Computers are used to control machinery, industrial processes (often replacing human operators), and many business processes. Unfortunately, software quality assurance is often the poor relation in many organizations. Most developers are not aware of the high cost of inferior quality. At the École de Technologie Supérieure (ÉTS) in Montréal, Canada, software quality assurance is taught in the lecture format within the software engineering undergraduate curriculum. This curriculum is based on the Guide to the Software Engineering Body of Knowledge (SWEBOK). The course stresses 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 failures. The lectures cover a wide spectrum of quality assurance techniques and tools. In addition to weekly three-hour lectures, the course includes a project in which students have an opportunity to measure the cost of quality and work with industrial software quality assurance techniques and tools. Universities need to emphasize quality in their programs, but many of them do not. Universities from which practitioners hire could model a software quality course after this one.

Key words: cost of quality, software quality and ethics, software quality assurance, software quality fundamentals, software quality improvement, software quality measurement

SQP References

Designing a Software Quality Assurance Course: An Effective Framework for Teaching
vol. 5, no. 3
Daniel J. Zrymiak and Abhijit Sen

GENERAL, KNOWLEDGE, CONDUCT, AND ETHICS

Teaching Software Quality Assurance in an Undergraduate Software Engineering Program

CLAUDE Y. LAPORTE, ALAIN APRIL, AND KHALED BENCHERIF,
École de Technologie Supérieure

INTRODUCTION

Quality is increasingly seen as critical to business success, customer satisfaction, and acceptance. Its absence may result in financial loss, dissatisfied users, and damage to the environment, and may even result in deaths. For example, the Therac-25, a computer-driven radiation system, seriously injured and killed patients by massive overdosing (Levenson and Turner 1993). Software quality assurance (SQA) becomes even more important when one considers all the software development projects that have failed and the financial losses generated by those failures. As reported by Charette (2005), software specialists spend about 40 to 50 percent of their time on avoidable rework.

The École de Technologie Supérieure (ÉTS) began offering its software engineering undergraduate program (see http://profs.logti.etsmtl.ca/departement/index.html) 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 the key ISO and IEEE stan-

dards, 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 more than 20 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 article is divided into four sections. First, the authors present an overview of the undergraduate program in software engineering. Next, they briefly introduce the Guide to the Software Engineering Body of Knowledge (SWEBOK) and the SQA knowledge area. The authors then present a detailed description of the SQA course. Following that, the current difficulties and future improvements are discussed. Finally, the authors conclude the article by raising issues related to the SQA course and its impact on the students in their professional lives.

OVERVIEW OF THE SOFTWARE ENGINEERING CURRICULUM

The software engineering curriculum is a 10-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, excluding courses such as mathematics, physics, management, and social sciences, which are common to all undergraduate engineering programs at the school.

The program was 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 Ordre des ingénieurs du Québec (OIQ) (Professional Association of Engineers of the Province of Québec).

THE GUIDE TO THE SWEBOK

The objectives of the SWEBOK (Abran et al. 2004) 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 (http://www.swebok.org). The

TABLE 1 List of software engineering courses

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

SWEBOK has also been published as ISO Technical Report 19759 (ISO 2005).

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 10 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 10 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. Each is treated in a chapter of the SWEBOK Guide. In the SQA course, the authors cover the software quality KA in depth, and also some elements of software configuration management.

Figure 1 illustrates the breakdown of the SQA KA into topics. 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 (Gotterbarn, Miller, and Rogerson 1999), 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" (Epstein 1994).

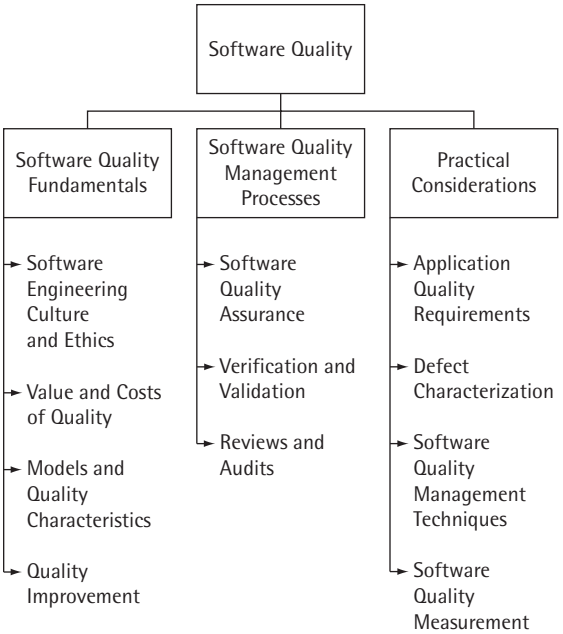
SOFTWARE QUALITY ASSURANCE COURSE

Lectures

The SQA course is composed of 13 three-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 (as illustrated in Figure 2). They must experience firsthand that an investment in prevention and appraisal will significantly reduce failure costs (for example, rework effort). As illustrated in Figure 2, rework cost was, at one point, as high as 45 percent of the total project cost.

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

FIGURE 1 Breakdown of software quality topics (ISO TR19759-05)



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 percent of their total project effort. This helps make them more receptive to SQA activities presented in lectures (see Table 2 for details of course lectures).

Use of Standards

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, the authors were not able to use the ISO standards, as they were too expensive for students. One of the authors recently finalized an agreement with the Canadian

FIGURE 2 Improvement data (Dion 1993; Haley 1996)

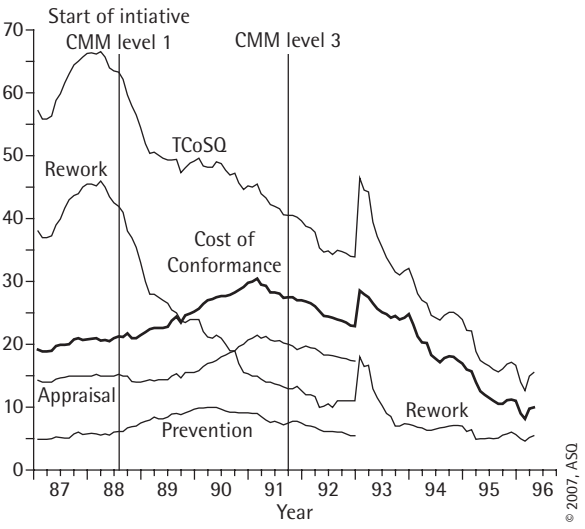


TABLE 2 List of SQA course topics

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, and ISO/IEC 9001 and 90003. Models such as the Capability Maturity Model®* Integration℠** (CMMI®) 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. 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 Quality Assurance Process Area of the CMMI.
6	**Software reviews:** The five types of reviews, as defined by the IEEE 1028 standard, 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.
9	**Verification and validation (V&V):** V&V practices are described. Students are asked to choose from a list of techniques which one they would like explained 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.

* Capability Maturity Model Integration is a service mark of Carnegie Mellon University.

** CMMI is registered with the U.S. Patents and Trademarks Office by Carnegie Mellon University.

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.

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 or her teammates, the authors create teams by randomly assigning three or four students to a team. The software engineering department at the ÉTS has installed commercial suites of tools such as IBM's Rational Software® and Parasoft Logiscope®, which were obtained at no cost through an educational agreement with the suppliers. 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 percent of their graduates will work in small and medium-sized enterprises (SMEs), the authors thought 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. The authors 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 12 two-hour laboratory sessions during a semester. They also undertake a 10-week project where they must apply the SQA concepts presented in the lectures. Table 3 briefly describes the laboratories that are part of the SQA course.

SQA Course Web Site

The Web site is an important repository for most of the teaching materials, and it is also used to post messages to students. It contains general information about the course, such as a syllabus, professors' contact information, and the content of the lectures and labs.

For each lesson, the professor provides a list of mandatory and optional readings. Most reading assignments are chapters from the required textbook

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 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) (Radice et al. 1985); 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).

(Galin 2004). This textbook, which has recently been adopted by the professors, is the first to adequately cover the SQA KA of the SWEBOK. In most, if not all, other quality assurance textbooks only a few aspects of SQA are covered, and the focus is largely on testing. As mentioned previously, testing is covered in another software engineering course of the ÉTS 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 FAQ section on problems typically encountered during lab sessions. This list also helps the student in charge assist other students during laboratory sessions. There is a similar page for the reading assignments and laboratories.

Conferences, Guest Speakers, and Resources

The SQA course students are invited to attend conferences, held at the ÉTS, organized by the Montréal Software Process Improvement Network (SPIN). Occasionally, a guest speaker is invited to present an industrial application of SQA. Also, support material is placed at the disposal of students. Table 4 lists these resources.

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 percent for mini-tests
- 35 percent for laboratories
- 40 percent for the final exam

The aim of the mini-tests is to ensure continuous learning by the students. In addition, at ÉTS the students evaluate, anonymously, both the course and the professor toward the end of the semester. Courses are assessed on a scale of 0 to 5, where 5 is a perfect score. This information is transmitted to the professor as input for continuous improvements.

CONCLUSION

Many changes have been made to the SQA course since its initial development in 2001. The challenge was to ensure that all these improvements met the objectives of the course. Following the improvements, the course scored 4.2, an improvement of 0.6 over the previous format. Adding practical content and tools has made the most significant difference in the scores.

TABLE 4 Resources available to students

Resources	Description
SWEBOK	Software Engineering Body of Knowledge
IEEExplore	Online library of all IEEE standards, as well as papers and magazines. The ÉTS has a subscription to this database, which is accessible via the library Web site.
CMMI	The Capability Maturity Model Integration is used during the lectures to illustrate SQA practices, such as peer review, software configuration management, and so on.
Standards	A selection of IEEE standards are printed as course notes and purchased by the students.
Web site	All the information concerning the course is available online. The course Web site contains the professor's teaching materials and text describing the content of the laboratories. Professors also post the laboratory schedule and messages there.
Textbook	Galin, D. 2004. *Software quality assurance—From theory to implementation.* Upper Saddle River, N.J.: Pearson Education.
Tools	CVS: For the software configuration management repository Bugzilla: For tracking and reporting bugs and change requests CheckStyle: To verify the source code conformance to the programming language standard Eclipse: A development environment with a multitude of plug-ins Logiscope: Product quality measurement The school also participates in the IBM academic program and can therefore use many software tools such as the IBM RequisitePro Traceability tool.
Course notes	Students purchase course notes at the University Coop. The notes contain standards, case studies, papers, and some examples of plans and specification documents. They also contain the IEEE/ACM code of ethics.

The authors think 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 (Laporte et al. 2005). However, the profession of software engineering is still young . . . and Rome was not built in a day.

REFERENCES

Abran, A., J. W. Moore, P. Bourque, and R. Dupuis, eds. 2004. *Guide to the software engineering body of knowledge.* Los Alamitos, Calif.: IEEE Computer Society Press.

Charette, R. 2005. Why software fails. *IEEE Spectrum* (September): 42-49.

Dion, R. 1993. Process improvement and the corporate balance sheet. *IEEE Software* 10, no. 4: 28-35. Figure abstracted from *IEEE Software.*

Epstein, R. G. 1994. *The case of the killing robot.* West Chester University. Available at epstein@golden.wcupa.edu.

Galin, D. 2004. *Software quality assurance—From theory to implementation.* Upper Saddle River, N.J.: Pearson Education.

Gotterbarn, D., K. Miller, and S. Rogerson. 1999. Computer society and ACM approve software engineering code of ethics. *IEEE Computer* (October).

Haley, T. J. 1996. Software process improvement at Raytheon. *IEEE Software* 13, no. 6, 33-41. Figure abstracted from *IEEE Software.*

ISO. 2005. *Software Engineering Body of Knowledge.* Technical Report ISO/IEC PRF TR 19759. Geneva, Switzerland: International Organization for Standardization (ISO).

Laporte, C. Y., A. Renault, J. M. Desharnais, N. Habra, M. Abou El Fattah, and J. C. Bamba. 2005. 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.

Levenson, N., and C. Turner. 1993. An investigation of the therac-25 accidents. *IEEE Computer* 26, no. 7: 18.41.

Radice, R. A. et al. 1985. A programming process architecture. *IBM Systems Journal* 24, no. 2: 79-90.

BIOGRAPHIES

Claude Y. Laporte is a professor at the École de Technologie Supérieure (ÉTS), an engineering school, where he teaches graduate and undergraduate courses in software engineering. His research interests include software process improvement in small and very small companies, and software quality assurance. He received a master's degree in physics from the Université de Montréal and a master's degree from the École Polytechnique de Montréal. He is the editor of an ISO/IEC-JTC1 SC7 working group tasked to develop software life-cycle profiles and guidelines for use in very small enterprises. He is member of the IEEE, the PMI, INCOSE, and l'Ordre des ingénieurs du Québec (Professional Association of Engineers of the Province of Québec). Contact him at the École de Technologie Supérieure's Department of Software and IT Engineering, 1100, rue Notre-Dame ouest, Montréal, Québec, Canada, H3C 1K3 or e-mail at Claude.Y.Laporte@etsmtl.ca.

Alain April is professor of software engineering at the École de Technologie Supérieure (ÉTS), Université du Québec, Montréal, Canada. He obtained a doctorate at the Otto von Guericke University of Magdeburg, Germany. His research interests are: software maintenance, software quality, and multimedia database management systems. He has worked in the IT industry for more than 20 years. April has contributed to the internal measurement of software of ISO 9126 (part 3) and is the associate editor of the SWEBOK software maintenance and quality chapters. April can be reached by e-mail at Alain.April@etsmtl.ca.

Khaled Bencherif is a core network design and optimization engineer with Ericsson. He graduated from the École de Technologie Supérieure (ÉTS) in Montreal, where he received a master's degree in software engineering. He served as a teaching assistant and was involved in projects related to software architecture, software quality assurance, and software safety using international standards. He also graduated from the Université de Versailles in France, where he received a master's degree in software parallelism. He then conducted a research project in peer-to-peer systems in collaboration with the IMAG-ID Laboratory. Prior to acquiring his academic experience and after holding a position as a computer science engineer in the economic studies department at that university, he became a total quality management consultant for a company in Algeria that produces electrical energy. He can be contacted by e-mail at khaled.bencherif@ericsson.com.