# Software Quality Assurance in an Undergraduate Software Engineering Program

*Claude Y. Laporte, Alain April*
*École de technologie supérieure (ÉTS)*
claude.laporte@etsmtl.ca, alain.april@etsmtl.ca

***Abstract*** *– Software tests are used by most organizations. However, many other software quality assurance practices are often neglected. Most developers are not aware of the high cost of inferior quality and its impact on the duration and budget of a project. At the École de technologie supérieure (ÉTS), software quality assurance (SQA) is taught in lecture format in the undergraduate software engineering curriculum. The SQA course covers the concepts of the business model and the cost of quality, to convince students of the importance of putting in place adequate prevention and evaluation practices, both to reduce the number of defects and to predict the extra effort needed to correct defects introduced as the work progresses.*

*The course includes a 10-week capstone project in which teams of 4 students apply the SQA practices taught in class in a software development assignment. The students collect measures throughout the 10-week period, and the performance of each team is analyzed. This analysis allows discussion to take place on the positive impact of SQA practices as a way to deliver quality software on time and within budget.*

**Keywords***:* software quality assurance, standards, ISO/IEC 29110, cost of quality, capstone project.

## 1. INTRODUCTION

As reported by Charette [2], software specialists spend about 40 to 50 percent of their time on avoidable rework. The ability of organizations to compete, adapt, and survive is increasingly dependent on quality, productivity, development time, and cost. Systems and software are growing larger and more complex every year. For example, top-of-the-line cars contain up to 100 million lines of code, 80 processors, and 5 bus systems (Charette, 2005). 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.

The École de technologie supérieure (ÉTS) began offering its undergraduate software engineering program in 2001. Recent publications have described the extensive use of capstone projects, tools, and frameworks in that program [8, 9]. The aim of this specific SQA course (see LOG330 in Table 1), which is mandatory in the ÉTS 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 and apply SQA practices in real situations. This also includes hands-on knowledge of the key ISO and IEEE standards, as well as how to use industrial and open source SQA tools in practice. The course allows students to apply a wide range of SQA practices throughout a software development cycle in a capstone project.

The professors who designed the SQA course, and are now teaching it, have combined industrial experience of more than 20 years, mainly in the telecommunication, defense, and railway sectors. The course is made up of lectures, practical exercises, and a team project. A continuous process of student evaluation is carried out to ensure that the concepts are well understood. The assessments are performed using exams, laboratory sessions, and mini-tests. The software tools, mostly open source, provide the necessary support to students to enable them to work with SQA as it is performed in industry.

This article is divided into two sections. First, the authors present an overview of the undergraduate program in software engineering offered at the ÉTS. They then present a detailed description of the SQA course, and of the laboratory sessions.

## 2. OVERVIEW OF THE SOFTWARE ENGINEERING CURRICULUM

The education system in Quebec is somewhat different from that in the rest of North America. Students attend secondary school for only five years, followed by two or three years at a *Cégep*, or General and Vocational College, which offers college-level programs for entry into university (with two-year pre-university programs) or

training for students for technical or technological careers, the latter being roughly the equivalent of community colleges elsewhere in North America. A student planning to attend university would spend two years at a Cégep, followed by three or four years at a university to earn an undergraduate degree. However, the ÉTS is exceptional, in that its programs are designed for students who have completed a three-year technology-oriented Cégep program. As a consequence, all students accepted at the ÉTS have received training in a technical discipline related to engineering. In the case of software engineering (SE), this means that the students have three years of programming education and experience prior to their arrival at the ÉTS. We are therefore able to teach concepts, such as OOD, OOA, and design patterns, during their first year, and software architecture during their second year, for example.

The ÉTS is exclusively an engineering school, and belongs to the University of Québec network of institutions. The school is relatively young, having been established in 1974. As of April 2013, the total number of active students was approximately 7,000, of which about 1,600 are graduate students. It is the fourth largest engineering school in Canada, in terms of the number of undergraduate students enrolled. The school offers undergraduate engineering programs in six disciplines, including SE, which is the focus of this paper. All the engineering programs at the ÉTS integrate cooperative (co-op) education, which involves practical work carried out in industry. Students must complete three co-op terms, which last four months each. The school places 2,400 students in 1,100 companies each year for these terms. The students are paid around $11,000 for their internships, and this constitutes an important revenue source for them during their studies, as well as establishing the student in an employer/employee relationship. Because of the incoming students' technical background and the co-op nature of our programs, the ÉTS is considered a very "hands-on" school, which it is, both by design and by mission. It is also worth noting that, according to 2004 statistics, 96% of companies in Quebec are small and medium-sized enterprises (SMEs), and 83% of those employ four people or fewer [11]. This reality is taken into account in the design of our programs and courses, especially given the fact that Quebec's economy includes 25% of Canada's IT firms [11].

The SE curriculum is a 10-term program, including the three mandatory 4-month paid internships. Courses are offered during all three 4-month terms of the academic year. Students may opt to complete their internships during the fall, winter, or summer terms. Every course includes a weekly 3-hour lecture and a weekly 2- or 3-hour laboratory session, where students must complete practical assignments. Table 1 lists the core SE courses in the curriculum, and excludes courses such as mathematics, physics, management, and the social sciences, which are common to all undergraduate engineering programs at the school.

**Table 1:** List of software engineering courses at the ÉTS

| Course ID | Course title |
|---|---|
| LOG121 | Object Oriented |
| LOG210 | Software Analysis and Design |
| LOG240 | Tests and Maintenance |
| LOG320 | Data Structures and Algorithms |
| LOG330 | Software Quality Assurance |
| LOG350 | User Interface Design and Evaluation |
| LOG410 | Needs and Requirements Analysis and Specifications |
| LOG430 | Software Architecture |
| LOG515 | Software Project Management |
| LOG550 | Real Time System Design |
| LOG610 | Network and Telecommunications |
| LOG625 | Introduction to Intelligent Systems |
| LOG660 | High Performance Database |
| LOG792 | Capstone Project |

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

## 3. SOFTWARE QUALITY ASSURANCE COURSE

### 3.1 Lectures

The SQA course is composed of thirteen 3-hour lectures. Each lecture topic is illustrated with industrial examples, international or professional standards, and process improvement model practices. To ensure that students grasp the importance of SQA activities, the business model concept and the cost of quality concept are stressed throughout the course. When performing SQA activities as part of their term projects, students must make tradeoffs between prevention, appraisal, conformity, and rework costs. They must experience firsthand that an investment in prevention and appraisal will significantly reduce failure costs in the future (rework effort, for example).

Since data on the cost of quality published in papers are often very far from the students' experience, and to ensure that the principles associated with that cost are

well understood, students are required to continuously measure the cost of rework 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 reveal a cost of failure of 50%, and sometimes 70%, of the total project effort.

The lectures, described in Table 2, follow the sequence set out in the French language textbooks published by the co-authors in 2009 [1,5]. An English version of the textbooks will be published by John Wiley in late 2013 or early 2014 [6].

**Table 2:** List of SQA course topics

| Lecture | Course title |
|---|---|
| 1 | **Introduction**<br>• How is software quality defined?<br>• Business models and the selection of software engineering practices<br>• Software errors, mistakes, and failures<br>• The quality of software<br>• Software quality assurance |
| 2 | **Quality culture**<br>• The cost of software quality<br>• What is a quality culture?<br>• The five dimensions of a software project<br>• The IEEE Code of Ethics for software engineer |
| 3 | **Quality requirements**<br>• Models of software quality (ISO/IEC 25000)<br>• The definition of software quality requirements<br>• The traceability of requirements in the software life cycle |
| 4 | **Standards and models**<br>• Standards, the cost of quality, and business models<br>• An overview of standards and best practice models<br>• The main quality management standards<br>• ISO/IEC/IEEE 12207<br>• IEEE-730<br>• Other models, standards, guidelines, and quality procedures<br>• ISO/IEC 29110 for very small entities<br>• Specific standards for an application domain (e.g. DO-178)<br>• ISO/IEC/IEEE 15289 standard for the description of software products |
| 5 | **Software Reviews**<br>• Personal review, desk check-type review<br>• Reviews described in IEEE-1028 and CMMI for development model<br>• The Walkthrough<br>• The Inspection<br>• The Project launch review<br>• Agile meetings<br>• Measures<br>• Selection of a review type |
| 6 | **Software Audit**<br>• Audit and problem resolution according to ISO/IEC/IEEE 12207<br>• Audit according to the CMMI-Dev model<br>• Audit according to IEEE-1028<br>• Corrective actions<br>• The audit and the software quality assurance plan<br>• A case study |
| 7 | **Verification and validation (V&V)**<br>• Costs and benefits of V&V<br>• Standards (IEEE-1012) and models which require or define V&V<br>• Independent V&V<br>• Traceability<br>• The validation of software<br>• Testing<br>• Checklists<br>• V&V techniques<br>• The V&V plan |
| 8 | **Configuration Management**<br>• The usefulness of software configuration management (SCM)<br>• SCM activities<br>• IEEE-828 Configuration management standard<br>• SCM library and branches<br>• Configuration control<br>• Configuration status accounting<br>• Configuration audit<br>• The implementation of SCM in a small organization<br>• SCM policy |
| 9 | **Policies, processes, and procedures**<br>• Policies<br>• Processes<br>• Procedures<br>• Organizational standards<br>• Documentation standard (ISO/IEC/IEEE 15289)<br>• Case study<br>• The personal software process |
| 10 | **Measurement**<br>• The importance of measurement<br>• The measurement process of ISO/IEC/IEEE 12207<br>• The Practical Software and Systems Measurement Method<br>• ISO/IEC/IEEE 15939 Measurement standard<br>• Measurement in the CMMI-Dev model<br>• The survey as a measurement tool<br>• The implementation of a measurement program<br>• Practical considerations<br>• The human side of measurement |
| 11 | **Risk management**<br>• Risk management according to standards ISO 12207, ISO 9001, and ISO 16326 and the CMMI-Dev model<br>• ISO/IEC/IEEE 16085 Risk Management standard |

| | |
|---|---|
| | • Practical considerations<br>• Human factors in risk management |
| 12 | **Management of suppliers and contracts**<br>• Supplier agreement management according to the CMMI-Dev model<br>• Management of external participants<br>• Life cycle of software acquisition<br>• Types of software contracts<br>• Contract reviews |
| 13 | **Software quality assurance plan**<br>• Introduction<br>• IEEE-730 and the software quality assurance plan |

## 3.2 Use of Standards

SE students and professors at the ÉTS have access to the full content of the IEEE electronic library, including all IEEE standards. These standards are used both in class and in laboratory sessions. Until recently, we were not able to use the ISO standards, as they were too expensive for students to buy. One of the authors has 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 website. The following standards are discussed in the SQA course:

- ISO/IEC/IEEE 24765 (Systems and software engineering - Vocabulary)
- ISO/IEC/IEEE 12207 (Systems and software engineering - Software life cycle processes)
- ISO/IEC 25000 (Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Guide to SQuaRE)
- ISO/IEC/IEEE 16085 (Systems and software engineering - Life cycle processes - Risk management)
- ISO 9001 (Quality management systems - Requirements)
- ISO/IEC 29110 (Software engineering - Life cycle profiles for Very Small Entities (VSEs))
- ISO/IEC 90003 (Software engineering - Guidelines for the application of ISO 9001:2000 to computer software)
- ISO/IEC/IEEE 15939 (Systems and software engineering - Measurement process)
- ISO/IEC/IEEE 15289 (Systems and software engineering - Content of life cycle information products (documentation)

During the course, students are introduced to the recently published ISO software development standard ISO/IEC 29110 [13] targeting Very Small Entities (VSEs). Students use the engineering and management guide included in ISO 29110 which is freely available in English or French from the ISO [3,4], as a framework to help them understand when software quality practices are used in a development project and why. They also use the guide as a framework for their team project.

## 3.3 Laboratory Sessions

The laboratory sessions have been designed in such a way that teams of students can apply the SQA practices presented in the lectures to their SQA term projects.

Students attend twelve 2-hour laboratory sessions during a semester. After completing the first two sessions on the code of ethics and the business models, students embark on a project in teams of four for a period of ten weeks where they must apply the SQA practices presented in the course, using the ISO/IEC 29110 standard as the framework for the project. The teachers randomly create the teams, to simulate an industrial context where an employee doesn't usually choose his teammates.

At the start of the project, the teams receive a copy of the Statement of Work (SOW), which they use to develop the project plan. During the planning phase of the project, the four students in a team must share the following roles, as defined in ISO/IEC 29110: analyst, designer, programmer, technical lead, and project manager. Monitoring this process is the responsibility of the teacher or the lab supervisor, who plays the role of customer.

During the first week of the project, students are also required to select and install the tools they will use during the project. For example, they must choose and install a document repository, a version control tool, and an issue tracking tool, among others.

Then, the four team members must complete and sign a contract which specifies the roles of each participant, the team deliverables, the expectations of each participant, and the operating rules which they agree to respect. The course website lists the objectives and deliverables for each of the ten weeks of the project. The site also contains all the templates required to produce the deliverables. The templates list the content of the documents required by ISO/IEC 29110, such as the project plan and the specifications of the software. The site also includes descriptions of the various reviews they have to perform (e.g. desk check, walkthrough) and forms for registering any anomalies, e.g. defects, they find during the review process.

Teams must estimate the effort that will be needed by each member to carry out the activities and deliverables required by ISO 29110. These estimates are recorded on a spreadsheet, and every week members of the team must record the hours they have worked on defined project

activities. Also, students must record their rework effort. At the end of the project, teams conduct a session on lessons learned (post-mortem), where they analyze the data logged on their spreadsheets. We ask them to explain the differences between the initial estimates and the actual effort expended, including the cost-of-quality components (e.g. rework effort), and ask them to generate findings and develop recommendations for a future project. Table 3 describes the laboratory components of the SQA course.

**Table 3:** Topics of the SQA laboratory sessions

**Business Models**

Description of the business model of the team project for assessing the project risks and selecting practices to mitigate those risks, using the business model descriptions presented in class.

**Code of Ethics**

Identification of potential violations of the IEEE/ACM Code of Ethics during execution of the team project.

**Team Project - Part 1 - Project Planning and installation of the work environment**

Objectives

- Objective 1: Perform the project planning activity according to the basic profile of ISO/IEC 29110, perform a desk check of the project plan;
- Objective 2: Select tools and set up the working environment (e.g. a version control tool and an issue tracking tool);
- Objective 3: Customize the measurement spreadsheet for the measurement of effort and the cost of quality for the project.

Deliverables

1. Project plan:
   - Profile of freedoms/constraints
   - Identification of the criticality of the project
   - Roles and responsibilities of team members
   - Version control strategy
   - Delivery instructions
2. Work environment *[installed and tested]*
3. Contracts among team members
4. Defect registration form (desk check of the project plan)
5. Measurement spreadsheet tailored to this project. *[updated with current data]*

**Team Project - Part 2 - Analysis and documentation of requirements**

Objectives

- Objective 1: Perform the software requirements analysis activity of ISO 29110;
- Objective 2: Perform a walkthrough to verify the specifications before they are submitted to the customer for approval.

Deliverables

1. Functional and nonfunctional requirement specifications [verified and baselined]
2. Audit results
3. Anomaly registration form
4. Validation results
5. Software user documentation *[preliminary]*
6. Measurement spreadsheet *[verified, baselined]*

**Team Project - Part 3 - Software architecture and detailed design**

Objectives

- Objective 1: Create the architecture and the detailed design, perform implementation and evaluation activities
- Objective 2: Perform a walkthrough to verify the architecture.

Deliverables

1. Software design *[verified, baselined]*
2. Verification results of the architecture document
7. Anomaly registration form
3. Traceability record *[verified, baselined]*
4. Test Procedures and test cases *[verified]*
5. Measurement spreadsheet *[verified, baselined]*

**Team Project - Part 4 - Software Construction**

Objectives

- Objective 1: Perform construction, implementation, and evaluation activities of ISO 29110;
- Objective 2: Perform a walkthrough to verify the components developed.

Deliverables

1. Software components *[corrected, baselined]*
2. Correction register (if necessary)
3. Anomaly registration form
4. Analysis of measures collected and recommendations
5. Traceability record [updated, baselined]
6. Change request form [ready to be signed by the client]
7. Measurement spreadsheet [*verified, baselined]*
8. Progress status record *[evaluated]*
9. Analysis of measurements collected and recommendations
10. Analysis of the cost of the quality measures

**Team Project - Part 5 - Software Integration and Tests**

Objective

- Perform integration and testing, execution, and evaluation activities of ISO 29110

Deliverables

1. Test procedures and test cases (updated if necessary) [baselined]
2. Software (i.e. components developed in the previous activity have been integrated) *[tested, baselined]*
3. Traceability record *[updated, baselined]*
4. Test report *[baselined]*
5. Product operation guide *[verified, baselined]*
6. User documentation *[verified, baselined]*
7. Measurement spreadsheet *[verified, baselined]*
8. Progress status record *[evaluated]*
9. Correction register (if necessary)

**Team Project - Part 6 - Product Delivery and Project Completion**

Objectives

- Objective 1: Perform the product delivery activity;
- Objective 2: Conduct a post-mortem review of the project.

Deliverables

1. Maintenance documentation *[verified, baselined]*
2. Software configuration *[delivered]*
3. Correction register (if required)
4. Acceptance form *[signed by the client]*
5. Software configuration *[accepted]*
6. Measurement spreadsheet *[verified, baselined]*
7. Information repository [*updated]*
8. Report on lessons learned

## 4. CONCLUSION

Many changes have been made to the SQA course since it was initially set up 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 version. Adding practical content and tools has made the most significant difference in the scores.

The authors 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. However, the profession of software engineering is still young...and Rome was not built in a day.

## References

[1] Alain April and Claude Y. Laporte, *Software Quality Assurance – Basic Concepts*. Hermes Publishing, 2009, (in French), 400 pp. {ISBN: 978-2-7462-3147-4}

[2] Robert Charette, Why software fails. *IEEE Spectrum*, pp. 42-49, September 2005.

[3] (ISO 2011) ISO/IEC TR 29110-5-1-2:2011 – Software Engineering - Lifecycle Profiles for Very Small Entities (VSEs) - Part 5-1-2: Management and engineering guide - Generic profile group: Basic profile, International Organization for Standardization/International Electrotechnical Commission: Geneva, Switzerland. Available at no cost from ISO: http://standards.iso.org/ittf/PubliclyAvailableStandards/c051153_ISO_IEC_TR_29110-5-1_2011.zip

[4] (ISO 2012) ISO/IEC TR 29110-5-1-1:2012 – Software Engineering - Lifecycle Profiles for Very Small Entities (VSEs) - Part 5-1-1: Management and engineering guide - Generic profile group: Entry profile, International Organization for Standardization/International Electrotechnical Commission: Geneva, Switzerland. Available at no cost from ISO: http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html

[5] Claude Y. Laporte and Alain April, *Software Quality Assurance – Advanced Concepts*, Hermes Publishing, 2009, (in French), 386 pp. {ISBN: 978-2-7462-3222-8}

[6] Claude Y. Laporte and Alain April, *Software Quality Assuranc*e, John Wiley and Sons, 2013. {ISBN: 978 1 1185 0182 5}

[7] Claude Y Laporte, Simon Alexandre and Rory O'Connor, "A Software Engineering Lifecycle Standard for Very Small Enterprises," in *O'Connor, R., et al. (eds.), Proceedings of EuroSPI. CCIS*, Springer, Heidelberg, vol. 16, pp. 129-141, 2008.

[8] Claude Y Laporte and Edgardo Palza Vargas, "The Development of International Standards to facilitate Process Improvements for Very Small Enterprises," in *Software Process Improvement and Management: Approaches and Tools for Practical Development*, IGI Global Publisher, pp. 34-61, 2012. Available from: http://profs.etsmtl.ca/claporte/Publications/Publications/IGI_Chapter_SPI_in_VSEs.pdf.

[9] Chris Fuhrman, Roger Champagne and Alain April, "Integrating Tools and Frameworks in Undergraduate Software Engineering Curriculum," in 34th International Conference on Software Engineering (ICSE), Zurich, Switzerland, pp. 1195-1204, 2012.

[10] Robert Dupuis, Roger Champagne, Alain April and Normand Séguin, "Experiments with adding to the experience that can be acquired from software courses," in Proceedings of the Seventh International Conference on the Quality of Information and Communications Technology (QUATIC 2010), September 29 - October 2, Porto, Portugal, pp. 1-6, 2010.

[11] Government of Canada – SME Financing Data Initiative, "Small business financing profiles," December 2011. http://www.ic.gc.ca/eic/site/061.nsf/eng/h_rd01200.html

[12] Claude Y Laporte, Normand Séguin and Gisele Villas Boas, Seizing the benefits of software and systems engineering standards, ISO Focus+, International Organization for Standardization, February 2013, pp 32-36.

## Biographies

**Claude Y. Laporte** is a professor at the École de technologie supérieure (ÉTS), a 7000-student engineering school, where he teaches graduate and undergraduate courses in software engineering. His research interests include software process improvement in small and very small entities, and software quality assurance. He earned a Master's degree in physics from the Université de Montréal and a Master's degree from Polytechnique Montréal. He is the project editor of the ISO/IEC-JTC1 SC7 working group tasked with developing ISO/IEC 29110 standards and guidelines for very small entities. 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). Professor Laporte can be reached at the ÉTS, Department of Software and IT Engineering, 1100 Notre-Dame St. W., Montréal, Québec, Canada, H3C 1K3, or by e-mail at claude.laporte@etsmtl.ca.

**Alain April** is a professor of software engineering at the École de technologie supérieure (ÉTS) and director of the Software Engineering Lab (GELOG). 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 25 years. Professor April contributed to ISO 9126 (part 3) in the section on the internal measurement of software, and is the associate editor of the software maintenance and software quality chapters of the SWEBOK. Professor April can be reached by e-mail at alain.april@etsmtl.ca.