Measuring the Cost of Software Quality of a Large Software Project at Bombardier Transportation

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

Project Editor of ISO/IEC 29110 for Very Small Entities

Professor Normand Séguin, Ph.D.

Universidad de Lima, Peru
September, 13th 2012

http://www2.ulima.edu.pe/
Content

• Introduction
• Business Rationale for Software Quality Assurance and the Measurement of the Cost of Quality
• Undergraduate/Graduate Software Quality Assurance Courses and the Measurement of the Cost of Quality
• Measurement of the Cost of Software Quality – A Case Study
• Conclusion
École de technologie supérieure
Over 6,3000 students
Each year 2,200 paid industrial internships in over 900 companies.
Students are paid about 36,500$ for 3 internships of 4 months

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

• 600 students (400 in Software Eng.)
• 20 Professors in the department have a mean industrial experience of 10 years.

www.etsmtl.ca
Overview of my Background

Oerlikon Aerospace

National Defence
Defence Nationale

(1973-1991)

ÉTS
Engineering for Industry

Department of Software and IT Engineering

(2000 - )
Project Editor of ISO/IEC 29110 Standards and Guides

Joint Committee for IT

Standardization of processes, supporting tools and supporting technologies for the engineering of software products and systems.

Mandated to develop Standards and Guides for Very Small Entities *

Very Small Entity - An entity (enterprise, organization, department or project) having up to 25 people.
**ISO/IEC 29110 Standards and Guides for Very Small Entities (VSEs)**

- **Entry** - Targets VSEs typically developing 6 person-month projects and start-up VSEs;
- **Basic** - Targets VSEs developing only one project at a time;
- **Intermediate** – Targets VSEs developing multiple projects within the organizational context;
- **Advanced** – Targets VSEs which want to sustain and grow as an independent competitive software development business.

*Very Small Entities* are enterprises, projects or departments having up to 25 people.
Content

• Introduction

• Business Rationale for Software Quality Assurance and the Measurement of the Cost of Quality

• Undergraduate/Graduate Software Quality Assurance Courses and the Measurement of the Cost of Quality

• Measurement of the Cost of Software Quality – A Case Study

• Conclusion
Software Development – What the customer wants

Faster!  Better Quality!

Cheaper!

More productivity!

No Cost

Over runs

No Delays!

First To Market

Best Approach = Better → Faster → Cheaper
Why Software Fails?

‘Studies have shown that software specialists spend about 40% to 50% of their time on avoidable rework rather than on what they call value-added work, which is basically work that’s done right the first time.

Once a piece of software makes it into the field, the cost of fixing an error can be 100 times as high as it would have been during the development stage.’

Software managers and software engineers ‘must’ select and use appropriate practices to reduce rework (i.e. waste or ‘scrap’)

Adapted from (Charette 2005)
Components of Project Cost

- **Cost of Quality**
  - Generation of plans
  - Software Development

- **Cost of Conformance**
  - Fixing defects
  - Re-testing
  - Re-reviews
  - Updating source code
  - Modifying documents

- **Cost of Performance**
  - Appraisal Costs
    - Reviews
    - Inspections
    - Testing
    - IV&V
    - Audits
  - Prevention Costs
    - Training
    - Methodologies
    - Tools
    - Data gathering

Adapted from (Haley et al., 1995)
Cost of Quality

- Data from Professional Software Engineers

<table>
<thead>
<tr>
<th></th>
<th>Site A American Engineers (19)</th>
<th>Site A American Managers (5)</th>
<th>Site B European Engineers (13)</th>
<th>Site C European Engineers (14)</th>
<th>Site D European Engineers (9)</th>
<th>Course A 2008 (8)</th>
<th>Course B 2008 (14)</th>
<th>Course C 2009 (11)</th>
<th>Course D 2010 (8)</th>
<th>Course E 2011 (15)</th>
<th>Course F 2012 (10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost of performance</td>
<td>41%</td>
<td>44%</td>
<td>34%</td>
<td>31%</td>
<td>34%</td>
<td>29%</td>
<td>43%</td>
<td>45%</td>
<td>45%</td>
<td>34%</td>
<td>40%</td>
</tr>
<tr>
<td>Cost of rework</td>
<td>30%</td>
<td>26%</td>
<td>23%</td>
<td>41%</td>
<td>34%</td>
<td>28%</td>
<td>29%</td>
<td>30%</td>
<td>25%</td>
<td>32%</td>
<td>31%</td>
</tr>
<tr>
<td>Cost of appraisal</td>
<td>18%</td>
<td>14%</td>
<td>32%</td>
<td>21%</td>
<td>26%</td>
<td>24%</td>
<td>18%</td>
<td>14%</td>
<td>20%</td>
<td>27%</td>
<td>20%</td>
</tr>
<tr>
<td>Cost of prevention</td>
<td>11%</td>
<td>16%</td>
<td>11%</td>
<td>8%</td>
<td>7%</td>
<td>14%</td>
<td>10%</td>
<td>11%</td>
<td>10%</td>
<td>8%</td>
<td>9%</td>
</tr>
<tr>
<td>Quality</td>
<td>71</td>
<td>8</td>
<td>23</td>
<td>35</td>
<td>17</td>
<td>403</td>
<td>19</td>
<td>48</td>
<td>35</td>
<td>60</td>
<td>55</td>
</tr>
</tbody>
</table>

Quality = Number of Defects/ 1,000 Lines of Code
Defect ‘Injection’ During Development

We must identify defects as soon as possible to reduce cost and schedule slippage.

Adapted from (Selby 2007)
We must implement software engineering practices to detect and correct 90% of defects as early as possible to reduce cost and schedule slippage.
Cost of Quality

% of Total Project Cost

Start of Initiative

Cost of Non Conformance (Rework)

Appraisal & Prevention Costs

Maturity Level


Adapted from (Haley et al., 1995)
Content

• Introduction
• Business Rationale for Software Quality Assurance and the Measurement of the Cost of Quality
• Undergraduate/Graduate Software Quality Assurance Course and the Measurement of the Cost of Quality
• Measurement of the Cost of Software Quality – A Case Study
• Conclusion
Software Quality Assurance Course at ETS

- **Fourteen Topics**
  - Chapter 1: Basic Software Quality Assurance Knowledge
  - Chapter 2: Quality Culture and Cost of Quality (COQ)
  - Chapter 3: Quality Requirements
  - Chapter 4: Software Engineering Standards and Models
  - Chapter 5: Reviews and COQ
  - Chapter 6: Software Audit
  - Chapter 7: Verification and Validation and COQ
  - Chapter 8: Tests and Software Quality Assurance
  - Chapter 9: Software Configuration Management and COQ
  - Chapter 10: Policies, Processes and Procedures and COQ
  - Chapter 11: Measurement and COQ
  - Chapter 12: Management of Suppliers and Contracts
  - Chapter 13: Risk Management and COQ
  - Chapter 14: Software Quality Assurance Plan

**Vol. 1**

**Vol. 2**

**French Textbooks**

- L'assurance qualité logicielle 1
- L'assurance qualité logicielle 2

**English Textbook**

Software Quality Assurance

Publication in 2013
Software Quality Assurance Course Laboratory

• **Software Development Project with ISO/IEC 29110**
  – Put in practice the software quality practices learned in class,
  – Each team manages and executes a project using ISO/IEC 29110
  • A process is a component of the COQ (i.e. Prevention)

ISO/IEC 29110 Standard for Very Small Entities
Graduate Software Engineering Program

- Twelve 3-credit courses and a 9-credit Project in Software engineering
- Students are typically from industry with a minimum of 2 years of experience
  - Students typically have an undergraduate degree in either computer science or management information systems and work for small and medium-sized organizations.

- **Software Quality Assurance Course**
  - Students, in teams of 3, have to do a Project in an Organization
    - Measurement of the Cost of Quality
Content

• Introduction
• Business Rationale for Software Quality Assurance and the Measurement of the Cost of Quality
• Undergraduate/Graduate Software Quality Assurance Courses and the Measurement of the Cost of Quality
• Measurement of the Cost of Software Quality – A Case Study
• Conclusion
Measuring the Cost of Software Quality of a Large Software Project at Bombardier Transportation in Canada

Laporte, C. Y., Berrhouma, N., Doucet, M., Palza-Vargas, E.

(Laporte et al. 2012)
Challenges Facing Railway Manufacturers

- Better, Faster, Cheaper
  - Criticality of software
    - Financially, environmentally or for human safety.
  - Multi-disciplinary system development,
  - Integrator-Suppliers Relationships,
  - Multi-country development,
  - Multi-cultural teams,
  - Downsizing/Merger/Turnover,
  - Off shoring.

ERTMS / ETCS (European Rail Traffic Management System / European Train Control System)
Overview of Bombardier

- Workforce of some 80,000 people in over 24 countries.
- **Bombardier Aerospace**
  - World leader in the manufacture of business jets and regional transport,
- **Bombardier Transportation**
  - Leader in the manufacture of rail transport equipment,
  - Manufactures locomotives, freight cars, and propulsion and control systems,
  - Provides systems and signalling equipment.
- Modern trains and subways are increasingly complex, and more and more subsystems are computer-controlled, such as propulsion and braking systems.
- At the time of this case study, there were over 30 software development centers within Bombardier Transportation, for a total of about 950 software engineers.

(Laporte et al. 2012)
The Software Development Group of Bombardier Transportation in Québec (Canada)

- A group of 30 software engineers whose role is to design, develop, and maintain embedded software for trains and subways
  - Software monitoring system used for collecting software maintenance information,
  - Software for controlling car inclination.

(Laporte et al. 2012)
Software Engineering Process of the Software Engineering Group *

* Located in Québec, Canada (Laporte et al. 2012)

Scope of ISO/IEC 15939
Objectives of the Measurement Project of the Software Development Group

1. Identify a project to measure the Cost of Software Quality (CoSQ),
2. Collect data on costs,
3. Categorize costs related to software quality,
4. Develop a data model for the measurement of CoSQ,
5. Analyze data collected, on the selected project, using the data model,
6. Present the CoSQ report to senior management,
7. Expand the measurement of the CoSQ to other software projects in the Software Development Group.

(Laporte et al. 2012)
Overview of the Project Selected

- Development of the software to control the subway of a large American city,
- Team of 15 software engineers,
- Large number of software tasks
  - 1,121 tasks
- Total effort of the completed project
  - 88,598 hours

(Laporte et al. 2012)
The Five Stages of the Measurement Project using the ISO/IEC 15939 Standard

1. Identification of the project tasks related to the CoSQ
   – Identify measurement requirements and resources
2. Development of a list of typical tasks related to the CoSQ
   – Define data measures
3. Categorization of the tasks related to the CoSQ
   – Select and plan measures
4. Development and application of weighting rules
   – Define measurement criteria
5. Determination of the confidence of the weighting rules
   – Define data collection and analysis

(Laporte et al. 2012)
Stage 1. Identification of Project Tasks Related to the CoSQ

- Software engineering process is composed of 4 ‘types’
  - Life Cycle,
  - Sub-process,
  - Process,
  - Activity.

<table>
<thead>
<tr>
<th>Type</th>
<th>Process Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Life Cycle</td>
<td>Primary Life Cycle Process</td>
</tr>
<tr>
<td>Process</td>
<td>Development</td>
</tr>
<tr>
<td>Sub-process</td>
<td>Code and Debug</td>
</tr>
<tr>
<td>Activity</td>
<td>Unit Tests</td>
</tr>
</tbody>
</table>

Representation of BSEP elements

<table>
<thead>
<tr>
<th>WBS Element</th>
<th>Task Name</th>
<th>Effort [hours]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code document</td>
<td>Monitoring-Unit Testing</td>
<td>91.1</td>
</tr>
</tbody>
</table>

Sample registration of a task in the Project accounting system

(Laporte et al. 2012)
Stage 2. Development of a list of tasks related to the Cost of Software Quality

- A list of tasks related to the CoSQ is developed for this project.

<table>
<thead>
<tr>
<th>Primary Life Cycle Processes</th>
<th>Organizational Life Cycle Processes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply</td>
<td>Management</td>
</tr>
<tr>
<td>Development</td>
<td>Estimate the Project</td>
</tr>
<tr>
<td>System Requirements Analysis</td>
<td>Plan the Project</td>
</tr>
<tr>
<td>System Architectural Design</td>
<td>Manage Risks</td>
</tr>
<tr>
<td>Software Requirements Analysis</td>
<td>Track the Project</td>
</tr>
<tr>
<td>Software Architectural Design</td>
<td>Infrastructure</td>
</tr>
<tr>
<td>Software Detailed Design</td>
<td>Plan Infrastructure Activities</td>
</tr>
<tr>
<td>Software Coding and Testing</td>
<td>Establish the Project Infrastructure</td>
</tr>
<tr>
<td>Software Integration</td>
<td>Establish the Global Infrastructure</td>
</tr>
<tr>
<td>Software Validation Testing</td>
<td>Maintain the Infrastructure</td>
</tr>
</tbody>
</table>

Subset of CoSQ-related Tasks

(Laporte et al. 2012)
Stage 3. Categorization of tasks related to the CoSQ

- All 1,121 tasks were sorted as follows:
  - Implementation (I),
  - Evaluation (E),
  - Prevention (P),
  - Rework (R) (internal and external anomalies).

<table>
<thead>
<tr>
<th>Task Identification</th>
<th>WBS Element</th>
<th>Task Name</th>
<th>I</th>
<th>E</th>
<th>P</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>2410</td>
<td>Code</td>
<td>Trace Requirements</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>-</td>
</tr>
</tbody>
</table>

Example of classification of the Requirements Traceability Task

(Laporte et al. 2012)
Stage 4. Development of weighting rules

- Many tasks belong to more than one CoSQ category
  - e.g. The ‘Test and coding’ task overlaps the Evaluation and Implementation CoSQ categories.
- Twenty-seven (27) weighting rules have been defined.

<table>
<thead>
<tr>
<th>Rule Number</th>
<th>Name of Rule</th>
<th>Typical Tasks</th>
<th>Implementation</th>
<th>Evaluation</th>
<th>Prevention</th>
<th>Rework</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Process Audit</td>
<td>Software Project &amp; Process Audit</td>
<td></td>
<td></td>
<td></td>
<td>100%</td>
</tr>
<tr>
<td>7</td>
<td>Review</td>
<td>Design review</td>
<td></td>
<td></td>
<td></td>
<td>100%</td>
</tr>
<tr>
<td>12</td>
<td>Problem Correction and Coding</td>
<td>Corrections, Debugging and Final Coding</td>
<td>30%</td>
<td>70%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Training</td>
<td>Training of new resources</td>
<td></td>
<td></td>
<td></td>
<td>100%</td>
</tr>
<tr>
<td>26</td>
<td>Follow-up &amp; Validation</td>
<td>Follow-up &amp; Validation (all releases)</td>
<td>85%</td>
<td></td>
<td></td>
<td>15%</td>
</tr>
</tbody>
</table>

(Laporte et al. 2012)
Stage 5. Determining the Confidence of Weighting Rules

• What is the level of confidence of the CoSQ measures?
• Another component was added to the rules
  – "H" for high precision,
  – "M" for medium precision, and
  – "L" for low precision.

<table>
<thead>
<tr>
<th>Confidence with the Weighting Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
</tr>
<tr>
<td>M</td>
</tr>
<tr>
<td>L</td>
</tr>
</tbody>
</table>

Total number of activities

(Laporte et al. 2012)
Cost of Software Quality Measurement Results

<table>
<thead>
<tr>
<th>Perform</th>
<th>Hours</th>
<th>59,231</th>
</tr>
</thead>
<tbody>
<tr>
<td>Percentage</td>
<td>67 %</td>
<td></td>
</tr>
</tbody>
</table>

- Total effort (i.e. cost) of the project: 88,598 hours
- Number of software tasks: 1,121 tasks

(Laporte et al. 2012)
Cost of Software Quality at Raytheon compared to Bombardier Transportation

- Total Cost of Software Quality (40%) = 32%
- Percentage of Evaluation (≈20%) = 17%
- Percentage of Rework (≈13%) = 11%
- Percentage of Prevention (≈7%) = 4%

Adapted from (Haley 1996)
Cost of Quality and Maturity Levels

• A study of the Software Engineering Institute (SEI) showed that rework varies between 15% and 25% of the cost of developing a CMM Maturity Level of 3.

<table>
<thead>
<tr>
<th>CMM Maturity Level</th>
<th>Percentage of Rework</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>≥ 50 %</td>
</tr>
<tr>
<td>2</td>
<td>25 % to 50 %</td>
</tr>
<tr>
<td>3</td>
<td>15 % to 25 %</td>
</tr>
<tr>
<td>4</td>
<td>5 % to 15 %</td>
</tr>
<tr>
<td>5</td>
<td>≤ 5 %</td>
</tr>
</tbody>
</table>

Relationship between CMM Maturity Levels and percentage of rework

(Gibson et al. 2006)
Recommandations

1. **Continue data collection and CoSQ Measurement**
   - Continue the data collection using procedures and methods (for example use of tools, database structures) within the SDG.

2. **Control activities related to the correction of problems**
   - Most defects are found mainly during the execution of software requirements, software design and coding tasks
     - Reduce the defects by increasing prevention effort
       - e.g. Execute more peer reviews (walk-through, inspection)

3. **Present the CoSQ measurement results to Management**
   - To enable them to develop better budgets for business process improvement.
   - Establish a scoreboard to display the CoSQ and the results of process improvement activities to illustrate the relationships between them.

(Laporte et al. 2012)
Recommandations

4. Measure the CoSQ at other Bombardier Transportation sites
   – To identify opportunities for reducing costs,
   – To provide a basis to budget development and quality activities,
   – To use the results of CoSQ measures to improve processes.

(Laporte et al. 2012)
Content

• Introduction
• Business Rationale for Software Quality Assurance and the Measurement of the Cost of Quality
• Undergraduate/Graduate Software Quality Assurance Courses and the Measurement of the Cost of Quality
• Measurement of the Cost of Software Quality – A Case Study
• Conclusion
Conclusion

• The concept of Cost of Software Quality is a powerful tool to help quantify the rework, identify weaknesses and prevention activities,

• The Software Quality Assurance courses we teach help software engineers in understanding the value of the quality assurance practices learned in class and put into practice in the laboratories,

• The Case study at Bombardier Transportation demonstrated the business value of the measurement of the Cost of Software Quality.

Best Approach = Better → Faster → Cheaper
Thank you for your attention
Contact Information

• Claude Y Laporte
  – Voice: +1 514 396 8956
  – E-Mail: Claude.Y.Laporte@etsmtl.ca
  – Web: http://profs.etsmtl.ca/claporte/English/index.html

• Public site of WG 24
  – Free access to Deployment Packages, presentation material and articles:
    • http://profs.logti.etsmtl.ca/claporte/English/VSE/index.html
References

- ISO/IEC 15289:2006 - Systems and software engineering - Content of systems and software life cycle process information products (Documentation)
- Konrad, M., Overview of CMMI Model, Presentation to the Montréal SPIN, November 21, 2000, Montréal, Canada.