Int. j. inf. tecnol. (March 2022) 14(2):1117–1130
https://doi.org/10.1007/s41870-020-00446-4

ORIGINAL RESEARCH

Identification and integration of security activities for secure agile development

Amit Sharma[1] · **R. K. Bawa**[1]

Received: 16 May 2019 / Accepted: 26 February 2020 / Published online: 5 March 2020

Abstract Agile software development is receiving the attention of software developers and researchers thanks to its fast software delivery and flexible development plan capabilities. The fast release and simplified documentation thus leads to the preference of the agile development model over several other traditional models. This, however, also raises critical concerns about the security issues. In this research work, we propose a framework for secure agile development. The selection of development methodology among agile versus plan driven approaches and the particular agile development method among Extreme Programming (XP), Crystal Clear, Scrum, Lean Development, Dynamic Software Development Method and Feature-Driven Development is made on the basis of the specific requirements of the project using empirical methods like AHP and PROMETHEE. Systematic Literature Review (SLR) and survey study are used to obtain the authentic industrial feedback, followed by the application of non-parametric statistical tests to identify and select the most suitable and beneficial security activities from well known security engineering processes like CLASP, Common Criteria, Cigital Touchpoints and Microsoft's SDL. A lightweight method is also introduced for integrating these security activities identified from SLR and survey study, using a dynamic integration algorithm without compromising the agility of the process. The proposed framework for integration of these security activities is implemented in java to automate the entire process and provides maximum benefit at a low integration cost.

Keywords Agile development · Security engineering · Agile security · CLASP · Common Criteria · Cigital Touchpoints · Microsoft SDL

1 Introduction

Agile development follows an informal and flexible approach which is different from plan-driven development which relies on extensive formalization and documentation. A very limited amount of formalization is required in agile development wherever necessary. It usually lays emphasis on informal, dynamic and tacit knowledge-driven methods to develop high business-value projects. The Agile Manifesto [1] clearly describes these core values. The highest priority is given to continuous and early delivery of the software to satisfy the customer. Changing requirements are welcomed, even late in the development. For the customer's competitive advantage, agile processes accommodate these changes. The primary measure of the progress is the working software. The best designs and architectures evolve from self-organizing teams.

Although the agile development approach is getting acceptance across the globe, it is found to have certain disadvantages related to security in software development [2, 3]. The main security issues with agile development arise from the informal communication, self-organizing team, tacit knowledge-driven methods and trust on individuals, as they conflict with the assurance and quality activities as required by conventional secure software

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s41870-020-00446-4) contains supplementary material, which is available to authorized users.

✉ Amit Sharma
amitsharmapkl@gmail.com

R. K. Bawa
rajesh.k.bawa@gmail.com

[1] Punjabi University, Patiala, Punjab, India

development methods. Some limitations are imposed on the projects by the agile processes, as all the requirements are not known in advance. Thus, it is not possible to create a complete picture of a product. Security Engineering (SE) processes are the collection of activities performed to deliver secure software, throughout the different phases of development process. As stated in many other studies [4, 5], the restriction of having the complete overview of the project creates hindrance while performing Security Engineering practices in an agile model. The fact is that there are no specific Security Engineering processes which are developed for agile development methods; companies are using existing security processes developed for waterfall model in their agile model.

In this paper, we have used a Systematic Literature Review and Survey Study to identify the most suitable and beneficial security activities from well-known waterfall Security Engineering processes like CLASP, Common Criteria, Cigital Touchpoints, Microsoft SDL and some other commonly used practices. Developers working in different agile development projects have participated from all over the world, and the majority of them are from India. By using the survey, we aim to identify and select the security activities of the existing SE processes which are most suitable in terms of benefit as well as cost, and also to identify the activities which cannot be performed in an agile model. The motive of this study is to use the experience of the industry to select those activities which can be easily integrated with agile projects. Instead of blindly integrating the security practices into agile processes, the framework provides an overall approach for secure development starting from the decision of selecting a development methodology between agile and plan-driven approach and then the selection of the most suitable agile development method among Extreme Programming (XP), Crystal Clear, Scrum, Lean Development, Dynamic Software Development Method (DSDM) and Feature-Driven Development (FDD) for the given particular project using empirical methods like AHP and PROMETHEE. Artificial Neural Network and Fuzzy Logic are also used to compensate the subjectivity of human decision-making. The main menu with sub menus is shown in Fig. 1. Multi criteria decision making in itself is a vast topic, hence making a detailed discussion is out of the scope of this paper. However, the most useful and widely used security activities are identified from the Systematic Literature Review and Survey Study. Based on the level of preference of the security activity in terms of cost as well as benefit, the statistical tests were conducted to find out the most beneficial and suitable security activities for each development phase. Thereafter, these security activities were selected for integration with the agile activities using a dynamic integration algorithm. Thus, this framework provides an integrated approach which covers the entire life cycle of a project and considers security throughout as its integral part. It is implemented in java with more than 15,000 lines of code to automate the entire process.

The rest of the paper is organized as follows. Section 2 presents the related work done in the field and Sect. 3 discusses the Security Engineering Processes. Section 4 covers the research methodology including a Systematic Literature Review and Survey method. The results from the empirical studies are provided in Sect. 5 along with the identification and then integration of security activities. Finally, Sect. 6 concludes the paper with the summary of the major findings and suggestions for the future work.

2 Related work

Most of the previous work regarding security issues in agile has focused mainly on literature survey and few researchers have also used industry feedback along with empirical methods for concluding their results. Beznosov and Kruchten [6] worked towards agile security assurance. They studied the mismatches between agile development methods and the security assurance techniques. Based on the literature studies, few techniques which are identified fit well with agile methods and the other few are rejected since they mismatch with agile. Siponen et al. [7] demonstrated how the security features can be integrated into agile methods. They identified the key security elements in agile software development and then illustrated how these can be implemented in FDD. Keramati et al. [8] used two SE processes namely, Comprehensive Lightweight Application Security Process (CLASP) and Microsoft SDL to identify and evaluate security activities and practices. The paper has presented an algorithm, which uses calculation of Agility Degree based on nine agility features and further using this for extending agile processes with security activities. Another approach is to integrate security activities from well established SE processes. Baca [9, 10] and Carlsson [11] studied different known Security Engineering processes and identified the security activities. Then, they produced a security based agile development process which is based on the activities from the famous security engineering processes. They claimed to provide the desired results by enhancing the security and without obstructing the agile process. Bartsch [12] provided perspectives from the practitioner's viewpoint on agile development security and report on the qualitative, exploratory results of interviews. Their results expanded the conceptual prior work and proposed concentrating on sufficient customer engagement, knowledge and competence of developer security and continuous improvement of the software development process. Shackleford [13] looked

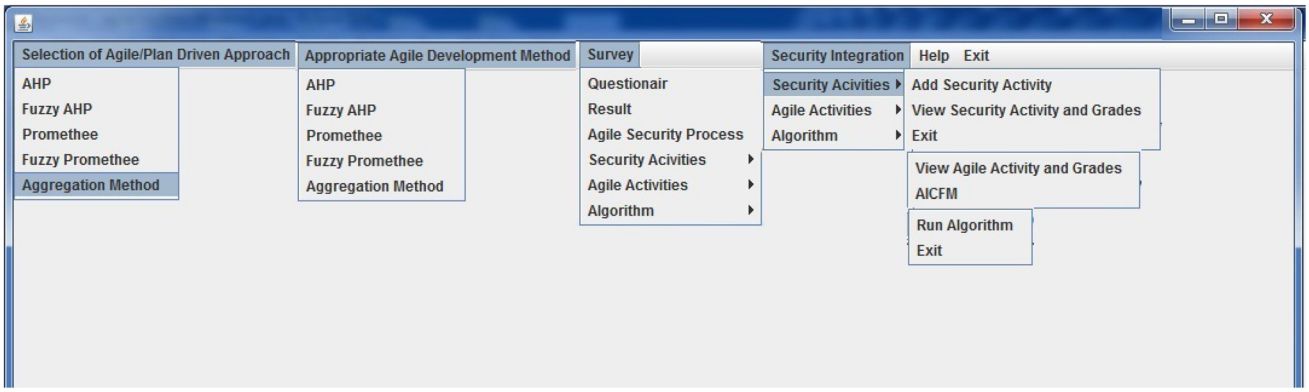

Fig. 1 Main menu of the framework for secure agile development

at the development of software from both development and security perspectives, and then analyzed which tools and technologies might contribute towards incorporating security into development cycles without overloading the system or causing too much overhead. Savola et al. [14] carried out an industrial pilot study in agile software development on risk-driven security metrics. The results showed that risk-driven security metrics had a practical potential, especially in providing early visibility of security effectiveness and performance. Wolff [15] explained how formal methods can be applied to the Scrum, which is one of the most popular agile development processes. The experiences of using a strategy variant in an industrial situation were summed up. GAO [16] submitted a report on federal challenges and effective practices in the implementation of agile processes. GAO recognized 32 techniques and practices beneficial in applying agile methods of software development to IT projects. Munetoh and Yoshioka [17] introduced a framework for developing Web applications through a model-assisted security testing process. They developed a tool called "RailroadMap," which extracted behavior model automatically from the Ruby-on-Rail code base. Rindell [18] presented a literature review of a selected set of agile methods to develop secure software. The results show a wide and well-documented adaptation of security activities in the development of agile software, with the observed activities covering the entire security life cycle of development. Harrison and Tzounis [19] discussed the pros and cons of Waterfall before moving on to the Agile Scrum methodology. A framework was created using the Application Security Verification Standard (ASVS) of the Open Web Application Security Project (OWASP). Rindell [20] worked on a project case consisting of a multi-team, multi-location, security engineering work, and development efforts, which were carried out according to the Scrum Framework. In this case research, the experiences of combining security engineering with agile development are reported, challenges are discussed and some security improvements to the Scrum are proposed.

This section covers the related work in detail along with the major differences with our work wherever applicable. Most of the above research work takes security aspects in bits and pieces that on a very small sample size. There is no comprehensive approach which takes care of the security in agile development throughout the development life cycle. Beznosov and Kruchten worked towards agile security assurance but eventually could not suggest any concrete solution to gap down the mismatches and only proposed a compromise for these mismatches. Siponen et al. suggested the use of abuse cases by illustrating how they can be implemented in FDD. The scope of this approach is very narrow as the use of abuse cases is just one of the several security measures which should be taken for secure agile development. Keramati et al. suggests security activities from two SE processes but without considering the fact that these security activities are originally meant for plan-driven approach, and the selection criteria for calculating the agility of these activities is also decided by a project analyst, which again does not fit with the agile principle of team work and decision-making. Dejan Baca suggested the integration of security activities from three SE processes by conducting an interview of 12 developers from a single telecommunication company Ericsson AB. His work focused mainly on Scrum agile method and a total of 10 activities were selected. In another work, Bengt Carlsson conducted a survey to identify and evaluate the security activities mainly from telecommunication and software companies with a small sample size of 41. As a result, a total of 16 security activities were identified. Firstly, the sample size in both the studies is very small, and this is not enough to generalize the results for such a huge software industry. They have applied t test for their analysis, but for an ordinal data, a non-parametric test is more powerful and appropriate to produce better results. Moreover, some of the selected security activities like counter measure graphs

seem to be selected because of local influence as they are the result of their own work. This makes it difficult to generalize them for the entire software development industry. In our work, we have considered four SE processes namely like CLASP, Common Criteria, Cigital Touchpoints, Microsoft SDL. Systematic Literature Review is done in order to get a firm base for conducting survey study. To get a large number of participants, we have used professional networks like Academia, Facebook, LinkedIn and Twitter. A total of 97 respondents, which provide a reasonable sample size, participated in the survey. As a result, the 20 security activities are selected that are both beneficial and compatible with agile process. Due to the data being ordinal, its normality was not fulfilled. Thus, the non-parametric test Wilcoxon–Mann–Whitney test was applied to find the significant difference between any two security activities. Effect-size was also calculated. Although security activities identified through the survey are selected, there is also a provision of adding and removing security activities, depending upon whether the project require some other more specific security activities. Further, to automate the process, an additional algorithm is also provided to integrate the selected security activities with agile activities.

3 Security engineering processes

This section covers the four Security Engineering processes namely the Microsoft's Secure Development Lifecycle (SDL), Comprehensive Lightweight Application Security Process (CLASP), Cigital Touchpoints (CT) and Common Criteria (CC). These are the four most popular and widely-accepted SE processes throughout the world. Different activities have been extracted from these SE processes and have been mapped to different phases of the development cycle starting from pre requirement till release. This information has also been presented to the respondents through our website (http://www.agilesecurity.esy.es). This gives the information about the activities such as the definition of the security activity along with its purpose and use.

Microsoft Security Development Lifecycle Process [21] has been proposed by Microsoft to increase the reliability of the process along with reducing the maintenance cost. This software development process has basically been derived from the classical spiral model and has been used for reducing software security related bugs. Many attempts have been made in order to make it more Agile friendly [22]. It contains the following security activities: Role Matrix, Security Requirements, Core Security Training, Quality Gates, Threat Modeling, Design Requirements, Attack Surface Reduction, Cost Analysis, Security Tools, Coding Rules, Static Analysis, Dynamic Analysis, Fuzzy Testing, Incident Response Planning, Attack Surface Review, Code Review and Final Security Review.

Common Criteria [23] is an ISO certified mature process. It is a framework that allows the user of the computer system to specify their security assurance and functional requirements. It consists of the following security activities: Agree on Definitions, Security Requirements, Critical Assets, Risk Analyses, UMLSec, Requirements Inspection, and Repository Improvement.

Cigitel Touchpoints [24] are described as lightweight best practices that are applied to various software artifacts. It improves the various security and quality aspects of the end product. It consists of the following security activities: Security Requirements, Risk Analyses, Abuse Cases, Assumption Documentation, Static Code Analyses, Penetration Testing, Red Team Testing, External Review, and Risk Based Testing.

CLASP is a Security Engineering process [25] that originates from the OWASP i.e. Open Web Application Security Project. It consists of the following security activities: Periodical Education, Initial Education, Detail Misuse Cases, Identify Resources and Trust Boundaries, Perform Security Analysis of System Requirements and Design, Security Metrics, Specify Operational Environment, Identify Global Security Policy, Identify User Roles and Resource Capabilities, Identify Attack Surface, Apply Security Principles to Design, Security Architecture, Perform Code Signing, Identify, Implement and Perform Security Tests, Perform Source-level Security Review, Operational Planning and Readiness.

Few other (O) security activities that are being used in the industry but are not a part of any of the above SE processes are: Redesign of the Internal Development Process, Meets Existing Security Framework, Adaptive Monitoring, Acceptance Criteria, Risk Metrics, Security Measurement Based on Risk Indicators, Pair Programming, Automated Acceptance and Unit Tests, Countermeasure Graphs.

A total of 65 security activities have been included out of these Security Engineering processes. Because of redundancy, incompatibility and results of previous studies, 35 security activities, which could be used for agile model, have been selected for further consideration in survey study. Although, depending upon the requirement of a particular project there is always a chance to include or exclude certain security activities. Therefore, during integration, provision is made to include or exclude such security activities as shown in Fig. 1. Furthermore, to diversify the scope of the study and to get comprehensive data about the other widely used security activities, two open-ended questions are also asked in the survey. Question 7 collects data regarding the Security Activities/

Methods which are used in the projects taken up by the participants and question 15 further asks for the recommendation regarding the security in agile development process. Hence, many other security activities have come out as the result of these questions, but they are subject to the need of a particular project, and for coming to any concrete conclusion, it needs further study and detailed analysis which is out of the scope of this paper.

4 Research methodology

The research work has been carried out using two approaches shown in Fig. 2. The use of the qualitative method is followed by the use of quantitative method. Systematic Literature Review is used from the qualitative method which focuses on the identification and analysis of data in different forms, which is usually non-numeric. Empirical approach of survey study has been used from the quantitative method. The main advantage of quantitative method is that it allows the statistical analysis for more accurate results using statistical tools like SPSS.

4.1 Systematic Literature Review

Systematic Literature Review is designed in three steps which include Planning, Conducting and Reporting. In the first step, a review protocol is developed after identifying the need of the review. In the second step, Data Extraction strategy, quality assessment and data analysis are performed. Finally, the third step is to report the review in a single phase. From the result of this Systematic Literature Review, the high profile Security Engineering processes and their activities are identified for further investigation. Moreover, the previous work done in this field is also identified and is used to further enhance the findings.

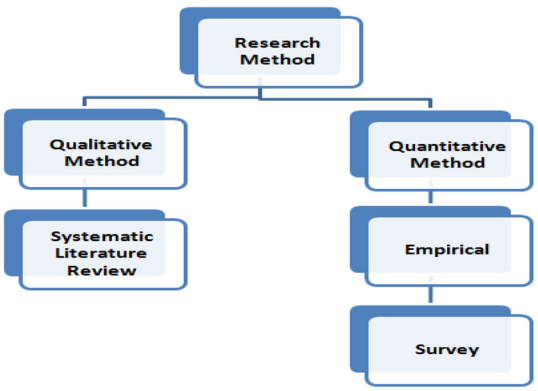

Fig. 2 Overview of research design

4.1.1 Need for Systematic Literature Review

The primary reasons for performing a Systematic Literature Review include identifying any deficiencies in the present research areas in terms of security in agile development and identifying the world-widely used Security Engineering processes along with their security activities. These security activities are further used for conducting a survey study in order to get the real industry feedback about the practical implementation of these security activities.

4.1.2 Search strategy

The search strategy shown in Fig. 3 is used to find the articles. The keywords to search are shown in Table 1. These are used either independently or in combination with others, to identify the articles related to secure software development, security engineering and security aspects in agile process and agile methods. The trial search is carried out using a variety of combinations of search terms in order to verify the quality of the search string. If the match comes out to be less than 90%, the keywords are modified to carry out the search again. If the match is more than 90%, the papers are stored for further investigation.

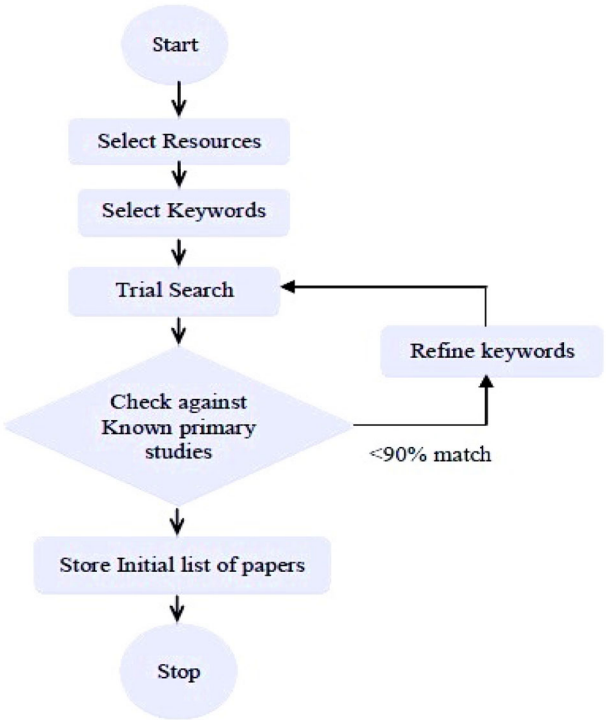

Fig. 3 Search strategy

Table 1 Search terms

S. no.	Search terms
1	Secure software development
2	Security engineering processes
3	Agile development
4	Extreme Programming/XP
5	Scrum
6	Feature Driven Development/FDD
7	3 OR 4 OR 5 OR 6
8	1 AND 7
9	2 AND 7

4.1.3 Selection criteria

The primary goal of the selection criteria for the study was to find and select the papers which were studied initially and were actually helpful in finding the relevant information for agile security aspects. This selection criterion ensured that legitimate and relevant studies could become component of the literature review. Furthermore, it also ensured that the review could not cover irrelevant and insignificant papers. To make this review more comprehensive, studies conducted in both academics and industry were included. The following types of articles were included: journal article, conference article and conference proceeding. It also excluded the articles on general discussions about agile adoption, as the primary focus was on identifying the Security Engineering processes and security practices in agile development. Hence, a selection procedure consisting of three phases was followed. In the first phase, duplicates and non-English papers were discarded from a total of 912 papers collected. During the second phase, a total of 597 papers and 30 non-comprehensive papers were excluded, based on inclusion/exclusion criteria. After this, the remaining 80 papers were selected for full text reading. In the final phase, insignificant papers were discarded after reviewing the full text. At the end, after following the above steps, only 20 papers remained for further studies.

4.2 Survey study

A survey is one of the best methods to get information and feedback in both qualitative and quantitative forms. It consists of five distinct steps: Design Survey Process, Develop Questions, Test and Train, Collect Data and Analyze Data. An online survey questionnaire has been used for data collection. This is an effective way to collect responses on the current security practices across the world from a large number of practitioner's working in different organizations.

So as to obtain the accurate results, the Survey Design was initiated by identifying the target population which consists of industry professionals with an experience in agile development. The inclusion criterion was formulated to include the professionals working for agile projects and who were familiar with the security aspects of the agile development. The non-probabilistic sampling method was used to gather the data, which included snowball sampling and availability sampling. The major consideration behind choosing this sampling technique against random sampling was the constraints like time, secrecy and unwillingness to respond to unfamiliar sources, faced in case of software engineers.

The next step is the Construction of Survey Instrument. An online survey tool SurveyMonkey was used to design the survey. This tool is one of the most popularly-used online survey tool among professionals because of its advanced features which include filters, crosstabs, data exports, text analysis etc. The survey questionnaire consisted of total 16 questions in all. Four of these questions were regarding the information about the respondents, 4 questions about the agile development efforts made by them, 6 concerning selection and evaluation of security activities and 2 optional questions one of which was open-ended. In addition to these questions, a cover note, which explains the purpose of the survey along with assurance to maintain the confidentiality, has also been attached. A link to our website (http://www.agilesecurity.esy.es) has been mentioned in the survey so as to expound the definitions and other information about the security activities. A total of 97 software professionals from agile industry participated in the survey.

The selection and evaluation of the security activities in the survey were based on the likert scale format. These activities were tested on the basis of two factors which include benefit and cost. The five-grade scale, of which the range varies from 1 (low) to 5 (high), was used. The survey instrument was evaluated and refined using a pilot study conducted with five industry professionals working for agile projects. The respondents were contacted through personal contacts and through other professional networks like Academia, Facebook, LinkedIn and Twitter. Invite features of SurveyMonkey, such as web link, email invitation and hyperlink were used to obtain the data from the respondents. The Reminder feature of SurveyMonkey was also utilized to send reminders to the respondents for completion of the survey. For data validation and access control, the URL of the survey link was shared and each participant was allowed to participate only once. This is one of the features provided by SurveyMonkey.

4.3 Validity evaluation

This section discusses the validity threats to Systematic Literature Review and Survey Study. While designing the study plan, threats to the validity of the results are very important to be taken into consideration so that the appropriate actions can be taken in advance to mitigate them. For SLR, three main threats were identified [26]. These include Threats to the Identification of Primary Studies, Publication Bias and Threat to Selection and Data Extraction Consistency. These validity threats were handled by adopting a transparent and unbiased data collection procedure to conclude the valid results, while considering at the same time, the trade-off of including peer-reviewed articles, accumulating reliable information and by formulation of systematic review protocol. As per Wohlin et al. [27], four important validity threats were taken into consideration for the survey study. These included External Validity, Internal Validity, Construct Validity and Conclusion Validity. First, the threat to external validity mainly concerned with generalizing the survey findings beyond the selected sample. This was mitigated by taking a large sample size and heterogeneous distribution of the samples from diverse industrial settings. Second, the threat to internal validity was mitigated by designing an understandable and readable survey which was further refined by pilot study. Further restriction of filling the survey only once, high completion rate and selection of respondents through professional networks like Academia, LinkedIn, Twitter and use of statistical hypothesis testing reduced the chances of causal relationship between the treatment and the results. Thirdly, the threat to construct validity was mitigated by clearly representing the survey objectives. The definition of each security activity was explained through a link to our website and the aim of the research was clearly defined through cover note, Mono operation bias was avoided by using SLR to collect data prior to survey design and the survey responses were completely anonymous. Finally, the threat to conclusion validity was mitigated by choosing high power statistical tests and the risk of violation of the statistical test assumption was limited by having a sufficiently large sample size. The Help link in the survey along with the provision to clarify any doubts through email and website mitigated the threat to the reliability of measures. Learning effect was minimized through the high response count and a restriction upon filling the survey not more than once.

5 Results and discussion

This section covers the details of the statistical tests applied as well the results produced by them. On the basis of the results, the Security Engineering processes are evaluated and the security activities, which are agile integrable and beneficial, are identified. Finally in Sect. 5.6, the process to integrate security activities using a dynamic integration algorithm is discussed.

5.1 Demographic data

A total of 97 software professionals participated in the survey. A majority of respondents being from India, several others from countries like the USA, Canada, England, Germany and Netherlands participated through LinkedIn, Twitter, Academia and other professional networks. Most of them are working in medium to large organizations like Xerox, HSBC Bank, Infosys, Cognizant, Accenture, Tech Mahindra, Wipro and HCL. More than 40% of the software professionals have an experience of more than 5 years. However 60% of the participants have about 1–3 years of experience in agile development. This shows that software professionals are shifting from traditional development methodology to agile development. In addition, Scrum, XP and Lean have emerged as the most popular agile development methods in use.

5.2 Statistical test

The security activities in each phase were compared with each other to find out the most beneficial and compatible security activity. The hypothesis thus derived mainly focused on whether there exists any significant difference among any two security activities belonging to the same development phase. The statistical analysis was carried out using IBM SPSS (Statistical Package for Social Sciences) statistical version 20. The analysis included frequency tables and bar graphs. All variables like Cost and Benefit were evaluated on the Likert scale. As the data was ordinal, its normality was not fulfilled. As a consequence, the non-parametric inference was used. For such data, the Median was compared using the Mann–Whitney test [28] and the Kruskal–Wallis test [29]. All statistical tests were seen at two-tailed level of significance ($p \leq 0.01$ and $p \leq 0.05$). Thereafter, the Wilcoxon rank test [30] was used. This is a non-parametric statistical hypothesis test used to compare two related samples and to find out whether their populations mean ranks differ. The Mann–Whitney test was used to find a significant difference among cost and benefit groups and then Kruskal–Wallis test was applied to find out the significant difference among the security activities

within a development phase. Wilcoxon rank test was further used to categorize different security activities into strongly preferable (+++), highly preferable (++) and merely more preferable (+) for both cost and benefit. A particular security activity was selected based on whether it is preferable in terms of both benefit and cost.

5.3 Statistical test results

Based on the level of preference of the security activity in terms of cost as well as benefit, the statistical tests have been conducted to find out the most beneficial and suitable security activities for each development phase. Based on the level of confidence, a security activity is categorized into strongly preferable (+++) for confidence level greater than 99.3%, highly preferable (++) for confidence level 99% to $< 99.3\%$ and merely more preferable (+) for confidence level 95% to $< 99\%$, as shown in Table 2.

In the pre requirement phase, all the stakeholders get knowledge about the product as well as the entire development process. For this phase, using Kruskal–Wallis Test, three security activities are taken into consideration with mean ranks, as shown in Fig. 4. Using the Kruskal–Wallis test, a significant difference between the security activities in terms of cost with p value 0.0001 is discovered. Nevertheless, there is no significant difference among them in terms of benefit as the p value is found out to be 0.169.

Initial education is strongly preferable in terms of cost, as compared to other two activities based on the significant difference among them in the Mann–Whitney test with p values 0.002 and 0.0001 respectively. The effect size of initial education come out to be 0.64 as compared to Redesign of the Development Internal Process and Meets Existing Security Framework with effect sizes as 0.26 and 0.06 respectively. In case of benefit, all the three activities provide reasonable benefit even though there is no significant difference found between them. The effect sizes of all the three activities come out to be 0.71, 0.61 and 0.63. Hence, in pre requirement phase, initial education is selected as the most suitable and beneficial security activity. Providing initial education to the team is very useful for the success of the project as it makes the team aware about the importance of security and Security Engineering. This initial education includes the basic security concepts and types of security breaches with their possible solutions. Redesign of the Development Internal Process and Existing Security Framework are beneficial but their implementation requires a huge amount of expensive resources and also poses threat to the agility of the process. Similarly, in the requirement phase, the requirements of the stakeholders are collected, understood and documented. In this phase, a total of 8 security activities are considered and among them 5 security activities namely Role Matrix, Agree on Definitions, Security Requirements, Identify Resources and Trust Boundaries, Specify Operational Environment are strongly preferred with significant difference among them both in terms of cost and benefit. For selected activities the effect sizes come out in the range of

Table 2 Security activities evaluation

Security activities	Cost	Benefit
Pre requirement		
Initial Education	+++	+++
Redesign of the Internal Development Process		+++
Meets Existing Security Framework		+++
Requirement phase		
Role Matrix	++	+++
Design Requirements		+++
Security Requirements	+	+++
Agree on Definitions	+++	+++
Identify Resources and Trust Boundaries	+	++
Specify Operational Environment	++	+++
Adaptive monitoring		++
Abuse Cases	+	
Design		
Security Architecture		+++
Secure Design Principles	+	+++
Assumption Documentation	++	++
Requirements Inspection	++	+++
Risk Analyses	+	+++
Critical Assets		+++
Quality Gates	+	+
Countermeasure Graphs	++	
Acceptance Criteria		+++
Risk Metrics		+++
Security Measurement Based on risk Indicators		++
Implementation		
Coding Rules	++	+++
Security Tools	+	+++
Automated Acceptance and Unit Tests		+++
Static Code Analyses	+	+++
Pair Programming		
Testing		
Dynamic Analysis	++	+++
Security Testing	+	+++
Vulnerability and Penetration Testing	+	+++
Release		
Operational Planning and Readiness	++	+++
Signing the Code	+++	+++
Incident Response Planning	+	+++
Repository Improvement	++	++
Final Security Review		+++

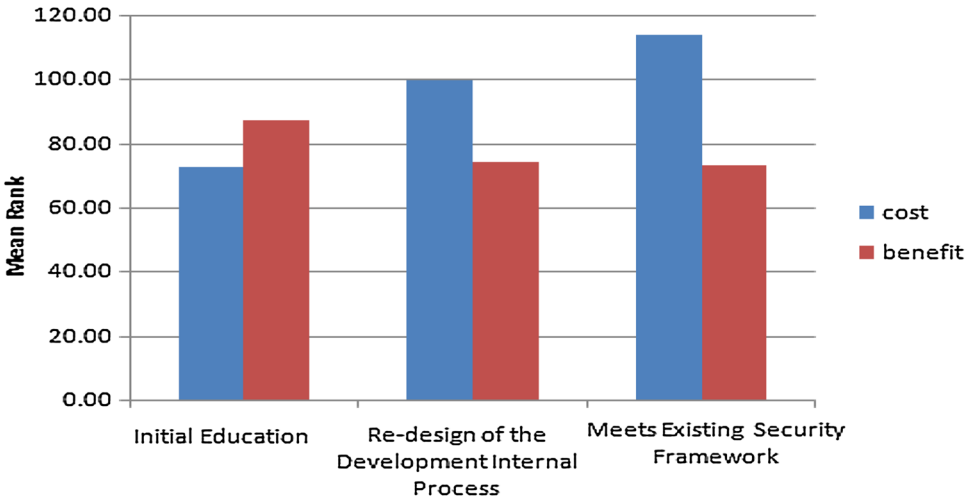

Fig. 4 Security activities analysis in the pre requirement phase

0.5–0.7 in terms of benefit. In terms of cost, the effect sizes come out in the range of 0.4–0.6. In Role Matrix, all user roles and their software access level are identified. Thus, Role Matrix plays a very important role for authentication and authorization at a very low cost. Definitions Agreement defines stakeholders and agrees on a common set of security definitions, together with the definition of organizational security policies and vision. Therefore, the Vision Document artifact is created in this activity to provide a solid foundation for implementing security. Additionally, since most of the resources required are already available for implementing various other projects being undertaken by the organization, it is much less expensive. Security Requirements assign security experts to a particular software project, identify and enumerate security and privacy functionality. This provides an edge to the project's security concerns and the security experts are assigned to different projects simultaneously. As a consequence, this activity provides high benefits with a relatively less cost. Identify Resources and Trust Boundaries defines the system architecture from a network view, identify data resources that a program can use and define where trusted and untrustworthy entities interact. Specify Operational Environment documents the operating environment's assumptions and requirements to assess the impact on security. Both of these activities provide considerable security benefits to an organization with relatively less resource requirement. In next phase, i.e., Design a total of 11 security activities are taken for statistical test and among them 4 security activities are selected. These include Secure Design Principles, Assumption Documentation, Requirements Inspection and Risk Analyses. In the implementation phase, a total 5 security activities are considered out of which 3 securities namely Coding Rules, Security Tools and Static Code Analyses are highly preferable based upon significant difference among them in terms of both cost and benefit. In terms of benefit, the effect sizes of the selected three activities are in the range of 0.65–0.8, and in terms of cost, the effect sizes are in the range of 0.3–0.45. Secure Design Principles make the design of applications more difficult by implementing security design principles and identifying security risks in third party components. Architects, designers and analysts should recognize potential attacks and clearly document assumptions in order to assess the effect on security. Requirements Inspection is performed to validate all the artifacts produced and is produced as a validation report. Its objective is to evaluate the quality of the work and outcomes of the team. Security analysts discover architectural flaws and prioritize them so that suitable mitigation can actually begin. In the early stages, disregarding risk analysis leads to expensive concerns down the path. In Coding Rules, the list of unsafe functions is determined and the same are replaced with safer options. Security Tools describes and publishes a list of authorized security tools to support the project (i.e. open source, commercially available and in-house developed) and related safety checks. In Static Code Analysis, tools for static analysis scan the source code and find prevalent vulnerabilities. Although all the three activities are highly beneficial from the implementation point of view, but coding rules are less costly to implement as compared to security tools and static code analyses. In the testing phase, among 3 security activities all the 3 are selected. These are Dynamic Analysis, Security Testing, Vulnerability and Penetration Testing, based on reasonable benefit and considerable cost of integration. Dynamic Analysis uses dynamic testing tools and track memory corruption, user privilege issues and other critical safety issues. It also explains the outcomes and develops a mitigation approach for a specific software program. Penetration Testing offers a useful insight into the software in its real environment. This is done by simulating

real working conditions and attack patterns. The Release phases consists of total 5 security activities in all, and among them 4 security activities, including Operational Planning and Readiness, Signing the Code, Incident Response Planning and Repository Improvement, are selected. The effect sizes of the selected activities in terms of cost come out to be in the range of 0.4–0.6. In terms of benefit, the effect sizes come out in the range of 0.5–0.75. Operational Planning and Readiness involves user manual writing, security architecture documentation, and so on. Incident response planning offers a response checklist that provides clear action rules for a security emergency. New model elements are found in Repository Improvement throughout the development of previous activities that could be considered to be used in future applications. In addition, the model elements already in the repository could be modified to improve their quality. These security activities are strongly preferable as there are significant differences among them both in terms of benefit and cost. Both of these activities provide considerable security benefits to an organization with relatively less resource requirement.

In a nutshell, 35 security activities are taken initially from a total of 65 security activities identified from Security Engineering Processes and Literature Survey. From these 35 security activities, 20 security activities are short listed based on cost and benefit in terms of significant difference among them through statistical tests. In terms of cost, 3(8.5%) activities are strongly preferable, 9(25.7%) activities are highly preferable and 10(28.6%) activities are merely more preferable. In terms of benefit, 74.2% activities are strongly preferable, 14.3% activities are highly preferable and no activity is selected as merely more preferable. Thus, it is very evident from the statistics that most of the security activities are highly beneficial even though many among them have considerably high cost of integration.

5.4 Security engineering processes evaluation

In this section, based on the results, the four Security Engineering processes Cigital Touchpoints, Microsoft's SDL, CLASP and Common Criteria are evaluated in terms of security activity coverage in agile model. As shown in Fig. 5, Microsoft's SDL and CLASP have dominated the overall activity coverage with 29% and 32% respectively while Common Criteria and Cigital Touchpoints have less security activity coverage with 18% each. Security activities from other SE processes have not been preferred at all. When it comes to considering each individual phase of the development cycle, the pre requirement phase is mainly dominated by Microsoft's SDL and CLASP with 50% coverage. In requirement phase, CLASP covers 44% while Microsoft's SDL and Common Criteria cover 22% each. Cigital Touchpoints covers the least with 11%. In Design phase, Common Criteria and Cigital Touchpoints dominate the activity coverage with 40% each while CLASP coverage is 20%. Implementation phase is mainly covered by Microsoft's SDL and Cigital Touchpoints with 75% and 25% coverage respectively. In testing phase, all the three Security Engineering processes; Cigital Touchpoints, Microsoft's SDL and CLASP, have the maximum activity coverage with 33% each. In release phase, CLASP is the most dominated with 50% while Common Criteria and Microsoft's SDL have security activity coverage with 25% each.

5.5 Agile integrable security activities

In this section, the most beneficial and compatible security activities are selected based on Table 2. The criteria for selecting a particular security activity is that it should be strongly, highly or more preferable in terms of cost. This means that he cost of integrating that security activity should be considerably low and in terms of benefit; it should be either strongly or highly preferable. Thus, it should provide high benefits to the security aspects of agile process. Based on these criteria, a security process covering beneficial and compatible security activities for each phase of software development is shown in Table 3.

5.6 Integration of security activities

In this section we propose an optional customized method to integrate security activities with agile activities apart from the agile security process shown in Table 3. As discussed in earlier section that however there are some best practices, guidelines and methods in security engineering which could be used to develop secure software. Unfortunately these security engineering activities are meant for the traditional waterfall approach. Therefore, there is need to arm agile methods with these security features and it seems perfectly acceptable to use these experienced and tested activities to ensure the secure development of software. But on the other hand, by integrating of some heavy weight activities may result in a process that will not remain agile and will eventually be unacceptable to the project. Thus, in order to deal with this problem, firstly these security activities have to be sorted according to agility. This can be defined by using the two parameters of cost and benefit. Cost can be measured by the level of difficulty of integration with agile process while benefit can be measured by the extent of value of using the activity with agile process. This can easily be decided by a panel of security experts along with the team members. Lower the cost, more it is preferable. Higher the benefit, the more it is

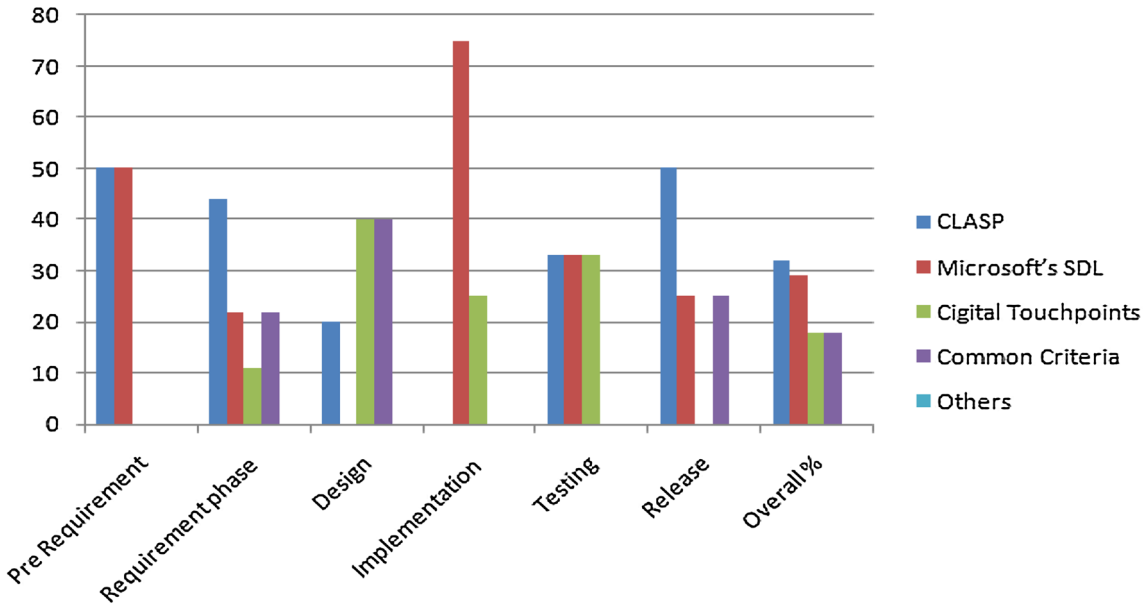

Fig. 5 Security activity coverage in SE processes

Table 3 Agile security process

Product Owner	
Pre requirement	*Requirement phase*
Initial Education (C, MS)	Role Matrix (C, MS)
	Security Requirements (MS, CC, C, CT)
	Agree on Definitions (CC)
	Identify resources and trust boundaries (C)
	Specify Operational Environment (C)
Development Team	
Design	*Implementation*
Secure Design Principles (C)	Coding Rules (MS)
Assumption Documentation (CT)	Security Tools (MS)
Requirements Inspection (CC)	Static Code Analyses (MS, CT)
Risk Analyses (CT, CC)	
Test Team	
Testing	
Dynamic Analysis (MS)	
Security Testing (C)	
Vulnerability and Penetration Testing (CT)	
Release	
Operational Planning and Readiness (C)	
Signing the Code (C)	
Incident Response Planning (MS)	
Repository Improvement (CC)	

preferable. In order to put them in the same scale for further summative assessment, the values are normalized. We cannot select an activity which is beneficial but very costly. Also, we cannot even select the one which has less cost but is less beneficial. So the agility value of these security activities is calculated by summing up the values of both cost and benefit for each activity. Greater value of agility indicates that the activity is more agile and preferable. Although most beneficial security activities are selected through SLR and survey study in previous section, an optional provision could also be provided for selection and

integration of security activities to automate the entire process.

Similarly, the list of agile activities is also obtained according to their agility value, which can be calculated using defined parameters. The agility value of an activity can be defined as the measure of its agile behavior. It represents the degree of activity compatibility with agile methodology. It is calculated using features like Iterative, Adaptive, Customer Interaction, Modularity, Frugality, Flexibility and Less Formalization. Using these characteristics of agile methodologies the project team assigns a value between 1 and 10. This leads to a one-column matrix named AgilityVector. Greater value of agility feature represents a higher level of compatibility with that feature and vice versa.

Any security activity cannot be integrated with all agile activities. Thus, there is a requirement to define a compatibility matrix. For example, vulnerability testing may be integrated with Final Test activity but it may not be compatible with Design Activity. Moreover, the compatibility cannot be defined in terms of only crisp values, i.e., only yes or no. Hence, Fuzzy Compatibility Matrix is used to define the extent of compatibility using membership value, which is named as Fuzzy Compatibility Matrix (FCM). Further, in order to avoid the selection of some heavy security activities, which would endanger the agile activity to remain agile, a Threshold value (TV) is introduced. This value could vary according to the need of the project. Thus, the dynamic integration algorithm is implemented in java which automates the whole process. Figure 6 shows the menu of integration tool, which includes adding and viewing security and agile activities. Figure 7 demonstrates the output of the integration algorithm.

6 Conclusion and future work

This paper provides an integrated framework for secure agile development according to the need of a particular project while keeping in consideration the requirement of every stakeholder including customer, team and project analyst. Initially, a systematic selection process is used to choose a development methodology between agile and plan driven approach. Thereafter, an appropriate agile development method among Extreme Programming (XP), Crystal Clear, Scrum, Lean development, Dynamic Software Development Method (DSDM) and Feature-Driven Development (FDD) is selected for the particular project based upon its specific requirements. Because of the absence of any analytical work on this topic we have used tested and widely used methods like AHP, PROMETHEE, ANN and Fuzzy Logic. By using these empirical methods, we have addressed the reliability issues which are questioned in case of agile development approach. Systematic Literature Review and Survey Study have been used to obtain the authentic industrial feedback followed by the application of statistical tests to identify and select the most suitable and beneficial security activities from well-known security engineering processes like CLASP, Common Criteria, Cigital Touchpoints and Microsoft's SDL. A lightweight method has also been introduced for integrating these security activities identified from SLR and Survey Study, using a dynamic integration algorithm without compromising the agility of the process. The proposed framework for integration of these security activities has been implemented in java to automate the entire process and provide maximum benefit at a low integration cost. Previous works have focused on security issues in bits and

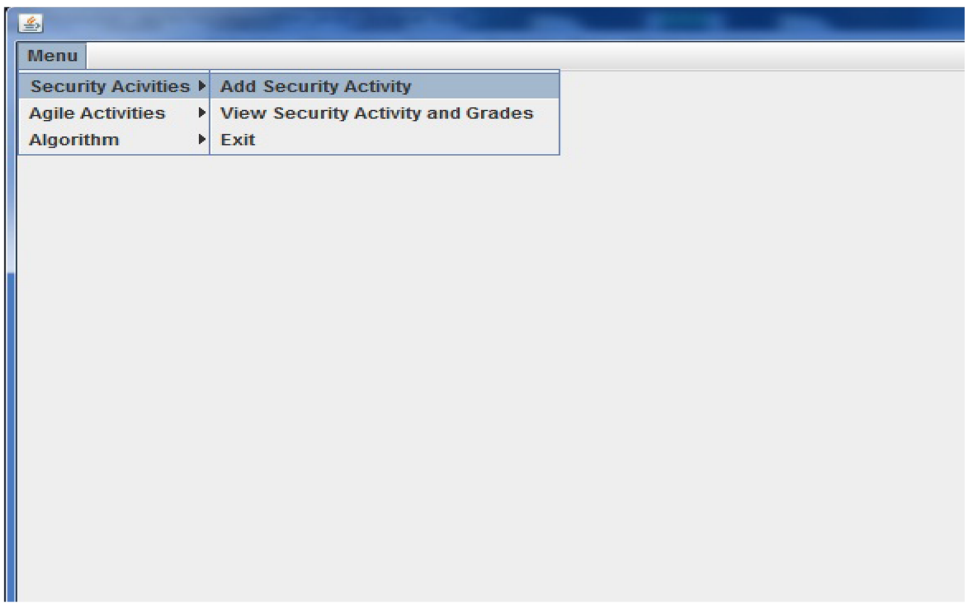

Fig. 6 Menu for adding and running algorithm for integration of security activities

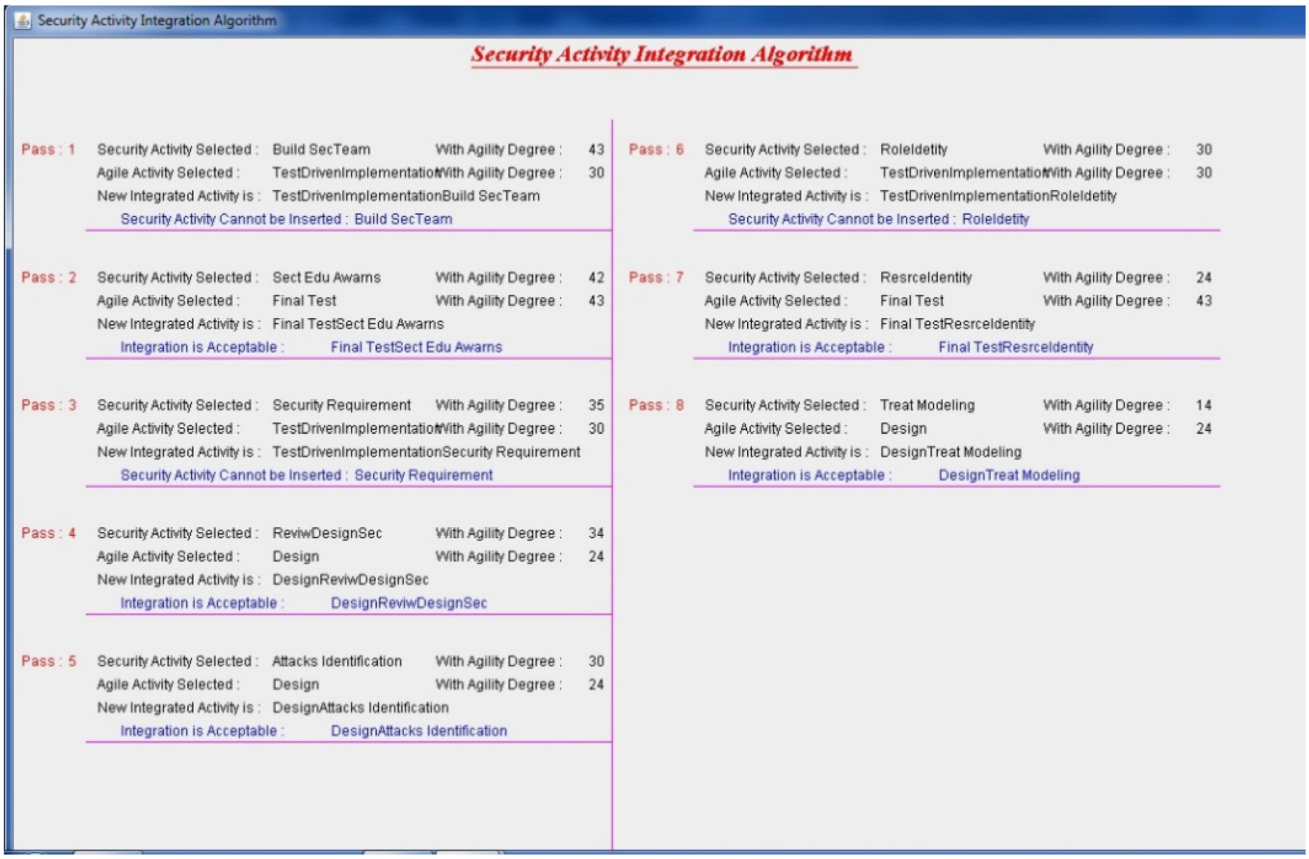

Fig. 7 Algorithm for integration

pieces. There was no comprehensive and integrated approach that covered security throughout the development cycle. However, this framework provides security at the project level as well as process level without even compromising the agility of the whole process.

To develop a secure agile development model there is a need to extract the most beneficial and easily integrable security activities from the well known SE processes, as there are no specific security activities developed especially for agile processes. If these security activities are blindly integrated into the agile process, it would result in making the agile process very heavy, thus not letting it remain agile. So based upon the industry experience and usage, a total of 20 such lightweight and beneficial security activities have been selected from an overall collection of 65 security activities. Most of the security activities are selected from CLASP and Microsoft's SDL. The main reason for this inclination is that CLASP activities do not need a sequential approach unlike others. Thus, this independent approach suits well with agile environment. Microsoft's SDL, on the other hand, is a tight process for constructing a software and moreover it is also modified to suit to agile environment by categorizing security activities based upon frequency of use.

For future work, there is a need to develop agile specific security activities either by modifying the existing security activities by making them light weight with similar benefits or by developing specialized lightweight security activities for agile model specifically. Furthermore, the security activities suggested in this paper need to be tested and evaluated in real industry environment. Apart from these security activities, many more security features are suggested in open-ended questions of the survey which need detailed analysis to come to any conclusion. There is also a scope to use artificial intelligence/soft computing for automatically selecting appropriate security activities according to the need of a particular project.

References

1. Beck K et al (2001) Manifesto for agile software development. Accessed 10 June 2019
2. Beznosov K, Kruchten P (2004) Towards agile security assurance. In: Proceedings of the 2004 workshop on new security paradigms, pp 47–54. ACM 1-59593-076-0/05/05
3. Bartsch S (2011) Practitioners' perspectives on security in agile development. In: Sixth international conference on availability,

reliability and security (ARES), pp 479–484. https://doi.org/10.1109/ares.2011.82
4. Wayrynen J, Boden M, Bostrom G (2004) Security engineering and eXtreme Programming: an impossible marriage? Extreme programming and agile methods, Calgary, Canada, August 15–18. https://doi.org/10.1007/978-3-540-27777-4_12
5. Bostrom G, Wayrynen J, Boden M, Beznosov K, Kruchten P (2006) Extending XP practices to support security requirements engineering. In: ACM SESS 06, Shanghai, China, May 20–21, pp 11–17. https://doi.org/10.1145/1137627.1137631
6. Beznosov K, Kruchten P (2004) Towards agile security assurance. In: Proceedings of the workshop on new security paradigms, September
7. Siponen M, Baskerville R, Kuivalainen T (2005) Integrating security into agile development methods. In: Proceedings of the 38th Hawaii international conference on system science. https://doi.org/10.1109/hicss.2005.329
8. Keramati H, Hassan S, Hosseinabadi M (2008) Integrating software development security activities with agile methodologies. In: IEEE/ACS international conference on computer systems and applications, AICCSA, pp 749–754
9. Baca D, Carlsson B (2011) Agile development with security engineering activities. In: Proceeding of the 2nd workshop on software engineering for sensor network applications, pp 149–158. https://doi.org/10.1145/1987875.1987900
10. Baca D (2012) Developing secure software in an agile process. Computer Science Department, Blekinge Institute of Technology Sweden, Karlskrona, pp 129–149
11. Carlsson B, Ayalew T, Kidane T (2013) Identification and evaluation of security activities in agile projects. In: 18th Nordic conference. https://doi.org/10.1007/978-3-642-41488-6_10
12. Bartsch S (2011) Practitioners' perspectives on security in agile development. In: Sixth international conference on availability, reliability and security (ARES), pp 479–484. https://doi.org/10.1109/ares.2011.82
13. Shackleford D (2011) Integrating security into development, no pain required. A SANS whitepaper
14. Savola R, Frühwirth C, Pietikäinen A (2012) Risk-driven security metrics in agile software development—an industrial pilot study. J Univers Comput Sci 18:1679–1702. https://doi.org/10.3217/jucs-018-12-1679
15. Wolff S (2012) Scrum goes formal: agile methods for safety-critical systems. In: IEEE formal methods in software engineering: rigorous and agile approaches (FormSERA), pp 23–29
16. GAO (2012) Effective practices and federal challenges in applying agile methods. Report to the Subcommittee on Federal Financial Management, Government Information, Federal Services, and International Security, Committee on Homeland Security and Governmental Affairs United States Senate. www.gao.gov/assets/600/593091.pdf. Accessed May 2017
17. Munetoh S, Yoshioka N (2013) RAILROADMAP: an agile security testing framework for web-application development. In: IEEE sixth international conference on software testing, verification and validation (ICST), pp 491–492. https://doi.org/10.1109/icst.2013.80
18. Rindell K, Hyrynsalmi S, Leppänen V (2017) Busting a myth: review of agile security engineering methods. ACM. https://doi.org/10.1145/3098954.3103170
19. Harrison S et al (2016) A security evaluation framework for U.K. E-government services agile software development. Int J Netw Secur Appl (IJNSA) 8(2):51–69. https://doi.org/10.5121/ijnsa.2016.8204
20. Rindell K, Hyrynsalmi S, Leppänen V (2019) Challenges in agile security engineering: a case study. In: Felderer M, Scandariato R (eds) Exploring security in software architecture and design. IGI Global, Hershey, PA, pp 287–312. https://doi.org/10.4018/978-1-5225-6313-6.ch012
21. Howard M, Lipner S (2006) The security development lifecycle. Microsoft Press, Redmond. https://doi.org/10.1016/S0925-7535(03)00047
22. Sullivan B (2008) Streamline security practices for agile development. MSDN Mag. https://doi.org/10.4018/jsse.2010070105
23. Keblawi F, Sullivan D (2006) Applying the common criteria in systems engineering. IEEE Secur Priv 4(2):50–55. https://doi.org/10.1109/msp.2006.35
24. McGraw G (2006) Software security: building security in. Addison-Wesley, Boston. https://doi.org/10.1109/msecp.2004.1281254
25. 'Category:CLASPActivity-OWASP'. https://www.owasp.org/index.php/CLASP_Concepts. Accessed 11 Feb 2019
26. Kitchenham B, Charters S (2007) Guidelines for performing systematic literature reviews in software engineering. Keele University, UK EBSE-2007-1. https://www.elsevier.com/__data/promis_misc/525444systematicreviewsguide.pdf
27. Wohlin C (2000) Experimentation in software engineering: an introduction, vol 6. Springer, Berlin
28. Fay MP, Proschan MA (2010) Wilcoxon–Mann–Whitney or t-test? On assumptions for hypothesis tests and multiple interpretations of decision rules. Stat Surv 4:1–39. https://doi.org/10.1214/09-SS051
29. Kruskal W (1952) Use of ranks in one-criterion variance analysis. J Am Stat Assoc 47(260):583–621. https://doi.org/10.2307/2280779
30. Dalgaard P (2008) Introductory statistics with R. Springer, Berlin, pp 99–100. https://doi.org/10.1007/978-0-387-75936-4