

# Not Teaching Software Engineering Standards to Future Software Engineers—Malpractice?

**Claude Y. Laporte**, École de technologie supérieure

**Mirna Muñoz**, Centro de Investigación en Matemáticas

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

Digital Object Identifier 10.1109/MC.2021.3064438  
Date of current version: 7 May 2021

During a roundtable at the New Delhi meeting of the International Organization for Standardization (ISO) committee responsible for software and systems engineering standards (ISO/IEC JTC1 SC7), a member of the audience asked if software engineering professors who 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 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.

The development of software is based only on the laws of logic and mathematics. Software engineering, like other engineering disciplines, is

*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.*

Software engineering, like other engineering disciplines, is based on the use of well-defined practices for ensuring the quality of its products.

based on the use of well-defined practices for ensuring the quality of its products. In software engineering, there are several standards that are actually guides for management practices. A rigorous process is the framework for the way standards are developed and approved, including, among others, ISO standards and standards from professional organizations such as IEEE.

As written by Moore a few years ago,<sup>2</sup>

*Standards are important, not because they represent best practices, but because they represent good enough practices. Courts generally*

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

### SAD OBSERVATIONS ABOUT SOFTWARE ENGINEERS

In a recent “Impact” column in *IEEE Software*,<sup>3</sup> the authors wrote, “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 1980s to document in standards the knowledge gained from successful and failed software projects. The portfolio of software engineering standards now covers the full spectrum, that is, from the cradle to the grave, of software engineering.

Another paradox is the fact that software engineering standards documenting codified knowledge and publicly available are not used, or ignored, by a large number of professors who are mandated to transfer software engineering practices to their software engineering students. Consequently, those students will end up in a 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 a form of technology transfer, and, if the right standards are selected and used correctly, they should have an economic impact in an organization.<sup>4</sup> The contribution of standards to the economy of some countries is illustrated in Table 1.

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

<b>Hooke's Law</b> $\sigma = E \cdot \epsilon$	<b>Gravitational Law</b> $\vec{F}_{A \rightarrow B} = -G \frac{M_A M_B}{AB^2} \vec{u}_{AB}$
<b>Newton's Law</b> $x(t) = \frac{1}{2} a \cdot t^2 + v_0 \cdot t + x_0$	
<b>Boyle-Mariotte's Law</b> $p_1 x V_1 = p_2 x V_2$	<b>Ohm's Law</b> $V = RI$
<b>Curie's Law</b> $E = -\vec{\mu} \cdot \vec{B}$	<b>Coulomb's Law</b> $F_{12} = \frac{q_1 q_2}{4\pi\epsilon_0} \frac{r_2 - r_1}{ r_2 - r_1 ^3}$
<b>Refraction Law</b> $\eta_1 \cdot \sin(\theta_1) = \eta_2 \cdot \sin(\theta_2)$	

FIGURE 1. The laws of nature that support engineering disciplines.<sup>1</sup>

**TABLE 1.** A comparative contribution of standards to national economies.<sup>5</sup>

	Germany (DIN)	United Kingdom (DTI)	Standards Council of Canada	Australia Standards	France (AFNOR)
<b>Period subject to analysis</b>	1961–1990	1948–2001	1981–2004	1962–2004	1950–2007
<b>Growth rate of gross domestic product (GDP) (%)</b>	3.3	2.5	2.7	3.6	3.4
<b>Contribution to growth of GDP (%)</b>	27.3	11	9	21.8	23.8
<b>Impact in % points on GDP growth</b>	0.9	0.3	0.2	0.8	0.8

**TABLE 2.** The advantages and disadvantages of voluntary standards reported.<sup>5,6</sup>

Advantages or benefits	Disadvantages or costs
<ul style="list-style-type: none"> <li>• Promote innovation</li> <li>• Improve efficiency of an organization</li> <li>• Increase competitiveness</li> <li>• Facilitate access to a wider market</li> <li>• Clarify the rules of a market</li> <li>• Improve quality of products and services</li> <li>• Promote improvement of processes</li> <li>• Facilitate partnerships</li> <li>• Improve the image and credibility of organizations</li> <li>• Promote a uniform terminology</li> <li>• Regularly updated</li> <li>• Facilitate the selection of suppliers and partners</li> <li>• Facilitate access to recognize knowledge</li> <li>• Facilitate access to investments and financing</li> </ul>	<ul style="list-style-type: none"> <li>• Difficult to understand</li> <li>• Cost of acquiring standards</li> <li>• Cost of standard implementation</li> <li>• Cost of certification</li> <li>• Require outside expertise to implement them</li> <li>• Conflicting standards</li> <li>• High number of standards available</li> <li>• Describe only “what has to be done” not “how to do it”</li> <li>• Insufficient guidance to select and apply them</li> <li>• Slow evolution of standard may impede innovation</li> <li>• Difficult and costly to apply in small organizations</li> <li>• Difficult to demonstrate “savings”</li> <li>• Many producers of standards</li> <li>• Perception that standards add unnecessary bureaucracy to an organization</li> <li>• Language barrier for users that are not proficient in English</li> </ul>

In addition to the known benefits of standards, five major lessons emerged from a French study<sup>5</sup>:

- › *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.

## QUALITY AND PRODUCTIVITY ISSUES IN SOFTWARE DEVELOPMENT

The recent cost of quality report from the Consortium for Information & Software Quality for the year 2020<sup>7</sup>

**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.**

- › *Product and service quality*: Standardization gives companies a great degree of control over safety-related problems and provides a genuine guarantee of quality.

reported that the total cost of poor software quality in the United States is US\$2.08 trillion. The report also revealed a 5–10× difference in performance between the top 10% and the bottom 10% of organizations sampled.

In 2018, the consortium reported that in the United States, US\$500 billion was being spent on finding and fixing software bugs. With the U.S. code base growing at ~7% per year and IT wages growing at ~3% year, the amount spent finding and fixing software bugs in 2020 would be ~US\$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, 19% of projects will be cancelled before completion, and 47% of projects are challenged (that is, over budget, behind schedule, or have low quality deliverables).<sup>8</sup>

According to Charette,

*Studies have shown that software specialists spend about 40% to 50% 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.<sup>8</sup>*

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 applied during the acquisition of a software system, product, or service and during the supply, development, operation, maintenance, and disposal of software products. These standards cover configuration management,

software testing, risk management, and software measurement.

Unfortunately, software engineering standards are initially developed by large organizations without having smaller settings in mind. Most small organizations do not have the expertise or the resources to participate in standard development. A large majority of enterprises worldwide is very small entities (VSEs), that is, enterprises, organizations, departments, or projects with up to 25 people.<sup>10</sup> In Europe, for instance, more than 92% of enterprises, called microenterprises, have up to nine employees and another 6.5% have between 10 and 49 employees.<sup>11</sup>

## 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.<sup>12</sup> One objective of the *SWEBOK Guide* is to provide a foundation for curriculum development and individual certification and licensing material. The *SWEBOK Guide* 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 Guide* 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.<sup>13</sup> The code indicates 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 behavior of and decisions made by professional software engineers, including practitioners, educators, managers, supervisors, and policy makers as well as trainees and students of the profession. Principles 3 and 8 provide information about the knowledge and the use of standards (see “Principles 3 and 8”).

Therefore, according to the *SWEBOK Guide* and the IEEE/ACM Software Engineering Code of Ethics and Professional Practice, professors who are not learning and teaching software engineering standards to software engineering students could be accused of malpractice. See “An Impossible Scenario?” for a case study.

## TEACHING SOFTWARE ENGINEERING STANDARDS—A SUCCESS STORY

A series of standards and guides have been developed to help very small entities, that is, entities having up to 25 people, in developing and maintaining software: the ISO/IEC 29110 series.<sup>10,15,16</sup> Universities in at least 21 countries are teaching ISO/IEC 29110. In Thailand,

## PRINCIPLES 3 AND 8

### PRINCIPLE 3—PRODUCT

**S**oftware 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.

## AN IMPOSSIBLE SCENARIO?

**A** software engineer in an eight-person company that develops 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 IEEE software engineering standards. But the developer did not know IEEE Standard 1028 that describes the types of software reviews with the procedures required for the execution of each type.<sup>14</sup> So after the developer completed the documentation of the requirements according to the organizational software process, he inspected the requirements by himself, using the checklist provided by the software process.

After installing the new software, the computer-controlled valves malfunctioned and caused damage in the chemical plant. One of the customer's technicians had to be brought to the emergency room of a hospital. The software supplier was asked to immediately correct the defective software. The supplier did not want to correct the software unless the customer paid an additional US\$50,000. The customer decided to sue the supplier of the defective software.

At the court hearing, the customer's lawyer requested evidence showing that an inspection of the requirements had been performed. The supplier could not provide that proof. The lawyer then interrogated the developer and asked him to describe how he inspected the requirements. The lawyer provided the judge with a copy of the IEEE-1028 standard as 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 had done an informal review instead of the inspection defined in the IEEE 1028 standard. If an inspection had been performed, the developer's colleagues could have detected the defects. In addition, if an inspection had been performed, proofs of execution (for example, a list of participants, list of defects detected, decisions made, and 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 for negligence by not performing an inspection 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 software supplier when the updated software would be delivered to the customer. The supplier indicated that it would take two weeks to rework the software: the complete development process would need to be executed since the defects in the requirements impact the test procedures, test cases, architecture, and 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 against the university he attended, on behalf of the hundreds of software engineering students at that 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 among software engineering students, lawyers, and universities that provide software engineering programs.



for instance, more than 10 universities are teaching ISO/IEC 29110. In Mexico, with the financial support of the secretariat of economy of the state of

projects, and software development centers (SDCs) of universities.<sup>17</sup>

SDCs are environments where students work in teams as a real organi-

ISO/IEC 29110. They were supported by a local research center that 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 ISO/IEC 29110 standard certification. 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, some ISO/IEC

The portfolio of software engineering standards now covers the full spectrum, that is, from the cradle to the grave, of software engineering.

Zacatecas, a six-step method has been developed to accelerate the implementation of ISO/IEC 29110 in VSEs, software engineering courses and capstone

zation to develop software for real customers either internal or external to the university. SDCs implemented a software development process using

## SURVEY OF A FEW SOFTWARE ENGINEERING PROFESSORS

**F**or 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.

The professors were asked to answer the following questions:

1. Why do you think 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 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
7. standards are not included in the curricula of their universities.

### Answers to question 2

The professors cited the following benefits:

1. students who have worked with standards 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 enemies that impose constraints on them
4. students improve their performance (for example, productivity and quality) when developing software
5. students learn discipline by covering all phases of development and give due importance to other subjects 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 so they would be able to teach them adequately
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.

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 have been translated into Czech, French, Portuguese, and Spanish and adopted as national standards by several countries. Third, teaching material and webinars are being 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, that is, developing a series of standards and guides, can readily be implemented by commercial as well as public VSEs and successfully taught to software engineering students.

## HELPING PROFESSORS TEACH SOFTWARE STANDARDS

There are a few ways to help professors teach software engineering standards. First, professors must start by acquiring and studying the standards selected for their courses. Then, they must prepare teaching material (for example, presentation material, exercises, and projects). Software engineering students will not learn about a standard if professors spend only a few minutes in class and present them with a few slides about a standard. To be of any use, students must not only study a standard and do a few exercises but put it in practice in a software development project in an environment similar to industry.<sup>18,19</sup>

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 in 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. See "Survey of a Few Software Engineering Professors."

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 in the development and implementation of a standard could also be an opportunity to publish papers. In the last 10 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.<sup>20</sup>

**"Courts generally view the application of standards as important evidence that engineers perform their work with appropriate diligence and responsibility."**

Some standards, like the IEEE-1028 standards about reviews<sup>13</sup> or the IEEE-828 standard about configuration management,<sup>14</sup> could be used in more than one course, providing a deeper knowledge to students, by performing reviews and configuration management activities in requirements, architecture, and programming courses. Finally, a one- or two-semester capstone project would be an ideal way to implement the 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<sup>14</sup> is about US\$160. Therefore, universities with a software engineering program should provide free or low-cost access to software engineering standards to their software engineering students, for example, through a membership to a professional society like the IEEE Computer Society.

**S**ince the software engineering discipline does not have as its foundation the laws of nature, not teaching software engineering standards,

that is, 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 quality, cost, and schedule objectives.


To protect these professors, those who teach software engineering and those who do not want to teach the software engineering standards pertinent to their courses, from malpractice, universities should provide them with 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 a real-world environment.<sup>17</sup>

The correct use of appropriate software engineering standards by software engineering students should increase their confidence level of achieving the objectives of a project, that is, delivering software to the customer with all of the functionalities and 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."<sup>3</sup>

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>. 

## ACKNOWLEDGMENTS

We thank the 30 professors from Ecuador, Mexico, Panama, Spain, and the United States who took the time to answer our survey.

## REFERENCES

1. C. Y. Laporte and A. April, *Software Quality Assurance*. Hoboken, NJ: Wiley, 2018.
2. J. W. Moore, "An integrated collection of software engineering standards," *IEEE Softw.*, 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," *IEEE Softw.*, vol. 36, no. 6, pp. 57–60, Nov.–Dec. 2019. doi: 10.1109/MS.2019.2932495.
4. C. Y. Laporte and F. Chevalier, "An innovative approach to the development of project management processes for small-scale projects in a large engineering company," in *Effective Standardization Management in Corporate Settings*, K. Jakobs, Ed. Hershey, PA: Business Science Reference, 2016, pp. 123–160. doi: 10.4018/978-1-4666-9737-9.ch007.
5. H. Miotti, "The economic impact of standardization technological change, standards growth in France," AFNOR, Paris, France, June 2009. [Online]. Available: <https://normalisation.afnor.org/wp-content/uploads/2016/06/Etude-ImpactEcoNorm-GB2009.pdf>
6. S. K. Land, "Results of the IEEE survey of software engineering standards users," in *Proc. Softw. Eng. Standards Symp. Forum (ISESS 97)*, Walnut Creek, CA, June 1–6, 1997, pp. 242–270.
7. H. Krasner, "The cost of poor software quality in the US: A 2020 report," in *Proc. Consortium Inf. Softw. QualityTM (CISQTM)*, Milford, MA, 2021.
8. "CHAOS2020. Beyond infinity," The Standish Group International Inc., Standish Group Report, Centerville, MA. Accessed 2021. [Online]. Available: <https://www.standishgroup.com/news/49>
9. R. N. Charette, "Why software fails [software failure]," *IEEE Spectr.*, 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," *Computer*, vol. 49, no. 8, pp. 84–87, Aug. 2016. doi: 10.1109/MC.2016.242.
11. R. Moll, "Being prepared—A bird's eye view of SMEs and risk management," *ISO Focus +*, Geneva, Switzerland, Rep., Feb. 2013.
12. P. Bourque and R. E. Fairley, Eds., *Guide to the Software Engineering Body of Knowledge: Version 3.0 (SWEBOK Guide)*. Los Alamitos, CA: IEEE Computer Society, 2014.
13. *Software Engineering Code of Ethics and Professional Practice*, IEEE-CS-1999. 1999. [Online]. Available: <https://www.computer.org/education/code-of-ethics>
14. *IEEE Standard for Software Reviews and Audits*, IEEE Standard 1028-2008, Aug. 15, 2008.
15. C. Y. Laporte and J. M. Miranda, "Delivering software and systems engineering standards for small teams—Feedback from very small entities, their customers, auditors and academia on ISO/IEC 29110," *Computer*, vol. 53, no. 8, pp. 79–83, Aug. 2020. doi: 10.1109/MC.2020.2993331.
16. C. Y. Laporte, M. Muñoz, J. Mejia Miranda, and R. V. O'Connor, "Applying software engineering standards in very small entities: From startups to grownups," *IEEE Softw.*, vol. 35, no. 1, pp. 99–103, Jan./Feb. 2018. doi: 10.1109/MS.2017.4541041.
17. M. Muñoz, J. Mejia, A. Peña, G. Lara, and C. Y. Laporte, "Transitioning international software engineering standards to academia: Analyzing the results of the adoption of ISO/IEC 29110 in four Mexican universities," *Comput. Standards Interfaces*, vol. 66, Oct. 2019. doi: 10.1016/j.csi.2019.03.008.
18. C. Y. Laporte, A. April, and K. Benchérif, "Teaching software quality assurance in an undergraduate software engineering program," *Softw. Qual. Professional J.*, ASQ, vol. 9, no. 3, pp. 4–10, 2007.
19. C. Y. Laporte, "International software engineering standards applied in undergraduate and graduate software quality assurance courses IEEE standards university," *IEEE Standards Education E-Magazine*, Nov. 2015. [Online]. Available: <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. doi: 10.1016/j.csi.2019.03.005.

**CLAUDE Y. LAPORTE** is an adjunct professor of software engineering at École de technologie supérieure, Montréal, Québec, H3C 1K3, Canada, and the lead editor of the ISO/IEC 29110 series of standards and guides. Contact him at [claporte@etsmtl.ca](mailto:claporte@etsmtl.ca).

**MIRNA MUÑOZ** is a professor of software engineering at Centro de Investigación en Matemáticas- Unidad Zacatecas, Zacatecas, 98160, Mexico, and co-editor of the ISO/IEC 29110 Agile Software Development Guide. Contact her at [mirna.munoz@cimat.mx](mailto:mirna.munoz@cimat.mx).