White-Box Testing Techniques IV

Software Testing and Verification

Lecture 10

Prepared by

Stephen M. Thebaut, Ph.D.

University of Florida
White-Box Testing Topics

• Logic coverage (lecture I)
• Dataflow coverage (lecture II)
• Path conditions and symbolic evaluation (lecture III)
• Other white-box testing strategies (e.g., “fault-based testing”) (lecture IV)
Other white-box testing strategies

• Program instrumentation
• Boundary value analysis (revisited)
• Fault-based testing
  – Mutation analysis
  – Error seeding
Program Instrumentation

- Allows for the measurement of white-box coverage during program execution.
- Code is inserted into a program to record the cumulative execution of statements, branches, du-paths, etc.
- Execution takes longer and program timing may be altered.
Program Instrumentation

- Allows for the **measurement of white-box coverage** during program execution.
- Code is inserted into a program to record the **cumulative** execution of statements, branches, du-paths, etc.
- Execution takes longer and program timing may be altered.
Program Instrumentation

- Allows for the measurement of white-box coverage during program execution.
- Code is inserted into a program to record the cumulative execution of statements, branches, du-paths, etc.
- Execution takes longer and program timing may be altered.
Boundary Value Analysis

(1) if \((X < Y)\) then
   A
(2) else
   B
end if else

- Applies to both program control and data structures.
- Strategies are analogous to black-box boundary value analysis.
Boundary Value Analysis

(1) if \((X < Y)\) then
   \(A\)
(2) else
   \(B\)
end_if_else

- Applies to both program control and data structures.
- Strategies are analogous to black-box boundary value analysis.
Fault-Based Testing

• Suppose a test case set reveals NO program errors – should you celebrate or mourn the event?

• Answer: it depends on whether you’re the developer or the tester... 😊

• Serious answer: it depends on the error-revealing capability of your test set.

• Mutation Analysis attempts to measure test case set sufficiency.
Fault-Based Testing

• Suppose a test case set reveals NO program errors – should you celebrate or mourn the event?

• **Answer:** it depends on whether you’re the developer or the tester... 😊

• Serious answer: it depends on the error-revealing capability of your test set.

• Mutation Analysis attempts to measure test case set sufficiency.
**Fault-Based Testing**

- Suppose a test case set reveals **NO program errors** – should you celebrate or mourn the event?

- **Answer:** it depends on whether you’re the developer or the tester... 😊

- **Serious answer:** it depends on the error-revealing capability of your test set.

- Mutation Analysis attempts to measure test case set sufficiency.
Fault-Based Testing

- Suppose a test case set reveals **NO program errors** – should you celebrate or mourn the event?

- **Answer:** it depends on whether you’re the developer or the tester… 😊

- **Serious answer:** it depends on the error-revealing capability of your test set.

- **Mutation Analysis** attempts to measure test case set sufficiency.
Mutation Analysis Procedure

1. Generate a large number of “mutant” programs by replicating the original program except for one small change (e.g., change the “+” in line 17 to a “-”; change the “<“ in line 132 to a “<=“; etc.).

2. Compile and run each mutant program against the test set.

(cont’d)
Mutation Analysis Procedure

1. Generate a large number of “mutant” programs by replicating the original program except for one small change (e.g., change the “+” in line 17 to a “-”; change the “<” in line 132 to a “<=”; etc.).

2. Compile and run each mutant program against the test set.

(cont’d)
Mutation Analysis Procedure (cont’d)

3. Compare the ratio of mutants “killed” (i.e., revealed) by the test set to the number of “survivors.”

---

• The higher the “kill ratio” the better the test set.

• What are some of the potential drawbacks of this approach?
3. Compare the ratio of mutants “killed” (i.e., revealed) by the test set to the number of “survivors.”

---

• The higher the “kill ratio” the better the test set.

• What are some of the potential drawbacks of this approach?
Mutation Analysis Procedure (cont’d)

3. Compare the ratio of mutants “killed” (i.e., revealed) by the test set to the number of “survivors.”

---

- The higher the “kill ratio” the better the test set.
- What are some of the potential drawbacks of this approach?
Error Seeding

• A similar approach, Error Seeding, has been used to estimate the “number of errors” remaining in a program.

• But such metrics are inherently problematic. For example, how many “errors” are in the following Quick Sort program?
A similar approach, Error Seeding, has been used to estimate the “number of errors” remaining in a program.

But such metrics are inherently problematic. For example, how many “errors” are in the following Quick Sort program?
Error Seeding

- A similar approach, *Error Seeding*, has been used to estimate the “number of errors” remaining in a program.

- But such metrics are inherently problematic. For example, how many “errors” are in the following Quick Sort program?

```plaintext
QSORT(X,N)  
    Return(X)  
END
```
Error Seeding Procedure

1. Before testing, “seed” the program with a number of “typical errors,” keeping careful track of the changes made.

2. After a period of testing, compare the number of seeded and non-seeded errors detected.

(cont’d)
Error Seeding Procedure

1. Before testing, “seed” the program with a number of “typical errors,” keeping careful track of the changes made.

2. After a period of testing, compare the number of seeded and non-seeded errors detected.

(cont’d)
3. If $N$ is the total number of errors seeded, $n$ is the number of seeded errors detected, and $x$ is the number of non-seeded errors detected, the number of remaining (non-seeded) errors in the program is about

$$x\left(\frac{N}{n} - 1\right)$$

• What assumptions underlie this formula?
• Consider its derivation...
Error Seeding Procedure (cont’d)

3. If $N$ is the total number of errors seeded, $n$ is the number of seeded errors detected, and $x$ is the number of non-seeded errors detected, the number of remaining (non-seeded) errors in the program is about

$$x\left(\frac{N}{n} - 1\right)$$

- What assumptions underlie this formula?
- Consider its derivation...
3. If $N$ is the total number of errors seeded, $n$ is the number of seeded errors detected, and $x$ is the number of non-seeded errors detected, the number of remaining (non-seeded) errors in the program is about

$$x(N/n - 1)$$

- What assumptions underlie this formula?
- Consider its derivation...
Derivation of Error Seeding Formula

Let $X$ be the total number of NON-SEEDED errors in the program.

Assuming seeded and non-seeded errors are equally easy/hard to detect, after some period of testing, $x:n \approx X:N$.

So,

$$X \approx \frac{xN}{n}$$

$$X - x \approx \frac{xN}{n} - x$$

$$\approx x\left(\frac{N}{n} - 1\right) \text{ as claimed.}$$
White-Box Testing Techniques IV

Software Testing and Verification

Lecture 10

Prepared by
Stephen M. Thebaut, Ph.D.

University of Florida