anzusehen sind.
- ❑ Zwei Testfälle sind, bezogen auf ein Kriterium K , **K -Glassbox-äquivalent (K -äquivalent)**, wenn für beide Testfälle das Glassbox-Test-Protokoll, bezogen auf die GBT-Entitäten für K , gleich ist.

$$T_1 \text{ ist } K\text{-äquivalent zu } T_2 \Leftrightarrow TP(T_1, K) = TP(T_2, K)$$

- ❑ Für die weiteren Betrachtungen wird die K -Äquivalenz als Ersatz für die starke Äquivalenz verwendet

Nutzen des Glassbox-Tests

- ❑ **Codeüberdeckungsmaße als Metrik zur Testgüte:** Objektives Vollständigkeitskriterium, das beispielsweise als Testendekriterium verwendet werden kann
- ❑ **Testsuite-Erweiterung:** Der Glassbox-Test zeigt Programmcode, der für eine Testsuite nicht ausgeführt wird und gibt so Hinweise auf fehlende Testfälle.
- ❑ **Grundlage für selektiven Regressionstest:** Anstelle der „rerun-all“-Strategie sollen nur einzelne, ausgewählte Testfälle ausgeführt werden.
- ❑ **Äquivalenzklassenbildung:** Eine Testsuite soll verkleinert werden, ohne (wesentlich) an Testgüte einzubüßen

-
- ❑ **Unterstützung beim Programmcodeverständnis:** Der Glassbox-Test zeigt, welcher Programmcode von welchem Testfall ausgeführt wird.

CodeCover



| | |
|------------------------------|--|
| Entwicklung | 2007-2008 als Studienprojekt am Institut für Softwaretechnologie der Universität Stuttgart. Heute Open Source-Projekt. |
| Homepage | www.CodeCover.org www.sourceforge.net/projects/codecover/ |
| Lizenz | Eclipse Public Licence (EPL) |
| Funktionen | Integration in eine Build-Umgebung (Ant, Eclipse und Batch), Code-Instrumentierung, Laufzeit-Protokollierung und Auswertung. |
| Sprachen | Java, COBOL (Erweiterbar) |
| Überdeckungs-metriken | Anweisungs-, Zweig-, Schleifen- und Termüberdeckung, MC/DC (Erweiterbar) |
| Besonderheit | Die ausgeführten GBT-Entitäten werden je einzeltem Testfall gespeichert (→ Testfallselektive Protokollierung) |

06.05.2009 © 2009, Rainer Schmidberger

WSR 2009



CodeCover: Überdeckungsbericht

06.05.2009 © 2009, Rainer Schmidberger

The screenshot displays the Eclipse IDE interface with the CodeCover plugin. The main editor shows the source code of `ActionHandlerEvent.java` with various lines highlighted in green, yellow, and red, indicating different levels of code coverage. The 'Coverage' window on the right provides a detailed overview of the coverage for various methods, including:

| Name | Statement | Branch | Loop | Strict Condition |
|-----------------------------|-----------|---------|---------|------------------|
| EnrollInfo | 37,3 % | 0,0 % | 0,0 % | 0,0 % |
| EnrollSingleDateAssignment | 56,5 % | 0,0 % | 100,0 % | 0,0 % |
| Equipment | 73,3 % | 100,0 % | 100,0 % | 100,0 % |
| EquipmentAssignment | 63,0 % | 25,0 % | 100,0 % | 0,0 % |
| Event | 58,4 % | 46,1 % | 31,1 % | 25,8 % |
| addAutoBookmark | 100,0 % | 100,0 % | 100,0 % | 100,0 % |
| applyToDoTemplate | 90,0 % | 50,0 % | 66,7 % | 14,3 % |
| calculateToDate | 68,8 % | 25,0 % | 100,0 % | 14,3 % |
| canCancel | 100,0 % | 66,7 % | 100,0 % | 0,0 % |
| canClose | 100,0 % | 80,0 % | 100,0 % | 0,0 % |
| canDelete | 0,0 % | 0,0 % | 100,0 % | 0,0 % |
| canInvite | 100,0 % | 60,0 % | 100,0 % | 0,0 % |
| canOnlineRegister | 100,0 % | 0,0 % | 100,0 % | 0,0 % |
| canRegister | 100,0 % | 87,5 % | 100,0 % | 60,0 % |
| canUndoCancel | 100,0 % | 50,0 % | 100,0 % | 0,0 % |
| canUndoClose | 100,0 % | 75,0 % | 100,0 % | 0,0 % |
| cancel | 0,0 % | 0,0 % | 0,0 % | 0,0 % |
| changeSortOrderParticipants | 0,0 % | 100,0 % | 100,0 % | 100,0 % |
| close | 100,0 % | 50,0 % | 33,3 % | 33,3 % |
| commit | 100,0 % | 50,0 % | 100,0 % | 0,0 % |
| dataExport | 37,5 % | 38,9 % | 100,0 % | 22,2 % |
| dataExport | 72,7 % | 41,7 % | 33,3 % | 12,5 % |
| deadlineReached | 100,0 % | 100,0 % | 100,0 % | 50,0 % |
| delete | 0,0 % | 0,0 % | 0,0 % | 0,0 % |
| enrollMaxWaitingListReached | 100,0 % | 50,0 % | 100,0 % | 0,0 % |
| enrollWaitingList | 100,0 % | 100,0 % | 100,0 % | 100,0 % |
| getActualAccountItem | 0,0 % | 100,0 % | 100,0 % | 100,0 % |
| getAmountEnroll | 100,0 % | 100,0 % | 100,0 % | 100,0 % |
| getAmountEnrolled | 58,3 % | 37,5 % | 16,7 % | 16,7 % |
| getBookingContinentalTerst | 100,0 % | 100,0 % | 100,0 % | 100,0 % |

The 'Test Sessions' window at the bottom shows a list of executed test cases:

| Name | Date | Time |
|---------------------|------------|----------|
| Testfälle Surfer | 21.11.2007 | 11:33:47 |
| Applikation Startup | 21.11.2007 | 16:00:13 |
| Veranstalter C | 21.11.2007 | 16:41:17 |
| veranstalter | 21.11.2007 | 17:50:38 |
| Administrator | 21.11.2007 | 17:50:38 |
| Veranstalter 1 | 22.11.2007 | 19:37:38 |

Anweisungs-, Zweig-, Schleifen und Termüberdeckung

Programmcode-Visualisierung

Ausgeführte Testfälle

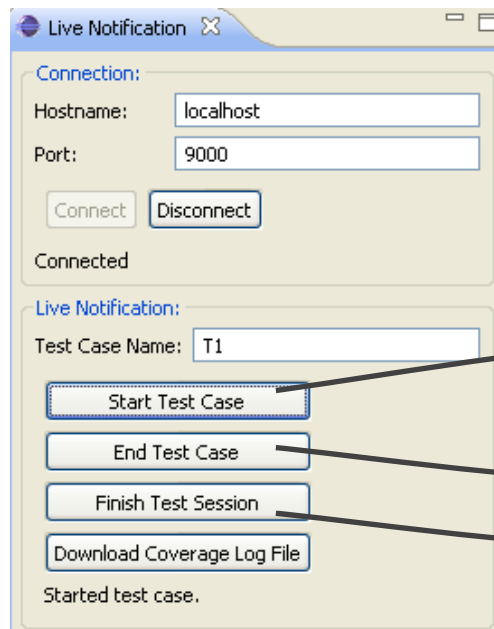
WSR 2009



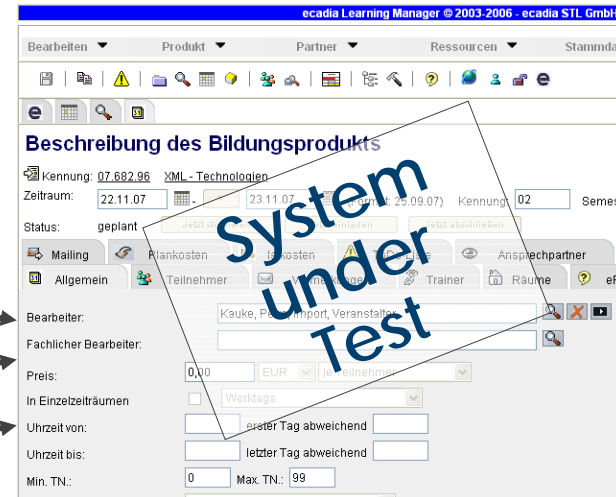
CodeCover: Identifikation der Testfälle (JMX)

06.05.2009 © 2009, Rainer Schmidberger

- ❑ CodeCover bietet einen Notifizierungs-Dialog:
 - ⇒ „Start Test Case“ signalisiert dem SUT den Beginn der Testdaten-Eingabe, der Testfall-Bezeichner wird übermittelt
 - ⇒ „End Test Case“ signalisiert das Ende der Testdaten-Eingabe
- ❑ Die Kommunikation erfolgt über JMX (Java Management Extension). Der JMX-Server wird bei der Instrumentierung dem SUT hinzugefügt.



JMX-Botschaften für Testfall-Beginn und -Ende



WSR 2009



CodeCover: Identifikation der Testfälle (Code)

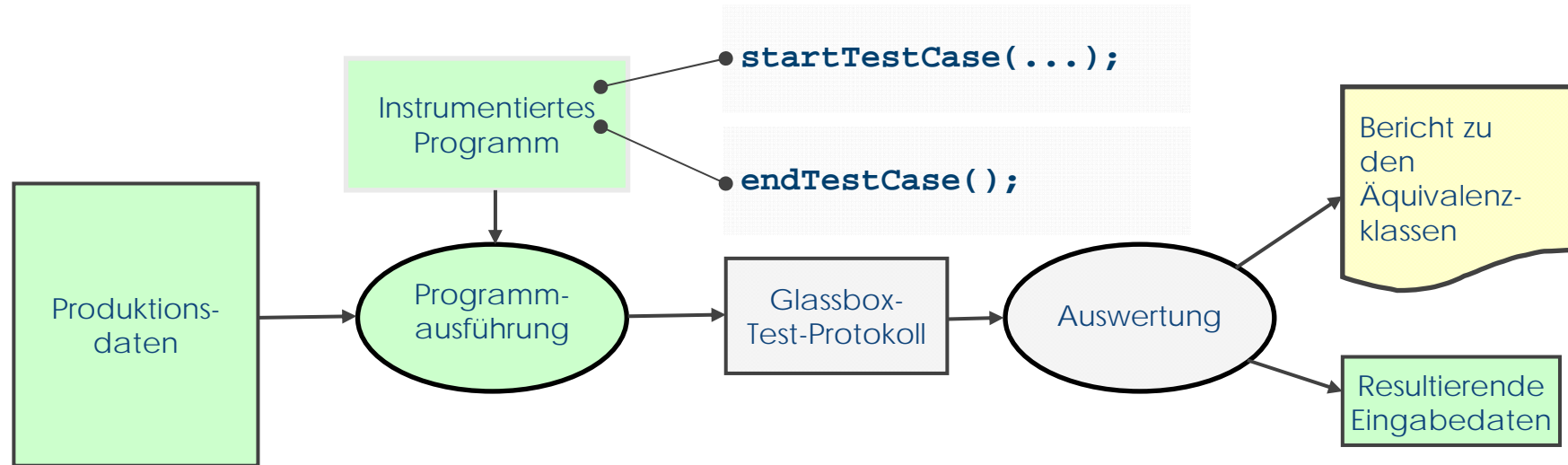
- ❑ Der Programmierer setzt an den Programmcodestellen, an denen die Verarbeitung von Eingabedaten beginnt, spezielle Markierungen:

```
// hier beginnt die Bearbeitung eines Datensatzes
// startTestCase("Testfall_1");
// hier endet die Bearbeitung des Datensatzes
// endTestCase();
// hier endet die Protokollierung
// finishTestSession();
```

- ❑ Die fett gedruckten Zeilen werden vom CodeCover Instrumentierer erkannt und in spezielle CodeCover-Aufrufe transformiert.
- ❑ Anstelle der Angabe „Testfall_1“ wird ein eindeutiges Kennzeichen des Eingabedatensatzes (z. B. ein Primärschlüssel o. ä.) angegeben.

Datenfluss zur Äquivalenzklassenbildung

06.05.2009 © 2009, Rainer Schmidberger



1. Produktionsdaten werden eingelesen und verarbeitet.
2. Je Datensatz wird ein GBT-Protokoll erhoben und abgespeichert.
3. Die GBT-Protokolle werden für ein vorgegebenes Kriterium K paarweise verglichen.
4. K-äquivalente Eingabedaten werden in Äquivalenzklassen zusammengefasst.
5. Die resultierenden Eingabedaten bilden sich aus je einem (beliebigen) Element der Äquivalenzklassen.

WSR 2009

se

Praxisbeispiel

- ❑ Das System-under-Test ist ein XML-Import eines kommerziellen J2EE-Informationssystems
- ❑ Die Soll-Resultate müssen „manuell“ aus der Spezifikation ausgelesen und aufwändig geprüft werden

| | |
|--------------------|------------------------------|
| Daten insgesamt | 720 |
| Kriterium | Anzahl an Äquivalenz-klassen |
| Anweisung | 32 |
| Zweig | 32 |
| Zweig und Schleife | 40 |

- ❑ Resultate
 - ⇒ Die Äquivalenzklassenbildung reduziert die Zahl der Eingaben um fast 95 %
 - ⇒ Die Äquivalenzklassen sind nicht gleich groß; Standardfälle/Sonderfälle sind deutlich zu erkennen
 - ⇒ Aus den Produktionsdaten sind mehr Testfälle entstanden, als die Tester angenommen hatten. Insbesondere Sonderbehandlungen der Geschäftslogik führen zu weiteren (guten) Testfällen.

Praxisbeispiel (2)

CodeCover-Bericht zu den Äquivalenzklassen

© 2009, Rainer Schmidberger

WSR 2009

| | 07.300.01 446 | 04.250.70 100 | 07.729.30 48 | 07.899.08 44 | 07.550.01 16 | 07.798.01 10 | 07.816.01 8 | 07.768.12 5 | 07.842.02 5 | 07.102.01 4 | 07.787.02 4 | 07.19 4 |
|---|------------------|-------------------|-------------------|-----------------|-----------------|-----------------|----------------|----------------|--------------------|-------------------|--------------------|------------|
| | 07.300.01 * | | | L4-1 | | | L4-1 | | L4-1 | | L4-1 | |
| 2 | 04.250.70 45 | * | 5 | 46 | 45 | B36 B32 | 46 | 15 | L4-1 | | 6 | 45 |
| | 07.729.30 43 | B37 B33 B61 | * | 44 | 43 | | 44 | 15 | 4 | B37 B33 B61 | L4-1 | 43 |
| | 07.899.08 | | | * | | | | | | | | |
| | 07.550.01 B43 | B43 | B43 | L4-1 B43 | * | B43 | L4-1 | | | | | |
| 6 | 07.798.01 46 | B37 B33 B61 | B60 B61 B58 | 47 | 46 | * | 47 | | | | | |
| | 07.816.01 B43 | B43 | B43 | B43 | | B43 | * | | | | | |
| | 07.768.12 30 | | B36 B32 | 31 | 30 | B36 B32 | 31 | * | L4-1 | | L4-1 B36 B32 | 30 |
| | 07.842.02 45 | | 5 | 45 | 45 | B36 B32 | 45 | 15 | * | | 5 | 45 |
| | 07.102.01 47 | B56 B62 | 7 | 48 | 47 | 4 | 48 | 17 | L4-1 B56 B62 | * | 8 | 47 |
| | 07.787.02 43 | B37 B33 | | 43 | 43 | | 43 | 15 | B37 B33 | B37 B33 | * | 43 |

Anzahl Elemente der Äquivalenzklasse

Datensatz (Testfall) - Bezeichner

Unterscheidende GBT-Entitäten ("2 hat mehr als 6")

Anzahl der mehr ausgeführten GBT-Entitäten

[JavaScript-Anwendung]

transfer.PISACatalogImport.doImportEvents - - default package.transfer.PISACatalogImport. - - dateTo != null

OK

Zusammenfassung

- ❑ Der Glassbox-Test kann zur Äquivalenzklassenbildung von Produktionsdaten eingesetzt werden
- ❑ Das Werkzeug CodeCover verfügt über entsprechende Funktionen
- ❑ Die Reduktion im Praxisbeispiel liegt bei 95 %; die entstandenen Testfälle (d.h. Äquivalenzklassen) werden von den Testern als sehr nützlich bezeichnet.
- ❑ Es sind weitere Industrieinsätze - auch für COBOL - geplant
- ❑ www.CodeCover.org

Vielen Dank für Ihre Aufmerksamkeit