

Testing - II

Outline

Review

Example

More Testing Tactics

Doing the Testing

Testing Testing

Process Issues

Another Example

(time permitting)

Tactical Issues for Testing

Driven by:

- Inputs
- Output
- Internals

Test Set vs. Test Execution

Black Box

- Ignores internals
- Driven by specifications
- Suitable for separate V&V team
- Unit testing up to entire system

Glass Box Testing

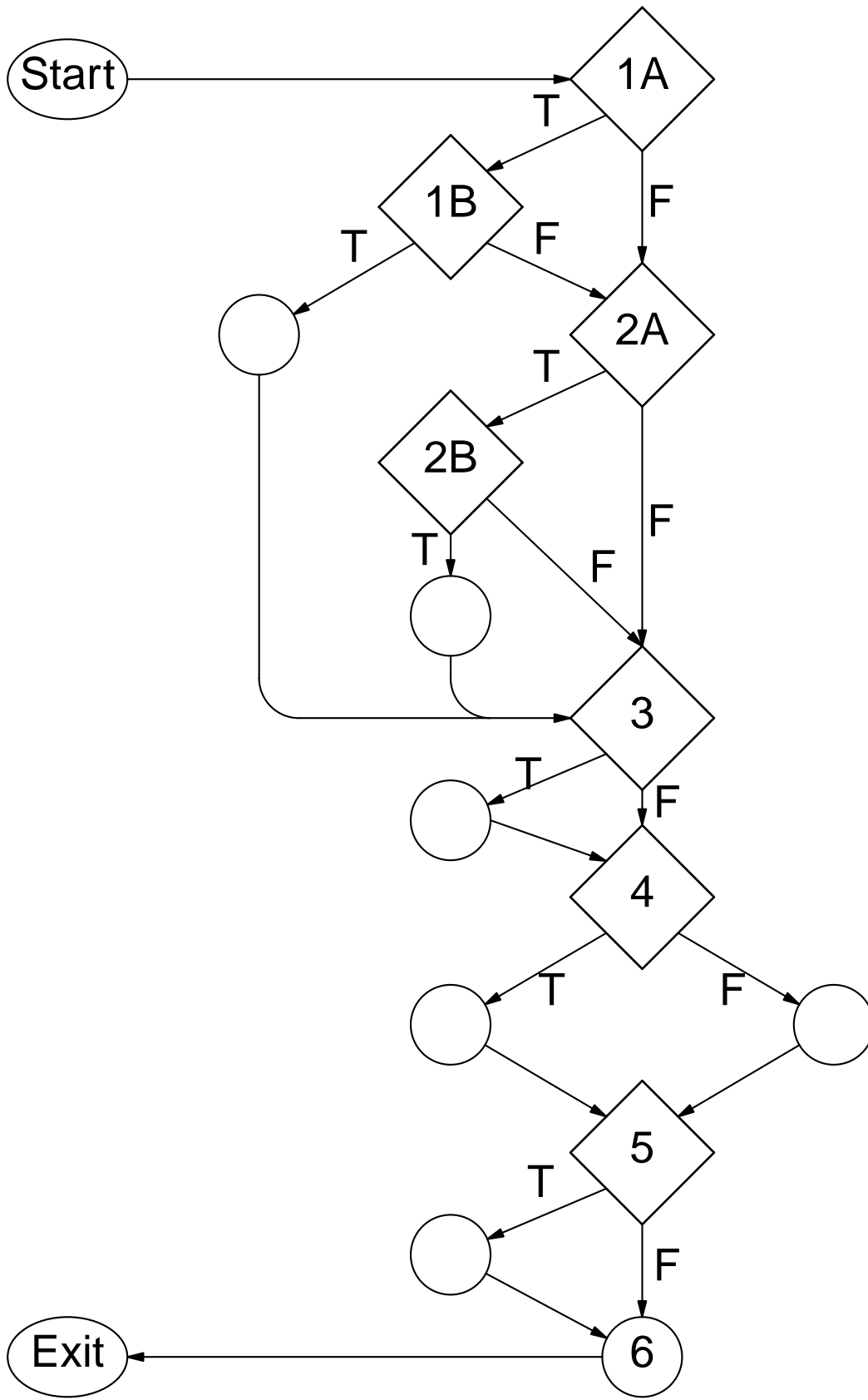
- Recognizes internals
(i.e. uses source code)
- Driven by design and implementation
- Unit testing
- Can apply tools
- Suitable for test evaluation

Glass Box Example

```
function ratecalc(# freight rate calc
  int disccat,    # discount category
  int contract,  # minimum discount
  int dist,      # distance shipped
  int wght )     # shipment weight
  returns int ;  # rate in cents

int discount ;   # as percent
int baserate ;  # in cents

if (disccat==1 & dist>=200) then      #1
  discount := 10
elsif (disccat==2 & dist>=300) then #2
  discount := 5
endif;
if ( contract>discount ) then          #3
  discount := contract
endif;
if ( weight <= 100 ) then              #4
  baserate := 1500
else
  baserate := 10*weight + 500
endif;
if ( dist > 50 ) then                   #5
  baserate := baserate*(0.9+dist/50.0)
endif;
return baserate*(100-discount)/100;    #6
end ratecalc;
```



Path Coverage for Example

#	nodes					remarks
	1	2	3	4	5	

#	disccat	contract	dist	wght	output	✓	remarks

Glass Box Testing Techniques

Heuristics for objects:

- start with graph of object interactions
- try to cause flow on every edge with every “message”
- for each object, one at a time
 - ◇ try object in all internal states
 - ◇ try object through all state transitions
 - ◇ try all methods of object, with parameter variations as above
- ✧ formal specifications (as with ADT’s)
immensely helpful
- ✧ “design for testability”
 - ⇒ provide test mode for objects

Testing with Persistent Data

Develop test cases based on ER model
e.g.: cardinality constraints

Compare 'before' and 'after' snapshots

Must try both internal and environment events

Stress testing

- absolute data volume
- system data flow

Timing tests

- critical for large systems
- very difficult

Testing MS Access

How would you implement an Access form in imperative code?

What does that imply about test designs?

Test Infrastructure

Build testing mechanisms
along with the rest of the project

- scaffolding
- stubs

Test Input

- data sets
- test scripts

Test points

Special testing environments

Test Scaffolds

Also called:

- drivers
- harnesses

“Simulate” operational environment

Often implemented just to feed test data sets

May require extension of user interface
for on-line testing

Look for help from toolkit/ADE

e.g. JUnit, especially TestCase and TestSuite

JUnit only helps, you must build scaffold from
components supplied by JUnit

Stubs

Stubs are dummy procedures that simulate real ones to be implemented later

Stubs required because:

- procedure too complex
- procedure indeterminate
- environment different

Implemented according to data flow

- in:* log value
- out:* return valid value
- both:* look-up value in table
- neither:* log trace

Limitations:

Stubs don't work well for *structural* operations
(e.g. invert a list)

Test Environment for Example

How would we test `ratecalc`?

Testing Testing

Goal: Answer the question

“Do the proposed tests adequately evaluate the program?”

Coverage tests -

Path/condition/dataflow tests are defined in terms of an “adequacy” condition

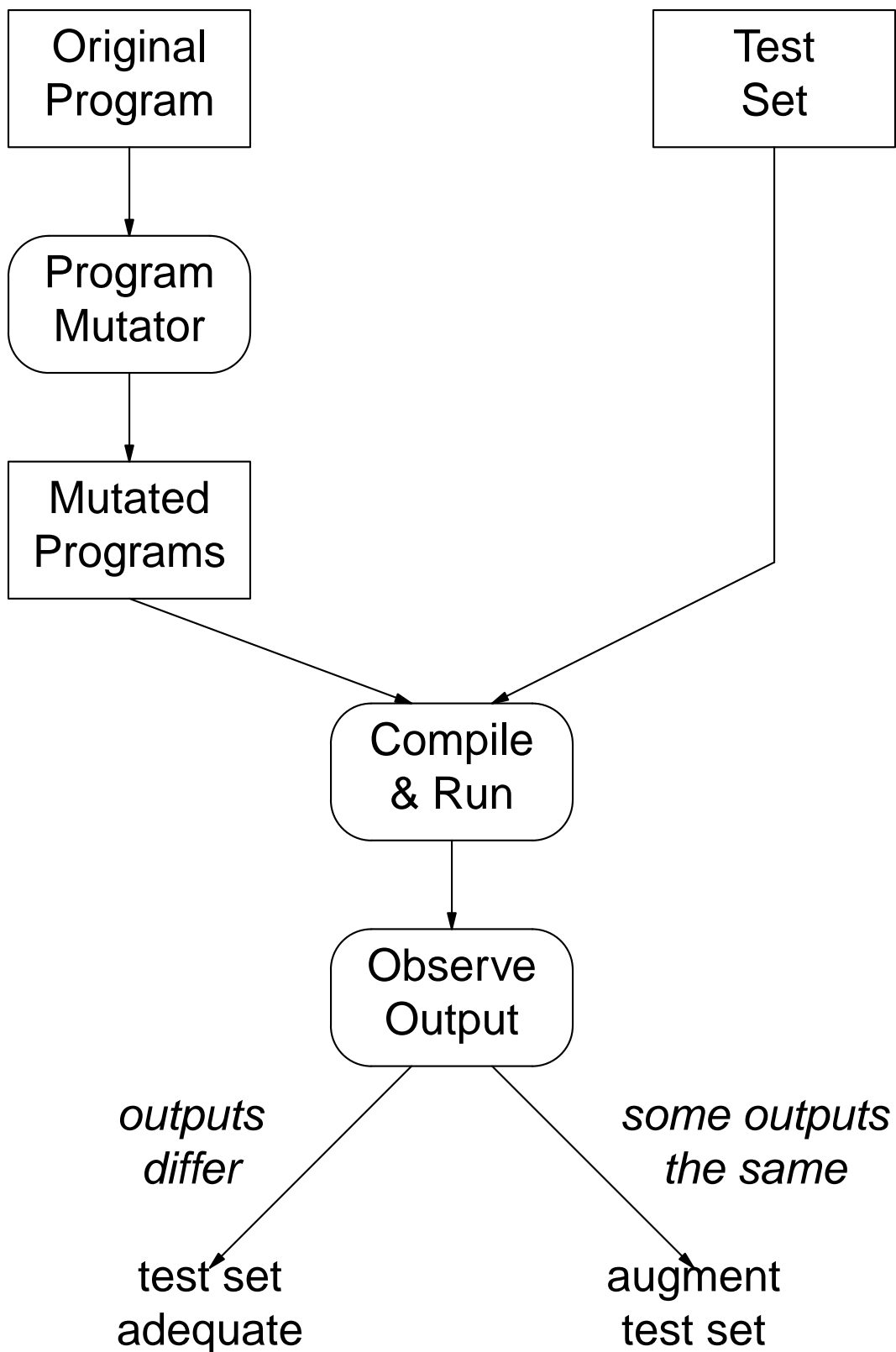
Can be applied after test design as an adequacy condition only

Mutation testing -

Interesting illustration of test testing

Brings probabilistic analysis to program testing

Mutation Testing



Testing Activities

1. **Develop test *plan***
what is to be tested, testing techniques used,
how tests are evaluated,
testing task assignment
 2. **Design tests**
how will tests make desired evaluations
 3. **Develop tests**
 - test data sets
 - test scripts
 - test program suites
 4. **Develop test machinery**
 - stubs
 - scaffolds
 - drivers
 - diagnostic & evaluation tools
 - specialized hardware
 5. **Evaluate tests**
 6. **Run tests**
 7. **Evaluate test results**
 - compare nominal and actual
 - postprocess output
- * **Monitor and evaluate test process**

Testing Tools

- static analyzers
- code auditors
- assertion processors
- test file generators
- test data generators
- test scaffolding (harnesses)
- output comparators
- symbolic execution systems
- environment simulators
- path/condition/dataflow analyzers
- mutation test tools
- ¬ debuggers

Testing & Organizational Behavior

Recognition of importance of testing is growing,
but not yet strong enough

Influence of perspective:

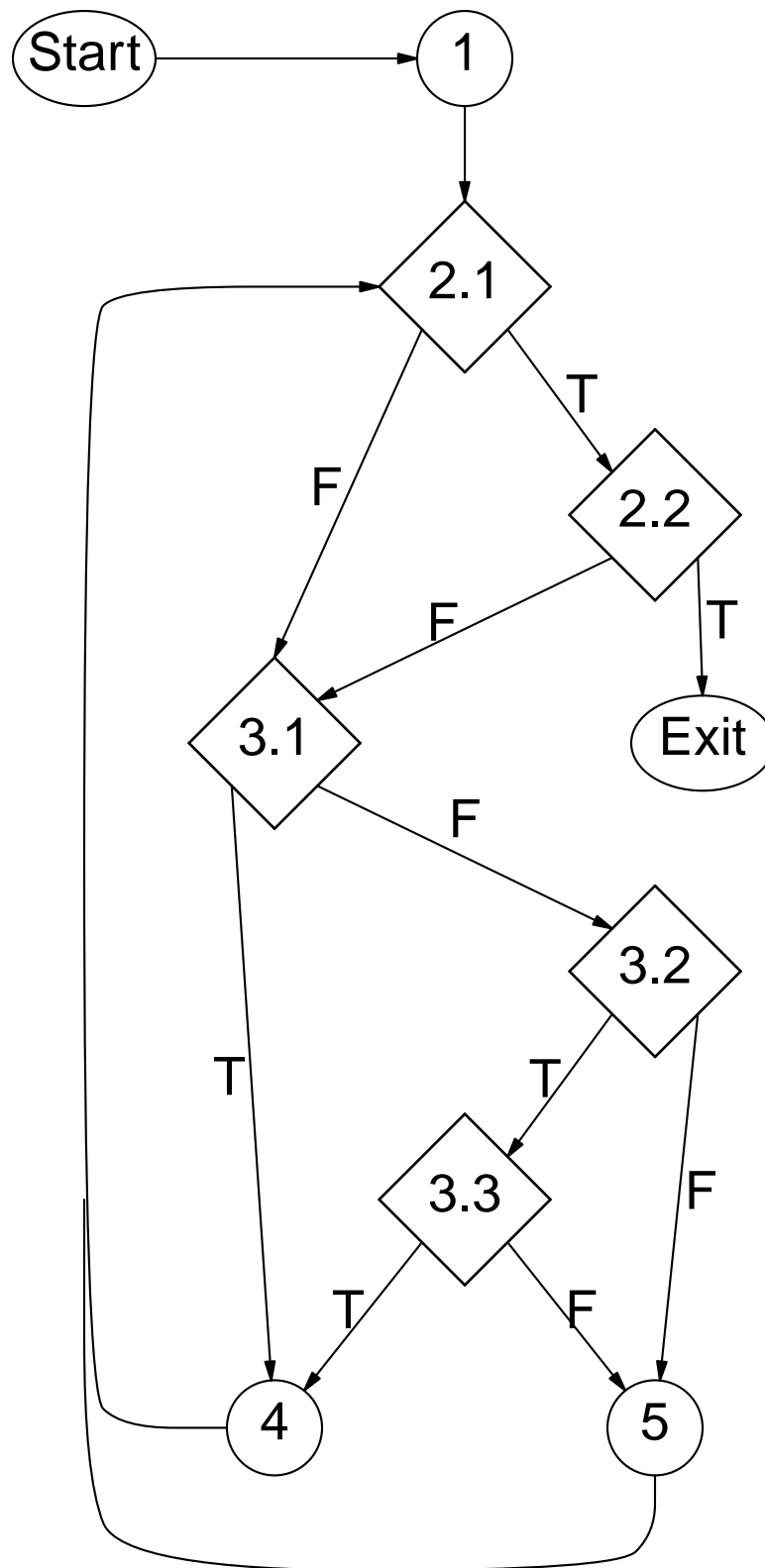
- A *good* test • • • is when
 - the program runs with no exceptions
 - programmer
 - defects are exposed
 - test engineer

Example Test Instance

```
#####
# merge subroutine -- Perl
##### input:
# A, B - files of integers, ascending
##### output:
# C      - A and B merged, ascending
##### pseudocode:
# WHILE input remaining
#   IF   (B empty) OR
#       (A not empty and A[1] < B[1])
#       output A[1]; read A
#   ELSE
#       output B[1]; read B
sub merge {
  my $A1, $B1, ( $A, $B, $C ) = @_ ;
  $A1 = int( <$A> );
  $B1 = int( <$B> ); #1
  while (not eof($A) && eof($B)) { #2
    if ( eof($B) or
          (!eof($A) && $A1 <= $B1) ) #3
      {printf $C "%d\n", $A1;
        $A1 = int(<$A> )} #4
    else
      {printf $C "%d\n", $B1;
        $B1 = int(<$B> )}; #5
  }; };

```

Control Flow Graph for Example



Test Covering for Example

1. 1, 2.1, 2.2, Exit
both files A and B are empty (finds bug)
2. 1, 2.1, 2.2, 3.1, 4, ...
impossible! (2.2 and 3.1 test the same condition)
3. 1, 2.1, 2.2, 3.1, 3.2, 3.3, ...
A is empty and B is not

et cetera

Test Scaffolding for Example

```
#!/bin/perl
```

```
sub merge;
```

```
# dumb minimal test scaffolding
```

```
open(A, "/tmp/edrbtsn/a");
```

```
open(B, "/tmp/edrbtsn/b");
```

```
open(C, ">/tmp/edrbtsn/c");
```

```
merge(A,B,C);
```

Test Scaffolding for Example

```
#!/bin/perl

sub merge;

# test scaffolding

$testdir = "/u/edrbtsn/ExampleProject/Tests";
$testcount = 1;

while ( -e "$testdir/mergeA$testcount" ) {
    open(A, "$testdir/mergeA$testcount");
    open(B, "$testdir/mergeB$testcount");
    open(C, ">/tmp/testresult");

    merge(A,B,C);

    close A; close B; close C;

    if ( system "diff /tmp/testresult
        $testdir/mergeR$testcount >
        /tmp/testCK$testcount" == 0 ) {
        printf "test case %d successful\n",
            $testcase }
    else { printf "test case %d unsuccessful\n",
        $testcase };
}; # end while
```