EPiC Series in Computing

Volume 58, 2019, Pages 107–116

Proceedings of 34th International Conference on Computers and Their Applications

Choosing the Best-fit Lifecycle Framework while Addressing Functionality and Security Issues

Salman M. Faizi[1] and Shawon S. M. Rahman[2]

[1] Ph.D. Candidate, Information Technology, Capella University, 225 South 6th St, Minneapolis, MN 55402, USA. salfaizi@outlook.com

[2]Associate Professor, Department of Computer Science and Engineering, University of Hawaii-Hilo, Hilo, HI 96720 sRahman@Hawaii.edu
and
Part-time Faculty, Capella University, 225 South 6th St, Minneapolis, MN 55402, USA

Abstract

Software application development must include implementation of core functionality along with secure coding to contain security vulnerabilities of applications. Considering the life cycle that a software application undergoes, application developers have many opportunities to include security starting from the very first stage of planning or requirement gathering. However, before even starting requirement gathering, the software application development team must select a framework to use for the application's lifecycle. Based on the application and organizational characteristics, software application developers must select the best-fit framework for the lifecycle. A software application's functionality and security start with picking the right lifecycle framework.

When it comes to application development frameworks, one size does not fit all. Based on the characteristics of the application development organization such as the number of application developers involved, project budget and criticality, and the number of teams, one of the five frameworks will work better than others.

Keywords: Software development lifecycle, software functionality, software security, application development, framework security

1 Introduction

Without a doubt, software applications are essential to the modern enterprise. Software applications face unique challenges and security threats. Cisco Systems [1] reported that in 2017, 64% of all denial of service attacks targeted applications. Attackers specifically target the applications because of the security vulnerabilities in them. The Open Web Application Security Project [2] identified the top ten most critical application security vulnerabilities: injection flaws, broken authentication, sensitive data exposure, XML external entities, broken access control, security misconfiguration, cross-site scripting, insecure deserialization, and insufficient logging and monitoring. The most effective way to address software application vulnerabilities is through secure coding practices during the life cycle of a software application [3]. By being aware of the application security vulnerabilities, software developers can design defenses against the vulnerabilities along with software functionality. Given the critical nature of software applications to the modern enterprise, the entire software development lifecycle must address vulnerabilities through the best-fit framework for the lifecycle. Thus, this study discusses best-fit frameworks for developing secure software applications that better address security vulnerabilities.

The organization of this paper is as follows. First, we describe the SDLC framework along with the incorporation of security in each phase. Next, we describe some of the current software security standards that are in use today. Next, we describe how various application development frameworks can be used for developing secure applications. Finally, we provide guidance on selecting the right framework.

2 Software Development Life Cycle

Software follows a clear lifecycle that covers all aspects of a software product from inception to retirement. Software development life cycle (SDLC) is a well-established, compressive framework for software development [4-6]. Figure 1 below shows the stages of SDLC: (a) planning, (b) analysis, (c) design, (d) implementation, (e) testing and integration, and (f) maintenance.

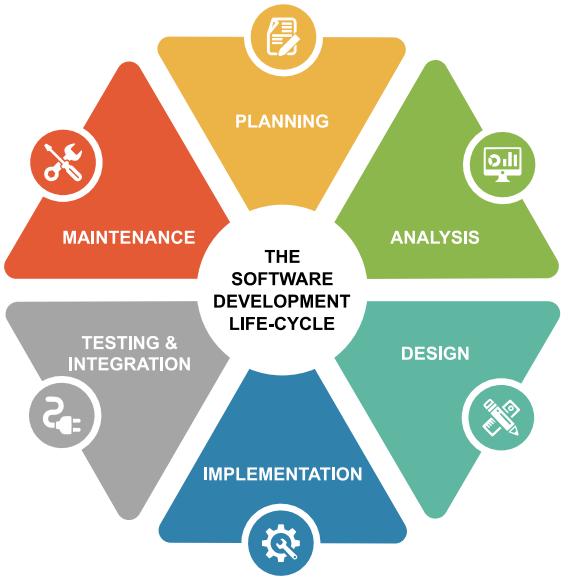

Figure 1: Stages of Software Development Life Cycle

SDLC is not a set of sequentially ordered phases; instead, a software development effort in a particular SDLC phase may return to an earlier phase if necessary, some phases may be skipped, a phase may be completed in parallel with another, or completed iteratively [5]. SDLC can follow several approaches or frameworks to trace the lifecycle of a software product. Software application developers can add security during any of the six stages of SDLC, and security need not be an afterthought. Studies have shown that software development efforts focus on functionality and usability and do not explicitly include cybersecurity in the SDLC process [7]. Due to competing priorities, software application developers often only focus on what they consider the core functionality and leave out security implementation for later stages [8, 9]. However, each stage of SDLC lends itself to including security. The following table, adapted from Karim et al. [7], shows some ways in which software application developers can incorporate security in each stage of SDLC.

Planning	**Analysis**	**Design**	**Implementation**	**Testing and Integration**	**Maintenance**
User Stories	Misuse Case	Threat Model	Security Modules	Unit Testing	Security Management Procedures
Functional Security Requirements	Mitigation Plan	Input Data Types	Known Security Vulnerabilities	Functional Testing	Monitoring Requirement
User Requirements		Security Use Cases		Penetration Testing	Security Upgrade
Non-Functional Security Requirements		Security		Fuzz Testing	Regression Testing

Table 1: Security Considerations in each stage of the Software Development Life Cycle

3 Software Security Standards

There are standards and guidance that address the security practice to help the software application development effort. Following is a list of examples of such standards and guidelines: (a) ISO/IEC 15408 provides an evaluation of IT Security via measures that are generally useful within the international community [10, 11]; (b) ISO/IEC 27001 is a best practice framework for the administration of data security, which highlights the threats to critical data and sets up controls to manage the threats [12, 13]; (c) SSE-CMM or Systems Security Engineering-Capability Maturity Model provides characteristics that an organization must ensure in its security engineering process to deliver effective security engineering [14, 15]; (d) ISO/IEC 21827 is centered on the Systems Security Engineering Capability Maturity Model (SSE-CMM) to help organizations identify security goals, support security lifecycle, and assess security posture [16, 17]; and (e) OWASP or Open Web Application Security Project's Top Ten while not a standard, is a frequently updated list of the ten most critical web application security risks which also provides some guidance on how to protect against the top ten critical web application security risks [18, 19]. Software application developers can refer to the standards and guidance above to bolster the security of their applications. For example, SSE-CMM along with ISO/IEC 15408 can provide concrete guidance for secure code development. Mesquida and Mas [20] carefully mapped relations between ISO/IEC 15504-5 software development base practices and ISO/IEC 27002 security controls to detail the changes that software application developers should make to their software development lifecycle to incorporate security controls. Thus, standards can provide a host of best practices to increase applications security.

4 Software Application Development Frameworks

In practice, the software development life cycle (SDLC), as shown in Figure 1, may follow several available frameworks for software application development. Vijayasarathy and Butler [21] found practitioners predominantly utilize four frameworks for software application development: Agile, traditional or waterfall, iterative, and hybrid. Additionally, [22] Security Development Lifecycle (SDL),

a specialized adoption of SDLC, is also popular [6]; this paper will also include it. Thus, the five frameworks that this paper will consider are Agile, traditional or waterfall, iterative, hybrid, and SDL.

4.1 Agile

Agile software application development framework promotes building software in small chunks with each chunk delivering incremental value. Some popular Agile methodologies include adaptive software development, agile modeling, agile unified process, crystal clear, scrum, scrum ban, and others [23]. Agile methodologies deliver software application frequently in short iterations to the customer. Not only does the customer receive working software early in the development lifecycle but the software applications also stay closely aligned to customer needs [24]. Software application developers can achieve alignment with the customer needs quickly in case the software deviates from customer needs.

Agile software development lends to the security of software applications. Each iteration of the Agile software development can include security requirements, but it requires the presence of an information security expert on the team [25]. The Agile application development team can maintain focus on security throughout the development process. The security requirements are analyzed, developed, and tested along with the development of other functions of the software application. Thus, Agile software development provides an opportunity for continuous focus on security.

In our experience, we have found that Agile framework provides benefits in increasing application development security by following an iterative approach. The application development teams can include security considerations in each iteration and evaluate their effectiveness and adjust as needed. Additionally, we observed that during application development each iteration provides additional clarity for the development team to fine-tune their approach to secure applications.

4.2 Waterfall or Traditional

Waterfall or traditional framework for software application development is the oldest and most well-known SDLC framework. The characteristic feature of this framework is the approach that is based on sequential steps. The approach is to go through phases in a step-by-step manner starting with planning and requirements then analysis, design implementation, testing and integration, and maintenance [26]. A phase must complete before the next phase can be started, Waterfall framework requires detailed planning, and the final product delivery takes an extended duration; thus, the customer does not see the software application until it is complete. In our experience with the Waterfall framework, the disciplined approach to upfront planning helps bring focus to security for applications. However, the long development cycles can mean evaluation of the security approach for the applications occurs late in the development cycle. Additionally, the delayed customer contact can lead to the deviation of the software application from the customer needs.

The waterfall framework provides an opportunity to include security in all its phases, but there is a risk that security may be overlooked. Studies have shown that software development efforts focus on functionality and usability and can often exclude security in the SDLC process [7]. However, if the success criteria for software development include functionality and usability, as well as security, the software application developers will include it. Due to competing priorities, it is challenging for software developers to deliver on all elements of the success criteria unless their leaders require security along with other elements [8, 9]. Thus, the waterfall framework provides plenty of opportunities for including security, but security must be a required element for software developers to include it.

4.3 Iterative

This framework refers to several methodologies that are iterative. Iterative methodologies include Rational Unified Process (RUP), Joint Application Development (JAD), and Rapid Application Development (RAD) [21]. RUP is an adaptable process framework to allow software teams to select elements that are appropriate for their needs. The goal of RUP, a disciplined method of assigning tasks and responsibilities within a software development organization, is to deliver high-quality, customer-focused software within a predictable schedule and budget [27]. Joint application design (JAD) is an approach where users and software application developers collaborate on information system planning, design, and other activities [28]. RAD focuses on prototyping and iterative development without the need for detailed planning since, in RAD, software application development contains the required planning [29]. Thus, the iterative framework is a collection of several methodologies that follow an iterative approach to developing a software application. The iterative framework lends itself to including information security in the software. Shirazi et al. [30] provide an approach called RUPSec that focuses on using RUP to develop secure software systems to address security threats. Similarly, other methodologies in the iterative framework can include security.

4.4 Hybrid

The hybrid framework includes software application development efforts that combine methodologies from frameworks. There are situations where a combination of software application development frameworks is employed. For example, someone may use JAD for requirement gathering and Waterfall for implementation. A small software development team developing highly critical applications can benefit from the hybrid framework [21]. Another reason to use the hybrid framework could be the presence of legacy information systems with established methodologies along with newer information systems that follow a different methodology. The software development teams must include focus on security.

4.5 Microsoft's Security Development Lifecycle (SDL)

Secure software application development is increasingly becoming essential. A comprehensive approach to secure software application development is Microsoft's SDL [31]. The middle five phases of SDL roughly correspond to SDLC's first three phases; requirement, design, and implementation phases of SDL correspond to analysis, design, and implementation [5]. The diagram below, from Microsoft [22], illustrates SDL.

SDL is employed as a sub-step in the overall software development lifecycle, and there is a higher-level lifecycle process that takes care of planning and maintenance. The SDL phase of Training is to allow the development team to learn about security basics and trends [5] because a typical developer is not likely to have a deep understanding of cybersecurity. Finally, the SDL phase of Response does not have an equivalent in SDLC, and it is intended to address security threats to a particular product [5, 31]. Thus, SDL is a specialized adoption of SDLC.

In our experience with the SDL framework, we observed that developers often do not have time to devote to the disciplined approach unless the use of the SDL framework is mandated. The requirement to train the application developers in security expertise and the development of threat models were beneficial when the application developers followed them.

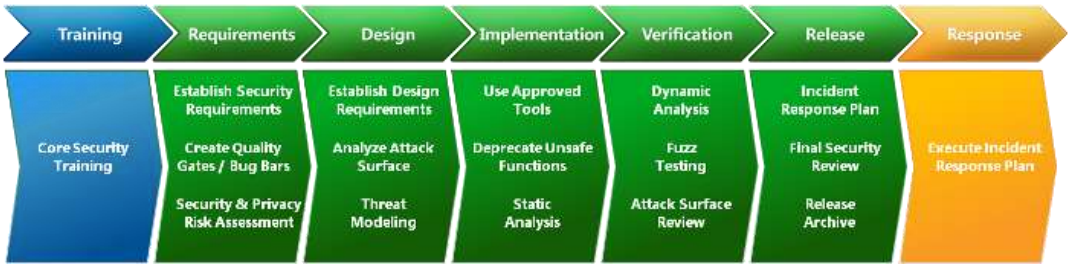

Figure 2: Microsoft's Security Development Lifecycle (SDL)

5 Choosing the Right Framework

This paper has presented five different frameworks. The question is which one an application development team should use. Vijayasarathy and Butler [21] recommend categorizing software application development projects along three factors: (a) organizational, (b) project, and (c) team, and based on their study, they described what criteria should make up the each of the three factors. First, the organizational factors include annual revenue and number of employees. Second, project factors include the budget allocated to the project and the criticality of the project to the company. Finally, team factors include the number of teams involved in a project and the number of individuals on a team. The table below adopted from Vijayasarathy and Butler [21] can help software application developers choose the best-fit framework for their software application.

	Characteristics		
Framework	**Organizational**	**Project**	**Team**
Agile	Moderate revenue. A small number of employees.	Low budget. Medium to high Criticality.	One team. Small team.
Traditional	High revenue. A large number of employees.	High budget. High criticality.	Multiple teams. Medium team.
Iterative	A small number of employees.	Medium budget. Medium to high criticality.	One team. Small team.
Hybrid	Organization size unimportant.	Medium budget. High criticality.	Small team.
SDL	Organization size unimportant.	High budget due to higher overhead. High criticality.	One to multiple teams.

Our view is that the best-fit framework applies to the functionality of the application as well as its security. In an application development project that we were part of, the developers would only include security aspects in the application unless security was required to clear the security review without which the developers could not release the application. Since security review boards tend to reject applications that do not adequately address security vulnerabilities, the application developers must come back to the security review board repeatedly after including additional security features in their application. We observed that often the application developers did not fully understand the security requirements and the security review board did not fully understand the application. In this case, the Agile methodology proved effective. With each iteration, the application developers understood security requirements better and the review board understood the application better. This increased understanding lead to implementation of specific measures to address security vulnerabilities and better specification of security requirements. Thus, when uncertainty is high, we recommend using the Agile framework.

6 Conclusion

Software applications are not only critical to an organization due to the functionality they provide but also expose the organization to security vulnerabilities since they are often an easy target for hackers. Before application developers start the first step in the software development lifecycle, they must select a best-fit lifecycle framework. The best-fit framework is the one that is the most appropriate for the characteristics of an organization, its software application development project, and the team developing the application. Once the developers have selected the best-fit lifecycle framework, they must then take steps to include security in every stage of the lifecycle. The organization must emphasize security along with the functionality to make sure that the developers prioritize it during development.

References

[1] Cisco Systems, "Cisco Annual Cybersecurity Report," Available: https://www.cisco.com/c/dam/m/digital/elq-cmcglobal/witb/acr2018/acr2018final.pdf?dtid=odicdc000016&ccid=cc000160&oid=anrsc005679&ecid=8196&elqTrackId=686210143d34494fa27ff73da9690a5b&elqaid=9452&elqat=2

[2] The Open Web Application Security Project, "OWASP top 10 - 2017: The ten most critical web application security risks," 2017, Available: https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf.

[3] M. Whitney, H. R. Lipford, B. Chu, and T. Thomas, "Embedding Secure Coding Instruction Into the IDE: Complementing Early and Intermediate CS Courses With ESIDE," *Journal of Educational Computing Research*, vol. 56, no. 3, pp. 415-438, 2017.

[4] M. Dawson, D. N. Burrell, E. Rahim, and S. Brewster, "Integrating software assurance into the software development life cycle (SDLC)," *Journal of Information Systems Technology & Planning*, vol. 3, pp. 49-53, 2010.

[5] J. S. Valacich and J. F. George, Modern Systems Analysis and Design, 8th Edition [VitalSource Bookshelf version]. 2016. [Online]. Available: https://bookshelf.vitalsource.com/books/9781323576656.

[6] M. Ramachandran, "Software security requirements management as an emerging cloud computing service," *International Journal of Information Management*, vol. 36, pp. 580-590, 2016.

[7] N. S. A. Karim, A. Albuolayan, T. Saba, and A. Rehman, "The practice of secure software development in SDLC: An investigation through existing model and a case study," *Security & Communication Networks*, vol. 9, pp. 5333-5345, 2016.

[8] A. Rehman and T. Saba, "Evaluation of artificial intelligent techniques to secure information in enterprises," (in English), *The Artificial Intelligence Review*, vol. 42, no. 4, pp. 1029-1044, Dec 2014 2014-11-14 2014.

[9] A. S. Sodiya, S. A. Onashoga, and O. B. Ajayi, "Towards building secure software systems," *Issues in Informing Science and Information Technology*, vol. 3, pp. 635-645, 2006.

[10] International Organization for Standardization. (2009, 10/28/2018). *Information technology — Security techniques — Evaluation criteria for IT security — Part 1: Introduction and general model*. Available: https://www.iso.org/standard/50341.html

[11] G. Troy, "Common criteria becomes ISO international standard 15408," (in English), *Journal of Research of the National Institute of Standards and Technology*, vol. 104, no. 5, p. 507, Sep/Oct Sep/Oct 1999.

[12] International Organization for Standardization. (2013, 10/28/2018). *ISO/IEC 27000 family - Information security management systems*. Available: https://www.iso.org/isoiec-27001-information-security.html

[13] A. Kurnianto, R. Isnanto, and W. Aris Puji, "Assessment of information security management system based on ISO/IEC 27001:2013 on subdirectorate of data center and data recovery center in ministry of internal affairs," Les Ulis, 2018, vol. 31: EDP Sciences.

[14] H. Yang, H. Kim, and H. Chang, "A study on industrial security experts demanding forecasting in intelligent sensor network," (in English), *International Journal of Distributed Sensor Networks*, 2015 2016-08-21 2015.

[15] Systems Security Engineering-Capability Maturity Model. (2018, 10/28/2018). *Cybersecurity Engineering in Company*. Available: http://www.sse-cmm.org/

[16] International Organization for Standardization. (2008, 10/28/2018). *Information technology -- Security techniques -- Systems Security Engineering -- Capability Maturity Model® (SSE-CMM®)*. Available: https://www.iso.org/standard/44716.html

[17] S. Paul and R. Vignon-Davillier, "Unifying traditional risk assessment approaches with attack trees," *Journal of Information Security and Applications*, vol. 19, no. 3, pp. 165-181, 2014.

[18] The OWASP Foundation, "OWASP Top 10 - 2017: The Ten Most Critical Web Application Security Risks," 2017, Available: https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf.

[19] K. Tuma, G. Calikli, and R. Scandariato, "Threat analysis of software systems: A systematic literature review," *Journal of Systems and Software*, vol. 144, pp. 275-294, 2018.

[20] A. L. Mesquida and A. Mas, "Implementing information security best practices on software lifecycle processes: The ISO/IEC 15504 Security Extension," *Computers & Security*, vol. 48, pp. 19-34, 2015.

[21] L. R. Vijayasarathy and C. W. Butler, "Choice of software development methodologies: Do organizational, project, and team characteristics matter?," *IEEE Software*, vol. 33, no. 5, pp. 86-94, 2016.

[22] Microsoft. (2018, 10/27/2018). *Microsoft Security Development Lifecycle*. Available: https://www.microsoft.com/en-us/SDL

[23] O. C. Buturugă, V. M. Gogoi, and I. A. Prodan, "Agile project management tools," (in English), *Academy of Economic Studies. Economy Informatics*, vol. 16, no. 1, pp. 19-26, 2016 2018-01-18 2016.

[24] G. Coleman, "Agile Software Development," (in English), *Software Quality Professional*, vol. 19, no. 1, pp. 23-29, Dec 2016 2017-02-10 2016.

[25] P. Maier, Z. Ma, and R. Bloem, "Towards a secure scrum process for agile web application development," presented at the Proceedings of the 12th International Conference on Availability, Reliability and Security, Reggio Calabria, Italy, 2017.

[26] A. Alshamrani and A. Bahattab, "A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model," (in English), *International Journal of Computer Science Issues (IJCSI)*, vol. 12, no. 1, pp. 106-111, Jan 2015 2015-03-05 2015.

[27] Rational Software Corporation, "Rational Unified Process - Best Practices for Software Development Teams," pp. 1-21Accessed on: 10/29/2018Available: https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

[28] E. J. Davidson, "Joint application design (JAD) in practice," *Journal of Systems and Software*, vol. 45, no. 3, pp. 215-223, 1999.

[29] A. Kazim, "A study of software development life cycle process models," (in English), *International Journal of Advanced Research in Computer Science*, vol. 8, no. 1, Jan 2017.

[30] M. R. A. Shirazi, P. Jaferian, G. Elahi, H. Baghi, and B. Sadeghian, "RUPSec: An extension on RUP for developing secure systems - requirements discipline," *International Journal of Computer and Systems Engineering*, vol. 1, no. 4, 2007.

[31] M. Felderer and B. Katt, "A process for mastering security evolution in the development lifecycle," (in English), *International Journal on Software Tools for Technology Transfer*, vol. 17, no. 3, pp. 245-250, Jun 2015 2015-06-23 2015.