FindBugs

Anne Van Devender, Stuart Vassey, and Levi Throckmorton

Motivation

- Raise awareness about the large number of easily-detectable bugs
- Even well tested code frequently has a large number of errors
- Adoption for automatic bug-finding programs
  - Directly towards most profitable to fix
  - Not distracting with smaller/ irrelevant bugs

Goals of the Tool

- Find repeated obvious errors
- An automated search process could save time
- Want to use broad methods
- Increase awareness of common bugs
- Avoid executing the program
- Avoid using test cases

Environments

- Several front-ends exist
  - Simple batch application generating text reports
  - Batch application generating XML reports
  - Interactive tool for browsing errors
- Users have created two more front-ends
  - Plugin for Eclipse
  - Plugin for Apache Ant

Installation in Eclipse

- Go to **help>software update**
- Choose the **available software** tab
- Click the **add site** button on the right and enter: http://findbugs.cs.umd.edu/eclipse
- Select the check boxes next to Find-bugs
- Hit **install**
- Click **next** and **finish**

Intended Uses:
What Bugs to Catch

- The obvious bugs
- Bugs that result from programmers gaps in knowledge
- Bug patterns-error prone coding practices that arise from erroneous design patterns, misunderstanding of language semantics, or simple and common mistakes

What Bugs to Catch (cont.)

- Four real categories of bug patterns
  - Single threaded correctness issues
  - Thread/synchronization issues
  - Performance Issues
  - Security and Vulnerability to malicious untrusted code
- 45 detectors used in Find-bugs
  - Simplest techniques possible
  - Classify the errors found

Bug Patterns

Dm: Method invokes toString() method on a String (DM_STRING_TOSTRING)

Calling String.toString() is just a redundant operation. Just use the String.

Nm: Class defines equal(Object); should it be equals(Object)? (NM_BAD_EQUAL)

This class defines a method equal(Object). This method does not override the equals(Object) method in java.lang.Object, which is probably what was intended.

Dm: Method invokes System.exit(...) (DM_EXIT)

Invoking System.exit shuts down the entire Java virtual machine. This should only been done when it is appropriate. Such calls make it hard or impossible for your code to be invoked by other code. Consider throwing a RuntimeException instead

How to find bugs...

- Bugs are developed by starting with real bugs
- Uses 300 different bug patterns
- Use information about types, constant values, and special flags
- Also traverses flow graphs for data flow analysis
- But NOT inter-method flow.

For example...

- http://findbugs.sourceforge.net/bugDescriptions.ht
-

```
/ if var.equals(null):
```

```
result.add(new AbstractAction("Quit") {
          public void actionPerformed(ActionEventev)
{
              System.exit(0);
          }
      });
      return result;
```

```
/ public String myvar;
```

Features

- Customize which bugs it finds
- Works with existing software development tools
- Scalable, flexible

Related Tools

- **PMD:** used with Java, but analyzes source code.
  - more for enforcing coding standards.
  - for example, empty try/catch blocks, complex expressions, etc.
  - customized to add your own types of bugs.
- **QJ Pro:** analyzes also the source code.
  - supports over 200 rules
  - possible to define additional rules
- **Metal**: C Code analyzer.
  - Similar, but takes memory problems into account

Limitations

- "True but low impact defects"
  - Deliberate e
  - Infeasible
  - Testing cod
- Only intra-procedural, [illegible]dural flow
- False Positive: someth[illegible] you label as correct, but its actually wrong

"Because its analysis is sometimes imprecise, FindBugs can report false warnings, which are warnings that do not indicate real errors. In practice, the rate of false warnings reported by FindBugs is less than 50%."

Future Work

- Add more types of bugs
- Make existing bug patterns more accurate

Works Cited

- Comparing Bug Finding Tools with Reviews and Tests (2005) Stefan Wagner, Jan Jürjens, Claudia Koller, Peter Trischberger, Technische Universität München In Proc. 17th International Conference on Testing of Communicating Systems (TestCom'05), volume 3502 of LNCS
- http://findbugs.sourceforge.net/
- Hovemeyer, D. and Pugh, W. 2004. Finding bugs is easy. In Companion To the 19th Annual ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages, and Applications (Vancouver, BC, CANADA, October 24 - 28, 2004). OOPSLA '04. ACM, New York, NY, 132-136. DOI= http://doi.acm.org/10.1145/1028664.1028717
- Ayewah, N., Pugh, W., Morgenthaler, J. D., Penix, J., and Zhou, Y. 2007. Evaluating static analysis defect warnings on production software. In Proceedings of the 7th ACM SIGPLAN-SIGSOFT Workshop on Program Analysis For Software Tools and Engineering (San Diego, California, USA, June 13 - 14, 2007). PASTE '07. ACM, New York, NY, 1-8. DOI= http://doi.acm.org/10.1145/1251535.1251536