Static Analysis

Black box testing, white box testing, testing types/strategies, test plans, test data, execution paths

Static analysis

Static analysis *is the process of evaluating a system or component based on its form, structure, content, or documentation [IEEE]*

- Purpose is to identify code that is likely to be problematic.
- Does not involve execution of the program.
- Some analysis can be done manually through code inspections, walkthroughs (identify where code doesn't meet specs).
- Some analysis is done via software metrics: cyclomatic complexity, size, depth of inheritance, etc. (identify areas with potential problems).
- Software tools are important! They can scan for violations of recommended programming practices.

Static analysis tools

- **EclEmma** (JaCoCo). Checks JUnit tests for code coverage.
- **FindBugs**. Looks for bug patterns.
- **CheckStyle**. Checks code against standards. Highly configurable.
- **PMD**. Looks for a variety of potential problems.

You can install all of these tools as eclipse plugins.

Tools are not perfect:

- False positives: bugs/issues that are not bugs/issues
- False negatives: actual bugs/issues that are missed
- Harmless bugs: Require manual check to resolve/ignore.

Code coverage

Code coverage is a measure of the degree to which code has been tested. It includes:

- **Method coverage** – Have all methods been called.
- **Statement coverage** – Have all statements been executed?
- **Condition/Decision coverage** – Have all conditions/decisions been executed with both true and false values?

Code coverage tool. **EclEmma** eclipse plugin.

Code coverage: benefits and limitations

Benefits:

1. A measure of how complete your test cases are.
2. Can identify paths through your code that you have missed testing.

Limitations:

1. High coverage does not guarantee code correctness.
2. Coverage does not tell you what requirements you missed.

Don't write test cases only to satisfy EclEmma.

EclEmma

EclEmma is an eclipse plugin built on EMMA, which is a Java code coverage tool. **JaCoCo** is an enhancement of EclEmma that runs on multiple development environments.

EclEmma measures (at the bytecode level) and reports coverage on the following:

- class
- method
- line
- basic block

EclEmma can detect when a single source code line is covered only partially.

CSC 216: Threshold is 80% line coverage per class.

CheckStyle

CheckStyle is an eclipse plugin that looks for style issues with your code. Style issue = where code does not meet departmental style guidelines.

- Style guidelines can be customized with a configuration file.
- Style checks: indentation, spacing, naming conventions, missing Javadoc comments, etc

Style is important.

- It helps others inspect your code.
- It helps YOU inspect your code.
- It makes code easier to maintain/modify.
- Documentation makes it easier/feasible for others to use your code.

PMD

PMD is an eclipse plugin that looks for style issues and potential problems like:

- Possible bugs - empty try/catch/finally/switch statements
- Dead code - unused local variables, parameters, private methods
- Suboptimal code - wasteful String/StringBuffer usage
- Overcomplicated expressions
- Duplicate code

PMD can be customized with a configuration file. CSC 216 uses a configuration file to:

- Enforce some style guidelines.
- Check for null pointer dereferences.
- Check for test methods without asserts.

FindBugs

FindBugs is an eclipse plugin that looks for serious and interesting problems based on bug patterns, including:

- Difficult/complicated language features
- Misunderstood API methods
- Ordinary mistakes: typos, wrong boolean operators, string comparison using ==, etc.

FindBugs can rate bugs. You should at a minimum remove bugs rated as:

- Scary
- Scariest

Bug examples

java 10

buggy3(int) : int

```java

public class Bug {

    public String buggy1(String s) {
        return this.buggy1(s + "a");
    }

    public String buggy2(String s) {
        return s.toUpperCase() + "xx";
    }

    public int buggy3(int value) {
        while (value < 27)
            System.out.println(value);
            value = value - 10;
        return value;
    }
}

```

References

- Sarah Heckman, CSC 216 slides.
- Heckman References:
  - Laurie Williams, "White-Box Testing," http://agile.csc.ncsu.edu/SEMaterials/WhiteBox.pdf
  - Brian Marick, "How to Misuse Code Coverage," http://www.exampler.com/testing-com/writings/coverage.pdf
  - Ayewah, Pugh, Hovemeyer, Morgenthaler, Penix, "Using Static Analysis to Find Bugs," IEEE Software, vol. 25, no. 5, 2008.
  - Ayewah, Pugh, Morgenthaler, Penix, Zhou, "Evaluating static analysis defect warning on production software," PASTE 2007.
  - Chess, Brian and McGraw, G. "Static Analysis for Security", *IEEE Security & Privacy*, Nov/Dec 2004.
  - Hovermeyer, David and Pugh, William, "Finding Bugs is Easy", OOPSLA 2004
  - Rutar, N., Almazan, C., and Foster, J., "A Comparison of Bug Finding Tools for Java", ISSRE 2004.