

JUnit Test Runner Sequence

- Test runner is given a list of test classes
- For each test class
 - Create an instance of the test class
 - For each test*() method
 - Run setUp() method
 - Run test method steps and checks
 - If a check fails, an assertion is thrown and the test method fails
 - Run tearDown() method
- Test runner produces a report
- Some test runners work interactively

Things to Notice

- The order of the test methods is not determined
- setUp() method makes some variables that are used in the tests
 - Officially called “fixtures”
- tearDown() frees memory, prevents results of one test from affecting the next
- Different types of checks
 - Using assert*
 - Using fail() to check that an exception is thrown
- Missing test cases: a new cart should be empty, add the same product twice, remove a product that was already removed, test isEmpty(), etc.

Setting Up JUnit in Eclipse JDT

- Add junit.jar to the project's build path.
 - Locate junit.jar in the eclipse/plugins directory hierarchy
- Create a new JUnit TestCase Wizard
 - File->New->Other...->Java->JUnit->Test Case
 - A Test Suite is a selection of existing Test Cases
 - Easier than creating a class and subclassing from TestCase
- Running a JUnit test case
 - Run->Run as...->JUnit Test the first time
 - After that, a Run task can be used

More Information

- **Eclipse Help**

- Help -> Help Contents -> Java Development User Guide -> Getting Started -> Basic Tutorial -> Writing and running JUnit tests

- **JUnit Home Page**

- <http://www.junit.org>

- **JUnit Primer**

- <http://www.clarkware.com/articles/JUnitPrimer.html>

Error-Based Techniques

- Equivalence Partitioning
 - Discussion topic
- Some heuristics

Common Time and Date Errors

- Y2K
- mm/dd/yyyy (the US) vs dd/mm/yyyy (rest of the world)
- 24 hour vs. 12 hour clock
- Formatting
 - Day 36 of a month?
 - Hour 27 of a day?
- Leap Years and Leap Days
 - Leap seconds

Testing with exceptions

- Unit tests involving exceptions

```
public void testRemoveItem(){
    try {
        _testCart.removeItem(_secondItem);
        fail("Should raise a product not found
exception");
    }
    catch (ProductNotFoundException success){

    }
}
```

More Time and Date Problems

- **Converting from Julian to Gregorian calendar**
 - In 1582, Pope Gregory XIII decreed that the calendar would be moved up 10 days and the new year will begin on January 1 instead of March 25
- **Time Zones**
 - International Dateline
- **Daylight Saving Time vs. Standard Time**
 - Some locations change some don't
 - Exact time and date of change
 - Missing and extra hours
 - Northern hemisphere vs. Southern hemisphere

Where do bugs hide?

- Escaping characters when converting between encoding schemes
 - HTML <-> Java <-> Database
 - Always test name fields with an Irish name
- Threads
 - When you get different results on the same input
- Memory
 - Loose fragments, overwrites, pointer arithmetic
 - Less of a problem in Java
- Delocalized plans
 - When different parts are implemented far apart from each other
- Incrementing loop variables
 - Off by one errors

Fault-Based Techniques

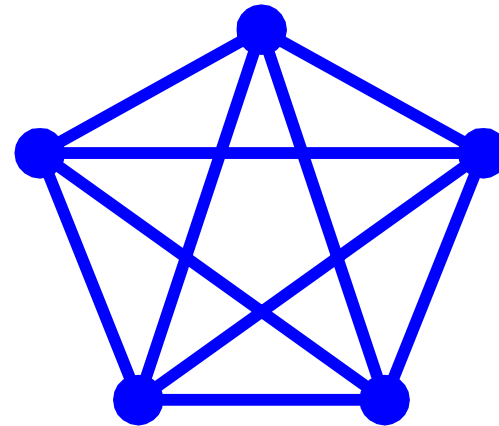
- **Fault Seeding or bebugging**
 - Method
 - Seed the program with a known number of faults
 - Perform testing
 - Calculate proportion of seeded faults uncovered
 - Extrapolate to real faults
 - Source of faults
 - Hand generated, but these faults may not be representative
 - Use two groups to perform testing, seed one group with the other's faults
 - Interpreting the proportion
- **Mutation Testing**
 - See van Vliet 13.6.2

Coverage-Based Techniques

- Coverage is expressed in terms of how much of the software work product has been covered by testing
 - Percentage of statements, paths, branches, etc.
 - Percentage of requirements
- Control-Flow Coverage
- Data-Flow Coverage
- Both are based on turning the work product into a graph

Graphs

- A graph consists of a set of nodes and edges
 - $G = \{N, E\}$
 - Nodes are the dots, or “things”
 - Edges connect the dots
- Examples:
 - Dot to dot pictures
 - Binary search trees
 - PERT charts
 - UML diagrams
- Area of study in mathematics and computer science
 - Four colour problem
 - Common data structure
 - Trees and lists are just graphs with extra rules



Constructing a Control Flow Graph

```
1 for (i = 0; i<n; i++) {
2     if (a[i] == true ){
3         System.out.println("1");
4     }
5     else {
6         System.out.println("0");
7     }
8 }
```

- Turn statements into nodes
- Add edges between nodes if you can traverse directly from head to tail
- Straight line sequences of nodes sometimes replaced by a single node
- Conditions have multiple edges coming out of them

Manual Test Techniques

- Code Reading
- Walkthroughs and Inspections

Inspections

- More effective than testing
 - Faster, more and different defects, identify origin of problems, not just symptoms
 - Can be done early in the development process
- Can inspect different software artifacts
 - Requirements
 - Design
 - Source Code
 - Test Cases
 - Data

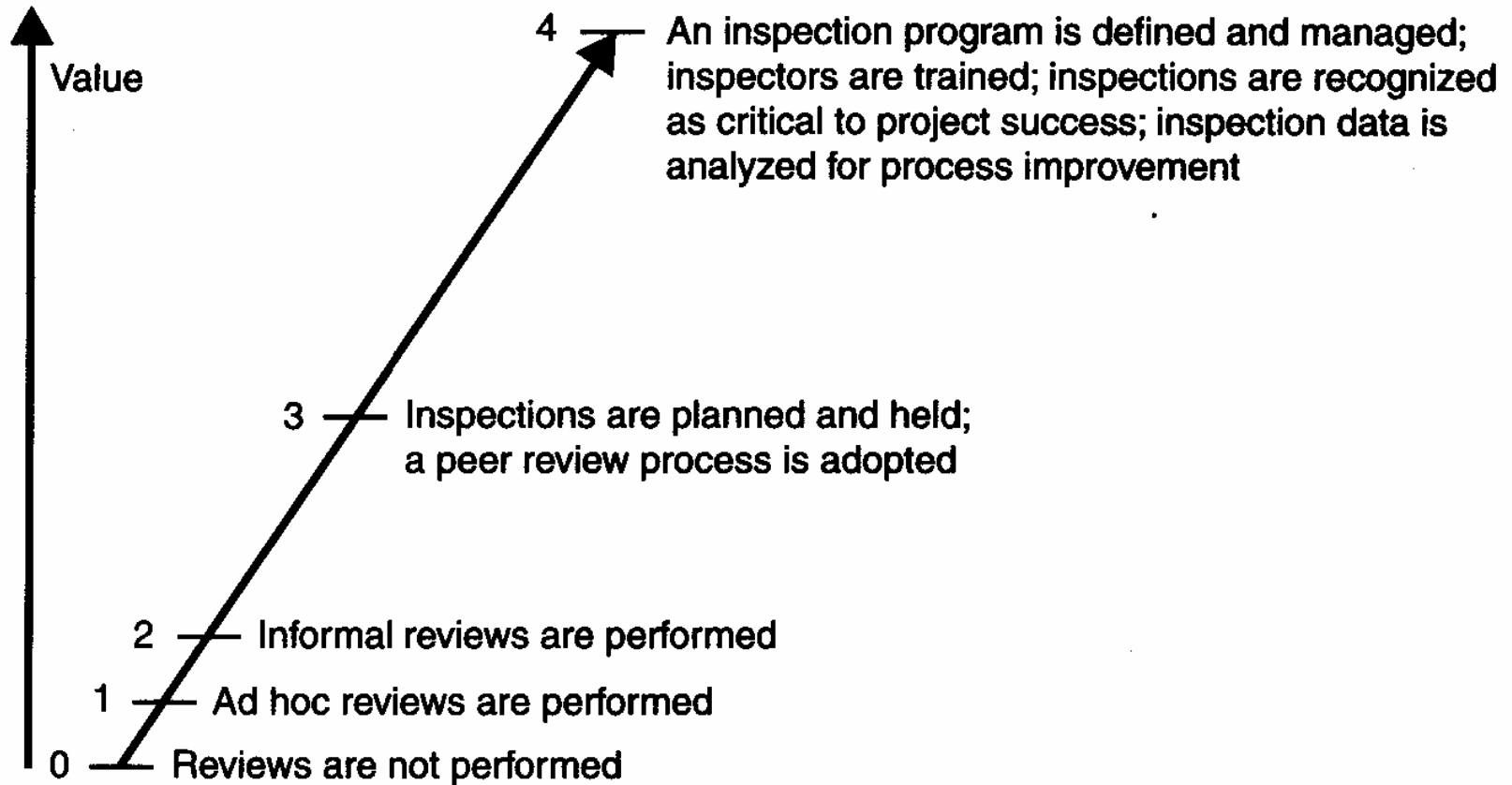
Why Do Reviews?

- Education
 - Learning from each other
 - Looking at examples with commentary
- Team Building
- Increase quality
 - More effective than testing for finding defects
 - Testing finds symptoms, reviews find problems
 - People are still smarter than computers
- Can test software work products other than code
 - Requirements, design, specification

Types of Peer Reviews

- Ad hoc
 - Can you come look at this?
- Peer deskcheck
 - All team members look at a piece of code
- Pair Programming
- Walkthrough
 - Author explains code to rest of team
- Team Review
 - Multiple team members review code in detail
 - Less formal and structured than inspections

Inspections



Targets of Inspections

- Can look for different things when inspecting
 - Sometimes focus on one, sometimes include many
- Logical Errors
- Consistency
- Style
- Error trapping
- Assumptions
- Interface
- Memory Management
- Race and deadlock conditions

Fagan Inspections

- **Planning**
 - Identify deliverable to be inspected, objectives, participants
- **Overview**
 - Kick-off meeting
- **Preparation**
 - Participants work independently to read code, look at documentation
- **Meeting**
 - Participants meet to compare notes, discuss code
- **Rework**
 - Corrections are made to code
- **Follow-up**
 - Check that “concerns” identified in review are addressed
- **Causal Analysis**
 - Find root causes of defects to find other similar ones

Fagan Inspections

- **Planning**
 - Identify deliverable to be inspected, objectives, participants
- **Overview**
 - Kick-off meeting
- **Preparation**
 - Participants work independently to read code, look at documentation
- **Meeting**
 - Participants meet to compare notes, discuss code
- **Rework**
 - Corrections are made to code
- **Follow-up**
 - Check that “concerns” identified in review are addressed
- **Causal Analysis**
 - Find root causes of defects to find other similar ones

Meeting

- **Moderator**
 - Chairs the meeting, keeps things moving
 - Promotes positive tone
 - Referees disputes
- **Reader**
 - Presents a section of the code using her/his own words
 - Paraphrase or interpretation
- **Recorder**
 - Logs concerns raised
 - Creates summaries and report after meeting
- **Author**
 - Learns from constructive criticism
 - Less active role

Meeting

- Typical inspection meeting will check 100–300 LOC/hour
- For Lab 6, participants will take turns as Moderator, Reader, and Author
- Everyone will be Recorder

Rework

- Issues identified during inspection are addressed in the work product
- For Lab 6, correct the code so it follows the coding convention and contains appropriate comments

Why Do Reviews

- Fault Detection Efficiency

	% design faults found	% coding faults found	combined
Design review	54	–	54
Code review	33	84	64
Testing	38	38	38

- Cost Effectiveness

Design Review	Code Review	Testing
8.44	1.38	0.17

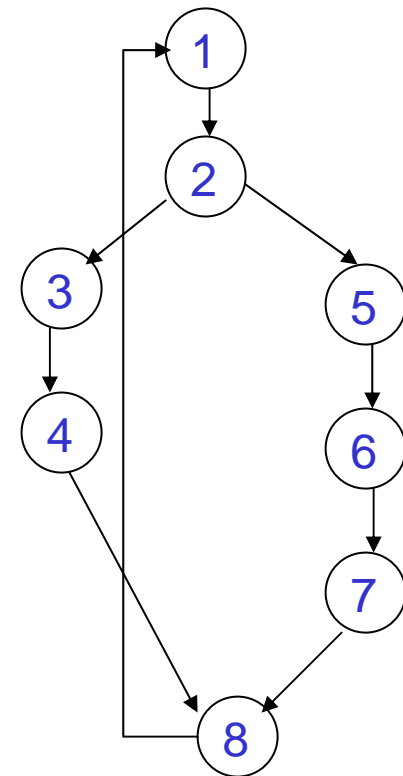
Why Reviews Aren't Done

- No time (front-loading of costs)
 - But there's time to fix the bugs?
- Poor software engineering culture
 - Myth: Only bad coders need reviews
 - Lack of experience
- Worried about what other people will think
 - Defensiveness, possessiveness, job security
- Code reviews are typically used only at top software technology organizations

Types of Control Coverage

- All-Paths

- The set of tests collectively traverse all possible paths
- How many possible paths in this graph?



More Types of Control Coverage

- All–Nodes or Statement coverage
 - Every statement in the program is executed by the test set
 - BUT, it's possible to have 100% statement coverage with a particular set of test data and still have an incorrect program
 - Example: arithmetic operator change
 - $a[i] \geq a[i-1]$ vs. $a[i] = a[i-1]$