C. Y. Laporte and M. Munoz, "Not Teaching Software Engineering Standards to Future Software Engineers-Malpractice ?" in *Computer*, vol. 54, no. 5, pp. 81-88, May 2021, doi: 10.1109/MC.2021.3064438

Not teaching software engineering standards to future software engineers – Malpractice ?

Claude Y Laporte and Mirna Muñoz

Abstract

Software engineering standards are essential sources of codified knowledge for all software engineers. Could the professors, that are not teaching software engineering standards to software engineering students, be accused of malpractice?

During a roundtable at the New Delhi meeting of the ISO committee responsible for software and systems engineering standards[1], a member of the audience asked if software engineering professors that do not teach software engineering standards to software engineering students could be accused of malpractice. By malpractice we mean any inappropriate, wrong, illegal or careless actions that a professional does while working.

As we well know, engineering domains such as mechanical, chemical, electrical or engineering are based on the laws of nature as discovered by scientists. Figure 1 illustrates some of the many laws of nature. Unfortunately, software engineering, unlike other engineering disciplines, is not based on the laws of nature.

Would it be conceivable that a professor teaching electrical engineering would not teach Ohm's law or a professor teaching chemistry would not teach the Boyle-Mariotte's law? By teaching we mean not only showing a few slides but requiring students to solve some problems and perform a lab experiment about this law.

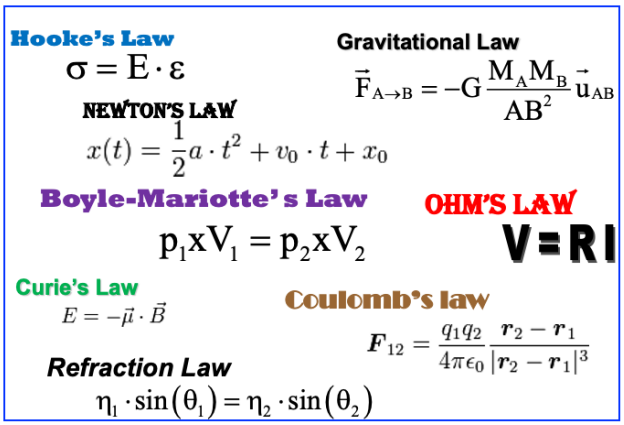

Figure 1. Laws of Nature that support engineering disciplines [1].

The development of software is based only on the laws of logic and mathematics. Software Engineering, like other engineering disciplines, is based on the use of well-defined practices for ensuring the quality of its products. In Software Engineering, there are several standards which are

[1] ISO/IEC JTC1 SC7 - Software and systems engineering, https://www.iso.org/committee/45086.html

actually guides for management practices. A rigorous process is the framework for the way standards are developed and approved including, among others, international ISO standards and standards from professional organizations such as IEEE.

As written by Moore a few years ago [2] "Standards are important, not because they represent best practices, but because they represent good enough practices. Courts generally view the application of standards as important evidence that engineers perform their work with appropriate diligence and responsibility. If accused of negligence or reckless conduct, an engineer can cite the standards used when he or she conducted the work to demonstrate that it was performed in accordance with codified professional practices".

If an engineer could be accused of negligence, why professors teaching to future software engineers, that ignore or do not teach software engineering standards could not be accused of malpractice?

SAD OBSERVATIONS ABOUT SOFTWARE ENGINEERS

In a recent article of the Impact column of IEEE Software [3], the authors wrote that «We had been hoping that would follow the same trajectory as its older established cousins, such as civil engineering, but we have seen no real evidence of this. »

The authors also wrote that we cannot call ourselves an engineering discipline unless we begin to systematically learn from our mistakes and, in software we seem to have an aversion to measurement. This is quite a paradox, since on one side software people are not learning from their mistakes and, on the other side, hundreds of experts around the world have been working since the 80's in documenting in standards knowledge gained from successful and failed software projects. The portfolio of software engineering standards now covers the full spectrum, i.e., from "cradle to grave", of software engineering.

Another paradox is the fact that those software engineering standards, that document codified knowledge and are publicly available are not used, or ignored, by a large number of professors that are mandated to transfer software engineering practices to their software engineering students. Consequently, those students will end up in software development organization with a large deficit of essential knowledge. Is this a case of negligence or malpractice?

WHY BOTHER WITH STANDARDS?

Standards are sources of codified knowledge and studies have demonstrated the benefits of them, such as product interoperability, increased productivity, market share gains, and improved interaction with stakeholders such as enterprises, government organizations and the public. Standards and associated technical documents could be considered as a form of technology transfer and, if the right standards are selected and used correctly, they should have an economic impact in an organization [4].

The contribution of standards to the economy of a few countries is illustrated in Table 1.

Table 1. Comparative contribution of standards to national economies (adapted from [5])

	Germany (DIN)	UK (DTI)	Standards Council of Canada	Australia Standards	France (AFNOR)
Period subject to analysis	1961-1990	1948-2001	1981-2004	1962-2004	1950-2007
Growth rate of GDP (%)	3.3	2.5	2.7	3.6	3.4
Contribution to growth of GDP (%)	27.3	11.0	9.0	21.8	23.8
Impact in % points on GDP growth	0.9	0.3	0.2	0.8	0.8

Advantages or benefits as well as disadvantages or costs have been reported regarding the use of voluntary standards. Table 2 lists a few of the advantages and disadvantages reported.

Table 2. Advantages and disadvantages of voluntary standards reported (adapted from [5],[6])

Advantages or benefits	Disadvantages or costs
- Promote innovation - Improve efficiency of an organization - Increase competitiveness - Facilitate the access to a wider market - Clarify the rules of a market - Improve quality of products and services - Promote improvement of processes - Facilitate partnerships - Improve the image, credibility of organizations - Promote a uniform terminology - Regularly updated - Facilitate the selection of suppliers and partners - Facilitate access to recognize knowledge - Facilitate access to investments and financing	- Difficult to understand - Cost of acquire standards - Cost of standard implementation - Cost of certification - Require outside expertise to implement them - Conflicting standards - High number of standards available - Describe only ‘what to be done’ not ‘how to do it’ - Insufficient guidance to select and apply them - Slow evolution of standard may impede innovation - Difficult and costly to apply in small organizations - Difficult to demonstrate ‘savings’ - Many producers of standards - Perception that standards add unnecessary bureaucracy to an organization - Language barrier for users that are not proficient in English

In addition to known benefits of standards, five major lessons emerged from a French study [5]:

- Company value enhancement: The knowledge capital contributed by corporate involvement in standardization work represents true value.
- Innovation: Standardization promotes the dissemination of innovation. It emphasizes a product’s advantages and constitutes a product selection tool.
- Transparency and ethics: Standards contribute to better compliance with the rules of competition. By establishing the rules of the game, standards make it easier to eliminate players who fail to comply.
- International: By promoting the development of international exchanges, standardization provides companies with a genuine passport for exporting their products.
- Product and service quality: Standardization gives companies a great degree of control over safety-related problems and provides a genuine guarantee of quality.

C. Y. Laporte and M. Munoz, "Not Teaching Software Engineering Standards to Future Software Engineers-Malpractice ?" in *Computer*, vol. 54, no. 5, pp. 81-88, May 2021, doi: 10.1109/MC.2021.3064438

QUALITY AND PRODUCTIVITY ISSUES IN SOFTWARE DEVELOPMENT

The recent cost of quality report of the Consortium for Information & Software Quality™ for the year 2020 [77] reported that the total Cost of Poor Software Quality in the US is $2.08 trillion. The report also revealed a 5-10X difference in performance between the top 10% and the bottom 10% of organizations sampled. In 2018, the Consortium reported that $500 billion was being spent on finding and fixing software bugs in the US. With the US code base growing at ~7% per year, and IT wages growing at ~3% year, then the amount spent in 2020 finding and fixing software bugs would be ~$607 billion.

In March 2020, the Standish Group released its report CHAOS 2020: Beyond Infinity. In that report they stated that only 35% of projects were fully successful with respect to time and budget, that 19% of projects will be cancelled before they get completed, and that 47% of projects are challenged (i.e., over budget, behind schedule, low quality deliverables) [8].

According to Charette, “Studies have shown that software specialists spend about 40 to 50 percent of their time on avoidable rework rather than on what they call value-added work, which is basically work that’s done right the first time. Once a piece of software makes it into the field, the cost of fixing an error can be 100 times as high as it would have been during the development stage” [8]. Measuring and reducing the percentage of avoidable rework should be one objective of most process improvement initiatives.

SOFTWARE ENGINEERING STANDARDS

Software Engineering standards should be a very important source of codified knowledge for academia that teaches the development and maintenance of software to future software engineers.

There is a large portfolio of Software Engineering standards that covers all activities of software development and management. For instance, standards provide descriptions of processes, activities, and tasks that are applied during the acquisition of a software system, product or service and during the supply, development, operation, maintenance and disposal of software products, a standard that covers configuration management, a series of software testing standards, a standard about risk management and one standard about software measurement.

Unfortunately, Software Engineering standards, have initially been developed by large organizations without having in mind smaller settings. Most small organizations do not have the expertise or the resources to participate to standard development. A large majority of enterprises worldwide are very small entities (VSEs), i.e., enterprises, organizations, departments or projects having up to 25 people [10]. 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 [11].

MALPRACTICE

Malpractice could be defined as any inappropriate, wrong, illegal or careless actions that a professional does while working. The Guide to the Software Engineering Body of Knowledge (SWEBOK Guide) and the IEEE/ACM Software Engineering Code of Ethics and Professional

Practice provide insight about the role of professors in teaching software engineering standards.

The SWEBOK Guide indicates that the benefits of software engineering standards are many and include improving software quality, helping avoid errors, protecting both software producers and users, increasing professional discipline, and helping technology transition [12]. One objective of the SWEBOK is to provide a foundation for curriculum development and for individual certification and licensing material. The SWEBOK provides an annex listing the relevant standards for each knowledge area. For instance, one main relevant standard of the software requirements knowledge area is "ISO/IEC/IEEE 29148:2011 Systems and Software Engineering—Life Cycle Processes—Requirements Engineering".

The SWEBOK provides this clause about professional liability:

> It is common for software engineers to be concerned with matters of professional liability. As an individual provides services to a client or employer, it is vital to adhere to standards and generally accepted practices, thereby protecting against allegations or proceedings of or related to malpractice, negligence, or incompetence.

The IEEE/ACM Software Engineering Code of Ethics and Professional Practice, intended as a standard for teaching and practicing software engineering, documents the ethical and professional obligations of software engineers [13]. The Code indicate that software engineers are those who contribute, by direct participation or by teaching, to the analysis, specification, design, development, certification, maintenance, and testing of software systems. The Code contains eight principles related to the behaviour of and decisions made by professional software engineers, including practitioners, educators, managers, supervisors and policy-

Principle 3 – PRODUCT

Software engineers shall ensure that their products and related modifications meet the highest professional standards possible. In particular, software engineers shall, as appropriate:

- 3.06. Work to follow professional standards, when available, that are most appropriate for the task at hand, departing from these only when ethically or technically justified.

Principle 8 – SELF

Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession. In particular, software engineers shall continually endeavor to:

- 8.05. Improve their knowledge of relevant standards and the law governing the software and related documents on which they work.

makers, as well as trainees and students of the profession. Principle 3 and 8 provide information about the knowledge and the use of standards.

Therefore, according to the SWEBOK Guide and the IEEE/ACM Software Engineering Code of Ethics and Professional Practice, professors that are not learning and teaching software engineering standards to software engineering students could be accused of malpractice.

C. Y. Laporte and M. Munoz, "Not Teaching Software Engineering Standards to Future Software Engineers-Malpractice ?" in *Computer*, vol. 54, no. 5, pp. 81-88, May 2021, doi: 10.1109/MC.2021.3064438

AN IMPOSSIBLE SCENARIO ?

A software engineer of 8-people company that develop computer-controlled valves for organizations such as pharmaceutical or chemical companies was supposed to conduct an inspection of the requirements that he documented. The contract with the customer indicated that software development shall be conducted using the IEEE software engineering standards. But the developer did not know the IEEE 1028 standard that describes types of software reviews with procedures required for the execution of each type [14].

The developer, according to the organizational software process, after he completed the documentation of the requirements, inspected, by himself, the requirements using the checklist provided by the software process.

After installing the new software, the computer-controlled valves malfunctioned and caused damages in the chemical plant. One technician of the customer had to be brought to the emergency service of a hospital. The supplier of the software was asked to immediately correct the defective software. The supplier did not want to correct the software unless the customer would pay an additional 50,000$. The customer decided to sue the supplier of the defective software.

At the court hearing, the lawyer of the customer requested an evidence showing that an inspection of the requirements had been performed. The supplier could not provide that proof. The lawyer then interrogated the develop and asked him to describe how he inspected the requirements. The lawyer provided the judge with a copy of the IEEE-1028 standard as an evidence. In the IEEE standard, the following note is added to the definition of inspection: "*Inspections are peer examinations led by impartial facilitators who are trained in inspection techniques".* It became evident that the developer performed an informal review instead of the inspection defined in the IEEE 1028 standard. If an inspection had been performed, the colleagues of the developer could have detected the defects. In addition, if an inspection had been performed, proofs of execution (e.g., list of participants, list of defects detected, decisions made, updated version of the requirements) could have been provided to the judge.

The judge decided that the supplier did not fulfil the requirements of the contract. The judge also blamed the developer of negligence for not performing an inspection of the requirements as described in the contract. The judge ordered the supplier to rework the software as described in the contract at no cost to the customer. The judge also asked the supplier of the software when the updated software will be delivered to the customer. The supplier indicated that it would take 2 weeks to rework the software, i.e., the complete development process will need to be executed since the defects in the requirements impact the test procedures, the test cases, the architecture and the code. Also, all integration tests would need to be executed, and the user manual would need modifications.

Outside the courthouse after the hearing, the president of the supplier fired the developer. The next day, the developer contacted a lawyer to launch a class-action lawsuit of negligence of the university he attended, on behalf on the hundred's software engineering students at his former university. The developer also broadcasted information about the class-action lawsuit on his social networks. The next day, that news became viral all over the world amongst software engineering students, lawyers and universities that provide software engineering programs.

C. Y. Laporte and M. Munoz, "Not Teaching Software Engineering Standards to Future Software Engineers-Malpractice ?" in *Computer*, vol. 54, no. 5, pp. 81-88, May 2021, doi: 10.1109/MC.2021.3064438

TEACHING SOFTWARE ENGINEERING STANDARDS - A SUCCESS STORY

A series of standards and guides have been developed to help very small entities, i.e., entities having up to 25 people, in developing and maintaining software: the ISO/IEC 29110 series [15,16,17]. Universities of at least 21 countries are teaching the ISO/IEC 29110. In Thailand for instance, over 10 universities are teaching the ISO/IEC 29110.

In Mexico, with the financial support of the secretariat of economy of the state of Zacatecas, a six-step method has been developed to accelerate the implementation of ISO/IEC 29110 in VSEs, in software engineering courses and capstone projects, and in software development centers (SDCs) of universities [18].

SDCs are environments where students work in teams as a real organization to develop software for real customers either internal or external to the university. SDCs implemented a software development process using the ISO/IEC 29110. They were supported by a local research Centre which provided workshops to software engineering professors, support for the implementation and improvement of their software processes. The SDCs of 10 universities have obtained, like any software commercial organization, the certification to the ISO/IEC 29110 standard. VSEs that hired graduates of the SDCs were very satisfied with their new employees.

Three important elements have accelerated the adoption of the ISO/IEC 29110 series by universities and VSEs of many countries. First, a few ISO/IEC 29110 documents, such as the management and engineering guides, are freely available from ISO. Second, to lower the adoption barrier, a few ISO/IEC 29110 documents been translated into Czech, French, Portuguese, and Spanish and adopted as national standards by several countries. Third, teaching material and webinars are been developed to help academia in learning and teaching the ISO/IEC 29110 series.

The ISO/IEC 29110 framework is still young, but its use by VSEs and universities proves that the series original objectives, i.e., developing a series of standards and guides that can readily be implemented by commercial as well as public VSEs and can successfully be taught to software engineering students.

HELPING PROFESSORS IN TEACHING SOFTWARE STANDARDS

There are a few ways to help professors in teaching software engineering standards. First, professors must start by acquiring and study the standards selected for the courses under their responsibility. Then, professors must prepare teaching material (e.g., presentation material, exercises, projects). Software engineering students will not learn about a standard if professors spend only a few minutes in class and present them a few slides about a standard. To be of any use, students have not only to study a standard and do a few exercises but put it in practice in a software development project in an environment similar to industry [19][20].

C. Y. Laporte and M. Munoz, "Not Teaching Software Engineering Standards to Future Software Engineers-Malpractice ?" in *Computer*, vol. 54, no. 5, pp. 81-88, May 2021, doi: 10.1109/MC.2021.3064438

Unfortunately, professors are rarely rewarded for the quality of their courses. They are mostly rewarded by the number of papers published in journals and conferences and research money they bring to their universities. Professors participating to the development and implementation of software engineering standards should also be rewarded, since these activities are part of the knowledge creation and diffusion mandates of all universities.

SURVEY OF A FEW SOFTWARE ENGINEERING PROFESSORS

For this article, we launched a quick survey to 30 software engineering professors. We are aware that the answers provided may not reflect what is happening at many universities around the world.

Professors were asked to answer the following questions:

1. Why did they consider that there is a lack of teaching standards at universities?
2. What benefits can be obtained when teaching standards in universities?
3. What needs should be satisfied to enable professors to teach standards in universities?

Answers to question 1

The professors cited the following reasons: 1) they are difficult to teach; 2) they are expensive; 3) the time available to teach them and put them in practice is too short; 4) a lack of knowledge about their benefits; 5) a lack of knowledge of professors, 6) standards are perceived as boring topics; 6) standards are not included in the curricula of their universities.

Answers to question 2

The professors cited the following benefits: 1) students having worked with standards have been that they have been placed in quality departments in companies; 2) students develop software of higher quality, on time and within the resources available; 3) students develop a comprehension that standards could help them, and they are not their enemies that impose constraints on them; 4) students improve their performance (e.g. productivity, quality) when developing software; 5) students acquire discipline by covering all phases of development and give due importance to other disciplines and not just to software development; 6) students improve their ability to work as a team by distributing responsibilities and assuming commitments; 7) students are better prepared for their career as professionals.

Answers to question 3

The professors cited the following needs: 1) the need for professors to be trained in standards such that they could be able to teach them in an adequate way; 2) to allow them to modify the curricula to add enough time to teach standards; 3) to select those standards according to objectives of the program; 4) to have specific places to develop software projects; 5) to improve the collaboration of universities with the software industry.

Professors could also attend, as an observer, the ISO or IEEE working groups that develop or improve software engineering standards. After attending a few meetings, professors could then formally join a working group as a full member. Their participation to the development and implementation of a standard could also be opportunities to publish papers. In the last ten years,

about 200 papers have been written by researchers, mainly in academia, about the ISO/IEC 29110 series, illustrating the interest in this set of standards and guides by academia [21].

A few standards, like the IEEE-1028 standards about reviews [13] or the IEEE-828 standard about configuration management [14], could be used in more than one course, providing a deeper knowledge to students, by performing reviews and configuration management activities in a requirements course, in an architecture course and in a programming course. Finally, a one or two-semester capstone project would be an ideal way for the implementation of standards learned in previous courses by teams of students.

Unfortunately, most software engineering standards are not free. Moreover, they are very expensive from a student point of view and even for professors of many countries. For example, the cost of the IEEE-1028 standard [14] is about 160$. Therefore, universities, that provide a software engineering program, should provide a free or low-cost access to software engineering standards to their software engineering students, e.g., through a membership to a professional society like the IEEE Computer Society.

CONCLUSION

Since the Software Engineering discipline does not have as its foundation the laws of nature, not teaching Software Engineering standards, i.e., as sources of documented knowledge, to software engineering students should be considered as malpractice even if Software Engineering standards will not give a guarantee of achieving the quality, cost and schedule objectives.

To protect from malpractice professors that teach Software Engineering and that do not want to teach the Software Engineering standards pertinent to their courses, universities should provide them the opportunity to be trained in standards and their benefits. Finally, universities should challenge them to improve their collaboration with industry so that software can be developed by their students in an environment similar to industry as described in [18].

The correct use of appropriate Software Engineering standards, by software engineering students, should increase the level of their confidence of achieving the objectives of a project, i.e., in delivering to the customer all the functionalities, with all the quality characteristics within budget and schedule.

If, software engineering professors do not teach software engineering standards to their students, remembering that a software engineering standard is documented knowledge gained from thousands of successful and failed projects, “We could be in for another “lost decade” if we plan to rush at new technology, forgetting everything we learned about decent software engineering” [3].

ACKNOWLEDGMENTS

We thank the 30 professors from Ecuador, Mexico, Panama, Spain and the USA that took the time to answer our survey.

C. Y. Laporte and M. Munoz, "Not Teaching Software Engineering Standards to Future Software Engineers-Malpractice ?" in *Computer*, vol. 54, no. 5, pp. 81-88, May 2021, doi: 10.1109/MC.2021.3064438

NOTES

Readers can learn more about ISO/IEC 29110 at http://profs.logti.etsmtl.ca/claporte/English/VSE/index.html. Several management and engineering ISO/IEC 29110 guides are available for free from the ISO at http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html

The SWEBOK Guide is also available as a free ISO technical report, “ISO/IEC TR 19759:2005 Software Engineering—Guide to the Software Engineering Body of Knowledge (SWEBOK)” and available at: http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html

REFERENCES

1. Laporte, C.Y., April, A., Software Quality Assurance, John Wiley and Sons Inc., ISBN: 978 1 1185 0182 5, 2018. https://www.wiley.com/en-ca/Software+Quality+Assurance-p-9781118501825
2. J. W. Moore, "An integrated collection of software engineering standards," in IEEE Software, vol. 16, no. 6, pp. 51-57, Nov.-Dec. 1999, doi: 10.1109/52.805473
3. M. van Genuchten and L. Hatton, "Ten Years of "Impact" Columns—The Good, the Bad, and the Ugly," in *IEEE Software*, vol. 36, no. 6, pp. 57-60, Nov.-Dec. 2019, doi: 10.1109/MS.2019.2932495.
4. Laporte, C. Y., Chevalier, F. (2016). An Innovative Approach to the Development of Project Management Processes for Small-Scale Projects in a Large Engineering Company. In K. Jakobs (Ed.), *Effective Standardization Management in Corporate Settings* (pp. 123-160). Hershey, PA: Business Science Reference. doi:10.4018/978-1-4666-9737-9.ch007
5. Miotti, H. (2009, June). The economic impact of standardization technological change. Standards growth in France, AFNOR.
6. Land, S. K. (1997, June 1-6). Results of the IEEE Survey of Software Engineering Standards Users. Software Engineering Standards Symposium and Forum, 1997. Emerging International Standards. ISESS 97, Walnut Creek, CA, pp. 242 – 270.
7. H. Krasner, *The Cost of Poor Software Quality in the US: A 2020 Report,* Consortium for Information & Software Quality™ (CISQ™)
8. CHAOS 2020 Beyond Infinity, Standish Group Report, The Standish Group International, Inc., https://www.standishgroup.com/news/49
9. R. N. Charette, "Why software fails [software failure]," in *IEEE Spectrum*, vol. 42, no. 9, pp. 42-49, Sept. 2005, doi: 10.1109/MSPEC.2005.1502528.
10. C. Y. Laporte and R. V. O'Connor, "Systems and Software Engineering Standards for Very Small Entities: Accomplishments and Overview," in Computer, vol. 49, no. 8, pp. 84-87, Aug. 2016, doi: 10.1109/MC.2016.242.
11. Moll, R. (2013, February). Being prepared – A bird’s eye view of SMEs and risk management. ISO Focus +, Geneva, Switzerland: International Organization for Standardization.
12. P. Bourque, R. E. Fairley, eds., Guide to the Software Engineering Body of Knowledge: Version 3.0 (SWEBOK Guide). IEEE Computer Society, 2014.
13. IEEE-CS-1999. Software Engineering Code of Ethics and Professional Practice, IEEE-CS/ACM, 1999, https://www.computer.org/education/code-of-ethics

14. "IEEE Standard for Software Reviews and Audits," in IEEE Std 1028-2008, vol., no., pp.1-53, 15 Aug. 2008, doi: 10.1109/IEEESTD.2008.4601584
15. C.Y. Laporte, Jezreel Mejia Miranda, “Delivering Software and Systems Engineering Standards for Small Teams - Feedback from Very Small Entities, their customers, auditors and academia on ISO/IEC 29110.”, IEEE Computer, Vol. 53, Issue 8, August 2020, pp. 79-83
16. C. Y. Laporte, M. Munoz, J. Mejia Miranda and R. V. O’Connor, "Applying Software Engineering Standards in Very Small Entities: From Startups to Grownups," in *IEEE Software*, vol. 35, no. 1, pp. 99-103, January/February 2018, doi: 10.1109/MS.2017.4541041.
17. Muñoz, M., Mejia, J., Peña, A., Lara, G., & Laporte, C. Y. (2019). Transitioning international software engineering standards to academia: analyzing the results of the adoption of ISO/IEC 29110 in four Mexican universities. *Computer Standards & Interfaces*, *66*, 103340
18. Laporte, C.Y., April, A., Benchérif, K., *Teaching Software Quality Assurance in an Undergraduate Software Engineering Program*, Software Quality Professional Journal, ASQ, Vol. 9, Issue 3, 2007, p 4-10.
19. Laporte, C.Y., International Software Engineering Standards Applied in Undergraduate and Graduate Software Quality Assurance Courses, IEEE Standards University, IEEE Standards Education E-Magazine, November 2015, http://www.standardsuniversity.org/issue/november-2015/
20. X. Larrucea and B. Fernandez-Gauna, “A mapping study about the standard ISO/IEC29110,” Comput. Stand. Interfaces, vol. 65, pp. 159–166, July 2019. 10.1016/j.csi.2019.03.005.

ABOUT THE AUTHORS

Claude Y. Laporte is an adjunct professor of software engineering at École de technologie supérieure and the Lead Editor of the ISO/IEC 29110 series of standards and guides for very small entities that develop systems or software. He is the IEEE liaison to ISO/IEC JTC1 Sub Committee 7, the committee responsible for systems and software engineering within ISO. Contact him at claude.laporte@etsmtl.ca.

C. Y. Laporte and M. Munoz, "Not Teaching Software Engineering Standards to Future Software Engineers-Malpractice ?" in *Computer*, vol. 54, no. 5, pp. 81-88, May 2021, doi: 10.1109/MC.2021.3064438

Mirna Munoz is a professor of software engineering at Centro de Investigación en Matemáticas and co-editor of the new ISO/IEC 29110 Agile Software Development Guide[2]. Contact her at mirna.munoz@cimat.mx

[2] https://www.iso.org/standard/76639.html