

Model Mining

Stammvorlesung Softwaretechnik
Andreas Zeller



Programmanalyse

Verifikation • Reengineering

Modell

Deduktion

Lauf



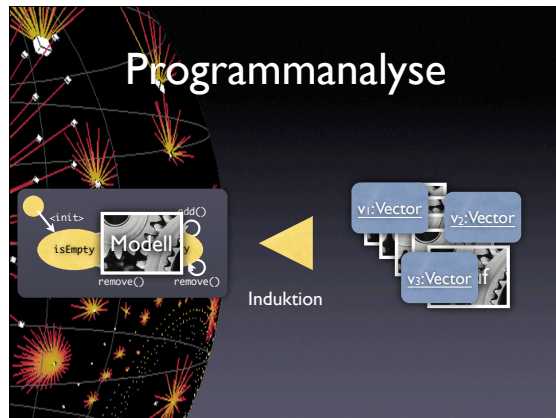
Programmanalyse

Verstehen • Diagnose • Anomalien

Modell

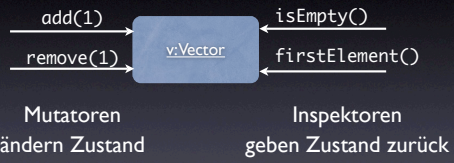
Induktion

Lauf



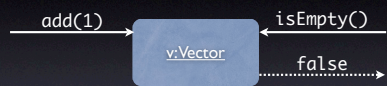
Model Mining

Objekt-Zustände



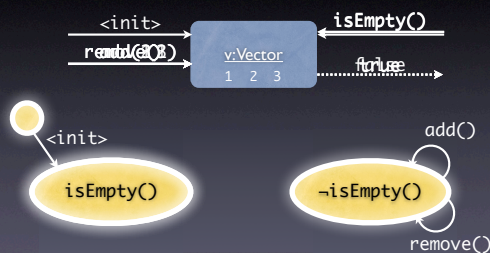
Unterscheidung per statischer Analyse

Modelle bilden

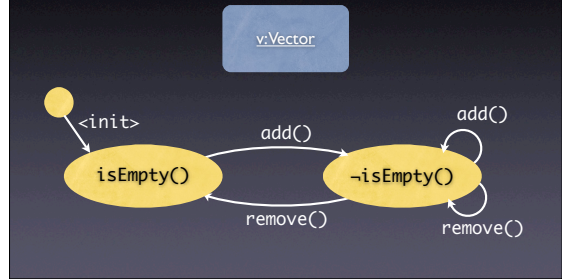


- Nach jedem Mutator-Aufruf bestimmen wir den Objekt-Zustand – mit Inspektoren
- Zustände bilden endlichen Automaten

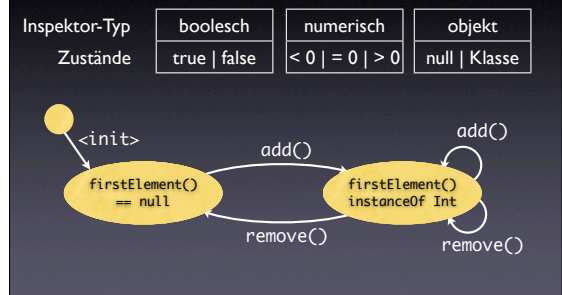
Modelle bilden



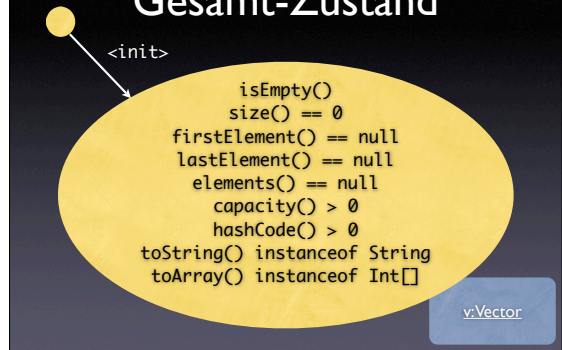
Modelle bilden



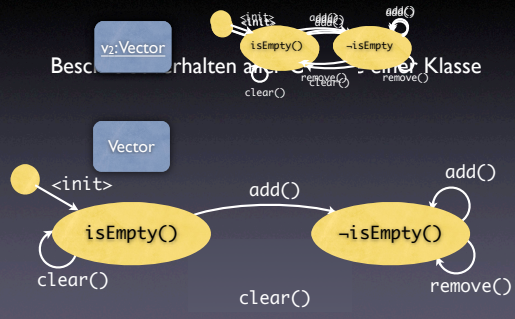
Äquivalenzklassen



Gesamt-Zustand



Klassen-Modell



Formaler Aufbau

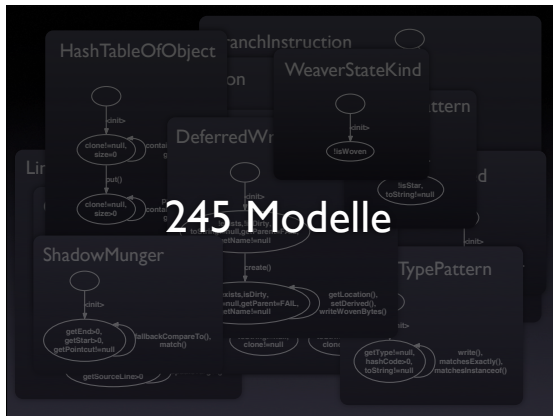
Konkreter Zustand $v \in V$ mit $v = (x_1, x_2, \dots, x_n)$
 x_i – Rückgabewert eines Inspektors

Trace $t = [(v_1, m_1, v'_1), (v_2, m_2, v'_2), \dots]$
 mit $v_i \in V$ und m_i – Name eines Mutators

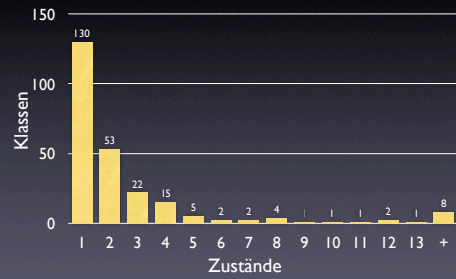
Zustandsabstraktion $abs: V \rightarrow S$

Modell mit Übergängen $s \xrightarrow{m} s'$ und Zuständen $s, s' \in S$

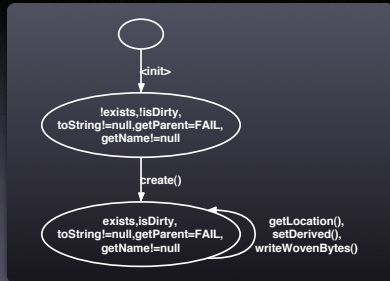
Übergangsbedingung $s \xrightarrow{m} s'$ mit $s, s' \in S$ gdw
 $\exists (v, m, v') \in t \cdot abs(v) = s \wedge abs(v') = s'$



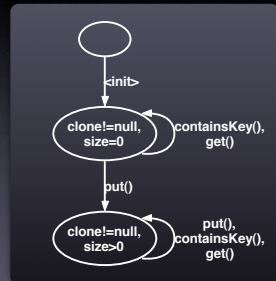
Modellgröße



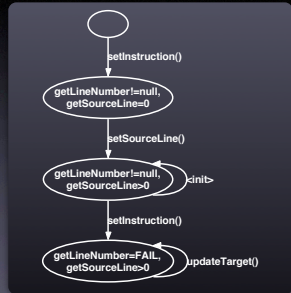
DeferredWriteFile



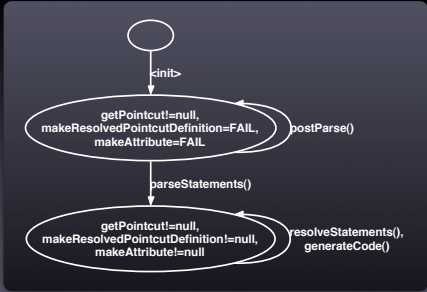
HashTableOfObject



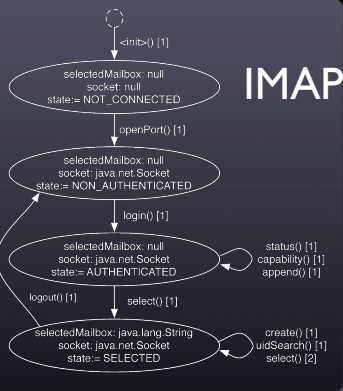
LineNumberGen



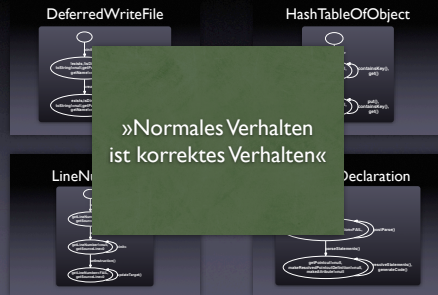
PointcutDeclaration



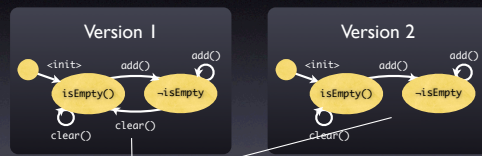
IMAPProtocol



Programme verstehen



Änderungen bewerten

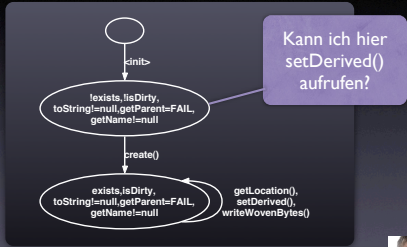


Unterschiede deuten auf Fehler hin



Tom Zimmermann

Verletzungen finden



Kann ich hier setDerived() aufrufen?



Holger Cleve



Andrzej Wasylkowski

Fehlerursachen suchen



- Welche Mutatoren sind fehlerrelevant?
- Vereinfachen mit Delta Debugging

```

void testVector()
{
    v.add(1);
    v.remove(1);
    assert(v.isEmpty());
}
    
```

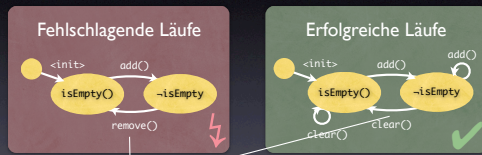


Stephan Neuhaus



Martin Burger

Anomalien erkennen



Unterschiede sind Hinweise auf Fehlerort



Christian Lindig



Valentin Dallmeier

The screenshot shows the Eclipse IDE interface. The top part displays the source code for `ThisJoinPointVisitor.java`. A large, semi-transparent watermark reading "AMT" is overlaid on the code. The left sidebar shows the Package Explorer and a Failure Trace for `BretcodeOptimizeTest`. The bottom console shows the CVS Resource History for `ThisJoinPointVisitor.java`, with a highlighted entry for version `v1.1` dated `12/16/02 7:02 PM` by `wisberg`, with the comment "initial version".

Verwandte Arbeiten



Michael Ernst, MIT
Dynamische Invarianten (TSE 2001)

Vector After `Vector.add()`:
 `this._size > 0`

Verwandte Arbeiten




Michael Ernst

John Whaley, Stanford
Extraction of OO Interfaces (ISSTA 2002)

`<init>`
 `_size = 0`

Verwandte Arbeiten



Michael Ernst



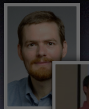
John Whaley



Ras Bodik, Berkeley
Mining Specifications (POPL 2002)

open() • read() • close()

Verwandte Arbeiten



Michael Ernst



John Whaley



Ras Bodik



Tom Henzinger, Stanford / EPFL
Permissive Interfaces (FSE 2005)

– statische Analyse –

Verwandte Arbeiten



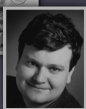
Michael Ernst



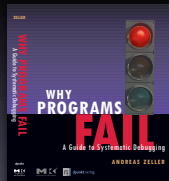
John Whaley



Ras Bodik

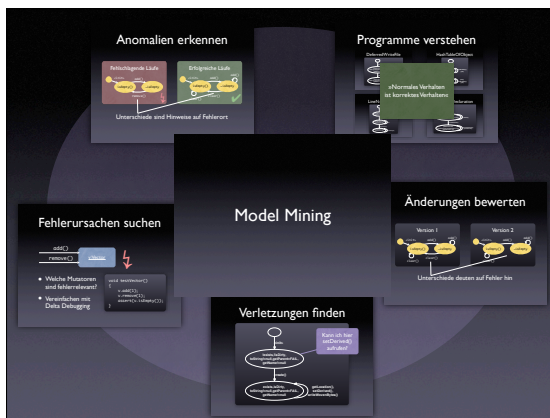
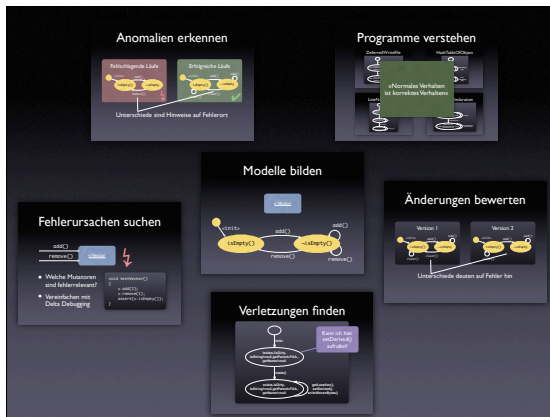
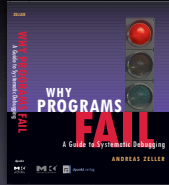


Tom Henzinger



Automated Debugging

- Vertiefungsvorlesung im Sommersemester 2006
- Vorlesung + Praktikum
- Einstieg für Abschlussarbeiten am Lehrstuhl



This work is licensed under the Creative Commons Attribution License. To view a copy of this license, visit

<http://creativecommons.org/licenses/by/2.5/>

or send a letter to Creative Commons, 559 Abbott Way, Stanford, California 94305, USA.