Statische Analyse

Holger Hans Peter Freyther[1]

[1]Freie Universität Berlin

Seminar zu Ursachen und Vermeidung von Fehlern in der Softwareentwicklung, 2006

Übersicht

Übersicht

Übersicht

Outline

Dynamische Analyse

- Ausführen des Programmes
- Resultate einzelner Komponenten testen
- Zur Laufzeit Invarianten prüfen

Statische Analyse

- Betrachtung des Codes ohne ihn laufen zu lassen

Finde den Fehler 1

```
System.out.println("Wo ist der Fehler?");
```

- Wo ist der Fehler?
- Wie beweise ich das kein Fehler vorhanden ist?

Finde den Fehler 2

```
int fib(unsigned int anzahl) {
if( anzahl ==  0 )return 0;
else if( anzahl == 1 ) return 1;
else return fib(anzahl-1)+fib(anzahl-2); }
```

- Wo ist der Fehler?
- Kann dies ein Werkzeug erkennen?

Finde den Fehler 3

```
int a [] = getArray();
for( int i = 0; i < 10*a.length; ++i)
 System.out.println(a[i]);
```

- Wo ist der Fehler?
- Kann dies ein Werkzeug erkennen?

Outline

Überblick an Werkzeugen

- Hoare Kalkül
- Compiler (GNU Compiler Collection)
- Interprozedurale und einfache intraprozedurale Werkzeuge (PREfix, coverity, PREfast, findbugs)
- Heute betrachten wir das intraprozedurale Werkzeug findbugs genauer

Outline

Funktionsweise

- findbugs liest Java Bytecode
- Detektoren suchen nach bekannten fehleranfälligen Spracheigenarten
- Gefunde Eigenarten werden berichtet
- Berichte können per GUI betrachtet werden oder als XML weiterverarbeitet werden.

Bekannte Fehlermuster

- Es werden über 200 unterschiedliche Fehlermuster erkannt
- Nullzeiger Derefenzierung
- Unitialisierte Variablen
- Ungenügende Sperrsynchronisation
- Und viele andere fehleranfällige Konstrukte
- Es ist einfach zu erweitern

Outline

Was ist findbugs

- Einfach
- Intraprozedural
- Unzuverlässlich (unsound)
- Perspektive der Betrachtung von Detektoren
  - Sind sie richtig oder falsch
  - Effektiv oder Ineffektiv

Outline

Anwendung auf die Quellen

- Kein Fehler ist so offensichtlich das er nicht gemacht wird
- Drei Fehler in SUNs Java Implementierungen gefunden
- etliche weitere Nullzeiger Derefenzierungen (SUN, eclipse, classpath)
- Inkonsistente Synchronisation (SUN, classpath)

Outline

Kurze Demonstration von findbugs

Outline

Was ist es nicht?

- Ersatz für den Korrektheits Beweis
- Ersatz für die dynamische Analyse
- Ersatz für die Benutzung der mentalen Fähigkeiten

Was sind die Schwächen?

- Per Entwurf ungenaue Ergebnisse
- Möglicherweise überwiegen die Falschmeldungen für einige Detektoren

Was sind die Stärken?

- Geschwindigkeit der Analyse
- Echte Fehler werden trotz der einfachen Detektoren gefunden
- Gute Integration in eclipse und ant
- Alle Schwächen sind beherrschbar

Empfehlung

- findbugs bietet die Chance Fehler zu finden
- findbugs findet Fehler die durch Hinschauen schwer zu finden sind
- findbugs findet unabhängig von schlechten Testfällen Fehler
- findbugs ist eine Chance bessere Software zu schreiben
- Denken schadet nicht

Outline

Microsoft Research PREfast, PREfix und Integration

- PREfix interprozedurales Werkzeug von Microsoft
- Integration und automatisches Eintragen von Fehlern in ”RAID”
- PREFast schnellere intraprozedurale Variante

PREfix zum Bewerten der Qualität

- Verwendung der von PREfix gespeicherten Daten um Aussagen über die Fehleranfälligkeit der Windows Module zu treffen [**?**].
- Zu 90% richtig
- Guter Indikator welche Module noch ein Mal geauditet werden sollten.
- PREfast ist ein grosses Problem für die weitere Verwendung dieser Analyse