Informatics 43

LECTURE 8-2

EMILY NAVARRO

Last Time

- Black box testing uses specifications to derive test cases
- We use equivalence class partitioning and boundary value analysis to choose test cases that guarantee a wide range of coverage
  - Typical values, boundary values, special cases, invalid input

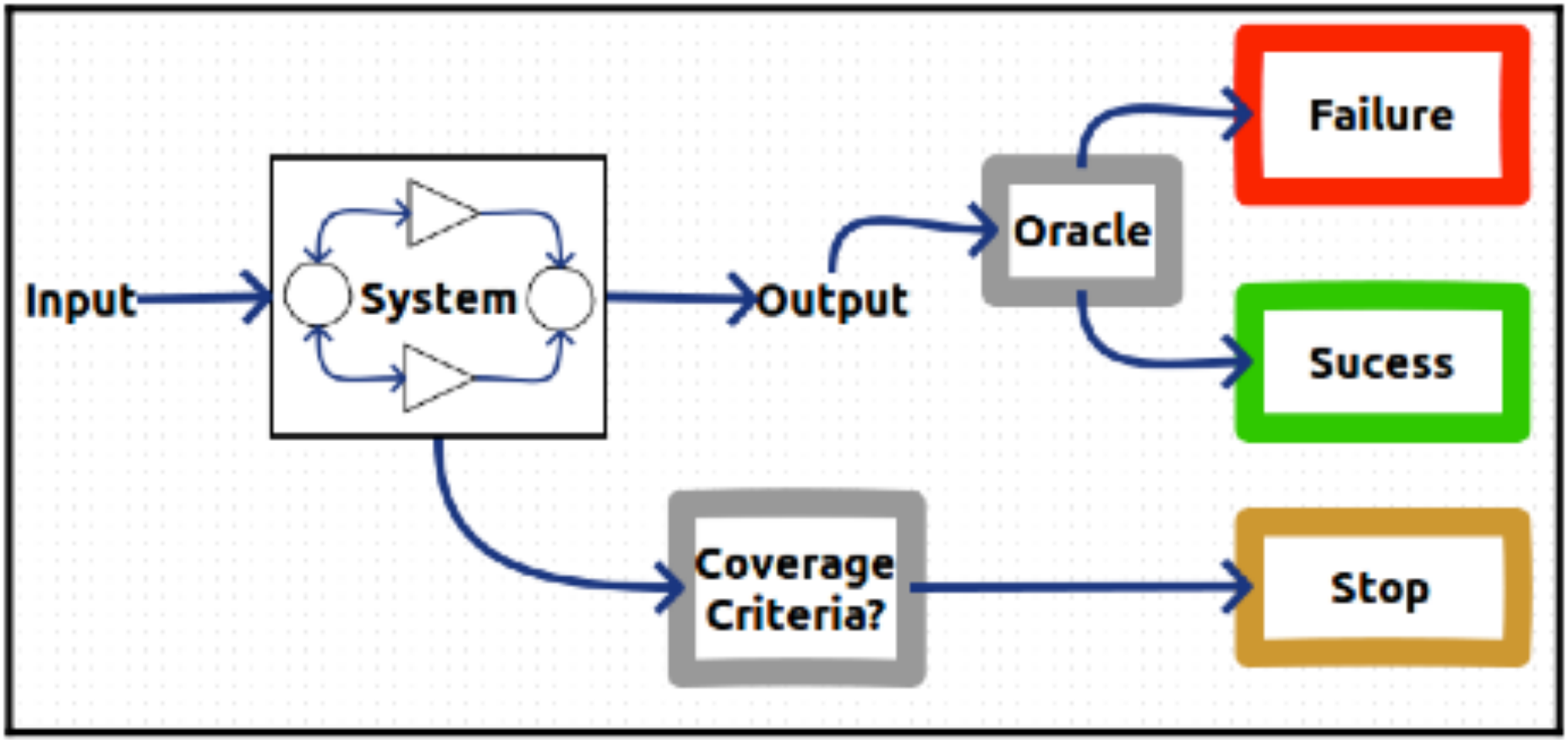
Today's Lecture

- White-box (Structural) Testing
- Miscellaneous testing topics
- Homework 3
- Quiz 5 study guide

Today's Lecture

- White-box (Structural) Testing
- Miscellaneous testing topics
- Homework 3
- Quiz 5 study guide

White-box / Structural Testing

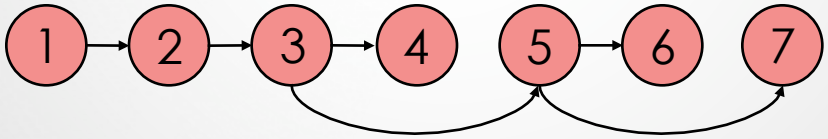


White-box / Structural Testing

- Use source code to derive test cases
  - Build a graph model of the system
  - State test cases in terms of graph coverage
- Choose test cases that guarantee different types of coverage
  - Node/statement coverage
  - Edge/branch coverage
  - Loop coverage
  - Condition coverage
  - Path coverage

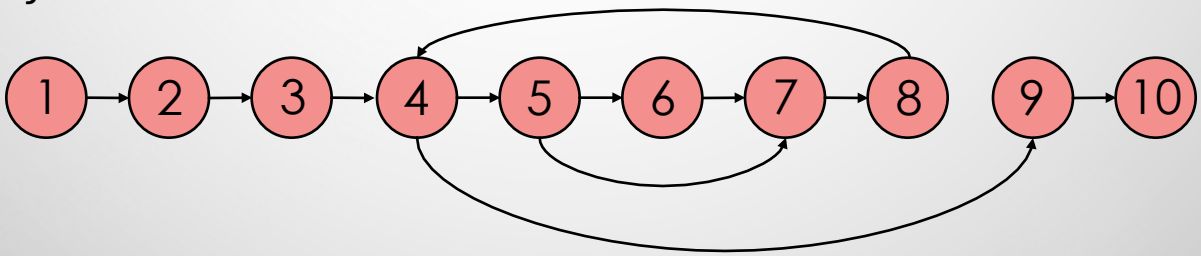
Example: Building the program graph

```
1  Node getSecondElement() {
2     Node head = getHead();
3     if (head == null)
4        return null;
5     if (head.next == null)
6        return null;
7     return head.next.node;
8  }
```



Example: Averaging quiz grades

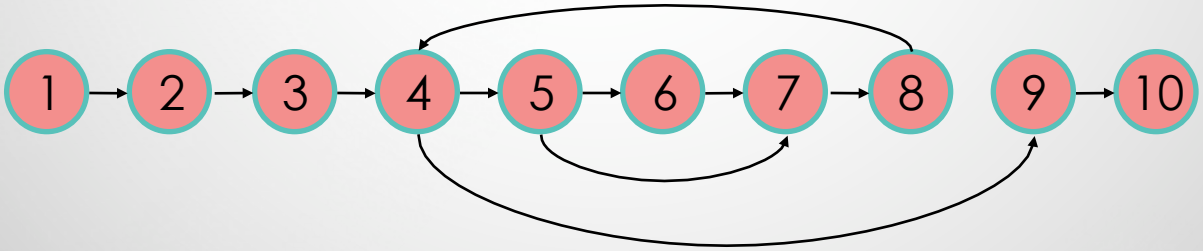
```
1   float quizAverage(float[] scores) {
2      float min = 99999;
3      float total = 0;
4      for (int i = 0 ; i < scores.length ; i++) {
5         if (scores[i] < min)
6            min = scores[i];
7         total += scores[i];
8      }
9      total = total – min;
10     return total / (scores.length – 1);
11  }
```



Node Coverage

- Select test cases such that every node in the graph is visited
  - Also called statement coverage
- Selects minimal number of test cases

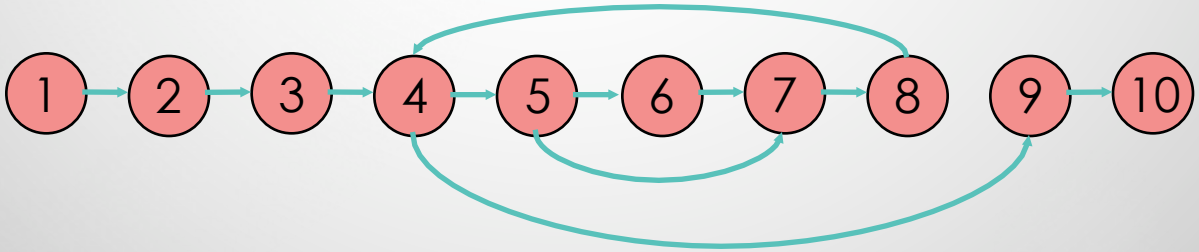
Test case: { 2 }



Edge Coverage

- Select test cases such that every edge in the graph is visited
  - Also called branch coverage
- More thorough than node coverage
  - More likely to reveal logical errors

Test case: { 1, 2 }

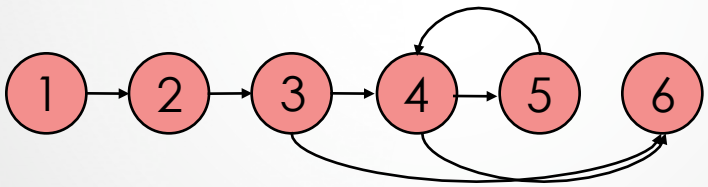


Another White Box Example

```
1 a = 17
2 read b from console
3 if a < b
4     while a < b-3
5         a = a / (b-50)
6 print a, b
```

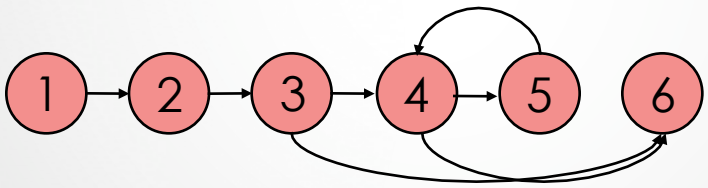
Another White Box Example

```
1 a = 17
2 read b from console
3 if a < b
4     while a < b-3
5         a = a / (b-50)
6 print a, b
```



Another White Box Example

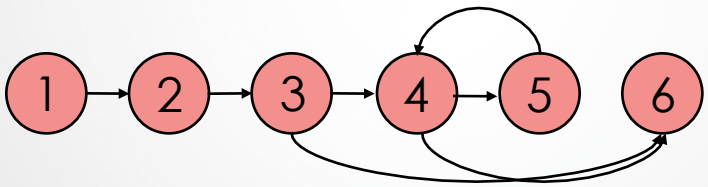
```
1 a = 17
2 read b from console
3 if a < b
4     while a < b-3
5         a = a / (b-50)
6 print a, b
```



What test cases are required to make sure every line of code is executed at least once? (Node coverage)

Another White Box Example

```
1 a = 17
2 read b from console
3 if a < b
4     while a < b-3
5         a = a / (b-50)
6 print a, b
```



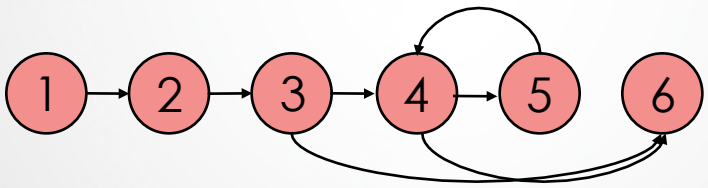
What test cases are required to make sure every line of code is executed at least once? (Node coverage)

b > 17 && b-3 > 17

or, b >= 21

Another White Box Example

```
1 a = 17
2 read b from console
3 if a < b
4     while a < b-3
5         a = a / (b-50)
6 print a, b
```

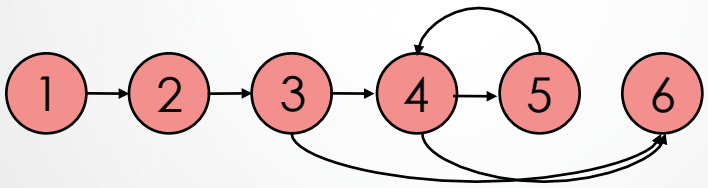


What test cases are required to make sure every line of code is executed at least once? (Node coverage)

Test case: { 21 }

Another White Box Example

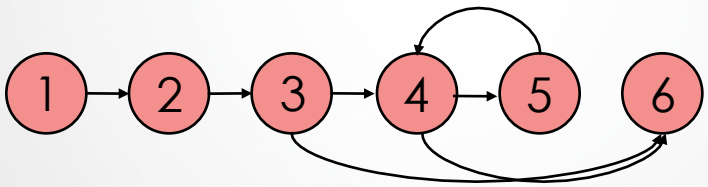
```
1 a = 17
2 read b from console
3 if a < b
4     while a < b-3
5         a = a / (b-50)
6 print a, b
```



What test cases are required to make sure every branch is taken at least once? (Edge coverage)

Another White Box Example

```
1 a = 17
2 read b from console
3 if a < b
4     while a < b-3
5         a = a / (b-50)
6 print a, b
```



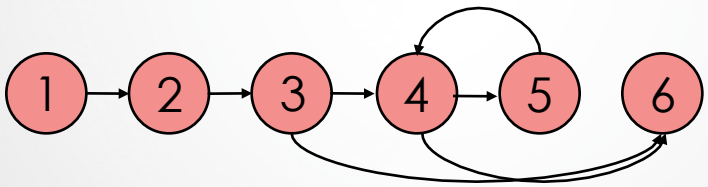
What test cases are required to make sure every branch is taken at least once? (Edge coverage)

b > 17, b <= 17

b-3 > 17, b-3 <= 17

Another White Box Example

```
1 a = 17
2 read b from console
3 if a < b
4     while a < b-3
5         a = a / (b-50)
6 print a, b
```

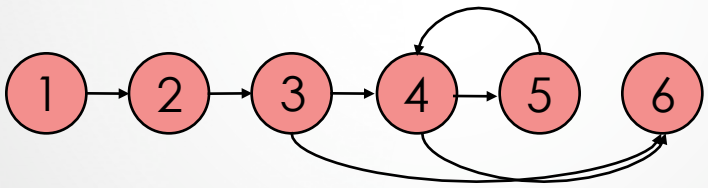


What test cases are required to make sure every branch is taken at least once? (Edge coverage)

Test cases: { 17 }, { 21 }

Another White Box Example

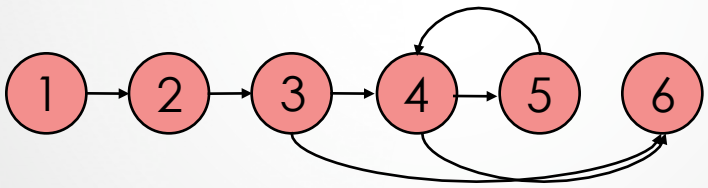
```
1 a = 17
2 read b from console
3 if a < b
4     while a < b-3
5         a = a / (b-50)
6 print a, b
```



What test cases are required to make sure that all possible exceptions are thrown? (Fault injection)

Another White Box Example

```
1 a = 17
2 read b from console
3 if a < b
4     while a < b-3
5         a = a / (b-50)
6 print a, b
```

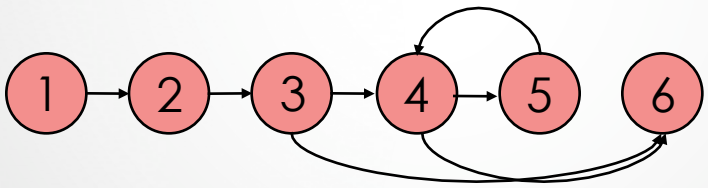


What test cases are required to make sure that all possible exceptions are thrown? (Fault injection)

b – 50 = 0

Another White Box Example

```
1 a = 17
2 read b from console
3 if a < b
4     while a < b-3
5         a = a / (b-50)
6 print a, b
```



What test cases are required to make sure that all possible exceptions are thrown? (Fault injection)

Test case: { 50 }

Other Coverage Criteria

- Loop coverage
  - Select test cases such that every loop *boundary* and *interior* is tested
- Condition coverage
  - Select test cases such that all conditions are tested
    - if (a > b || c > d) …

Other Coverage Criteria

- Path coverage
  - Select test cases such that every path in the graph is visited
  - Loops are a problem
    - 0, 1, average, max iterations
  - Most thorough…
  - …but is it feasible?

Challenges with White-box Testing

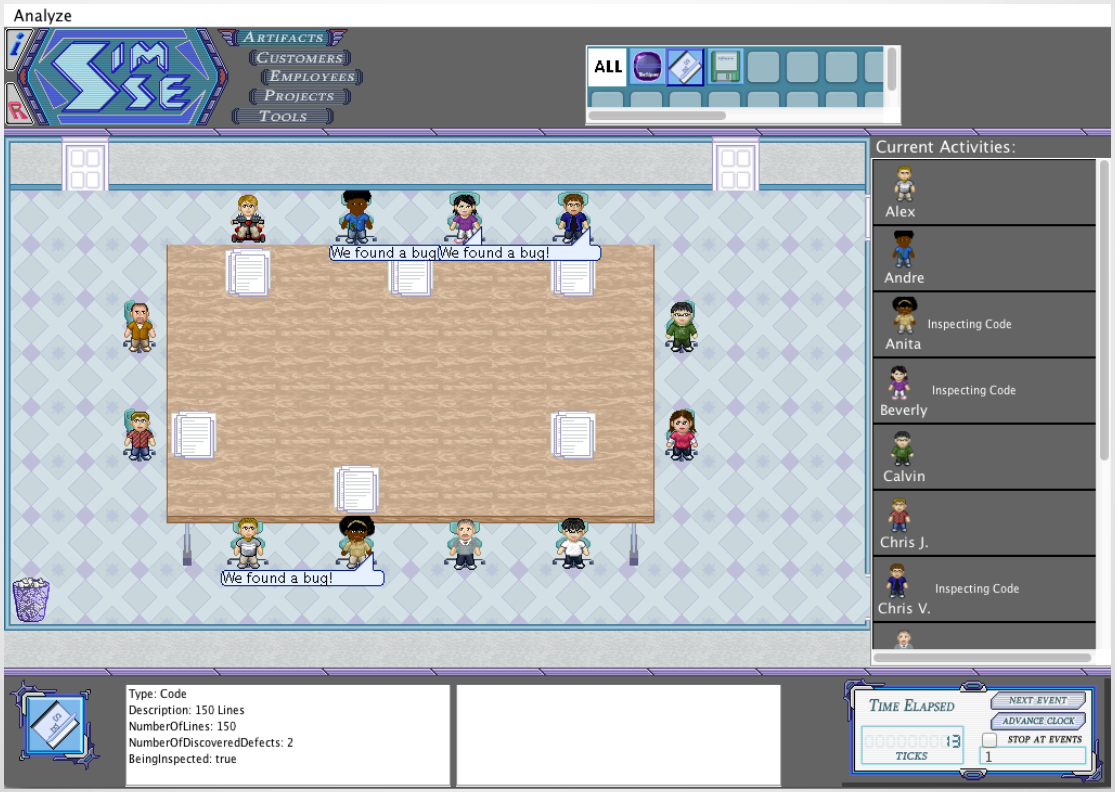
- Can be useful for identifying under-tested parts of a program
- Can cover all nodes or edges without revealing obvious faults
- Some nodes, edges, or loop combinations may be infeasible

Today's Lecture

- White-box (Structural) Testing
- Miscellaneous testing topics
- Homework 3
- Quiz 5 study guide

Inspections and Reviews

- Humans read documents and look for defects
- Surprisingly effective



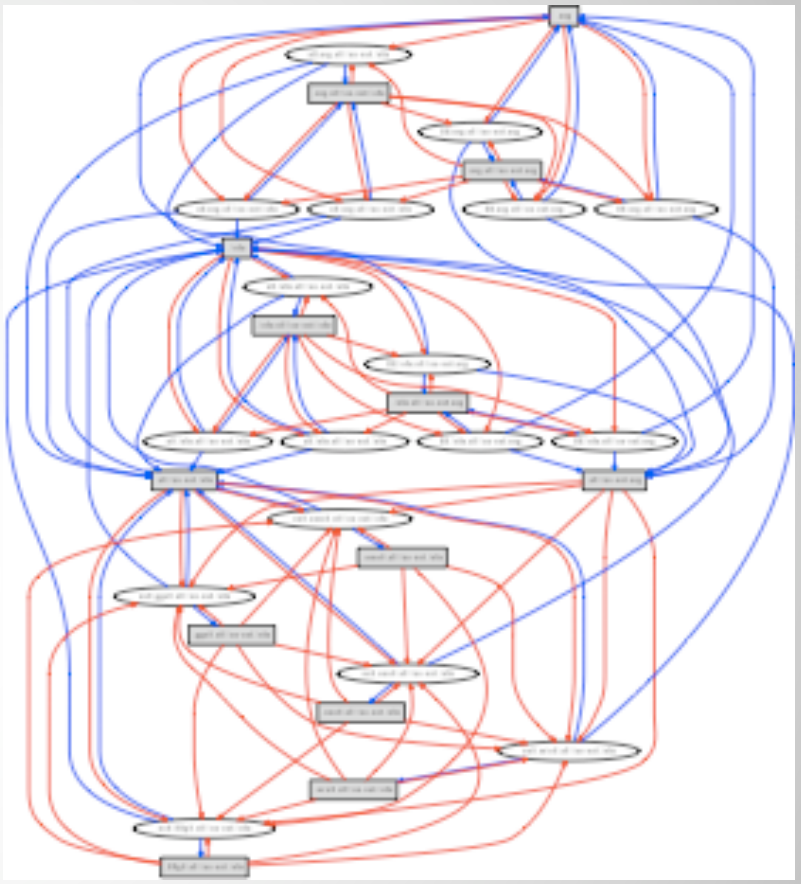
Formal Methods

- Proofs of correctness
- Note: verification only
- Usually done with formal specifications

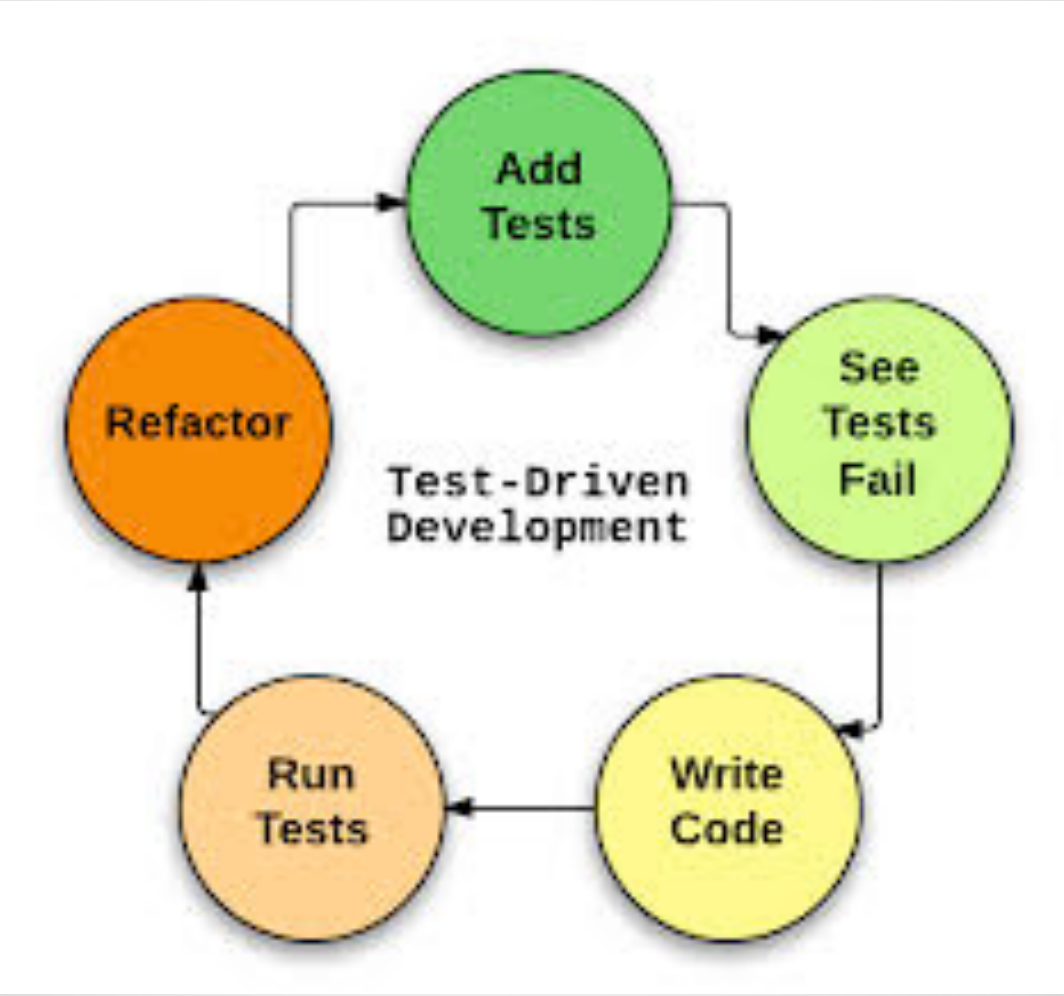
$$
\begin{array}{|l}
project : OZSpec \rightarrow UMLDiagram \\
\hline
\forall (oz, uml) : project \bullet \\
\quad \{c : oz \cap Classdef \bullet c.name\} = \{c : uml.classes \\
\qquad \bullet c.name\} \bullet \forall c_1, c_2 : oz \cap Classdef \bullet \exists_1 c' : \\
\qquad\quad uml.classes \bullet c'.name = c_1.name \\
\qquad\quad c'.attris = \{cls : Classdef \mid cls \in oz \bullet cls.name\} \\
\qquad\qquad \lhd c_1.state.decpart \\
\qquad\quad c'.ops = \{o : Opdef \mid o \in c_1.ops \bullet o.name\} \\
\qquad c_2.name \in \{t : \mathrm{ran}\ c_1.state.decpart \bullet t.name\} \Rightarrow \\
\qquad\quad \exists_1 (c_1', c_2') : uml.agg \bullet c_1'.name = c_1.name \\
\qquad\qquad \wedge\ c_2'.name = c_2.name \\
\qquad c_2.name \in \{inh : \mathrm{dom}\ c_1.inherit \bullet inh.name\} \Rightarrow \\
\qquad\quad \exists_1 (c_1', c_2') : uml.inh \bullet c_1'.name = c_1.name \\
\qquad\qquad \wedge\ c_2'.name = c_2.name
\end{array}
$$

Static Analysis

- A computer program analyzes source code and finds defects (without running the code)
- Results are reviewed by a person because many "errors" are not errors at all



Test-Driven Development (TDD)



Test-Driven Development (TDD)

- Test cases are produced based on the developer's intuitions and experience
- Usually leads to writing more tests and simpler code
- Usually achieves at least node coverage

Testing: A Look Back

- Quality assurance
- Testing
- Black-box (Specification-based) Testing
- White-box (Structural) Testing
- Miscellaneous testing topics
  - Inspections and reviews
  - Formal methods
  - Static analysis
  - Test-driven development

Testing Topics Not Covered

- Combinatorial Testing
- Regression Testing
- Performance Testing
- Load Testing
- Debugging
- Failure injection
  - e.g., Chaos Monkey
- Fuzz Testing
- …

Today's Lecture

- White-box (Structural) Testing
- Miscellaneous testing topics
- Homework 3
- Quiz 5 study guide

Homework 3

- Multi-person
  - What if people "step on each other's toes?"
  - What if people need to access each other's files?
- Multi-version
  - Each version has some shared files, and some files specific to that version
  - Versions may replace each other over time
  - A prior version may need to be examined or modified

Version Control Software

- a.k.a. “Revision control,” “Source control”
- Manages changes over time by multiple people

Homework 3

- Homework 3 will introduce you to the concepts and software behind version control, using the example of Git
- Three parts
  - Part A (due 5/27)
    - Install Git, perform some basic commands
  - Part B (due 5/31)
    - Create a local repository
  - Part C (due 6/2)
    - Download and examine a public repository

Homework 3

- Homework 3 is posted
- If you are new to Git, go to discussion tomorrow for an overview
  - If not, feel free to skip (or just go and get your midterm)
- For technical help/questions with the assignment, post on Piazza (try Google first for general issues)

Today's Lecture

- White-box (Structural) Testing
- Miscellaneous testing topics
- Homework 3
- Quiz 5 study guide

Quiz 5 study guide

- Black-box testing
  - Equivalence class partitioning
  - Boundary value analysis
- White-box testing
  - Node coverage
  - Edge coverage

Next Time

- Software process models
- Discussion tomorrow– Git overview for beginners + midterm return