Lecture 21:
Static Analysis Tools

Where static analysis tools fit

Example tools

Limitations of static analysis

Static Analysis

Analyzes the program without running it
- Doesn't need any test cases
- Doesn't know what the program is supposed to do
- Looks for violations of good programming practice
- Looks for particular types of programming error

Where it fits as a verification technique:
1) Avoid dumb mistakes
   - Pair Programming
   - Code Inspection
   - Developer unit testing ("test case first" strategy)
2) Find the dumb mistakes you failed to avoid
   - Style Checkers
   → Static Analysis
3) Make sure the software does what it is supposed to
   - Black box and system testing
   - Independent testing

(Note: Also need validation techniques!)

How Static Analysis Works

```
void print_to_file(string filename)
{
    if (path_exists(filename)) {
        // FILENAME exists; ask user to confirm overwrite
        bool confirmed = confirm_loss(filename);
        if (!confirmed)
            return;
    }
    // Proceed printing to FILENAME...
}
```

"…should have…"

Manual Inspection?
(impractical or impossible)

Correctness Property P

Automatically construct models for analysis

Implies

Class structure and inheritance   State Machine Model   Dataflow graph
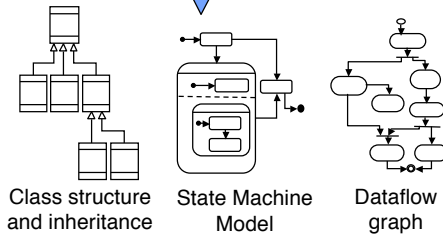
Automatic check of derived model

Model Property P'

Example tools

FindBugs

Originally a research project at U Maryland
Has large number of bug patterns
http://findbugs.sourceforge.net/

JLint

Developed by Konstantin Knizhnik, updated by Cyrille Artho
http://jlint.sourceforge.net/

PMD ("Programming Mistake Detector"??)

written by Tom Copeland
focuses on inefficient code, e.g. over-complex expressions
http://pmd.sourceforge.net/

ESC/Java (Extended Static Checker for Java)

Originally developed at Compaq Research
ESC/Java2 is open source, managed at U College Dublin
http://kind.ucd.ie/products/opensource/ESCJava2/

Different tools find different bugs

```java
import java.io.*;
public class foo{
  private byte[] b;
  private int length;
  Foo(){ length = 40;
    b = new byte[length]; }
  public void bar(){
    int y;
    try {
      FileInputStream x =
        new FileInputStream("Z");
      x.read(b,0,length);
      c.close();}
    catch(Exception e){
      System.out.println("Oopsie");}
    for(int i = 1; i <= length; i++){
      if (Integer.toString(50) ==
          Byte.toString(b[i]))
        System.out.print(b[i] + " ");
    }
  }
}
```

variable never used (detect by PMD)

Method result is ignored (detected by FindBugs)

Don't use '==' to compare strings (detected by FindBugs and JLint)

May fail to close stream on exception (detected by FindBugs)

Array index possibly too large (detected by ESC/Java)

Possible null dereference (detected by ESC/Java)

5

Different tools find different bugs

Bug Category	Example	ESC/Java	FindBugs	JLint	PMD
General	Null dereference	✓	✓	✓	✓
Concurrency	Possible deadlock	✓	✓	✓	✓
Exceptions	Possible unexpected exception	✓			
Array	Length may be less than zero	✓		✓	
Mathematics	Division by zero	✓		✓	
Conditional, loop	Unreachable code due to constant guard		✓		✓
String	Checking equality with == or !=		✓	✓	✓
Object overriding	Equal objects must have equal hashcodes		✓	✓	✓
I/O stream	Stream not closed on all paths		✓		
Unused or duplicate statement	Unused local variable		✓		✓
Design	Should be a static inner class		✓		
Unnecessary statement	Unnecessary return statement				✓

6

Limitations of Static Analysis

Large numbers of false positives

Tool reports large number of things that aren't bugs

Programmer must manually review the list and decide

Sometime too many warnings to sort - E.g. in Rutar et. al. (approx 2500 classes)

	ESC/Java	FindBugs	JLint	PMD
Concurrency Warnings	126	122	8883	0
Null dereferencing	9120	18	449	0
Null assignment	0	0	0	594
Index out of bounds	1810	0	264	0

False negatives

Types of bugs the tool won't report

(increased risk if we filter results to remove false positives?)

Harmless bugs

Many of the bugs will be low priority problems

Cost/benefit analysis: Is it worth fixing these?

7

Which bug is worse?

```
int x = 2, y = 3;
if (x == y)
    if (y == 3)
        x = 3;
else
    x = 4;
```

```
String s = new ("hello");
s = null;
System.out.println(s.length());
```

Detected by:
PMD (if using certain rulesets)

Not detected in testing

Detected by:
JLint,
FindBugs,
ESC/Java

Also detected in testing

8

4