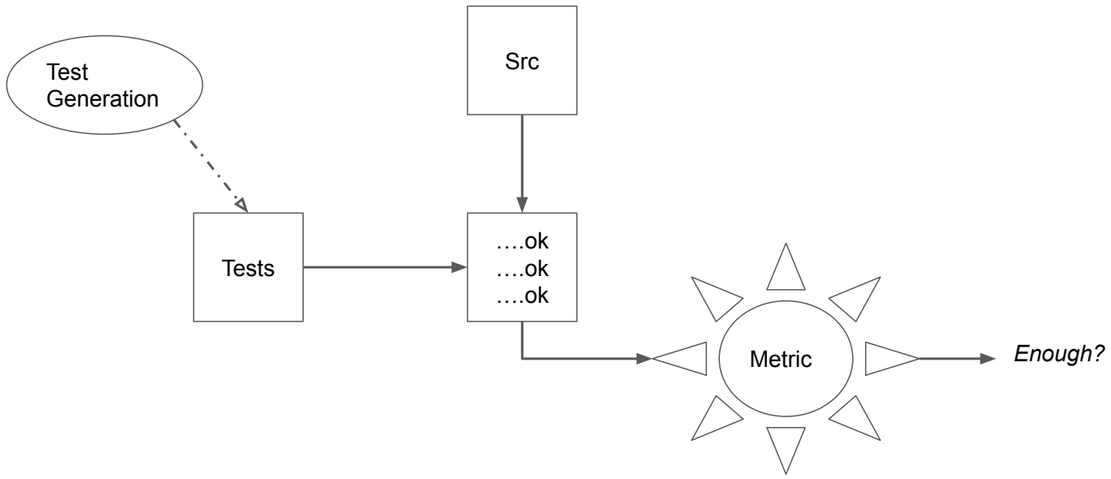
COM4506/6506: Testing and Verification in Safety Critical Systems

Dr Ramsay Taylor

Contents

- Apart from Code coverage, how can we test our tests?
- Mutation Testing
- Model Inference

Test Adequacy



Things other than Code Coverage

```
public class FtoC {

    public static double ftoc(double tempf) {
        double x, y, result;

        x = tempf-32;
        y = 5.0 / 9.0;
        result = x * y;

        return(result);
    }

    public static double ctof(double tempc) {
        double x, y, result;

        y = 9.0 / 5.0;
        x = tempc * y;
        result = x + 32;

        return(result);
    }

}
```

Covering the code, even with MC/DC styles of coverage, still doesn't prove that the code **works correctly**.

```
@Test
public void TestMinus40() {
    assert(FtoC.ftoc( tempf: -40) == FtoC.ctof( tempc: -40));
}
```

Things other than Code Coverage

```java
public class FtoCStupid {
    public static double ftoc(double tempf) {

        return(0.0);
    }

    public static double ctof(double tempc) {

        return(0.0);
    }

}
```

A broken implementation has the same property, and the test achieves 100% coverage!

```java
@Test
public void TestMinus40() {
    assert(FtoC.ftoc( tempf: -40) == FtoC.ctof( tempc: -40));
}
```

Things other than Code Coverage

```java
public class FtoCStupid {
    public static double ftoc(double tempf) {

        return(0.0);
    }

    public static double ctof(double tempc) {

        return(0.0);
    }

}
```

Code coverage assures that you have tested **all of the code that you wrote**, it doesn't tell you anything about **the code you *should have written***!

Good tests should test the **required behaviour** of the system.

Testing the actual implementation is good too, though!

Mutation Testing

"Testing shows the presence, not the absence of bugs."

Edsger Dijkstra, 1969

…so, let's give it something to find!

Mutation Testing

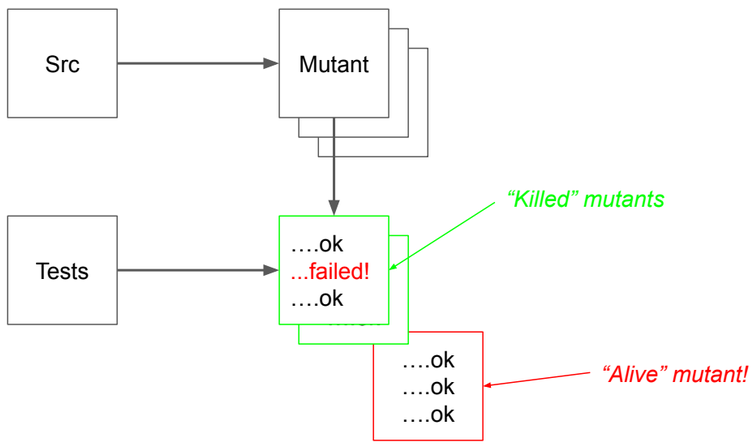
```java
public class FtoC {

    public static double ftoc(double tempf) {
        double x, y, result;

        x = tempf-32;
        y = 5.0 / 9.0;
        result = x * y;

        return(result);
    }

    public static double ctof(double tempc) {
        double x, y, result;

        y = 9.0 / 5.0;
        x = tempc * y;
        result = x - 32;

        return(result);
    }

}
```

If we create **deliberate faults**, will the tests identify them?

```java
@Test
public void TestMinus40() {
    assert(FtoC.ftoc( tempf: -40) == FtoC.ctof( tempc: -40));
}
```

```
java.lang.AssertionError Create breakpoint
    at FtoCTest.TestMinus40(FtoCTest.java:9) <25 internal calls>
```

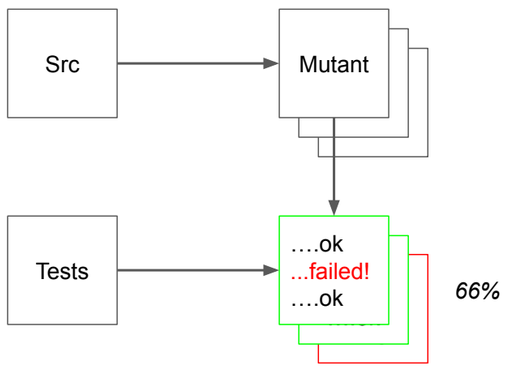
Mutation testing



Mutation Operators

- Mutation operators might simulate *common mistakes*
  - Wrong variable names
  - Wrong operators
  - Reversed parameters
- Or they might be designed to produce common faults
  - Unexpected exceptions
  - Wrong types/operations
  - Commented out functional statements

Mutation Testing Metric



66%

Percentage of “killed” mutants is the numeric metric

Semantic Equivalence

```
public static double ftoc(double tempf) {
    double x, y, result;

    x = tempf-32;
    y = 5.0 / 9.0;
    result = x * y;

    return(result);
}
```

```
public static double ftoc(double tempf) {
    double x, y, result;

    y = 5.0 / 9.0;
    x = tempf-32;
    result = x * y;

    return(result);
}
```

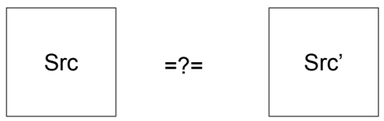
No test set can kill this mutant!

… because there is no difference in the function of the code.

Semantic Equivalence

We have to exclude semantically equivalent mutants when we calculate the success rate of the test set.

Unfortunately, determining whether mutants are semantically equivalent is usually not easy automatically.



Mutation Testing vs Code Coverage

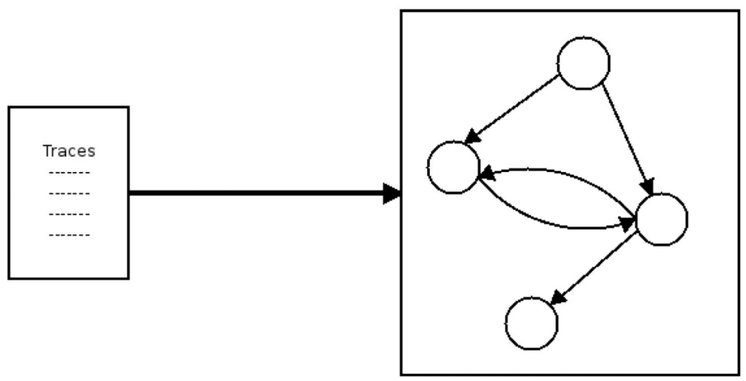
Pros:

- Identifies whether the test set is testing the *function* of the code
- Can give a more general sense of the quality of the test
  - would it find this sort of thing if it was there?

Cons:

- Requires you to compile and test many copies of the code
- Semantic equivalence might invalidate the value to some extent

Alternative Test Adequacy Metrics



Summary

- Code coverage evaluates whether the test set is testing *this* code.
- Mutation testing shows what the test set will do to *other* code.
- ...both have value!
- There are some other approaches to Test Adequacy.
- ...no, I don't want to talk about Test Adequacy Adequacy!