Rethinking Security in Agile Development Methodology

Aastha Dhamija

Seattle Pacific University

Author Note

Aastha Dhamija, School of Business, Government and Economics

Contact: dhamijaa@spu.edu

Abstract

Agile development methodologies are gaining traction in the market for their nimbleness to adapt to changing customer requirements and providing working software in short timelines. It focuses more on human interactions and changing world and less on processes or defined contracts, which makes it attractive to provide quick prototypes but lacks quality and security constraints. Agile wants things to be simple but security assurance is considered as a lengthy and complex process, so how do we incorporate these poles apart approaches into each other and make agile more secure? This paper is trying to find those clashes and synergies in agile and security through existing research and make agile development methodology appropriate for mission-critical projects.

Keywords: Agile, Security, Abuse cases, Spikes, Security Analyst

Rethinking Security in Agile Development Methodology

With the current paradigm of digitization and constantly changing needs of various businesses, the software industry has found its solace in Agile Software Development Methodology. Agile gives them the flexibility to incorporate change in the development process at regular intervals and measures the team's progress in terms of quick working software, both of which are a win-win situation for any customer. Hence for competitive reasons, almost all software businesses use agile as the preferred method for software development, which is good for maintaining flexibility but lacks proper security measures. Security being a complex subject requires thorough analysis and comprehensive documentation, which goes against the agile manifesto of achieving quick working prototypes and keeping documentation to a minimum.

Security considerations also change drastically within each domain and depending on the sensitivity of the product or application but are rarely incorporated in the fast-moving cycle of agile methodology. For example, security considerations in case of building a product in healthcare department which is directly linked to patients' well-being would focus more on quality and safety of the product over fast working prototype - hospitals would not test a half-baked product on any patient's life to get constructive feedback. Similar would be a case for making an application for national security which again would focus more on quality than quickness of product delivery. Earlier these security considerations were limited to the above-mentioned domain-specific or sensitive conditions, but with ongoing digitization and increasing rate of cyber-attacks, it has become a necessity in every application. Adding a layer of security post completion of the development process usually, result in sub-quality products and compromise on the depth of required security as per Siponen et. al (2005).

The current security measures are lengthy in nature, require a lot of documentation and are considered as an extra effort. This paper focuses on identifying the security requirements which can be incorporated into the agile software development lifecycle with minimum disturbance while keeping all its positive features for maintaining flexibility with higher quality and security assurances. The paper is organized as follows: Section 2 – Background and Literature review covers a brief of all published research papers and articles which formulate the outline of this paper, followed by section 3 which highlights the clashes between traditional security assurance and agile methodology and uncovers the security requirements (from both system and human perspective) for maintaining agility. Section 4 highlights the new additions which can be done in the existing requirement collection phase of agile methodology and how it can be incorporated in the software development life cycle. And the last section talks about the conclusion and scope for future research.

Background and Literature Review

This section provides the background on the core subjects of this paper i.e. Agile development methodology, security assurance practices and their experimental methods of correlation through published studies and research papers.

Agile Methodology

For the purpose of this paper, Agile manifesto and its 12 principles as mentioned in Agileallinace.org has been taken as the base and it defines

“Agile as the ability to create and respond to change and talks about Agile philosophy of choosing working software over comprehensive documentation, individuals and interactions over tools and processes, customer collaboration over contract negotiation and responding to change over following a plan”.

They consider Agile as a mindset and focus more upon the nimble yet effective aspects of software development and prioritize human factor over defined processes.

Security Assurance

For the purpose of this paper, the Microsoft Security Development lifecycle (SDL) and Open Web Application Security Project (OWASP) Secure coding practices have been taken into consideration. These approaches or methodologies provide best practices, validation checklists, inspection tools and much more for incorporating and validating security aspects in the development lifecycle. The practices mentioned above are highly effective in maintaining security but are not nimble enough to be incorporated in agile methodology.

Experimental studies on incorporating security into agile methodology

Despite agile being used for mission-critical products or applications, corresponding security considerations are still very low. A lot of researchers have tried to reconcile the security approach into an existing agile methodology. Beznosov & Kruchten (2004) talks about similarities and clashes of agile methodology and traditional security assurance methods and provides ways of how to convert those clashes into synergies. Siponen et. al (2005) talks about agility essentials for security requirements and how information can be classified as security subjects and objects and how they can be incorporated in all phases of software life cycle development from requirements analysis to testing phase. Assal & Chiasson (2018) talks about the human aspect of security in the development life cycle and how developers are viewed as the weakest link in the security chain and what is the impact of external factors like organizational culture, security knowledge and experience of a security incident, have on end security assurance. Sonia & Singhal (2014) focuses on the weighted average method to identify which security practices can be termed as both agile

and effective and remove the remaining ones. And lastly, Mougouei et. al (2013) introduces the concept of security spikes and how to incorporate that in agile - scrum life cycle.

Security essentials for Agile Methodology

This section highlights the clashes between Agile Methodology and traditional security assurance methods and derives security requirements from both system and human perspective to be considered in agile development methodology:

Clashes between Agile Methodology and traditional Security Assurance

Even the idea of incorporating security into agile methodology is conflicting as agile promises quick working prototypes with simple and iterative development with minimum documentation and the ability to change rapidly. On the contrary, security methods are complex to understand, require lengthy documentation in terms of checklists and validations and do not deal well with continuous changes, as to date security is considered as a post-development audit procedure. Beznosov & Kruchten (2004) noted that though most of the security practices are in conflict with agile methodology, there are still a few which are aligned with the agile development process and some which can be automated or modified a bit to fall in that league. Classification is given as follows:

1. **Matches -** Pair programming, code review by following coding standards. This method incorporates the peer to peer communication which gives immediate feedback and encourages both agility and security
2. **Independent of development methodology** - Version control and change tracking tools. These tools are used in all development methodology be it agile or waterfall and ensures step by step security logging.

3. **Possible Automation** - penetration testing, system testing for known vulnerabilities, standard code review for security standards. All these can be automated in-house or using pre-approved tools without creating significant budgetary or time overheads.
4. **Mismatch** - all the security techniques which have a heavy reliance on extensive documentation fall under this category like formal/ informal validation, change authorization, external review, etc.

Hence for incorporating security in the agile development life cycle, security methods or approaches should follow the below-mentioned system and human aspects.

System essentials for software security

1. Security techniques should be effective, adaptive to current agile methodology and easily incorporated into its existing life cycle
2. Security techniques should be adaptable to changing requirements - thereby satisfying the current needs to ongoing businesses, maintaining customer satisfaction and nimble enough to adapt to changes
3. Security techniques should be simple, intuitive and not hinder the current development life cycle - leads to greater adoption

Human aspects of software security

Assal & Chiasson (2018) identifies “Developers as the weakest link in the security chain” and showcases the impact of external factors such as organization culture, security knowledge, and experience of prior security incidents, etc., on security assurance. It further says that developers focus majorly on functional requirements and become blind to security imperfections and it takes an extra effort to make them interested in security aspects of software development.

1. Incorporate quality and security in organizational culture by including security training for everyone and especially for developers at all levels
2. Include a Security analyst along with business analysts at the requirements definition stage to make both the customer and development team aware of the risks and possible harm it can cause to amplify the impact.

For instance, while building a banking application, project manager always prefers to have developers from banking domain to understand the nitty-gritty of that solution and make domain-specific test cases for checking effectiveness and efficiency of the built solution, similarly having a security engineer or security trained engineer would support in highlighting the underlying risks in time, incorporating them in the workflow and proposing rectifying actions as well. Thus, considering all the above-mentioned facts, security should be considered at the start of the project lifecycle to incorporate it as an essential requirement or feature rather than an afterthought. The next section of the paper talks about the changes that can be done in the requirements definition phase of the agile software development life cycle to start the product building with a security first attitude.

Key Security elements in Agile Software Development

As per Microsoft SDL practices, the optimal time to define the security requirements is during the initial design and planning stages as this allows development teams to integrate security in ways that minimize disruption.

Siponen et. al (2005) defines key security elements that can be utilized further for integrating security right at the start of agile software development. They propose the following classification:

1. **Security objects:** In the requirement analysis phase, identifies key assets of the organization in discussions with the customer and then all use cases with those sensitive assets are denoted as “security enriched use cases”
2. **Security subjects:** Identify authorized persons or people who have the highest access and may accidentally compromise software security. These are again termed as security enriched use cases. Defines security-relevant objects and actors who are potential security subjects.
3. **Security classification and risk management:** Security sensitivity classification by defining which security subject has access to which security object and mark them as “classified” and rank their security priority

For highly sensitive use cases as per the above classification, security analysts can formulate **abuse cases** that identify potential threat scenarios and help identify unauthorized actions. These abuse cases will help capture risks and their probable mitigation plans and can be prioritized by estimating the cost and frequency of occurred attack. Its feasibility can be determined by estimating the cost of recovery and customer importance.

The abuse case can be viewed in Figure 1. It is an updated form of a use case which has been modified to capture security information.

Use Case Name: Access student sensitive data | **Security Object**		UC-1	Priority: High | **Security 1**
Actor | **Security Subject**	**Hacker | Internal User**		
Description			
Trigger			
Type	External	Temporal	

Preconditions	
Normal Course	1. **Highlight Secure objects it is touching**
Postconditions	1. **What security flaw can be uncovered**
Exceptions	
Risks	
Frequency of attack	
Mitigation plan	
Cost of recovery	
Priority	

Figure 1. Abuse Case Template

Now to implement these abuse cases in the ongoing agile lifecycle, Mougouei et. al (2013) introduced the concept of "**Security spikes".** Spikes are introduced in the agile development lifecycle to accommodate development activities that do not have a tangible customer valued end product. For example, if the development team is working on new technology and need some time to do deep dive in that technology space, then a technological spike is introduced in the release cycle in a time-boxed manner. It does not have a tangible end output, but it helps increase the development team's velocity in other sprints. Similarly, there are functional spikes which are introduced to understand a complicated functional spec into more detail by maybe making a prototype or doing additional research in a time-boxed manner.

In this manner, it would be beneficial to introduce security spikes in release cycles to create or expand abuse cases for classified security use cases. And once these abuse cases become a part

of product backlog then it runs as a normal agile life cycle without hindering its regular course of action.

Future research scope and Conclusion

It is clear from the paper that software development methodology has modernized to cater to changing requirements but are still stuck in traditional security methods. This paper highlights the need for security in agile development methodology, showcases the clashes between traditional security assurance practices and rapidly changing agile methodology and derives system and human aspects of how agile security should be. This is followed by the definition of key security elements like security object, subject and classification and how we can incorporate the position of the security analyst to produce abuse cases in the requirement analysis phase itself and further expand and incorporate those in the agile life cycle through security spikes.

Future scope: We are still stuck in traditional security assurance methods and trying to modify those to accommodate into the agile life cycle, maybe it's time to discover entirely new security methods as we moved from waterfall to agile and this time make security also as human-centered instead of process/tool centered.

References

Assal, H., & Chiasson, S. (2018). Security in the Software Development Lifecyle. *Symposium on Usable Privacy and Security (SOUPS).* Baltimore.

Beznosov, K., & Kruchten, P. (2004). Towards Agile Security Assurance. *New Security Paradigms Workshop (NSPW).*

Mougouei, D., Mohd Sani, N., & Almasi, M. (2013). S-Scrum: a Secure Methodology for Agile Development of Web Services. *World of Computer Science and Information Technology (WCSIT)*, 15-19.

Siponen, M., Baskerville, R., & Kuivalainen, T. (2005). Integrating Security into Agile Development Methods. *Hawaii International Conference on System Sciences (HICSS).* Hawaii: IEEE Xplore.

Sonia, & Singhal, A. (2014). Selection of Security Activities for Integration with Agile Methods after Combining thier Agility and Effectiveness. *International Journal of Web Applications (IJWA)*.

OWASP Top 10. (2017). (n.d). Retrieved from https://owasp.org: https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf

Agile Manifesto. (n.d). Retrieved from https://www.agilealliance.org/agile101/the-agile-manifesto/

Security Engineering. (n.d). Retrieved from https://www.microsoft.com/en-us/securityengineering/sdl/practices