Teaching Software Quality Assurance in an Undergraduate Software Engineering Program

Claude Y. Laporte*
Claude.Y.Laporte@etsmtl.ca
+1 514 396-8956

Alain April*
Alain.April@etsmtl.ca
+1 514 396-8682

Khaled Bencherif**
khaled.bencherif@ericsson.com
+213 21 77 11 84

** École de technologie supérieure*
Department of Software and It Engineering
1100, Notre-Dame Street West, Montréal, Québec, Canada, H3C 1K3.

*** Ericsson Algeria S.A.R.L*
71 Rue Mohamed Belkacemi, El Madania,
16075 Algiers, Algeria

Abstract
Automation or industrial automation is the use of computers to control industrial machinery and processes, replacing human operators. It is a step beyond mechanization, where human operators are provided with machinery to help them in their jobs. The most visible part of automation can be said to be industrial robotics. Industrial automation relies heavily on software quality. Software quality assurance is taught within the Software Engineering undergraduate curriculum at the École de technologie supérieure in Montréal, Canada. Throughout the course we stress the concept of the cost of quality to convince students of the importance of putting in place adequate prevention and appraisal practices in order to reduce software project costs and failures. The lectures cover a wide spectrum of quality assurance techniques and tools. In addition to weekly 3-hour lectures, the course includes a project in which students have an opportunity to work with industrial software quality assurance techniques and tools.

Keywords: software quality assurance, software quality measurement, software quality improvement, software quality fundamentals, cost of quality, software quality and ethics.

Introduction

Automation or industrial automation is the use of computers to control industrial machinery and processes, replacing human operators. It is a step beyond mechanization, where human operators are provided with machinery to help them in their jobs. The most visible part of automation can be said to be industrial robotics. Industrial automation relies heavily on software quality. Software quality assurance is taught in the lecture format within the Software Engineering undergraduate and graduate curriculum at the École de technologie supérieure (ÉTS) in Montréal, Canada. This curriculum, based on the Guide to the Software Engineering Body of Knowledge (SWEBOK), stresses in the concept of the cost of quality to convince students of the importance of putting in place adequate prevention and appraisal practices in order to reduce software project costs and failures. The lectures cover a wide spectrum of quality assurance techniques and tools. In addition to a weekly 3-hour lecture, the course also includes practical sessions in which students have an opportunity to work with industrial automation software quality assurance techniques and tools.

Quality is increasingly seen as critical to industrial automation success, customer satisfaction and acceptance. Its absence may result in financial loss, dissatisfied customers and damage to the environment, and may even endanger lives [1]. Software quality assurance (SQA) becomes even more important when we consider all the software development projects that have failed, and the financial losses generated by those failures [2].

ÉTS began offering its software engineering undergraduate program in 2001. The aim of the SQA course, which is mandatory in this software engineering curriculum, is to ensure that software engineering students are aware of the importance of SQA, and that they understand and are able to manage its theoretical and practical aspects. This includes knowledge of key standards from the International Organization for Standardization (ISO) and IEEE, as well as how to use SQA tools in practice. The course allows students to apply SQA practices across the whole software life cycle.

The professors who designed the SQA course and are now teaching it have over 25 years of industrial experience, mainly in the telecommunications and defense sectors. The course is made up of lectures, practical exercises and group projects. A continuous process of student evaluation is carried out to ensure that the concepts are well understood. Assessments are performed using exams, laboratory sessions and mini-tests. Commercial tools and open-source software tools provide the necessary support to students to enable them to work with SQA as it is performed in industry.

This paper is divided into four sections: In section 1, we present an overview of the undergraduate program in software engineering as it applies to industrial automation. In section 2, we briefly introduce the SWEBOK Guide and the SQA knowledge area. In section 3, we present a detailed description of the SQA course. The current difficulties and future improvements are discussed in section 4. Finally, we conclude this paper by raising issues related to the SQA course and its impact on the students in their professional lives.

Section 1. Overview of the Software Engineering Curriculum

The software engineering curriculum is a ten-term program, and includes three four-month mandatory paid internships in industry. Courses are offered during all three four-month terms of the year. Students may opt to complete their internships during the fall, winter or summer terms. Every course includes a weekly three-hour lecture and a weekly two- or three-hour laboratory session where students must complete practical assignments. Laboratories are equipped with modern equipment as well as software. Table 1 lists the software engineering courses in the curriculum. Optional software engineering courses are not shown in the table.

The program has been designed to meet the criteria of the Canadian Engineering Accreditation Board. The program was accredited for the first time in 2001 and for a second time in 2006. Graduates of the ÉTS are automatically admitted to Quebec's professional engineering body (OIQ: Ordre des ingénieurs du Québec, Professional Association of Engineers of the Province of Québec).

Section 2. The Guide to the Software Engineering Body of Knowledge

The objectives of the Guide to the Software Engineering Body of Knowledge (SWEBOK) [3] are to characterize the content of the software engineering discipline, to promote a consistent view of software engineering worldwide, to clarify the place, and set the boundary, of software engineering with respect to other disciplines, and to provide a foundation for curriculum development and individual licensing material. The SWEBOK Guide is a project of the IEEE Computer Society. It is a consensually validated document available free of charge[1]. The SWEBOK has also been published as ISO Technical Report 19759 [4].

Course Label	Course Title
LOG120	Software Design
LOG220	Advanced Object-Oriented Programming
LOG230	Management of Software Development Process
LOG310	Formal and Semi-Formal Languages (optional)
LOG320	Data Structures and Algorithms
LOG330	**Software quality assurance**
LOG340	User Interface Analysis and Design
LOG410	Requirements Analysis and Specification
LOG420	Software Architecture and Design
LOG510	Quality Control and Measurement
LOG520	Systems Security
LOG540	Analysis and Design of Telecommunications Software
LOG550	Design of Real-Time Computer Systems
LOG610	Telecommunication Networks
LOG620	Algorithm Analysis
LOG630	Introduction to Databases
LOG640	Introduction to Parallel Processing
LOG650	Compilation Techniques
LOG660	High-Performance Databases
LOG710	Principles of Operating Systems and Systems Programming
LOG720	Distributed Object-Oriented Architecture
LOG730	Introduction to Distributed Systems
LOG740	Interactive Multimodal Systems
LOG790	Capstone Project

Table 1 List of Software Engineering Courses

The SWEBOK Guide is oriented toward a variety of audiences. It is aimed at serving public and private organizations in need of a consistent view of software engineering for defining education and training requirements, classifying jobs, and developing performance evaluation policies and career paths. It also addresses the needs of practicing software engineers and software engineering managers, and the officials responsible for making public policy regarding licensing and professional guidelines. In addition, professional societies defining their certification rules and educators drawing up accreditation policies for university curricula will benefit from consulting the SWEBOK Guide, as will software engineering educators and trainers engaged in defining curricula and course content. The SWEBOK Guide seeks to identify and describe the subset of software engineering knowledge that is generally accepted. Generally accepted knowledge applies to most projects most of the time, and widespread consensus validates its value and effectiveness.

The SWEBOK Guide is subdivided into ten

[1] www.swebok.org

knowledge areas (KAs), the descriptions of which are designed to discriminate among the various important concepts, permitting readers to find their way quickly to subjects of interest. Upon finding such a subject, readers are referred to key papers or book chapters selected because they present the knowledge succinctly. The ten KAs are: Software Requirements, Software Design, Software Construction, Software Testing, Software Maintenance, Software Configuration Management, Software Engineering Management, Software Engineering Process, Software Engineering Tools and Methods, and Software Quality. In the SQA course, we cover the Software Quality KA in depth, and also some elements of Software Configuration Management.

Figure 2 illustrates the breakdown of the SQA KA into topics [5]. This KA covers static techniques, those which do not require the execution of the software being evaluated, while dynamic techniques are covered in the Software Testing KA. One of the authors was the editor of the quality chapter of the SWEBOK and ensured the alignment of the SQA course content with the SWEBOK. As an example, the ethics topic is covered by a class presentation of the IEEE/ACM Code of Ethics [6], followed by a two-hour practical session where students have to identify clauses of the Code that were violated in a Case Study entitled “The Case of the Killing Robot” [7].

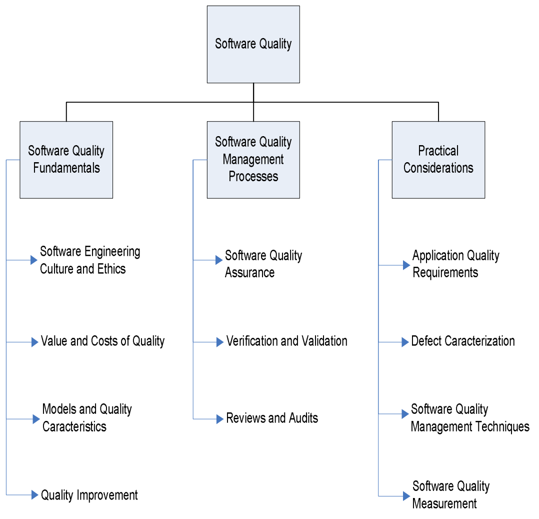
Software Quality
Software Quality Fundamentals
Software Quality Management Processes
Practical Considerations
Software Engineering Culture and Ethics
Value and Costs of Quality
Models and Quality Caracteristics
Quality Improvement
Software Quality Assurance
Verification and Validation
Reviews and Audits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

Figure 2. Breakdown of software quality topics [4]**.**

Section 3. Software Quality Assurance Course

In this section, the lecture material of the SQA course is presented.

3.1 Lectures

The SQA course is composed of thirteen 3-hour lectures. It is designed to follow, as closely as possible, the SQA topic sequence presented in the SWEBOK. Each lecture topic is supported by industrial examples, international standards clauses and process improvement model practices. To ensure that students grasp the importance of SQA activities, the concept of the cost of quality is stressed throughout the course. When performing SQA activities as part of their term projects, students must make tradeoffs between prevention, appraisal, conformance and rework costs (see Figure 3). They must experience, first-hand, that an investment in prevention and appraisal will result in a significant reduction in failure costs (e.g. rework effort).

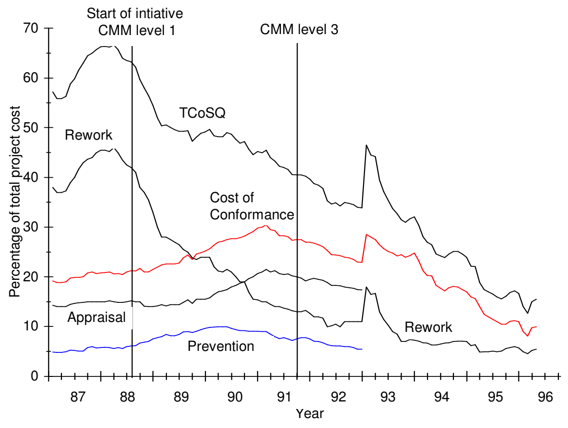

Figure 3. Improvement data [8]

Since data published in papers are sometimes very remote from an undergraduate student’s experience, and to ensure that the principles associated with the cost of quality are well understood, students are required to measure that cost at many points in their term projects. They are also required to analyze their data and draw conclusions on the benefits of SQA activities. Students are often amazed that their own project data may show a cost of failure of 50 and sometimes 70% of total project effort. This helps to make them much more receptive to SQA activities presented in lectures (see Table 2 for details of course lectures).

3.2 Use of Standards

ÉTS has signed an agreement with the IEEE to the effect that, for a nominal fee, all software engineering students and professors have access to the full content of the IEEE electronic library. This includes all IEEE standards. These standards are used in class, both as reading assignments and in the laboratory sessions. Until recently, we were not able to use the ISO standards, as they were too expensive for students. One of the authors has recently finalized an agreement with the Canadian ISO standards providers: the Standards Council of Canada (SCC). The agreement allows all registered SQA students to download standards selected by the professor from the SCC Web site.

3.3 Laboratory Sessions

The laboratory sessions have been designed in such a way that teams of students will apply the SQA theory presented in the lectures to their SQA term projects. Also, to simulate an industrial context where an employee does not usually select his teammates, we create teams by randomly assigning 3 or 4 students to a team. The software engineering department at the ÉTS has installed commercial suites of tools like IBM's Rational Software® and Parasoft Logiscope®, which were obtained, at no cost, through an educational agreement with the suppliers.

Table 2. List of SQA course topics.

Lecture	Description
1	**Introduction:** Introduction to software quality, definitions and the cost of quality.
2	**Code of Ethics:** IEEE/ACM and OIQ Code of Ethics for software engineers. Concrete examples of violations of the Code of Ethics are presented, in order to show the importance of ethical behavior in the engineering profession.
3	**Standards and Models:** Key standards, such as IEEE 12207[9], and ISO/IEC 90003 [10]. Models such as the Capability Maturity Model®[2] Integration SM[3] (CMMI®) [11] are also used to describe SQA process content. Lectures illustrate how those standards are used to develop processes such as the IBM-Rational Unified Process® (RUP).
4	**Quality Model:** Software product quality requirements are developed using the ISO/IEC 9126 quality model [12]. Criticality levels, according to the IEEE 1012 standard, are also presented. This lecture sets the stage for defining quality requirements and how to assess them during a software project.
5	**Software Life Cycle Process:** The following are presented: the software engineer's obligations with respect to quality, the structure and content of IEEE 12207, the SQA process and activities using IEEE12207, and the Process and Product Quality Assurance Process Area of the CMMI.
6	**Software Reviews:** The five types of review, as defined by the IEEE 1028 standard [13], are presented.
7	**Software Inspection:** The inspection process of the IEEE 1028 standard and peer review practices of the CMMI are presented. The cost and benefit data associated with the use of inspections are discussed.
8	**Software Quality Assurance Plan (SQAP):** The software quality assurance plan content, according to standard IEEE 730, is presented [14].
9	**Verification and Validation (V&V):** V&V practices according to the IEEE 1012 standard are described. Students are asked to choose from a list of techniques which one they would like us to explain in detail.
10	**Software Configuration Management (SCM):** SCM practices and SCM roles are presented, according to the CMMI and the IEEE 828 standard.
11	**Measurement:** The measurement program and specific measurements, their objectives in SQA and the implementation of a measurement program are covered.
12	**Supplier SQA:** The activities relating to the management of the subcontractors according to the CMMI are discussed with regard to the contribution of suppliers in providing a quality product. Detailed contract clauses are presented.
13	**Risk and Quality:** The identification of the risks related to SQA and the content of risk management standard IEEE 1540 and ISO/IEC 16085 are covered.

[2] Capability Maturity Model Integration is a service mark of Carnegie Mellon University.

[3] CMMI is registered with the US Patents and Trademarks Office by Carnegie Mellon University.

Laboratories were modified, in the summer of 2005, to add open-source software tools such as CVS for configuration management and Bugzilla for defect-tracking. Since 80% of our graduates will work in small and medium-sized enterprises (SMEs), we thought that exposing them to low-cost tools might help in the deployment of SQA practices in organizations with scarce resources. Also, many students are members of student clubs. These clubs have problems similar to those of very small enterprises (VSEs): limited budget, scarce resources and high turnover. We were pleased to learn that clubs like the remote-controlled submarine and the unmanned piloted helicopter clubs had implemented a few of the SQA practices and open-source tools presented in the SQA course.

Students attend twelve 2-hour laboratory sessions during a semester. They also undertake a 10-week project, where they have to apply the SQA concepts presented in the lectures. Table 3 briefly describes the laboratories that are part of the SQA course.

Table 3. Topics of the SQA laboratories

Topic	Description
1	**Code of Ethics**: 1) Study the IEEE/ACM Code of Ethics and the robot killer case study; 2) Find the violated clauses of the Code; and 3) 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 9126 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: Analyze the variations and the 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).

3.4 SQA Course Web site

The Web site is an important repository for most of the teaching materials, and is also used to post messages to students. It contains general information about the course, such as syllabus, professors' contact information, and the content of the lectures and labs. It is composed of 13 sections, one matching each of the 13 topics presented in the lectures.

For each lesson, the professor recommends mandatory and optional readings. Most reading assignments are chapters from the required textbook [15]. This textbook, which has recently been adopted by the professors, is the first to adequately cover the SQA KA of the SWEBOK. In most other quality assurance textbooks, if not all, only a few aspects of SQA are covered, and the focus of attention is largely on testing. Testing is covered in another software engineering course (LOG510) of the ETS curriculum.

In addition, templates, spreadsheets and forms are available on another Web page. For example, this page contains a template for the team contract, and spreadsheets for the walkthrough and inspection, as well as the spreadsheet for capturing the estimation and the cost of quality. Finally, this page contains an frequently asked questions (FAQ) section on problems typically encountered during lab sessions. This list also helps the student in charge to assist other students during laboratory sessions. There is a similar page for the reading assignments and laboratories.

3.5 Conferences and Guest Speakers

The SQA course students are invited to attend conferences, held at the ETS, organized by the Montréal SPIN (Software Process Improvement Network). Also, occasionally, a guest speaker is invited to present an industrial application of SQA.

3.6 Student evaluation and course evaluation by students

The students are evaluated using mini-tests, laboratories and a final exam. The distribution of the marks is as follows:

- 25% for mini-tests
- 35% for laboratories
- 40% for the final exam

The aim of the mini-tests is to ensure continuous learning by the students. In addition, the students evaluate, anonymously, both the course and the professor towards the end of the semester. This information is transmitted to the professor as input for continuous improvements.

Section 4 Conclusion

Many changes have been made to the SQA course. The challenge was to ensure that all these improvements met the objectives of the course. We think that the current SQA course lectures and laboratory sessions provide a solid foundation for future software engineers, even though SQA is still perceived as a low priority by most SMEs and VSEs [16]. However, the profession of software engineering is still very young, and we know that Rome was not built in a day.

References

[1] Levenson, N, Turner, C, *An Investigation of the Therac-25 Accidents*, IEEE Computer, 1993;Vol. 26, Issue 7; p18-41

[2] Charette, R., Why Software Fails, IEEE Spectrum, September 2005, p42-49.

[3] Abran, A., Moore, J.W., Bourque, P. and Dupuis, R. (eds.). *Guide to the Software Engineering Body of Knowledge*. IEEE Computer Society Press, 2004.

[4] International Organization for Standardization. *Software Engineering Body of Knowledge,* Technical Report ISO/IEC PRF TR 19759, 2005.

[5] April, A., Reeker, L., & Wallace, D. Software quality. *Guide to the software engineering body of knowledge (Ironman version)* (pp 157-170). IEEE-Computer Society Press. Los Alamitos, California, USA, 2004.

[6] Gotterbarn, D., Miller, K., Rogerson, S., *Computer Society and ACM Approve Software Engineering Code of Ethics*, IEEE Computer, October 1999.

[7] Epstein, R.G., *The Case of the Killing Robot*, West Chester University, epstein@golden.wcupa.edu.

[8] Haley, T.J., Software process improvement at Raytheon. *IEEE Software, 13*(6), 33–41. *Figure abstracted from IEEE Software,* 1996.

[9] IEEE 12207, IEEE/EIA 12207.0-1996. *Industry Implementation of Int. Std. ISO/IEC 12207:95, Standard for Information Technology-Software Life Cycle Processes*, 1996.

[10] ISO/IEC 90003, International Organization for Standardization. Software Engineering: *Guidelines for the application of ISO9001:2000 to computer software*, ISO/IEC Standard 90003:2004. International Organization for Standardization/International Electrotechnical Commission: Geneva Switzerland, 2004.

[11] Software Engineering Institute. *Capability maturity model integration for software engineering (CMMi)* (pp. 94–528). Pittsburgh, PA: Carnegie Mellon University. Version 1.2, CMU/SEI-2006-TR-008, 2006.

[12] ISO9126, International Organization for Standardization, *Software Engineering-Product Quality Part 1: Quality Model*, ISO/IEC Standard 9126-1. International Organization for Standardization/International Electrotechnical Commission: Geneva Switzerland, 2001.

[13] IEEE 1028, IEEE Std 1028-2002, *IEEE Standard for Software Reviews*: IEEE, 2002.

[14] IEEE 730, IEEE Std 730-2002. *IEEE Standard for Software Quality Assurance Plans*: IEEE, 2002.

[15] Galin, D., Software Quality Assurance – From Theory to Implementation, Pearson Education Limited, 2004.

[16] Laporte, C.Y., Renault, A., Desharnais, J. M., Habra, N., Abou El Fattah, M., Bamba, J. C., *Initiating Software Process Improvement in Small Enterprises: Experiment with Micro-Evaluation Framework*, *SWDC-REK,* International Conference on Software Development, University of Iceland, Reykjavik, Iceland May 27 - June 1, 2005. pp 153-163.