Software Quality
CS 563

Naveed Arshad
Assistant Professor
Lahore University of Management Sciences

CS 563 - Software Quality

- Instructor
  - Naveed Arshad
  - Room 405, Library Building
  - Email: naveedarshad@lums.edu.pk

- Lecture Timings
  - Mondays and Tuesdays
  - 10:15 – 11:30 AM

- Office Hours
  - Monday – Thursday
  - 11:30 AM – 1:15 PM

- Textbook:
  - "Software Testing: A Craftsman’s Approach" by Paul Jorgensen
  - “Testing Computer Software” by Cem Kaner (Only two chapters)
  - Papers and other resources from the web

- Teaching Assistant
  - Furqan Ahmad Minhas
  - Email: minhas@lums.edu.pk

- Website
  - Will be announced soon.
CS 563 - Software Quality

• Aims
  – Concepts, theory and practice of software quality through testing, inspection, and measurement of software systems

• Topics
  – Software Quality
  – Testing Methods and Roles
  – Inspections, Reviews and Refactoring
  – Software Metrics

CS 563 - Software Quality

• Grading Instruments
  – 30% Assignments
  – 20% Quizzes
  – 20% Midterm (1)
  – 30% Final (1)
What is Quality?

- Quality
  - Not a single idea - many aspects
- Popular View
  - In everyday life, usually thought of as intangible, can be felt or judged, but not weighed or measured
  - “I know it when I see it” - implies that it cannot be controlled, managed, or quantified
  - Often influenced by perception rather than fact - e.g., a Toyota Corolla may be spoken of as a “quality” car, in spite of the fact that its reliability and repair record is no better than a Honda Civic.

What is Software Quality?

- Quality
  - Not a single idea - many aspects
- Popular View
  - In everyday life, usually thought of as intangible, can be felt or judged, but not weighed or measured
  - “I know it when I see it” - implies that it cannot be controlled, managed, or quantified
  - Often influenced by perception rather than fact - e.g., a Toyota Corolla may be spoken of as a “quality” car, in spite of the fact that its reliability and repair record is no better than a Honda Civic.
What is Software Quality?

• Professional View
  – Quality is not just a marketing and perception issue, it is a moral and legal requirement - we have a professional responsibility associated with the software we create
  – Professionals must be able to demonstrate, and to have confidence, that they are using “best practices”
  – In practical terms, therefore, product quality must be measurable in some way
  – Product quality is spoken of in terms of
    • conformance to requirements - including timeliness, cost
    • fitness for use - does it actually do the job?
    • freedom from errors and failures - is it reliable and robust?
    • customer satisfaction - are users happy with it?

What is Software Quality?

• Software quality is normally spoken of in terms of several different dimensions often called quality parameters. These can be split (roughly) into two groups
  – Technical Quality Parameters
    • correctness, reliability, capability, performance, maintainability
  – User Quality Parameters
    • usability, installability, documentation, availability
Technical Quality Parameters

- **Correctness** - lack of bugs and defects measured in terms of defect rate (# bugs per line of code)
- **Reliability** - does not fail or crash often measured in terms of failure rate (#failures per hour)
- **Capability** - does all that is required measured in terms of requirements coverage (% of required operations implemented)
- **Maintainability** - is easy to change and adapt to new requirements measured in terms of change logs (time and effort required to add a new feature) and impact analysis (#lines affected by a new feature)
- **Performance** - is fast and small enough measured in terms of speed and space usage (seconds of CPU time, MB of memory, etc.)

Technical Quality Parameters

- **Usability** - is sufficiently convenient for the intended users measured in terms of user satisfaction (% of users happy with interface and ease of use)
- **Installability** - is convenient and fast to install – measured in terms of user satisfaction (#install problems reported per installation)
- **Documentation** - is well documented - measured in terms of user satisfaction (% of users happy with documentation)
- **Availability** - is easy to access and available when needed - measured in terms of user satisfaction (% of users reporting access problems)
Software Quality Principle

• Principle 1: Know What You Are Doing
  – In the context of software quality, this means continuously understanding what it is you are building, how you are building it and what it currently does
  – This requires organization, including having a management structure, reporting policies, regular meetings and reviews, frequent test runs, and so on
  – We normally address this by following a software process with regular milestones, planning, scheduling, reporting and tracking procedures

Software Quality Principle

• Principle 2: Know What You Should be Doing
  – In the context of software quality, this means having explicit requirements and specifications
  – These must be continuously updated and tracked as part of the software development and evolution cycle
  – We normally address this by requirements and use-case analysis, explicit acceptance tests with expected results, explicit prototypes, frequent user feedback
  – Particular procedures and methods for this are usually part of our software process
Software Quality Principle

- Principle 3: Know How to Measure the Difference
  - In the context of software quality, this means having explicit measures comparing what we are doing to what we should be doing
  - Achieved using four complementary methods
    - **Formal Methods**
      - consists of using mathematical models or methods to verify mathematically specified properties
    - **Testing**
      - consists of creating explicit inputs or environments to exercise the software, and measuring its success
    - **Inspection**
      - consists of regular human reviews of requirements, design, architecture, schedules and code
    - **Metrics**
      - consists of instrumenting code or execution to measure a known set of simple properties related to quality

Formal Methods

- Formal methods include
  - formal verification - proofs of correctness
  - abstract interpretation - simulated execution in a different semantic domain, e.g., data kind rather than value
  - state modeling - simulated execution using a mathematical model to keep track of state transitions
  - other mathematical methods
- In practice, formal methods are used directly in software quality assurance in only a small (but important) fraction of systems
- Primarily safety critical systems such as onboard flight control systems, nuclear reactor control systems, embedded systems such as automobile braking systems and medical equipment, and so on
- Use of formal methods requires mathematically sophisticated programmers, and is necessarily a slow and careful process, and very expensive
Testing

• Testing
  Testing includes a wide range of methods based on the idea of running the software through a set of example inputs or situations and validating the results

• Types of Testing
  1. Acceptance testing,
  2. Functionality Testing
  3. Interface testing
  4. Regression testing
  5. Unit Testing
  6. GUI Testing
  7. Structural Testing
  8. And many more

Inspection and Metrics

• Inspection
  – Inspection includes methods based on a human review of the software artifacts
  – Includes methods based on requirements reviews, design reviews, scheduling and planning reviews, code walkthroughs, and so on
  – Helps discover potential problems before they arise in practice

• Metrics
  – Software metrics includes methods based on using tools to count the use of features or structures in the code or other software artifacts, and compare them to standards
  – Includes methods based on code size (number of source lines), code complexity (number of parameters, decisions, function points modules or methods), structural complexity (number or depth of calls or transactions), design complexity, and so on
  – Helps expose anomalous or undesirable properties that may reduce reliability and maintainability
Lessons Learned from Software Development

Mars Orbiter

Mars, July 4, 1997
Lost contact due to real-time priority inversion bug
NASA Mars Climate Orbiter

WASHINGTON (AP) -- For nine months, the Mars Climate Orbiter was speeding through space and speaking to NASA in metric. But the engineers on the ground were replying in non-metric English. It was a mathematical mismatch that was not caught until after the $125-million spacecraft, a key part of NASA's Mars exploration program, was sent crashing too low and too fast into the Martian atmosphere. The craft has not been heard from since.

Denver Airport’s Baggage Handling System

• (Scientific America) -- Scheduled for takeoff by last Halloween (1993), the airport's grand opening was postponed until December to allow BAE Automated Systems time to flush the gremlins out of its $193-million system. December yielded to March. March slipped to May. In June the airport's planners, their bond rating demoted to junk and their budget hemorrhaging red ink at the rate of $1.1 million a day in interest and operating costs, conceded that they could not predict when the baggage system would stabilize enough for the airport to open.
What is This Thing?

A380 vs. B747

Boeing 747
- Seating: 416
- Internal cabin width: 6.1m

Airbus A380
- Seating: 555 (max 565)
- Internal cabin width: 6.56m

Source: Airbus/Boeing
A380 Delays

• A380 is being developed in four countries
  – Spain
  – Germany
  – France
  – UK
• The whole program delayed by 2 years because of the configuration problems of the wiring software.
• According to the latest report on 3 Oct 2006, cost of the consequent two-year delay to Airbus is estimated to be **4.8 billion euros**.

Software Hall of Shame

<table>
<thead>
<tr>
<th>Year</th>
<th>Company/Program</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1997</td>
<td>U.S. Internal Revenue Service</td>
<td>Tax modernization effort canceled after $4 billion is spent.</td>
</tr>
<tr>
<td>1997</td>
<td>State of Washington</td>
<td>Department of Motor Vehicle (DMV) system canceled after $40 million is spent.</td>
</tr>
<tr>
<td>1997</td>
<td>Oxford Health Plans Inc.</td>
<td>Billing and claims system problems contribute to quarterly loss; stock plummeted, leading to $3.4 billion loss in corporate value.</td>
</tr>
<tr>
<td>1996</td>
<td>Arianespace [France]</td>
<td>Software specification and design errors cause $250 million Ariane 5 rocket to explode.</td>
</tr>
<tr>
<td>1996</td>
<td>FinMynra Drug Co.</td>
<td>$40 million ERP system abandoned after deployment, forcing company into bankruptcy.</td>
</tr>
<tr>
<td>1995</td>
<td>Toronto Stock Exchanges [Canada]</td>
<td>Electronic trading system abandoned after $25.8 million** in spent.</td>
</tr>
<tr>
<td>1994</td>
<td>U.S. Federal Aviation Administration</td>
<td>Advanced Automation System cancelled after $2.6 billion is spent.</td>
</tr>
<tr>
<td>1994</td>
<td>State of California</td>
<td>DMV system canceled after $44 million is spent.</td>
</tr>
<tr>
<td>1994</td>
<td>Chemical Bank</td>
<td>Software error causes a total of $15 million to be deducted from 100 000 customer accounts.</td>
</tr>
<tr>
<td>1993</td>
<td>Allstate-Insurance Co.</td>
<td>Office automation system abandoned after deployment, costing $100 million.</td>
</tr>
<tr>
<td>1993</td>
<td>Greyhound Lines Inc.</td>
<td>Bus reservation system crashes repeatedly upon introduction, contributing to revenue loss of $61 million.</td>
</tr>
<tr>
<td>1992</td>
<td>Budget Rent-A-Car, Hilton Hotels, Marriott</td>
<td>Travel reservation system canceled after $165 million is spent.</td>
</tr>
</tbody>
</table>
Software Hall of Shame

<table>
<thead>
<tr>
<th>YEAR</th>
<th>COMPANY</th>
<th>OUTCOME (COSTS IN US $)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2005</td>
<td>Hudson Bay Co. (Canada)</td>
<td>Problems with inventory system contribute to $33.3 million loss.</td>
</tr>
<tr>
<td>2004-05</td>
<td>UK Inland Revenue</td>
<td>Software errors contribute to $3.45 billion tax-credit overpayment.</td>
</tr>
<tr>
<td>2004</td>
<td>Avon Europe PLC (UK)</td>
<td>Enterprise resource planning (ERP) system canceled after $64.5 million is spent.</td>
</tr>
<tr>
<td>2004</td>
<td>Ford Motor Co.</td>
<td>Purchasing system abandoned after deployment costing approximately $400 million.</td>
</tr>
<tr>
<td>2004</td>
<td>J Sainsbury PLC (UK)</td>
<td>Supply chain management system abandoned after deployment costing $527 million.</td>
</tr>
<tr>
<td>2004</td>
<td>Hewlett-Packard Co.</td>
<td>Problems with ERP system contribute to $60 million loss.</td>
</tr>
<tr>
<td>2003-04</td>
<td>AT&amp;T Wireless</td>
<td>Customer relations management (CRM) upgrade problems lead to revenue loss of $60 million.</td>
</tr>
<tr>
<td>2002</td>
<td>McDonald's Corp.</td>
<td>The renovation information-purchasing system canceled after $170 million is spent.</td>
</tr>
<tr>
<td>2002</td>
<td>Sydney Water Corp. [Australia]</td>
<td>Billing system canceled after $32.2 million is spent.</td>
</tr>
<tr>
<td>2002</td>
<td>CIGNA Corp.</td>
<td>Problems with CRM system contribute to $445 million loss.</td>
</tr>
<tr>
<td>2001</td>
<td>Nike Inc.</td>
<td>Problems with supply-chain management system contribute to $90 million loss.</td>
</tr>
<tr>
<td>2001</td>
<td>Knobloch Corp.</td>
<td>Supply chain management system canceled after $90 million is spent.</td>
</tr>
<tr>
<td>2000</td>
<td>Washington, D.C.</td>
<td>City payroll system abandoned after deployment costing $95 million.</td>
</tr>
<tr>
<td>1999</td>
<td>United Way</td>
<td>Administrative processing system canceled after $2 million is spent.</td>
</tr>
<tr>
<td>1999</td>
<td>State of Mississippi</td>
<td>Tax system canceled after $1.2 million is spent; state receives $1.25 million damages.</td>
</tr>
<tr>
<td>1999</td>
<td>Hershey Foods Corp.</td>
<td>Problems with ERP system contribute to $95 million loss.</td>
</tr>
<tr>
<td>1998</td>
<td>Snap-on Inc.</td>
<td>Problems with order-entry system contribute to revenue loss of $60 million.</td>
</tr>
</tbody>
</table>

Cost of Software Bugs

WASHINGTON (COMPUTERWORLD) - Software bugs are costing the U.S. economy an estimated **$59.5 billion each year**, with more than half of the cost borne by end users and the remainder by developers and vendors, according to a new federal study.

Improvements in testing could reduce this cost by about a third, or $22.5 billion, but it won't eliminate all software errors, the study said. Of the total $59.5 billion cost, users incurred 64% of the cost and developers 36%. 

26
Next Class

• “Testing Computer Software” by Kaner
• Read Chapters 3, 4
  – Chapter 3: “Test Types and their place in the software development process”
  – Chapter 4: “Software Errors”