UNIVERSITÄT
DUISBURG
ESSEN

# Automatisierte Bewertung von Programmieraufgaben

**Dr. Michael Striewe**
**Universität Duisburg-Essen**

PALUNO
The Ruhr Institute for Software Technology

# Über 50 Jahre Forschung

- Automatische Bewertung von Programmieraufgaben hat eine sehr lange Forschungstradition:

  **Jack Hollingsworth: *Automatic graders for programming classes*, Communications of the ACM, 1960, Volume 3, 528-529**

- Seitdem haben sich zwei Dinge geändert:
  - Programmieren ist schwieriger geworden, da es mehr und komplexere Programmiersprachen gibt
  - Computer sind leistungsfähiger geworden
- Das Grundproblem ist aber noch immer dasselbe:
  - Wie kann ein Computer feststellen, ob eine Lösung korrekt ist?
  - Wie kann ein Computer sinnvolles Feedback geben?

# Unsere eigene Geschichte

- Ausgangslage (vor etwa 15 Jahren):
  - Grundlagenvorlesung „Programmierung“ mit ~600 Teilnehmern
  - ~80% Durchfallquote in der Klausur
  - Grund: Mangelnde Beschäftigung mit Übungsaufgaben
- Lösungsansatz 1 (WS 2001/2002):
  - Zwei manuell kontrollierte Testate im Semester
  - Wegen Arbeitsaufwands gescheitert
- Lösungsansatz 2 (ab WS 2005/2006):
  - Sechs automatisch kontrollierte Testate im Semester (+ Verpflichtende Übungsaufgaben)
  - Arbeitsaufwand machbar, aber Raumproblem
- Verbesserung (ab WS 2009/2010):
  - Nutzung der PC-Hall statt kleiner Rechnerpools
  - Funktionierendes Modell für den Regelbetrieb

# Das E-Assessment-System JACK

- Kontrolle von Java-Aufgaben nach verschiedenen Kriterien:
  - Statische Prüfung
    - Compiler
    - Elemente des Quellcodes
    - Codestil
  - Dynamische Prüfung
    - Testfälle
    - Code-Coverage
    - Laufzeit
- Feedback in verschiedenen Formen:
  - Punktzahl
  - Fehlermeldungen
  - Traces
  - Visualisierung von Datenstrukturen

# Das E-Assessment-System JACK

## Ergebnisübersicht

**Dynamic result:** 10
**Static Result:** 100

**Gesamtergebnis:** 37
Gesamtergebnis ist berechnet als **0.3 * Static Result + 0.7 * Dynamic result**
Mindestergebnis für eine korrekte Lösung ist **50**

Seite aktualisieren

## Static result

Keine Kommentare

## Dynamic result

**(-) Fehler in Test 1b: java.lang.ArithmeticException: / by zero**

in der Klasse 'Miniprojekt2' in der Methode 'flagge1' in Zeile 12 - Die Division '/' durch 0 und die Modulorechnung '%' bei Division durch 0 sind keine erlaubten mathemtischen Operationen.

Du kannst dir einen Trace des Testfalles anzeigen lassen.

**(+) Fehler in Test 2a:**
**(+) Fehler in Test 2b:**
**(-) Fehler in Test 2c:**

Die Methode 'flagge2(5)' erzeugt zwar das richtige Muster, aber es werden nicht die richtigen Zahlen verwendet.

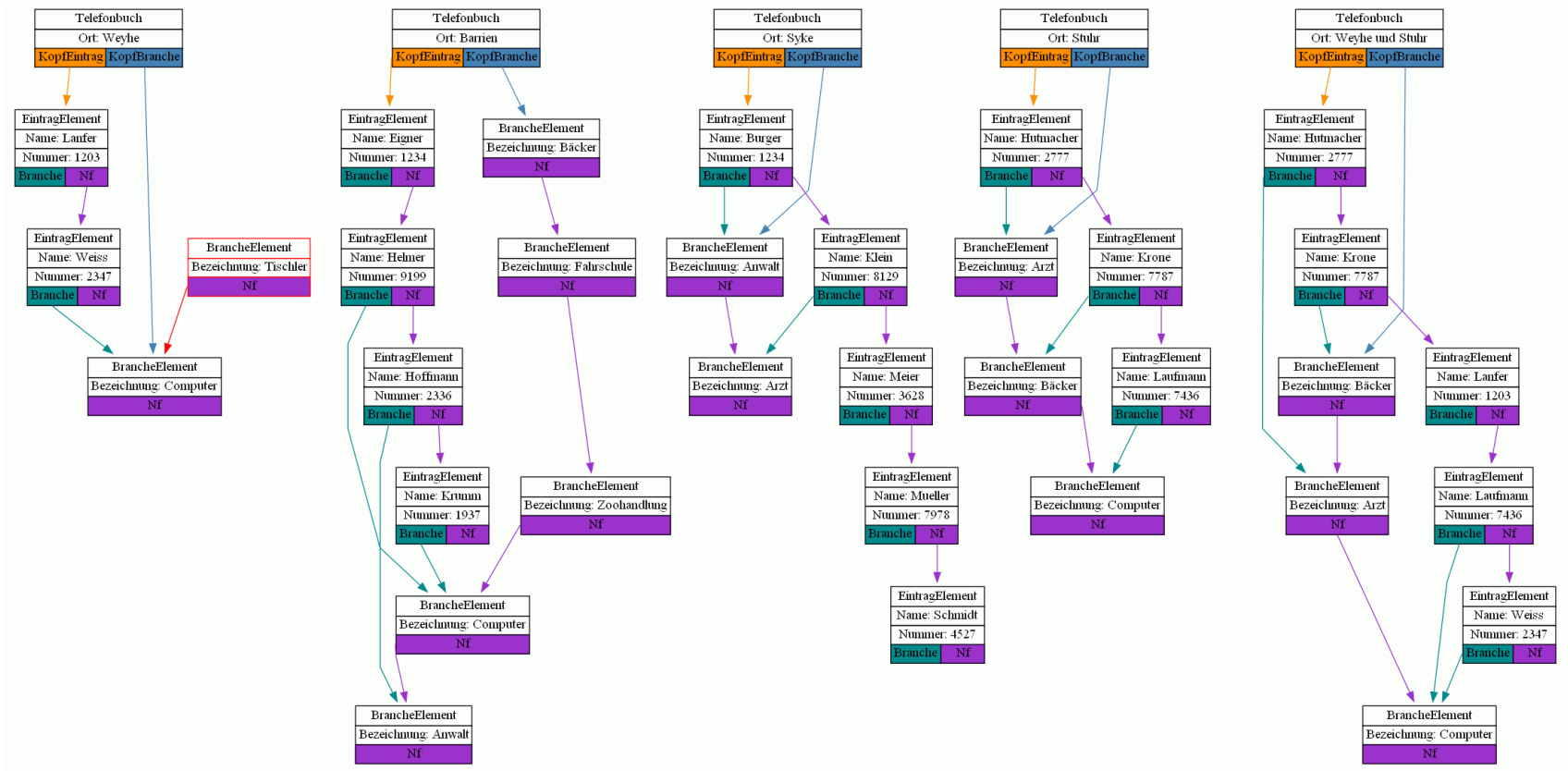
| Deine Ausgabe | Erwartete Ausgabe |
|---|---|
| ..2..<br>..2..<br>43210<br>..2..<br>..2.. | ..3..<br>..3..<br>54321<br>..3..<br>..3.. |

Du kannst dir einen Trace des Testfalles anzeigen lassen.

**(-) Warnung in Test 1c: Testfall übersprungen**

Bevor Test 1c ausgeführt wird, muss Test 1b zumindest ein Ergebnis liefern.

# Das E-Assessment-System JACK



# Einsatzszenarien

- Freier Übungsbetrieb (Self-Assessment):
  - Fokus auf hilfreiches Feedback und Visualisierung
  - Punktzahl und –fortschritt als Motivation
  - Adaptives Systemverhalten
- Strukturierte Übung (Formatives Assessment):
  - Fokus auf faire Bepunktung
  - Feedback als zusätzlicher Leistungsindikator
  - Adaptive Leistungsmessung
- Prüfungsbetrieb (Summatives Assessment):
  - Fokus auf faire Bepunktung
  - Feedback als erklärender Kommentar
  - Manuelle Nachkorrektur

# Weitere Systeme (für Java)

- Praktomat
  - 1998 entwickelt (Uni Passau), 2011 grundlegend überarbeitet (Uni Karlsruhe)
  - Compiler, Test über *JUnit*, Code-Stil über *CheckStyle*
- ASB
  - 2007 entwickelt (HS Trier)
  - U.a. statische Prüfung des Bytecodes über *FindBugs*
- GATE
  - 2011 entwickelt (TU Clausthal)
  - Prüfungsorientierter Einreichungsprozess
  - Plagiatskontrolle
- eduComponents (2006), Graja (2013)
  - Keine Standalone-Tools, sondern zur Integration in CMS bzw. LMS
- Insgesamt über 25 Systeme in den letzten 15 Jahren

# Aufwand bei uns

- Vorbereitung pro Aufgabe:
  - Aufgabenstellung und Musterlösung
  - Testfälle, Prüfregeln, etc. inklusive aussagekräftiger Fehlermeldungen
  - Punkteschemata, Gewichtungen
  - Anmeldung, Login-Kennungen
- Nachbereitung pro Aufgabe:
  - Bei Übungen stichprobenartige Überwachung: ~1 Stunde
    - … auch bei 1.500 Abgaben bei einer Übungsaufgabe!
  - Bei Testaten Endkontrolle: ~1 Stunde
- Gesamtaufwand pro Testat: 5-10 Stunden (+ Aufsicht)

# Stolpersteine

- Vorbereitung ist wichtig
  - Nimmt mehr Zeit ein als die Nachbereitung
  - Aufwand reduziert sich später wieder durch Wiederverwendung von Inhalten
- Übliche Arbeitsweisen und Notfallpläne passen nicht mehr unbedingt
  - Keine Eingriffe „mal eben“ während/nach dem Testat
  - Dafür insgesamt sauberere Durchführung, Nachbearbeitung, Dokumentation
- Die Ansprüche steigen
  - Studierende erwarten schnelle Rückmeldung
  - E-Assessments werden als fester Bestandteil der Lehre gesehen

# Fazit

- Durchführung elektronischer Prüfungen auch für komplexe Aufgabentypen möglich
- Gegenüber anderen Prüfungsformen deutlich erhöhter Vorbereitungsaufwand für
  - organisatorische Aspekte (z.B. Bereitstellung geeigneter Räume)
  - inhaltliche Aspekte (z.B. Konfiguration der Punktevergabe)
- Gleichzeitig noch größere Zeitersparnis bei der Nachbereitung und hohe Flexibilität bei der Wiederverwendung bereits konfigurierter Aufgaben
- Sicht der Studierenden:
  - Keine spezifischen Schwierigkeiten mit dieser Prüfungsform
  - Mit Durchführungsmodus und Ergebnissen der automatischen Bewertung im Wesentlichen einverstanden
- Bisher keinerlei prüfungsrechtliche Schwierigkeiten

# Kontakt

- http://www.s3.uni-duisburg-essen.de/forschung/e-learning-and-e-assessment/jack/
- http://jack-demo.s3.uni-due.de/


- **Dr. Michael Striewe**
  Universität Duisburg-Essen
  michael.striewe@paluno.uni-due.de