Ein Werkzeug zur Überdeckungsmessung für kontrollflussbezogene Testverfahren

Hendrik Seffler

HU Berlin

Was?

Entwicklung eines Werkzeugs zur Überdeckungsmessung für kontrollflussbezogene Testverfahren

- Analyse der Struktur von Softwaresystemen
- nicht-funktionales Testen
- Entwurf von Grund auf (Entwicklung durch alle Phasen der Softwareentwicklung)

Warum?

Test im Rahmen dieses Projektes bisher mit *ATOS*

- funktionsorientiertes Testen
- Testautomatisierung
- keine Prüfung der Struktur

Warum?

Fragen, die *ATOS* nicht beantwortet (nicht beantworten kann)

- Alles, was Kenntnis des Quellcodes voraussetzt
- Wurden alle Anweisungen ausgeführt?
- Warum wurden welche Flüsse durch das Programm gewählt?
- Welche Teile des Programms sind besonders komplex (fehleranfällig)?

Wie?

Kenntnis der Interna eines Programms notwendig

- Parsen des Quellcodes
- Darstellung von Kontrollflüssen

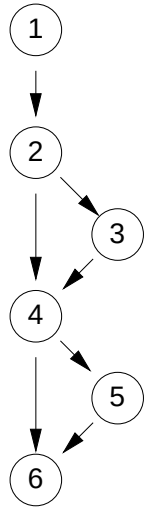
Qualitätssicherung

- Analysierende Verfahren
  - Verifikation
  - Komplexiätsmetriken
- Testende Verfahren
  - Mit Programmausführung
    - Black−Box (funktionsorientiert)
    - White−Box (strukturorientiert)
  - Ohne Programmausführung
    - Audit
    - Review

Was will man wissen?

Ermittlung von Kontrollflüssen

```
void example(boolean a, boolean b)
{
    if (a)
        do();
    if(b)
        doAlso();
}
```



Ziel: Interne Datenstruktur und Visualisierung

Was will man wissen?

Statische ...

- zyklomatische Komplexität
- essentielle Komplexität

... und dynamische Analyse

- Anweisungs-/Zweigüberdeckung
- Bedingungsüberdeckung
- Pfadtests

Was ist gefordert?

- Analyse von Komplexitätsmaßzahlen und Überdeckungsmaßen
- Zielsprachen: Java & C++
- Darstellung und Auswertung

Was ist notwendig?

- Portabilität $\rightarrow$ Java
- Analyse des Quellcodes durch einen Parser $\rightarrow$ *JavaCC*
- *Instrumentierung* des geparsten Codes
- Darstellung von Graphen $\rightarrow$ *JGraph*

Design

Entwurf von mehreren unterschiedlichen Modulen

- GUI - Visualisierung
- Kontrollflussgraphen
- Datenanalyse

Design - GUI

Der grundsätzliche Aufbau der GUI basiert auf der Idee von *Sichten*

- Quellcode, Kontrollfluss, Struktur, Testfälle, Überdeckung

Aus diesen Sichten werden jeweils komplexere Sichten zusammengesetzt

- Komplexität (statische Analyse)
- Überdeckung (dynamische Analyse)
- Architektur (dynamische Analyse)

Design - GUI

Neben der Auswertung existieren noch einige Hilfsfunktionen

- Projekterstellung
- Instrumentierung von Quellcode
- Projektreport
- Hilfe/Dokumentation

Design - Daten

Die GUI greift auf einen gemeinsamen Datenbestand zu

- Kontrollflussgraphen
- Testfalldaten (Knotendurchläufe, Bedingungen, Pfade)

Die Aufgabe des Parsers ist es, Quellcode in einer der unterstützen Sprachen einzulesen und in eine sprachunabhängige Struktur abzubilden.

Design - Daten

Zugriff über Operationen, die unabhängig von der GUI auf den Daten operieren

- `GraphMaker`
- `ComplexityChecker`
- `CoverageChecker`

Trennung zwischen Darstellung und Auswertung

Design - `GraphMaker`

Die Klasse `GraphMaker` bereitet Daten für die interne Darstellung und die Darstellung am Bildschirm auf

- Visualisierung
- Analyse von Komplexität
- Finden von Pfaden
- Anweisungs- und Kantenüberdeckung

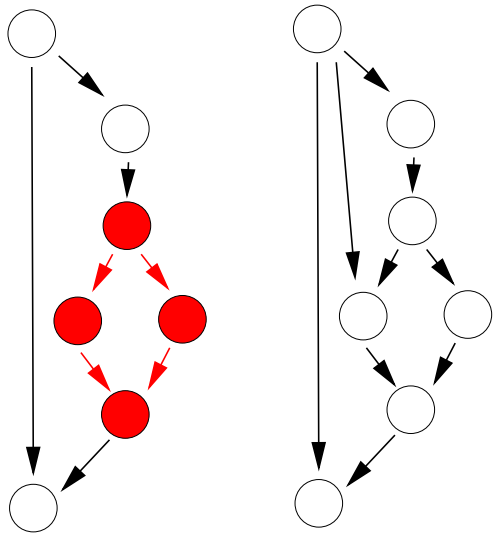
Design - `ComplexityChecker`

Mit `ComplexityChecker` wird Komplexität von Graphen überprüft
Bereitstellung einer einfachen Schnittstelle auf komplexe Algorithmen

- `getCyclomaticComplexity (Function f)`
- `getEssentialComplexity (Function f)`

Design - `ComplexityChecker`

- zyklomatische Komplexität
  $v(G) = |E| - |V| + 2$
- essentielle Komplexität
  sukzessive Entfernung von *Primes*

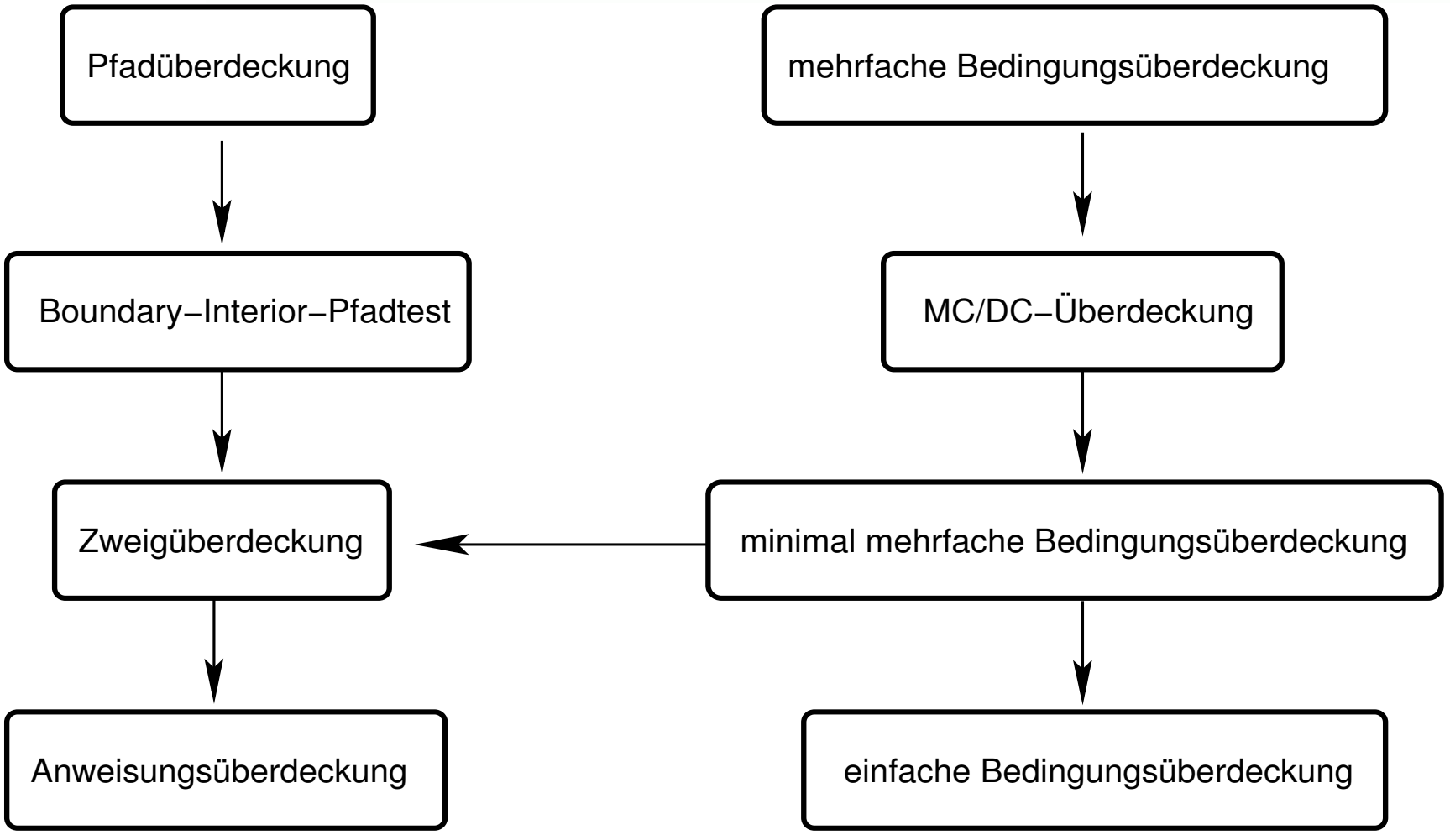


Design - CoverageChecker

Analyse von Überdeckung aus verschiedenen Quellen

- Betrachtung von Knoten des Graphen
- Analyse der einzelnen Bedingungen
- Untersuchung von Pfaden

Design - CoverageChecker



Implementierung - Vorgehen

Iteratives Vorgehen, da Parser und Auswertung parallel entwickelt

- Wahl einer Entwicklungsumgebung
- Versionsverwaltung
- Quellcode-Dokumentation

Implementierung - Größe

Umfang des entwickelten Systems nach Code-Zeilen

- GUI: 6746 Zeilen
- Kontrollflussgraph: 2100 Zeilen
- Projekt-Verwaltung: 2037 Zeilen
- Überprüfung: 1668 Zeilen

Bewertung

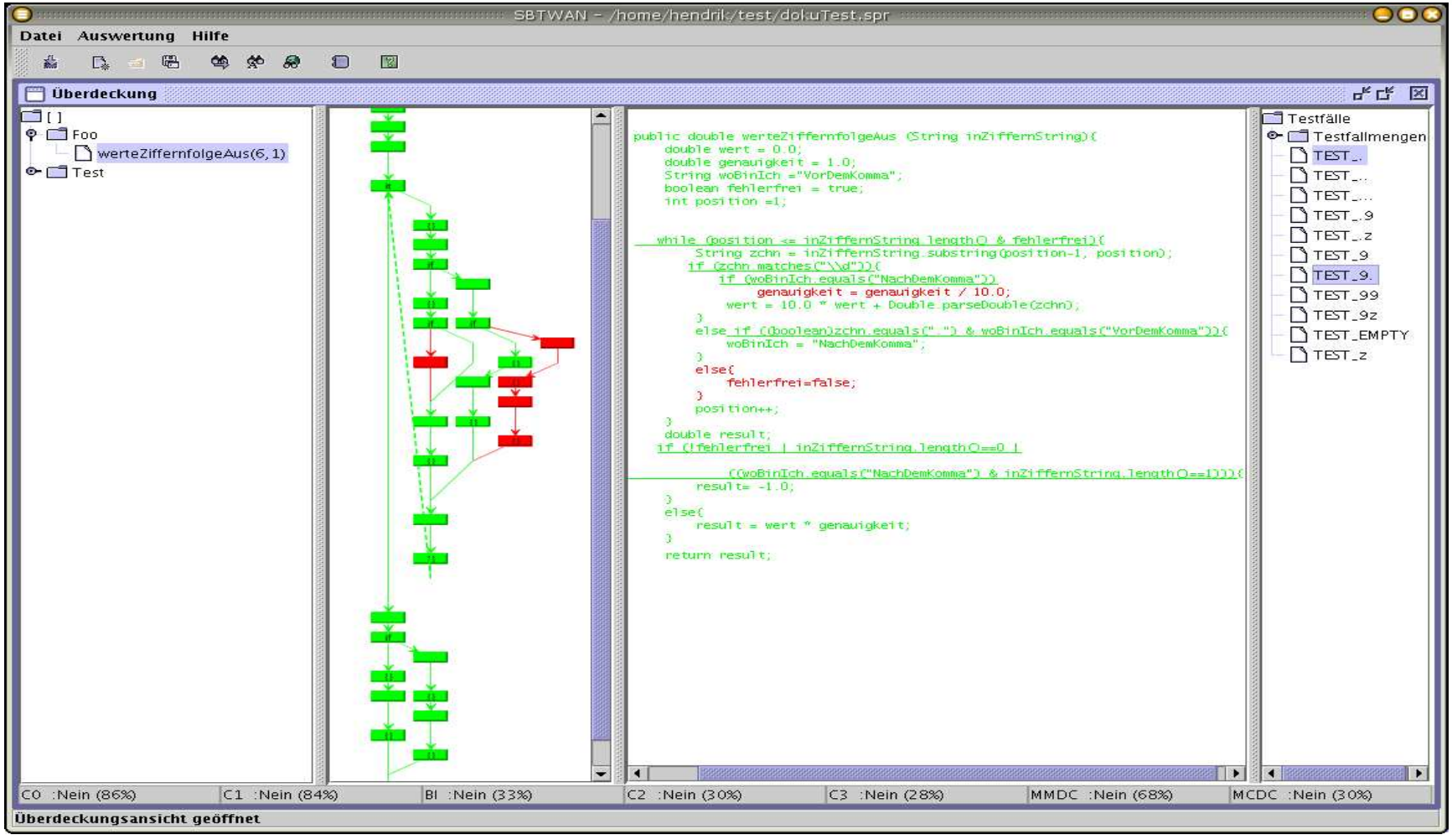
Sprachunabhängiges Framework zum Arbeiten auf Quellcode verschiedener Sprachen. Einheitliche Schnittstelle und Zugriff auf Programmstruktur über Kontrollflussgraphen und Bedingungen.

- Überdeckungsmaße
- Komplexität

Auf dieser Basis sind weitere Erweiterungen denkbar

- weitere Überdeckungsmaße
- Aufrufgraphen
- Profiling

Demonstration



Fragen?

- Fragen?