Chapter 2 Overview of Testing

CS4930 Software Testing

An overview of testing

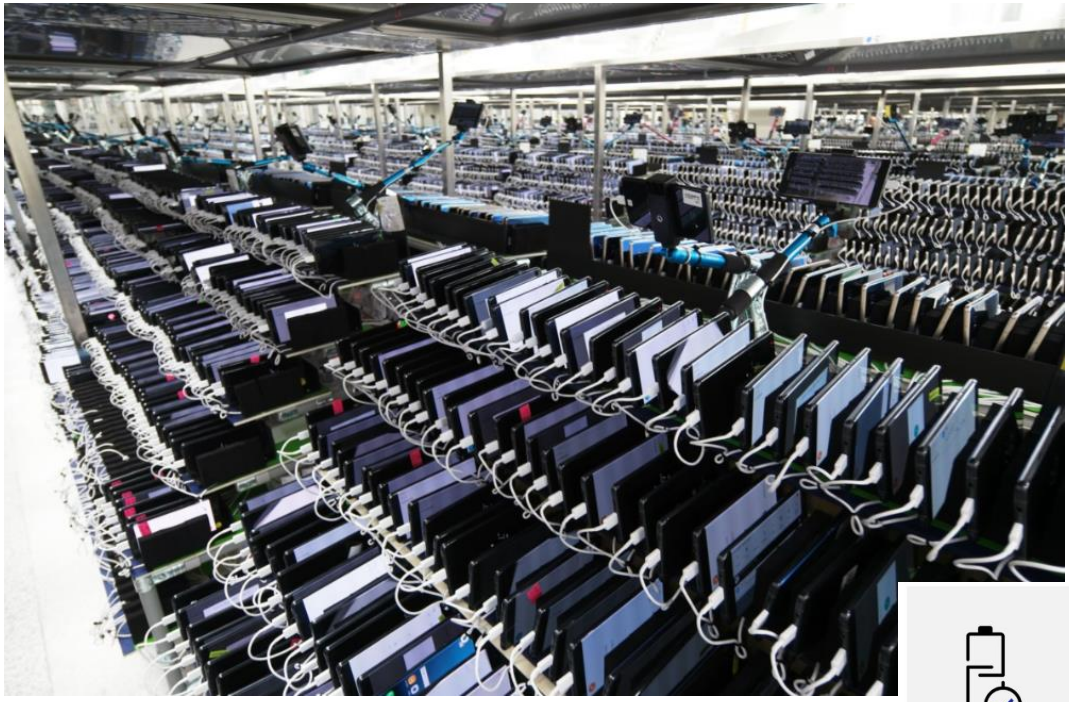
- “As we strive to implement the new features of our application, there is one thing we can say with absolute certainty – that at the same time, we also introduce new defects”

Challenge of testing

- How can we guarantee 100% zero defect

Challenge of testing

- How can we guarantee 100% zero defect



Challenge of testing

- Test Simple function
  - Adds two, thirty two bits numbers together and returns the results. If we assume we can execute 1000 test cases per second, how long will it take to thoroughly test this function ?
  - 585 million years

Challenge of testing

- Formal test design techniques
  - Boundary value analysis
  - Equivalence partitioning

What is software testing

- To ensure a program corresponds to its specification
- To uncover defects in the software
- To make sure the software doesn’t do what it is not supposed to do
- To have confidence that the system performs adequately
- To understand how far we can push the system before it fails
- To understand the risk involved in releasing a system to its users

Formal definitions of testing

- Testing is any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results
  - Positive Testing
- Testing is the process of executing a program or system with the intent of finding defects
  - Negative Testing
- Testing is the process by which we explore and understand the status of the benefits and risk associated with the release of a software system
  - Manage or mitigate risk of failure

Challenge of testing

- Other challenges
  - What if the function needs to interoperate with other functions within the same application ?
  - What if the calculation is driven via a complex graphical user interface with the user able to type the addition values into fields and push the buttons to perform the calculation in any arbitrary order ?
  - What if this function has to be delivered on a number of different hardware platforms, each of which could have different configurations ?

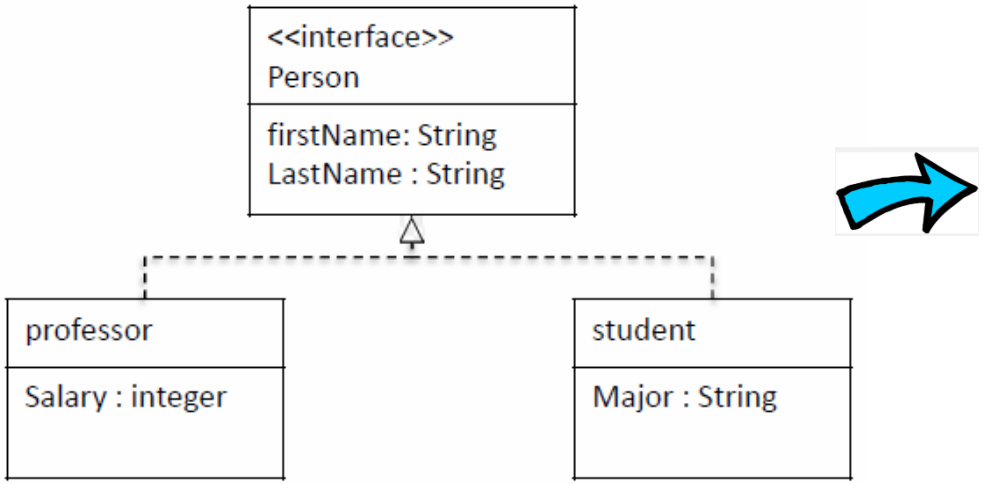
Verification and Validation

- Verifications is
  - the process by which it is confirmed by means of examination and provision of objective evidence that specific requirements have been fulfilled
  - The process of determining whether the products of a given phase of the software development process fulfill the requirements established during the previous phase
  - Are we building the product right ?

- Validation is
  - the process by which it is confirmed that the particular requirements for specific intended use are fulfilled
  - The process of evaluating software at the end of software development to ensure compliance with intended usage
  - Are we building the right product ?

Verification and Validation

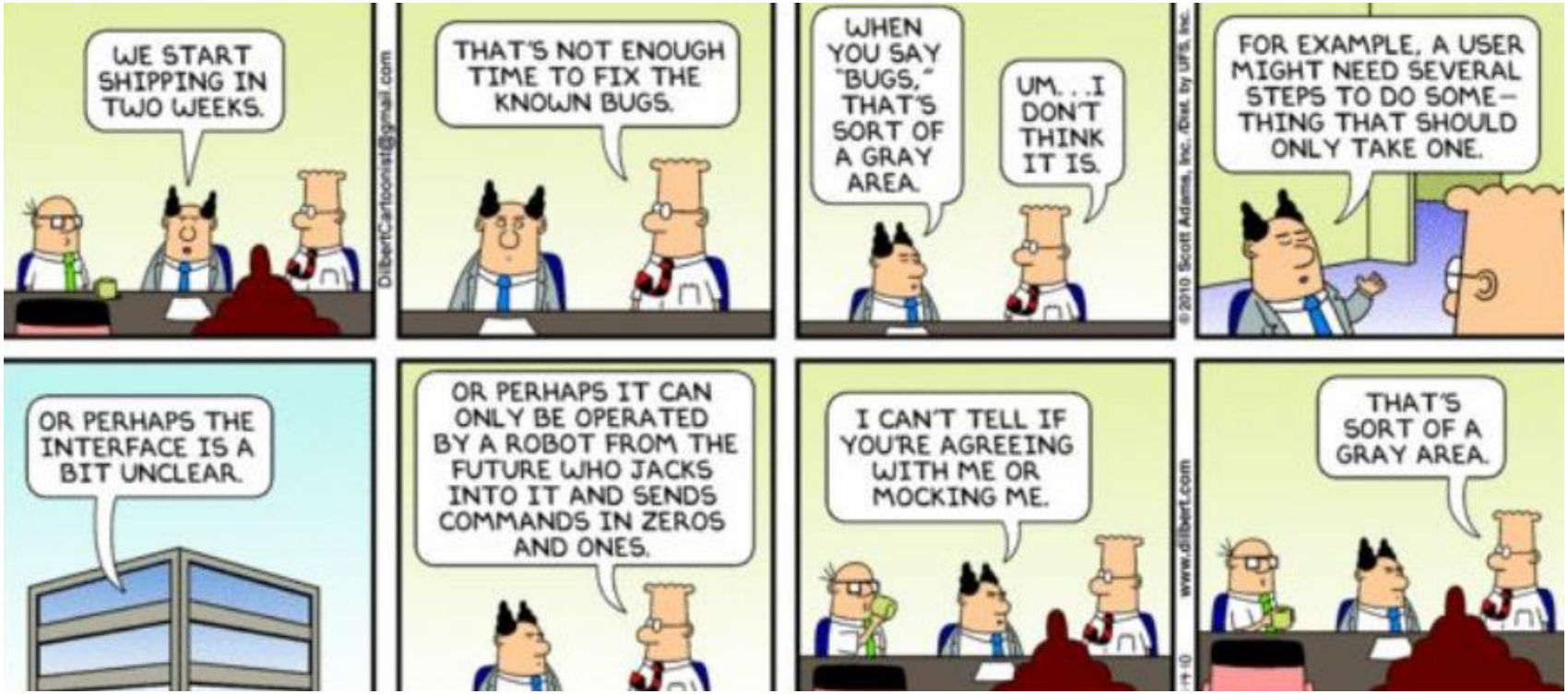
Example



Important Terms
Validation & Verification (*IEEE*)

Example

What is the cost of not testing ?



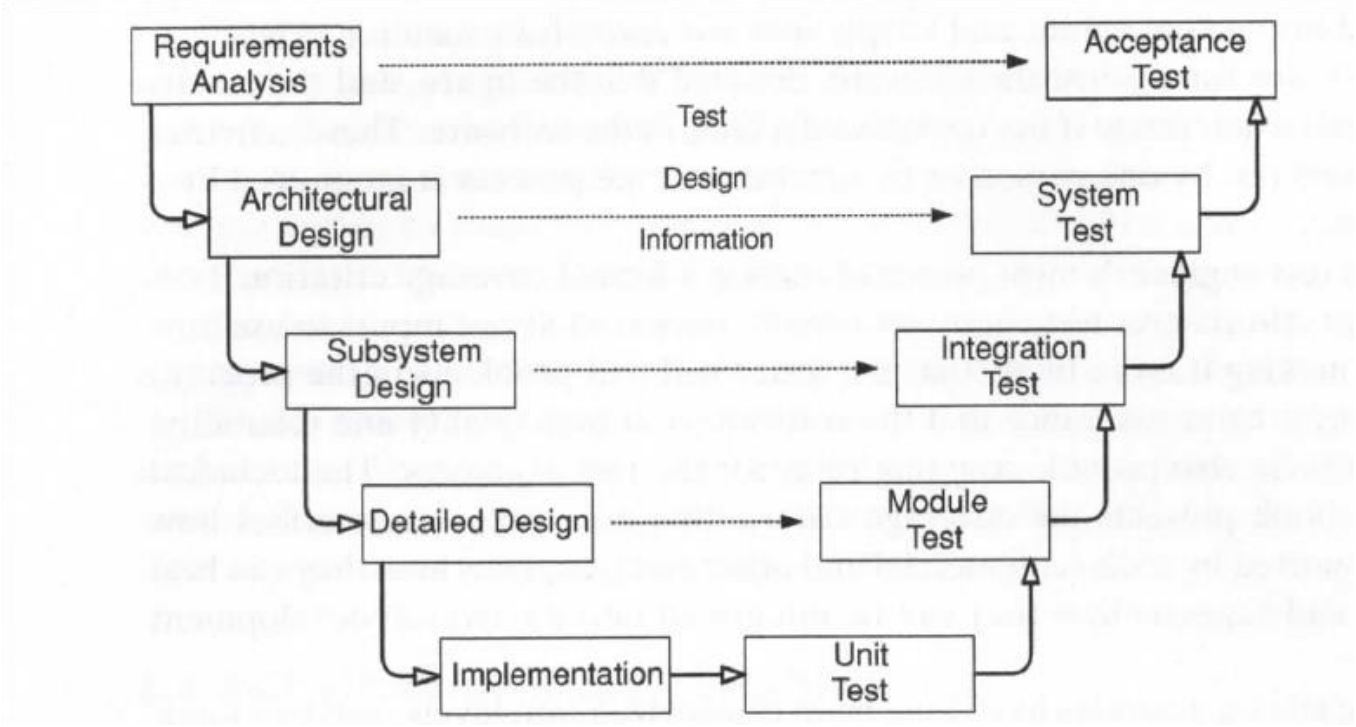
What is the cost of not testing ?

Program Managers often say:
"Testing is too expensive."

- Not testing is even more expensive
- Planning for testing after development is prohibitively expensive
- A test station for circuit boards costs half a million dollars …
- Software test tools cost less than $10,000 !!!

What is the cost of not testing ?

Program Managers often say:
"Testing is too expensive."



Software Faults, Errors & Failures

- Software Fault : A static defect in the software
- Software Failure : External, incorrect behavior with respect to the requirements or other description of the expected behavior
- Software Error : An incorrect internal state that is the manifestation of some fault

Software Faults, Errors & Failures

Example

```
public static int numZero (int[] x) {
  // Effects: if x == null throw NullPointerException
  //    else return the number of occurrences of 0 in x
  int count = 0;
  for (int i = 1; i < x.length; i++)
  {
    if (x[i] == 0)
    {
      count++;
    }
  }
  return count;
}
```

Software Faults, Errors & Failures

Example

```
public int findLast (int[] x, int y) {
//Effects: If x==null throw NullPointerException
//   else return the index of the last element
//   in x that equals y.
//   If no such element exists, return -1
  for (int i=x.length-1; i > 0; i--)
  {
    if (x[i] == y)
    {
      return i;
    }
  }
  return -1;
}
  // test:  x=[2, 3, 5]; y = 2
  //        Expected = 0
```

(a) Identify the fault.
(b) fault, but no error.
(c) error, but **not** a failure.
Hint: Don't forget about the program counter.

Software Faults, Errors & Failures

Example

```
public static int lastZero (int[] x) {
//Effects: if x==null throw NullPointerException
//   else return the index of the LAST 0 in x.
//   Return -1 if 0 does not occur in x

  for (int i = 0; i < x.length; i++)
  {
    if (x[i] == 0)
    {
      return i;
    }
  }
  return -1;
}
  // test:  x=[0, 1, 0]
  //        Expected = 2
```

(a) Identify the fault.
(b) fault, but no error.
(c) error, but **not** a failure.
Hint: Don't forget about the program counter.

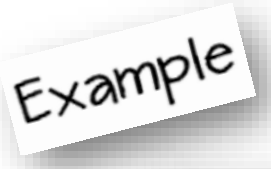
Software Faults, Errors & Failures

Example

```
public int countPositive (int[] x) {
//Effects: If x==null throw NullPointerException
//   else return the number of
//        positive elements in x.
   int count = 0;
   for (int i=0; i < x.length; i++)
   {
      if (x[i] >= 0)
      {
         count++;
      }
   }
   return count;
}
   // test:  x=[-4, 2, 0, 2]
   //        Expected = 2
```

(a) Identify the fault.
(b) fault, but no error.
(c) error, but **not** a failure.
Hint: Don't forget about the program counter.

Software Faults, Errors & Failures




```
public class Test{
    public static void main(String[] args){
       int i=1; int L-value, H-value;
      if ( L-value == 1 ||  H-value == 1) {
          temp=1;
      } else {
          temp = 32 * L-value+H-value;
        }
     }
}
```

3

If instead of 32, we had 3 erroneously.

Testing & Debugging

- Testing : Finding inputs that cause the software to fail
- Debugging : The process of finding a fault given a failure

Fault & Failure Model

(RIP) Three conditions necessary for a failure to be observed

1. Reachability : The location or locations in the program that contain the fault must be reached

2. Infection : The state of the program must be incorrect

3. Propagation : The infected state must propagate to cause some output of the program to be incorrect

Testing – the bottom line

- “Zero Defect Software” , “Defect Free System”
- It is impossible to perform sufficient testing to be completely certain a given is defect free
- Making the process as efficient as possible

Testing Levels Based on Test Process Maturity

- Level 0 : There's no difference between testing and debugging
- Level 1 : The purpose of testing is to show correctness
- Level 2 : The purpose of testing is to show that the software doesn't work
- Level 3 : The purpose of testing is not to prove anything specific, but to reduce the risk of using the software
- Level 4 : Testing is a mental discipline that helps all IT professionals develop higher quality software

Level 0 Thinking

- Testing is the same as debugging
- Does not distinguish between incorrect behavior and mistakes in the program
- Does not help develop software that is reliable or safe

Level 1 Thinking

- Purpose is to show correctness
- Correctness is impossible to achieve
- What do we know if no failures?
  - Good software or bad tests?
- Test engineers have no:
  - Strict goal
  - Real stopping rule
  - Formal test technique
  - Test managers are **powerless**

This is what hardware engineers often expect

Level 2 Thinking

- Purpose is to show failures
- Looking for failures is a negative activity
- Puts testers and developers into an adversarial relationship
- What if there are no failures?

Level 3 Thinking

- Testing can only show the presence of failures
- Whenever we use software, we incur some risk
- Risk may be small and consequences unimportant
- Risk may be great and the consequences catastrophic
- Testers and developers work together to reduce risk

This describes a few "enlightened" software companies

Level 4 Thinking

A mental discipline that increases quality

- Testing is only one way to increase quality
- Test engineers can become technical leaders of the project
- Primary responsibility to measure and improve software quality
- Their expertise should help the developers

This is the way "traditional" engineering works