CSE 403
Lecture 11

Static Code Analysis

Reading:
IEEE Xplore, "Using Static Analysis to Find Bugs"

slides created by Marty Stepp
http://www.cs.washington.edu/403/
FindBugs

- **FindBugs**: Java static analysis tool that focuses on bugs and usage errors in code.
  - null pointers
  - useless/dead code
  - unclosed I/O streams
  - infinite loops
  - infinite recursion

- FindBugs has been run on the actual JDK 1.6 source, the Eclipse source, and many errors.
  - What kind of bugs and problems were found?
Checkstyle

- **Checkstyle**: A static analysis tool that focuses on Java coding style and standards.
  - whitespace and indentation
  - variable names
  - Javadoc commenting
  - code complexity
    - number of statements per method
    - levels of nested if/loops
    - lines, methods, fields, etc. per class
  - proper usage
    - import statements
    - regular expressions
    - exceptions
    - I/O
    - thread usage, ...
• Fairly essential, used on most large programming projects.

  – Why? Why not Makefiles or shell scripts instead?

  – What are these tools aiming to do?

  – What other tools integrate well with them?

  – What features would you want from an automated build tool?
Ant

- **Ant** ("another neat tool"): A Java build management tool.
  - developed by Apache to help build their Tomcat web server
  - expanded into a general tool

- Ant is a commonly used build tool for Java programs giving many more build options than the old "Make" utility.
  - built for Java, so it understands Java concepts like:
    - classpath,
    - javac, .class files,
    - JARs,
    - JUnit, etc.
An Ant Build File

• Similar to Make, but Ant uses `build.xml` instead of Makefile:

```xml
<project>
  <target name="name">
    <tasks>
    <target name="name">
      <tasks>
    </target>
  </target>
</project>
```

• A **task** can be a command such as:

  ```xml
  <javac ... />
  <mkdir ... />  
  <delete ... />
  ```

Ant build.xml Example

```xml
<project>
  <target name="clean">
    <delete dir="build"/>
  </target>

  <target name="compile">
    <mkdir dir="build/classes"/>
    <javac srcdir="src"
          destdir="build/classes"/>
  </target>
</project>
```
Ant Task Integration

- To integrate other tools with Ant, download **custom Ant tasks** for those tools.
  - JUnit Ant task
  - Checkstyle Ant task
  - FindBugs Ant task
  - ...

- Search for these, and instructions for adding them, on Google
JUnit Ant Task Example

```xml
<project>
  <property name="src" value="./src" />
  <property name="lib" value="./lib" />
  <property name="classes" value="./classes" />
  <property name="test.class.name" value="com.xyz.MyTestSuite" />
  <path id="test.classpath">
    <pathelement location="${classes}" />
    <pathelement location="/path/to/junit.jar" />
    <fileset dir="${lib}"
      <include name="**/*.jar" /></fileset>
  </path>
</project>

<!-- Define the Ant task for running JUnit: -->
<target name="test">
  <junit fork="yes" haltonfailure="yes">
    <test name="${test.class.name}" />
    <formatter type="plain" usefile="false" />
    <classpath refid="test.classpath" />
  </junit>
</target>

- on command line:  ant  test
• Ant integrates nicely with Eclipse.
  – You can set up a "Build", "Run", or "Debug" task that uses Ant.
  – Eclipse can create an Ant build file for you from an existing project that builds its code.
  – Eclipse also has an Ant build file editor:
**Maven**

- **Maven**: A project management, comprehension, and build tool.
  - A successor / replacement for Ant.
  - Made by Apache, makers of Ant.

- Differences from Ant:
  - more powerful; higher level of abstraction
  - great for generating reports and visualizations
  - can run integration tests and continuous integration *(seen later)*
  - can handle deployment of an app or site
Maven and Eclipse

- Since Maven is newer, tool support (e.g. Eclipse integration) was slower to arrive, but it is generally mature now