Why Do We Test Software?

Dr. Isaac Griffith Idaho State University

Inspiration

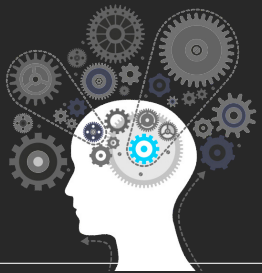
"A true professional does not waste the time and money of other people by handing over software that is not reasonably free of obvious bugs; that has not undergone minimal unit testing; that does not meet the specifications and requirements; that is gold-plated with unnecessary features; or that looks like junk." – Daniel Read

Outcomes

Idaho State University | Computer Science

After today's lecture you will be able to:

- Understand the current nature of software.
- Understand the difference between a fault, an error, and a failure.
- Understand the cost of software failures
- Understand the testing in the 21st century
- Understand test process maturity.
- Understand the goals of software testing.



The 21st Century

CS 3321

Testing in the 21st Century

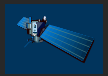
Idaho State University | Computer Science

- Software defines **behavior**
  - network routers, finance, switching networks, other infrastructure
- Today's software **market**:
  - is much **bigger**
  - is more **competitive**
  - has more **users**
- **Embedded Control** Applications
  - airplanes, air traffic control
  - spaceships
  - smartphones
  - memory seats
- **Agile** processes put increased pressure on testers
  - **Programmers** must **unit** test - with no training or education!
  - Tests are key to **functional requirements** - but who builds those tests?

Software is a Skin of Civilization














Software Faults, Errors & Failures

Idaho State University | Computer Science

- **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

Faults in software are equivalent to design mistakes in hardware.
Software does not degrade.

Fault and Failure Example

Idaho State University | Computer Science

- A patient gives a doctor a list of **symptoms**
  - **Failures**
- The doctor tries to diagnose the root cause, the **ailment**
  - **Fault**
- The doctor may look for **anomalous internal conditions** (high blood pressure, irregular heartbeat, bacteria in the blood stream)
  - **Errors**

Most medical problems result from external attacks (bacteria, viruses) or physical degradation as we age. Software faults were there at the beginning and do not ”appear” when a part wears out.

A Concrete Example


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

A Concrete Example


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

- **Fault**: Should start search at 0, not 1

A Concrete Example


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

- **Fault**: Should start search at 0, not 1
- Test 1:
  - Input: [2, 7, 0]
  - Expected: 1
  - **Error**: i is 1, not 0, on the first iteration
  - **Failure**: none

A Concrete Example

Idaho State University | Computer Science

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

- **Fault**: Should start search at 0, not 1

A Concrete Example

Idaho State University | Computer Science

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

- **Fault**: Should start search at 0, not 1
- Test 2:
  - Input: [0, 2, 7]
  - Expected: 1
  - **Error**: i is 1, not 0, Error propagates to the variable count
  - **Failure**: count is 0 at the return statement

The Term Bug

Idaho State University | Computer Science

- **Bug** is used informally
- Sometimes **speakers mean fault**, sometimes **error**, sometimes **failure** … often the speaker doesn't know what it means!
- This class will try to use words that have **precise**, **defined**, and **unambiguous** meanings.

"It has been just so in all of my inventions. The first step is an intuition, and comes with a burst, then difficulties arise–this thing gives out and [it is] then that '**Bugs**'–as such little faults and difficulties are called–show themselves and months of intense watching, study and labor are requite…"
– Thomas Edison

The Term Bug

Idaho State University | Computer Science

- **Bug** is used informally
- Sometimes **speakers mean fault**, sometimes **error**, sometimes **failure** … often the speaker doesn't know what it means!
- This class will try to use words that have **precise**, **defined**, and **unambiguous** meanings.

"an analyzing process must equally have been performed in order to furnish the Analytical Engine with the necessary operative data; and that herein may also lie a possible source of **error**. Granted that the actual mechanism is unerring in its processes, the cards may give it wrong orders." – Ada, Countess Lovelace (notes on Babbage's Analytical Engine)

Spectacular Software Failures

Idaho State University | Computer Science

- **NASA's Mars Lander**: September 1999, crashed due to a units integration fault.
- **THERAC-25 radiation machine**: Poor testing of safety-critical software can cost *lives*: 3 patients were killed
- **Ariane 5 explosion**: Exception-handling bug: forced self-destruct on maiden flight (64-bit to 16-bit conversion) ~$370 million
- **Intel's Pentium FDIV fault**: public relations nightmare (not to mention invalidation of scientific results)

Spectacular Software Failures

Idaho State University | Computer Science

- **NASA's Mars Lander**: September 1999, crashed due to a units integration fault.
- **THERAC-25 radiation machine**: Poor testing of safety-critical software can cost *lives*: 3 patients were killed
- **Ariane 5 explosion**: Exception-handling bug: forced self-destruct on maiden flight (64-bit to 16-bit conversion) ~$370 million
- **Intel's Pentium FDIV fault**: public relations nightmare (not to mention invalidation of scientific results)

We need our software to be dependable
Test is *one* way to assess dependability

Northeast Blackout of 2003

- 508 generating units and 256 power plants shut down
- Affected 10 million people in Ontario, Canada
- Affected 40 million people in 8 US states
- Financial losses of $6 Billion USD

- The alarm system in the energy management system failed due to a software error and operators were not informed of the power overload in the system.

Costly Software Failures

Idaho State University | Computer Science

- NIST report, “The **Economic Impacts** of Inadequate Infrastructure for Software Testing” (2002)
  - Inadequate software testing costs the US alone between $22 and $59 billion annually
- **Huge losses** due to web application failures
  - **Financial** services: $6.5 million per hour (just in USA)
  - **Credit card sales** applications: $2.4 million per hour
- In Dec 2006, amazon.com’s **BOGO** offer turned into a **double discount**

Costly Software Failures

Idaho State University | Computer Science

- NIST report, "The **Economic Impacts** of Inadequate Infrastructure for Software Testing" (2002)
  - Inadequate software testing costs the US alone between $22 and $59 billion annually
- **Huge losses** due to web application failures
  - **Financial** services: $6.5 million per hour (just in USA)
  - **Credit card sales** applications: $2.4 million per hour
- In Dec 2006, amazon.com's **BOGO** offer turned into a **double discount**

World-wide monetary loss due to poor software is staggering

Testing in the 21st Century

Idaho State University | Computer Science

- More **safety** critical, **real-time** software
- **Embedded** software is ubiquitous … check your pockets
- **Enterprise** applications means bigger programs, more users
- Paradoxically, free software **increases** our expectations!
- **Security** is now all about software faults
  - **Secure** software is **reliable** software
- The **web** offers a new deployment platform
  - Very **competitive** and very **available** to more users
  - Web apps are distributed
  - **Web apps** must be highly reliable

Industry desperately needs our inventions!

What Does this Mean?

What Does this Mean?

Software testing is extremely important

What Does this Mean?

Idaho State University | Computer Science

Software testing is extremely important

What are we trying to do when we test?
What are our goals?

Test Process Maturity

CS 3321

Verification & Validation (IEEE)

- **Validation**: The process of evaluating software at the end of software development to ensure compliance with intended usage

- **Verification**: The process of determining whether the products of a given phase of the software development process fulfill the requirements established during the previous phase

Verification & Validation (IEEE)

- **Validation**: The process of evaluating software at the end of software development to ensure compliance with intended usage

- **Verification**: The process of determining whether the products of a given phase of the software development process fulfill the requirements established during the previous phase

IV&V stands for "independent verification and validation"

Test Process Maturity

Idaho State University | Computer Science

- **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 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**

This is what we teach undergraduate CS majors

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**

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 2 Thinking

Idaho State University | Computer Science

- 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**?

This describes most companies.

How can we move to a team approach??

Level 3 Thinking

Idaho State University | Computer Science

- 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 consequences catastrophic

Level 3 Thinking

Idaho State University | Computer Science

- 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 consequences catastrophic

This describes a few ”enlightened” software companies

Level 4 Thinking

Idaho State University | Computer Science

A mental discipline that increase quality

Level 4 Thinking

A mental discipline that increase 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**

A mental discipline that increase 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

Where are You?

Idaho State University | Computer Science

Are you at level 0, 1, or 2?

Is your organization at work at level 0, 1, or 2? Or 3?

We hope to teach you to become ”change agents” in your workplace...
Advocates for level 4 thinking

Goals and Costs

CS 3321

Tactical Goals: Why Each Test?

Idaho State University | Computer Science

If you don't know why you're conducting each test, it won't be very helpful

Tactical Goals: Why Each Test?

Idaho State University | Computer Science

If you don't know why you're conducting each test, it won't be very helpful

- **Written test objectives** and requirements must be documented
- What are your planned **coverage** levels?
- How much testing is **enough**?
- Common objective - **spend the budget** … **test until the ship-date** …
  - Sometimes called the "**date criterion**"

Why Each Test?

Idaho State University | Computer Science

If you don’t start planning for each test when the functional requirements are formed, you’ll never know why you’re conducting the test

Why Each Test?

If you don't start planning for each test when the functional requirements are formed, you'll never know why you're conducting the test

- 1980: "The software shall be easily **maintainable**"
- Threshold **reliability** requirements?
- What fact does each test try to **verify**?
- **Requirements** definition teams need testers!

Cost of Not Testing

Poor Program Managers might say: ”Testing is too expensive.”

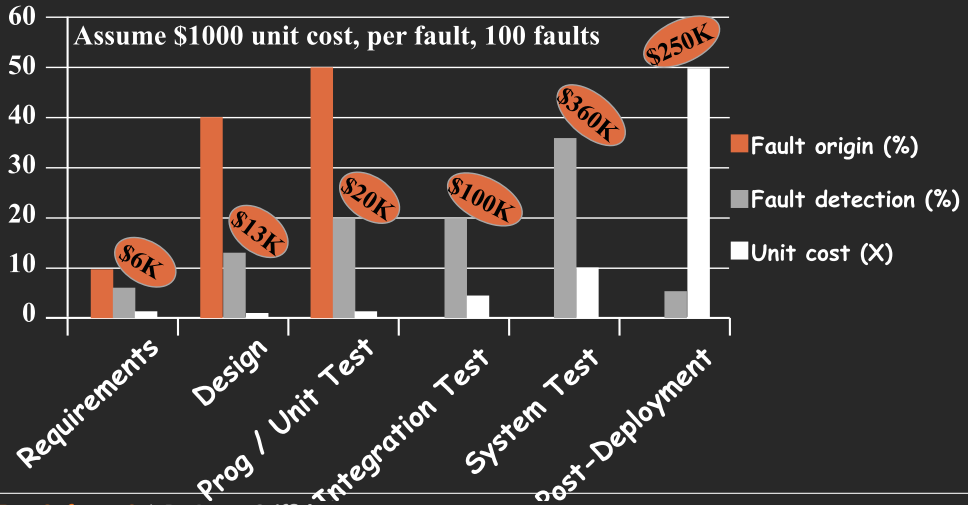
Cost of Not Testing

Idaho State University | Computer Science

Poor Program Managers might say: ”Testing is too expensive.”

- Testing is the **most time consuming** and expensive part of software development
- *Not* testing is even **more expensive**
- If we have too little testing effort early, the cost of testing **increases**
- Planning for testing after development is **prohibitively** expensive

Cost of Late Testing



Summary

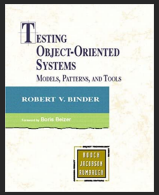
A tester's goal is to eliminate faults as early as possible

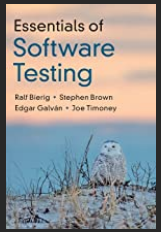
- Improve quality
- Reduce cost
- Preserve Customer Satisfaction

Things to Learn

Idaho State University | Computer Science

- Input Space Partitioning
- Boundary Value Analysis
- Decision Table Testing
- Black-, White-, and Gray-Box Testing
- Graph Coverage
  - Statement Coverage
  - Branch Coverage
  - All Paths Coverage
- OO Testing
- Application Level Testing
- Mutation Testing
- Metamorphic Testing
- Test Planning and Automation





For Next Time

- Review the Reading
- Review this Lecture
- Come to Class

Are there any questions?