A Comprehensive And Reliable Technique For Identifying Various Security Activities And Incorporating In Sdlc

Sushil Kumar[a], Avinash Kaur[b], Ashish Jolly[c]

[a,b] Department of Computer Science & Engineering, Lovely Professional University, Jalandhar, Punjab,India
[c] Department of Computer Science, Government PG College, Ambala Cantt, Haryana, India
[a] sk93recj@gmail.com, [b] avinash.14557@lpu.co.in, [c] ashishjolly76@gmail.com

Article History: Received: 11 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 23 May 2021

Abstract: The agile product development methodology is a series of concepts used for the creation of applications. In the creation of agile applications, strategies evolve by cooperation of self-organizing, cross-functional teams using suitable methods for their context. This paper provides principles that should be implemented in the agile product creation phase. Approximately 500 agile app engineers across the globe have engaged in the study. A questionnaire pertaining to the different protection practices to be implemented during each process of agile software creation was demanded in the survey. Based on this analysis, we have carried out the most effective and useful compliance practices that can be implemented across the multiple phases of agile software development. 80% of agile developers voted for initial preparation, which is an essential protection practice to be implemented during the pre-requirement process. Similarly, 75 per cent said that protection criteria during the requirement process, 95 per cent opted for risk analysis during the design phase, 80 per cent said that coding guidelines during the development phase, 62 per cent said that they defined, conducted and enforced safety checks during the testing phase, and 77 per cent said that the final safety check during the release phase would be integrated during the various agile software phases. The proposed analysis overcomes the problems of agile design and protection by offering a simple overview of the security practices implemented throughout the multiple phases of software growth. We have bridge-in, a protection difference between the conventional paradigm of waterfalls and the in-practice agile model of growth.

Keywords: Innovative product creation, Information protection, Waterfall model, Weakness, Agile technology

1. Introduction

Protection is an essential computational component and the system's most inoperative requirement.Protection is still a big problem and could get even worse in the future owing to greater uncertainty, usability and extensibility. Software developers are then expected to set up safe and stable organizations. In the last few years (1995-2008) as seen in Fig, figures from the Machine Emergency Response Team (CERT) recorded a substantial rise in security-related vulnerabilities in software engineering. As seen in the graph, the vulnerabilities found have decreased over a period of years since continuous growth. In comparison, the National Center of Quality and Technology paper reports that, owing to bad digital devices, the United States loses billions of extra dollars a year[4]. The price would be reduced by decreasing the error rates at each point of the creation and production of software. Therefore, the finished output

It could cost less over its lifespan.

Figure 1: Vulnerabilities Reported to CERT Centre

Taking into account the perceptions appeared in Figure 1, a method of managing programming improvement progress must be built up, and this could advance security during all product improvement cycles[6]. In any case, settling programming is regularly observed as a post improvement development, and next to no record is taken of this in the advancement of coding. Administrative enactment requires the reconciliation of security with the existence pattern of programming advancement from the beginning until the item is being used so as to acquire a

sheltered item. In any event, when programming creates all through its richness cycle, the security part of the item ought to be given unique consideration with programming improvement procedures[32].

The improvement of uses today incorporates the organization of rapid application devices. To request to create items effectively, organizations transform their customary business approach into nimble conveyance rehearses[7]. Changes are therefore done to improve the clout of programming advancement and are presently the result of more programming applications and frameworks dispersed on the web. Light-footed improvement of utilizations impacted the overall creation of programming.

In any event, when the dexterous strategy in the current programming industry has become a mainstream demonstrating procedure, it has been accounted for that the wellbeing approach has disservices regarding secure programming advancement[29]. Albeit a few exploration articles have expressed, current SE methodology in dexterous ventures are difficult to execute. This truly is because of genuine limitations, for example, a deficient layout of an item, moderately higher speed and absence of lithe plan forced documentation[8].

Programming Security:

The most difficult issue in PC security today is programming, as programming inconveniences are normal and there's a developing issue[28]. Likewise, in future, the issue could turn out to be a lot of more terrible as

- Modern tech capacities in an unforgiving organized condition.
- Expandable frameworks, for example, Java virtual apparatus and .Net, become normal and fuse versatile code dangers.
- The level of unpredictability is expanding.

The security of the product and application can split into two parts: programming security and security of application.

Process of Security Engineering:

The collection of activities achieved for creating, keeping, and conveying a secure programming item is referred to as security building progression; security exercises may be either sequential or iterative. The objective is to fabricate powerful, mistake free programming by utilizing frameworks, procedures, apparatuses and approaches for tending to security worries in various periods of programming advancement life cycle (SDLC)[26,13].

Programming frameworks that executed by secure programming forms can

- Perform well during the assaults by either contradicting the misuse of shortcomings in the product or persevere through the disappointments.
- Limit the harm from an assault set off deficiency disappointment and recuperate rapidly from those disappointments that the product couldn't persevere.

Programming Vulnerability:

Programming weaknesses will be shortcomings in a product plan, execution and structure that would be inadvertent or intentional[23]. Programming weaknesses permit an assailant to decrease the framework's data confirmation. More often than not, the world commits to PC security by and large covering essential open innovation issues, for example, which firewall is ideal, what cryptography is, the thing that enemy of infection items are acceptable or the issues including a pernicious assault and crazy infection. Be that as it may, behind each malignant assault and PC security issue there is shaky software[24]. On the off chance that the created programming gets out of hand, it may confront unwavering quality, accessibility, wellbeing and security issues. Saltines don't make security gaps; they basically misuse them. Most security related weaknesses emerge from abandons that are accidentally embedded in the product in the plan and usage period of the development[25]. Regular programming weaknesses incorporate cushion flood, store flood, race condition, design string bugs, helpless irregular number generator, SQL infusion, forswearing of administration (DOS) and lost trust[23].

Figure 2 beneath illustratesthe different stages of development, clearly depicting various vulnerabilities exploited by attacker and how bugs affect the progression of development.

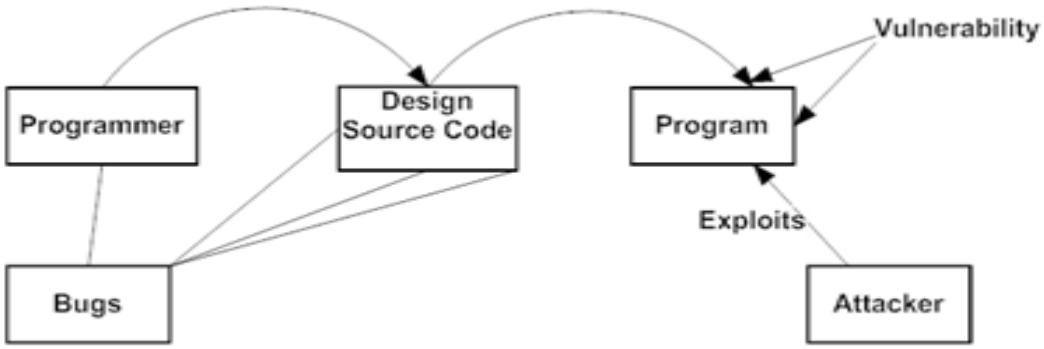


Figure 2: Software Defects

- Bug - a developer's error, for example that once exceeds a for-circle.
- The vulnerability - a frame defect that is evident to an aggressor, such as replicating information to an unlimited check support.
- Exploit - an attacker created a small program to misuse the weakness, such as to help the source of flood software.

Spry Development Methodology:

Programming progression is an unpredictable undertaking that necessitates communication and collaboration among the engineers, a task manager, clients, and all project accomplices. There are several methodologies available to improve the progress of programming processes. There are two highly desired practices: path or game plan based theory and rapid programming improvement. Course model, constantly utilized as a bit of programming improvement strategy, is a dynamic system process in which streaming is unequivocally downwards (like a course) through the times of masterminding, assessment, structure, coding, testing, conveyance and backing[17].

Then again, programming model is"an iterative and steady way to deal with oversee programming movement which is self-filtering through social events inside a stunning association structure with "basically enough" work that passes on top notch courses of action in a fiscally brilliant and favorable way which is up to the mutabledesires of its assistants". Agile Manifesto described the composed model, on the other hand with the course model. The emphasis in the course practices, the game plan based theory, is on the proper system, gadgets, and related documents. Regardless of what could be anticipated, in agile methodology, the focus is on individuals and their relationships, scripting rather than documentation, complex customer endeavor rather than agreement game-plan, and responding to change after a course of action.

In the most recent couple of years, liberal bits of the thing industry have moved the programming improvement framework from an unfaltering course to a more flexible handy programming progress process. Different nimble programming progress methodologies are available, for instance, exceptional programming (XP), scrum, incorporate driven unforeseen development (FDD), lean programming headway, valuable stone approaches,dynamic structures headway theory (DSDM)[15], etc. While there are different contrasts between these perspectives, they depend upon some run of the mill rules, most advancement improvement, investment, joint effort and system versatility for the term of the present example of the undertaking.

The Agile Manifesto is strengthened by the adoption of 12 ideals:

- Consumer commitment through before schedule and nonstop transport of significant programming
- Welcome propelling necessities, even late being made
- Deliver working programming generally, in shortest possible time rather than months or years.
- Specialists and architects must collaborate on a daily basis for the entire duration of the project.
- Build reaches out around pushed people, give them condition and bolster they need and trust them to take care of everything
- Face-to-stand up to discussion is the most productive and productive technique for offering data to and inside an improvement gathering

- Working writing computer programs is the essential extent of movement

Doable new development - the benefactors, draftsmen and clients ought to be able to keep up a reliable pace uncertainty, consistent regard for specific perfection and incredible structure improves availability

Ease - the claim to fame of strengthening the extent of work not done is indispensable

Self-sifting through groups

At predictable intervals, the gathering considers how to wind up more viable, at that point tunes and adjusts its lead fittingly

These are highlights of an unmistakable nimble system from other fundamental or incorporated full-cycle forms (conventional or cascade improvement technique) .

The light-footed SDLC is extremely shared, iterative and steady progression (see Figure 3). The job of spry venture individuals is considerably more powerful than in cascade venture individuals. In the cascade advancement strategies authorities of every improvement stage are answerable for their errand and to move the ancient rarities to the pros of the following improvement stage and it proceeds along these lines. In spite of what may be normal, lithe programming engineers work personally with their accomplices to grasp their necessities?Similarly, they use automated testing to understand and validate their solution. At that point, the arrangement appears to the partner for criticism and thus implanting disfigurements at each phase during the turn of events. This implies, lithe engineers have in general aptitudes of the full venture life cycle[14].

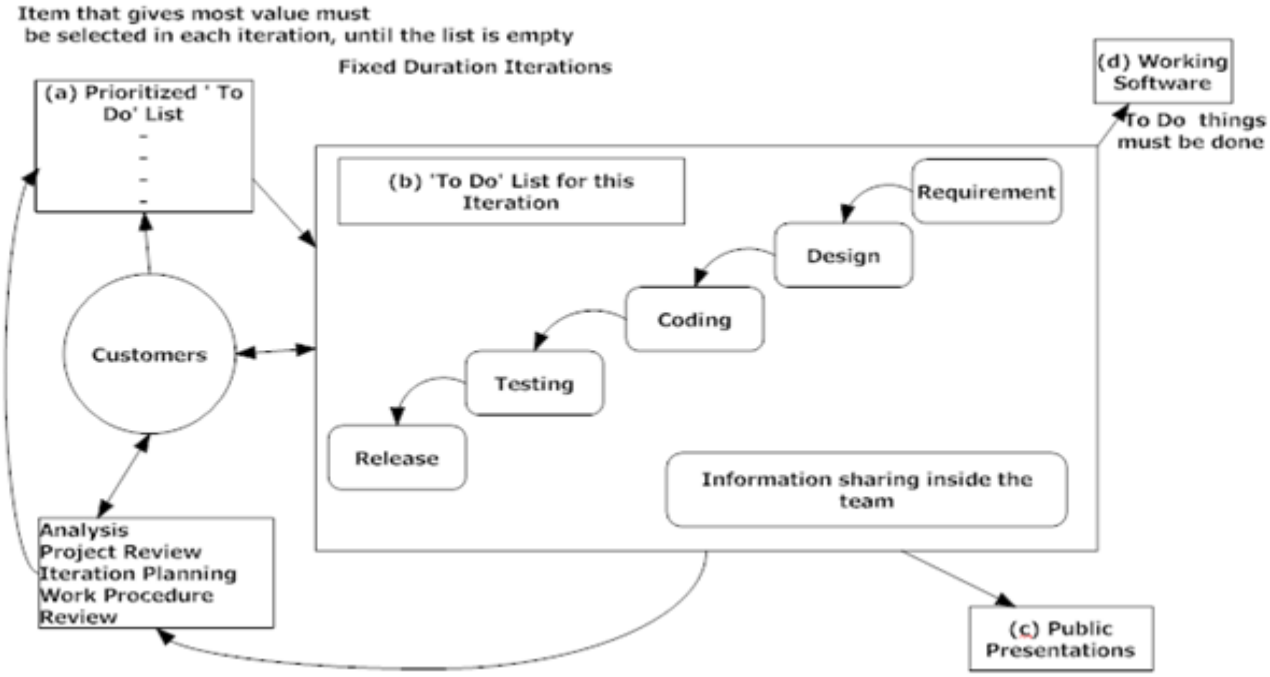


Figure 3: Software Development by Agile approach.

The point of the coordinated improvement method is to put joint effort with the client to investigate their necessities and convey a last programming item. In a dexterous advancement methodology as appeared in Figure3, assignments are bifurcated into little enlargements (a) with least turn of events and don't legitimately incorporate long haul improvement. Duplications are brief timeframe settings that traditionally last from one to about a month. Duplication includes a cross-useful group utilized in all the product improvement life cycle: arranging/pre-necessity, prerequisites, structure, coding, testing and delivery (b). At the finish of emphases an operational programming item is uncovered to benefactors for criticism (c). This diminishes total hazard and allows the undertaking to adapt to changes quickly. Despite the fact that, cycle probably won't improve plentiful usefulness to justify a market discharge, the point is to issues (d). In this way, so as to deliver an item or new highlights, different cycles may be required.

Agile Security:

Deft methodologies are commonly applied for web and framework usage progression where cybersecurity perils are clear yet hypothetical agents ensure that agile strategies are lacking for security-fundamental endeavors.

By the by of security risks, the deft frameworks have scarcely any features unequivocally keeping an eye on security issues. One motivation behind why the nimble strategies disdain security considerations may abbreviate from a misapprehension that security concedes the improvement strategy. Next, the essential clarification is that no security building procedures are established explicitly for the quick method model. As a result, deft efforts have been made to employ winning course SE structures for the advancement of secure programming. Regardless, these staggering weight SE performs are not as considerable as in course progression since they are proposed explicitly for a game plan developed course improvement methodology. Researchers suggest that there is inconsistency between standard SE structures and the thoughts and practices proposed by the apt announcement. This proposed deft methods work with short headway expands that adjust adequately to change, while existing SE structures trust on a course improvement setting to reduce give up through overpowering and fanatic techniques[12].

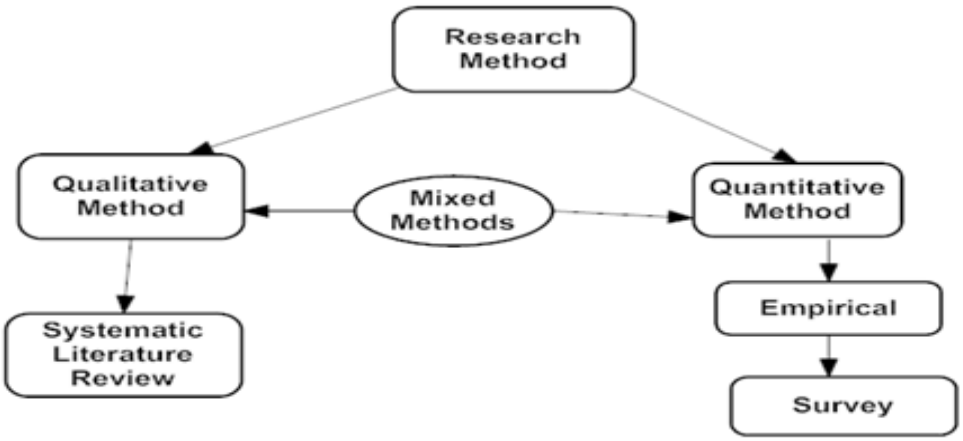


Figure 4: Research Design Overview

The exploration procedure we have utilized and its means are articulated in detail in the accompanying subcategories.

Points and Objectives:

The reason for the review is to investigate what security exercises from conspicuous SE movements are misused in the contemporary coordinated industry and recognize the most adjusted and important security exercises to nimble ventures and in conclusion to prescribe the most adjusted security exercises to a deft model.

The item is achieved by tending to the accompanying points:

1. Examine cutting edge security in deft procedure
2. Examine cutting edge SE forms

- Recognize prominent SE forms
- Recognize the security exercises carried out by those SE types.

3. Sort the recognized security exercises into the recommended progression stages.
4. Get ready review polls dependent on the recognized security exercises
5. Lead review
6. Propose the most adjusted security achievements to nimble model
7. Give recommendation to future improvement in the zone.

Overview Construction:

The overview instrument is structured on Google structure. A study comprising 06 inquiries is created. All inquiries were alluded to different security exercises to be consolidated during each period of nimble programming improvement. Before taking up the review the respondents are required to enter the insights about their Name, association, email id and so on. In addition, a spread sheet outlining the consistency of the investigation and methodology, as well as a depiction of each security activity, is included to the overview. The promise of confidentiality[25], anonymity, and our promise to provide the final summary result to the respondent encouraged us to obtain more respondents eager to answer the research questions and provide more practical responses. This study enlisted the help of over 450 Agile Software Professionals.

The result of the investigation is as per the following:

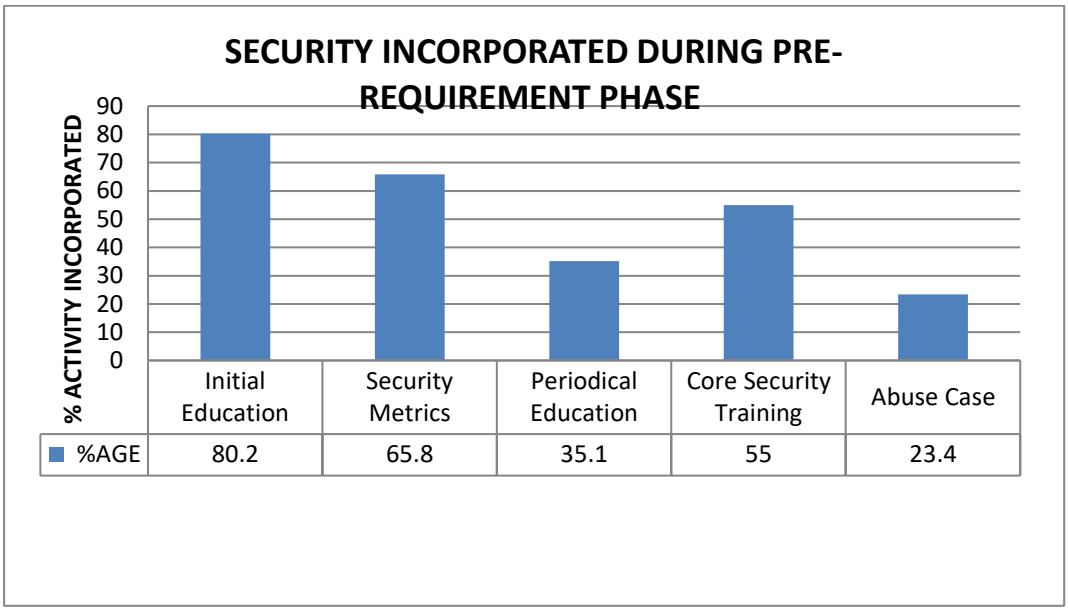


Figure 5: Security incorporated during Pre-requirement Phase

Figure 5: shows that the security activities which are important during the pre-requirement phase are Initial Education and security metrics.

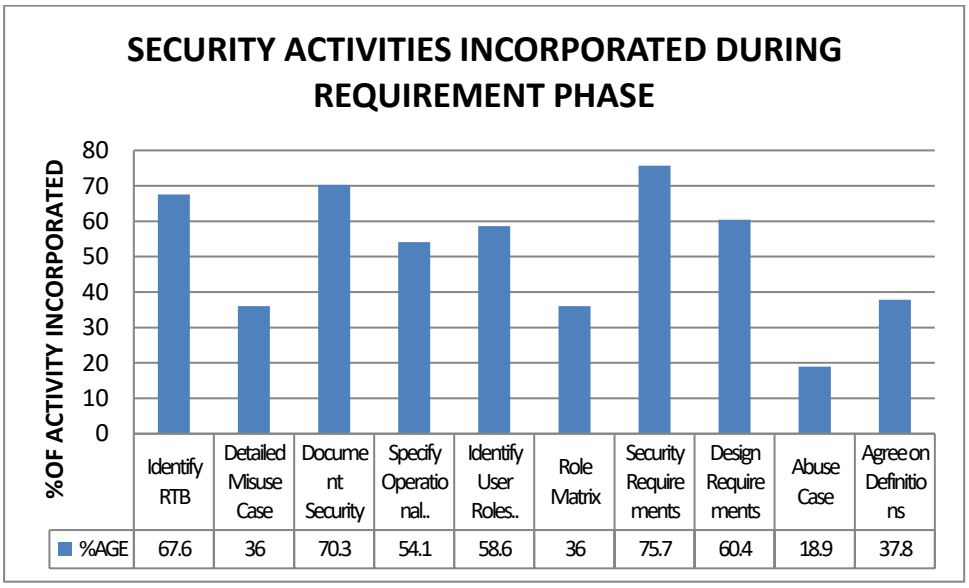


Figure 6: Security Activities Incorporated during Requirement Phase

Figure 6: shows that the security activities which are important during requirement phase are security requirement and document security.

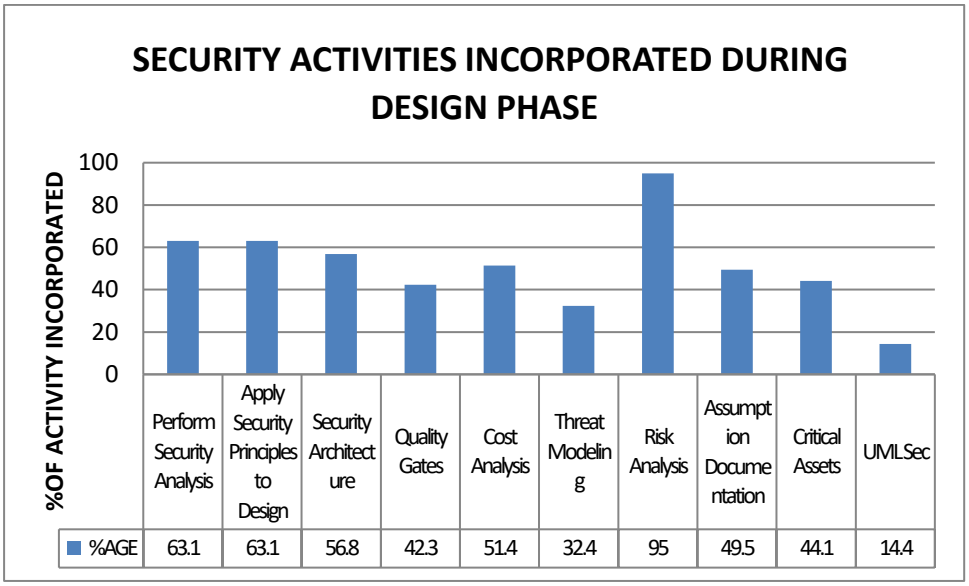


Figure 7: Security Activities Incorporated during Design Phase

Figure 7: shows that the security activities which are important during the design phase are risk analysis, apply security principles to design and perform security analysis of system requirements & design.

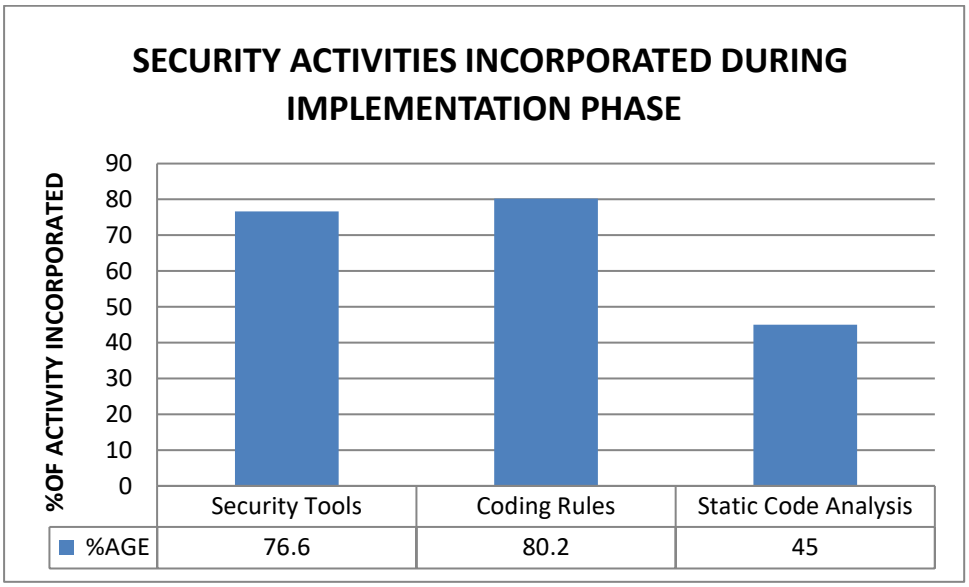


Figure 8: Security Activities Incorporated during Implementation Phase

Figure 8: shows that the security activities which are important during the implementation phase are Coding rules and Security Tools.

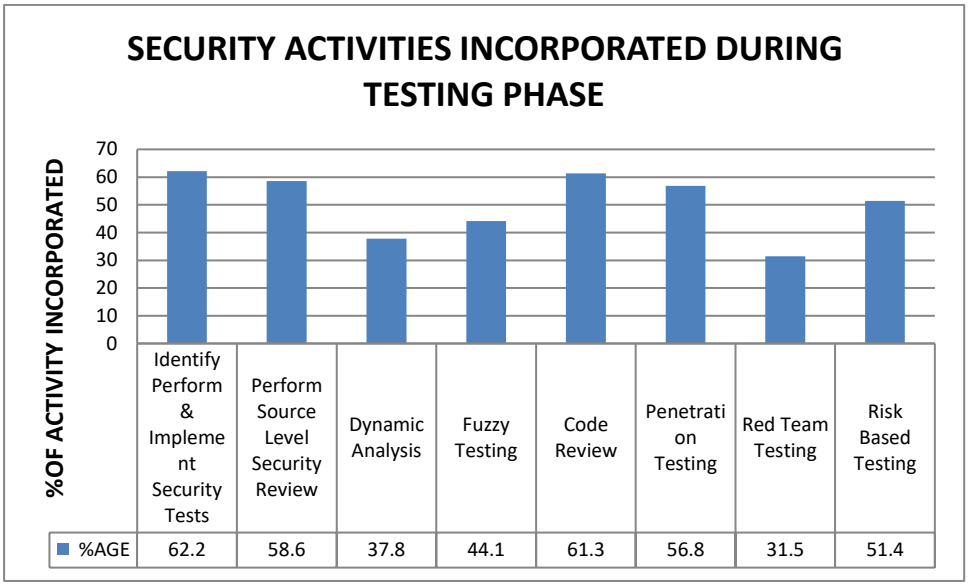


Figure 9: Security Activities Incorporated during Testing Phase

Figure 9: shows that the security activities which are important during testing phase are Identify Perform and Implement Security Tests and Code Review.

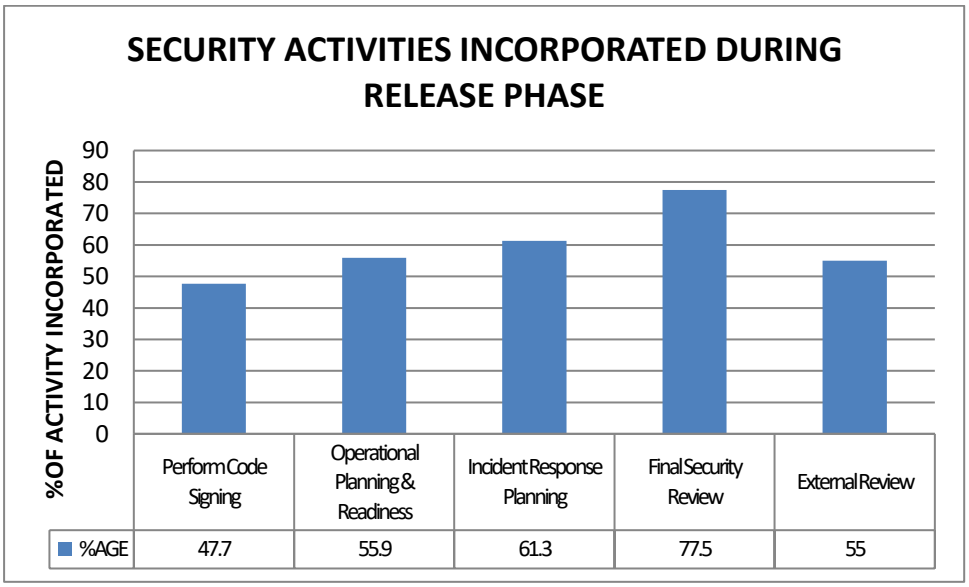


Figure 10: Security Activities Incorporated during Release Phase

Figure 10: shows that the security exercises which are significant during discharge stage are Final Security Review and Incident Response Planning.

Coming up next are the discoveries identified with generally good and useful security action that can be joined during various periods of Agile Software improvement.

Sr.no	Software development Phase	Compatible Security Activity
1.	Pre-Requirement Phase	Initial Education
2.	Requirement Phase	Security Requirement
3.	Design Phase	Risk Analysis
4.	Implementation Phase	Coding Rules
5.	Testing Phase	Identify, Perform and implement security tests
6.	Release Phase	Final Security Review

Table 1: Compatible Security Activities during various Software Development Phases

2. Related Works:

Nooper Davis et al. presented impressive data about traditional processes, morals, life-cycle models, backgrounds, and procedures that deals with the support to secure software development. Dave Shackleford et al. have demonstrated security glitches affect operating system mechanisms, client requests, web requests or particular code that deals with the generation of power or other apparatus control schemes, the mainstream of well-publicized weaknesses are connected to coding problems and application subjects. Microsoft[4] described a way to hold unimportant software security observations by Agile software development approaches, like Extreme Programming and Scrum procedure. The main objective was to achieve a high Microsoft Security Development Lifecycle using Agile procedures in such a way that upholds the philosophies of both the Agile approaches and the SDL development procedures. Nor Shahriza Abdul Karim, ArwaAlbuolayan, Tanzila Saba, AmjadRehman et al. proposed a case study in which the practices being cast off in software growth in Saudi Arabia and defined a prototype for integrating safety into the SDLC environment. Nooper Davis et al. presented impressive data about traditional processes, morals, life-cycle models, backgrounds, and procedures that deals with the support to secure software development.Nabil M. Mohammeda et al. presented impressive data about traditional processes, morals, life-cycle models, backgrounds, and procedures that deals with the support to secure software development. Wentao Wang et al. explains the requirement of security in open source software development.A linear method for identifying security needs was proposed by Wentao Wang et al. The first to quantify function values for needs in OSS projects involves logistic regression modeling (RM). The correlation matrix of all functional values is then used to categorize safety and protection conditions.

The dual issue for framework positioning and access monitoring setup is resolved by Zoltán Ádám Mann. The issue is formulated and quadratic mixed integer schemes are used.Kalle Rindell et al. explains the importance of security in agile software development.Adel Mohammad et al. analyze the new developments in data security to create stable software.The objective was to classify the suitable means of presenting security actions in the SDLC environment.

3. Proposed Work:

This section deals with the results and discussions for the security of the agile development for the evaluation in the secure manner. The whole simulation is based on the JAVA environment.

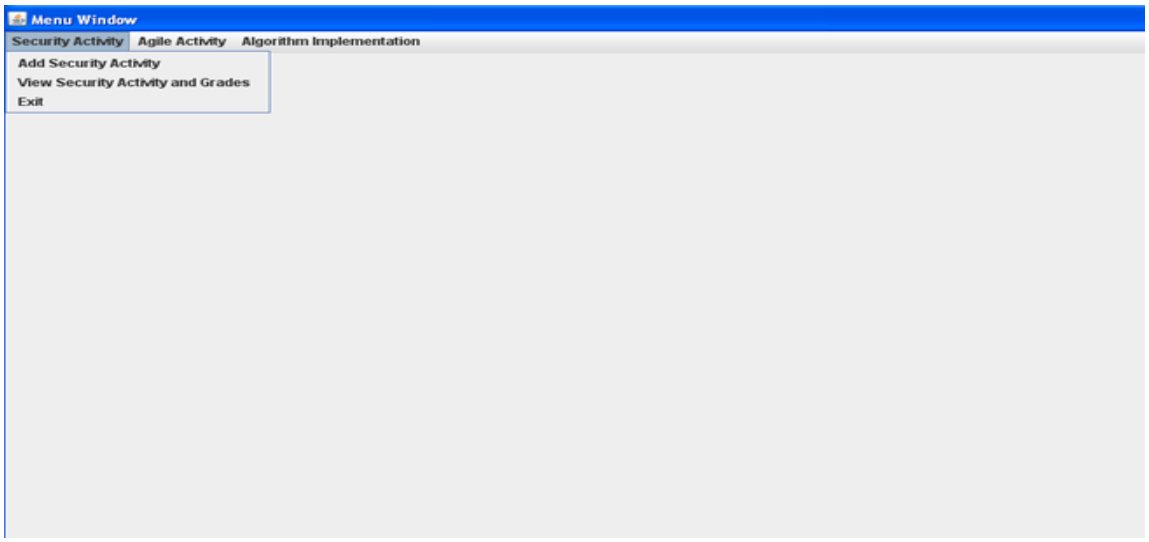


Fig 11: Main menu

The fig 11 shows the main GUI panel which is implemented in java for the implementation of the data. In this Graphical user interface there are various options for the agile activities and implementation.

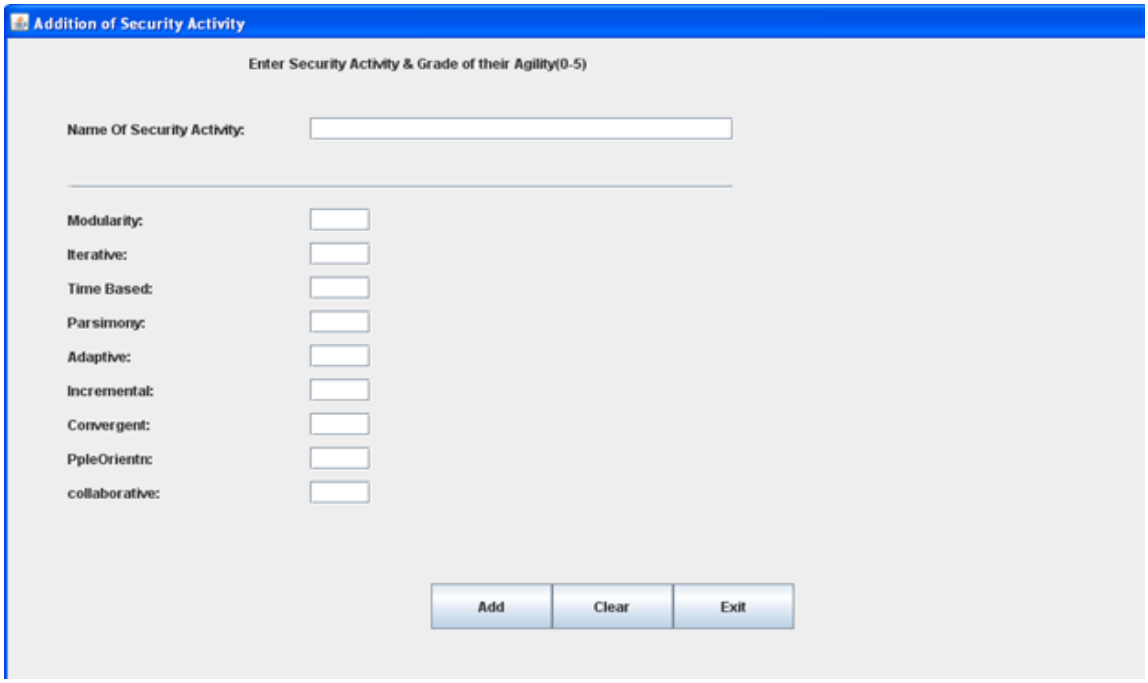


Fig 12: Adding Security

The fig 12 shows the GUI panel to add the security activity for the evaluation with different attributes and also the pushbuttons to add such options, clear the options and exit from the process.

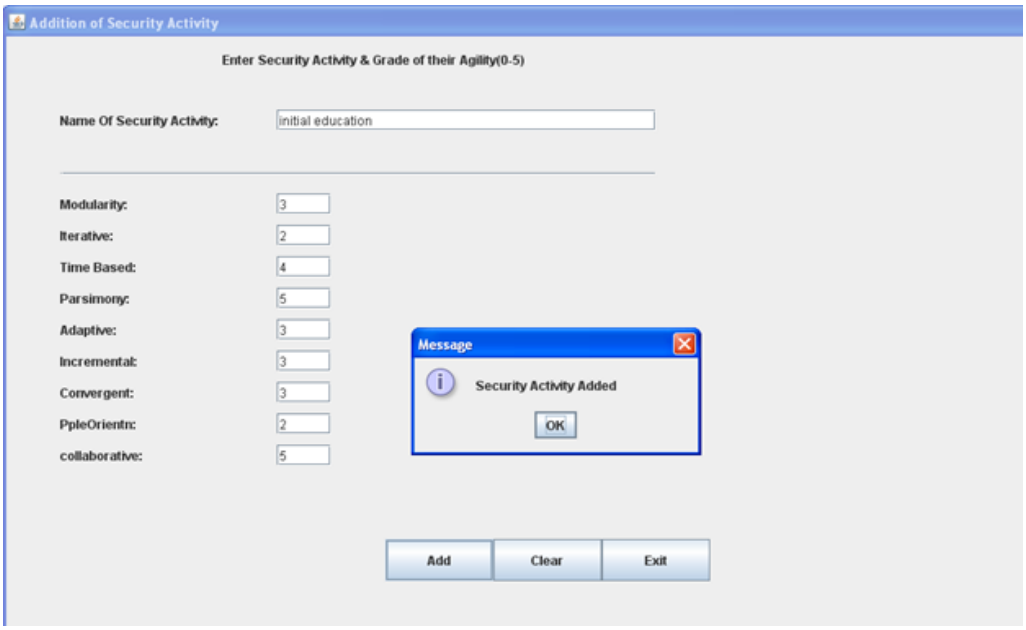


Fig 13: Added Activity

The fig 13 shows the GUI panel and shows that the secure activity is added which shows in the form of the message box with the applied security name in the GUI application.

	Modularity	Iterative	Time Based	Parsimony	Adaptive	Incremental	Convergent	PpleOrientn
Initial Education&training	3	5	5	5	5	5	5	5
Security Requirement	4	4	4	5	4	3	3	4
Identify Trust Boundary	2	5	2	5	4	2	4	2
Role Matrix	3	4	3	4	3	3	2	4
Risk Analysis	2	5	2	5	4	3	4	2
Threat Modeling	1	4	2	1	1	1	3	1
Static Code Analysis	5	5	5	4	5	2	5	5
Coding Rules	3	4	4	2	5	3	4	2
Security Testing	1	5	3	3	5	2	4	4

Fig 14: List of securities

The fig 14 shows the GUI in which the list of securities are shown and also the attributes which are entered during the addition of the secure activity in the adding process. These are the list of total numbers of securities that are added for the secure agile development.

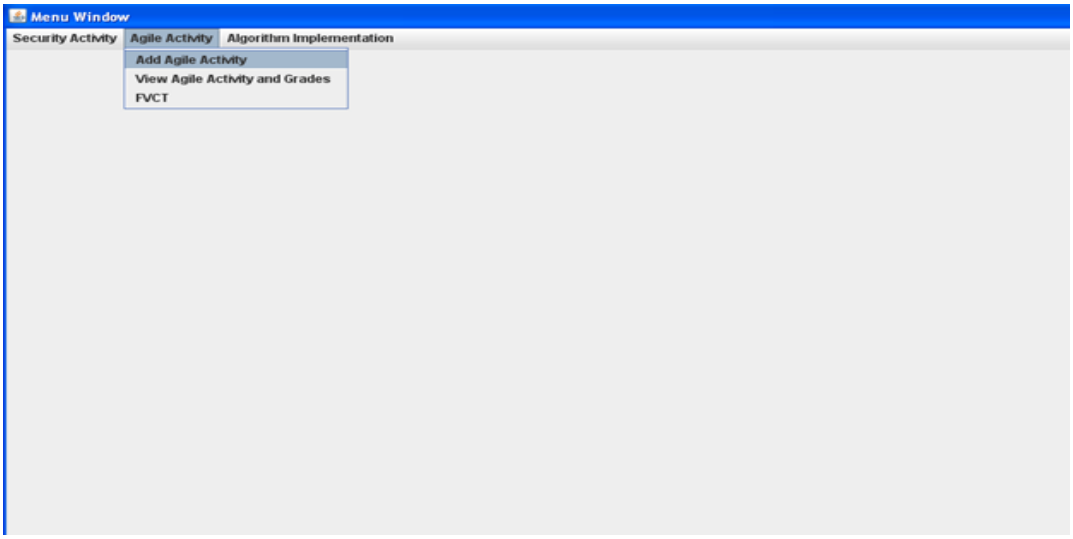


Fig 15 (a) Agile Activity process

Fig 15(b) Agile Activity Addition

The Fig 15(a) and (b) shows the agile menu and also the agile activity addition with necessary attributes for the secure agile lifecycle development

Fig 16: Agile Activities and Grade of their Agility

The fig 16 shows the addition of all the agile activities with the necessary attributes or we can the inputs for the evaluation of the agile development in the GUI panel.

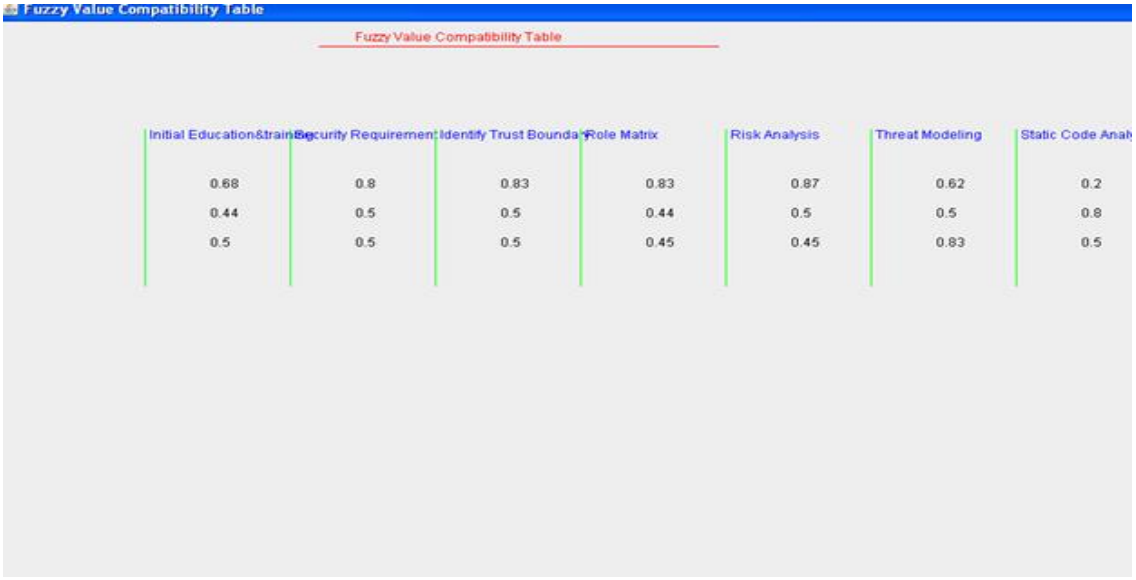


Fig 17: Fuzzy Value Compatibility Table

The fig 17 shows the fuzzy compatible table with the necessary attributes that how much security requirement is needed and risk analysis is there and also the threat modeling which is one of the main crucial steps for the evaluation of the secure agile development cycle. The main structure of Fuzzy Logic deals with the capability and the compatibility among two proposals, in such a way that the modeled process through the inference system is autonomous from the precise probability distributions involved.

Fig 18: Security Activity Integration Table

The fig 18 shows the Compatibility of Security with Agile Activity for Integration during SDLC.

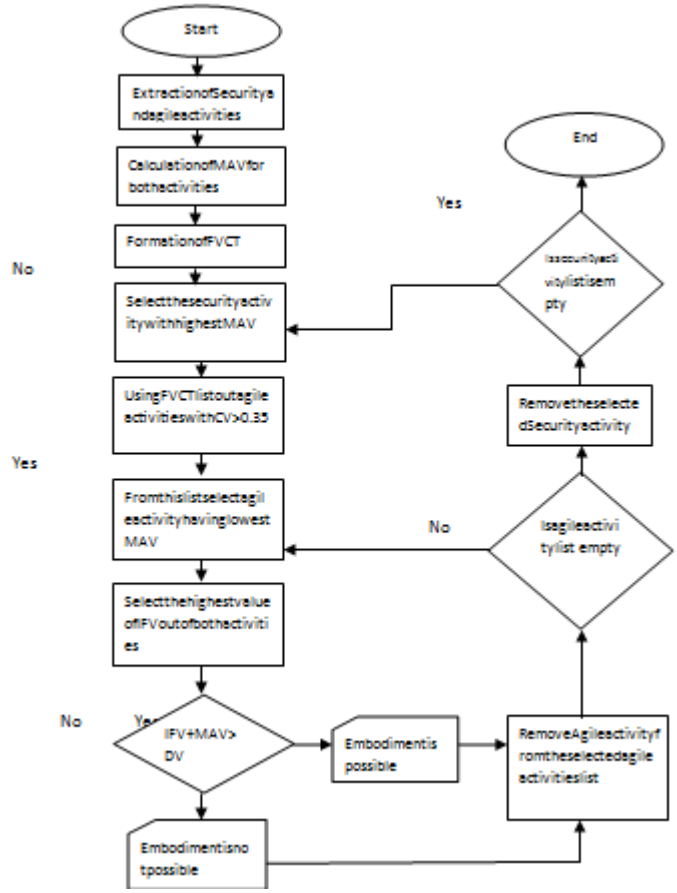


Figure3.1 Flow Chart Showing the steps involved in embodiment of Security and agile Activities

Proposed Algorithm:

Step 1: Start

Let the security activity S1, S2, S3 … Sn such that the selection of the Sn has the highest MAVsa.

Where Sn= Security Activity

Step 2: Listing out the agile happenings from the FCVT

Step 3: Evaluate the compatibility value (CV) greater than threshold value (TV) of 0.35

Step 4: Selection of the agile activity having lowest MAV

Step 5: Checking of the influencing factor for selected agile activity and selected security activity in IFV table. Select the highest IFV among both the activities.

Step 5: Start For

Remove the agile activity from the selected agility activity list.

Repeat the step until the agile activities list is empty.

End For

Step 6:Start For

Remove security movement based on the security activity table. Repeat steps until the security activity table is not empty.

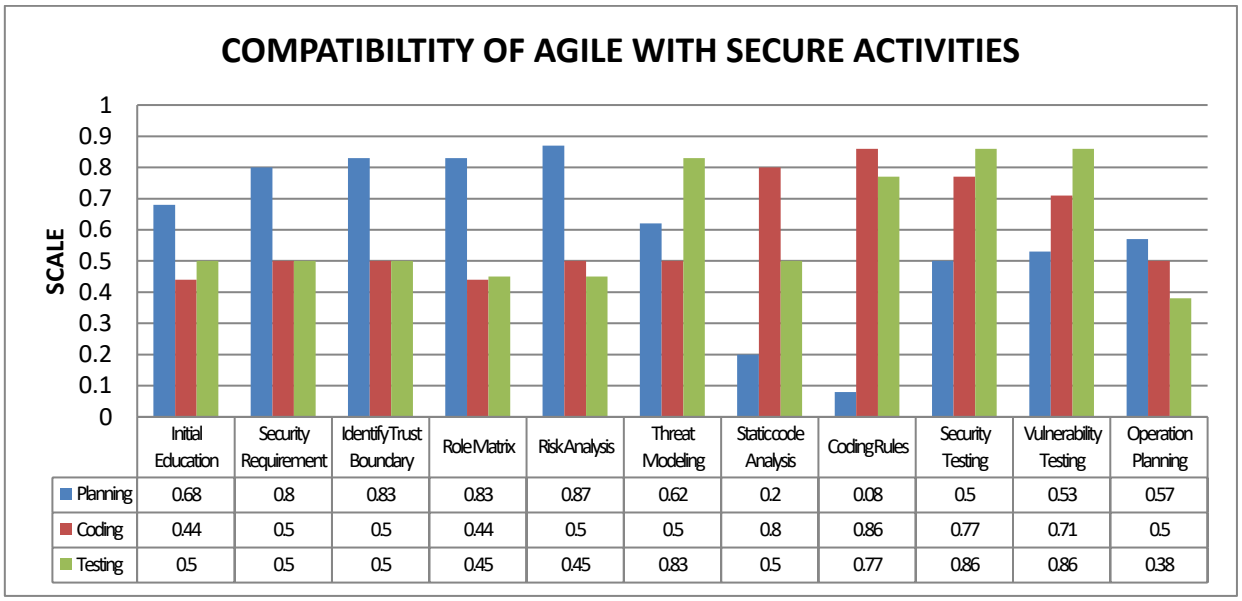
End For

Step 7: Evaluation of the security activity embodiment which measures the importance of security activity in particular agile development phase.

Step 8: Stop

4. Result & Conclusion:

It can be analyzed fromFig 20 that security activity, initial education is most compatible with planning activity as compare to other agile activity, so initial education must be integrated with the planning. It deals in high estimation compatibility of secure activity with agile activity. From this FVCT only those agile activities are selected which possess minimum threshold value of 0.35.The threshold values 0f 0.35 means that at least the selected agile activity is 35% compatible for the embodiment. The list is formed with all the agile activities which possess the value greater than that of 0.35. From this list the agile activity having least MAV is selected. The least MAV means that selected agile activity is possess very less feature of agile methodology. The reason for selecting lowest MAV is to check the integration for the worst agile activity because of its lowest MAV. Out of selected agile and security activities, highest IFV is selected. Now if the value of combination of MAV of agile activity and IFV is more than that of Delta Value than the embodiment is possible of selected agile and security activity otherwise it is not possible to embodiment both activity. The DV depends upon the priority of the security activity embodiment. This value measures the importance of security activity in particular agile development phase. The DV depends upon the project manager expertise and can vary from development phase to phase, company to company depending upon other factor such as environment of the company, software delivery time, software security quality etc.



	Initial Education	Security Requirement	Identify Trust Boundary	Role Matrix	Risk Analysis	Threat Modeling	Staticcode Analysis	Coding Rules	Security Testing	Vulnerability Testing	Operation Planning
Planning	0.68	0.8	0.83	0.83	0.87	0.62	0.2	0.08	0.5	0.53	0.57
Coding	0.44	0.5	0.5	0.44	0.5	0.5	0.8	0.86	0.77	0.71	0.5
Testing	0.5	0.5	0.5	0.45	0.45	0.83	0.5	0.77	0.86	0.86	0.38

Fig 20: Compatibility of Agile with Secure Activities

5. Conclusion:

Protection is a significant consistency function of the software artefact and, in order to ensure this, we need to listen to it throughout the lifespan of software creation. Agile processes remain, though, incapable of delivering a successful consumer product. Current waterfall SE systems, for example, are not in agreement with the agile method model due to the tremendous limits imposed by this theoretical description. Furthermore, the significance of the SE approach is unquestionably accepted with both the waterfall and the agile models. As a consequence, agile companies are motivated to use conventional SE waterfall systems to deliver efficient applications. The paper is seen as a move forward in the examination of waterfall SE systems and their physical methods used in the recent agile industry. In order to provide effective applications with an agile environment, there is a need for a SE-process that can fix security concerns in any step of the implementation; no SE-process is designed specifically for the agile lifecycle of the project. This may also be achieved by combining the most reliable and useful safety achievements of waterfall SE progressions with agile development or by constructing a new SE-process.

References

1. N. Davis, “Secure software development life cycle processes: A technology scouting report”, No. CMU/SEI-2005-TN-024.Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst, 2005.
2. D. Robel, "SANS Institute InfoSec Reading Room." (2015).
3. N.A. Karim, , A. Albuolayan, T. Saba, and A. Rehman, "The practice of secure software development in SDLC: an investigation through existing models and a case study," Security and Communication Networks 9, no. 18, pp: 5333-5345, 2016.

4. M. Howard and Steve Lipner, The security development lifecycle. Vol. 8. Redmond: Microsoft Press, 2006.
5. C.T. Lin, H. Chiu, Y.H. Tseng, "Agility Evaluation Using Fuzzy Logic", International Journal of Production Economics, Volume 101 (2), pp: 353–368, June 2006.
6. "Comprehensive, Lightweight Application Security Process", http://www.owasp.org, 2006.
7. Beznosov and Kruchten, "Towards Agile Security Assurance'' NSPW '04 Proceedings of the 2004 Workshop on New Security Paradigms, pp: 47-54, 2004.
8. B. De Win, R. Scandariato, K. Buyens, J. Grégoire, and W. Joosen, 'On the secure software development process: CLASP, SDL and Touchpoints compared', Information and software technology, vol. 51, no. 7, pp. 1152–1171, 2009.
9. D. Baca 'Developing secure software in an agile process', Computer Science Department, Blekinge Institute of Technology Sweden, pp. 129-149, 2012.
10. D. Baca and B. Carlsson, 'Agile development with security engineering activities', in Proceeding of the 2nd workshop on Software engineering for sensor network applications, 2011, pp. 149–158.
11. D. Mellado, E. Fernandez-Medina, and M. Piattini, 'A comparison of the Common Criteria with proposals of information systems security requirements', in Availability, Reliability and Security, 2006. ARES 2006.The First International Conference on, 2006, p. 8–pp.
12. G. G. Miller, "The Characteristics of Agile Software Processes ", Proceedings of the 39th International Conference. And Exhibition on Technology of Object-Oriented Languages and Systems (TOOLS 01)", 1530-2067/01, IEEE 2001.
13. J. Grégoire, K. Buyens, D. Win, R. Scandariato, W. Joosen, "On the Secure Software Development Process: CLASP and SDL Compared". 29th International Conference on Software Engineering Workshops (ICSEW 07) 0-7695-2830-9/07, IEEE, 2007
14. HosseinKeramati, Seyed-Hassan Mirian-Hosseinabadi, "Integrating Software Development Security Activities with Agile Methodologies", IEEE/ACS International Conference on Computer Systems and Applications, AICCSA, 2008.
15. M. Howard, S. Lipner, "The Security Development Lifecycle – SDL: A Process for Developing Demonstrably More Secure Software", Microsoft Press, 2006.
    I. Flechais, M. A. Sasse, and S. Hailes, 'Bringing security home: a process for developing secure and usable systems', in Proceedings of the 2003 workshop on New security paradigms, 2003, pp. 49–57.
16. J. Gregoire, K. Buyens, B. D. Win, R. Scandariato, and W. Joosen, 'On the secure software development process: CLASP and SDL compared', in Proceedings of the Third International Workshop on Software Engineering for Secure Systems, 2007, p. 1.
17. L.A Zadeh, "Fuzzy Sets", Information and Control, Volume 8 (3), pp: 338–353, 1965.
18. L.A Zadeh, "The Concept of a Linguistic Variable and its Application to Approximate Reasoning-I", Information Sciences Volume 8 (3), pp: 199–249, 1975.
19. "Manifesto for Agile Software Development," http://www.agilemanifesto.org
20. [21]http://www.microsoft.com/security/sdl/default.aspx
21. M. Siponen, R. Baskerville, T. Kuivalainen, "Integrating Security into Agile Development Methods", In Proceedings of the 38th Annual Hawaii International, 2005.
22. S. Sonia and S. Archana, 'Integration Analysis of Security Activities from the Perspective of Agility', IEEE, pp. 40–47, 2012.
23. S. Goldman, R. Nagel, and K. Preiss, "Agile Competitors and Virtual Organizations", Chapter 3, Van Nostrand Reinhold, 1995
24. M. Zulkernine, I. A. Sheikh, "Software Security Engineering: Towards Unifying Software Engineering and Security Engineering". Copyright
25. Nabil M. Mohammeda , Mahmood Niazia,b,∗ , Mohammad Alshayeba , Sajjad Mahmooda "Exploring software security approaches in software development lifecycle: A systematic mapping study",Computer Standards and Interfaces, Elsevier ,pp. 107-115, 2016.
26. Wentao Wang a , Kavya Reddy Mahakala a , Arushi Gupta a , Nesrin Hussein a , Yinglin Wang b " Data on security requirements in open-source software projects", Data in Brief, Elsevier,pp. 1-4,2018.
27. Wentao Wang∗,a , Kavya Reddy Mahakalaa , Arushi Guptaa , Nesrin Husseina , Yinglin Wangb "A linear classifier based approach for identifying security requirements in open source software development" Journal of Industrial Information Integration, Elsevier,pp.34-40, 2018.
28. Zoltán Ádám Mann "Secure software placement and configuration" Future Generation Computer Systems, Elsevier,pp.243-253, 2020.
29. [30] Shohreh Hosseinzadeh∗,a , Sampsa Rautia , Samuel Lauréna , Jari-Matti Mäkelän , Johannes Helvetica , Sami Hyrynsalmi , Ville Leppänen " Diversification and obfuscation techniques for software security: A systematic literature review" Information and Software Technology, Elsevier,pp.72-93,2018.

30. Kalle Rindell *,a , Jukka Ruohonen a , Johannes Holvitie a , Sami Hyrynsalmi b,c , Ville Leppanen ¨ a "Security in agile software development: A practitioner survey" Information and Software Technology, Elsevier,pp.106-118, 2020
31. Liu Ping, Su Jin, Yang Xinfeng" Research on Software Security Vulnerability Detection Technology" International Conference on Computer Science and Network Technology,IEEE, pp.1873-1876, 2011.
32. Masahito Saitoa , Atsuo Hazeyamab* , Nobukazu Yoshiokac , Takanori Kobashid , Hironori Washizakie , Haruhiko Kaiya f , Takao Ohkubog "A Case-based Management System for Secure Software Development Using Software Security Knowledge" International Conference on Knowledge Based and Intelligent Information and Engineering Systems,IEEE,PP.1092-1100,2015.
33. K. Tuma∗ , G. Calikli, R. Scandariato "Threat analysis of software systems: A systematic literature review" The Journal of Systems & Software,Elsevier,pp.275-294, 2018.
34. Adel Mohammad,Ja'far Alqatawna, Mohammad Abushariah "Secure Software Engineering: Evaluation of Emerging Trends" International Conference on Information Technology,IEEE,pp.814-818,2017.