Teaching Software Quality Assurance in an Undergraduate Software Engineering Program
Claude Y Laporte, Alain April, Khaled Bencherif

Presented by Claude Y Laporte
Professor
Department of Software and IT Engineering
École de technologie supérieure, Québec, Canada.

Fifth International Conference on
INDUSTRIAL AUTOMATION
June 11-13th, 2007

Content

• Introduction
• Business Rationale for Software Quality Assurance
• Cost of Poor Software Quality
• Components of Software Quality Assurance Course
• Conclusion
École de technologie supérieure

Over 4500 students, 125 professors, 25 general senior lecturers and 200 lecturers.

2200 paid industrial internships in over 900 companies (2004).

Undergraduate Programs
- Software Engineering
- IT Engineering
- Construction Engineering
- Production Engineering
- Electrical Engineering
- Mechanical Engineering
- Logistics and Operations Engineering

- Graduate Programs
  - Software Engineering
  - Information Technology
  - Other programs

www.etsmtl.ca

Undergraduate Software Engineering Program

<table>
<thead>
<tr>
<th>Software Requirements</th>
<th>Introduction to Databases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Architecture</td>
<td>Introduction to Parallel Processing</td>
</tr>
<tr>
<td>Software Design</td>
<td>Compilation Techniques</td>
</tr>
<tr>
<td>Advanced Object-Oriented Programming</td>
<td>High Performance Databases</td>
</tr>
<tr>
<td>Formal and Semi-Formal Languages</td>
<td>Principles of Operating Systems and Systems Programming</td>
</tr>
<tr>
<td>Data Structures and Algorithms</td>
<td>Distributed Object-Oriented Architecture</td>
</tr>
<tr>
<td>Software Quality assurance</td>
<td>Introduction to Distributed Systems</td>
</tr>
<tr>
<td>Quality Control and Metrics</td>
<td>Telecommunication Networks</td>
</tr>
<tr>
<td>User Interface Analysis and Design</td>
<td>Algorithms Analysis</td>
</tr>
<tr>
<td>Systems Security</td>
<td>Interactive Multimodal Systems</td>
</tr>
<tr>
<td>Analysis and Design of Telecommunications Software</td>
<td>Design of Real-Time Computer Systems</td>
</tr>
<tr>
<td></td>
<td>Capstone Project</td>
</tr>
</tbody>
</table>
Cost of Non Conformance and Quality of Product

• Quiz
  – For your projects, what is the Cost on Non Conformance
    • As a percentage (%) of the Project Cost?

  – What is the quality of your product (Defects/Unit of size)?
    • e.g. Number of Defects /Thousand Line of Code (KLOC)

Cost of Quality

Percentage of Rework on Projects

<table>
<thead>
<tr>
<th>Company</th>
<th>Percentage</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRW</td>
<td>30%</td>
<td>(Boehm, 1987)</td>
</tr>
<tr>
<td>NASA-SEL</td>
<td>40%</td>
<td>(McGarry, 1987)</td>
</tr>
<tr>
<td>Hewlett-Packard</td>
<td>33%</td>
<td>(Duncker, 1992)</td>
</tr>
<tr>
<td>Raytheon</td>
<td>41%</td>
<td>(Dion, 1993)</td>
</tr>
</tbody>
</table>

Note: Rework = Waste or Scrap
Business Rationale for Implementing Software Quality Assurance (SQA)

• What Do Management and Organisation Want?
  • Predictable Content and Quality
  • Predictable Schedule
  • Predictable Cost

• Benefits of SQA to Management and Organisation
  – Provide with objective insight into processes and work products (adapted from CMMI)
  – Provide proven industrial processes and practices to managers and engineers
  – Provide real-time hard data to help decision making
    • Completion of products - 90% completion syndrome
    • Measure of quality of products
  – Provide information of difficulties and potential improvements to software processes and products.

Typical Software Defect Profile

KLOC = Thousand Lines of Code

Wheeler, D., Bryckynski, B., Meeson, R, Software Inspection – An Industry Best Practice, Institute of Electrical and Electronics Engineers (IEEE), 1996, p 10
Typical Software Defect Profile

- Requirements: 5 (20)
- Design: 10 (40)
- Code: 15 (100)
- Unit Test: 7 (50)
- Integration Test: 3 (20)
- System Test: 1 (10)

Reduced Rework Costs

KLOC = Thousand Lines of Code

Wheeler, D., Brykcynski, B., Meeson, R, Software Inspection – An Industry Best Practice, Institute of Electrical and Electronics Engineers (IEEE), 1996, p 10

Guide to the Software Engineering Body of Knowledge (SWEBOK)

- Objectives
  - Characterize the content of the software engineering discipline,
  - Promote a consistent view of software engineering worldwide,
  - Set the boundary of software engineering with respect to other disciplines,
  - Provide a foundation for curriculum development and individual licensing material

www.swebok.org
Software Engineering Knowledge Areas

1. Software Requirements
2. Software Design
3. Software Construction
4. Software Quality (static techniques)
5. Software Testing
6. Software Maintenance
7. Software Configuration Management
10. Software Engineering Process

Related Disciplines

- Computer Engineering
- Computer Science
- Mathematics
- Project Management
- Management
- Quality Management
- Software Ergonomics
- Systems Engineering

ISO/IEC Certification of Software Engineering Professionals

- To respond to the need for portability of software engineering professional certifications,
- To facilitate the exchange of professionals between different countries,
- To provide the processes needed to establish, administer, and maintain a certification scheme,
- Certification body will administer:
  - The certification activity, including all procedures and activities intended to demonstrate the qualifications of software engineering professionals.
- SWEBOK will serve as a reference model for software engineering professional certifications.
Software Engineering Standards and SQA

• **Importance of Standards in Software Engineering**
  – No law of physics in software engineering!
  – Software Engineering Standards can be used in Court.

• **Standards covered in class and laboratory**
  – **ISO/IEC**
    • ISO/IEC 9126-Software Engineering-Product Quality.
    • ISO/IEC 90003-Software Engineering: Guidelines for the application of ISO 9001 to computer software
  – **IEEE**
    • IEEE 730 - IEEE Standard for Software Quality Assurance Plans
    • IEEE 828 - IEEE Standard for Software Configuration Management Plans
    • IEEE 1028- IEEE Standard for Software Reviews *
    • IEEE 1012 - Software Verification and Validation
    • IEEE 12207 - Software Lifecycle Processes.
  – **De Facto**
    • Capability Maturity Model Integration (CMMI), Software Engineering Institute (SEI).
Types of Reviews

- IEEE 1028 - Standard for Software Reviews

SQA Laboratory Topics

- **Code of Ethics** - Robot killer Case Study
- **Team Project**
  - Plan, design and program new features, manage configuration, defects and changes, perform reviews and evaluate software quality.
  - Draft a quality assurance plan (IEEE-730), a project plan and a software requirement specifications document (SRS).
  - Carry out a review (walk-through) (IEEE 1028)
  - Perform Software Configuration Management and Traceability (IEEE 1012).
    - Write a CM procedure
    - Use Open Source tool (CVS)
    - Update effort estimation
    - Carry out a review (walk-through)
  - Design, program and test.
    - Design and program additional features in existing software
    - Update effort estimation
    - Carry out a review (walk-through)
  - Perform problem/change management and defect management
    - Log anomalies using Open Source tool – Issue tracking tool (Bugzilla)
    - Carry out a review (inspection) (IEEE 1028)
  - Assess Product Quality.
  - Finalize/update quality assurance plan (IEEE 730)
Conclusion

• Our students learn how to engineer a software product and to implement quality assurance practices and tools.

• Many changes have been made to the SQA course since its initial development in 2001.
  – Following the improvements, this course has scored 4.2/5.0
    • An improvement of 0.6 on the previous format.
  – Adding the utilisation of tools has made the most significant difference in the scores.

• The current SQA course lectures and laboratory sessions provide a solid foundation for future software engineers.

• SQA is still perceived as a low priority by most Small and Medium Enterprises (SMEs) and Very Small Enterprises (VSEs).
  – They have not calculated their cost of poor quality yet!

• The profession of software engineering is still young, and we know that Rome was not built in a day.

Back-up Slides
SQA Course Topics

1. **Introduction**: Introduction to software quality, definitions and the cost of quality.
3. **Standards and Models**: Key standards: IEEE 12207 and ISO/IEC 90003. Models such as the Capability Maturity Model Integration (CMMI).
5. **Software Life Cycle Process**: The SQA process and activities
6. **Software Reviews**: Types of review, as defined by the IEEE 1028 standard
7. **Software Inspection**: The inspection process and the cost and benefit.
8. **Software Quality Assurance Plan**: According to standard IEEE 730
9. **Verification and Validation**: V&V practices according to the IEEE 1012 standard
10. **Software Configuration Management**: Management of Change
11. **Measurement**: Product and Process Measures
12. **Supplier SQA**: The activities relating to the management of the subcontractors with regard to the contribution of suppliers in providing a quality product.
13. **Risk and Quality**: Identification of risks related to SQA

Resources Available to Students

<table>
<thead>
<tr>
<th>SWEBOK</th>
<th>Software Engineering Body of Knowledge</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEEEExplore</td>
<td>Online library of all IEEE standards, papers and magazines.</td>
</tr>
<tr>
<td>CMMI</td>
<td>The Capability and Maturity Model Integration is used during the lectures to illustrate SQA practices, such as peer review.</td>
</tr>
<tr>
<td>Standards</td>
<td>ISO and IEEE standards</td>
</tr>
<tr>
<td>Web site</td>
<td>All the information concerning the course is available online. The course Web site contains the professor’s teaching materials</td>
</tr>
<tr>
<td>Textbook</td>
<td>Galin, D., “Software Quality Assurance – From Theory to Implementation”</td>
</tr>
<tr>
<td>Tools</td>
<td>CVS: Software Configuration Management repository</td>
</tr>
<tr>
<td></td>
<td>Bugzilla: Tracking and reporting bugs and change requests</td>
</tr>
<tr>
<td></td>
<td>CheckStyle: To verify the source code conformance to the programming language standard</td>
</tr>
<tr>
<td></td>
<td>Eclipse: Development environment with a multitude of plug-ins.</td>
</tr>
<tr>
<td></td>
<td>Logiscope: Product quality measurement</td>
</tr>
<tr>
<td>Course notes</td>
<td>Contain standards, case studies, papers and some examples of plans</td>
</tr>
</tbody>
</table>
**SQA Laboratory Topics**

1. **Code of Ethics**: Robot killer case study; Find the violated clauses of the Code; and Determine the responsibilities.

2. **Team project - Draft a project plan and a software requirement specifications document (SRS)**: 1) Software quality plan (IEEE standard 730); 2) Take into consideration the customer's local Java programming rules; 3) Develop project plan and estimates by Cost of Quality items; 4) Use the IBM Rational SRS template (functional and nonfunctional requirements (use ISO/IEC 9216 for the nonfunctional requirements)); 5) Carry out a walkthrough of the documents produced; and 6) Carry out a traceability analysis with the IBM Rational RequisitePro tool or Excel.

3. **Team project - Software Configuration Management (SCM) and Traceability**: 1) Implement the SCM plan using IEEE 730 and IEEE 828; 2) Document the SCM procedure using the Entry-Task-Verification-Exit notation (ETVX) (Rad85); 3) Update the project effort estimation; 4) Read documentation, and configure and test CVS tool for the following roles: system administrator, the individual responsible for configuration management and users; and 5) Carry out a walkthrough of the document produced.

4. **Team project - Programming and test**: 1) Program additional features into existing software; 2) Test the software produced; 3) Update information on the IBM RequisitePro/Excel, Bugzilla and CVS tools; and 4) Update the project effort.

5. **Team project - Problem/change and defect management and inspection**: 1) Complete section 4.8 of the IEEE 730 standard; 2) Document the change/problem and defect management procedure using the ETVX notation; 3) Read the documentation, configure and test the Bugzilla tool; 4) Print the statistics and management reports from Bugzilla; 5) Carry out a walkthrough; and 6) Update the project effort.

6. **Team project - Product Quality Assessment**: 1) Assess source code conformance to customer standards using CheckStyle and software complexity/quality using Logiscope.

7. **Team project - Finalize/update quality assurance plan**: 1) Finalize the plan according to IEEE standard 730; 2) Inspect the plan; 3) Carry out an evaluation of team members (peer evaluation); 4) Carry out a project postmortem; and 5) Use the effort estimation and project tracking data to: 1. Analyze the variations and costs of quality, explain the variations, explain how, in similar projects, the tasks could be carried out to minimize the variations and the costs of an absence of quality (rework).

---

**Software Quality Knowledge Area**

- **Software Quality**
  - **Software Quality Fundamentals**
    - Software Engineering Culture and Ethics
    - Value and Costs of Quality
    - Models and Quality Characteristics
    - Quality Improvement
  - **Software Quality Management Processes**
    - Software Quality Assurance
    - Verification and Validation
    - Reviews and Audits
  - **Practical Considerations**
    - Application Quality Requirements
    - Defect Characterization
    - Software Quality Management Techniques
    - Software Quality Measurement

*Note: Tests are covered in LOG 510*
Quality of Software

• **Effort to find and fix**
  – Finding and fixing a software problem after delivery is often 100 times more expensive than finding and fixing it during the requirements and design phase.

• **Amount of avoidable rework**
  – Current software projects spend about 40 to 50 percent of their effort on avoidable rework.

• **Software quality at delivery**
  – About 40 to 50 percent of user programs contain nontrivial defects.

ISO Software & Systems Engineering Standards

![Bar chart showing the number of standards published and maintained from 1987 to 2007.](chart.png)
**Inspection Process**

1. **100 Plan Inspection**
2. **110 Conduct Kickoff Meeting**
3. **120 Conduct Document Checking**
4. **130 Conduct Logging Meeting**
5. **140 Edit Document**

- **Checklists**
- **Rules**
- **Source**
- **Process improvements**

**Change requests**

**150 Complete Follow-up Exit & Release**


---

**Inspection Effectiveness**

<table>
<thead>
<tr>
<th>CMM Maturity Level</th>
<th>Defect Removal Effectiveness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 5</td>
<td>90% +</td>
</tr>
<tr>
<td>Level 4</td>
<td>75% - 90%</td>
</tr>
<tr>
<td>Level 3</td>
<td>65% - 75%</td>
</tr>
<tr>
<td>Level 2</td>
<td>50% - 65%</td>
</tr>
<tr>
<td>Level 1</td>
<td>Less than 50%</td>
</tr>
</tbody>
</table>

Why do software projects fail so often?

• **Among the most common factors:**
  – Unrealistic or unarticulated project goals
  – Inaccurate estimates of needed resources
  – Badly defined system requirements
  – Poor reporting of the project’s status
  – Unmanaged risks
  – Poor communication among customers, developers, and users
  – Use of immature technology
  – Inability to handle the project’s complexity
  – Sloppy development practices
  – Poor project management
  – Stakeholder politics
  – Commercial pressures

**Defect Density**

Defect Density = Defects/Thousand DSI

**Cost of Quality**

- Prevention
- Appraisal
- Rework
- TCoSQ
- Cost of Conformance