Lecture 2 - Testing Basics

Lecture 2 - Testing Basics

TESTING BASICS

Expected Behavior
vs
Observed Behavior

Expected Behavior
vs
Observed Behavior

EXPECTED BEHAVIOR VS OBSERVED BEHAVIOR

This is the key concept to this entire class!

You need to know what you are testing for before you can test for it!*

Under circumstances X...
Y is expected to happen.

When I type 2 + 2 on my calculator, I expect to get to 4. However, I see 3.

Circumstances: Type 2 + 2 =
Expected Behavior: See 4
Observed Behavior: See 3

Keep this in mind as we go through these slides...

.... and the rest of the semester!

EXPECTED BEHAVIOR VS OBSERVED BEHAVIOR

This is the key concept to this entire class!

You need to know what you are testing for before you can test for it!*

Under circumstances X...
Y is expected to happen.

When I type 2 + 2 on my calculator, I expect to get to 4.
However, I see 3.

```
Circumstances: Type 2 + 2 =
Expected Behavior: See 4
Observed Behavior: See 3
```

Prezi

Keep this in mind as we go through these slides...

.... and the rest of the semester!

Why Test Software?

Put Yourself in the Role of CEO...

Idea
Marketing
Development
Testing
Sales

Why Waste Time on Testing? Our Developers are Good, Right?

"Software bugs, or errors, are so prevalent and so detrimental that they cost the U.S. economy an estimated $59.5 billion annually, or about 0.6 percent of the gross domestic product, according to a newly released study commissioned by the Department of Commerce's National Institute of Standards and Technology (NIST)." -NIST Report, 2002

Relative Cost of Fixing Defects

Requirements Analysis:	1x
Software Design:	~2x
Software Development:	6.5x
Testing:	15x
Deployment:	100x

Golden Rule of Testing

Find defects EARLIER rather than later!

In order to do so, we need:
1. A process
2. Standard Terminology
3. Agreed-upon Theory

Ad hoc is not good enough!

EXAMPLE

Military Command & Control System
Functional Test Lead
Project of > 2.5 megaSLOC
> 10 Years in Development
> 80 Developers
> 100 pages of requirements

< 15 TESTERS!

Remember Last Lecture?

One simple function... return a lower-case version of String

= more than 10 different cases!

Let's say 1,000 functions...
each with 10 cases.

So, 10,000 individual cases.

However, you have to deal with inter-relations. This means permutations.

Remember your discrete math..

10,000! (factorial)

2.8 * 10 ^ 35,659
tests necessary!

(that's a lot)

(~ 2 ^ 80 atoms in observable Universe)

This is the art and science of testing... knowing what to test and what not to test.

He who knows when to fight, and when not to fight, will be victorious.
-Sun-Tzu, "The Art of War"

Put Yourself in the Role of CEO...

Idea

Marketing

Development

Testing

Sales

Why Waste Time on Testing?
Our Developers are Good, Right?

Prezi

"Software bugs, or errors, are so prevalent and so detrimental that they cost the U.S. economy an estimated $59.5 billion annually, or about 0.6 percent of the gross domestic product, according to a newly released study commissioned by the Department of Commerce's National Institute of Standards and Technology (NIST)." -NIST Report, 2002

Relative Cost of Fixing Defects

Requirements Analysis: 1x
Software Design: ~2x
Software Development: 6.5x
Testing: 15x
Deployment: 100x

Golden Rule of Testing

Find defects EARLIER rather than later!

In order to do so, we need:

1. A process
2. Standard Terminology
3. Agreed-upon Theory

Ad hoc is not good enough!

EXAMPLE

Military Command & Control System
Functional Test Lead
Project of > 2.5 megaSLOC
> 10 Years in Development
> 80 Developers
> 100 pages of requirements

< 15 TESTERS!

Remember Last Lecture?

One simple function... return a lower-case version of String

= more than 10 different cases!

Let's say 1,000
functions...
each with 10 cases.

So, 10,000 individual cases.

However, you have to deal with inter-relations. This means permutations.

Remember your discrete math..

10,000! (factorial)

2.8 * 10 ^ 35,659

tests necessary!

(that's a lot)

(~ 2 ^ 80 atoms in observable Universe)

This is the art and science of testing...
knowing what to test and what not to test.

He who knows when to fight, and when not to fight, will be victorious.

-Sun-Tzu, "The Art of War"

Prezi

Equivalence Classes

Equivalence Class Partitioning

Partition Testing Parameters by Expected Result

Example: Bus rides are...
... free for children under 2 years old.
... $1.00 for children under 18, but older than 2.
... $1.00 for senior citizens, 65 or older.
... $2.00 for everybody else.

Equivalence Classes

Babies under 2 -> 0
Children > 2 && < 18 -> 1
Adults > 18 && < 65 -> 2
Senior Citizens > 65 -> 1

Note that babies and seniors are NOT the same equivalence class!

Another Example...

Undergrad students get 20% off pizza
Grad students get 20% off pizza
TAs get 10% off pizza

Offers can be combined.
TAs can be undergrad, grad, or neither.
Students must be EITHER grad xor undergrad - can't be both.
Final discount is % addition

Equivalence Classes

Undergrad only -> 20%
Grad only -> 20%
TA only -> 10%
Undergrad + TA -> 30%
Grad + TA -> 30%

Equivalence classes must be PARTITIONED

NO

They need to have a STRICT PARTITIONING

Another example...

Imagine an online store that sells one item (a "quux"). Users can add or remove this item from their shopping cart by clicking + or - buttons. Users can buy 1 or more quuxes. Users can remove quuxes from their shopping cart. Cart displays EMPTY when no quuxes in it.

1. User adds quux to empty cart (+ : 0 -> 1)
2. User adds quux to non-empty cart (+ : (n>0 -> n+1))
3. User removes quux, making cart empty (- : 1 -> 0)
4. User removes quux, cart is not empty (- : n>0 -> n-1)
5. User attempts to remove quux from empty cart (- : 0)

Note how we reduced a potentially limitless testing set ((+ : 5 -> 6, (+ : 6 -> 7), etc.) to five test cases.

Equivalence Class Partitioning

Partition Testing Parameters by Expected Result

Example: Bus rides are...

... free for children under 2 years old.

... $1.00 for children under 18, but older than 2.

... $1.00 for senior citizens, 65 or older.

... $2.00 for everybody else.

Prezi

Equivalence Classes

Babies under 2 -> 0
Children > 2 && < 18 -> 1
Adults > 18 && < 65 -> 2
Senior Citizens > 65 -> 1

Note that babies and seniors are NOT the same equivalence class!

Another Example...

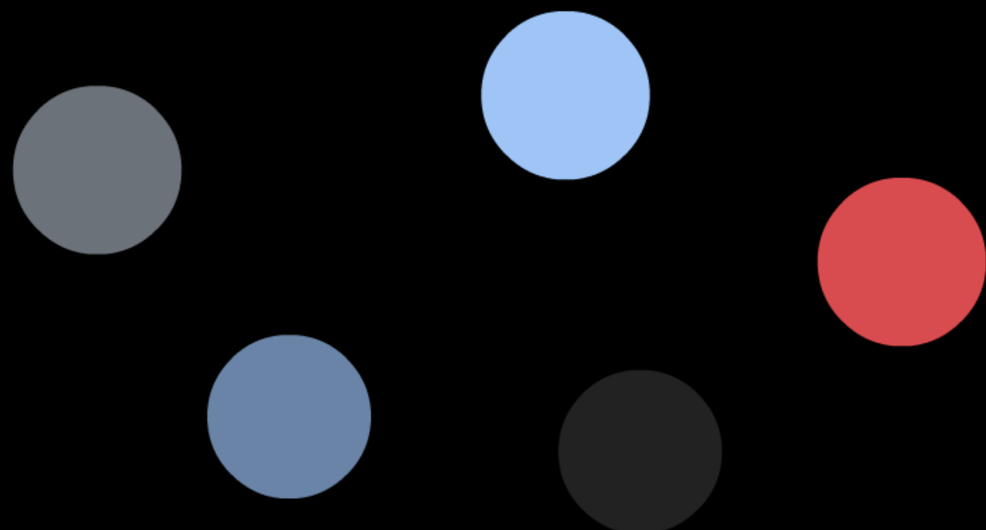
Undergrad students get 20% off pizza
Grad students get 20% off pizza
TAs get 10% off pizza

Offers can be combined.
TAs can be undergrad, grad, or neither.
Students must be EITHER grad xor undergrad - can't be both.
Final discount is % addition

Equivalence Classes

Undergrad only -> 20%
Grad only -> 20%
TA only -> 10%
Undergrad + TA -> 30%
Grad + TA -> 30%

Equivalence classes must be PARTITIONED



NO

They need to have a

STRICT PARTITIONING

Another example...

Imagine an online store that sells one item (a "quux"). Users can add or remove this item from their shopping cart by clicking + or - buttons. Users can buy 1 or more quuxes. Users can remove quuxes from their shopping cart. Cart displays EMPTY when no quuxes in it.

1. User adds quux to empty cart (+ : 0 -> 1)
2. User adds quux to non-empty cart (+ : (n>0 -> n+1))
3. User removes quux, making cart empty (- : 1 -> 0)
4. User removes quux, cart is not empty (- : n>0 -> n-1)
5. User attempts to remove quux from empty cart (- : 0)

Note how we reduced a potentially limitless testing set ((+ : 5 -> 6, (+ : 6 -> 7), etc.) to five test cases.

Boundary Value Testing

Theory:
Problems are more prevalent on boundaries of equivalence classes, less prevalent in the middle.

Example: Bus rides are...
... free for children under 2 years old.
... $1.00 for children under 18, but older than 2.
... $1.00 for senior citizens, 65 or older.
... $2.00 for everybody else.

Equivalence Classes
Babies =
[0,1]
Children =
[2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17]
Adults =
[18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64]
Seniors =
[65,66..INF]

Where are problems most likely?
Babies =
[0,1]
Children =
[2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17]
Adults =
[18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64]
Seniors =
[65,66,67..INF]

So you try to test the boundaries as well as the "interior values"...
Babies =
[0,1]
Children =
[2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17]
Adults =
[18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64]
Seniors =
[65,66,67..INF]

"Hidden" Boundary Values (a/k/a Implicit Boundary Values
* MAXINT, MININT
* Resource limitations
* Allocation limitations
* Undefined values (e.g., sqrt(-1)

Prezi

Theory:
Problems are more prevalent on boundaries of equivalence classes, less prevalent in the middle.

Example: Bus rides are...
... free for children under 2 years old.
... $1.00 for children under 18, but older than 2.
... $1.00 for senior citizens, 65 or older.
... $2.00 for everybody else.

Prezi

Equivalence Classes

Babies =

[0,1]

Children =

[2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17]

Adults =

[18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64]

Seniors =

[65,66..INF]

Where are problems most likely?

Babies =
[0,1]
Children =
[2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17]
Adults =
[18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64]
Seniors =
[65,66,67..INF]

So you try to test the boundaries as well as the "interior values"...

Babies =
[0,1]
Children =
[2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17]
Adults =
[18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64]
Seniors =
[65,66,67..INF]

"Hidden" Boundary Values (a/k/a Implicit Boundary Values

* MAXINT, MININT
* Resource limitations
* Allocation limitations
* Undefined values (e.g., sqrt(-1)

Prezi

Base Cases, Edge Cases, Corner Cases
Success Cases, Failure Cases

Base Case

An element in an equivalence class that is not around a boundary, OR, an expected use case.

Edge Case

An element in an equivalence class that is next to/near a boundary, OR, an unexpected use case.

Examples

```
Babies =
[0,1]
Children =
[2,3,4,5,6,7,8,9,10,11,12,13,14
,15,16,17]
Adults =
[18,19,20,21,22,23,24,25,26,27,
28,29,30,31,32,33,34,35,36,37,3
8,39,40,41,42,43,44,45,46,47,48
,49,50,51,52,53,54,55,56,57,58,
59,60,61,62,63,64]
Seniors =
[65,66,67..INF]
```

Corner Case
(or Pathological Case)

Cases which only occur outside of normal operating parameters. By analogy with "edge case" - where multiple edges intersect.

Corner Cases

```
-1, 3 + 7i, 9.3, "foo"
Babies =
[0,1]
Children =
[2,3,4,5,6,7,8,9,10,11,12
,13,14,15,16,17]
Adults =
[18,19,20,21,22,23,24,25,
26,27,28,29,30,31,32,33,3
4,35,36,37,38,39,40,41,42
,43,44,45,46,47,48,49,50,
51,52,53,54,55,56,57,58,5
9,60,61,62,63,64]
Seniors =
[65,66,67..INF]
```

Success Case
vs
Failure Case

Success cases should return the CORRECT value.
Failure cases should do... something else (throw exception, return NaN, return default value, etc.)

Base Case

An element in an equivalence class that is not around a boundary, OR, an expected use case.

Edge Case

An element in an equivalence class that is next to/near a boundary, OR, an unexpected use case.

Prezi

Examples

```
Babies =
[0,1]
Children =
[2,3,4,5,6,7,8,9,10,11,12,13,14
,15,16,17]
Adults =
[18,19,20,21,22,23,24,25,26,27,
28,29,30,31,32,33,34,35,36,37,3
8,39,40,41,42,43,44,45,46,47,48
,49,50,51,52,53,54,55,56,57,58,
59,60,61,62,63,64]
Seniors =
[65,66,67..INF]
```

Corner Case
(or Pathological Case)

Cases which only occur outside of normal operating parameters. By analogy with "edge case" - where multiple edges intersect.

Corner Cases

-1, 3 + 7i, 9.3, "foo"

```
Babies =
[0,1]
Children =
[2,3,4,5,6,7,8,9,10,11,12
,13,14,15,16,17]
Adults =
[18,19,20,21,22,23,24,25,
26,27,28,29,30,31,32,33,3
4,35,36,37,38,39,40,41,42
,43,44,45,46,47,48,49,50,
51,52,53,54,55,56,57,58,5
9,60,61,62,63,64]
Seniors =
[65,66,67..INF]
```

Success Case
vs
Failure Case

Success cases should return the CORRECT value.
Failure cases should do... something else (throw exception, return NaN, return default value, etc.)

Prezi

Black / White / Grey Box Testing

Black-Box Testing

Testing with NO KNOWLEDGE of actual interior structure of application.

Step 1: Input
Step 2: ???
Step 3: Output

Example

Testing a website...
1. Accessing via browser
2. Using curl or similar tool
3. Running scripts against external interface

White Box Testing

Testing the internals of the system; with full knowledge of the code, architecture, etc.

Step 1: Examine code
Step 2: Write tests to test code
Step 3: Execute tests
Step 4: Expected code execution

Example

Testing a website
1. Unit tests
2. Profiling tools
3. Code hooks

Grey Box Testing

A hybrid approach - still using input and output, but informed by the structure of the underlying program.

e.g., classes, comms (TCP vs UDP), algorithms

Step 1: Examine code, architecture, etc.
Step 2: Write tests with this knowledge
Step 3: Input
Step 4: ??? (Well, kinda)
Step 5: Output

Prezi

Black-Box Testing

Testing with NO KNOWLEDGE of actual interior structure of application.

Step 1: Input

Step 2: ???

Step 3: Output

Example

Testing a website...
1. Accessing via browser
2. Using curl or similar tool
3. Running scripts against external interface

White Box Testing

Testing the internals of the system; with full knowledge of the code, architecture, etc.

Step 1: Examine code
Step 2: Write tests to test code
Step 3: Execute tests
Step 4: Expected code execution

Example

Testing a website
1. Unit tests
2. Profiling tools
3. Code hooks

Grey Box Testing

A hybrid approach - still using input and output, but informed by the structure of the underlying program.

e.g., classes, comms (TCP vs UDP), algorithms

Prezi

Step 1: Examine code, architecture, etc.
Step 2: Write tests with this knowledge
Step 3: Input
Step 4: ??? (Well, kinda)
Step 5: Output

Static and Dynamic Testing

Static Testing = Code is not executed

Dynamic Testing = Code is executed (at least partially)

Static Testing Examples

```
Code Reviews
Walkthroughs
Requirement Analysis
Source Code Analysis
  * Model Checking
  * Finite State Analysis
  * Complexity Analysis
```

Example

metrics plugin for Eclipse

```
McCabe's Cyclomatic Complexity
Efferent Couplings
Lack of Cohesion in Methods
Lines Of Code in Method
Number Of Fields
Number Of Levels
Number Of Locals In Scope
Number Of Parameters
Number Of Statements
Weighted Methods Per Class
```

That darned Halting Problem!

Dynamic Testing

```
Code is executed

OBSERVED results
are compared with
EXPECTED results
```

Example

jUnit Unit Testing

```java
@Test
 public void testIterateEven() {
  Collatz c = new Collatz();
  assertEquals(c.iterate(4), 2);
 }

 @Test
 public void testIterateOdd() {
  Collatz c = new Collatz();
  assertEquals(c.iterate(5), 16);
 }
```

Example

rspec Specification Test in Ruby

```ruby
describe Bowling, "#score" do
  it "returns 0 for all gutter game" do
    bowling = Bowling.new
    20.times { bowling.hit(0) }
    bowling.score.should eq(0)
  end
end
```

Example

Selenium Acceptance Test for web app

```java
public class temp script extends SeleneseTestCase {
    public void setUp() throws Exception {
        setUp("http://localhost:8080/", "*iexplore");
    }
    public void testTemp script() throws Exception {
        selenium.open("/BrewBiz/web/");
        selenium.click("link=Start The BrewBiz Example");
        selenium.waitForPageToLoad("30000");
        selenium.type("name=id", "bert");
        selenium.type("name=Password", "biz");
        selenium.click("name=dologin");
        selenium.waitForPageToLoad("30000");
    }
}
```

Static Testing = Code is not executed

Dynamic Testing = Code is executed (at least partially)

Static Testing Examples

Code Reviews
Walkthroughs
Requirement Analysis
Source Code Analysis
* Model Checking
* Finite State Analysis
* Complexity Analysis

Example

metrics plugin for Eclipse

McCabe's Cyclomatic Complexity
Efferent Couplings
Lack of Cohesion in Methods
Lines Of Code in Method
Number Of Fields
Number Of Levels
Number Of Locals In Scope
Number Of Parameters
Number Of Statements
Weighted Methods Per Class

That darned Halting Problem!

Dynamic Testing

Code is executed

OBSERVED results
are compared with
EXPECTED results

Prezi

Example

jUnit Unit Testing

```
@Test
 public void testIterateEven() {
  Collatz c = new Collatz();
  assertEquals(c.iterate(4), 2);
 }

 @Test
 public void testIterateOdd() {
  Collatz c = new Collatz();
  assertEquals(c.iterate(5), 16);
 }
```

Example

rspec Specification Test in Ruby

```
describe Bowling, "#score" do
  it "returns 0 for all gutter game" do
    bowling = Bowling.new
    20.times { bowling.hit(0) }
    bowling.score.should eq(0)
  end
end
```

Example

Selenium Acceptance Test for web app

```
public class temp script extends SeleneseTestCase {
    public void setUp() throws Exception {
        setUp("http://localhost:8080/", "*iexplore");
    }
    public void testTemp script() throws Exception {
        selenium.open("/BrewBizWeb/");
        selenium.click("link=Start The BrewBiz Example");
        selenium.waitForPageToLoad("30000");
        selenium.type("name=id", "bert");
        selenium.type("name=Password", "biz");
        selenium.click("name=dologin");
        selenium.waitForPageToLoad("30000");
    }
}
```

Lecture 2 - Testing Basics