CS 4387/5387
SOFTWARE INTEGRATION AND V&V

DR. SALAMAH SALAMAH
ISALAMAH@UTEP.EDU
OFFICE: CCS.0608
915-747-6671

Outline

- Syllabus
- Learning Outcomes
- Introduction to Software Engineering
- Software Quality Assurance
  - What
  - Why
- Individual Quiz
- Next week
Upon completion of this course, you should be able to:

- Define the elements of software quality,
- Define the terminology commonly utilized in the V&V area,
- Explain and be capable of using representative techniques for software V&V,
- Explain the theoretical and practical limitations of V&V approaches,
- Evaluate the applicability and likely effectiveness of a V&V approach for various artifacts,
- Develop an outline and a V&V plan for a project,
- Evaluate and apply different testing techniques applicable throughout software development cycle,
- Assess the effectiveness of a V&V plan with respect to its objectives,
- And analyze the economics of quality requirements.
Topics

- Appraisal QA Techniques
  - Formal Reviews and Inspections
- Failure QA Techniques
  - Testing
- Formal Verification and Validation Techniques
  - Model checking, theorem proving, and runtime monitoring
- Developing and Monitoring Testing Plans

So What Exactly is Software Engineering?
System Engineering

During the 1950s and 1960s the area of system engineering was developed to address the problem of developing large complex systems in an economical and efficient manner.

- Example systems: Rockefeller Center – NYC, Apollo Space Program, Denver International Airport, IBM System 360 (A Computer System)

- In groups of 3: introduce yourself, and discuss the components of a systems (5 minutes)

- A system consists of:
  - a set of entities; a set of activities; a description of the relationship between entities and activities; and a definition of the boundary of the system.

- Systems Engineering includes the typical engineering techniques concerned with analysis, design, construction and testing of an engineering product; but it also includes management topics such as project organization, planning, scheduling and tracking, estimating, and team management.

---

Computer System Components

- Spend another five minutes with your new-found friends, and come up with the components of computer systems
  - hardware
  - software
  - people
  - procedures
  - documentation
What is Software?

- According to the IEEE
- Software is:
  - Computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system.
    - Code
    - Procedures
    - Documentations
    - Data

Software quality is the sum of the qualities of each of these components.

Software Engineering

- The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software
- Software is a critical component of many products in the current marketplace
  - Medical devices
  - Automobiles
  - Aircraft
  - Mobile devices

Development must adhere to a defined process to collect data for an adequate management.

---

Software Engineering – Good news - 1

B.S. in Software Engineering

<table>
<thead>
<tr>
<th>Occupational Title</th>
<th>Employment, 2008</th>
<th>Projected</th>
<th>Change, 2008-18</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number</td>
<td>Percent</td>
<td>Number</td>
</tr>
<tr>
<td>Computer Programmers</td>
<td>426,700</td>
<td></td>
<td>414,400</td>
</tr>
<tr>
<td>Computer Software Engineers, Applications</td>
<td>514,800</td>
<td>689,900</td>
<td>175,100</td>
</tr>
<tr>
<td>Computer Software Engineers, Systems Software</td>
<td>394,800</td>
<td>515,000</td>
<td>120,200</td>
</tr>
</tbody>
</table>

Software Engineering – Good news - 3

- Best jobs in America in 2015 as reported by CNNMoney
  - # 1: Software Architect (23% job growth 10 year forecast)
  - # 8: Database Developer (23%)
  - # 32: Software QA Manager (15%)


- Glassdoor: 25 Best Jobs in America for 2015
  - # 2: Software Engineer

  http://www.glassdoor.com/blog/jobs-america

- IEEE-CS May 15, 2012: Why Software Engineering is Still the Best Job in the World

  http://buildyourcareerblog.computer.org/2012/05/15/why-software-engineering-is-the-best-job-in-the-world
Software Engineering – Not so good news

- Wide use and complexity of developed software systems is yet to be met with a quality of a workforce of graduates capable of efficiently and correctly implementing complex systems.
- U.S Government Accountability Office reports on the continuing challenges the U.S. government faces in creating large-scale, reliable, software-intensive systems on schedule, within budget, and with expected functionality.
  - 31% of projects are canceled before they ever get completed
  - 52% of projects cost 189% of their original estimates
  - Software errors cost U.S. economy $59.5 billion annually.
  - $22.2 billion can be saved to U.S. economy if verification is done at earlier stages (NIST, June 2002).
- Consequences of software defects
  - Ariane 5 accident

State of the Practice

- Software has proved to be one of the most useful and popular products in modern times.
- Software is an important part of almost every aspect of human endeavor today.
- Professionals with software engineering knowledge and skills are in high demand.
- Unfortunately, serious problems still need to be addressed:
  - problems with schedule
  - problems with cost
  - problems with quality
Many software products have poor quality.
- Software system testing typically finds 25 defects per 1000 lines of code.

Software development is costly.
- On average, software projects overrun their original cost estimates by 189%.

Serious schedule overruns are common.
- On average, the time overrun for software projects is 222% of the original time.

In your group:
- Can you think of reasons behind the problems in software engineering (7 minutes)

Software is becoming larger and more complex.
- A Microsoft Office installation consumes over 3GB of memory
- percent software requirements to system requirements
  - B2 bomber => 60%
  - F22 fighter => 80%

Others?
Goal of Software Engineering

- A commonly stated goal of software engineering is to deliver quality software, on time, and within budget.

“No Silver Bullet” - Fred Brooks

Difficulties in developing software:
- essence - inherent difficulty
- accident - temporal difficulty (not eternal)

[Brooks 95]
Essences

- complexity
- changeability
- invisibility

Accidents

- programming languages
- response time
- dissimilar and non-standard programming environments
Proposed Silver Bullets

- Ada, et al
- OOP
- AI
- Automatic/ Graphical Programming
- Program Verification
- Environments/Tools/Workstations

What is Software Quality?

- Any ideas?

- Different views:
  - How well does the software satisfy the customer requirements?
  - How well does it satisfy the needs of the user?
  - How effective was the development of the software?
  - How easy is the software to use and maintain?

- Three Quality Dimensions
  - Quality of Product
  - Quality of Process
  - Quality related to Business Environment
Software Quality Characteristics -1

- Reliability:
  - provides the services expected by the users in a correct and consistent manner

- Efficiency:
  - does not waste computer resources (space and time)

- Maintainability:
  - easy to modify

- Usability:
  - offers appropriate user interfaces and external documentation

Software Quality Characteristics -2

- Additional desirable characteristics
  - Easy to Understand
  - Easy to Test
  - Low Cost
  - Reusable Components
  - Delivered On Time
  - Portable

- Overall Objective:
  - produce quality software, on time, within budget
Summary

- Software has become critical to advancement in almost all areas of human endeavor.

- There are serious problems in the development of complex software systems.

- Software Engineering has the objective of solving these problems by producing quality software, on time, within cost.

- To achieve this objective we have to focus in a disciplined manner on both the quality of the product and of the process used to develop the product.

Five Minutes Break
Uniqueness of Software Systems

- Testing a ballpoint pen
  - Does the pen write in the right color, with the right line thickness?
  - Is the logo on the pen according to company standards?
  - Is it safe to chew on the pen?
  - Does the click-mechanism still work after 100000 clicks?
  - Does it still write after a car has run over it?

Goal: develop software to meet its intended use!

- But: human beings make mistake!
- Products of any engineering activity must be verified against its requirements throughout its development
  - Bridge, Automobile, Television, Word Processor

- To verify a bridge, engineers must verify its design, construction, process…

- Software must be verified in the same spirit. BUT, how easy is this?
In Class Reading (AMGAL Example)

- 5 minutes
- How could software developers get away with this?

What is Unique About Software Systems

- What is special about software products and why the emphasis on software quality?

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Software Products</th>
<th>Other Industrial Products</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complexity</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What is Unique About Software Systems

- What is special about software products and why the emphasis on software quality?

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Software Products</th>
<th>Other Industrial Products</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complexity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Visibility</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Software Products</th>
<th>Other Industrial Products</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complexity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Visibility</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nature of development and production process</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What is Unique About Software Systems

- What is special about software products and why the emphasis on software quality?

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Software Products</th>
<th>Other Industrial Products</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complexity</td>
<td>Usually very complex; Large number of operational options</td>
<td></td>
</tr>
<tr>
<td>Visibility</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nature of development and production process</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What is Unique About Software Systems

- What is special about software products and why the emphasis on software quality?

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Software Products</th>
<th>Other Industrial Products</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complexity</td>
<td>Usually very complex; Large number of operational options</td>
<td>Much lower complexity; Lower number of options (thousands)</td>
</tr>
<tr>
<td>Visibility</td>
<td>Invisible: impossible to detects omissions by sight</td>
<td>Visible: easy to detect defects by sight</td>
</tr>
<tr>
<td>Nature of development and production process</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What is Unique About Software Systems

- What is special about software products and why the emphasis on software quality?

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Software Products</th>
<th>Other Industrial Products</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complexity</td>
<td>Usually very complex; Large number of operational options</td>
<td>Much lower complexity; Lower number of options (thousands)</td>
</tr>
<tr>
<td>Visibility</td>
<td>Invisible: impossible to detects omissions by sight</td>
<td>Visible: easy to detect defects by sight</td>
</tr>
<tr>
<td>Nature of development and production process</td>
<td>Opportunity to detect defects arises only at the end</td>
<td>Detection happens in all phases: Planning, Development, Manufacturing</td>
</tr>
</tbody>
</table>
Software Quality: Basic Definitions - 1

- SwE testing literature and courses are filled with confusing/inconsistent use of terms
  - Error
    - a mistake made by a human (in a software development activity)
  - defect (or fault)
    - the result of introducing an error into a software artifact (SRS, SDS, code, etc.)
  - failure
    - a departure from the required behavior for a system

---

Error, Defect, Failure

![Diagram showing the relationship between human error, fault, and failure.](image)
When are Defects Introduced?

- The majority of defects are introduced in earlier phases.
  - Requirements are the top factor in a project’s success or failure.

<table>
<thead>
<tr>
<th>Phase</th>
<th>% of defects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>56</td>
</tr>
<tr>
<td>Design</td>
<td>27</td>
</tr>
<tr>
<td>Code</td>
<td>7</td>
</tr>
<tr>
<td>Other</td>
<td>10</td>
</tr>
</tbody>
</table>
Cost of fixing defects

- Relative cost of fixing defects
  - benchmark: cost of fixing defect in requirements phase = 1

<table>
<thead>
<tr>
<th>Phase found</th>
<th>Cost ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>1</td>
</tr>
<tr>
<td>Design</td>
<td>3 - 5</td>
</tr>
<tr>
<td>Coding</td>
<td>10</td>
</tr>
<tr>
<td>Unit / integration testing</td>
<td>15 - 40</td>
</tr>
<tr>
<td>System / acceptance testing</td>
<td>30 - 70</td>
</tr>
<tr>
<td>Production</td>
<td>40 - 1000</td>
</tr>
</tbody>
</table>

Program Behavior

- Specification (expected)
- Program (observed)
- Missing Functionality (sins of omission)
- Extra Functionality (sins of commission)
- "Correct" Portion
Correctness - 1

- Relative term: Program P is correct with respect to specification S
- Do specification and program meet customer expectations?

Correctness - 2

- Can we demonstrate correctness?

```java
int someMethod (int x) {
    x = x – 1;  //should be x = x + 1
    x = x / 30000;  //integer division. No remainder
    return x;
}
```

- If `someMethod` is implemented using 16-bit integers then how many test cases will we need?
  - Domain is [-32768 to 32767]
  - 65536 possibilities
Correctness - 3

```java
int someMethod (int x) {
    x = x - 1;  //should be x = x + 1
    x = x / 30000;  //integer division. No remainder
    return x;
}
```

<table>
<thead>
<tr>
<th>Input (x)</th>
<th>Expected Result</th>
<th>Actual Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>42</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-32768</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>32767</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Very few test cases will discover this defect. What are they? In teams (5 minutes)

Correctness - 4

```java
int someMethod (int x) {
    x = x - 1;  //should be x = x + 1
    x = x / 30000;  //integer division. No remainder
    return x;
}
```

<table>
<thead>
<tr>
<th>Input (x)</th>
<th>Expected Result</th>
<th>Actual Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>29999</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>30000</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>-29999</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>-30000</td>
<td>0</td>
<td>-1</td>
</tr>
</tbody>
</table>
Correctness - 5

- Almost always IMPOSSIBLE

Finally: Definition of Software Quality

- Conformance to requirements.
- Narrowest sense of software quality.
  - Lack of bugs.
  - Low defect rate (\# of defects/size unit)
  - High reliability (number of failures per \( n \) unit of time of operation).
    - Mean Time To Failure (MTTF): probability of failure-free operation in a specified time.
Software Quality: IEEE Definition

- The degree to which a system, component, or process meets specified requirements.

Software Quality: Pressman

- Conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software.
Importance of Software Quality

- Software is a major component of computer systems (about 80% of the cost) – used for:
  - Communication (e.g. phone system, email system).
  - Health monitoring.
  - Transportation (e.g. automobile, aeronautics).
  - Economic exchanges (e.g. e-commerce).
  - Entertainment.
  - etc.

- Software defects are extremely costly in term of:
  - money
  - reputation
  - loss of life

Notable consequences of software errors-1

- 1988 shooting down of Airbus 320 by the USS Vincennes - cryptic and misleading output displayed by tracking software.

- 1991 patriot missile failure - inaccurate calculation of time due to computer arithmetic errors.


- On June 3, 1980, the North American Aerospace Defence Command (NORAD) reported that the U.S. was under missile attack.

- First operational launch attempt of the space shuttle, whose real-time operating software consists of about 500,000 lines of code, failed - synchronization problem among its flight-control computers.
Notable consequences of software errors-2

- **Ariane 5 explosion June 4, 1996**
  - First flight of the European Ariane 5 launcher crashed about 40 seconds after takeoff.
  - Explosion was the result of a software error:
    - Uncaught exception due to floating-point error: conversion from a 64-bit integer to a 16-bit signed integer applied to a larger than expected number.
    - Module was re-used without proper testing from Ariane 4.
    - Error was not supposed to happen with Ariane 4.
    - No exception handler.

Notable consequences of software errors-3

- **Mars Climate Orbiter - September 23, 1999**
  - Mars Climate Orbiter, disappeared as it began to orbit Mars.
  - Cost about $US 125-million.
  - Failure due to error in a transfer of information between a team in Colorado and a team in California.
    - One team used imperial units (e.g., inches, feet and pounds) while the other used metric units for a key spacecraft operation.
Notable consequences of software errors-4

- Mars Polar Lander - December, 1999
  - Mars Polar Lander, disappeared during landing on Mars.
  - Failure most likely due to unexpected setting of a single data bit.
    - Defect not caught by testing.
    - Independent teams tested separate aspects.

So?

- Software systems are everywhere
- This is great
- But also scary
- Considering that we can rarely show correctness

- V&V is hard
  - Single technique won’t do it
  - More testing is not the answer
  - Knowing which technique and tools to use when is essential
  - Quality has to be built in and not done after the fact
Fun Activity of the Day

- Show us how good a tester you are
- Winner gets 1 point added to the end of the semester final grade

Individual Quiz (15 min)
Triangle program (simple version)

- Triangle problem is the most widely used example in software testing literature.
- The program accepts three integers, $a$, $b$, and $c$ as input. The three values are interpreted as representing the lengths of sides of a triangle. The program prints a message that states whether the triangle is scalene, isosceles, or equilateral.

- On a sheet of paper, write a set of test cases (i.e., specific sets of data) that you feel would adequately test this program.

<table>
<thead>
<tr>
<th>Test case #</th>
<th>$a$</th>
<th>$b$</th>
<th>$c$</th>
<th>Expected output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>isosceles</td>
</tr>
<tr>
<td>2</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Evaluation of set of test cases (one point for each “yes” answer)

61

1. Do you have a test case that represents a valid scalene triangle? (note that test cases such as 1,2,3 and 2,5,10 do not warrant a “yes” answer, because there does not exist a triangle having such sides.)
2. Do you have a test case that represents a valid equilateral triangle?
3. Do you have a test case that represents a valid isosceles triangle? (2,2,4 would not be counted)
4. Do you have at least three test cases that represent valid isosceles triangles such that you have tried all three permutations of two equal sides (e.g., 3,3,4; 3,4,3; and 4,3,3)?
5. Do you have a test case in which one side has a zero value?
6. Do you have a test case in which one side has a negative value?
7. Do you have a test case with three integers greater than zero such that the sum of two of the numbers is equal to the third? (That is, if the program said that 1,2,3 represents a scalene triangle, it would contain a bug.)
8. Do you have at least three test cases in category 7 such that you have tried all three permutations where the length of one side is equal to the sum of the lengths of the other two sides (e.g., 1,2,3; 1,3,2; and 3,1,2)?
9. Do you have a test case with three integers greater than zero such that the sum of two of the numbers is less than the third? (e.g., 1,2,4 or 12,15,30)
10. Do you have at least three test cases in category 9 such that you have tried all three permutations (e.g., 1,2,4; 1,4,2; and 4,1,2)?
11. Do you have a test case in which all sides are 0 (i.e., 0,0,0)?
12. Do you have at least one test case specifying non-integer values?
13. Do you have at least one test case specifying the wrong number of values (e.g., two rather than three, integers)?
14. For each test case, did you specify the expected output from the program in addition to the input values.

Semester Teams
Semester Teams

Next Week

- We will discuss appraisal quality assurance techniques (inspections and reviews)
- In class inspection of the Digital Home SRS document
- Reading Assignment:
  - DH Customer Need Statement
  - DH High Level Requirements Definition (HLRD)
  - DH Background Scenario
  - DH Team Biographical Sketches

- This will be your first graded work of the semester so make sure you come to class prepared.