Towards the integration of security practices in the software implementation process of ISO/IEC 29110: A mapping

Mary-Luz Sánchez-Gordón[1], Ricardo Colomo-Palacios[2], Alex Sánchez[3], Antonio de Amescua Seco[1] and Xabier Larrucea[4]

[1] Universidad Carlos III de Madrid, Computer Science Department
Av. Universidad 30, Leganés, 28911, Madrid, Spain
mary_sanchezg@hotmail.com, amescua@inf.uc3m.es
[2] Faculty of Computer Sciences, Østfold University College
Postboks 700, 1757 Halden, Norway
ricardo.colomo-palacios@hiof.no
[3] LogicStudio
Ciudad del Saber, Building 235 Panama
alex.sanchez@logicstudio.net
[4] Tecnalia, Bizkaia, Spain
xabier.larrucea@tecnalia.com

Abstract. Secure software practices are gradually gaining relevance among software practitioners and researchers. This is happening because today more than ever software is becoming part of our lives and cybercrimes are constantly appearing. Despite its importance, its current practice in the software industry is still scarce. Indeed, software security problems are divided 50/50 between bugs and flaws. In particular, it remains a significant challenge for software practitioners in small software companies. Therefore, there is a need to support small companies in changing their existing ways of work to integrate these new and unfamiliar practices. The aim of this study is twofold. First, to help building an awareness of the software security process among practitioners in small companies. Second, to help the integration of these practices with software implementation process of ISO/IEC 29110 which results in an extension of the latter with additional activities identified from the industry best practices. Nevertheless, the extension proposal is to be performed selectively, based on the value of the software as an asset to the stakeholders and on stakeholders needs.

Keywords: Software Security, CSSLP, S-SDLC, small companies, VSE, ISO/IEC 29110

1 Introduction

Today, more than ever, software has become part of our lives. It is integrated into systems that we use every day and we are increasingly dependent on those systems to

work. Specially, developed nation´s economy and defense depend in large part on the reliable execution of software. In fact, software is ubiquitous —software is everywhere— affecting all aspects of our personal and professional lives [1]. Therefore, security becomes an important issue and a crucial requirement for software systems [2]. New security challenges arise when new —or old— technologies are put to new use. Opening the Internet to commercial use in the early 1990s raised the importance of security policies for remote transactions [3]. There is evidence that global interconnectedness combined with the proliferation of hacker tools means that today's computer systems are actually less secure than equivalent systems a decade ago [4]. Applications have been vulnerable for as long as they have existed. Over the past few years, aside from operating systems, they have been cited as the leading vector for attacks. In 2011, National Vulnerability Database maintained by US National Institute for Standards and Technology (NIST)[1] stated that 92% of the reported vulnerabilities are in applications and not due to insecure networks. That same year, top vulnerabilities included[2]: SQL injection, integer overflow, buffer overflow, uncontrolled format string, missing authentication, missing or incorrect authorization, and reliance on untrusted inputs in a security decision (aka tainted inputs). Already in 2003, the programming errors were 64% of the vulnerabilities in the NIST database, and 51 out of those were repeated basic mistakes such as buffer overflows, cross-site scripting, injection flaws [5].

According to the "2015 (ISC)² Global Information Security Workforce Study", 72% of the survey respondents (13,930 qualified information security professionals), indicated that application vulnerabilities are their top security concern [6]. Moreover, regarding cloud security threats, they pointed out that data breaches and data loss topped the list of concerns. In this sense, the "2016 Cost of Data Breach Study: Global Analysis" carried out by Ponemon Institute [7], stated that average total cost of a data breach for the 383 companies participating in this research increased from $3.79 to $4 million. The average cost paid for each lost or stolen record containing sensitive and confidential information increased from $154 in 2015 to $158 by 2016. Furthermore, Gartner [8] claims that worldwide spending on IT security products and services will reach $81.6 billion in 2016; an increase of 7.9% over 2015. Another issue is the loss of credibility, while intangible, has tangible repercussions. Paying the extra cost of developing software correctly from the start reduces the cost of fixing it after it is deployed —and produces a better, more robust, and more secure product [9]. This approach reduces the need to patch the software in order to fix security holes. Moreover, incorporating security in software is often misunderstood as an impediment to business agility and not necessarily as an enabler for the business to produce quality and secure software [10]. In certain situations, security in software is not even considered, it is overlooked [11, 10]. Moreover, the common approach towards the inclusion of security within a software system is to identify security requirements after the definition of a system [2]. Thus, incorporating security in later stages of software development will increase the risks of introducing security vulnerabilities into software. A

[1] http://nvd.nist.gov/

[2] http://cwe.mitre.org/top25/#Listing

vulnerability is a software defect that an attacker can exploit [9]. Software security problems are divided 50/50 between bugs and flaws [12]. A bug is an implementation-level software problem whereas a flaw is a design-level or architectural software defect.

The most critical difference between secure software and insecure software lies in the nature of the processes and practices used to specify, design, and develop the software [9]. Traditionally, Software Engineering deals with security as a non-functional requirement and usually considers it after the definition of the systems [10]. Consequently, software that is developed with security in mind is typically more resistant to both intentional attack and unintentional failures [9, 13]. Security in the software development life cycle (SDLC) is necessary but not sufficient. In practice, there are a lot of aspects that are not part of the life cycle for any particular application such as building the security team, maintaining legacy code, gaining the organization's support (budget and respect), establishing an education and training program, establishing standards and metrics, handling breaches and incidents, tooling and building feedback loops for continuous improvement [13].

In the context of this research, a vast majority of small software companies are very small entities (VSEs) —enterprise, government, or not-for-profit organizations; departments; or projects with up to 25 people who develop systems with hardware and software components and/or software products [14]. They face unique challenges [15], their products are sold to their customers directly or are integrated into those developed by larger organizations, possibly distributed to thousands of users worldwide [14]. In addition to the time constraints placed on software development projects, scarce human resources are also noted as a limitation [15, 16]. Without enough resources in order to achieve all the essential tasks within the required timeline, considering the uptake of increased work effort (including security during software development) is noted as being difficult if not improbable. Therefore, the aim of this study is twofold. First, to create an awareness of the software security process among practitioners in small companies. Second, to provide the first approach towards narrowing the gap between security and ISO/IEC 29110 standard, which fix well in this type of companies. The integration of these practices with software implementation (SI) process of ISO/IEC 29110 results in an extension of the latter with additional activities identified from the industry best practices in security software —Certified Secure Software Lifecycle Professional (CSSLP) common body of knowledge (CBK). This body of knowledge was chosen because it is an approach agnostic and focuses on SDLC. Nevertheless, the extension proposal is to be performed selectively on the basis of the value of the software as an asset to the stakeholders and on stakeholders needs. The remainder of this paper is structured as follows: Section 2 outlines the CSSLP CBK and SI process ISO/IEC 29110. In Section 3 the research approach and results are presented. Finally, Section 4 summarizes a conclusion as well as outlines future work.

2 Background

2.1 Models and Software Security Certifications

There are a variety of models such as CERT Resilience Management Model "RMM", Building Security in Maturity Model "BSIMM", Capability Maturity Model Integration for Acquisitions "CMMI", SwA Forum Processes and Practices Group Process Reference Model "PRM", and OWASP Software Assurance Maturity Model "SAMM". Similarly, Microsoft has a method for software development and on how to develop secure software which is called Security Development Lifecycle (SDL). Each one has their advantages and disadvantages. It seems overwhelming to non-security experts like software practitioners in small companies. As a consequence of the need for training and certification on software security, the industry has developed certifications based on specific languages and/or platforms. In what follows, there are some of the relevant certifications [17].

- Global Information Assurance Certification (GIAC) offers three software security certifications[3]. The certifications in Java and .NET —GIAC Secure Software Programmer-.NET/.JAVA (GSSP-.NET/.JAVA). And the GIAC Certified Web Application Defender (GWEB).
- The EC-Council offers a program known as EC-Council certified secure programmer (ECSP) that has certifications in Java and .NET[4] —Certified Secure Programmer in .NET (ECSP.Net) and Certified Secure Programmer in .Java (ECSP-Java).
- The CERT Secure Coding Professional Certificates is part of Carnegie Mellon University's Software Engineering Institute. This certificate program is designed for developers who are programming in C and C++, or Java language
- The International Information System Security Certification Consortium, also known as ISC[2] offers a Certified Secure Software Lifecycle Professional (CSSLP) certification. CSSLP was designed to validate SDLC security competencies based on a common body of knowledge (CBK).

The CSSLP CBK approach was chosen as a reference for this study due to its agnostic approach, its independence of any specific language or vendor and its focus on SDLC.

2.2 The Certified Secure Software Lifecycle Professional Common Body of Knowledge

The Certified Secure Software Lifecycle Professional (CSSLP) common body of knowledge (CBK) provides a comprehensive approach to building secure systems by incorporating security into all phases of the software lifecycle. The International Information Systems Security Certification Consortium (ISC) sponsors the CSSLP cer-

[3] http://www.giac.org/certifications/software-security
[4] https://www.eccouncil.org/programs/certified-secure-programmer-ecsp/

tification[5]. The CSSLP certification is international in its scope and therefore, does not explicitly address US standards such as those from NIST. The CSSLP certification emphasizes best practices in secure software development and covers the following eight domains of the CSSLP CBK [10]:

- **Secure Software Concepts** include the core software security requirements and foundational design principles as they relate to issues of privacy, governance, risk and compliance. The aim is to understand the software methodologies needed in order to develop software that is secure and resilient to attacks.
- **Security Software Requirements** provide concepts related to understanding the importance of identifying and developing software with secure requirements which could be incorporated in order to produce software that is reliable, resilient and recoverable.
- **Secure Software Design** gives an understanding of how to ensure that software security requirements are included in the design of the software. That means secure design principles and process.
- **Secure Software Implementation/Coding** allows understanding the importance of programming concepts that can effectively protect software from vulnerabilities. That means software coding vulnerabilities, defensive coding techniques and processes, code analysis and protection, and environmental security considerations that should be factored into software.
- **Secure Software Testing** includes the overall strategies and plans of functional and security testing that should be performed, the criteria for testing, concepts related to impact assessment and corrective actions, and the test data lifecycle.
- **Software Acceptance** provides an understanding of the requirements for software acceptance, paying specific attention to compliance, quality, functionality, and assurance before software is released or deployed into production.
- **Software Deployment, Operations, Maintenance and Disposal** is focused on the identification of processes during installation and deployment, operations and maintenance. Finally, disposal that can affect the ability of the software to remain reliable, resilient, and recoverable in its prescribed manner.
- **Supply Chain and Software Acquisition** give an understanding of the importance of supplier sourcing and being able to validate vendor integrity, from third-party vendors to complete outsourcing. That means how to manage risk through the adoption of standards and best practices for proper development and testing across the entire lifecycle of products.

2.3 ISO/IEC 29110 Standard

Although ISO/IEC 29110 is an emerging standard, a series of pilot projects have been completed in several countries utilizing some of the deployment packages (DPs) developed [18]. This ISO/IEC 29110 standard is applicable to Very Small Entities (VSEs). VSEs are enterprises, organizations, departments or projects of up to 25 peo-

[5] https://www.isc2.org/csslp/default.aspx

ple. This standard has a generic profile group that provides a four-stage roadmap for VSEs that do not develop critical systems or critical software: Entry, Basic, Intermediate and Advanced profiles [19]. This study is based on the Basic profile which describes the development practices of a single application by a single project team. Basic profile has two interconnected processes: Project Management (PM) and Software Implementation (SI). This study focuses on the SI process because its goal is to achieve a software product that satisfies the needs and expectations of all potential users, including security issues.

Software Implementation Process. The aim of the SI process is to achieve systematic performance of the analysis, design, construction, integration, and test activities for new or modified software products according to the specified requirements. The activities of the SI process are:

- **Software Implementation Initiation** ensures that the Project Plan established in Project Planning activity is committed to by the Work Team
- **Software Requirements Analysis** analyzes the agreed Customer's requirements and establishes the validated project requirements.
- **Software Architectural and Detailed Design** transforms the software requirements to the system software architecture and software detailed design
- **Software Construction** develops the software code and data from the Software Design.
- **Software Integration and Tests** ensures that the integrated Software Components satisfy the software requirements.
- **Product Delivery** provides the integrated software product to the Customer

3 Research Approach

Due to increasing recognition of the importance of security throughout the entire life cycle, new initiatives strengthening ties for security within the SDLC have been conducted. However there is a need to assist organization in processes that minimize and ideally prevent security vulnerabilities [1]. This is especially true for small companies because they find hard to deal with their software process [15]. Consequently, it is important to harmonize software processes and security issues. The authors carried out mapping, as it is one of the most widely used strategies in harmonizing software processes. They follow the guidelines provided at [20] including these steps: 1) Analyze the models; 2) Design the mapping; 3) Carry out the mapping; 4) Present the outcomes and 9) Analyze the results. In what follows, the mapping performed is described using the method provided.

3.1 Models Analysis

The ISO/IEC 29110 standard and CSSLP certification were chosen for this study based on their approach agnostic and growing relevance among small organizations

and security professionals, respectively. The first activity is to analyze each reference model involved in a mapping process. ISO/IEC 29110 and CSSLP CBK were studied in detail.

3.2 Mapping Design

The design involves the following activities: (i) Identification of elements to be compared, they are the SI process of ISO/IEC 29110 standard and the practices (in each domain) of CSSLP CBK. (ii) Direction of the comparison, it is from ISO/IEC 29110 standard to CSSLP CBK. (iii) Comparison scale definition, authors use a "traffic light" scale for the one to one mapping. This scale is also used in our previous works [21]:

- E: explicit, the item has appeared in the framework's definition.
- I: implicit, the item has not appeared explicitly in the framework definition. Inferred by the authors or referred inside a previous work of the authors.
- U: unavailable, the item has not appeared anyway.

(iv) Comparison template definition, all these values are analyzed and checked from a holistic point of view and the authors determine to what extent activities (from a SI process of ISO/IEC 29110) that are related to practices/techniques of CSSLP CBK could be extended.

3.3 Mapping

This mapping is an iterative process because the comparison is performed completely on one ISO/IEC 29110 activity and then on the others in turn. At the same time, it is incremental due to the comparison outcome grows and evolves with each iteration until it becomes the final one. The authors analyze the ISO/IEC 29110 with CSSLP CBK. For the SI process of ISO/IEC 29110 all activities are studied. Moreover, the authors identified specific practices and/or techniques of CSSLP CBK. The objective is not to set a naïve approach which compares just names. Therefore, a relationship between reference models is defined first and then a drilling down process analyzing in detail these relationships helps us to identify fine grained relationships. All mappings are managed by using several spreadsheets where ISO/IEC 29110 activities are displayed as rows and practices/techniques of CSSLP CBK are displayed as columns. As a result, a set of practices and techniques to help practitioners understand how to secure their development processes and to apply those principles in practice is defined. It represents an extension to the ISO/IEC 29110 standard.

3.4 Outcomes

Figure 1 depicts the resulting mapping for domains of CSSLP CBK. Each one has a fulfillment result based on the intersection of activities of ISO/IEC 29110. At first glance the result is not surprising but it is interesting from a security perspective. At

high level, two domains, *Secure Software Concepts* and *Supply Chain and Software Acquisition*, have not appeared anywhere. Moreover, *Software Deployment, Operations, Maintenance and Disposal* are partially related with *Product Delivery.* Although *Software Implementation Initiation* is not apparently related at this level, it is crucial because it ensures that the project plan, including security issues, is committed by the team. The other five domains receive more coverage —*Requirement, Design, Implementation, Testing, and Acceptance*— as can be seen below. Next, the activities of SI process ISO/IEC 29110 and the pertinent practices of CSSLP CBK.

Fig. 1. Mapping between ISO/IEC 29110 to CSSLP CBK

Implementation initiation. This activity prepares the team for the remainder of the activities and brings together all the necessary tools to accomplish the project. Small companies' constraint is limited budget for setting up the environment. At this point, it is important to be familiar with what each tool or technology can be used for and how it can impact the overall state of software security.

Software Requirement Analysis. This activity studies users' needs and expectations to define the project scope and identify key functionalities, including non-functional requirements —and security is often considered as such. Moreover, it is worth nothing that small companies cannot afford to have security experts and there are several types of security requirements that address the various principles of software security. Nevertheless, protection needs can be elicited using several methods including brainstorming, surveys, policy decomposition, data classification and use and misuse case modeling. The policy decomposition process is made up of breaking down high-level requirements into granular finer level software security requirements. Data classification can help with assuring that appropriate levels of security controls are assigned to data based on their sensitivity levels. Finally, use and misuse case modeling, sequence diagrams and subject–object models can be used to glean software security requirements.

Software Design. This activity is the keystone in the SDLC. Failure to describe a design architecture that will incorporate all the requirements is a common reason for

project failure. Thus semantic or business logic flaws are related to design issues. Small companies' constraint is limited development team. When they design software, possible threats and security taken into account, and they should take into consideration secure design principles to assure confidentiality, integrity, and availability. In fact, the time that is necessary to fix identified issues is shorter when the software is still in the design phase. Below some considerations that could be useful to do that.

- Determine entry and exit points that an attacker could use to compromise the software asset or the data it processes. Take into account the following principles: least privilege, separation of duties, layered defense, fail secure, economy of mechanisms —KISS (Keep It Simple Stupid) principle —, complete mediation, open design, least common mechanism, and leveraging existing components.
- Design considerations address the core security elements of confidentiality, integrity, availability, authentication, authorization, and auditing.
- Design process includes attack surface evaluation, threat modeling, control identification and prioritization, and documentation. Threat models are useful to identify and prioritize controls (safeguards) that can be designed, implemented (during the development phase), and deployed but it take time.
- Proven software architectures and technologies can be leveraged to enhance security in software. For instance, authentication, identity Management (IDM), Credential Management, Flow Control, Auditing/Logging, Data Loss Prevention (DLP), Virtualization and Digital Rights Management
- Review of the software's design and architecture from a security perspective.

Software construction. This activity entails developers producing components using a systematic approach. Small companies' constraints are limited development team and short time to deliver. In fact, writing secure code is an important and critical factor in order to ensure the resiliency of software security controls. The mapping performed using the method provided is described in the following section.

- The security advantages or lack thereof of software development methodology must be taken in account.
- Build security protection controls based on common coding vulnerabilities and an understanding of how an attacker will try to exploit the software (because of limited space, details are not included but are available on [10]).
- Secure software development processes include versioning, code analysis and code review.
- Build Environment and Tools Security. The main kinds of build tools are compilers, packers and packagers —e.g. the Red Hat Package Manager (RPM) and the Microsoft Installer (MSI).

Software integration and tests. This activity comprises running a set of tests and identifying issues that must be. Small companies' constraints are limited development team. In this sense, security testing can be used to determine the means and opportunities by which software can be attacked. These tests are as follows:

- Both white box and black box security testing —e.g. fuzzing, scanning and penetration testing— are used to determine the threats to software. They are based on knowledge of how to test for common software vulnerabilities.
- Testing related to software security issues are testing for input validation, injection flaws testing, testing for nonrepudiation, testing for spoofing, failure testing, cryptographic validation testing, testing for buffer overflow defenses, testing for privilege escalations defenses, and anti-reversing protection testing.
- The use of tools which are applicable to the specific situation. Some of the common security tools include: reconnaissance (information gathering) tools, vulnerability scanners, fingerprinting tools, sniffers/protocol analyzers, password crackers, web security tools —e.g., scanners, proxies, vulnerability management—, wireless security tools, reverse engineering tools —assembler and disassemblers, debuggers, and decompilers—, source code analyzers, vulnerability exploitation tools, security-oriented operating systems, and privacy testing tools.
- Fixing defects must never be performed directly in the production environment, and proper change management principles must be used to promote fixes from development and test environments into the user acceptance testing (UAT) and production environment.

Software product delivery. This activity ensures there would be no delays in order to gain product acceptance so the customer completes the payment to the company. Small companies' constraint is short time to deliver and ensure that software is not only operationally hack-resilient, but also compliant with applicable regulations —i.e. there is a formal software acceptance process which comprises the validation of security requirements and the verification of security controls. In what follows crucial considerations about the pre- and post-installation software security.

- Security requirements need to be validated and security controls verified by internal and/or independent third party security testing. Software must not be deployed/released until it has been certified and accredited that the residual risk is at the appropriate level.
- Hardening of software implicates:
  - Remove maintenance hooks before deployment.
  - Remove debugging code and flags in code.
  - Change the way to write code not to contain any sensitive information —i.e. unneeded comments, dangling code, or sensitive information from comments in code.
- Enforcement of security principles means:
  - Use pre-installation checklists in order to ensure that the needed parameters required for the software to run are appropriately configured.
  - Grant appropriate administrative rights (least privilege) to the software during the installation process.
  - Not allow developers access to production systems to install software.
- Development and test environment must be the same as the production environment in which the software will be deployed post-acceptance.

- Bootstrapping and secure startup could be achieved using hardware's trusted platform module (TPM) chip.

4 Conclusions

Despite of the growing understanding of the importance of including of security throughout the SDLC, it is usually treated superficially and the typical security process is to add a standard set of security mechanism, such as authentication, into the system [22]. While the current business environment is fast-paced and increasingly exposed to threats, software practitioners must go beyond when developing software. Although security throughout the SDLC requires allocation of resources such as time and has an impact on the project life cycle, it is worthy and valuable. In this paper, authors shed light on this fact. Its aim is to raise awareness on its importance and to provide the argument for better enforcement of security and its practices.

Furthermore, this paper presents the integration of these practices with software implementation (SI) process of ISO/IEC 29110. As a result of this, an extension of the latter with additional activities identified from the best practices of CSSPL CBK is presented. Since clearly an organization cannot protect and prevent every risk and threat, the extension proposal is to be performed selectively on the basis of the value of the software as an asset to the stakeholders and on stakeholders needs. It is worth noting that some of these practices appear to have common sense validity but there are others not so obvious deserving more attention. This study is a first step in exposing and addressing the challenging landscape of security in small companies. As future work, a sub-set of the extension will be adapted in a small company because the security should be properly considered as part of its software development process.

References

1. O'Connor, R.V., Colomo-Palacios, R.: Security Awareness in the Software Arena. In: Engemann, K. (ed.) Routledge Companion to Risk, Crisis and Security in Business. Routledge (2017).
2. Salini, P., Kanmani, S.: Survey and Analysis on Security Requirements Engineering. Comput Electr Eng. 38, 1785–1797 (2012).
3. Gollmann, D.: Computer security. Wiley Interdiscip. Rev. Comput. Stat. 2, 544–554 (2010).
4. Garfinkel, S.L.: The Cybersecurity Risk. Commun ACM. 55, 29–32 (2012).
5. Heffley, J., Meunier, P.: Can source code auditing software identify common vulnerabilities and be used to evaluate software security? In: 37th Annual Hawaii International Conference on System Sciences, 2004. pp. 1–10 (2004).
6. Suby, M., Dickson, F.: Global Information Security Workforce Study. Frost & Sullivan (2015).
7. Ponemon Institute LLC: 2016 Cost of Data Breach Study: Global Analysis. (2016).

8. Gartner Says Worldwide Information Security Spending Will Grow 7.9 Percent to Reach $81.6 Billion in 2016, http://www.gartner.com/newsroom/id/3404817.
9. Allen, J.H., Barnum, S., Ellison, R.J., McGraw, G., Mead, N.R.: Software Security Engineering: A Guide for Project Managers. Addison-Wesley Professional (2008).
10. Mano, P.: Official (ISC)2 Guide to the CSSLP. CRC Press (2015).
11. Daud, M.I.: Secure Software Development Model: A Guide for Secure Software Life Cycle. Presented at the Proccedings of the International MutiConference on Engineers and Computer Scientists (IMECS) , Hong Kong (2010).
12. McGraw, G.: Software Security: Building Security In. Addison-Wesley Professional (2006).
13. Chess, B., Arkin, B.: Software Security in Practice. IEEE Secur. Priv. 9, 89–92 (2011).
14. Laporte, C.Y., O'Connor, R.V.: Systems and Software Engineering Standards for Very Small Entities: Accomplishments and Overview. Computer. 49, 84–87 (2016).
15. Sánchez-Gordón, M.-L., O'Connor, R.V.: Understanding the gap between software process practices and actual practice in very small companies. Softw. Qual. J. (2015).
16. Sánchez-Gordón, M.-L., O'Connor, R.V., Colomo-Palacios, R.: Evaluating VSEs Viewpoint and Sentiment Towards the ISO/IEC 29110 Standard: A Two Country Grounded Theory Study. In: SPICE 2015. pp. 114–127. Springer-Verlag, Gothenburg, Swedan (2015).
17. Grover, M., Durham, N.C., Cummings, J., Janicki, T.: Moving Beyond Coding: Why Secure Coding Should be Implemented. J. Inf. Syst. Appl. Res. (2016).
18. O'Connor, R.V., Laporte, C.Y.: The Evolution of the ISO/IEC 29110 Set of Standards and Guides. Int. J. Inf. Technol. Syst. Approach IJITSA. 10, 1–21 (2017).
19. ISO: Software engineering – Lifecycle profiles for Very Small Entities (VSEs) Part 5-1-2: Management and engineering guide: Generic profile group: Basic Profile. , Geneva (2011).
20. Baldassarre, M.T., Caivano, D., Pino, F.J., Piattini, M., Visaggio, G.: Harmonization of ISO/IEC 9001:2000 and CMMI-DEV: from a theoretical comparison to a real case application. Softw. Qual. J. 20, 309–335 (2011).
21. Sánchez-Gordón, M.-L., Colomo-Palacios, R., Herranz, E.: Gamification and Human Factors in Quality Management Systems: Mapping from Octalysis Framework to ISO 10018. In: EuroSPI 2016. pp. 234–241. Springer-Verlag, Graz, Austria (2016).
22. Haralambos, M., Giorgini, P.: Integrating Security and Software Engineering: Advances and Future Visions: Advances and Future Visions. Idea Group Inc (IGI) (2006).