Indian Journal of Science and Technology, *Vol 9(S1), DOI: 10.17485/ijst/2016/v9iS1/89214, December 2016*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Delineation and Elucidation of Security Activities in Agile Software Development

Sushil Kumar[1]* and Ashish Jolly[2]

[1]Department of Computer Science and Engineering, Maharaja Agasen University, Baddi – 174103, Himachal Pradesh, India; sk93recj@gmail.com
[2]Department of Computer Science, Government College, Kaithal – 136027, Haryana, India; ashishjolly76@gmail.com

Abstract

Agile programming advancement process is an arrangement of standards utilized for programming improvement. In Agile programming improvement the arrangements develop through coordinated effort between self-sorting out, cross-useful groups using the fitting practices for their unique situation. This paper suggests guidelines that may be followed in agile software development process. We have conducted a survey approximately 500 agile software developers, around the globe, have taken part in it. In survey a questionnaire related to the various security activities to be incorporated during every phase of agile software development were asked. Based on this survey we have accomplished the most compatible and beneficial security activity that can be incorporated during different phases of agile software development. 80% of agile developers voted for the initial education which is the important security activity to be incorporated during Pre-requirement phase. Similarly 75% said that security requirements during requirement phase, 95% opted for risk analysis during design phase, 80% said coding rules during implementation phase, 62% said identify, perform and implement security tests in testing phase and 77% said final security review in release phase are to be incorporated during different phases of agile software development. The Proposed work overcomes the issues in agile model and security by giving a quick comprehension of the security activities incorporated during different phases of software development. We have bridge-in, the security gap between traditional waterfall model and in-practice agile development model.

Keywords: Agile Software Development, Agile Security, Software Security, Vulnerability, Waterfall Model

1. Introduction

For programming security is the important aspect[1]. As of now, security is a serious problem cause severe because of the expanded unconventionality, accessibility and extensibility[2]. Along these lines, the product engineers are forced to make more secure associations. There had been a critical augmentation in security related programming vulnerabilities as indicated in CERT[3] statistics as shown in Figure 1. Billions of dollars go down the drain because of low quality software produced. The cost is said to be brought around limiting the blunder rate at each stage in programming development and evolution[4]. As an outcome, the last item ought to cost less over its lifetime[4].

Considering the measurements appeared in Figure 1, there is a need to build up a way to deal with programming advancement and that could promise security at every period of programming life cycle[5]. Nevertheless,

*Author for correspondence

settling programming is normally seen as a post improvement movement and very little consideration is paid to it amid the advancement of programming[2]. Incorporated of security parameters with every phase of software development life cycle is a must till its completion and till that is in use[6]. Since programming grows up through its richness cycle, programming advancement techniques ought to give unique regard for the security part of the product[1].

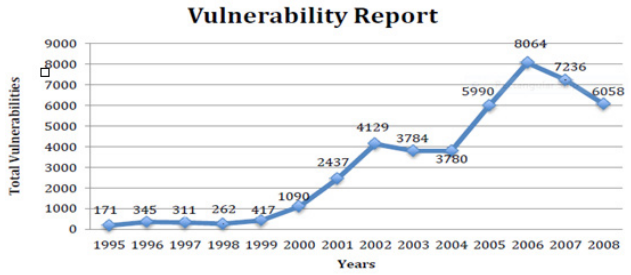

Figure 1. Vulnerabilities reported to CERT centre.

Today's product improvement business requires rapid programming conveyance from the advancement squad. In hotel to give fast conveyance of items, associations make changes from their routine improvement way to deal with agile development method[7]. In this manner changes are done as a push to expand the power of programming advancement and are in like manner an after effect of more online circulated programming items and stages. Agile software development has impacted on the process of software development around the globe[8].

Although agile method is widely used in the current software industry, from a security perspective, this approach is reported to having many flaws related to secure software development[9]. Therefore it is difficult to put into action in agile model of development.

2. Software Security

Security in software development is an experiment .The reason for this is that argumentative networked environment and mobile code threats. The security can further be classified into two categories[10].

- Security of Software
- Application Security

Security of software is about building secure software and application security is securing software after development

In order to develop better software, we need to plug the problems during design and execution phase of software development, rather than taking care of the problems after application is developed. This kind of methodology will curtail down the development cost of a software[11].

3. Security Engineering Process

This involves the group of activities achieved for developing, maintaining and delivering a secure software; security activities can be either iterative or sequetial[12]. The aim is to generate robust, free of bugs software by using systems, procedures, tools and approaches for tackling security concerns in several phases of software development[13,14].

Secure software procedures when applied during the development of software are able to:

- Perform well during the disorders by opposing the manipulation of flaws in the software program or by enduring the catastrophe.
- Damage control from an attack-initiated problem failure and recuperate rapidly from those failures that the software was in capable to endure.

4. Vulnerability of Software

The flaws in design, operation and configuration of software are vulnerability[15]. There is reduction in the system information warranty due to software vulnerabilities.

The people are generally technological aware, so that attackers can be kept away, they are aware of firewalls, anti-viruses and various cryptography techniques, but in spite of all this, attacks are unavoidable. The reason for this is insecure software[16]. The attackers generally exploit the security loop holes. The vulnerability arises because of defects during design and implementation phase of software development[25]. The major weaknesses of software comprise overflow of buffer, number over flow, race condition, format string bugs, poor unique number generator, SQL injections, Denial Of Service (DOS) and misplaced trust[17,18].

Figure 2 below shows how a programmer creates bugs in the software development processes and how an attacker exploits the vulnerability found in a program.

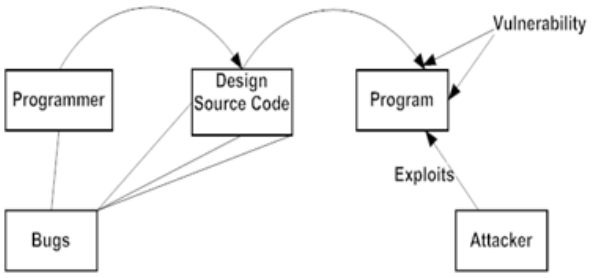

Figure 2. Software defects.

- Bug - An error, flaw, failure or fault in a computer program that causes it to produce an incorrect result, or to behave in unintended ways.
- Vulnerability - A weakness which allows an attacker to reduce a system's information assurance[19].
- Exploit - Use of software, data, or commands to "exploit" a weakness in a computer program to carry out some form of nasty intent.

5. Agile Development Methodology

Two C's i.e. communication and coordination plays vital role in successful implementation of project. So therefore there has to be regular communication and coordination between all the stakeholders of the project[20,21]. Software development is broadly classified into two categories: waterfall development approach and agile development approach.

Waterfall model[23], frequently utilized as a part of programming improvement procedure, is an effective outline process which is flowing downwards (like a waterfall) through the phases of conception, initiation, analysis, design, construction, testing, implementation and maintenance[22].

Then simply again, software model is "an iterative and progressive way to cope with programming progression which is self-sorting away groups inside a government system with "simply enough" function that offer stop notch arrangements in a financially savvy and auspicious way which fulfills the ever growing demand of change[23,24].

In the latest current few years, substantial parts of software business have moved programming improvement strategy from an inflexible design to a far more adaptable agile programming advancement process[25]. Many agile software development methodologies like extreme programming (XP), scrum, Feature Driven Development (FDD), lean software development, crystal methodologies, Dynamic Systems Development Methodology (DSDM), are available[26]. While there are numerous contrasts between these philosophies they rely upon some normal standards, most advance improvement, cooperation, joint effort and procedure versatility for the duration of the life span of the undertaking.

The Twelve principles of agile doctrine[27,28]:

- Consumer loyalty through ahead of schedule and uninterrupted delivery of valuable software
- Welcome evolving necessities, even past due being developed
- Deliver working software usually, from two or three weeks to several weeks, with a propensity to the shorter timescale
- Specialists and engineers must interact personally every day all through the endeavor
- Build tasks around spurred people, give them environment and strength ,they need and trust them to care for business
- Face-to-face discussion is the most productive and successful technique for passing on data to and within an improvement group[29].
- Working applications are the essential parameter of advancement

Sustainable development, Simplicity, Self-sorting out groups which meet at regular intervals for finding out the best possible solution to a problem. These are the features that distinct agile methodology from waterfall development method[30].

As depicted in Figure 3 agile software designers work intimately with their partners and customers to understand their requirements. Additionally, they utilize pair programming to realize and test their solution. Then, the patron gives expert advice on the solution provided and in this way infusing deformities each and every stage during the development. This implies that agile designers are highly skilled for the accomplishment of project[31,32].

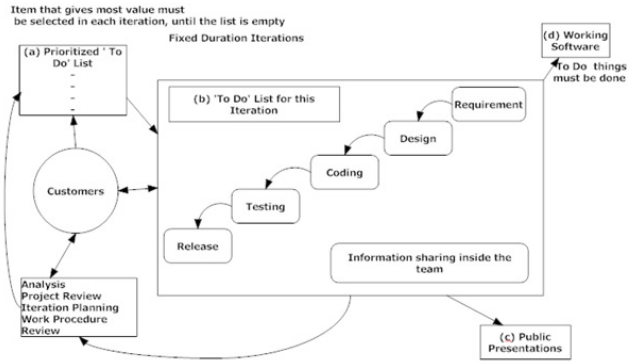

Figure 3. Agile methodology for software development.

Tasks are broken into small fragments[34] as shown in Figure 3 (a) List is prepared for the entire Do's" and priority is accordingly assigned. The module is accomplished based on priority accorded for entire software development life cycle. (b). after completion of iterations on various phases of SDLC an operational software product is shown to patrons for feedback (c). The iterations may not improve the functionality of the released version, but actually it removes the bug. Thereby reducing the risk and aids the project to acclimate to deviations swiftly. (d). multiple iterations are required for release of new featured software product[35].

6. Agile Security

The analysts' claim that agile methods are insufficient for security critical projects but due to competitive reasons they are widely used for web and network related development[36]. One of the misapprehension can be that security hinders the expansion process another important reason can be that there is no specific security procedures specifically designed for agile software development[37]. Therefore agile software development is not having no option than to use security procedures designed for waterfall model based software development. These heavy weight security parameters are not valid for agile development, as they are designed specifically for waterfall model based development[38]. This might be possible due to dissimilar operations between traditional SE Operations and Agile operations. This means that agile methods support brief progress boosts that adapt easily to change, while in SE processes change is difficult and bugs are curtailed down by using heavy security parameters[38,39]. But due to need of industry we need to follow agile software development by providing high quality security parameters without altering the agility of a process.

7. Research Methodology

The blind of quantitative and qualitative processes are used for this research, the literature review is carried out and important security and agile characteristics[40] are identified and accordingly survey is designed[41]. The research process is depicted in the following subcategories.

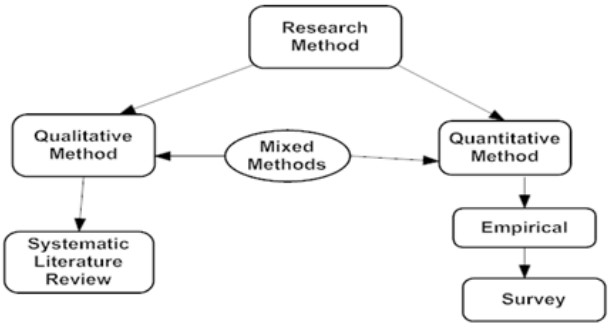

Figure 4. Research design overview.

8. Aims and Objectives

The aim of survey is to identify important security activities in every phase of software development, as prevalent in recent software industry and to recommend attuned security activities for agile software development model.

The goals are accomplished by the following:

1. Agile methodology security features are being studied.
2. Traditional software engineering processes are evaluated
3. Important software engineering operations are identified
4. Security activity accomplished by software engineering operation is to be pin pointed.
5. Security activities are categorized as per different development stages.
6. Survey form is drafted based on the discovered security activities
7. Survey is carried out.
8. Propose the most attuned security achievements to the model
9. Suggest the future scope for improvement.

9. Survey Construction

The survey is designed on Google form. A study consisting of 06 questions is developed. All questions were referred to various security activities to be incorporated during every stage of agile software development. Before taking up the survey the respondents are required to furnish details regarding their Name, corporation, and email id and so forth In addition, a cover sheet elucidating the perseverance of the analysis and techniques and the interpretation of every security activity is appended. Due to confidentiality[25] and promise to share the previous survey result to the respondent, helped us to get good response from the participants. A lot more than 450 Software Professionals have participated in this study.

The outcome of the study is as follows:

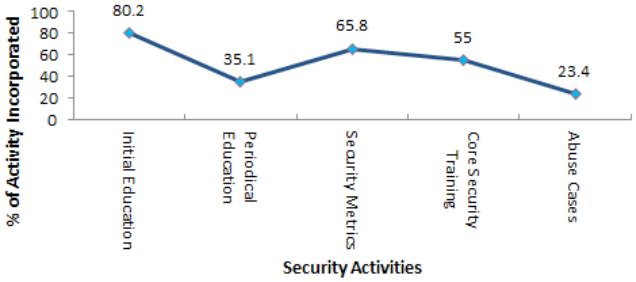

Figure 5. Security incorporated during pre-requirement phase.

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

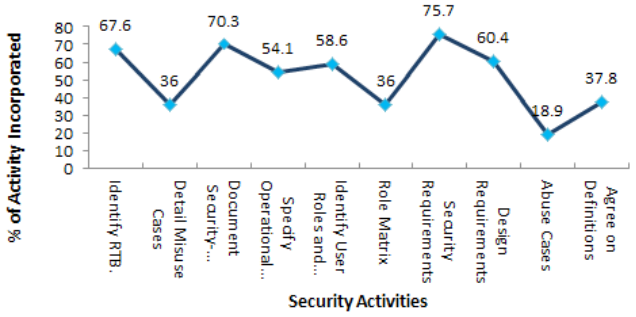

Figure 6. Security activities incorporated during requirement phase.

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

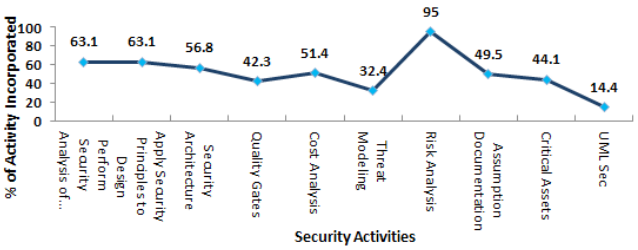

Figure 7. Security activities incorporated during design phase.

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

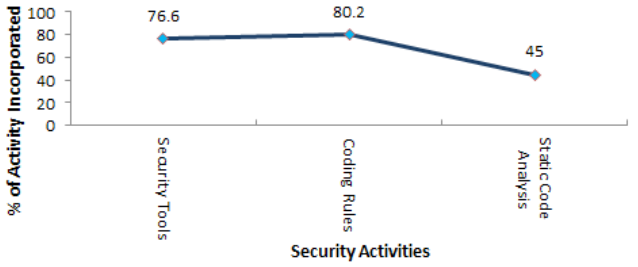

Figure 8. Security activities incorporated during implementation phase.

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

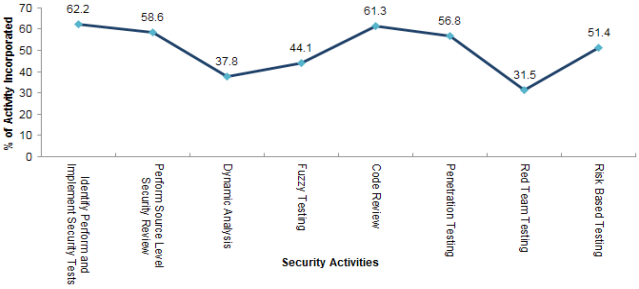

Figure 9. Security activities incorporated during testing phase.

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

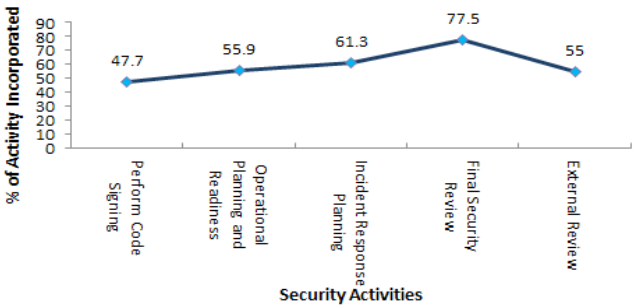

Figure 10. Security activities incorporated during release phase.

Figure 10. shows that the security activities which are important during release phase are Final Security Review and Incident Response Planning.

The following are the finding related to most compatible and beneficial security activity that can be incorporated during different phases of Agile Software development.

Table 1. Compatible security activities during various software development phases

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

10. Conclusion

Security is a critical quality parameter, so cannot be overlooked and must be taken care during software development life cycle. As such there is no security processes specifically designed for agile software development, therefore the security aspect is slightly ignored. The study shows that agile industry is forced to use traditional waterfall SE processes for producing dependable software. The study which we have carried out identify the important security activities phase wise ,which once integrated during every phase of Software engineering process will build good quality secure agile software.

11. References

1. Keramati H, Mirian-Hosseinabadi SH. Integrating software development security activities with agile methodologies. In Institute of Electrical and Electronics Engineers (IEEE) /ACS International Conference on Computer Systems and Applications; 2008 Mar 31. p. 749–54.
2. Siponen- M, Baskerville R, Kuivalainen T. Integrating security into agile development methods. In the Proceedings of the 38th Annual Hawaii International Conference on System Sciences, Institute of Electrical and Electronics Engineers (IEEE); 2005 Jan 3. p. 185a.
3. CERT statistics [Internet]. 2016 [updated 2016 Sep 27; cited 2016 Oct 7]. Available from: https://www.cms.gov/Research-Statistics-Data-and-Systems/Monitoring-Programs/Medicare-FFS-Compliance-Programs/CERT/index.html?redirect=/cert.
4. Azham Z, Ghani I, Ithnin N. Security backlog in scrum security practices. In Institute of Electrical and Electronics Engineers (IEEE) 5th Malaysian Conference in Software Engineering (MySEC); 2011 Dec 13. p. 414–7.
5. Beznosov K, Kruchten P. Towards agile security assurance. In the Proceedings of the Association for Computing Machinery (ACM) workshop on New Security Paradigms; 2004 Sep 20. p. 47–54.
6. Musa SB, Norwawi NM, Selamat MH, Sharif KY. Improved extreme programming methodology with inbuilt security. In Institute of Electrical and Electronics Engineers (IEEE) Symposium on Computers Informatics (ISCI), Kuala Lumpur; 2011. p. 674–9.
7. Zurko ME, Simon RT. User-centered security. In the Proceedings of the Association for Computing Machinery (ACM) Workshop on New Security Paradigms; 1996 Sep 17. p. 27–33.
8. Wäyrynen J, Bodén M, Boström G. Security engineering and extreme programming: An impossible marriage?. In the Conference on Extreme Programming and Agile Methods, Springer Berlin Heidelberg; 2004 Aug 15. p. 117–28.
9. Davis N. Secure software development life cycle processes: A technology scouting report. Carnegie Mellon University, USA; 2005 Dec. p. 1–23.
10. Chivers H, Paige RF, Ge X. Agile security using an incremental security architecture. In International Conference on Extreme Programming and Agile Processes in Software Engineering, Springer Berlin Heidelberg; 2005 Jun 18. p. 57–65.
11. Bartsch S. Practitioners' Perspectives on Security in Agile Development. In Sixth International Conference on Availability, Reliability and Security (ARES), Institute of Electrical and Electronics Engineers (IEEE); 2011 Aug 22. p. 479–84.

12. Singhal A. Integration analysis of security activities from the perspective of agility. In Institute of Electrical and Electronics Engineers (IEEE) Agile India; 2012 Feb 17. p. 40–7.
13. Zadeh J, De Volder D. Software development and related security issues. In the Proceedings of the Institute of Electrical and Electronics Engineers (IEEE) SoutheastCon; 2007 Mar 22. p. 746–8.
14. Gilliam DP, Wolfe TL, Sherif JS, Bishop M. Software security checklist for the software life cycle. In the Proceedings of the Twelfth Institute of Electrical and Electronics Engineers (IEEE) International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE); 2003 Jun 9. p. 243–8.
15. Gregoire J, Buyens K, Win BD, Scandariato R, Joosen W. On the secure software development process: CLASP and SDL compared. In the Proceedings of the Institute of Electrical and Electronics Engineers (IEEE) Computer Society Third International Workshop on Software Engineering for Secure Systems; 2007 May 20. p. 1.
16. De Win B, Scandariato R, Buyens K, Grégoire J, Joosen W. On the secure software development process: CLASP, SDL and touch points compared. Information and Software Technology. 2009 Jul 31; 51(7):1152–71.
17. Miyachi C. Agile software architecture. Association for Computing Machinery (ACM) SIGSOFT Software Engineering Notes. 2011 Mar 14; 36(2):1–3.
18. Ge X, Paige RF, Polack FA, Chivers H, Brooke PJ. Agile development of secure web applications. In the Proceedings of the Association for Computing Machinery (ACM) 6th international conference on Web engineering; 2006 Jul 11. p. 305–12.
19. McGraw G. Building secure software: better than protecting bad software. Institute of Electrical and Electronics Engineers (IEEE) Software. 2002; 19(6):57–8.
20. McGraw G. Software security. Institute of Electrical and Electronics Engineers (IEEE) Security and Privacy. 2004 Mar; 2(2):80–3.
21. Mead NR, Allen JH, Barnum S, Ellison RJ, McGraw G. Software Security Engineering: A Guide for Project Managers. Addison-Wesley Professional; 2004 Apr 21.
22. Ambler S. Agile modeling: effective practices for extreme programming and the unified process. John Wiley and Sons; 2002 Aug 14.
23. Sletholt M, Hannay J, Pfahl D, Langtangen H. What do we know about agile practices in scientific software development. Computing in Science and Engineering. 2011; 14(2):24–37.
24. Benediktsson O, Dalcher D, Thorbergsson H. Choosing a development life cycle: Comparing project and product measures. International Conference on Software and Systems Engineering and their Applications. 2004 Nov 30; 1(3):1–14.
25. Flechais I, Sasse MA, Hailes S. Bringing security home: a process for developing secure and usable systems. In the Proceedings of the Association for Computing Machinery (ACM) workshop on New security paradigms; 2003 Aug 13. p. 49–57.
26. McGraw G. Software security: building security in. Addison-Wesley Professional; 2006.
27. Baca D, Petersen K, Carlsson B, Lundberg L. Static code analysis to detect software security vulnerabilities-does experience matter?. In Institute of Electrical and Electronics Engineers (IEEE) International Conference on Availability, Reliability and Security (ARES); 2009 Mar 16. p. 804–10.
28. Goertzel KM, Winograd T, McKinley HL, Holley P, Hamilton BA. Security in the software lifecycle. Making Software Development Processes—and Software Produced by Them—More Secure; 2006 Aug. p. 1–9.
29. Dyba T, Dingsoyr T. What do we know about agile software development?. Institute of Electrical and Electronics Engineers (IEEE) software. 2009 Sep; 26(5):6–9.
30. Davis N, Humphrey W, Redwine ST, Zibulski G, McGraw G. Processes for producing secure software. Institute of Electrical and Electronics Engineers (IEEE) Security and Privacy. 2004 May; 2(3):18–25.
31. Daud MI. Secure software development model: A guide for secure software life cycle. In the Proceedings of the International Multi Conference of Engineers and Computer Scientists. 2010 Mar 17; 1:17–19.
32. Bhardwaj D. Scrumming it up. A Survey on Current Software Industry Practices; 2010.
33. Buyens K, Scandariato R, Joosen W. Process activities supporting security principles. In Institute of Electrical and Electronics Engineers (IEEE) 31st Annual International on Computer Software and Applications Conference (COMPSAC). 2007 Jul 24; 2:281–92.
34. Kitchenham B. Procedures for performing systematic reviews. Joint Technical Report, Keele University, UK. 2004 Jul; 33(2004):1–26.
35. Bhatia MP, Kumar A, Beniwal R. Ontologies for software engineering: past, present and future. Indian Journal of Science and Technology. 2016 Mar 22; 9(9):1–16.
36. Su Y, Tang H. Construct stereoscopic teaching system of software engineering course based on CDIO. Indian Journal of Science and Technology. 2012 Dec 1; 5(12):3788–91.
37. Wankhede HS, Kiwelekar AW. Qualitative assessment of software engineering examination questions with bloom's taxonomy. Indian Journal of Science and Technology. 2016 Feb; 9(6):1–7.
38. D'Souza MJ, Rodrigues P. Extreme pedagogy: An agile teaching-learning methodology for engineering education.

Indian Journal of Science and Technology. 2015 May 1; 8(9):828–33.

39. Alshareet OM. An empirical study to develop a Decision Support System (DSS) for measuring the impact of quality measurements over Agile Software Development (ASD). Indian Journal of Science and Technology. 2015 Jul 7; 8(15):1–17.
40. Gandomani TJ, Nafchi MZ. Agility assessment model to measure agility degree of agile software companies. Indian Journal of Science and Technology. 2014 Jul 23; 7(7):955–9.
41. Abdelhaq M, Hassan R, Ismail M. A study on the vulnerability of AODV routing protocol to resource consumption attack. Indian Journal of Science and Technology. 2012 Nov 1; 5(11):3573–7.