IEEE Standards University - E-Magazine – November 2015

International Software Engineering Standards Applied in Undergraduate and Graduate Software Quality Assurance Courses[1]

Claude Y. Laporte, Eng., Ph.D.

École de technologie supérieure (ÉTS)

1100 Notre-Dame Street West, Montréal

Québec, Canada, H3C 1K3

claude.laporte@etsmtl.ca

Abstract

At the École de technologie supérieure, an 8000-student engineering school, software quality assurance (SQA) is taught in the undergraduate and graduate software engineering curricula. The course includes a 10-week project in which teams of students apply the SQA practices taught in class in a software development project. A new international software standard, ISO/IEC 29110, has been used in the team-projects. This standard as well as a set of management and engineering guides have been developed specifically for very small projects and organizations. Throughout the 10-week project, student-teams collect measures, such as the number of defects and the rework effort, and the performance of each team is analyzed. This analysis allows discussion to take place on the impacts of SQA practices as a way to support the development of quality software on time and within budget.

INTRODUCTION

Systems and software are growing larger and more complex every year. For example, mainstream cars have about 20-30 million lines of code and, top-of-the-line cars contain up to 100 million lines of code (Fleming 2014). According to Humphrey, it has been found that developers involuntarily inject about 100 defects into every 1,000 lines of the code they write (Humphrey 2005). Defects are not injected just in code, defects are unfortunately also injected, amongst others, in the requirements document, the architecture. There is a tremendous challenge to detect and correct the defects, especially defects of software critical components such as the braking system, before consumers use the cars. Therefore, a signification portion of the software development budget is allocated to the detection and correction of these defects. As reported by Charette, software specialists spend about 40 to 50 percent of their time on avoidable rework (Charette 2005). Software quality assurance (SQA) provides many software engineering practices needed to produce quality at the level of world class organizations having a defect escape rate of about 1 defect per 1,000 lines of code (Nolan et al 2015).

The École de technologie supérieure (ÉTS), an 8000-student engineering school of Montréal, began offering its graduate software engineering program in 1997 and its undergraduate software engineering program in 2001. The professor who designed the SQA courses has a combined industrial experience of more than 20 years, mainly in the defense, and transportation sectors. The aim of the SQA courses, which are mandatory in the ÉTS undergraduate and graduate software engineering curricula, is to ensure that software engineering students are aware of the importance of SQA, and that they understand and are able to apply SQA practices in a wide range of organizations (e.g. size and business domain). This also includes hands-on knowledge of the key ISO and IEEE standards, as well as how to use SQA tools. The courses allow students to apply a wide range of SQA practices throughout a software development cycle in a software development project simulating a mature industrial environment.

---

[1] http://www.standardsuniversity.org/e-magazine/november-2015/international-software-engineering-standards-applied-in-undergraduate-and-graduate-software-quality-assurance-courses/

Software development companies of the Montréal area, where the ÉTS is located, were surveyed. As illustrated in Table 1, and it was found that close to 80% of software development companies have fewer than 25 employees. The survey also showed that over 50% of companies have fewer than 10 employees.

Table 1: Size of software development companies in the Montreal area (translated form Gauthier 2004)

Size (employees)	Software Enterprises		Jobs	
	Number	%	Number	%
1 to 25	540	78%	5,105	29%
26 to 100	127	18%	6,221	36%
over 100	26	4%	6,056	35%
TOTAL	**693**	**100%**	**17,382**	**100%**

The large percentage of small organizations is not unique to the Montréal area. In Europe for instance, over 92% of enterprises, called micro-enterprises, have up to 9 employees and another 6.5% have between 10 and 49 employees. Micro enterprises account for 70% to 90% of enterprises in *the Organisation for Economic Co-operation and Development* countries and about 57% in USA (OECD 2005). The SQA courses of ÉTS have been designed having in mind that a high number of our graduates will work in micro, small and medium enterprises or in small and medium scale projects of large organizations.

SOFTWARE QUALITY ASSURANCE COURSES

The SQA undergraduate and graduate courses are composed of thirteen 3-hour lectures as illustrated in table 2[2]. The graduate and undergraduate courses are quite similar since they are targeted at students that have not taken an SQA course before. Each lecture topic is illustrated with industrial examples, international or professional standards, and weekly reading assignments (in French (April 2011) and (Laporte 2011), in English (Laporte 2016)). To ensure that students grasp the importance of SQA activities, the concept of business model (e.g. the risks associated to a business domain) and the cost of quality (e.g. prevention, evaluation, rework effort) are stressed throughout the course.

Table 2: List of SQA courses topics

Lecture	Course title
1	Introduction (Business models)
2	Quality culture (Cost of Quality, IEEE Code of ethics for software engineer)
3	Quality requirements
4	Standards and models
5	Software reviews
6	Software audit
7	Verification and validation
8	Configuration management
9	Policies, processes, and procedures
10	Measurement
11	Risk management
12	Management of suppliers and contracts
13	Software quality assurance plan

[2] The topics are described in more details in (Laporte and April 2013b)

Many standards are presented in the SQA course. As an example, the IEEE-1028 (IEEE 2008) is used to cover the reviews and audit topics, the IEEE-1012 (IEEE 2012) is used to cover the verification and validation topic, and the IEEE standard for software quality processes, IEEE-730 (IEEE 2014), is used to cover many topics of the SQA courses.

The quality requirements topic of the SQA courses is composed of 3 subjects: models of software quality, definition of software quality requirements and traceability of requirements in the software life cycle. The ISO/IEC 25010 (ISO 2011a) standard defines two quality models: quality in use and product quality. As defined in ISO 25010, quality in use is the degree to which a product or system can be used by specific users to meet their needs to achieve specific goals with effectiveness, efficiency, freedom from risk and satisfaction in specific contexts of use. In ISO 25010, the product quality model categorizes product quality properties into eight characteristics (functional suitability, reliability, performance efficiency, usability, security, compatibility, maintainability, portability). Each characteristic is composed of a set of related subcharacteristics. As an example, the reliability characteristic, which is defined as the degree to which a system, product or component performs specified functions under specified conditions for a specified period of time, is composed of the following four subcharacteristics: maturity, availability, fault tolerance and recoverability.

Students that do not use a software development framework, such as ISO/IEC 29110 presented in the next section, are often amazed that their own project data may reveal a percentage of rework of 50%, and sometimes even up to 70%. Students of the SQA courses are required to continuously measure the cost of rework in their team projects. They are also required to analyze their data and draw conclusions about the cost/benefit of SQA practices.

Overview of ISO/IEC 29110

ISO/IEC 29110 has been originally developed for a vast majority of very small entities (VSEs) that do not develop critical systems or critical software (Laporte et al 2008). A VSE is defined, in ISO/IEC 29110, as an enterprise, an organization, a department or a project having up to 25 people.

ISO/IEC 29110 provides VSEs with a four-step road map or also called 'Profiles'; the four profiles of ISO/IEC 29110 are: Entry, Basic, Intermediate and Advanced (ISO 2011b). VSEs targeted by the Entry profile are VSEs working on small projects (e.g. at most six person-months effort) and for start-up VSEs. The Basic profile describes development practices of a single application by a single project team of a VSE. The Intermediate profile is targeted at VSEs developing more than one project with more than one team. The Advanced profile is target to VSEs that want to sustain and grow as an independent competitive business.

At the request of the ISO working group mandated to develop ISO/IEC 29110, technical reports and guides are available at no cost from ISO. The Management and Engineering Guides, the most valuable documents for VSEs, have been translated in French, in Portuguese, Czech and in Spanish. Japan has translated and adopted ISO/IEC 29110 as a Japanese national standard and a German version should be part of the catalogue of the German standard organization DIN[3]. The reader who would like to read more about the standards and guides is invited to consult the articles publicly available on the public web site of the ISO/IEC 29110[4].

Figure 1 illustrates the two processes of the Basic profile of ISO/IEC 29110, as described in the Management and Engineering guide (ISO 2011c), for VSEs developing software: the Project Management (PM) process and the Software Implementation (SI) process. Each process is composed of a set of activities and each activity is composed of a set of tasks.

[3] Deutsches Institut für Normung

[4] http://profs.etsmtl.ca/claporte/English/VSE/index.html

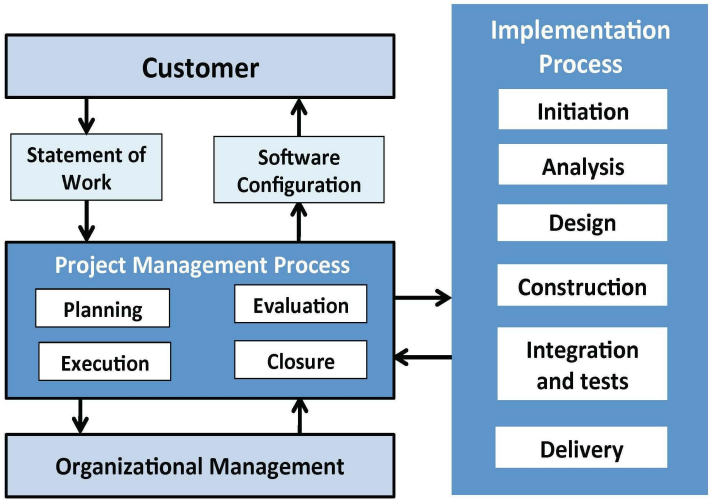

Figure 1. Project Management and Software Engineering Processes of ISO/IEC 29110 (Laporte et al 2013)[5]

The ISO working group mandated to develop ISO/IEC 29110 decided to include a project management process since it is a weakness of many VSEs and their financial success depends on successful project completion within schedule and on budget, as well as on making a profit. The other process of ISO/IEC 29110 is the process, titled software implementation process, dedicated to the development of a software product and its documentation.

For illustration purposes, one task of the ISO/IEC 29110 Project Planning activity is listed in Table 3. On the left side of the table are listed the roles involved in a task. The Project Manager (PM) and the Customer (CUS) are involved in this task. On the right side on the table, we listed the product required as an input to perform a task as well as the products produced by a task. All tasks are described using this format in the management and engineering guides.

Table 3. Example of 1 task of the Project Planning Activity (ISO 2011b)

Role	Task List	Input Products	Output Products
PM CUS	PM.1.2 Define with the Customer the *Delivery Instructions* of each one of the *Deliverables* specified in the *Statement of Work.*	Statement of Work *[reviewed]*	Project Plan Delivery Instructions

Deployment Packages to facilitate the implementation of ISO/IEC 29110

A novel approach taken to assist VSEs with the deployment of ISO/IEC 29110 and to provide guidance on the actual implementation of the Management and Engineering Guides in VSEs, a series of Deployment Packages (DPs) have been developed to define guidelines and explain in more detail the processes defined in the ISO/IEC 29110 profiles. The elements of a typical DP are: description of processes, activities, tasks, steps, roles, products, templates, checklists,

[5] For universities teaching systems engineering, a set of ISO/IEC 29110 systems engineering management and engineering guides, similar to the software engineering guides, have been published by ISO (ISO 2014, 2015, Laporte 2013a, Laporte 2104).

examples, references and mapping to standards and models, and a list of tools. The mapping shows that a deployment package has explicit links to standards, such as ISO/IEC/IEEE 12207 (ISO 2015), or models, such as the CMMI® for Development (SEI 2010). By implementing a DP, a VSE can see its concrete step to achieve or demonstrate coverage to ISO/IEC 29110.

DPs were designed such that a VSE can implement its content, without having to implement the complete framework, i.e. of the management and engineering guide, at the same time. For the Basic profile, as illustrated in figure 2, a set of nine DPs has been developed and are freely available[6].

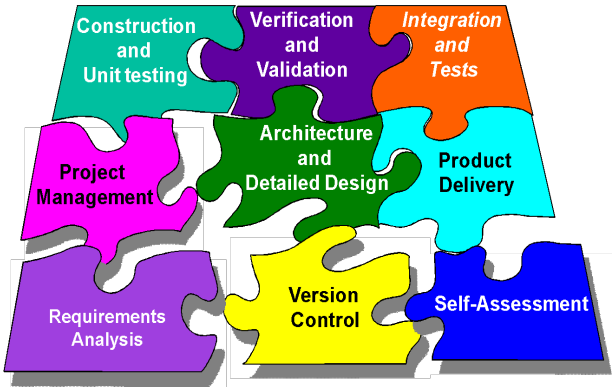

Figure 2. Deployment Packages to support the Software Basic Profile (Laporte 2008)

In the next sections, we describe how ISO/IEC 29110 standard was used by undergraduate and graduate students in implementing software engineering practices in a team-project.

Laboratory sessions and the student team-project

The first two laboratory sessions cover the IEEE code of ethics and business models (Iberle 2002). The business models describe different domains (e.g. custom systems written on contract, commercial and mass-market firmware) where software engineering practices are selected in a development process using a set of criteria such as the criticality of the software (i.e. the potential of harming a user when a software fails), and the cost of correcting defects once the software is in operation. Students are asked to select software engineering practices, such as the SQA practices presented in class, which are adequate for a specific business model.

After completing the sessions on the code of ethics and the business models, students embark on a project in teams of four students for a period of ten weeks where they must apply the SQA practices presented in the course, using the Basic profile of ISO/IEC 29110 as the framework for their software development project. Professors of the SQA courses randomly select the members of each team, 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. The professor plays the role of the president of the Acme manufacturing company and the student teams play the role of the software developers of Acme. At the beginning of the project, the president gives to the teams a SOW describing the functionalities to be developed. To reflect the reality of any organization, a few 'defects' have been intentionally inserted in the SOW. As an example, a SOW listed the functionalities of a new model of a washing machine that the president of Acme wants to produce. As an example, four washing cycles were described (e.g. soaking time, washing time, water temperature). In the description of one washing cycle, the water temperature was given in Fahrenheit (F) instead of Celsius (C). During that project, once the architecture was finalized, the president came to the developers

[6] http://profs.logti.etsmtl.ca/claporte/English/VSE/index.html

with a change request impacting a few documents of the project already delivered by the teams (e.g. project plan, specification, architecture). At another stage of the project, teams were given the high-level schematic (black-box) of the hardware of the new washing machine (e.g. microcontroller, display, sensors, actuators). Students were required to write code components that would turn on/off actuators and read data from digital sensors. To simulate a real development environment where mistakes are made, one hardware component had been 'omitted' in the schematic. This 'omission' required the students to initiate a second change request. From one semester to another, only the SOW has to be changed. As an example, a SOW for the development of software for a rice cooker and a crockpot (i.e. a slow cooker) were provided to student teams.

The course website lists the objectives and deliverables for each of the ten-week 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 types of reviews they have to perform (e.g. desk check, walkthrough) and the forms for registering defects, they detected during reviews.

During the planning phase of the project, the students in a team must share the following roles, as defined in ISO/IEC 29110: analyst, designer, programmer, technical lead, and project manager. At the beginning of the project, the four-team members must also complete and sign a 'contract' that specifies the roles of each participant, the deliverables of each team member, the expectations of each participant, and the operating rules which they agree to respect.

Teams must estimate the effort that will be needed by each member to carry out the activities and deliverables required by ISO/IEC 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.

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 tool, a version control tool, and an issue tracking tool.

Table 4 describes the 6 parts of the SQA student team-project that map to the management and engineering guide of ISO/IEC 29110. The 6 parts of the project are synchronized with the weekly lectures and reading assignments.

Table 4: The SQA team-project (adapted from Laporte and April 2013b)

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 (review) 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]*

Part 2 - Analysis and Documentation of Requirements

Objectives

- Objective 1: Perform the "Software requirements analysis" activity of ISO 29110;
- Objective 2: Perform a walkthrough (review) 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 (audit performed by teaching assistant)
3. Anomaly registration form
4. Validation results
5. Software user documentation *[preliminary]*
6. Measurement spreadsheet *[verified, baselined]*

Part 3 - Software architecture and detailed design

Objectives

- Objective 1: Perform the "Create the architecture and the detailed design" activity of ISO 29110;
- 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]*

Part 4 - Software Construction

Objectives

- Objective 1: Perform the '"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 customer]
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 collected

Part 5 - Software Integration and Tests

Objective

- Perform the "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)

Part 6 - Product Delivery and Project Completion

Objectives

- Objective 1: Perform the "Product delivery" activity;
- Objective 2: Conduct a lessons learned 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 customer]*
5. Software configuration *[accepted]*
6. Measurement spreadsheet *[verified, baselined]*
7. Information repository [*updated]*
8. Report on lessons learned

As described in ISO/IEC 29110, a traceability matrix is developed to connect the requirements, to the architecture, to the software components and to the tests. One advantage of a traceability matrix is the rapid identification of the software components impacted when requirements are modified, added, or deleted during a project. A fragment of a traceability matrix is presented in Table 5.

Table 5: Traceability Matrix

Traceability Matrix

Date (yy-mm-dd): _____

Title of project: _____

	Name (Print)	Signature	Date (yy-mm-dd)
Verified by:			
Approved by:			

Identification Number	Text of the need	Text of the requirement	Verification method	Title or ID of Use Case	Title or ID of Code Module	Title or ID of test Procedure	Verification Date	Person who performed the verification	Result of verification

In addition to the documents required by ISO/IEC 29110, students have to produce a lessons learned report and an analysis of the metrics collected. This report captures, from their point of view, what went well, what could have been done better and what surprised them during the 10-week project.

CONCLUSION

Many changes have been made to the SQA courses since they were initially set up over 10 years ago. One main objective was to get undergraduate and graduate students not only to learn from our textbooks but to apply SQA practices in a team-project, simulating a mature industrial environment, using an appropriate framework such as the management and engineering guide of ISO/IEC 29110. The SQA lectures and laboratory sessions provide a solid foundation for software engineers and software developers, even though SQA is still perceived as a low priority by many software development organizations.

Since the SQA courses cover the concept of the cost of quality, students are aware of the importance of using adequate prevention and evaluation practices, both to reduce the number of defects in the software, and to estimate the extra effort and time needed to correct defects (i.e. rework) introduced as the work progresses. With the help of the business models taught in class, students are in a better position to select a set of SQA practices that are adequate in a specific domain.

At the beginning of the SQA course, testing was often perceived by many students as the main technique to get quality 'built-in' software. A few weeks after the beginning of the team-project, it is very stimulating and rewarding for a professor to learn that students are applying some practices presented in class in other courses. Some students even say that they easily see where SQA practices could be applied, almost the next day, in their work environment. A few software engineers working in industry mentioned that their managers should also attend the SQA course.

When peer reviews are presented in class, an exercise is conducted on a short document seeded with many defects. All students are given the same time to review the document and report defects detected. Then students are asked to count the number of defects detected. Typically, there is a range of 10 between the students who detect the lower number of defects versus the students who detects the highest number of defects. As we walk through the document and pinpoint, on by one, the defects, it is an 'intellectually for significant event' for students when they 'discover' many defects which were right in front of them since the beginning of the exercise. Many of them are so surprised about their own detection performance that, when asked if they should conduct peer reviews, in documents such as the requirement specifications, the answer is a vibrant 'Yes' !

Since the SQA team-project involves the use of a wide spectrum of software engineering practices, students become aware, for the first time, how practices taught, often in isolation (e.g. in silo), in other courses (e.g. requirements course, testing course, project management course) are used together by a team in a development project. Some students even mentioned that it was the first time that they understand the usefulness of the software engineering practices and how these practices interact together in a team-project.

The software engineering practices of the SQA courses of ÉTS should help organizations of all sizes in getting the low defect escape rates of world class organizations, such as Rolls-Royce, which has achieved an overall defect removal effectiveness of 90% and a defect escape rate of 0.03 defects per 1000 lines of code (Nolan et al. 2015).

References

(April 2011) Alain April and Claude Y. Laporte, *Software Quality Assurance – Basic Concepts*. Hermes Publishing, 2011, (in French), 400 pp. {ISBN: 978-2-7462-3147-4}

(Canada 2011) 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

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

(Fleming 2014) Fleming, B., An Overview of Advances in Automotive Electronics, IEEE vehicular technology magazine, March 2014, pp. 4 -9

(Gauthier 2004) Gauthier R., Une force en mouvement, La Boule de Cristal, Centre de recherché informatique de Montréal, January 22, 2004. (in French)

(Humphrey 2005) Humphrey, W.S. PSP: A Self-Improvement Process for Software Engineers. Reading, MA: Addison-Wesley, 2005.

(Iberle 2002) Iberle K., But Will It Work For Me, Proceedings of the Pacific Northwest Software Quality Conference, p. 377-398, Portland, United States, 2002, on line: http://www.kiberle.com

(IEEE 2008) IEEE Std 1028-2008, IEEE Standard for Software Reviews and Audits, The Institute of Electrical and Electronics Engineers, Inc.

(IEEE 2012) IEEE Std 1012-2012, IEEE Standard for System and Software Verification and Validation, The Institute of Electrical and Electronics Engineers, Inc.

(IEEE 2014) IEEE Std 730-2014, Standard for Software Quality Assurance Processes, The Institute of Electrical and Electronics Engineers, Inc.

(ISO 2011a) ISO/IEC 25010:2011 Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models, International Organization for Standardization /International Electrotechnical Commission: Geneva, Switzerland. http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=35733

(ISO 2011b) ISO/IEC TR 29110-1:2011 – Software Engineering - Lifecycle Profiles for Very Small Entities (VSEs) - Part -1: Overview, International Organization for Standardization /International Electrotechnical Commission: Geneva, Switzerland. Available at no cost from ISO:
http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html

(ISO 2011c) 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

(ISO 2014) ISO/IEC TR 29110-5-6-2:2014 - Systems and software Engineering – Lifecycle Profiles for Very Small Entities (VSEs) – Systems engineering - 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/index.html

(ISO 2015) ISO/IEC TR 29110-5-6-1:2015 - Systems and software Engineering – Lifecycle Profiles for Very Small Entities (VSEs) – Systems engineering - 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

(ISO 2015) ISO/IEC/IEEE 12207:2015, Systems and software engineering – Software life cycle processes. International Organization for Standardization/International Electrotechnical Commission: Geneva, Switzerland.

(Laporte 2008) Laporte, C.Y., Alexandre, S, Renault, A., *Developing International Standards for Very Small Enterprises*, IEEE Computer, March 2008, Volume 41, number 3, pp. 82-85

(Laporte 2011) Laporte, C.Y., April, A., Software Quality Assurance – Advanced Concepts, Hermes Publishing, 2009 (in French), 386 pp. {ISBN: 978-2-7462-3222-8}

(Laporte 2013a) Laporte, C.Y., O'Connor, R., Fanmuy, G., International Systems and Software Engineering Standards for Very Small Entities, CrossTalk - The Journal of Defense Software Engineering, May/June 2013, Vol. 26, No 3, pp 28-33. Available at: http://www.etsmtl.ca/Professeurs/claporte/documents/publications/Crosstalk_May_2013_Laporte.pdf

(Laporte 2013b) Laporte, C.Y., April, A., Software Quality Assurance in an Undergraduate Software Engineering Program, Proc. 2013 Canadian Engineering Education Association (CEEA13) Conf., Montréal, June 17-20, 2013. Available at: http://www.etsmtl.ca/Professeurs/claporte/documents/publications/CEEA_Paper-109.pdf

(Laporte 2014) Laporte, C.Y., O'Connor, R., A Systems Process Lifecycle Standard for Very Small Entities: Development and Pilot Trials, 21th European Software Process Improvement Conference (Euro SPI 2014), CCIS 425, pp. 13-24, Springer-Verlag, Heidelberg, Luxembourg, June 25-27, 2014.

(Laporte 2016) Laporte, C. Y., April, A., Software Quality Assurance, John Wiley and Sons, 2016. {ISBN: 978 1 1185 0182 5}

(Nolan 2015) Nolan, A., Pickard, A., Russell, J., Schindel, W., When two is good company, but more is not a crowd, INCOSE International Symposium 2015, Seattle July 13-16, 2015.

(OECD 2005) SME and Entrepreneurship Outlook, 2005 Edition. Organization for Economic Co-Operation and Development, Paris, 2005.

(SEI 2010) Software Engineering Institute. (2010). CMMI for Development, Version 1.3, Pittsburgh PA: Carnegie Mellon University. CMU/SEI-2010-TR-033.

Biography

Claude Y. Laporte is a professor at the École de technologie supérieure (ÉTS), a 10,000-student engineering school, where he teaches graduate and undergraduate courses in software engineering. He has worked in defense and transportation organizations for over 20 years before joining ÉTS. He earned a Ph.D. from the Université de Bretagne Occidentale (France). He received an honorary doctorate in 2013. He is, since 2005, the Project Editor of the ISO/IEC 29110 standards and guides for very small entities developing systems or software. 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). He is the co-author of two French books on software quality assurance, published in 2011, and one English textbook, on the same topic, to be published by John Wiley and Sons in 2016.

Web site address:
http://profs.etsmtl.ca/c/English/index.html