

Werkzeugunterstützte Testfallermittlung für den funktionalen Test mit dem Klassifikationsbaum-Editor CTE

Joachim Wegener, Matthias Grochtmann
Daimler-Benz AG, Forschung Systemtechnik, Berlin

Zusammenfassung

Ein unverzichtbarer Bestandteil der Software-Prüfung ist ein systematischer Test. Wichtigste Voraussetzung für einen gründlichen Software-Test ist die Ermittlung relevanter Testfälle. Mit den Testfällen werden Art und Umfang der Prüfung und damit die Güte des Tests festgelegt. Obwohl die Testfallermittlung eine große Bedeutung für die Qualität des Tests hat, fehlen insbesondere im Bereich des funktionalen Tests Werkzeuge, die auf leistungsfähigen Methoden basieren und dem Tester in dieser Phase eine effiziente Unterstützung bieten. Diese Lücke soll der im vorliegenden Beitrag vorgestellte Klassifikationsbaum-Editor CTE schließen. Dabei handelt es sich um einen syntaxgesteuerten, graphischen Editor für die Testfallermittlung mit der Klassifikationsbaum-Methode. Die Klassifikationsbaum-Methode beruht auf der Idee des Partition-Testing. Sie unterstützt die anschauliche und systematische Ermittlung von Testfällen und ist besonders gut für eine Rechnerunterstützung geeignet, da sie den Prozeß der Testfallermittlung in einzeln automatisierbare Teilaufgaben zerlegt und über eine graphische Darstellung verfügt, die eine Visualisierung der Testfallermittlung unter Verwendung moderner graphischer Oberflächen erlaubt.

Stichworte: Software-Test, Partition-Testing, Testfallermittlung, Testautomatisierung, Test-Werkzeug

1 Einleitung

Die Qualität und Zuverlässigkeit von Software werden entscheidend durch den Einsatz konstruktiver und analytischer Methoden bei der Entwicklung bestimmt. Aufgabe der konstruktiven Qualitätssicherung ist es, Fehler im Software-Produkt von vornherein zu vermeiden. Da es aber angesichts der Komplexität größerer Software-Systeme praktisch unmöglich ist, Fehler bei der Software-Entwicklung vollständig auszuschließen, sollen mit Hilfe analytischer Maßnahmen noch vorhandene Restfehler möglichst frühzeitig erkannt und beseitigt werden. Die in der Praxis am häufigsten angewendete analytische Methode ist der Software-Test, der als einziges Verfahren Informationen über die dynamischen Eigenschaften der Software liefern und die realen Einsatzbedingungen wie Zielrechner und Compiler berücksichtigen kann.

Sämtliche in einem Programm vorhandenen Fehler lassen sich prinzipiell nur durch einen erschöpfenden Test auf-

decken, bei dem das Testobjekt mit allen möglichen Eingangswerten, -wertekombinationen und -wertereihenfolgen geprüft wird. Ein erschöpfender Test ist in der Praxis aber aus technischen und ökonomischen Gründen nur für wenige triviale Programme durchführbar. Ziel des systematischen Software-Tests muß es im praktischen Einsatz daher sein, durch die Ausführung des Testobjekts mit ausgewählten Eingaben bei einem vertretbaren Testaufwand möglichst viele Fehler zu finden. Gleichzeitig soll ein bestandener Test Vertrauen in die korrekte Funktionsweise des Testobjekts begründen.

Der Test läßt sich grob in die Phasen Testfallermittlung, Testdatengenerierung, Sollwertbestimmung, Testdurchführung und Testauswertung unterteilen. Diese werden von den übergreifenden Tätigkeiten zur Testplanung, Testorganisation und Testdokumentation begleitet. Die für die Qualität des Tests entscheidende Phase ist die Testfallermittlung, weil hier Art und Umfang der Prüfung und damit die Güte des Tests festgelegt werden. Ein Testfall spezifiziert eine bestimmte Eingabekonstellation, mit der das Testobjekt geprüft werden soll, d. h. er umfaßt eine Menge von Eingabewerten. Ein Testfall abstrahiert von einem tatsächlichen Testdatum und bestimmt dieses nur soweit, wie dies für die jeweils intendierte Prüfung notwendig ist.

Obwohl die Ermittlung von Testfällen eine herausragende Bedeutung für die Qualität des Tests hat, fehlen für diese Phase insbesondere im Bereich des Funktionstests rechnergestützte Werkzeuge, die auf leistungsfähigen Testmethoden basieren und einen wirksamen Beitrag zur Effizienzsteigerung des Software-Tests leisten. Diese Lücke soll der im vorliegenden Beitrag beschriebene Klassifikationsbaum-Editor CTE schließen.

Ausgehend von einer kurzen Beschreibung des Stands von Wissenschaft und Technik bei der Testautomatisierung, die vor allem die Probleme einer Werkzeugunterstützung für die Testfallermittlung aufzeigen soll, wird mit der Klassifikationsbaum-Methode ein neuer, zur Automatisierung besonders gut geeigneter Ansatz für die systematische Ermittlung von Testfällen eingeführt und an einem kleinen Beispiel veranschaulicht. Anschließend wird der Klassifikationsbaum-Editor CTE vorgestellt, der auf der Klassifikationsbaum-Methode basiert und eine effektive Rechnerunterstützung für die Testfallermittlung liefert. Das Vorgehen bei der Testfallermittlung mit diesem Werkzeug wird durch die exemplarische Bearbeitung eines größeren Beispiels gezeigt. Das fünfte Kapitel schildert die Erfahrungen aus dem praktischen Einsatz des CTE. Der Beitrag schließt

mit einer Zusammenfassung und Wertung der bisher geleisteten Arbeiten und gibt einen Ausblick auf zukünftige Schwerpunkte.

2 State of the Art

In der Vergangenheit sind zahlreiche Methoden und Werkzeuge zum Testen von Software entwickelt worden, die zum Teil grundsätzlich unterschiedliche Ansätze verfolgen und sich in ihrer Leistungsfähigkeit und ihrem Aufwand erheblich unterscheiden. Eine umfangreiche Beschreibung und eine vergleichende Bewertung der vorhandenen Verfahren wurden unter anderem von DeMillo et al. [1], Liggesmeyer [2] und Grimm [3] gegeben.

Die beiden wichtigsten Ansätze für die systematische Testfallermittlung sind der funktionale Test bzw. Funktionstest ("Black-Box-Test") und der Strukturtest ("White-Box-Test").

Bei den Funktionstests werden die Testfälle aus der funktionalen Spezifikation des Testobjekts abgeleitet. Dabei steht das Ziel im Vordergrund, möglichst viele der spezifizierten Programmfunktionen zu testen. Außer den realen Schnittstellen des Testobjekts werden keine Informationen über die Implementierung des Programms verwendet.

Für Strukturtests basiert die Testfallermittlung hingegen auf der Struktur und den Algorithmen des zu testenden Programms. Die Testziele werden unter Bezugnahme auf Strukturelemente wie Anweisungen, Zweige oder Bedingungen definiert. Die Strukturtests lassen sich in kontrollflußorientierte und datenflußorientierte Verfahren untergliedern. Eines der gebräuchlichsten Strukturtest-Verfahren ist der Zweigtest, bei dem im Laufe der Prüfung alle Verzweigungen des Programms mindestens einmal durchlaufen werden sollen.

Der entscheidende Vorteil der Funktionstests gegenüber den Strukturtests ist die Testfallermittlung aus der funktionalen Spezifikation, da nur sie einen applikationsorientierten Test mit praxisrelevanten Daten wirkungsvoll unterstützt und so die Berücksichtigung der Programmiervorgaben gewährleistet. Durch einen Strukturtest kann nicht festgestellt werden, in welchem Umfang spezifizierte Anforderungen in das Programm umgesetzt worden sind oder ob vorgesehene Funktionen im Laufe der Programmentwicklung unberücksichtigt geblieben sind.

Für eine Reihe von strukturorientierten Testverfahren sind Werkzeuge verfügbar, die insbesondere die verfahrensspezifischen Aktivitäten wie Instrumentierung und Ablaufüberwachung gut unterstützen und Routinetätigkeiten automatisieren. Sie können die genannten, prinzipiellen Schwächen der Strukturtests aber nicht beseitigen. Bei den funktionalen Testverfahren gibt es einige vielversprechende Ansätze, die aus formalen Spezifikationen Testfälle ableiten, z. B. Bernot et al. [4] und Doong et al. [5]. Ihre Verbreitung ist allerdings aufgrund der noch mit formalen Spezifikationsansätzen verbundenen Probleme und der mangelnden Akzeptanz dieser Methoden in der Praxis gering. Für Verfahren, die auf informalen Spezifikationen basieren,

beschränkt sich die Werkzeugunterstützung in der Regel auf Routinetätigkeiten wie die Testdokumentation, die Testdurchführung und die Testauswertung. Die übrigen Testaktivitäten werden nicht oder nur unzureichend automatisiert. Besonders auffällig ist das Fehlen einer methodisch fundierten Rechnerunterstützung für die die Qualität des Tests bestimmende Phase der Testfallermittlung. Gründe hierfür sind in den Unzulänglichkeiten der existierenden Verfahren zu suchen. Den meisten Methoden liegt eine unzureichende Systematik beim Zerlegen des Eingabedatenraums in eine endliche Anzahl testrelevanter Teilmengen zugrunde. Sie sind nicht operationalisiert und größtenteils unhandlich. Viele Verfahren sind auch unübersichtlich und schwer erlernbar. Die Testfallermittlung erfolgt dementsprechend heuristisch und mehr oder weniger unsystematisch. Außerdem hängt der Erfolg der Prüfung stark von der Intuition des Testers und vom Detaillierungs- und Exaktheitsgrad der Spezifikation ab.

Eine Übersicht marktgängiger Testwerkzeuge enthalten unter anderem DeMillo et al. [1] und Graham [6].

Neben einer leistungsfähigen Testmethodik muß ein Testwerkzeug für den Funktionstest über eine komfortable Benutzungsoberfläche verfügen, die den Tester leitet, ohne seine Kreativität einzuschränken. Das Werkzeug muß dem Tester außerdem ermöglichen, seine praktischen Erfahrungen in den Test einzubringen.

3 Klassifikationsbaum-Methode

Eine mögliche Lösung für die oben genannten Probleme beim funktionalen Test bietet die Klassifikationsbaum-Methode [7], indem sie eine weitreichende Unterstützung für die Testfallermittlung liefert. Die Klassifikationsbaum-Methode ist besonders gut für eine Werkzeugunterstützung geeignet, da sie den Prozeß der Testfallermittlung in einzelne Teilaufgaben zerlegt. Der Tester kann dadurch von einem Werkzeug, das die einzelnen Aktivitäten entsprechend automatisiert, bei der Ermittlung von Testfällen geleitet werden. Außerdem beinhaltet die Klassifikationsbaum-Methode eine graphische Notation, die eine Visualisierung der Testfallermittlung unter Verwendung moderner graphischer Oberflächen ermöglicht. Die Klassifikationsbaum-Methode ist eine Weiterentwicklung der "category-partition method" von Ostrand und Balcer [8].

Die grundsätzliche Idee der Klassifikationsbaum-Methode ist es, zuerst den Eingabedatenraum des Testobjekts nach verschiedenen testrelevanten Gesichtspunkten jeweils getrennt voneinander in Klassen zu zerlegen, um dann durch die Kombination geeigneter Klassen zu Testfällen zu kommen. Wichtigste Informationsquelle ist dabei die funktionale Spezifikation, in der das geforderte Verhalten des Testobjekts beschrieben ist.

Im ersten Schritt sammelt der Tester die für den Test relevanten Gesichtspunkte. Jeder Gesichtspunkt soll eine eng begrenzte und damit übersichtliche Unterscheidung der möglichen Eingaben für das Testobjekt erlauben. Anschließend wird unter jedem Gesichtspunkt eine Zerlegung des

Eingabedatenraums vorgenommen. Diese Zerlegung ist eine Klassifikation im mathematischen Sinne, d. h. die Menge aller möglichen Eingaben wird disjunkt und vollständig in Teilmengen, sogenannte Klassen, unterteilt. Da die Zerlegung jeweils getrennt unter nur einem Gesichtspunkt erfolgt, ist sie relativ leicht durchführbar. Zu jedem Gesichtspunkt entsteht eine Klassifikation.

Vielfach ist es sinnvoll, Klassifikationen einzuführen, die nicht den gesamten Eingabedatenraum unterteilen, sondern nur eine Klasse einer anderen Klassifikation. Dieses Vorgehen kann über mehrere Ebenen fortgesetzt werden, bis eine präzise Unterscheidung der möglichen Eingaben erreicht ist. Durch die rekursive Anwendung von Klassifikationen auf Klassen entsteht ein Baum von Klassifikationen und Klassen, der Klassifikationsbaum.

Im letzten Schritt werden die Testfälle gebildet. Ein Testfall entsteht durch die Kombination von Klassen unterschiedlicher Klassifikationen, wobei aus jeder Klassifikation genau eine Klasse berücksichtigt wird. Dabei ist auf die logische Vereinbarkeit der Klassen zu achten. Der Tester wählt insgesamt so viele Kombinationen als Testfälle, wie notwendig sind, um alle Gesichtspunkte ausreichend, auch in ihrer Kombination, zu berücksichtigen.

Die Methode bietet eine anschauliche graphische Darstellung der mehrstufigen Zerlegung in Form des Klassifikationsbaums. Dabei werden unterhalb der Wurzel des Baums die Klassifikationen der ersten Ebene als benannte Rechtecke dargestellt, darunter werden die Bezeichner für die jeweiligen Klassen angeordnet. Entsprechend werden die tieferliegenden Klassifikationen mit ihren Klassen jeweils unter der zugehörigen Klasse notiert. Um Klassenkombinationen als Testfälle festzulegen, wird der Klassifikationsbaum als Kopf einer Tabelle verwendet, in der die zu kombinierenden Klassen markiert werden.

Anhand eines einfachen Beispiels wird die Grundidee der Klassifikationsbaum-Methode erläutert:

Das Testobjekt ist ein Computer-Vision-System, dessen Aufgabe es ist, die Größe von unterschiedlichen geometrischen Objekten zu bestimmen. Die möglichen Eingaben sind verschiedene Bausteine. Die Aufgabe des Testers besteht darin, diesen Eingabedatenraum sinnvoll zu strukturieren und mit Hilfe der Spezifikation eine endliche Anzahl relevanter Testfälle zu definieren.

Zunächst werden die zur Klassifikation geeigneten Gesichtspunkte gesammelt, beispielsweise die Größe, die Farbe und die Form eines Bausteins (Bild 1). Die Bildung einer Klassifikation unter dem Gesichtspunkt Farbe führt zum Beispiel zu einer Zerlegung in rote, grüne und blaue Bausteine, die nach der Form zu einer Zerlegung in Kreise, Dreiecke und Rechtecke. Die verschiedenen Klassifikationen und ihre Klassen werden in einem Klassifikationsbaum notiert (vergl. Baum in Bild 2). Für die Klasse Dreieck wird ein weiterer Gesichtspunkt eingeführt, die Dreiecksart. Daraus ergibt sich eine weitere Klassifikation in gleichseitige, gleichschenklige und ungleichseitige Dreiecke. In der Tabelle zu dem Baum werden in Bild 2 beispielhaft einige mögliche Testfälle markiert. Testfall 3 beschreibt beispiels-

weise den Test mit einem kleinen blauen gleichschenkligen Dreieck.

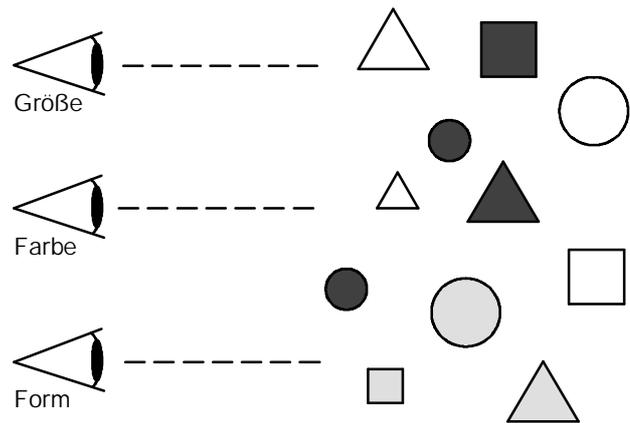


Bild 1: Gesichtspunkte zur Klassifizierung des Eingabedatenraums

Durch die Einführung relevanter und für eine präzise Unterteilung des Eingabedatenraums ausreichender Klassifikationen, die Auswahl einer geeigneten Menge von Testfällen und die Durchführung des Tests mit jeweils einem beliebigen Testdatum zu jedem Testfall ist eine gründliche und systematische Prüfung des Testobjekts möglich.

4 Klassifikationsbaum-Editor CTE

Mit dem Klassifikationsbaum-Editor CTE (Classification-Tree Editor) soll ein Beitrag zur Automatisierung der Testfallermittlung im Rahmen von Funktionstests geleistet werden. Der CTE basiert auf der im vorangehenden Kapitel eingeführten Klassifikationsbaum-Methode und ermöglicht dem Tester den effizienten Einsatz dieser Methode. Der Klassifikationsbaum-Editor wird in den folgenden Abschnitten ausführlich vorgestellt. Anschließend soll durch die beispielhafte Bearbeitung eines weiteren Testobjekts ein Eindruck von der Testfallermittlung mit dem CTE vermittelt werden.

4.1 Aufbau und Leistungsmerkmale des CTE

Da sich die Klassifikationsbaum-Methode stark auf die graphische Darstellung des Klassifikationsbaums und der zugehörigen Kombinationstabelle stützt, bietet sich ein syntaxgesteuerter, graphischer Editor als Werkzeugunterstützung für die Methode an. Der Benutzer wird in beiden Phasen der Klassifikationsbaum-Methode, d. h. beim Entwurf des Klassifikationsbaums und bei der Definition von Testfällen in der zu dem Baum gehörenden Kombinationstabelle, durch den CTE unterstützt. Dafür verfügt der Editor über zwei auf die jeweilige Aktivität spezialisierte Arbeitsbereiche, in denen der Baum und die Tabelle interaktiv bearbeitet werden.

Der CTE arbeitet in einem eigenen Fenster auf dem Bildschirm (Bild 2). Das oberste Element der Benutzeroberfläche ist eine Menüleiste (1), über die der Benutzer Zugriff

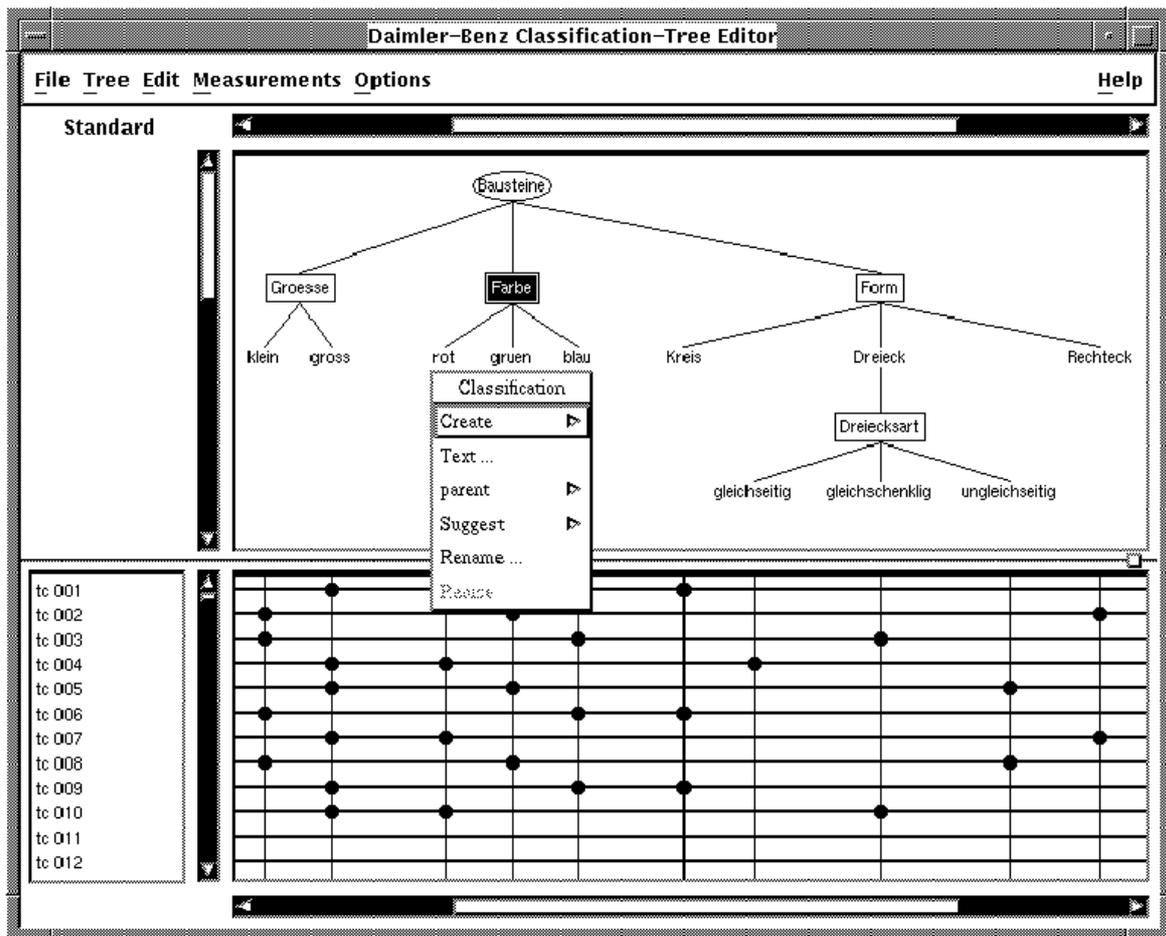


Bild 2: Klassifikationsbaum-Editor CTE mit Testfallermittlung für das Computer-Vision-System

auf mehrere Pull-down-Menüs hat, in denen unter anderem Operationen zum Laden, Speichern, Drucken und Editieren von Klassifikationsbäumen und der zugehörigen Kombinationstabellen zur Verfügung stehen. Darunter befinden sich das Statusfeld (2) und die von einem horizontalen und einem vertikalen Scrollbalken begrenzte Klassifikationsbaum-Zeichenfläche (3). Der Tester entwirft in diesem Arbeitsbereich den Klassifikationsbaum, mit dem der Eingabedatenraum des Testobjekts in Klassen zerlegt wird. Das Statusfeld zeigt dabei den jeweils aktuellen Bearbeitungszustand des Fensters an. Der erstellte Baum bildet den Kopf der Kombinationstabelle (4), die in der unteren Hälfte des Fensters bearbeitet wird. Die Testfälle werden durch die Markierung von Kreuzungen aus Tabellenzeilen und -spalten spezifiziert. Jede Tabellenzeile entspricht dabei einem Testfall. Die Testfälle sind durchnummeriert (5). Die Arbeitsfläche zur Testfallspezifikation verfügt ebenfalls über einen horizontalen und einen vertikalen Scrollbalken. Die horizontalen Scrollbalken der beiden Arbeitsbereiche werden synchronisiert, während sich die vertikalen Scrollbalken unabhängig voneinander bewegen lassen. Eine weitere Komponente der Benutzungsoberfläche bilden unterschiedliche Popup-Menüs (6), die der Benutzer in den Arbeitsflächen aufrufen kann, um spezifische Kommandos für das jeweils selektierte Objekt auszuwählen. Die Höhe

der Arbeitsbereiche kann relativ zueinander verändert werden.

Um die Testfallermittlung für den Programmanwender so komfortabel wie möglich zu gestalten, werden zahlreiche Funktionen durch den Klassifikationsbaum-Editor automatisiert. Hierzu zählen unter anderem die Überprüfung von Syntaxregeln, die sich aus der Klassifikationsbaum-Methode ergeben, das Zeichnen von Verbindungslinien zwischen den einzelnen Elementen des Klassifikationsbaums, das Ableiten einer passenden Kombinationstabelle für den jeweils aktuellen Baum und die Erstellung einer umfangreichen Testdokumentation.

Das Zeichnen des Klassifikationsbaums und das Ausfüllen der Kombinationstabelle erfolgen in den Arbeitsflächen des Editors syntaxgesteuert und objektorientiert. Die Einhaltung der syntaktischen Regeln, die sich aus der Klassifikationsbaum-Methode ableiten lassen, wird durch den CTE aktiv unterstützt. Dadurch werden die Bedienung des Werkzeugs erleichtert und das Verständnis für die Methode gefördert. Beispielsweise ist es nicht möglich, innerhalb eines Testfalls mehrere Klassen derselben Klassifikation zu markieren. Für die Bearbeitung des Klassifikationsbaums und der entsprechenden Tabelle wählt der Benutzer Objekte aus, die er mit den Funktionen des Editors manipuliert. Selektierte Objekte werden invers dargestellt. In der Zeichen-

fläche bilden die Elemente des Klassifikationsbaums die Objekte, in der Tabelle sind es die Testfälle. Die möglichen Operationen werden dem Benutzer in den Pull-down- und Popup-Menüs des Editors angeboten. Dabei richtet sich die Anwählbarkeit der Menüpunkte nach dem aktuell ausgewählten Inhalt der Arbeitsflächen, ist also kontextsensitiv.

Eine aussagekräftige und nachvollziehbare Dokumentation, aus der Art und Umfang des Tests deutlich hervorgehen, ist insbesondere in der industriellen Praxis bei der Abnahme von Tests durch Auftraggeber oder Zulassungsbehörden von erheblicher Bedeutung. Daher liegt ein Schwerpunkt bei der Entwicklung des Klassifikationsbaum-Editors CTE auf der Dokumentation der Testfallermittlung. Neben dem Ausdrucken des Klassifikationsbaums und der Kombinationstabelle lassen sich die Testfälle automatisch in eine textuelle Notation überführen. Das entstehende Dokument ist auch als Grundlage für die weiteren Aktivitäten des Software-Tests geeignet, z. B. für die Testdatengenerierung. Ferner können zu allen Klassifikationsbaum-Elementen und Testfällen beliebige Kommentare angegeben und in die Dokumentation aufgenommen werden. Dadurch kann der Tester seine Entscheidungen verdeutlichen und beispielsweise die für die Prüfung des Testobjekts gewählten Gesichtspunkte erläutern.

Für die Strukturierung umfangreicher Klassifikationsbäume enthält der CTE Verfeinerungsmechanismen, so daß auch die Testfallermittlung für große Testprobleme wirkungsvoll unterstützt werden kann. Es werden Klassifikations- und Klassen-Verfeinerungen unterschieden. Jede Verfeinerung wird in einem eigenen Fenster bearbeitet und kann - wie das Wurzelfenster auch - ikonifiziert werden. Der Benutzer erhält so individuelle Möglichkeiten zur Gestaltung seines Arbeitsbereichs. Bei der Verwendung von Verfeinerungen muß der Tester jedoch zwischen der Übersichtlichkeit des gesamten Klassifikationsbaums, die durch eine große Zahl von Verfeinerungen abnimmt, und der Anschaulichkeit der Darstellung in den einzelnen Fenstern abwägen.

4.2 Beispiel

Im folgenden wird an einem Beispiel die Bearbeitung eines umfangreicheren Testproblems mit dem Klassifikationsbaum-Editor CTE erläutert. Dabei werden die vorhandenen Verfeinerungsmechanismen eingesetzt. Das Beispiel stammt selbst aus dem Test des CTE. Es soll die Prozedur

```
is_line_covered_by_rectangle( IN P1, P2, R1, R2 : POINT,
                             OUT is_covered : BOOLEAN)
```

getestet werden, welche prüft, ob eine durch die Punkte P1 und P2 gegebene Linie von dem durch R1 (linke obere Ecke) und R2 (rechte untere Ecke) gegebenen Rechteck (mit Seiten parallel zu den Achsen des Koordinatensystems) überdeckt wird. Bild 3 verdeutlicht die Aufgabe des Testobjekts anhand eines Rechtecks und verschiedener Beispiels-Linien. Weiterhin definiert die Abbildung Quadranten (gestrichelte Linien), die die möglichen Lagen der Punkte zum Rechteck bezeichnen.

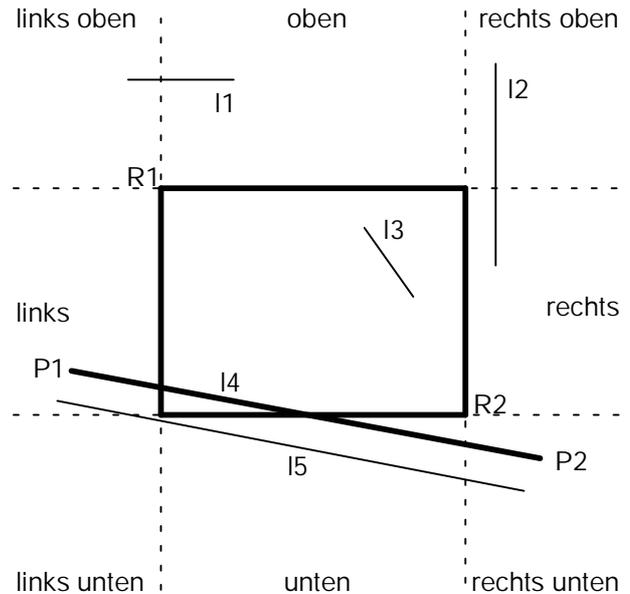


Bild 3: Rechteck mit Beispiels-Linien

Mit der Prozedur wird im CTE festgestellt, welche Verbindungslinien zwischen Elementen des Klassifikationsbaums beim Aufdecken eines Fensters neu gezeichnet werden müssen. Entsprechende Prozeduren finden sich auch in vielen anderen Software-Systemen aus der graphischen Datenverarbeitung.

Für das Beispiel wird davon ausgegangen, daß die Prozedur mit mathematischen Koordinaten und nicht mit Pixeln arbeitet. Weiterhin wird als Vorbedingung für die Prozedur angenommen, daß P1 und P2 nicht identisch sind sowie daß R1 und R2 ein im Sinne der obigen Prozedurbeschreibung gültiges Rechteck darstellen.

Für das Testobjekt wurde eine Testfallermittlung mit dem CTE durchgeführt, wobei 49 Testfälle festgelegt worden sind. Bild 4 zeigt das Wurzelfenster des Editors mit dem entstandenen Klassifikationsbaum und einem Ausschnitt aus der zugehörigen Kombinationstabelle.

Mit dem Baum, der ausgehend von der Klassifikationsbaum-Wurzel schrittweise dadurch erstellt wird, daß Elemente des Baums ausgewählt und durch Kindelemente strukturiert werden, wird der Eingabedatenraum zunächst danach unterschieden, inwieweit Überdeckung vorliegt, ob P1 bzw. P2 im Rechteck liegen oder außerhalb und welche Eigenschaften die Linie besitzt. Liegt Überdeckung vor, wird weiter nach dem Grad der Überdeckung unterschieden.

An verschiedenen Stellen im Baum werden Verfeinerungssymbole verwendet, um an anderer Stelle eine genauere Unterscheidung vorzunehmen.

Exemplarisch wird eine Verfeinerung gezeigt. Bild 5 enthält die Klassen-Verfeinerung außerhalb des Rechtecks für den Punkt P1. In der Verfeinerung werden die Lage des Punktes zum Rechteck und sein Abstand vom Rechteck unterschieden.

Liegt der Punkt P1 innerhalb des Rechtecks, so wird seine Lage in der Klassen-Verfeinerung innerhalb des Rechtecks

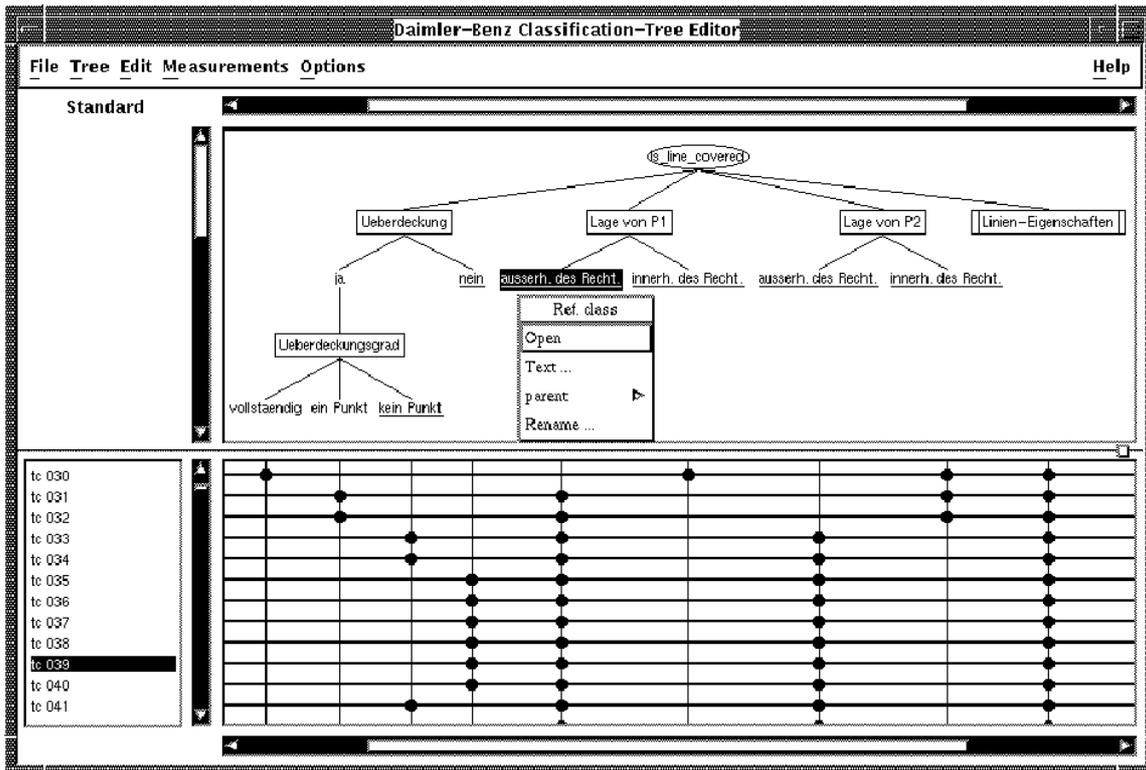


Bild 4: Wurzelfenster des CTE für "is_line_covered_by_rectangle"

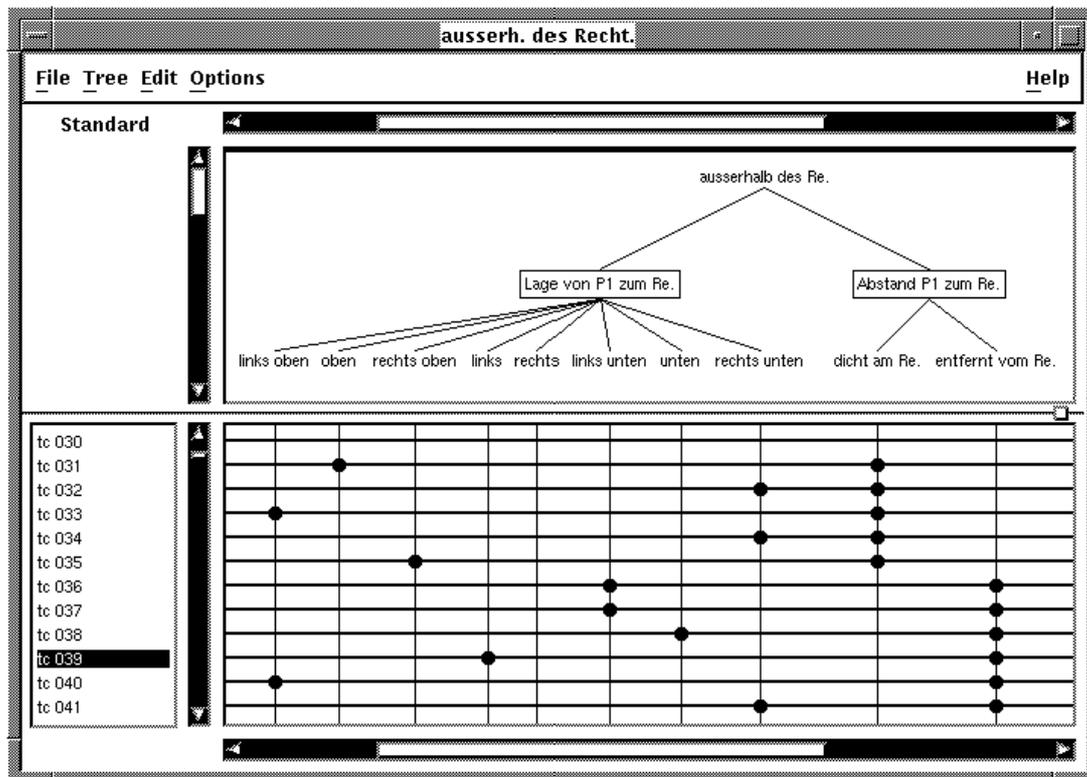


Bild 5: CTE-Fenster für die Klassen-Verfeinerung "P1 außerhalb des Rechtecks"

danach weiter klassifiziert, ob der Punkt auf einer Ecke, auf einem Rand oder im Inneren des Rechtecks liegt.

Die Lage von P2 wird analog zu P1 behandelt.

Innerhalb der Klassen-Verfeinerung kein Punkt wird der Fall, daß eine Linie teilweise überdeckt wird, obwohl beide Endpunkte der Linie außerhalb des Rechtecks liegen, ge-

nauer danach unterteilt, wo die Linie das Rechteck schneidet, beispielsweise durch einen Eckpunkt oder durch zwei Punkte verschiedener Seiten. In der Klassen-Verfeinerung zu Überdeckung nein wird insbesondere der Fall isoliert, daß die Linie nicht vom Rechteck überlagert wird, aber mit minimalem Abstand am Rechteck vorbeiläuft.

Die Gesichtspunkte für Linieneigenschaften werden in einer Klassifikations-Verfeinerung zusammengefaßt. Hier wird zunächst nach dem Linienverlauf in horizontale, vertikale und schräge Linien klassifiziert, dann werden weitere Unterscheidungen getroffen, z. B. nach der Richtung der Linien.

Insgesamt enthält der Klassifikationsbaum 21 Klassifikationen mit zusammen 70 Klassen. Jede dieser Unterscheidungen kann einen wichtigen Anteil an der Aufdeckung potentieller Fehler haben. Dies trifft auf naheliegende Gesichtspunkte wie die Überdeckung zu, aber auch auf speziellere Unterscheidungen, wie die nach der Richtung einer horizontalen Linie, mit der Sonderfälle der für dieses Testobjekt bedeutsamen Geradengleichung isoliert werden können.

Die Testfälle werden durch die Markierung der jeweils gewünschten Klassen in der Kombinationstabelle spezifiziert. Aus den 49 Testfällen wird Testfall 39 herausgegriffen, um einen Eindruck von der Art der Testfälle zu geben. Er entspricht der Beispiels-Linie 15 aus Bild 3 und wird in seiner textuellen Notation vorgestellt.

Testfall 39:

Überdeckung: nein
minimaler Abstand Linie – Rechteck: sehr klein
Lage von P1: außerhalb des Rechtecks
Lage von P1 zum Rechteck: links
Abstand von P1 zum Rechteck: entfernt vom Rechteck
Lage von P2: außerhalb des Rechtecks
Lage von P2 zum Rechteck: rechts unten
Abstand von P2 zum Rechteck: entfernt vom Rechteck
Linienverlauf: schräg
Richtung der Linie (P1 → P2): links oben → rechts unten
Steigung der Linie: sehr gering

Testfall 39 dient der Prüfung, ob das Testobjekt auch dann richtig zwischen Überdeckung und Nicht-Überdeckung unterscheidet, wenn keiner der beiden Endpunkte der Linie im Rechteck liegt. Dabei wurden für diesen Testfall bestimmte Lagen für P1 und P2 sowie ein bestimmter Linienverlauf gewählt. Die Forderung eines sehr kleinen minimalen Abstands zwischen Linie und Rechteck ist ein zusätzlicher Grenzwertest. Testfall 39 wird ergänzt durch den Testfall 26, in dem eine ähnliche Konstellation beschrieben wird, jedoch eine Überdeckung gefordert wird (Bild 3, Linie 14).

Um jede Klasse des beschriebenen Klassifikationsbaums in mindestens einem Testfall zu berücksichtigen, sind 17 Testfälle erforderlich. Die insgesamt definierten 49 Testfälle bewirken einen Test mit einer breiten Spanne von Eingabesituationen, wobei insbesondere durch die geschickte Wahl von speziellen Klassenkombinationen versucht wird, fehlerträchtige Situationen zu isolieren.

5 Erfahrungen aus dem praktischen Einsatz des CTE

Der Klassifikationsbaum-Editor CTE wurde bereits erfolgreich für den Test von Beispielen aus der industriellen Praxis eingesetzt. Die wichtigsten Anwendungsfälle waren ein elektronisches Schiffstagebuch, ein Leitsystem für die Flugfeldbefehrerung eines internationalen Großflughafens sowie ein Erkennungssystem für Briefverteilanlagen und Formularleser. Darüber hinaus wird der CTE für den Selbsttest eingesetzt. Diese Beispiele veranschaulichen das weite Einsatzspektrum der Klassifikationsbaum-Methode und des Werkzeugs.

In den Erprobungen hat sich der CTE stets als wertvolle Unterstützung für den Tester erwiesen. Seine Bedienung ist einfach zu erlernen, und durch die enthaltenen Automatisierungen gestaltet sich der Umgang mit der Klassifikationsbaum-Methode sehr komfortabel. Von den Anwendern wurde während des Einsatzes hervorgehoben, daß das Werkzeug den Tester leitet, ohne seine Kreativität einzuschränken. Ebenso positiv wurden die zahlreichen Möglichkeiten zur übersichtlichen Dokumentation der Testfallermittlung beurteilt, die Art und Umfang des Tests gut widerspiegeln. In allen Anwendungsfällen gewährleistete der Einsatz der Klassifikationsbaum-Methode dabei eine gute Fehleraufdeckungsrate.

Von den Benutzern wurden auch zahlreiche Hinweise auf Verbesserungsmöglichkeiten des Klassifikationsbaum-Editors gegeben, die teilweise bereits in das Werkzeug eingeflossen sind und unter anderem zu zusätzlichen Editierfunktionen geführt haben. Andere Vorschläge sollen in Zusammenarbeit mit Unternehmensbereichen des Daimler-Benz-Konzerns realisiert werden. Hierzu zählen ein automatisches Ausfüllen der Tabelle nach bestimmten Kombinationsregeln und die Fähigkeit zum Multi-User-Betrieb.

6 Zusammenfassung, Wertung und Ausblick

Mit dem Klassifikationsbaum-Editor CTE steht dem Tester ein leistungsfähiges Werkzeug für die übersichtliche und systematische Spezifikation von Testfällen zur Verfügung. Es handelt sich um einen graphischen Editor, der auf der Klassifikationsbaum-Methode basiert. Der CTE bietet damit im Gegensatz zu den marktgängigen Werkzeugen eine methodisch fundierte Rechnerunterstützung für den funktionalen Test in der die Qualität des Software-Tests entscheidend bestimmenden Phase der Testfallermittlung. Dadurch werden die Durchführung von effizienten und reproduzierbaren Tests erheblich erleichtert und eine deutliche Verbesserung der Software-Qualität erreicht. Dementsprechend lassen sich die Kosten für die Systementwicklung und -wartung reduzieren.

Für die Testfallermittlung entwirft der Benutzer in der Zeichenfläche des Editors einen Klassifikationsbaum, der

alle für den Test relevanten Gesichtspunkte abdeckt und den Eingabedatenraum des Testobjekts in Äquivalenzklassen unterteilt. Dabei erlaubt es die Klassifikationsbaum-Methode, die möglichen Eingaben zunächst unabhängig voneinander unter verschiedenen Gesichtspunkten zu betrachten. Jeder Gesichtspunkt bleibt überschaubar und führt zu einer disjunkten und vollständigen Einteilung des Eingabedatenraums. Die entstehenden Klassen können wieder durch weitere Klassifikationen unterteilt werden. Diese stufenweise Zerlegung wird als Baum notiert und dann als Kopf einer Kombinationstabelle verwendet, in der die konkreten Testfälle durch die Kombination logisch vereinbarer Klassen unterschiedlicher Klassifikationen festgelegt werden.

Die verschiedenen Funktionen des Klassifikationsbaum-Editors ermöglichen eine effiziente und benutzerfreundliche Erstellung von Klassifikationsbäumen und der dazugehörigen Kombinationstabellen. Die Orientierung am Prinzip der direkten Manipulation, die kontextsensitive Gestaltung der Menüs und die weitreichende Unterstützung der aus der Klassifikationsbaum-Methode ableitbaren syntaktischen Regeln erleichtern das Verständnis der Methode und die Bedienung des Editors. Mit den Druckfunktionen des CTE kann automatisch eine vollständige Dokumentation der Testfallermittlung erstellt werden, in der Art und Umfang des Tests ausführlich beschrieben und anschaulich wiedergegeben werden. Der Test ist dadurch jederzeit einfach nachvollziehbar. Die entstehenden Dokumente sind außerdem eine gute Basis für die nachfolgenden Testaktivitäten.

Der Klassifikationsbaum-Editor wird bereits in einer Reihe von Unternehmensbereichen des Daimler-Benz-Konzerns erprobt und ist dort auf positive Resonanz gestoßen. Darüber hinaus wird er für den Test der selbstentwickelten Software eingesetzt. Der weitere Ausbau und eine Markteinführung des CTE sind geplant.

Zukünftige Arbeiten am Klassifikationsbaum-Editor sollen ein automatisches Ausfüllen der Kombinationstabelle ermöglichen. Dafür müssen sich logische Abhängigkeiten zwischen den Klassen verschiedener Klassifikationen definieren lassen. Ferner soll eine Automatisierung der Testdatengenerierung durch die Angabe formaler Spezifikationen zu den Klassen des Klassifikationsbaums unterstützt werden. Im Zuge der Entwicklung des CTE wurden bereits entsprechende Erfahrungen mit Prolog als Sprache zur Spezifikation von Testfällen gesammelt [9]. Aktuelle Forschungsarbeiten beziehen sich auf die Generierung ganzer Klassifikationsbäume oder Teilbäume. Es wird ein wissensbasiertes System erstellt, das aus einer technischen Beschreibung des Testobjekts einen Klassifikationsbaum erzeugt oder vorhandene Bäume ergänzt. Die Vorschläge des Werkzeugs basieren auf seinem allgemeinen und anwen-

dungsspezifischen Wissen über Gesichtspunkte und Klassen, die für eine Zerlegung des Eingabedatenraums geeignet sind.

Die bisherigen Erprobungen des CTE haben sich auf sequentielle Programme konzentriert. Die zukünftigen Arbeiten sollen den Bedienungskomfort weiter steigern und dem Klassifikationsbaum-Editor neue Anwendungsfelder erschließen. Hierzu zählen die Testfallermittlung für das Prüfen von parallelen Prozessen, verteilten und Realzeit-Systemen, die sich durch eine größere Komplexität und die Betrachtung von zeitlichen Aspekten miteinander kommunizierender Prozesse bzw. unter Echtzeitbedingungen arbeitender Rechnersysteme auszeichnen. Durch die Integration des Editors in ein phasenübergreifendes Testsystem soll eine weitgehende Rechnerunterstützung für alle Testaktivitäten erreicht werden.

Literatur

- [1] DeMillo, R.A.; McCracken, W.M.; Martin, R.J.; Passafiume, J.F. (1987): *Software Testing and Evaluation*. Benjamin/Cummings Publishing Company, Menlo Park, CA, 1987.
- [2] Liggesmeyer, P. (1990): *Modultest und Modulverifikation*. State of the Art. BI-Wissenschaftsverlag, Mannheim, Wien, Zürich, 1990 (Angewandte Informatik, Band 4).
- [3] Grimm, K. (1992): *Systematisches Testen sicherheitsrelevanter Software – Methoden, Verfahren und Werkzeuge*. In: *Informatik zwischen Wissenschaft und Gesellschaft*, Kreowski, H.-J. (Ed.), Informatik-Fachberichte 309, Springer-Verlag, Heidelberg, 1992, S. 66 bis 107.
- [4] Bernot, G.; Gaudel, M.C.; Marre, B. (1990): *Software Testing based on Formal Specifications: a Theory and a Tool*. Rapport de Recherche no. 410, Université de Paris-Sud, Centre d'Orsay, Laboratoire de Recherche en Informatique, Bât. 490, 9/405 Orsay, Frankreich, Juni 1990.
- [5] Doong, R.-K.; Frankl, P.G. (1991): *Case Studies on Testing Object-Oriented Programs*. Proceedings of the ACM SIGSOFT - Symposium on Software Testing, Analysis, and Verification (TAV 4), Victoria, British Columbia, Kanada, Oktober 1991, S. 165 bis 177.
- [6] Graham, D.R. (Ed.) (1991): *Computer-Aided Software Testing: The CAST Report*. Unicom Seminars Ltd., Middlesex, UK, 1991.
- [7] Grochtmann, M.; Grimm, K.: *Classification Trees for Partition Testing*. Erscheint im *Journal of Software Testing Verification & Reliability*, Wiley.
- [8] Ostrand, T.; Balcer, M. (1988): *The Category-Partition Method for Specifying and Generating Functional Tests*. *Communications of the ACM*, Volume 31, Number 6, Juni 1988, S. 676 bis 686.
- [9] Grimm, K.; Grochtmann, M.; Pitschinetz, R. (1991): *Ein Software-Testsystem mit rechnergestützter Testfallermittlung und Testdatengenerierung*. In: *Software-Entwicklungs-Systeme und -Werkzeuge*, Scheibl, H.-J. (Ed.), 4. Kolloquium der Technischen Akademie Esslingen, September 1991, S. 9.3-1 bis 9.3-8.