P: ISSN NO.: 2394-0344 RNI No.UPBIL/2016/67980 VOL-2* ISSUE-11* February- 2018
E: ISSN NO.: 2455-0817 *Remarking An Analisation*

# Integrating Security in Agile Software Development

**K.C. Joshi**
Computer Centre,
Faculty of Education and Allied Sciences,
M.J.P. Rohilkhand University,
Bareilly

**Rajat Rajput**
Computer Centre,
Faculty of Education and Allied Sciences,
M.J.P. Rohilkhand University,
Bareilly

**Abstract**

Agile is popular approach in software development. Agile software development refers to a group of software development methodologies based on iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams. Agile methodologies provides better results in return on investment, quality, team morale stakeholder satisfaction and delivery time.Security has can not be given the attention it needs when developing software with Agile methods. Now a days there are strong regulatory and privacy requirements to protect private data, security must be treated as a high priority.

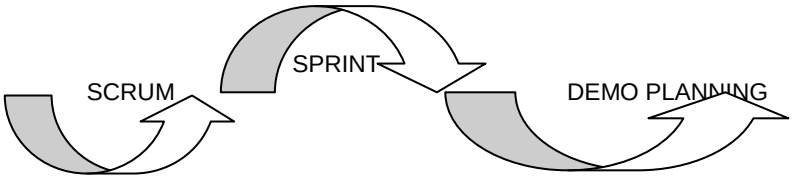
Agile software development is a set of techniques for building intended software with fewer errors and better predictability. Each technique or approach is intended to address well-known weaknesses in traditional 'Waterfall' development: complexity, poor communication, and infrequent code validation.

In this paper we will discuss on agile methodologies security, extreme programming, agile manifesto, future aspects and technological development of agile computing.

**Keywords:** Agile, Extreme Programming, Scrum, Agile Development, SDLC.
**Study Duration:** The papers reference periods are from 1994 to 2015.

**Introduction**

Microsoft use Agile software development and management methods to build their applications. Now a days highly inter connected world, where their are strong regulatory and privacy requirements to protect private data security must be treated as a high priority. The strength of Agile is that it can save organization significant amounts of development time and money, while still allowing them to deliver high quality software. Agile practices are related to improved product quality customer satisfaction, and developer productivity than traditional waterfall practices. Agile practices have a significant impact in developing software in recent few years. Agile cycle includes includes Scrum, Sprint and demo-planning. Agile cycle are short and there is no directed mapping from Agile cycles to Waterfall stages.




The Fundamental characteristics of software development approaches:

1. Specification, design and implementation are interleaved.
2. The system is developed in a series of versions.
3. User interfaces are often developed in an interaction development environment that allows for quick redesign.

Agile methods are not well suited for critical and complex system. Things that may come the way of and development:

1. Customers are not able or willing to participate
2. Team members are not comfortable for intense involvement
3. Prioritizing changs can be difficult
4. Maintaining simplicity requires extra work.

**Aim of the Study**

The aim of the study is to make a lot interest in the field of software engineering and the caught the attention of software engineers an researchers world wide. Also the aim of this paper is to organise, analyase and make sense out of the dispersed field of agile software development

methods.

**Review of Literature**

Agile Methods are actually a collection of different techniques (or practices) that share the same values and basic principles. Many are, for example, based on iterative enhancement, a technique that was introduced in 1975. Another important model to take into account in these discussionsis the Capability Maturity Model (CMM). In the '80s, developing a software program meant adhering to the then-prevalent Software Development Life Cycle (SDLC) or Waterfall Methodology. The need was to adopt software development methodologies which were 'lightweight', had scope for changes during the development, were iterative in nature and involved frequent feedbacks.

Independently, Kent Beck rediscovered many of these values in the late 1990s when he was hired by Chrysler to save their failing payroll project, Chrysler Comprehensive Compensation (C3).

Jeff Sutherland and Ken Schwaber conceived the scrum process in the early 1990s. Scrum was codified in 1995 in order to present it at an object-oriented conference in Austin, Texas. They published it in the form of a paper titled "SCRUM Software Development Process."

Perhaps various agile and iterative techniques would still be in the minority were it not for the Agile Manifesto, codified at that 2001 meeting in Snowbird.

In the mid 1990s.Earlier known as lightweight, these methodologies were soon put under an umbrella called Agile development in 2001 at the Snowbird Ski Resort, Utah, where these 17 thought-leaders came together for the first time. Thus came into existence the Agile Manifesto , which uncovered better ways of developing software.

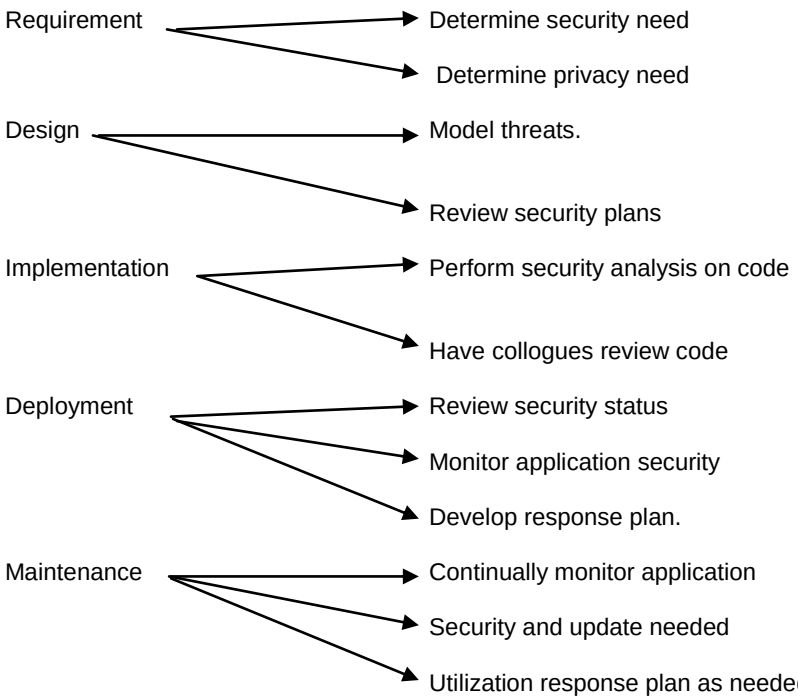
At the Better Software/Agile Development West conference in 2011, Kent Beck presented a keynote on how to accelerate software delivery. He gave this one-hour talk without ever once mentioning agile.

**Characteristics of Agile Methods**

1. It is assume that the size of project is small size.
2. Continuous re-factoring.
3. Current version focus.
4. Customer commitment.
5. The importance of individuals.

**SDLC and Agile Waterfall Models**

Several well known SDLC integrate security in different ways.Security can be integrated into any there phases. In most organization that use a variant of the waterfall model. The SDLC with developers working with security can be seen as:




One of toughest part of bringing security into an Agile programs is process modification. Agile includes whatever work gets done in a sprint - it does not bend to security, so you need to bend security to fit Agile. The diagram show a waterfall development process, listing the sequence of steps, can perform by a different team.

Architecture → Design → Development → Coding/Debugging → Quality Assurance → Release Management

Here design consists threat modeling, development consists static analysis, regression and testing, quality assurance consists fuzzing & exception testing and release management consists firewall policy, patches the deployment environment.

**Major Agile Methodologies**

**SCRUM**

The scrum framework is widely used in the public and private sector and its terminology is often used a Agile discussion e.g. Scrum iteration are called sprints, which are bundled in releases.

Scrum was based on the concept that for the development of new, complex products, the best results occur when small and self-organizing teams are given objectives rather than specific assignments. The team had the freedom to determine the best way of meeting those objectives. Scrum also defined time-boxed iterative development cycles whose goal was to deliver working software. Today, most teams that claim to practice an agile methodology say they're using scrum. SCRUM works on the principles of continuous improvement, teamwork and empiricism.

**Extreme Programming (XP)**

Extreme Programming (XP) is an intense, disciplined and agile software development methodology focusing on coding within each software development life cycle (SDLC) stage. Extreme programming (XP) is a software development methodology that combines many of the best practices in software development, and then takes them "to the extreme". XP also requires constant communication between the customer and the developer team, as well as among the developers. Extreme programming is one of the concepts of agile methodology.XP programmers start to generate codes at the very beginning so “At the end of the day, there has to be a program.”XP uses Unit Tests which are automated tests, and the programmer will write tests as many as possible to try to break the break the code he or she is writing. A person can work on XP as- programmer, customer, teacher and coach.

There are 12 major practices in XP - planning game, small release, metaphor, simple design, testing,re factoring, pair programming collective ownership, continuous integration, 40 hours week, on side customer, coding standards.

Extreme Programming (XP) is a software engineering methodology, the most prominent of several agile software development methodologies.

The goal of extreme programming is being described as follows:

An attempt to reconcile humanity and productivity

1. A mechanism for social change.
2. A path to improvement.
3. A style of development.
4. A software development discipline.

The basic activities perform by XP can be studied as follows:

**Coding**

In XP coding is considered the only important product of the system development process.

**Testing**

XP emphasizes to always check if a function works is by testing it.

**Listening**

This ability will enable them to understand what customers want and develop solutions which match customers’ needs and desires as close as possible.

**Designing**

Without proper design in the long run system becomes too complex and projects could come to a halt. It is then important to create a design structure that organizes the logic in the system so too many dependencies in the system can be avoided.

**Agile Methodologies and Security**

Agile methods or agile processes generally promote a disciplined project management process that encourages frequent inspection and adaptation, a leadership philosophy that encourages teamwork, self-organization and accountability, a set of engineering best practices intended to allow for rapid delivery of high-quality software, and a business approach that aligns development with customer needs and company goals. Software developers need to integrate security with agile development methodologies for providing systems with reasonable security like authentication and / or authorization in them. Integrating agile methodologies with security enhances overall product trust worthiness. Agile processes based on extension of MDA approach, create flexibility for developing, executing and testing in weekly, incremental and iterative phases, using executable UML design. Traditionally it was felt that agile processes are in contradiction to developing secure software because of their light weight and not very formal approach. Abuser stories can be used for agile security requirements. One of the predominant usages of agile processes is extreme programming which needs to give architecture important relevance and should be made security.

Agile Software development basically uses a people oriented approach for organized teams. Documentation of Agile architectures can be done extending Views and Beyond (V and B) architecture documentation. Most successful usage of agile software development is applied by using extreme programming methodology for changing requirements. Java programming language can be used following agile software development by using extreme programming test driven development. Microsoft Visual Studio Visual C# programming language can also be implemented using agile patterns. Agile software development has many approaches like extreme programming, feature driven development, lean development, scrum etc. Agile Software Engineering extends traditional software engineering principles tailed to agile specific methodology for reducing cost and time in software development. Empirical software engineering approach can be used for agile development by following systematic approach for extreme programming and scrum.

**Agile Manifesto**

Agile development refers to any development process that is aligned with the concepts of the Agile Manifesto. The Manifesto was developed by a group fourteen leading figures in the software industry, and reflects their experience of what approaches do and do not work for software development. The Agile Manifesto was created as an alternative to document-driven, heavyweight software development processes such as the waterfall approach.

The four core values of agile software development as stated by the Agile Manifesto emphasize.

1. Individuals and interactions over processes and tools.
2. Working software over comprehensive documentation.
3. Customer collaboration over contract negotiation.
4. Responding to change over following a plan.

The agile manifesto principles are-

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable Software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage Deliver working software frequently, from a couple of weeks to a couple of months, with a preference for the shorter timescale
3. Business people and developers work together daily throughout the project.
4. Build projects around motivated individuals, give them the environment and support they need and trust them to get the job done.
5. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.
6. The best architectures, requirements and designs emerge from self-organizing teams.Simplicity, the art of maximizing the amount of work not done.is essential.
7. Continuous attention to technical excellence and good design enhances agility.
8. Agile processes promote sustainable development. The sponsors, developers and users should be able to maintain a constant pace indefinitely.
9. Working software is the primary measure of progress.
10. The most efficient and effective method of conveying information with and within a development team is face-to-face conversation.

The 12 principles laid down in the Agile Manifesto have been adapted for managing a variety of business and IT-related projects, including business intelligence (BI)[28] . They include:

1. Satisfying 'customers' through early and continuous delivery of valuable work.
2. Breaking big work down into smaller components that can be completed quickly.
3. Recognizing that the best work emerges from self-organizing teams.
4. Providing motivated individuals with the environment and support they need and trust them to get the job done.
5. Creating processes that promote sustainable efforts.
6. Maintaining a constant pace for completed work.
7. Welcoming changing requirements, even late in a project.
8. Assembling the project team and business owners on a daily basis throughout the project.
9. At regular intervals, having the team reflect upon how to become more effective, then tuning and adjusting behavior accordingly.
10. Measuring progress by the amount of completed work.
11. Continually seeking excellence.
12. Harnessing change for competitive advantage.

**Future Prospects**

The IT industry is now opening its eyes to the age-old industrial practices borrowed from the Toyota production system (TPS), the 3M company, non-IT industries, like healthcare and finance, have started embracing agile concepts faster than others.

Agile methods do not create secure code, and on further analysis, the perception is reality. There is very little "secure agile" expertise available in the market today.

1. This needs to change but the only way the perception and reality can change is by actively taking steps to integrate security requitements into agile developing methods.
2. Some of the future aspects can be seen as follows:
3. From developers point of view the quality of the developers is important.
4. From customer point of view agile, methods emphasize continues customer involvement, while traditional methods expect periodic customer involvement at formal millstone.
5. From requirement point of view, agile methods suggest they work better than traditional methods when requirements are volatile, much greater than 1% per month.
6. From architecture point of view, good architectures facilitate the addition of future functionality because they have, through planning, anticipated that functionality to a useful degree.
7. From re factoring part of view refectory is revising and simplifying the design or implementation of a software product without disturbing its functionality
8. From size point of view, agile methods can be adopted to larger (50-250 people) projects.
9. From balancing and risks point of view total risk exposure in project where a combination of agile and plan-driven methods are appropriate.

**Conclusion**

Secure software comes from interdependently verifying field's integrated application created and by secure or insecure development as completely accurately and affordably as possible. Agile is all about being more responsive and better meeting customer requirements and these days nobody can credibly a gue that security is not key requirement. Agile approach is one of the most popular software development approach and methodologies using this appropriate need to be modified to adapt the increase threats and vulnerabilities information system.

Extreme programming is very helpful when you have enough resource and budget to invest on big project. It will improve code quality and help you to protract in the market. Agile has been criticized for lacking security due to consideration of security throughout the agile development life cycle and absence of the dedicated response person, having a fair knowledge of software security with defined responsibilities.

**References**


1. *Shahram Jalaliniya, Farzaneh Fakhredin, "Enterprise Architecture and Security Architecture Development", Master Thesis, Department of Informatics, Lund University, June 2011, PP.1 - 95.*
2. *Heiko Tillwick, Martin S Olivier, "A Layered Security Architecture: Design Issues", in Proceedings of the Fourth Annual Information Security South Africa Conference (ISSA 2004), July 2004, PP. 1 -6.*
3. *Vibhu Saujanya Sharma, Pankaj Jalote, Kishor S.Trivedi,"Evaluating Performance Attributes of Layered Software Architecture", CBSE 2005 LNCS 3489 PP. 66-81.*
4. *Yann-Gael Gueheneuc, Giuliano Antoniol, "DeMIMA: A Multilayered Approach for Design Pattern Identification", September October 2008, IEEE Transactions on Software Engineering, vol. 34, no. 5, PP.667- 684.*
5. *George Farah, "Information Systems Security Architecture: A Novel Approach to Layered Protection", SANS Institute, September 2009, PP. 15-16.*
6. *Martin Fowler and Jim Highsmith, "The Agile Manifesto", August 2001. www.martinfowler.com /articles/newMethodology.html),*
7. *Kenneth E. Kendall, "Extreme Programming in Practice: A Human-Valued Approach to the DSI" Conference Management System, Decision Line Vol. 35,October 2004*
8. *Scott Withrow, "Extreme Programming: Do these 12 practices make perfect?," http://articles. tech republic.com.com/5100-22_11-1046488.html*
9. *B. Rumpe, P. Scholz, "Scaling the Management of Extreme Programming Projects, Projects & Profits Special Issue on Management of Extreme Programming Projects", Vol. III (8), pp. 11-18. ICFAI Press, Hyderabat, August 2003.*
10. *Clement James Goebel III, "Extreme Programming Practices Used to Facilitate Effective Project Management", Menlo Institute LLC, 2003.*
11. *www.onlamp.com/pub/a/onlamp/2003/07/31/extremeprogramming.html*
12. *www.ddj.com/dept/architect/184411706*
13. *www.msonewyork.com/agilemethod.html*
14. *Venkatesh Krishnamurthy , "A Glimpse into the Future of Agile Software Development" , Techwell April 17, 2013.*
15. *Ken Schwaber, Mike Beedle, "Agile Software Development with Scrum", Prentice Hall, 2001, pp. 89-94.*
16. *Sutherland, Jeff. "Scrum Web Home Page: A Guide to the SCRUM Development Process". Jeff Sutherland's Object Technology Web Page, 1996 //www.tiac.net/users/jsuth/scrum/index.html>*
17. *Schwaber, Ken. "Controlled Chaos: Living on the Edge." American Programmer, April 1996.*
18. *Aberdeen Group. "Upgrading To ISV Methodology For Enterprise Application Development. Product" Viewpoint 8:17, December 7, 1995.*
19. *Gartner, Lisa. "The Rookie Primer". Radcliffe Rugby Football Club, 1996 //vail.al.arizona.edu /rugby/rad/rookie_primer.html>*
20. *Pittman, Matthew. "Lessons Learned in Managing Object-Oriented Development". IEEE Software, January, 1993, pp. 43-53.*
21. *Booch, Grady. "Object Solutions: Managing the Object-Oriented Project". Addison-Wesley, 1995. 11 Graham, Ian. Migrating to Object Technology. Addison-Wesley, 1994.*
22. *www.findwhitepapers.com*
23. *www.cs.umd.edu*
24. *www.techbeacon.com*
25. *www.microsoft.com/sdl*
26. *blog.venturepact.com*
27. *www.sans.org*
28. *www.techwell.com*
29. *www.design-reuse.com*
30. *Dave Shackllford, "Integrating Security into Development, No Pain Required", September 2011, ASANS Whitepaper.*
31. *David A. power, "Software development: Effective practices and Federal challenges in Applying agile Methods". July 2012, GAO -12-681.*