1. Introduction to Software Quality

SE 3730 / CS 5730 Lecture Notes
Yan Shi
UW-Platteville
Outline

- What is software quality?
  - Quality
  - Software
  - Software quality
  - Software errors, faults and failures
  - Software quality factors

- What is software quality assurance?
  - Definition
  - Objectives
  - Components of the software quality assurance system
Why do we care?

- Software QA engineering salaries (data from glassdoor.com)
  - Apple: $81k ~ 134k
  - eBay: $86k ~ 121k
  - National average: $78k

- Poor quality software can be costly

<table>
<thead>
<tr>
<th></th>
<th>The Cost of Inadequate Software Testing Infrastructure (Billions)</th>
<th>Potential Cost Reduction from Feasible Infrastructure Improvements (Billions)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Developers</td>
<td>$21.2</td>
<td>$10.6</td>
</tr>
<tr>
<td>Software Users</td>
<td>$38.3</td>
<td>$11.7</td>
</tr>
<tr>
<td>Total US Costs</td>
<td>$59.5</td>
<td>$22.2</td>
</tr>
</tbody>
</table>
History’s worst software bugs

- July 28, 1962 -- Mariner I space probe
  - Translate from paper to computer code
- 1982 -- Soviet gas pipeline
  - CIA’s bug passed Soviet inspection
- 1985-1987 -- Therac-25 medical accelerator
  - Race condition
- 1988 -- Buffer overflow in Berkeley Unix finger daemon
  - No input limit for gets()
- 1988-1996 -- Kerberos Random Number Generator
  - No true random seed
- January 15, 1990 -- AT&T Network Outage
  - Crash and reboot message crashes neighbor switches
- 1993 -- Intel Pentium floating point divide
- 1995/1996 -- The Ping of Death
- June 4, 1996 -- Ariane 5 Flight 501
  - Converting 64-bit float to 16-bit signed integer...
- November 2000 -- National Cancer Institute, Panama City
  - Four shielding blocks, or five?
The 4 Worst Software Bugs of 2015

Year 2038...

- **Are you prepared for the Unix 2038 problem?**

- UNIX operating system handle dates by counting how many seconds a date is since 01/01/1970.

- This number is stored on these computers as a “signed 32-bit integer”.

- 2,147,483,647 seconds after 01/01/1970 only takes us up to the January 19, 2038!!!
SOFTWARE QUALITY
What is software?

IEEE definition:

“Software is computer programs, procedures and possibly associated documentation and data pertaining to the operation of a computer system.”

Four components of software:

- Computer programs (the code)
- Procedures
- Documentation
- Data necessary for operating the software system
What is quality?

- "Quality is when your customers come back, not your products" – Margaret Thatcher

- Conformance to requirements
- **Fit** for customer’s use

- Low **defect rate** (# of defects/unit)
- High **reliability**
What is software quality?

- Two alternative IEEE definition:
  Software quality is:

  1. The degree to which a system, component, or process meets specified requirements. --Philip Crosby
     
     What if there is a error in the requirement?

  2. The degree to which a system, component, or process meets customer or user needs or expectations.
     
     --Joseph Juran
     What if we have irresponsible customers throwing last minute requirement changes?

Do you see any issues with these definitions?
What is Software Quality?

- Pressman’s definition (2000)

**Software quality is** conformance to

1. explicitly stated functional and performance requirements,
2. explicitly documented development standards, and
3. implicit characteristics that are expected of all professionally developed software.
Software errors, faults and failures

From standards by IEEE Computer Society

- **Error** – human interaction which produce an incorrect result.
- **Fault** – representation of an error.
- **Failure** – occurs when a fault executes.
Software errors, faults and failures

Software development process

software error
software fault
software failure
Causes of Software Errors

1. Faulty requirements definition by the client
2. Client-developer communication failures
3. Deliberate deviations from software requirements
4. Logical design errors
5. User interface and procedure errors
6. Coding errors
7. Non-compliance with documentation and coding instructions
8. Documentation errors
Consider the differences between a Ford Explorer and Microsoft Word:

- Which is more complicated?
- What are the differences in their natures?
- What is the production process for each of them?
- When is it possible to detect defects during the production process?
The Software Quality Challenge

- The uniqueness of software products compared to industrial products
  - High complexity: *millions of operation possibilities.*
  - Invisibility of the product
  - Limited opportunities to detect defects during the product development and production process
    - product development ➔
      - production planning ➔
      - manufacturing
    - only opportunity is Product development
Development process relation to defects

- Majority of defects are introduced in earlier phases

<table>
<thead>
<tr>
<th>Phase</th>
<th>Percentage of defects</th>
<th>Effort to fix defects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>56</td>
<td>82</td>
</tr>
<tr>
<td>Design</td>
<td>27</td>
<td>13</td>
</tr>
<tr>
<td>Code</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>Others</td>
<td>10</td>
<td>4</td>
</tr>
</tbody>
</table>

- Relative cost of fixing defects

<table>
<thead>
<tr>
<th>Phase in which found</th>
<th>Cost Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>1</td>
</tr>
<tr>
<td>Design</td>
<td>3 – 6</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>

Requirement document is one of the most important elements for achieving software quality!
Learn@UW/D2L - Recurring Service Issue

Posted Sep 7, 2012 4:58 PM

Faculty, Staff and Students:

On behalf of our Learn@UW Executive Committee, I/we want you to know that we are taking steps now to improve the performance of Desire2Learn (D2L) systems at all UW campuses.

We are aware that many of you have experienced slow or delayed responses in D2L over the past few days. After D2L was upgraded to version 10 this summer, we noticed some occasional performance issues. Immediately, Learn@UW technical staff (the entity at UW-Madison’s Division of Information Technology that manages the D2L System) and D2L staff began working on solutions to improve performance. Later, Microsoft experts were engaged to help solve the problem. Numerous patches to the system have been applied. However, these efforts have yielded only temporary success, and the root cause of these performance problems has not yet been identified. This is not uncommon in a complex environment like ours, but it is still very, very frustrating.

We will not tolerate this level of performance. While technical experts continue around the clock to resolve the issues, the Learn@UW Executive Committee is exploring alternatives. We are consulting with top D2L executives to determine what assistance D2L can provide beyond finding the problem and what viable alternatives exist.

We are confident that we are receiving the best attention possible from both vendors. Top leadership at UW System, Learn@UW Utility and Desire2Learn are engaged in this process.

As we work through this we ask for your continued patience, and we will continue to keep you posted on our efforts to restore the level of performance that faculty, staff, and students deserve.
“Good” requirement document

- The D2L problem:
  - The software fulfills the basic requirements.
  - The software suffers from poor performance in important areas (maintenance, reliability...)

**WHY?**

- Lack of predefined requirements to cover these important aspects!

- Classify quality requirements into quality factors.
  - McCall’s factor model (1977)
# Software quality factors

## Software quality factors

<table>
<thead>
<tr>
<th>Product operation factors</th>
<th>Product revision factors</th>
<th>Product transition factors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correctness</td>
<td>Maintainability</td>
<td>Portability</td>
</tr>
<tr>
<td>Reliability</td>
<td>Flexibility</td>
<td>Reusability</td>
</tr>
<tr>
<td>Efficiency</td>
<td>Testability</td>
<td>Interoperability</td>
</tr>
<tr>
<td>Integrity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usability</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Software quality factors

Quality software

- Product transition
  - Portability
  - Reusability
  - Interoperability

- Product operation
  - Correctness
  - Reliability
  - Efficiency

- Product revision
  - Maintainability
  - Flexibility
  - Testability
  - Usability
  - Integrity
Product operation factors

- Correctness: output specifications
  - output mission, accuracy, completeness of required output
  - up-to-date, availability of the information
  - standards for coding and documenting the system

- Reliability
  - minimize failure rate
  - failure recovery

- Efficiency
  - resources needed to perform software function (processing, data storage, communication, battery consumption, etc.)

- Integrity
  - software system security, access rights

- Usability
  - ability to learn, perform required task
Product revision factors

- Deal with the complete range of software maintenance activities:
  - corrective maintenance,
  - adaptive maintenance and
  - perfective maintenance.

- Maintainability:
  - effort to identify and fix software failures
    (modularity, documentation, etc.)

- Flexibility: adaptive and perfective
  - degree of adaptability
    (to new customers, tasks, etc.)

- Testability
  - support for testing
    (e.g. log files, automatic diagnostics, etc.)
Product transition factors

- **Portability**
  - adaptation to other environments
    (hardware, software)

- **Reusability**
  - use of software components for other projects

- **Interoperability**
  - ability to interface with other components/systems
Examples

- “The average person-time required to fix a category 2 defect (including regression testing and documentation update) shall not exceed one person week.”
- “The average time needed to port the component from Windows 8 to Windows 10 shall not exceed two person weeks.”
- “The component shall be delivered with associated test software and test data that is sufficient to enable regression testing of all of its associated functional requirements.”
- “All reoccurring batch processes shall begin execution within 10 minutes of their regularly scheduled start times.”
- “The Global Personal Marketplace (GPM) system shall protect the communicated data it receives from unsophisticated attack involving unauthorized addition, modification, deletion, or replay during execution of the Buyer use cases as indicated in table 2.”
Examples (continued)

- “A minimum of 40% of the architecture of the application shall be reused from one of the standard reference architectures stored in the reuse repository.”

- “The average person-time required to extend the application to enable buyers to obtain insurance shall not exceed 60 person days.”

- “At least 90% of a statistically valid sample of users shall rate the overall user experience as being either very positive or positive on the following scale (very positive, positive, neutral, negative, very negative).”

- “The application shall properly interoperate with existing barcode readers.”

- “The component shall not fail more than an average of 3 times per year.”

What is Software Quality Assurance (SQA)?

- The IEEE definition

Software quality assurance is:

1. A planned and systematic pattern of all actions necessary to provide adequate confidence that an item or product conforms to established technical requirements.

2. A set of activities designed to evaluate the process by which the products are developed or manufactured. Contrast with quality control.
What is SQA?

- Expanded definition

Software quality assurance is:

- A systematic, planned set of actions necessary to provide adequate confidence that the software development process or the maintenance process of a software system product conforms to established functional technical requirements as well as with the managerial requirements of keeping the schedule and operating within the budgetary confines.
Components of quality assurance

- Pre-project
- Project life cycle activities assessment
- Infrastructure error prevention and improvement
- Software quality management
- Standardization, certification and SQA system assessment
- Organizing for SQA - the human components
Pre-project components

- Contract review
  - Agreements on functional specifications, budget and schedule.
  - Evaluate the capacity of professional staff and customers
  - Evaluate risks

- Development and quality plans (Integrated)
  Development plan
  - Schedules
  - Required manpower and hardware resources
  - Risk evaluations
  - Organizational issues
  - Project methodology, development tools
  - Software reuse plan

  Software quality assurance plan (SQAP)
  - Quality goals
  - Criteria for starting and ending each project stage
  - SQA activities.
Project life cycle component

- Reviews
  - Formal design reviews (DR)
  - Peer reviews: inspections and walkthroughs
- Expert opinions
- Software testing
- Software maintenance
  - Corrective maintenance
  - Adaptive maintenance
  - Functionality improvement maintenance
  Similar SQA components as for the development process
- Assurance of the quality of the external participant’s work
Infrastructure components for error prevention and improvement

Goal: prevent software faults and improve productivity

- Procedures and work instructions ➞ past experience
  - Procedure: generally applicable to the entire organization
  - Work instruction: detailed directions for unique teams
- Templates and checklists
- Staff training, retraining and certification
- Preventive and corrective actions
  - Systematic study of past failures and successes
- Configuration management
- Documentation control
Management SQA components

- Project progress control
  - Focus on resource usage, schedules, risks, budget
- Software quality metrics
- Software quality costs
  - Cost of control
  - Cost of failure
SQA standards, certification and assessment components

- Quality management standards (what)
  - SEI CMM assessment standards
  - ISO 9001:2008 - Quality management systems
  - ISO 9001-3 - Quality Management and Quality Assurance Standards - Part 3: Guidelines for the application of ISO 9001 to the development, supply and maintenance of software

- Project process standards (how)
  - IEEE 1012 standard for System and Software Verification and Validation
  - ISO/IEC 12207 standard for Systems and software engineering -- Software life cycle processes
Quality control:
- A set of activities designed to evaluate the quality of a developed or manufactured product.
- Main objective: withhold any product that does not qualify.
- Completed before the product is shipped to the client.

Quality assurance:
- Main objective: minimize the cost of guaranteeing quality.
- Prevent the cause of errors; detect and correct them early in the development process.
- performed throughout the software life cycle.
Verification vs. Validation

- **Verification**
  - Main purpose is to detect defects in the artifacts of the system under development.

- **Validation**
  - Main purpose is to show that the system under development meets user needs, requirements, and expectations.
Verification vs. Validation

- Reviews
- Walkthrough
- Unit test
- Integration test
- System test
- Acceptance test

- Verification
- Validation
Quality Cost vs. Cost of Quality

- Includes all costs of quality-related activities.

Quality costs =
  + Prevention costs
  + Detection and Appraisal costs
  + Failure costs
    - Internal failure costs
    - External failure costs

Cost of Quality (COQ) – refers to the cost of correcting defects once found.