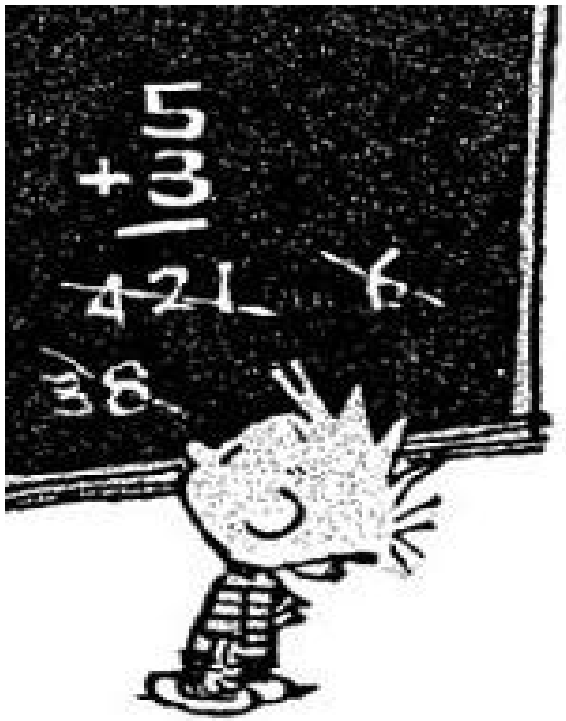
Bachelorarbeit: Trial&Error-Erkennung

Vorstellung Bachelorarbeit

Automatisches Erkennen von Trial-and-Error-Episoden beim Programmieren



von
Hannes Restel

Projektseite:
http://projects.mi.fu-berlin.de/w/bin/view/SE/ThesisTrialError

Gliederung

1)Einführung: Was ist Trial&Error (*T&E*)
   - T&E in Bezug auf Programmierung

2)Motivation und Ziel der Bachelorarbeit

3)Mein Vorgehen
   1)Theorie
   2)Praxis

4)Vorschläge / Tipps / Hinweise

Hinweise

- Bachelorarbeit ist in sehr früher Phase
- Die hier verwendete Syntax ist nicht ausgereift
  - Enthält wahrscheinlich Fehler/Ungenauigkeiten
  - Wird erst in B.Sc.-Arbeit formal festgelegt
- Vorgestellte Episoden und Beispiele sind noch rein hypothetisch

Was ist Trial&Error ??

- Bezeichnet Vorgänge/ Verhaltensweisen zu einem Ziel zu gelangen (*trial*), ohne den Weg zum Ziel zu kennen und dabei bewusst Fehlschläge in Kauf zu nehmen (*error*)
  - ist meist lösungsorientiert / produktorientiert
    - d.h. Erreichen des Ziels ist entscheidend und nicht Weg
    - fehlende Abstraktionen des Lösungswegs zum Erkenntnisgewinn

Was ist Trial&Error II ??

- Zwei Arten von T&E
  - Bewusst
    - Als Methodik zu verstehen
    - Bsp.: Benutzer verwendet Suchmaschinen als sein „Gehirn“/Orakel
  - Unbewusst
    - Diese Vorgänge sind Ziel der Bachelorarbeit
    - Benutzer probiert wirklich aus, ist sich also überhaupt nicht sicher, was richtig und was falsch ist

Trial&Error in der Programmierung

- zwei Arten von T&E festgestellt
  - Nicht-semantisches T&E
    - Auch syntaktisches T&E genannt?
  - Semantisches T&E
- Programmieren ist meist eine Mischung des bewussten und unbewussten T&E

nicht-semantisches T&E

- Kein Schwerpunkt der Bachelorarbeit
  - (wahrscheinlich) ungeeignet für erfolgreiche Defektvermeidung
- Bezug auf
  - pure Syntax der Programmierung
  - sehr einfache und häufige Fehler, welche Entwicklungsumgebungen selbst beheben können
- Sehr stark von Entwicklungsumgebungen geprägt
- Beispiele:
  - Hoffen auf Auto-Vervollständigung
  - Komplexere Verschachtelungen
  - Fehler provozieren und sich dann nach Anweisungen der Entwicklungsumgebung richten
  - (Video) Kommandozeile

nicht-semantisches T&E II (Videobeispiel)

[Starte MediaView...]

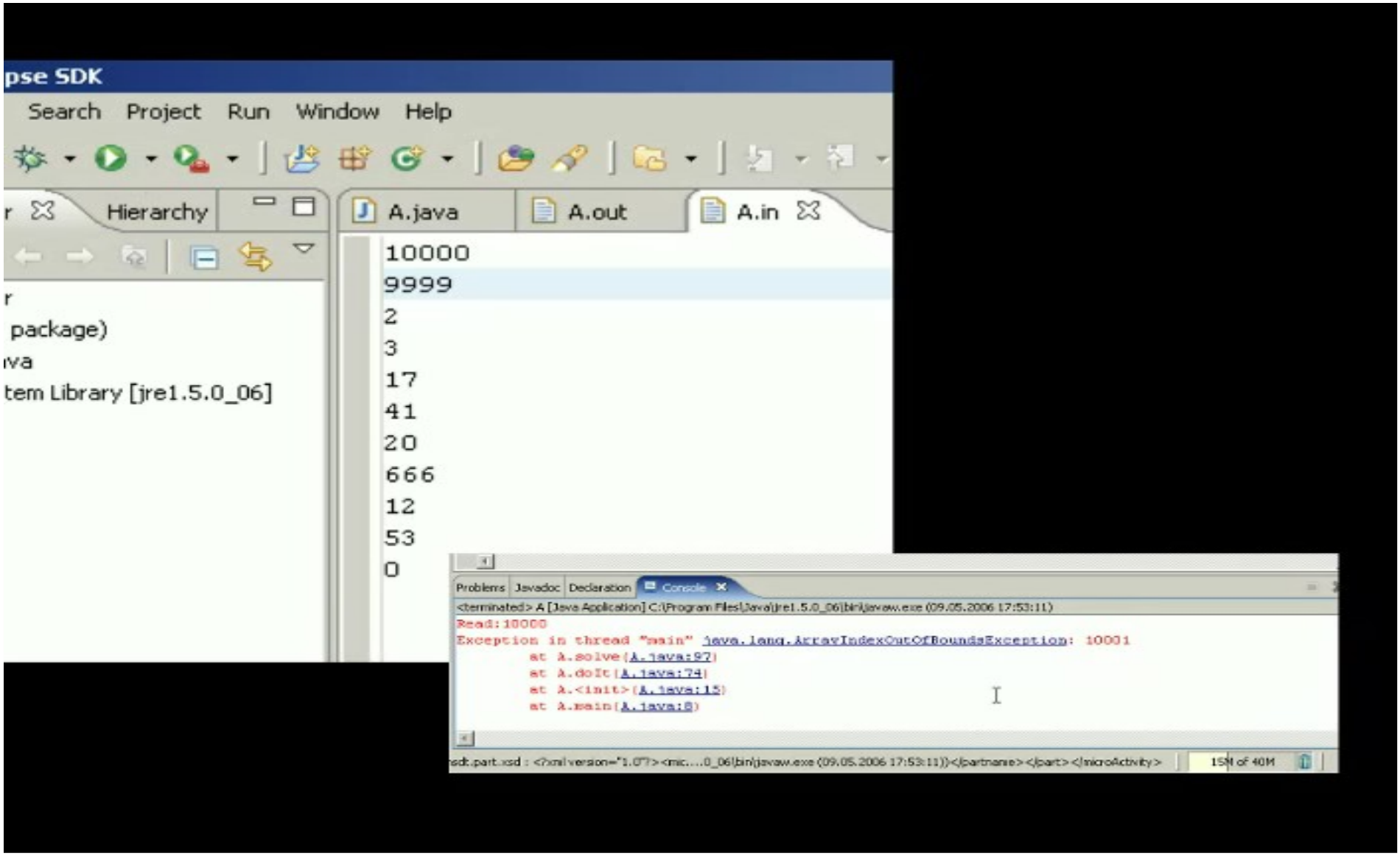
nicht-semantisches T&E III

- Noch vor 10 Jahren:
  - Definition von nicht-semantischem T&E wäre weniger umfangreicher ausgefallen
  - Entwicklungsumgebungen waren sehr viel weniger ausgereift
  - Entwicklungsumgebungen bilden neue Schicht zwischen Entwickler und seinem Code
- Seeehr viel häufigeres Auftreten von Nicht-semantischem T&E als semantischem T&E
- semantisches und nicht-semantisches T&E schwer zu differenzieren

semantisches T&E

- Unsere Definition vom Semantischen T&E:
  - Bezieht sich auf alle Fehler, welche logischer/semantischer/inhaltlicher Natur sind
  - Beginn erst in der Phase, wo Code compilierbar ist, also
  - Wenn Programm ausgeführt wird.
    - Problem: Interpretersprachen
  - Beginn also erst in Testphase/Qualitätssicherungsphase?
  - Bsp: (Video)

semantisches T&E II (Videobeispiel)



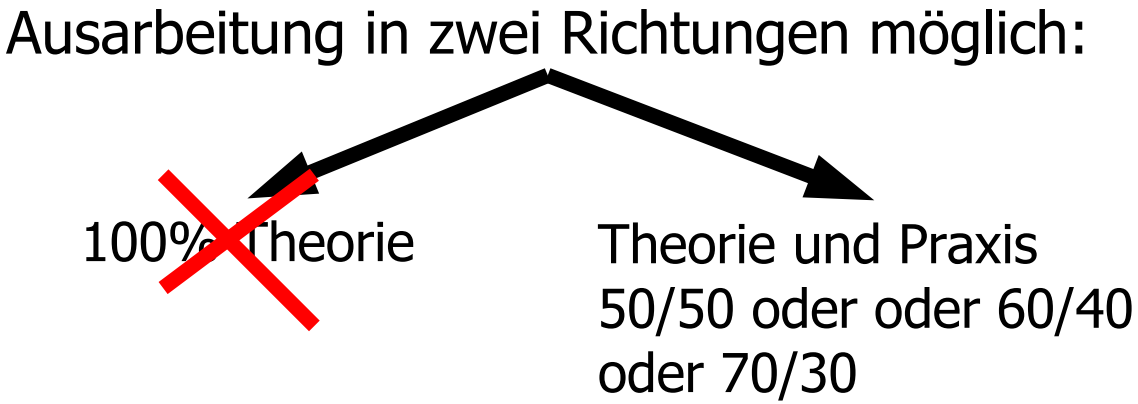
semantisches T&E III

- Einige (mögliche) Indikatoren für sem. T&E
  - Häufiges Ausführen des Programms mit anschließender Codeänderung
  - Mehrfach veränderte Eingabemenge (z.B. eine Datei) für das Programm
    - Mit welcher Eingabemenge arbeitet Programm korrekt?
  - Plötzliches Auskommentieren von Code
  - Vereinfachen des Codes
    - Bereits vorhandene komplexe Ausdrücke durch simplere ersetzen
  - Hinzufügen mehrerer *system.out.println(..)*-Befehle

Motivation und Ziele der Bachelor-Arbeit

- T&E sehr „unsicheres“ Programmieren, da kein System dahinter
  - Wenn es klappt, unterbleibt meist Analyse, warum es klappt
- Programmierer nicht sicher, ob durch T&E keine Fehler (*errors*) in Code eingebaut sind
- Ziel: Defektvermeidung:
  - Erkennung von T&E-Vorgehen bzw. T&E-Episoden, um evtl. Defekte im Code besser zu lokalisieren
  - (Den Programmierer warnen, wenn er T&E macht)
- Ziel: Erkenntnisse und Systematisierung:
  - Bisher keine (uns bekannten) Studien über T&E Analyse in Programmierung, ja in EDV generell (Internet/Usability/usw.)
  - deshalb: Einführung einer Syntax

Motivation und Ziele der Bachelor-Arbeit II



- Theorie ist:
  - Ausformulierung der Erkenntnisse und basierend darauf
  - Spezifikation von T&E-Episoden
  - Formalisierung der Syntax
- Praxis ist:
  - Implementierung der T&E-Episoden in Java auf Basis von Eclipse- und ECG-Ereignissen
  - Benutzung des ECG-Lab

Mein Vorgehen

- Da Bachelorarbeit, steht nur wenig Zeit zur Verfügung
  - Vorgehen deshalb *phänomenologisch*
  - Bedeutung: Sammeln von Erkenntnissen, aber nur deren Auswirkungen werden betrachtet
- Sammeln von möglichen T&E-Episoden
  - mittels Interviews
  - mittels Selbstreflexion / Überlegen
- Durchführen von Interviews
- Beobachten von Programmierern
  - Aufgezeichnete Videos (ggf. mit ECG-Daten)
  - Live
    - Nur einige wenige Probanden, dafür umso genauer
    - Erfahrene und unerfahrene Programmierer beobachten
- Entwerfen und Implementieren des Episodenerkenners

Mein Vorgehen II

- Problematik
  - Rein qualitative Analyse, nicht quantitativ
  - Beobachtungen werden nicht repräsentativ sein
  - Rein pragmatischer Ansatz
    - Deshalb keine harten Beweise
  - Vertrauen in mich
    - Dass ich keine Beobachtungen erfinde
  - Kann nicht bei jeder Beobachtung einen ScreenCapturer laufen

Anregungen...

Anregungen / Hinweise / Tipps ??

Ende...

Vielen Dank!