Integration of Security Practices in Agile Software Development

Subjects: Computer Science, Software Engineering

Contributor: Yolanda Valdés-Rodríguez , Jorge Hochstetter-Diez , Jaime Díaz-Arancibia , Rodrigo Cadena-Martínez

Software development must be based on more than just the experience and capabilities of your programmers and your team. The importance of obtaining a quality product lies in the risks that can be exploited by software vulnerabilities, which can jeopardize organizational assets, consumer confidence, operations, and a broad spectrum of applications.

secure development secure software software process

1. Introduction

Advances in information and communication technologies allow people and organizations to have greater connectivity [1]. Bringing this access to the masses has specific risks associated with its use, especially considering that we rely heavily on software systems in various daily activities we may perform [1][2]. Due to that, it is relevant to ensure various security issues [3]; for example, we have to expect that the software will continue to function correctly under malicious attack [4].

Secure software is designed, implemented, configured, and operated to fulfill essential properties: to continue functioning in the presence of computer attacks or mitigate damage and recover as quickly as possible [5]. Software developers must design, develop and deploy our systems with a secure mindset, applying strategies that minimize the likelihood of exposure and impact to threats [6][7]. However, in practice, this is different; software development is based on adopting a reactive approach, which consists of assessing the security of applications once they have been developed, focusing on the later stages of the software life cycle. Fixing bugs in this way helps; however, this approach proves to be more costly when resolving software security flaws at later stages [8].

Thus, organizations have been forced to define and implement a set of countermeasures that allow us to secure our information assets against casual or deliberate attacks. Yet, these seem insufficient in the face of the increase in this type of attack worldwide, making it essential for organizations to define security policies that consider the wide variety of attacks to which they are exposed [9][10].

The traditional response to this scenario focuses on reducing risk by standardizing information security and defining and implementing best practices or controls.

For example, the ISO27001 [11], NIST [12], and COBIT [13] standards propose a set of controls that must be complied with to secure an organization’s information assets and operational continuity.

Although it has been demonstrated that the implementation of the security controls of these standards reduces security incidents, what is really implemented is a layer or security shield on top of the existing systems, which means the software systems themselves are not aware of the anomalous behaviors of the users, and, therefore, as software systems, they do not have a reaction protocol for it, that is, once the standards penetrate the security shield provided (if it exists), the software systems are exposed [10][14].

A complementary approach is the determination of the particular security requirements of each organization, reflected as non-functional requirements of their software systems [15][16]. From this point, the different phases of the software production cycle must accommodate these requirements, which means that security aspects must be represented as requirements; they must be considered in the design, in the development of test cases, in the coding, and application of tests, packaging, and delivery of the product [10][17].

A software product can be vulnerable to its construction failures or provoked attacks [18][19]. To reduce vulnerabilities and make the product secure, measures such as the integration of security concepts in all its development stages or approaches or methodologies that allow the integration of security into the software life-cycle must be applied [18][20].

2. Security Practices in Agile Software Development

Agile software development is gaining acceptance as a flexible approach. This approach prioritizes the software’s continuous and early delivery, changing the requirements even in advanced stages of development and adapting to the customer’s needs [21]. Although the agile approach is gaining more and more followers, it reveals that it has certain security-related disadvantages.

Generally, security is not considered in any of the phases of the software development life cycle (SDLC) [22]. At best, it is covered through the definition of non-functional requirements; for this reason, it is either not taken into account, or it is done at the end of the project [23]. Security is seen as an element that increases project development and delivery times, which goes against agile principles [24].

There are some software development lifecycles that are starting to consider agile principles:

- Correctness by Construction (CbyC), is a highly effective method for developing software that requires critical levels of safety and provability. The main objectives of this methodology are to minimize the defect rate and increase resilience to change, achieved through two fundamental principles: making it very difficult to introduce bugs and ensuring that bugs are identified and eliminated as early as possible. To achieve these goals, CbyC seeks to ensure that software is correct from the start through rigorous safety requirements, a detailed definition of system behavior, and a robust and verifiable design [25][26].

- ViewNext, model proposed by [27] is an agile adaptation of the S-SDLC [28], which incorporates security best practices from known models along with other security tasks, based on the spiral model, is integrated into normal software engineering life cycles. The model corrects weaknesses present in previous models and follows a preventive approach, making it an effective alternative for secure software development. Known as Agile and Secure Software Development Life, this model has been the subject of study in [29].

- Microsoft SDL Agile, it is an adaptation of the SDL Methodology (Security Development Lifecycle) that was developed by Microsoft to integrate security into agile software development processes [30]. The Agile SDL methodology focuses on integrating security into each iteration of the agile software development process. Rather than following a “wait until the end” approach to integrating security, the Agile SDL methodology promotes the inclusion of security activities in all phases of the agile development process. Security activities include early risk identification, defining secure user stories, performing security testing in each iteration, and implementing security best practices in the agile development process. The Agile SDL methodology is based on the agile software development lifecycle, which includes planning, analysis, design, implementation, testing, and maintenance. By integrating security into each stage of this lifecycle, it is possible to ensure that the software developed is secure and complies with security requirements.

- Building Security In Maturity Model (BSIMM), is a security maturity model used to describe the practices and processes used by leading software security organizations to develop, improve and maintain effective software security programs [31]. BSIMM focuses on assessing organizations’ software security programs by measuring their maturity in 12 common security practices. This model helps organizations develop their own software security program and provides a tool for ongoing assessment of software security maturity over time. The latest version of the model, BSIMM10, released in 2020, addresses agile properties of software development. It includes practices and processes relevant to agile approaches, such as continuous integration and continuous delivery, security automation, security management in the product backlog, and security collaboration between development teams. In addition, BSIMM10 focuses on the importance of security in the context of agile frameworks, such as Scrum and DevOps.

The software development models share a common focus on improving software safety throughout the software development life cycle. BSIMM and SAMM are software security maturity models that measure the maturity of software security programs and provide guidance for improving them, although BSIMM focus specifically on agile properties. Meanwhile, S-SDLC and McGraw’s Secure Software Development Life Cycle Process are secure software development life cycle models that integrate security into each stage of the development process. While S-SDLC provides general guidelines and best practices for developing secure software, McGraw’s Secure Software Development Life Cycle Process focuses on eliminating vulnerabilities through a secure architecture from the outset. On the other hand, Correctness by Construction is a methodology that seeks to produce correct software from the beginning through a rigorous definition of security requirements, a solid and verifiable design, and a preventive approach to avoid introducing errors. Finally, SDL Agile is an adaptation of Microsoft’s SDL model that integrates security into agile software development processes. This model focuses on security automation, security management in the product backlog, and security collaboration between development teams, fostering collaboration and continuous integration of security throughout the software development lifecycle. In summary, while these models share a common goal of improving software security, each offers a unique and complementary approach to achieving it.

Software assurance is the confidence that a system meets all its security requirements. In most of those requirements of interest to customers and users of the software, this confidence is based on specific evidence collected and evaluated through assurance techniques [32].

The techniques or mechanisms established for information security are considered rigid to respond to the changes and advances that are presented in the changing security environment, where there is a need for a more agile method to deal with new threats and vulnerabilities [33]. Thus, the traditionally established security mechanisms are no longer effective when used with software development methodologies adapted to the needs of the current environment, such as agile methodologies [34].

The use of agile methodologies in software development implies, on several occasions, not considering the good practices of secure development, whose purpose is to guarantee the fulfillment of the own security policies of the software development [35][36][37][38][39][40].

Several authors state that developing secure software using agile methodologies is challenging. Applying security practices in agile methodologies presents challenges because agile methodologies support requirements changes prefer frequent deliveries, and their practices do not include security engineering activities [36].

The paper published by [41] discusses defects in the requirements specification stage, which generally in security aspects are misunderstood and incorrectly specified due to lack of security expertise. These concerns become even more challenging in agile contexts, where lightweight documentation is generally produced. To address this problem, the indicated article proposes an approach to review security-related aspects of web application requirements specifications in agile contexts. The methodology considers user stories as inputs and relates them to the OWASP (Open Web Application Security Project) security properties, which must be verified and then generate a reading technique to help reviewers detect defects. The methodology was evaluated through three experimental tests performed with 56 novice software engineers, measuring effectiveness, efficiency, usability, and ease of use. The results indicate that the proposed methodology has a positive impact on the number of vulnerability findings in terms of effectiveness and efficiency.

There seems to be a clear need for a software development model which addresses security issues at any stage of the software life cycle and considers the benefits of agile models. In this context [23] proposes a model that introduces security as a crucial element in software development environments and, at the same time, leverages agile properties.

On the other hand, Sharma et al. [42] offers a framework for agile development that addresses security, considering customer requirements. The implementation of this Framework has been implemented in Java to automate the whole process, although the author points out that the suggested security activities should be tested and evaluated in a real industrial environment.

References

1. Faheem, M.; Shah, S.B.H.; Butt, R.A.; Raza, B.; Anwar, M.; Ashraf, M.W.; Ngadi, M.A.; Gungor, V.C. Smart grid communication and information technologies in the perspective of Industry 4.0: Opportunities and challenges. Comput. Sci. Rev. 2018, 30, 1–30.
2. Lee, M.; Yun, J.J.; Pyka, A.; Won, D.; Kodama, F.; Schiuma, G.; Park, H.; Jeon, J.; Park, K.; Jung, K.; et al. How to respond to the fourth industrial revolution, or the second information technology revolution? Dynamic new combinations between technology, market, and society through open innovation. J. Open Innov. Technol. Mark. Complex. 2018, 4, 21.
3. Liou, J.C.; Duclervil, S.R. A survey on the effectiveness of the secure software development life cycle models. In Innovations in Cybersecurity Education; Springer: Berlin/Heidelberg, Germany, 2020; pp. 213–229.
4. McGraw, G. From the ground up: The DIMACS software security workshop. Secur. Privacy IEEE 2003, 1, 59–66.
5. Castellaro, M.; Romaniz, S.; Ramos, J.C.; Feck, C.; Gaspoz, I. Aplicar el Modelo de Amenazas para incluir la Seguridad en el Modelado de Sistemas. In Proceedings of the V Congreso Iberoamericano de Seguridad Informática—CIBSI, Bogota, Colombia, 22–24 January 2016; Volume 16.
6. Hernández Yeja, A.; Porven Rubier, J. Procedimiento para la seguridad del proceso de despliegue de aplicaciones web. Rev. Cuba. Cienc. Inform. 2016, 10, 42–56.
7. Pecka, N.S. Making Secure Software Insecure without Changing Its Code: The Possibilities and Impacts of Attacks on the DevOps Pipeline. Ph.D. Thesis, Iowa State University, Ames, IA, USA, 2022.
8. Konstantinidou, C.A.; Lang, W.; Papadopoulos, A.M.; Santamouris, M. Life cycle and life cycle cost implications of integrated phase change materials in office buildings. Int. J. Energy Res. 2019, 43, 150–166.
9. Symantec. Symantec. Internet Security Threat Report. Available online: https://www.symantec.com/security-center/threatreport (accessed on 23 February 2023).
10. Diéguez, M.; Cares, C. Anticipation models (anti-models) for a proactive cyber defence. In Proceedings of the IX Congreso Internacional de Computación y Telecomunicaciones, Lima, Peru, 11–13 October 2017; pp. 247–254.

11. ISO. ISO/IEC27001. Information Security Management. Available online: https://www.iso.org/standard/82875.html (accessed on 23 February 2023).
12. ISO. NIST, Cybersecurity. Available online: http://www.iso.org/iso/catalogue_detail?csnumber=54533 (accessed on 20 February 2023).
13. ISACA. Control Objectives for Information and Related Technologies (Cobit). Available online: http://www.isaca.org/KnowledgeCenter/cobit/Pages/Products.aspx (accessed on 21 February 2023).
14. Ključnikov, A.; Mura, L.; Sklenár, D. Information security management in SMEs: Factors of success. Entrep. Sustain. Issues 2019, 6, 2081.
15. Meridji, K.; Al-Sarayreh, K.T.; Abran, A.; Trudel, S. System security requirements: A framework for early identification, specification and measurement of related software requirements. Comput. Stand. Interfaces 2019, 66, 103346.
16. Ansari, M.T.J.; Pandey, D.; Alenezi, M. STORE: Security threat oriented requirements engineering methodology. J. King Saud Univ.-Comput. Inf. Sci. 2022, 34, 191–203.
17. Mishra, N.; Pandya, S. Internet of things applications, security challenges, attacks, intrusion detection, and future visions: A systematic review. IEEE Access 2021, 9, 59353–59377.
18. López-Rodríguez, S.A.; García-Peña, V.R. Metodologías de desarrollo de software seguro con propiedades agiles. Polo Conoc. 2021, 5, 1027–1046.
19. Filus, K.; Domańska, J. Software vulnerabilities in TensorFlow-based deep learning applications. Comput. Secur. 2023, 124, 102948.
20. Kumar, R.; Goyal, R. On cloud security requirements, threats, vulnerabilities and countermeasures: A survey. Comput. Sci. Rev. 2019, 33, 1–48.
21. Sinha, A.; Das, P. Agile methodology vs. traditional waterfall SDLC: A case study on quality assurance process in software industry. In Proceedings of the 2021 5th International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech), Kolkata, India, 4–5 May 2021; pp. 1–4.
22. Futcher, L.; von Solms, R. SecSDM: A usable tool to support IT undergraduate students in secure software development. In Proceedings of the HAISA, Crete, Greece, 6–8 June 2012; pp. 86–96.
23. De Vicente Mohino, J.; Bermejo Higuera, J.; Bermejo Higuera, J.R.; Sicilia Montalvo, J.A. The application of a new secure software development life cycle (S-SDLC) with agile methodologies. Electronics 2019, 8, 1218.
24. Fowler, M.; Highsmith, J. The agile manifesto. Softw. Dev. 2001, 9, 28–35.

25. Croxford, M.; Chapman, R. Correctness by construction: A manifesto for high-integrity software. J. Def. Soft. Eng. 2005, 5–8.
26. Abundis, C.J.B. Metodologías para desarrollar software seguro. Recibe. Rev. Electron. Comput. Inform. Biomed. Electron. 2013, 3, 1–6.
27. Lindo, A.C. AC Modelos de Desarrollo Seguro del Software. 2023. Available online: https://web.fdi.ucm.es/posgrado/conferencias/AndresCaroLindo-slides.pdf (accessed on 23 February 2023).
28. Hudaib, A.; AlShraideh, M.; Surakhi, O.; Khanafseh, M. A survey on design methods for secure software development. Int. J. Comput. Technol. 2017, 16, 7047–7064.
29. Núñez, J.C.S.; Lindo, A.C.; Rodríguez, P.G. A preventive secure software development model for a software factory: A case study. IEEE Access 2020, 8, 77653–77665.
30. Microsoft. SDL—Agile Requirements. 2023. Available online: https://learn.microsoft.com/en-us/previous-versions/windows/desktop/ee790620(v=msdn.10)?redirectedfrom=MSDN (accessed on 27 February 2023).
31. BSIMM. BSIMM Frameworks. 2023. Available online: https://www.bsimm.com/ (accessed on 27 February 2023).
32. Chechik, M.; Salay, R.; Viger, T.; Kokaly, S.; Rahimi, M. Software assurance in an uncertain world. In Proceedings of the Fundamental Approaches to Software Engineering: 22nd International Conference, FASE 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, 6–11 April 2019; pp. 3–21.
33. Tawalbeh, L.; Muheidat, F.; Tawalbeh, M.; Quwaider, M. IoT Privacy and security: Challenges and solutions. Appl. Sci. 2020, 10, 4102.
34. Beznosov, K.; Kruchten, P. Towards agile security assurance. In Proceedings of the 2004 Workshop on New Security Paradigms, Virtual, 20–23 September 2004; pp. 47–54.
35. Tøndel, I.A.; Jaatun, M.G.; Cruzes, D.S.; Williams, L. Collaborative security risk estimation in agile software development. Inf. Comput. Secur. 2019, 27, 508–535.
36. Oueslati, H.; Rahman, M.M.; ben Othmane, L. Literature review of the challenges of developing secure software using the agile approach. In Proceedings of the 2015 10th International Conference on Availability, Reliability and Security, Toulouse, France, 24–28 August 2015; pp. 540–547.
37. Bhasin, S. Quality assurance in agile: A study towards achieving excellence. In Proceedings of the 2012 Agile India, Bengaluru, India, 17–19 February 2012; pp. 64–67.
38. Newton, N.; Anslow, C.; Drechsler, A. Information security in agile software development projects: A critical success factor perspective. In Proceedings of the 27th European Conference on

Information Systems (ECIS), Uppsala, Sweden, 8–14 June 2019.

39. Rindell, K.; Ruohonen, J.; Holvitie, J.; Hyrynsalmi, S.; Leppänen, V. Security in agile software development: A practitioner survey. Inf. Softw. Technol. 2021, 131, 106488.
40. Kramer, J.D. Developmental test and requirements: Best practices of successful information systems using agile methods. Def. AR J. 2019, 26, 128–150.
41. Villamizar, H.; Kalinowski, M.; Garcia, A.; Mendez, D. An efficient approach for reviewing security-related aspects in agile requirements specifications of web applications. Requir. Eng. 2020, 25, 439–468.
42. Sharma, A.; Bawa, R. Identification and integration of security activities for secure agile development. Int. J. Inf. Technol. 2020, 14, 1117–1130.