EN.601.422 / EN.601.622

Software Testing & Debugging

Instructor and Meetings

- Dr. Ali Darvish (darvish@jhu.edu, www.cs.jhu.edu/~darvish/)
  Office hours: TBD
- **Meetings:** Tuesday and Thursday 1:30pm-2:45pm on Zoom
- **Course Website:** http://jhu-st.github.io/cs422_s21
  Piazza: http://piazza.com/jhu/spring2021/en601422622
  Gradescope: https://www.gradescope.com/courses/227037
- **TAs:** Farnaz Yousefi, Srishti Dhamija

  Office hours: TBD

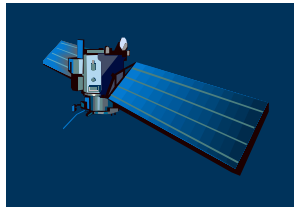
Plan for Today

- Why “Software Testing”?
- Logistics
- (warm-up) Exercise

Software is everywhere

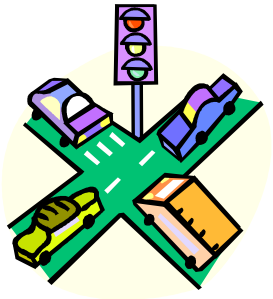


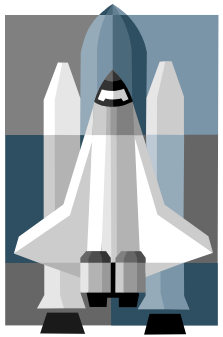











Do you see any bugs?

```
public class NumFinder {
 private int smallest = Integer.MAX_VALUE;
 private int largest = Integer.MIN_VALUE;

 public void find(int[] nums) {
   for(int n : nums) {
     if (n < smallest)
       smallest = n;
     else if (n > largest)
       largest = n;
   }
 }
  // getters for smallest and largest
}
```

Terminology

- **Fault**: a static defect in the software
- **Error**: An incorrect internal state
- **Failure**: External incorrect behavior with respect to the requirements or expected behavior
- **Bug**: an informal term used in place of any of the above
- **Debugging**: Given a failure, find/fix the fault

Fault and Failure Analogy

- 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

Fault vs Error vs Failure

```
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: i should start from 0 not 1

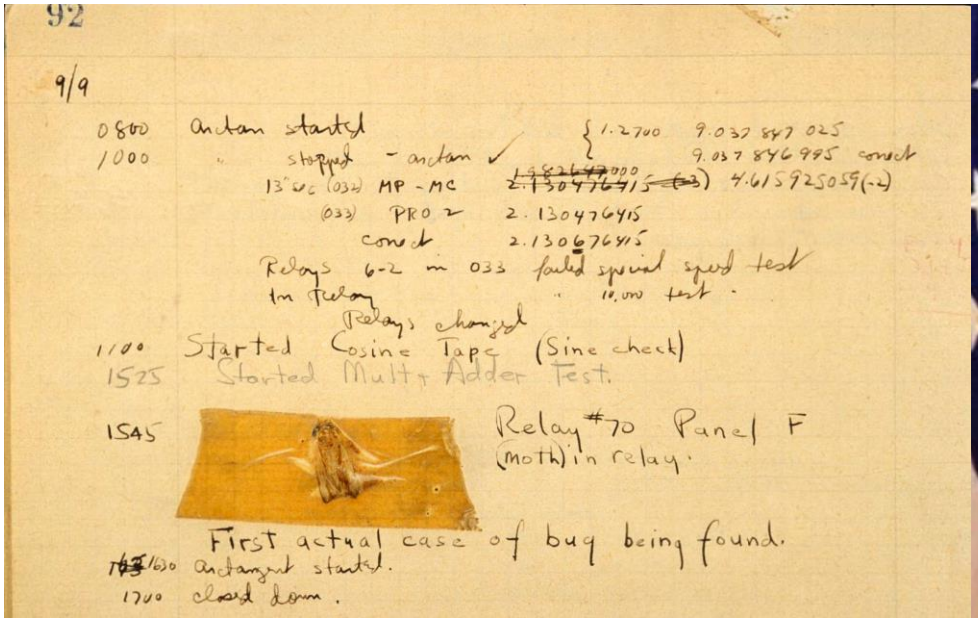
Error: i is 1, not 0, on the first iteration
Failure: none

Error: i is 1, not 0, on
The error propagates to count
Failure: count is 0 at the end

Test 1
[2, 7, 0]
Expected: 1
Actual: 1

Test 2
[0, 2, 7]
Expected: 1
Actual: 0

First Ever Bug



Grace Hopper found a moth stuck between the relays on the Harvard Mark II computer she was working on

Mars Climate Orbiter Loss

- https://en.wikipedia.org/wiki/Mars_Climate_Orbiter

Therac-25 Radiation Therapy Machine

- Therac-25: the killer machine!
  - race conditions in the codebase led to excessive radiations and death of three patients!

Ariane 5 Rocket Explosion on June 4, 1996

https://www.youtube.com/watch?v=PK_yguLapgA

Ariane 5 Rocket Explosion on June 4, 1996

- **Cause**: a 64-bit floating-point number relating to the horizontal velocity of the rocket with respect to the platform was converted to a 16-bit signed integer causing a data overflow.

- **Estimated Cost**: a decade of development costing $7 billion

https://www.youtube.com/watch?v=PK_yguLapgA

Northeast Blackout of 2003

- https://en.wikipedia.org/wiki/Northeast_blackout_of_2003
- The outage affected about 10 million people in Canada and about 40 million in eight U.S. states.

Northeast Blackout of 2003

- **Cause**: software bug in the alarm system in the control room of FirstEnergy, an Akron, Ohio–based company, which rendered operators unaware of the need to redistribute load after overloaded transmission lines drooped into foliage. What should have been a manageable local blackout cascaded into collapse of the entire Northeast region.
- **Estimated Cost**: contributing to 11 deaths and about $6 billion USD

Toyota Prius ABS Brake Software Recall

- https://www.usnews.com/news/articles/2014/02/12/toyota-to-recall-19-million-prius-cars-due-to-software-problem

Toyota to Recall 1.9 Million Prius Cars Due to Software Problem

The company says a software problem can cause the vehicles to stop while they are being driven.

By **Danielle Kurtzleben**, Staff Writer Feb. 12, 2014, at 11:06 a.m.

Toyota is recalling nearly 2 million of its Prius hybrids. DAVID ZALUBOWSKI/AP

Korean Education Software Fault

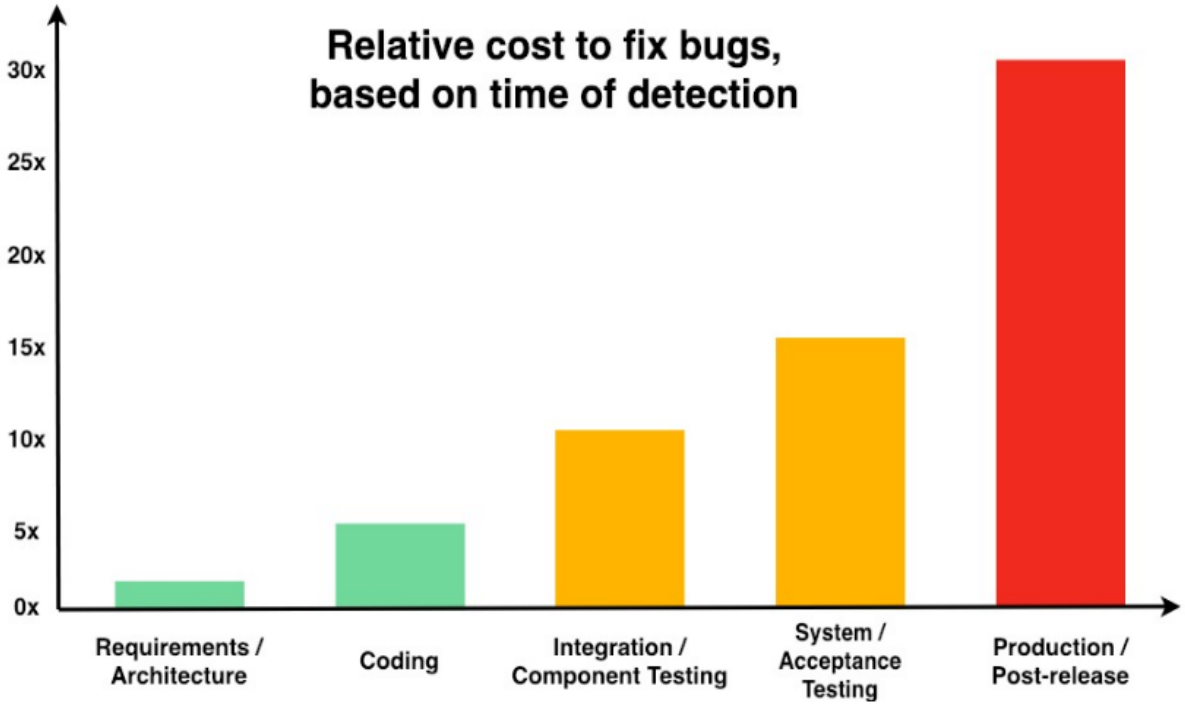
- http://koreajoongangdaily.joins.com/news/article/article.aspx?aid=2939367
- In 2011, Samsung Electronics' Education info system miscalculated grades of about 29000 middle and high students, affecting the college admission process leading to a government investigation

Importance of Software Testing

- A 2002 report by NIST:
  - Defective software costs US economy $59.5 billion per year
  - Improved testing could reduce the cost by one third
- Blumenstyk reported web application failures cost:
  - $150K in media companies
  - $2.4 million per hour in credit card sales
  - $6.5 million per hour in financial service market

https://www.nist.gov/sites/default/files/documents/director/planning/report02-3.pdf
Blumenstyk, Web application development – bridging the gap between QA and development, www.stickyminds.com, 2006

Cost of Bugs!



"Most defects end up costing more than it would have cost to prevent them. Defects are expensive when they occur, both the direct costs of fixing the defects and the indirect costs because of damaged relationships, lost business, and lost development time." *—Kent Beck, Extreme Programming Explained*

What Does This Mean?

Software Testing is a crucial activity

The Goal of Software Testing

Why do we test software?

Ensuring that the software works as expected

The Goal of Software Testing

Why do we test software?

~~Ensuring that the software works as expected~~

The Goal of Software Testing

Why do we test software?

To produce high quality software by revealing and repairing as many faults as possible

The Goal of Software Testing

Why do we test software?

“Software testing should be used to show the presence of bugs, not their absence”

– Edward W. Dijkstra

The Goal of Software Testing

- Issue with the “common sense” goal:

show correctness ➔ impossible/impractical to achieve, due to (potentially) infinite number of possible test input values

Psychology of Software Testing

- Psychological issue of the “common sense” goal:

show correctness ➔ will be inclined (subconsciously) towards your goal ➔ write tests that work fine rather than cause failures

Psychology of Software Testing

- A fundamental issue of the “common sense” goal:

show correctness ➔ what does “correctness” really mean?

The Goal of Software Testing

show correctness ➔ impossible to achieve, due to (potentially) infinite number of possible test input values

exhaustive path testing, via adequate test input values, might seem to be the answer
(i.e., executing all possible paths of control flow graph)

The Goal of Software Testing

- Why exhaustive path testing is not the answer either:
  - Even if we cover all the possible execution paths, the program might be loaded with faults:
    - no guarantee the program complies with its specifications
    - missing paths
    - data sensitivity:

```
if (a - b < c)
    System.out.println (“ a minus b is less than c”);
```

Psychology of Software Testing

- Implications of the “correct” perspective:
  - testing is difficult
  - switch successful test vs unsuccessful test terminology
    - to our thinking, a successful test should reveal a fault
  - A program that does everything it is supposed to do is not necessarily correct
    - might do extra unnecessary things also

Psychology of Software Testing

Testing is a destructive (even sadistic!) process

A major shift in our mentality!

Psychology of Software Testing

eventually we want to increase the confidence about the software

This is best achieved by diligent search for faults

Software Testing is hard!

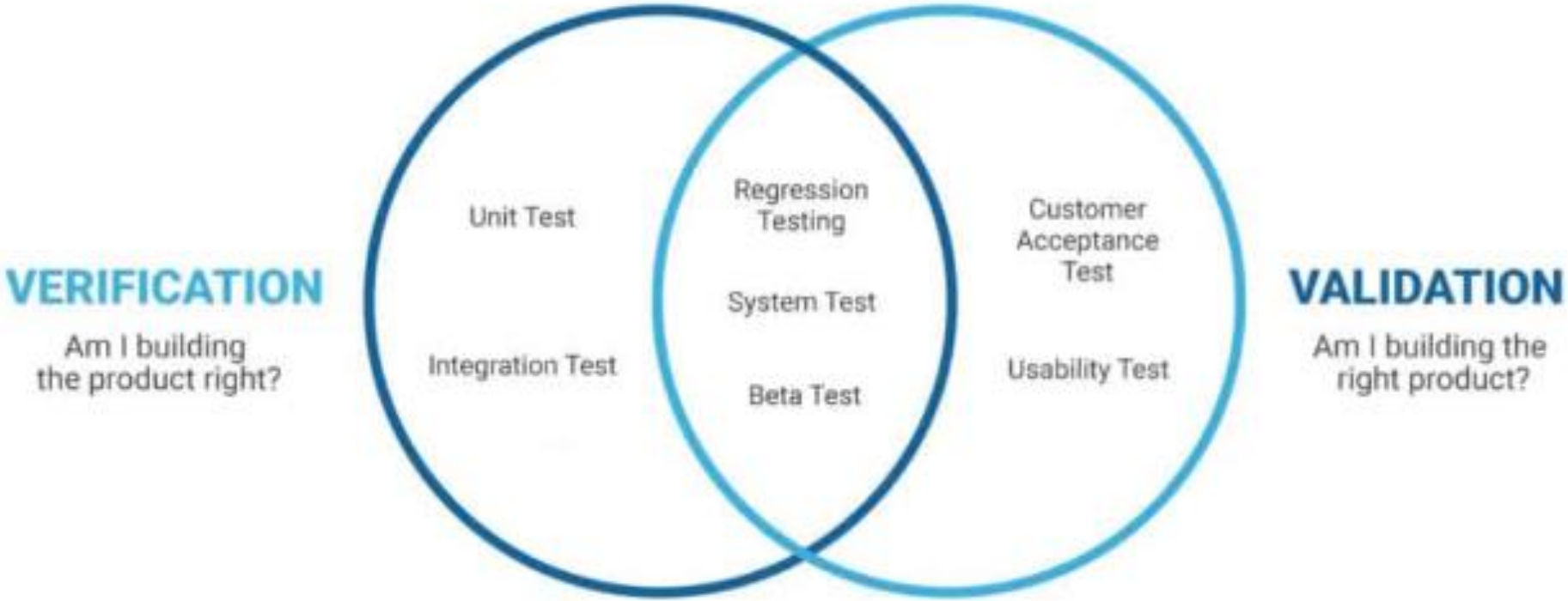
- **Pesticide paradox**: "Every strategy you use to prevent or find bugs leaves a residue of subtler bugs against which those strategies are ineffectual.“
  - e.g., using unit testing only
- Testing is context-dependent:
  - Mobile vs. web vs. desktop
- Absence-of-errors fallacy

Verification & Validation

- **Verification**: determining whether the software fulfills the requirements. In other words, whether the software meets its specifications or not.

- **Validation**: determining whether the produced meets the intended usage (i.e., comprehensively and exclusively captures user needs)

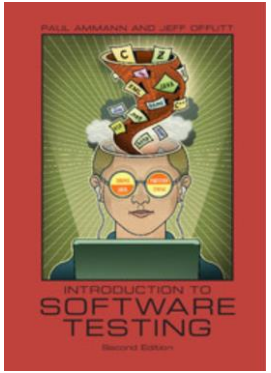
Verification vs Validation



Verification & Validation

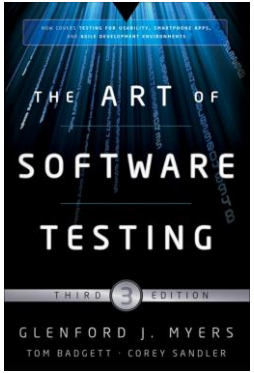
- **Verification**:
  - more technical
  - does not reveal flaws in the requirements/specifications
- **Validation**:
  - depends on the domain knowledge
  - exposes flaws in the specification and requirement deficiencies/discrepancies

Recommended Texts



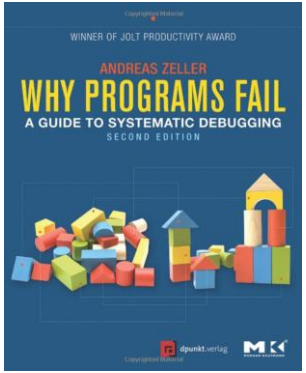
Introduction to Software Testing

2nd Edition by Paul Ammann and Jeff Offutt



The Art of Software Testing

3rd Edition by Glenford Myers et al.



Why Programs Fail

2nd Edition by Andreas Zeller

Software Testing: From Theory to Practice

Online book available at http://sttp.site

Grading Breakdown

Grading Breakdown		Grading Scale	
0%	Participation (expected)	[100%, 99%]; (99%, 93%]; (93%, 90%]	A+; A; A-
35%	4-5 homework assignments	(90%, 88%]; (88%, 83%]; (83%, 80%]	B+; B; B-
15%	Seminar	(80%, 78%]; (78%, 73%]; (73%, 70%]	C+; C; C-
30%	Team-based final project	(70%, 68%]; (68%, 63%]; (63%, 60%]	D+; D; D-
20%	Comprehensive test	(60%, 0%]	F

Homework

- 4-5 Homework assignments
- Solo work
- Submit by the specified deadline
  - *** a budget of three late days***
- Either written (pen-paper), hands-on (programming, automated tools etc.), or combo

Seminar

- Study research papers as a team
- Teamwork (2/3 people)
- Write a thoughtful report
- Make a 20 -25 minutes in-class presentation
- A pool of papers/topics to choose from will be provided

Semester Project

- Hands-on
  - Conduct a systematic and thorough testing and/or debugging on an existing piece of software
- Teamwork (2/3 people)
- Possible deliverables:
  - Testing artifacts if any
  - **REQURIRED** The SUT
  - **REQUIRED** Code, scripts, configuration files, tools etc.
  - **REQUIRED** A thorough final report
  - **REQUIRED** A step-by-step instruction document explaining how to setup and run the tests
  - **REQUIRED** A concise in-class presentation

Comprehensive test

- A closed-book, comprehensive test that covers all the material taught in the class over the semester. It tests your knowledge and understanding of the concepts, theories and methodologies discussed.
- It is more geared to test your understanding of the concepts rather than memorization.