SEN3320/6320: Software Verification and Validation

School of Electrical Engineering and Computing

Semester 1, 2020

Code Review

Common Wisdom ?

- Programmers are smart
- Smart people don’t make dumb mistakes
- We have good techniques (e.g., unit testing) for finding bugs early
- So, bugs remaining in production code must be subtle, and finding them must require sophisticated techniques

Can You Find The Bug?

```
if (listeners == null)
listeners.remove(listener);
```

- Found in JDK1.6.0, b105, sun.awt.x11.XMSelection
- lines 243-244

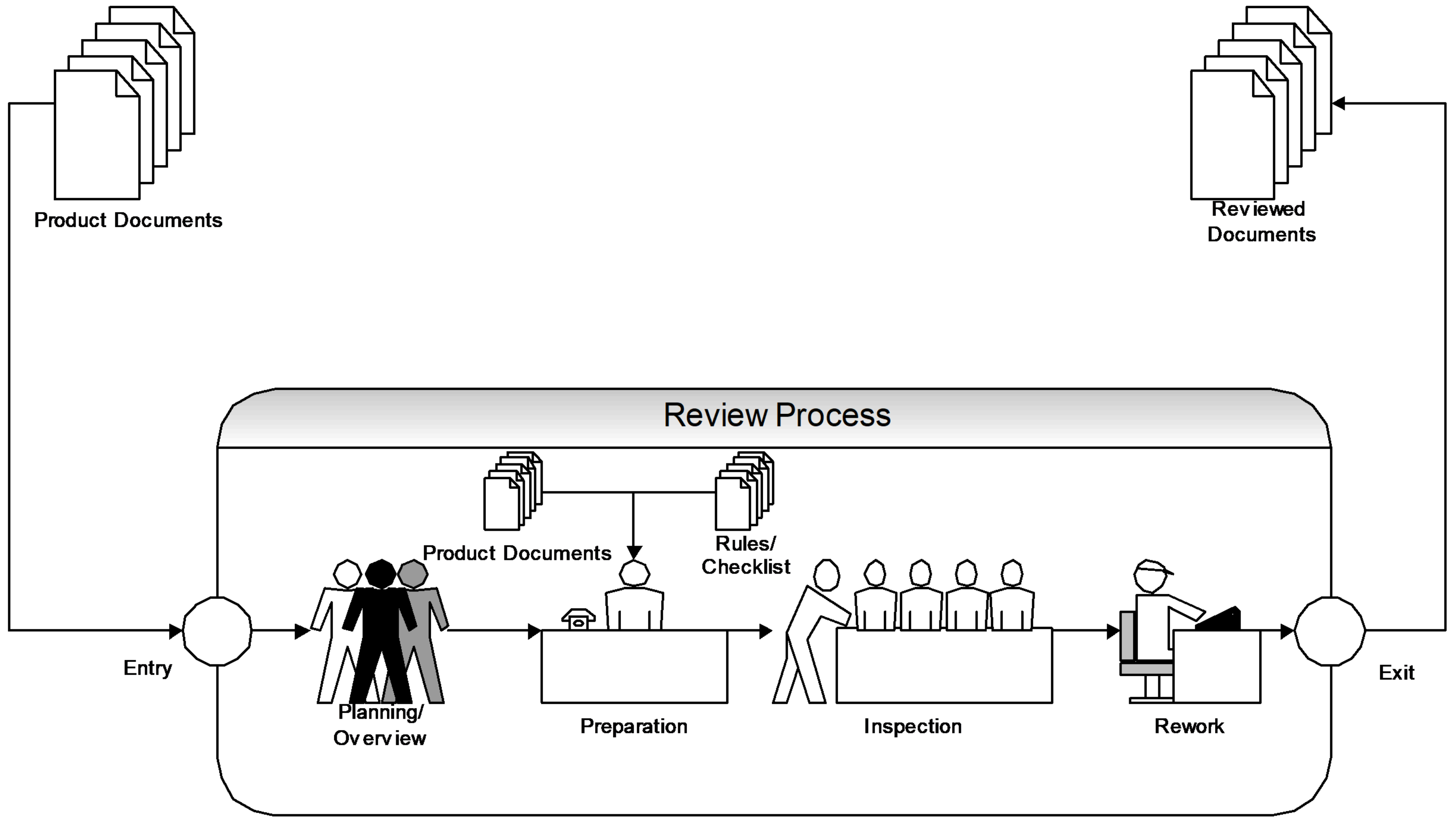
Software Reviews

- Software reviews are a “quality improvement processes for written material”.
- Goal: By detecting defects early, and preventing their leakage downstream, the higher cost of later detection and rework is eliminated.
- Software products that can be reviewed:
  - Software requirements specifications
  - Software design descriptions
  - Source code (Code Review)
  - Release notes
  - …

Types of Code Review

- Documented evidence that identifies:
  - Ad-hoc review
  - Pass-round
  - Walkthrough
  - Group review
  - Formal inspection

A Typical Formal Inspection Process



Code Review Basic Steps

- Perform an examination of the software products
- Detect:
  - Defects (bugs)
  - Violation of coding standards
  - Code smells
  - Other problems
- Look for code patterns that indicate problems based on prior experience.
- Static analysis tools can also help with code review.

Bug Patterns (Infinite recursive loop)

A student’s assignment may contain a bug:

```
public WebSpider() {
  WebSpider w = new WebSpider();
}
```

Bug Patterns (Infinite recursive loop)

Found 5 instances of the same bug pattern (Infinite recursive loop)

- Including one written by Joshua Bloch

```
public String foundType() {
   return this.foundType();
}
```

- 27 across all versions of JDK, 40+ in Google’s Java code

Bug Patterns (null pointer bugs)

//com.sun.corba.se.impl.naming.cosnaming.NamingContextImpl
```
if (name != null || name.length > 0)
```

//com.sun.xml.internal.ws.wsdl.parser.RuntimeWSDLParser
```
if (part == null | part.equals(""))
```

// sun.awt.x11.ScrollPanePeer
```
if (g != null)
    paintScrollBars(g,colors);
g.dispose();
```

Bug Patterns (null pointer bugs)

Does this code contain a null pointer bug?

```
String s = null;
if (x > 0) s = “x”;
if (y > 0) s = “y”;
return s.hashCode();
```

Bug Patterns (SQL Injection)

System is vulnerable when user input is passed without tests

```
String query = "select * from user where username='";
query += req.getParameter("userID");
query += "' and password = '"+req.getParameter("pwd")+“’”;
```

```
SELECT * FROM user WHERE username='' OR 'a'='a' --AND password=‘’”;
```

More Bug Patterns /1

- Common behaviors of all software
  - a/b => b!=0 — Divide by 0
  - a.field => a!=null — Null Pointer Reference
  - a[x] => x<a.length() — Buffer Overflow
  - p.malloc() => p.free() — Memory Leak
  - lock(s) => unlock(s) — deadlock
  - while(x<0) => F(x>=0) — Infinite Loop
  - <script> xxx </script> => ! User_input -> xxx — XSS

More Bug Patterns /2

- Common Java Mistakes and Bug Patterns

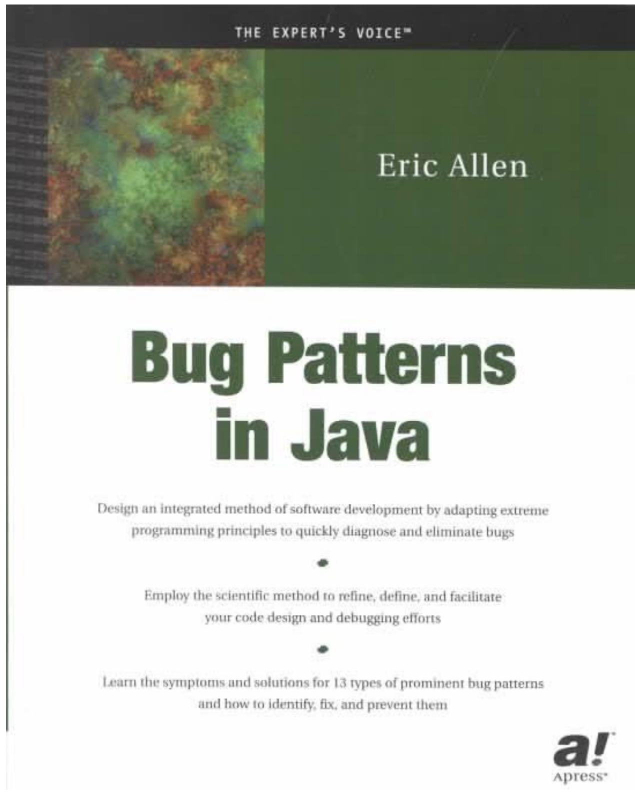
http://www.flexbit.com/resources/html/jmistakes.htm#No14

- Vulnerabilities related bug patterns

https://find-sec-bugs.github.io/bugs.htm

- Book: Bug Patterns in Java

https://www.apress.com/gp/book/9781590590614


THE EXPERT'S VOICE™

Eric Allen

Bug Patterns in Java

Design an integrated method of software development by adapting extreme programming principles to quickly diagnose and eliminate bugs

•

Employ the scientific method to refine, define, and facilitate your code design and debugging efforts

•

Learn the symptoms and solutions for 13 types of prominent bug patterns and how to identify, fix, and prevent them

a! Apress®

Code Review – Checking Code Smell

- Code smells are indications of poor coding and design choices that can cause problems during the later phase of software development.
- A code smell is a hint that something has gone wrong somewhere in your code. Use the smell to track down the problem. (http://wiki.c2.com/?CodeSmell)

Duplicated code

```
extern int a[];
extern int b[];
int sumofa = 0;
for (int i = 0; i < 4; i + +)
sum += a[i];
int averageofa= sum/4;
————————-
int sumofb = 0;
for (int i = 0; i < 4; i + +)
sum += b[i];
int averageofb = sumofb/4;
```

Example: Duplicate Conditional Fragments:

The same fragment of code is in all branches of a conditional expression. Refactor by moving it outside of the expression.

```
if (isSpecialDeal()) {
  total = price * 0.95;
         send();
       } else {
  total = price * 0.98;
         send();
}
```

```
if (isSpecialDeal())
 total = price * 0.95;
         else
 total = price * 0.98;
       send();
```

A List of Code Bad Smells

- Duplicated Code
- Long Method
- Large Class (Too many responsibilities)
- Long Parameter List (Object is missing)
- Feature Envy (Method needing too much information from another object)
- Lazy Class (Do not do too much)
- Middle Man (Class with too much delegating methods)
- Temporary Field (Attributes only used partially under certain circumstances)
- Message Chains (Coupled classes, internal representation dependencies)
- Data Classes (Only accessors)
- …

Checking Coding Conventions /1

- Your code should be readable!
- Formatting conventions ensure *consistency* and therefore, familiarity for readers
- Indentation: Indent when starting out new blocks of code

Sun's Coding Conventions for Java
–*http://www.oracle.com/technetwork/java/codeconvtoc-136057.html*

JavaScript Coding Style
–https://google-styleguide.googlecode.com/svn/trunk/javascriptguide.xml

Code Conventions for the JavaServerPages Technology
–http://java.sun.com/developer/technicalArticles/javaserverpages/code_convention/

Checking Coding Conventions /2

Checking identifier names:

- Choosing meaningful names
- Class names start upper-cased
  – e.g., BankAccount, Vehicle, VehicleApplet
- Variable and Method names *start*lower-cased
  - e.g., aliceAccount, hondaCivic, nCount, etc.
- Constants are ALL_CAPS and words in name are separated by an underscore
  – e.g., PI, MAX_WIDTH, DEFAULT_WIDTH

Checking Coding Conventions /3

• Block-style comments

/*

* This is a multi-line comment.

* Use when you need to write a long comment about a fragment

*/

Checking comments

• One-line comments

–/* C-style comments */

• use to put descriptive notes *before* a code fragment

–// C++ style comment

- use at the end of the line, to describe variables or short pieces of code
- also use for commenting-out code

• javadoc comments

– like block-style, but starts with /** instead

– use immediately before classes, methods, and fields

Industry Experience with Reviews

- Aetna Insurance Company:
  - Found 82% of errors, 25% cost reduction.
- Bell-Northern Research:
  - Inspection cost: 1 hour per defect.
  - Testing cost: 2-4 hours per defect.
  - Post-release cost: 33 hours per defect.
- Hewlett-Packard
  - Est. inspection savings (1993): $21,454,000

Benefits: Quality Improvement

- Reviews can find 60-100% of all defects.
- Review data can assess/improve quality of:
  - Work product.
  - Software development process.
  - Review process itself.
- Reviews reduce total project cost, but have non-trivial cost (~15%).
- Early defect removal is 10-100 times cheaper.
- Reviews distribute domain knowledge, development skills, and corporate culture.

Common Problems in Code Review

- Insufficient Preparation
- Moderator Domination
- Incorrect Review Rate
- Ego-involvement and Personality Conflict
- Issue Resolution and Meeting Digression
- Recording Difficulties and Clerical Overhead

Code Review with Static Analysis

- Static Analysis
    - Analyses your program without executing it
    - Doesn’t depend on any test cases
    - Generally, doesn’t know what your software is supposed to do
    - Looks for bug patterns
    - Not a replacement for testing
    - But many defects can’t be found with static analysis

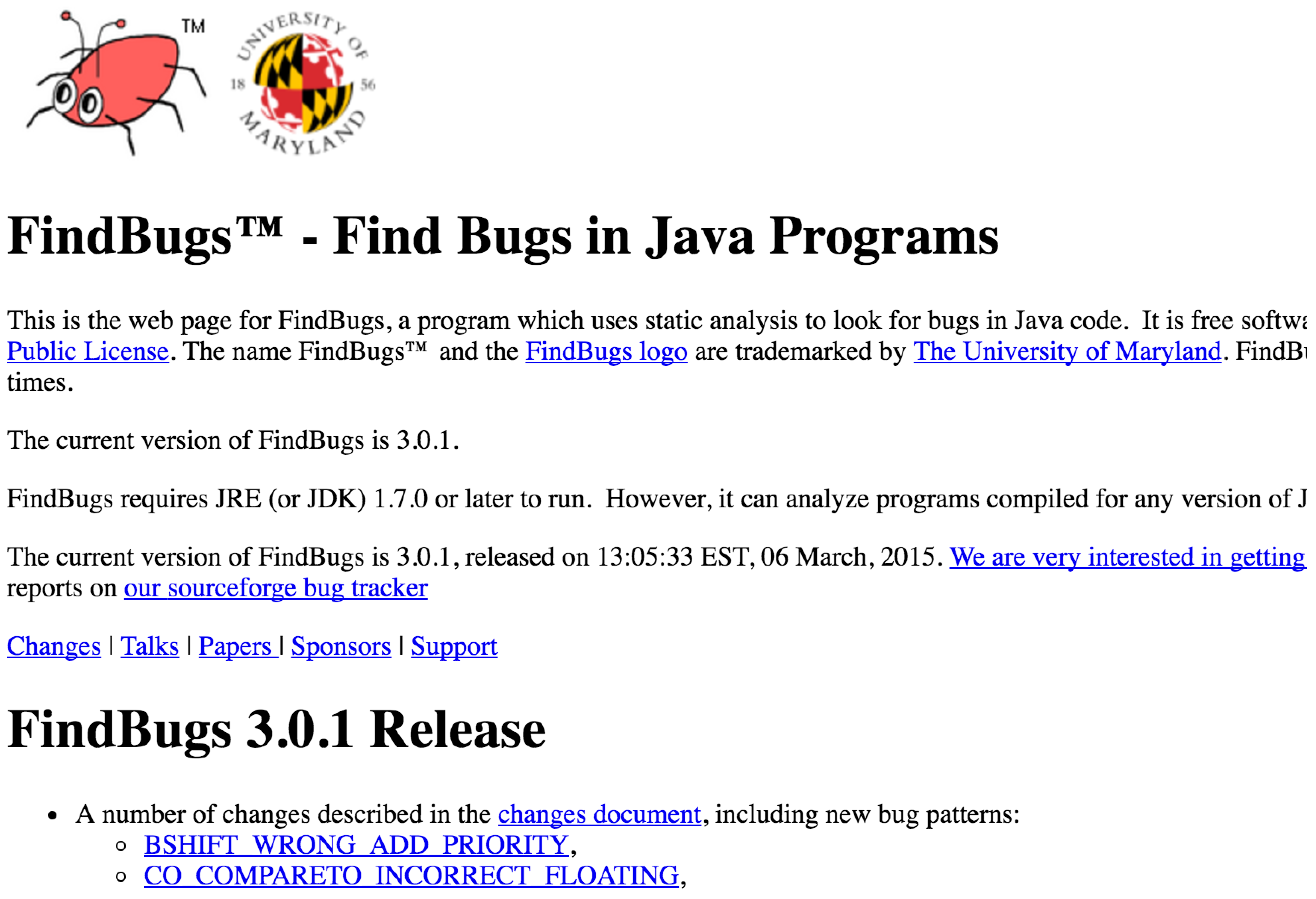
FindBugs

- License: Lesser GNU Public License (LGPL)
- Author: The University of Maryland
- Statistics: downloaded more than a million times
- Current version: 3.0.1.
- Homepage: http://findbugs.sourceforge.net/

Patterns to be checked

- 400+ code patterns
  - Bad Practice
  - Correctness
  - Performance
  - Dodgy code
  - Vulnerability to malicious code
- Examples (http://findbugs.sourceforge.net/bugDescriptions.html)
  - Equals method should not assume type of object argument
  - Collection should not contain themselves: !(s.contains(s))
  - Should not use string.tostring()
  - ...

Demo



TM

FindBugs™ - Find Bugs in Java Programs

This is the web page for FindBugs, a program which uses static analysis to look for bugs in Java code. It is free softwa
Public License. The name FindBugs™ and the FindBugs logo are trademarked by The University of Maryland. FindB
times.

The current version of FindBugs is 3.0.1.

FindBugs requires JRE (or JDK) 1.7.0 or later to run. However, it can analyze programs compiled for any version of J

The current version of FindBugs is 3.0.1, released on 13:05:33 EST, 06 March, 2015. We are very interested in getting
reports on our sourceforge bug tracker

Changes | Talks | Papers | Sponsors | Support

FindBugs 3.0.1 Release

- A number of changes described in the changes document, including new bug patterns:
  - BSHIFT_WRONG_ADD_PRIORITY,
  - CO_COMPARETO_INCORRECT_FLOATING,

Success Stories

- Google runs FindBugs over all Java code in their main repository
  - 1,663 issues identified and reviewed
  - 1,190 reported as bugs to developers
  - 805 fixed by developers
- EBay uses 2 developers to audit/review FindBugs warnings
  - 10 times more effective at finding P1 bugs than using two testers
  - Has put a lot of work into reviewing which bug patterns are significant for EBay

More static bug detection tools

- Lint (http://www.unix.com/man-page/FreeBSD/1/lint)
  - A code style enforcing tool for C language
  - Find bad coding styles and raise warnings
    - Bad naming
    - Hard coded strings
    - …
- PMD Findbugs (https://pmd.github.io/)
  - An extensible cross-language static code analyzer.
  - It finds common programming flaws like unused variables, empty catch blocks, unnecessary object creation, and so forth.
  - Additionally it includes CPD, the copy-paste-detector, to find duplicated code.
- A list of static code analysis tools:

  https://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis

References

- IEEE Standard for Software Reviews, IEEE Std 1028-1997, 1998 pp 1-26, Annex B
- Carolyn Seaman, IFSM 498/698D and CMSC 491/691S System Maintenance Lecture Notes, UMBC, 2002.
- "A Comparison of Bug Finding Tools for Java", by Nick Rutar, Christian Almazan, and Jeff Foster, University of Maryland.
- David Hovemeyer and William Pugh. 2004. Finding bugs is easy. SIGPLAN Not. 39, 12 (December 2004), 92-106.
- Finding More Null Pointer Bugs, But Not Too Many, by David Hovemeyer, York College of Pennsylvania and William Pugh, Univ. of Maryland, 7th ACM SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering, June, 2007

Thanks!

Testing
Software
development
quality
test
design
articles
management
risk
good
People
Project
information
Agile