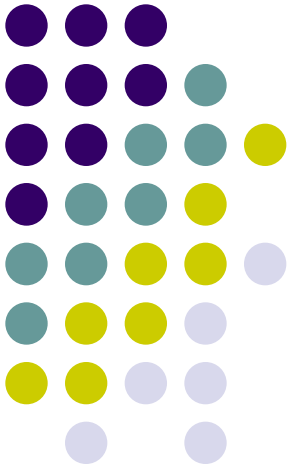
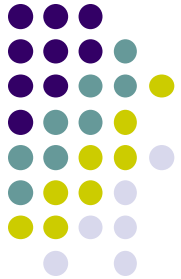
Kiểm thử phần mềm Software testing

Kiểm thử phần mềm
Software testing

Bộ môn Công nghệ Phần mềm
Khoa công nghệ thông tin,
Đại học Bách khoa Đà Nẵng
Email: ltmhanh@ud.edu.vn



Điều kiện tiên quyết

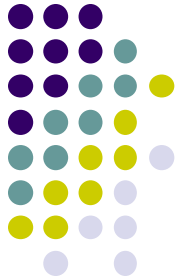


- Cần kinh nghiệm kiểm thử
- Một vài sự hiểu biết về các pha phát triển của dự án có thể có ích.
- Toán học có thể trợ giúp.

Tài liệu tham khảo

1. **Paul Jorgensen, Software Testing-A Craftsman's Approach, CRC Press, 1995.**
2. Spyos Xanthakis, Pascal Régnier, Constantin Karapoulios, Le test des logiciels, Hermes Science, 2000.
3. Hung Q. Nguyen and al., Testing application on the Web, John Wiley & Sons, 2004.
4. Ilene Burnstein, Practical Software Testing, Springer, 2003.
5. Glenford J. Myers, The art of software testing, Wiley, 2004.
6. Cem Kaner, Jack Falk, Hung Q. Nguyen, Testing Computer Software, 2nd Edition, John Wiley & Sons, 1999.
7. Boris Beizer, Software Testing Techniques, International Thomson Computer Press, Second Edition, 1990.
8. Neil Bitzenhofer, Software Testing and Verification, Course, MSSE, 2008.
9. Paul Ammann and Jeff Offutt, Introduction to Software Testing, Cambridge University Press, Cambridge, UK, ISBN 0-52188-038-1, 2008.
10. Mauro Pezzè, Michal Young, Software Testing and Analysis: Process, Principles, and Techniques, John Wiley & Sons.

Mô tả môn học



- Bao gồm cả lý thuyết và thực hành của kiểm thử phần mềm.
- Sinh viên sẽ tham gia vào các hoạt động kiểm thử::
  - Phân tích tài liệu yêu cầu để xác định các điều kiện kiểm thử.
  - Viết kế hoạch kiểm thử
  - Thiết kế, tạo và thực thi các test cases sử dụng các cách tiếp cận kiểm thử khác nhau
  - Ghi lại các lỗi (Record defects)
  - Viết báo cáo kiểm thử (Write a test report)

Nội dung

- Session 1: Introductory lecture
  - Introductions and expectations
  - Course overview
  - Contents

Nội dung

- Session 2: Introduction to Software Testing
  - Definitions, Principles, Axioms
  - Stages of testing
  - Perspectives on Software Testing
  - A little math

Nội dung

- Session 3: Requirements analysis
  - Software Development Life Cycle (SDLC)
  - Software Development stage
    - Requirements
  - Testing and requirements
    - Learn to think like a tester
  - Some examples
  - Writing test requirements

Nội dung

- Session 4

  Exercise 1: Examining requirements

Nội dung

- Session 5: Structural Testing
  - White box testing / Structural testing
  - Graph Theory
  - Control flow criteria
  - Data flow criteria
  - Graph Coverage for Source Code
  - Testing State Behavior
  - Syntax-based Testing

Nội dung

- Session 6: Static Testing
  - Reviews and the test process
  - Types of review
  - Static analysis

Nội dung

- Session 7: Functional Testing/Black-box
  - Introduction to functional testing
  - Functional testing techniques
    - Boundary Value testing
    - Equivalence Class testing
    - Special Value testing
    - Decision Tables

Nội dung

- Session 8: Test Documentation
  - Test Plan
    - The need for test plans
    - The structure of test plans
    - A Test Plan Template
    - A Test Plan example
    - Testing on a large project
  - Test Cases
    - Test Case Design
    - Test Case Examples

Nội dung

- Session 9
  Exercise 2: Writing a test plan and test cases

Nội dung

- Session 10: Integration & System Testing
  - Levels of Testing
  - Integration Testing
  - System Testing
  - Additional System Test Categories

Nội dung

- Session 11: Defect Reports/Test Reports
  - Handling Defects
  - Bug Tracking System
  - Test Reports
  - Examples

Nội dung

- Session 12: Object-oriented Testing
  - Why OO Testing?
  - Impact of OO on Testing
  - OO Testing Phases
  - Testing OO Systems
  - Specific OO Testing Techniques

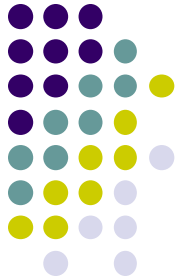
Nội dung

- Session 13: Test Automation and Tools
  - Test Automation
  - Test tools

Nội dung

- Session 14: Other topics
  - Metrics

Why test?

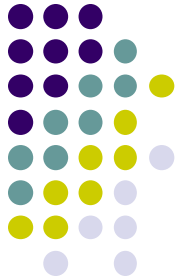


- List of 107 software failures that should have been caught by testing

  http://www.cs.tau.ac.il/~nachumd/verify/horror.html
- One vital consideration from Myers' book

  "The Art of Software Testing"
- Mars Climate Orbiter
- Mars Polar Lander

Software Errors



1. The Mars Climate Orbiter crashed in September 1999 because of a "silly mistake": wrong units in a program.
2. The 1988 shooting down of the Airbus 320 by the USS Vincennes was attributed to the cryptic and misleading output displayed by the tracking software.
3. Death resulted from inadequate testing of the London Ambulance Service software.
4. Several 1985-7 deaths of cancer patients were due to overdoses of radiation resulting from a race condition between concurrent tasks in the Therac-25 software.
5. Errors in medical software have caused deaths. Details in B.W. Boehm, "Software and its Impact: A Quantitative Assessment," *Datamation,* 19(5), 48-59(1973).
6. An Airbus A320 crashes at an air show.
7. A China Airlines Airbus Industrie A300 crashes on April 26, 1994 killing 264. Recommendations include software modifications.

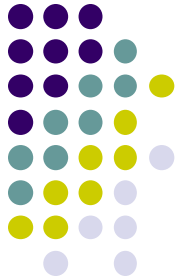
Software Errors

- **The Explosion of the Ariane 5**
  - On June 4, 1996 an unmanned Ariane 5 rocket launched by the European Space Agency exploded just forty seconds after its lift-off from Kourou, French Guiana. The rocket was on its first voyage, after a decade of development costing $7 billion. The destroyed rocket and its cargo were valued at $500 million. A board of inquiry investigated the causes of the explosion and in two weeks issued a report. It turned out that the cause of the failure was a software error in the inertial reference system. Specifically 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. The number was larger than 32,767, the largest integer storeable in a 16 bit signed integer, and thus the conversion failed.

Software Errors

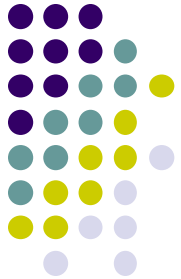
- **The Patriot Missile Failure**
  - **On February 25, 1991, during the Gulf War, an American Patriot Missile battery in Dharan, Saudi Arabia, failed to track and intercept an incoming Iraqi Scud missile. The Scud struck an American Army barracks, killing 28 soldiers and injuring around 100 other people. A report of the General Accounting office, GAO/IMTEC-92-26, entitled *Patriot Missile Defense: Software Problem Led to System Failure at Dhahran, Saudi Arabia* reported on the cause of the failure. It turns out that the cause was an inaccurate calculation of the time since boot due to computer arithmetic errors. Specifically, the time in tenths of second as measured by the system's internal clock was multiplied by 1/10 to produce the time in seconds. This calculation was performed using a 24 bit fixed point register. In particular, the value 1/10, which has a non-terminating binary expansion, was chopped at 24 bits after the radix point. The small chopping error, when multiplied by the large number giving the time in tenths of a second, led to a significant error. Indeed, the Patriot battery had been up around 100 hours, and an easy calculation shows that the resulting time error due to the magnified chopping error was about 0.34 seconds. A Scud travels at about 1,676 meters per second, and so travels more than half a kilometer in this time. This was far enough that the incoming Scud was outside the "range gate" that the Patriot tracked. Ironically, the fact that the bad time calculation had been improved in some parts of the code, but not all, contributed to the problem, since it meant that the inaccuracies did not cancel.**

Software Errors



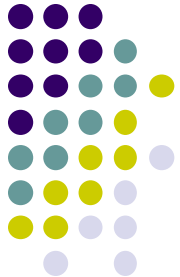
- **Y2K**
  - Spent some billions dollars

Chapter 1 of the textbook
A Perspective on Testing



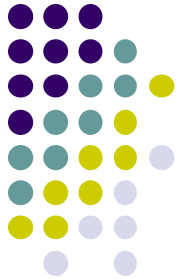
- Basic Definitions
  - Error – a mistake in design, coding, requirements, even testing
  - Fault – the representation of the error
  - Failure – what happens when the fault “executes”
  - Incident – the user-visible manifestation of the failure

A Perspective on Testing



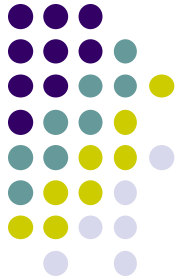
- More Definitions
  - Testing – the process of finding errors and of validating the program/system
  - Test Case – a test case has
    - Inputs
    - Steps
    - Outputs
    - Expected results
  - Process
    - Test plan,
    - Write test cases
    - Run the test cases
    - Evaluate results

A Perspective on Testing



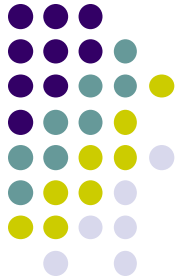
- Test Cases
  - Test Cases will be discussed in detail in Session 7 and throughout the course.
  - “Testing entails establishing the environment, providing the inputs (running the test case), observing outputs, and comparing to expected outputs.”
  - Test Cases are developed, reviewed, used, managed, and saved – and hopefully reused!

A Perspective on Testing



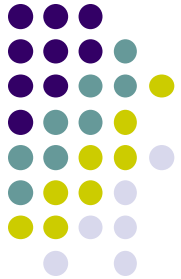
- Identifying Test Cases
  - Functional Testing
    - The program is a function that maps input values to output values
    - The only information used is the software specification
    - In our Venn diagram, the Functional Test Cases are a subset of S
    - Further elaborated on in Part II
    - Math background: Chapter 3
    - We will discuss in Session 6

A Perspective on Testing



- Identifying Test Cases
  - Structural Testing
    - Uses the information inside the “black box” – the actual implementation
    - In our Venn diagram, the Structural Test Cases are a subset of P.
    - Further elaborated on in Part III of the text
    - Math background: Chapter 4
    - We will discuss this in Session 4
    - Main method: Test coverage metrics

A Perspective on Testing



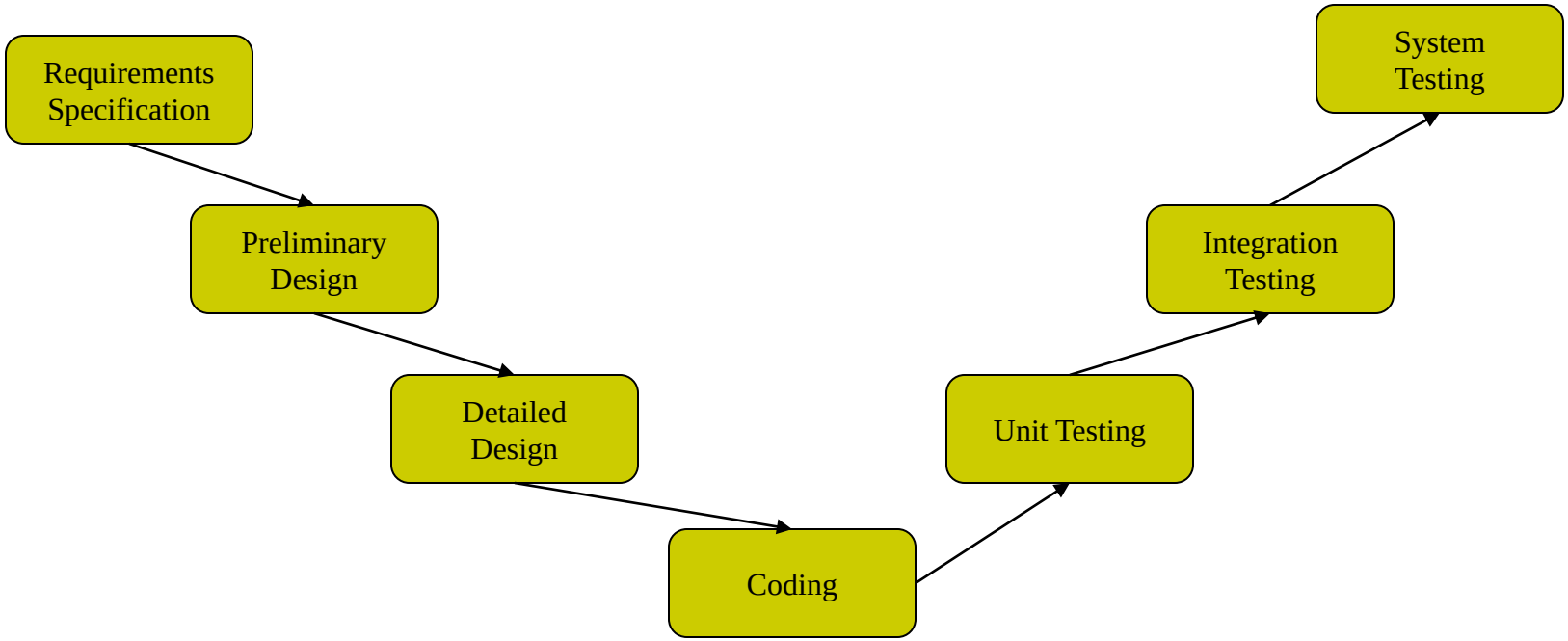
- Identifying Test Cases
  - Comparing the two (Functional vs Structural)
    - We will discuss this in Sessions 4 and 6

    "If all specified behaviors have not been implemented, structural test cases will never be able to recognize this.

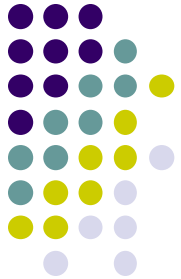
    Conversely, if the program implements behaviors that have not been specified, this will never be revealed by functional test cases."

A Perspective on Testing

- Levels of Testing
  - Again, this will be covered in detail in Session 2.



Testing a Program



A program that we want to test reads in 3 integer values – these 3 values are interpreted as the lengths of the sides of a triangle.

The program prints a message that states whether the triangle is

- Equilateral (all 3 sides equal)
- Isosceles (exactly 2 of the 3 sides are equal)
- Scalene (all 3 sides are of a different length)

On a sheet of paper, write specific sets of test data that you feel would adequately test this program.

You don't have to put your name on the paper.

You have 10 minutes maximum.