CS1632, Lecture 4: Test Plans and Breaking Software

BILL LABOON

You've got requirements.
You're looking for defects.

How?

Develop a test plan!

Formality

- This could be as formal or informal as necessary.
- Think about what you are testing – what level of responsibility / tracking is necessary?

- Throw-away script?
- Development tool?
- Internal website?
- Enterprise software?
- Commercial software?
- Operating system?
- Avionics software?

Testing is context-dependent

- How you test
- How much you test
- What tools you use
- What documentation you provide

- ...All vary based on software context.

Formal Test Plans

- A test plan is a sequence of test cases.
- A test case is the fundamental “unit” of a test plan.

A test case mainly consists of…

- Preconditions
- Execution Steps
- Postconditions

See IEEE 829, "Standard for Software Test Documentation", for more details

Example

Assuming an empty shopping cart, when I click "Buy Widget", the number of widgets in the shopping cart becomes one.

Preconditions: User is on main page of site, with an empty shopping cart
Execution Steps: Click "Buy Widget“
Postconditions: Shopping cart displays one widget

Example

Assuming that the SORT_ASCENDING flag is set, calling the sort method with [9,3,4,2] will return a new array with the original data sorted from low to high, i.e., [2,3,4,9].

Precondition: SORT_ASCENDING flag is set
Execution steps: Call .sort method with argument [9,3,4,2]
Postconditions: [2,3,4,9] is returned

We also want to add:

- Identifier: A way to identify the test case
  - Could be a number
  - Often a label, e.g. INVALID-PASSWORD-THREE-TIMES-TEST
- Description: A description of the test case, describing what it is supposed to test.

EXAMPLE

TEST CASE IDENTIFIER: FUN-IGNORE-CALL-UI
DESCRIPTION: This test verifies that the phone software properly ignores a call when the user presses the "Ignore call" button.
PRECONDITIONS: A phone call is being made to the phone. The screen should be displaying two buttons, "Answer Call" and "Ignore Call", and the ringtone should be being played.
EXECUTION STEPS: Press "Ignore Call"
POSTCONDITIONS: The ringtone has stopped. The phone's display has returned to the main screen.

Test Plan

- These do not always test an entire system
- They may test a subsystem or related piece of functionality
  - Examples:
  - Database Connectivity Test Plan
  - Pop-up Warning Test Plan
  - Pressure Safety Lock Test Plan
  - Regression Test Plan

Pressure Safety Lock Test Plan

LOW-PRESSURE-TEST
HIGH-PRESSURE-TEST
SAFETY-LIGHT-TEST
SAFETY-LIGHT-OFF-TEST
RESET-SWITCH-TEST
RESET-SWITCH2-TEST
FAST-MOVEMENT-TEST
RAPID-CHANGE-TEST
GRADUAL-CHANGE-TEST
MEDIAN-PRESSURE-TEST
LIGHT-FAILURE-TEST
SENSOR-FAILURE-TEST
SENSOR-INVALID-TEST

A group of test plans make up a test suite…

- Regression Test Suite
  - Pressure Safety Regression Test Plan
  - Power Regulation Regression Test Plan
  - Water Flow Regression Test Plan
  - Control Flow Test Plan
  - Security Regression Test Plan
  - Secondary Safety Process Test Plan

Test Run – An actual run-through of a test plan or test suite.

- Analogy time: class vs object, test plan vs test run
  - The test plan is the structure, but you need to actually execute it to find out anything
- During the test run, the tester manually executes each test case and sets the status

Possible Statuses

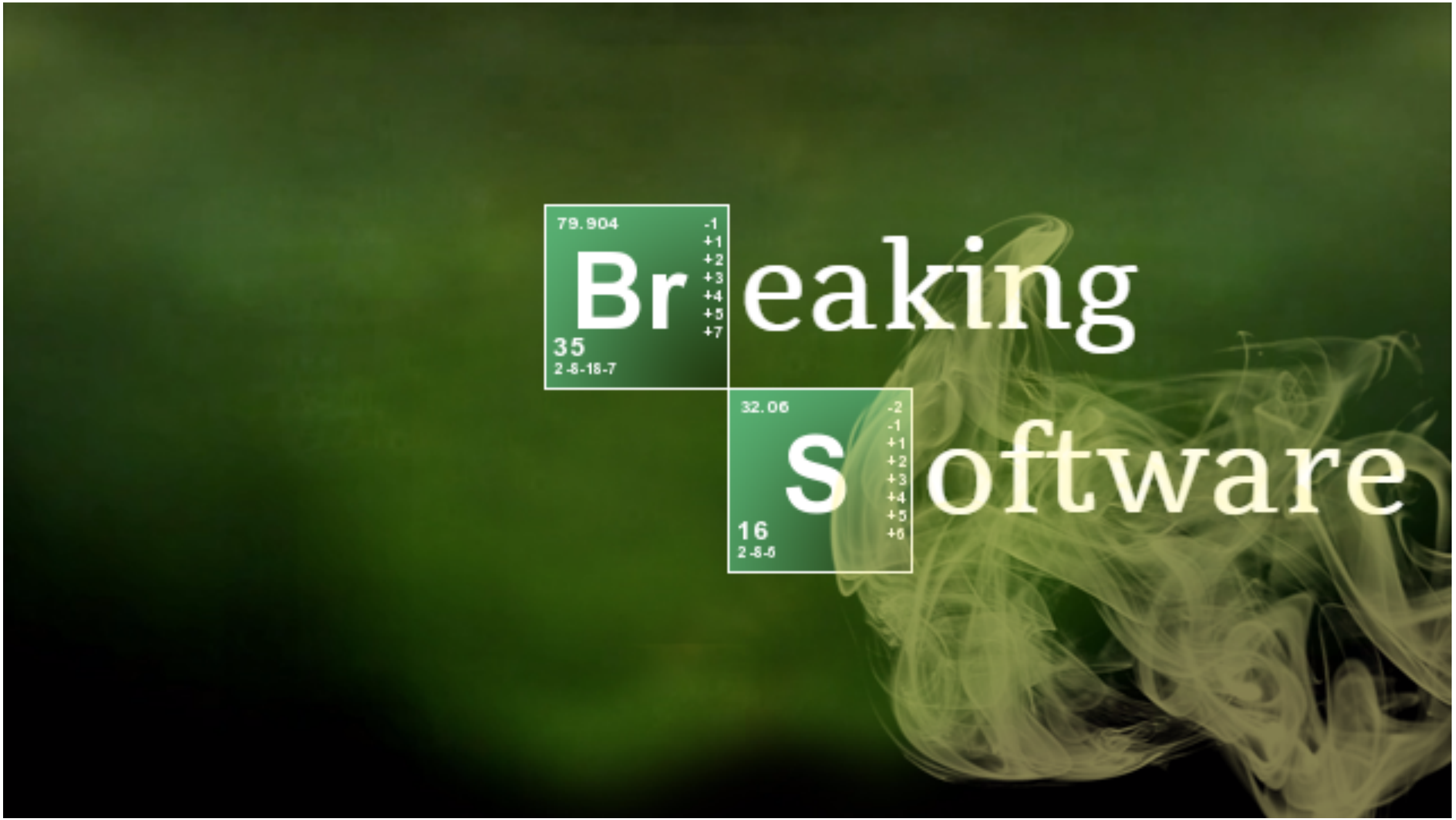
- PASSED
- FAILED
- PAUSED
- RUNNING
- BLOCKED
- ERROR

Defects

- If the test case fails, a defect should be filed
  - Unless the test case has already failed, of course.
  - You don't need to re-file a duplicate of the defect!
- Note the level of formality involved will vary based on the domain

Creating a test plan…

- Start top-down: what is a good way to subdivide the system into features (test plans)?
- For a given feature (test plan), what aspects do I want to test?
- For each aspect, what test cases do I want that will hit different equivalence classes / success or failure cases / edge or corner cases / etc.?
- How deep should I go down?
- Try to have test cases be independent of each other, and reproducible!


79.904
-1
+1
+2
+3
+4
+5
+7
Br
35
2-8-18-7
eaking
32.06
-2
-1
+1
+2
+3
+4
+5
+6
S
16
2-8-6
oftware

Software tends not to break much on the happy path.

- It breaks when there's…
  - unexpected input.
  - malicious users.
  - systems going down.
  - when you're off in the wilderness.

Logic Errors:
The logic of the program is incorrect

```
if (student.isTaking(cs1632)) {
    student.setHappy(false);
} else {
    student.setHappy(true);
}
```

Off-by-one error:
a subset of logic errors where values are specified incorrectly by one unit

```
if (student.getNumCredits() > 120) {
    student.setCanGraduate(true);
} else {
    student.setCanGraduate(false);
}
```

Floating point errors

```
double oneVal = 1.0 / 857.0;
double total = oneVal * 857.0;

System.out.println("Should be 1.0, actually = "
+ total);

boolean areEqual = (total == 1.0);
System.out.println("Are equal? " + areEqual);
```

Integration errors:
Errors at boundaries between systems/subsystems.

```
int startDistanceInKilometers = 14;
spacecraft.setDistance(startDistanceInKilometers);

...

public class Spacecraft
    public void setDistance(int distanceInMiles) {
        ...
    }
}
```

Errors of assumption:
The developer or system makes an assumption which turns out to be incorrect, or at odds with other assumptions.

```
OutputFile.write(TAB_DELIMITED);

...

InputFile.read(COMMA_DELIMITED);
```

MISSING DATA ERRORS:
An error occurs because needed data is missing and the system cannot operate properly without it.

```
public static void main(String[] args) {
    System.out.println(args[3]);
}
```

BAD DATA ERRORS:
System cannot handle improperly formatted or invalid data.

```
Enter two numbers to divide: 7 0
Exception in thread "main“
    java.lang.ArithmeticException: / by zero
```

DISPLAY ERRORS:
The data is correct but not displayed properly.

```
double pi = Math.PI;
System.out.printf("Pi is equal to...%.1f!",
  pi);
```

Null pointer error:
The program dereferences a null pointer.

```
String oneILove = null;
oneILove = oneILove.toUpperCase();

System.out.printf("This one goes out
to the one I love," + oneILove);
```

I/O Errors:
The system encounters an unexpected state of disk, network, or other I/O and cannot handle it.

```
try {
   // read in file
} catch (FileNotFoundException e) {
   // AAAGH WHAT DO I DO
   System.exit(1);
}
```

Configuration error:
The system could work correctly, but it was not configured to work correctly.

```
'javac' is not recognized as an
internal or external command,
operable program or batch file.
```

The list goes on…

- Accessibility errors
- Domain-specific errors
- Version mismatch errors
- Distributed system errors
- Logging Errors
- Interface Errors
- Logging Errors
- Regulatory or Legal Errors