A Comprehensive Approach for Agile Development Method Selection and Security Enhancement

Ruchi Sharma
Department of Computer Science and Engineering
Universal Group of Institutions, Lalru, Punjab, India

Amit Sharma
Department of Computer Science and Engineering
The ICFAI University, Solan, Himachal Pradesh, India

Dr. R.K Bawa
Department of Computer Science
Punjabi University, Patiala, Punjab, India

Abstract- Today's software development business requires fast software delivery from the development team. For this reason, organizations are transforming their conventional development approaches to agile approaches .Rapid change of customer requirements has also propelled the emergence of an agile approach due to such merits as fast release and simplified documents. An agile approach aims at maximizing productivity and effectiveness. However, most agile-developed software today needs to satisfy baseline security requirements, so that we need to focus on how to achieve this level for typical agile projects. This paper provides a roadmap for making decision about whether to use Agile development or not for a given project and if selected , it further suggests which Agile Development method is better suited for among popular methods –Scrum and Extreme Programming, Crystal Clear, Lean development, the Dynamic Software Development Method (DSDM) and Feature-driven Development (FDD). It also addresses the major concern of security requirements on projects using an agile approach. We proposed a lightweight method to enhance the security features by integrating security activities from Security engineering processes without compromising the agility in the agile approach.

Keywords – Agile Development, Scrum, XP, DSDM, FDD, Crystal Clear, Agile Security.

I. INTRODUCTION

A. Agile development

Agile development follows an opposing approach from the formalization and control in plan-driven development. Instead, agile development formalizes processes only where necessary and emphasizes informal and intensive interaction to craft systems with high business-value. Agile development refers to a set of values shared by related development methods, such as SCRUM and XP. The Agile Manifesto [1] lists the overarching values in abstract terms:

- Individuals and interactions over processes and tools,
- Working software over comprehensive documentation,
- Customer collaboration over contract negotiation,
- Responding to change over following a plan.

In agile development, responsiveness is emphasized over the reliability of standardized development processes. The process is more likened to learning than to the application of prior knowledge. Agile methods can be described as "generative" instead of "adaptive" learning, applying double-loop learning. Also, the "command and control" approach of plan-driven models is exchanged for a more democratic model to profit from the tacit knowledge of the individuals in the team. More specifically than the Agile Manifesto, the principles of agile development are to

"deliver something useful," "rely on people," "encourage collaboration," "technical excellence," "do the simplest thing possible," and "be adaptable."

Popular methods in the agile ecosystem have a broad range of characteristics. More liberal methods, such as Crystal Clear and Scrum, formalize the development process to a lesser degree than more heavy-weight methods, such as Feature-Driven Development. Two distinct philosophies can be contrasted here: The liberal methods with an optimistic view that development teams are able to tailor the process to fit their particular development environment in the most efficient way. Conversely, the pessimists rather specify the process in detail to prevent failures from adaptation or problems in large or high-reliability projects.

The rest of the paper is organized as follows. Proposed embedding and extraction algorithms are explained in section II. Experimental results are presented in section III. Concluding remarks are given in section IV.

B. An overview of Agile Development Methods

Several agile development methods have been formulated since the late 1980s. To analyze the breadth in agile development, we have taken popular methods – Scrum and Extreme Programming, and, less widely-spread, Crystal Clear, Lean development, the Dynamic Software Development Method (DSDM) and Feature-driven Development (FDD). The elaboration would enrich the reasoning on agile development and its relation to security in software development.

In the following, we discuss each of the studied development models briefly, including their values, principles, and practices.

Scrum

Scrum implements "empirical process control" through feedback loops between development and process adaption [12] [13]. Scrum emphasizes the roles and responsibilities in the process, and high-level practices for requirements management, and thus focuses on the project management perspective. The product owner (customer) is in charge of setting priorities, developers in the team create the product, and a Scrum master is responsible for an effective Scrum process. Central to the Scrum process are the iterating development phases of 30 days, called "sprint", each resulting in a deliverable product. The sprint is planned in a planning meeting and the product owner places work items into the sprint backlog, a list of features that need to be completed by the team within the sprint.

Extreme Programming

Extreme Programming (XP) is a bottom-up development method in that specific programming practices play a central role in defining the development process [4]. The XP development process is similarly structured as the Scrum process, but the iterations are shorter with weekly cycles. In a planning game, the customer chooses the work, which is described in stories, for the subsequent iteration. In addition, XP proposes quarterly cycles for long-term, "big- picture" planning. In contrast to Scrum, XP implements more specific practices concerned with the project environment and the development, such as informative workspaces and pair programming.

Crystal Clear

Crystal Clear is part of the Crystal family of agile development methods. The Crystal methods should provide "barely sufficient" process guidance and, thus, offer different method weights for different project types [12]. As a side effect, the Crystal family is an example of how different the actual implementations of agile processes are for specific development contexts in terms of team size and product criticality. Crystal Clear is the most liberal with a small number of characteristics and leaves the team room for individual practices. When compared to XP, for instance, with its strict approach to development practices, Crystal Clear should be easier to implement and can even serve as a backup if the introduction of XP fails. An interesting aspect that is explicitly formulated as a practice in Crystal is "personal safety": Participants need to trust each other and no-one should be hindered by fear of personal consequences of mistakes.

Lean Development

Lean Development is the adaption of lean-production principles as implemented by Toyota in manufacturing to software development [14] and has a pronounced management perspective (top-down). Accordingly, the first of

four process phases, a start-up phase, is used to identify and reduce project risks. In a "steady state", development cycles (shorter than 60 days) integrate incremental analysis, design, tests, and integration. The last two phases, transition and renewal, are enacted to continue development after the customer accepted a product's delivery. In Lean Development, everything should be changeable, resulting in one principle to "build the application to be more tolerant" to changes [12]. This principle is in contrast to the principle of choosing the simplest approach in XP. Similarly, a Lean-Development principle is "Domain, not point solutions", thus explicitly not targeting a very specific and simple solution, but broader ones that are often more complex.

Dynamic Software Development Method

The Dynamic Software Development Method (DSDM) originates in the Rapid Application Development (RAD) community and formalizes RAD practices [12] [15]. DSDM defines three interrelated development phases that are iteratively applied. In the functional modeling phase, requirements are defined by way of functional prototypes. The design/build phase produces design prototypes, which are then turned into the final product and deployed in the implementation phase. With 15 defined types of process deliveries and 11 roles, DSDM has a considerable process complexity. However, most deliveries may occur in form of prototypes, thus corresponding to the agile principle of early running code.

Feature-driven Development

Feature-driven development (FDD) combines model-driven development with agile practices [16]. The FDD method is precisely and formally defined. The method is divided into 5 processes, of which three (developing an overall model, building a feature list, and planning features) are run once and two (designing a feature and building a feature) are run iteratively for every feature. The up-front processes are the main differences to the simpler agile methods like Scrum. While these processes should only take up a limited amount of project resources and are ongoing in case of later changes, the focus is on having a more precise plan and less need for rework and refactoring. In this respect, FDD is more plan-driven then the other agile methods [12]. On the other hand, FDD is reported to be particularly well suited for scaling agile development to larger projects. Development roles are more differentiated, for example, including a chief-programmer role, responsible for assigning work items to individual programmers.

C. Security in agile development

In literature, there is a discussion on whether agile development methods and the underlying principles are appropriate to develop secure software. One reason is that the agile development proponents did explicitly not target high-risk software development. Kent Beck rather states in his XP book that XP in itself is not suitable for high-reliability requirements. However, security is not only relevant for high-reliability projects, but affects most software that is being developed.

The main issue with agile development concerning security is that the team-emphasizing, dynamic and tacit-knowledge-driven methods conflict with the assurance activities as demanded by traditional secure software development methods. However, there are indications that agile development improves quality. Moreover, plan-driven development also poses challenges to secure software development that might be less critical in agile development. Early planning of security requirements may conflict with the changing requirements in practice, which agile development is better prepared for. Also, to address the challenges to security in agile development, various enhancements have been proposed to agile methods.

II. PROPOSED ROADMAP

A. Whether to Choose Agile Development or not?

It's always a complicated task for a project analyst to analyze the development of a software project; especially when it comes to evaluation of the development methodology and estimation of the project's success considering every related aspect. The goal of this section is to simplify this task for the analyst and clarify the important aspects of a project which develop with agile methodologies.

Selection Procedure

In order to calculate a value that can help the analyst choose the best methodology to manage a project, some questions should be answered. Those answers will be the data the analyst needs to put into the formulas [18]. These questions derived from the comparison of Agile methodologies with other methodologies and the Agile manifesto and the features of Agile methodologies.
An answer to each question is a value between 1 to 7 (Likert Scale). The value shows the importance of each factor asked in the question. (1 indicates the least importance, 7 indicates the most.) The questions have been asked based on different side of each project for example customer's point of view or project manager's point of view and etc.
The major questions concerns:

Customer

1) How much does the customer know about his/her requirements and his/her project?
2) How fast the customer wants the project to be delivered?
3) How much does the customer support and collaborates with developers during the whole development operations?

Project development process

1) How complicated the project is?
2) How important is the risk management for this project?
3) How dynamic the project is?
4) How important the documenting process is?
5) How are the different modules of the project coupled to each other?

Production Team

1) How many members are in the team?
2) How many experienced members in the team are?
3) How much facility the team has for communication among developers?
4) How important security of information in the team is?

Product

1) How clear major problems of the project are?
2) How familiar this kind of project is to the team members?
3) How important is the quality of the product?

If analyst has more factors in mind regarding the project, he/she simply can add them as questions. In contrast, analyst can remove any needless questions in his/her point of view. Normally, all questions have the same importance factor and its value is 1. However, if some factors will be more important to the analyst he/she can change the importance factor manually by adding weights. Further simple mathematical techniques or complex one's like Factor analysis can be used for precise estimation of Agility.

B. Selection of most Appropriate Agile Development Method

In essence, each agile development method is a collection of practices, which are supported by values and principles. Based upon the previous calculations and characteristics of different agile methods, which are discussed next, the most appropriate agile development method is selected.

Lenses for the Selection of appropriate agile method

The agile development methods evolved independently, prior to the publication of the Agile Manifesto, and differ in coverage and the approach to their specification, regarding principles, practices, roles, and concreteness [17]. Due to this heterogeneousness agile methods were often described alongside each other [12], without a systematic comparison.

In practice, the decision of which agile method to apply cannot be taken based on a one-dimensional score, but needs to compare the given context to the characteristics of the methods. To further systematically analyze the agile methods, we examine four major dimensions of agility for a high-level comparison. The four dimensions –

process formalization, averseness to change, process overhead, project size/reliability – are strongly interrelated with the agile values as defined in the Agile Manifesto [1] [19]. We discuss the rationale for the relative and qualitative findings on the methods for each of the dimensions in the following and present the results schematically in Figure1.

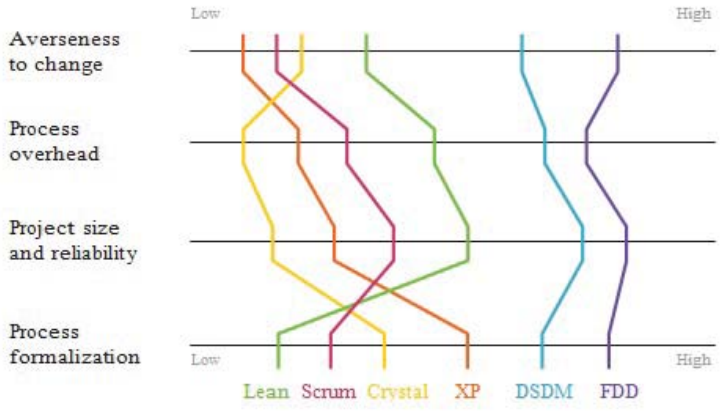


Figure1. Characteristics of the compared agile methods

Process formalization

Depending on the method's background, each has a distinct approach on how formal and concrete the definitions of the processes and practices are in contrast to a more liberal approach of teams adapting process and practices as needed in specific environments. High process formalization often increases the method's averseness to change (fixations from formalized procedures) and also entails higher process overhead (from prescribed activities), but may increase the process reliability (activities are prescribed) and its adequacy for large projects (more formalized interaction). Similarly, process formalization conflicts with agile values, for example, with the focus on individuals and (implicit) interactions, and on working software.
Figure 2 shows the extent of process formalization in the given agile development methods.

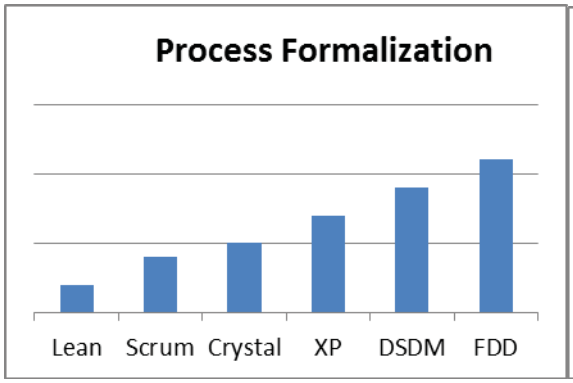


Figure2. Process Formalization

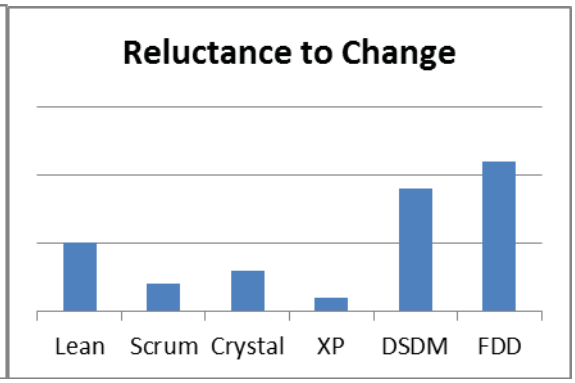


Figure3. Reluctance to change

Reluctance to change

Depending on the process, methods may be more or less open to changes to an original plan and to the process itself. While larger teams may work more efficiently with less volatile requirements, since this improves parallel development, changes to requirements are likely to occur during the development process if the resulting product should deliver added value. Figure 3 shows the extent of reluctance to change in given agile development methods.

Resource/Process Overhead

A high process formalization may lead to more overhead when implementing and operating a method. Process complexity is dominated by the number and depth of hierarchy of different processes and phases that are enacted as part of the development method. Another aspect is the number of required practices and of roles that are defined as part of the method. Figure shows the extent of process overhead in given agile development methods.

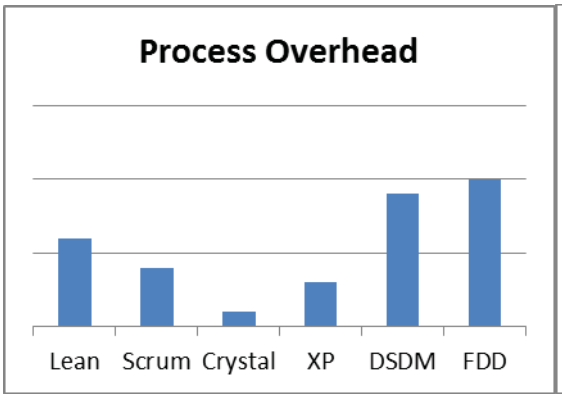


Figure 4. Process Overhead

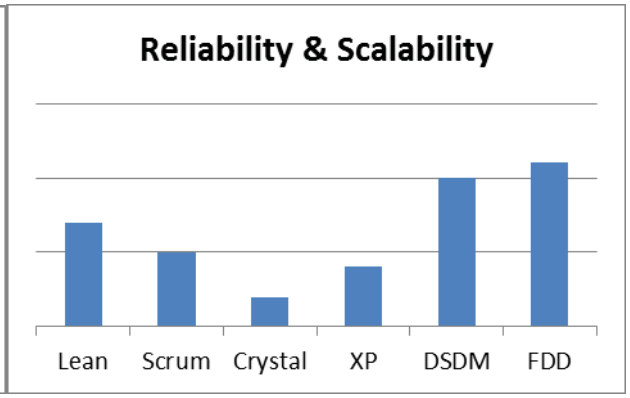


Figure5. Reliability & Scalability

Reliability and Scalability

Larger projects typically require more defined communication processes since "osmotic" communication loses its efficacy [12]. Similarly, high reliability often requires more defined processes to implement formal process requirements and traceability [14]. Some of the analyzed agile methods are explicitly tailored for small teams and medium reliability requirements. Figure 5 shows the extent of reliability & scalability in given agile development methods.

Selection

The high-level comparison of the agile methods indicates that the methods are compartmentalized into two groups. On the left-hand side, Figure1 shows the more liberal methods Crystal Clear, XP, Scrum, and Lean development. On the right, the more heavy-weight agile methods FDD and DSDM represent the other end of the spectrum of agile methods. These two methods have more complex processes but are claimed to be better suited for larger, high reliability projects.

On the basis of the calculations made by using the questions asked initially, a most appropriate agile development method will be chosen, which will be best suited with the actual requirements of the project.

C. Integration of Security in Agile Development Method

The growing trend towards the use of agile techniques for building software and the increase in security breaches over the past few years means that it is essential to integrate existing high-profile Security engineering (SE) processes with agile processes. Moreover, as there are no SE processes developed specifically for an agile setting, organizations have used existing waterfall SE processes in their agile processes. However, the reliability of the SE processes commonly used in the waterfall model has not yet been evaluated in an agile development setting. Accordingly, existing security activities within water-fall SE processes used in current agile processes are investigated. Four high-profile waterfall SE processes (CLASP, Microsoft SDL, Cigital Touchpoints and Common Criteria) are investigated [19]. Based on these SE processes, a total of 41 security activities are obtained which are used for further investigation.

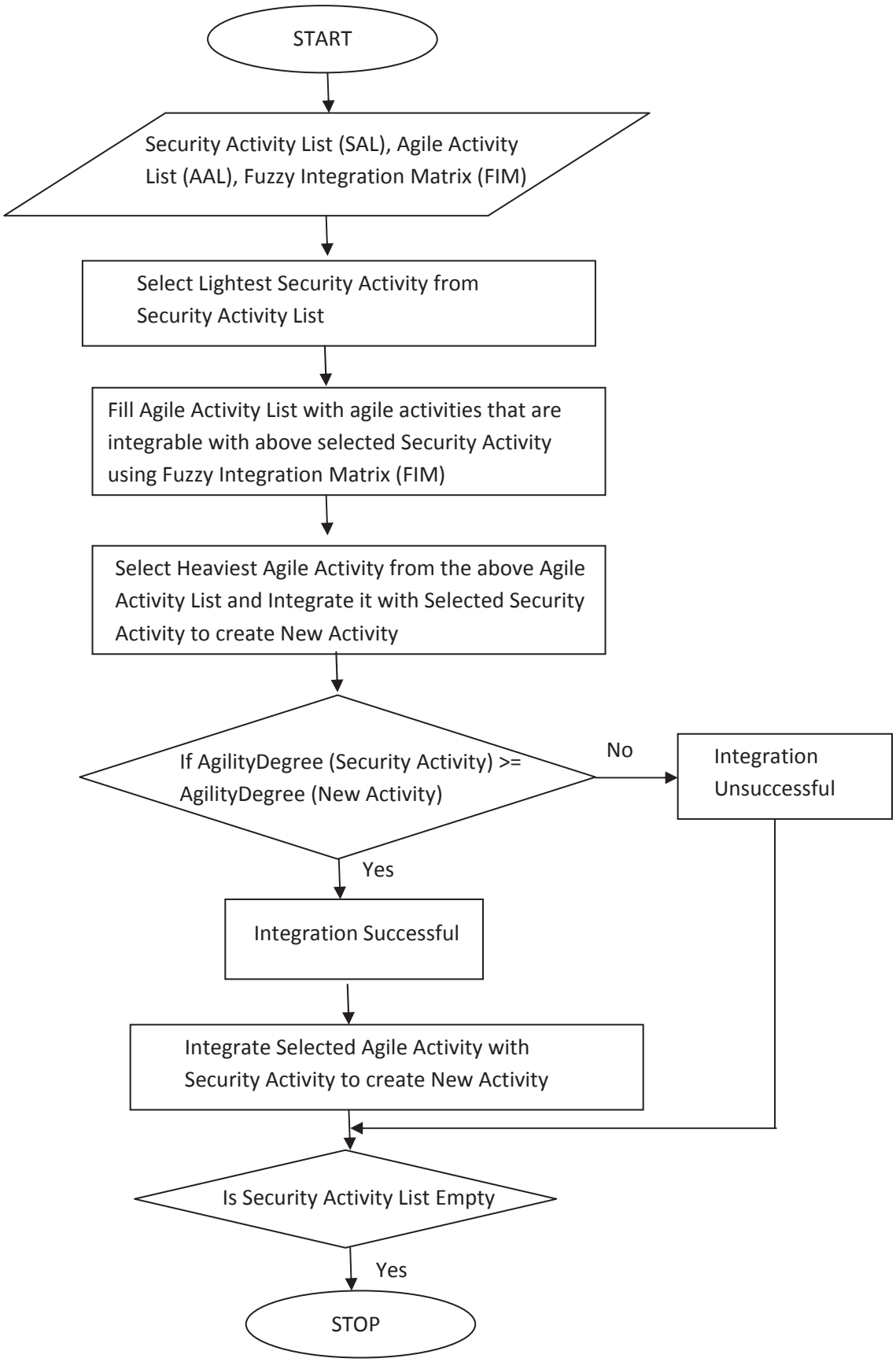
Security Activities		
Requirement		*Implementation*
Security Metrics (CLASP)		Static Code Analyses (SDL, CT)
Initial Education (CLASP, SDL)		Security Tools (SDL)
Security Requirements (CLASP, SDL,		Coding Rules (SDL)
Abuse Cases (CLASP, CT)		Pair Programming (O)
Agree on Definitions (CC)		
Role Matrix (CLASP, SDL)		*Testing*
Design Requirements (SDL)		Vulnerability & Penetration Testing (CT)
Identify Trust Boundary (CLASP)		Red Team Testing (CT)
Identify Global Security Policy (CLASP)		Risk Based Testing (CT)
Specify Operational Environment		Dynamic Analysis (SDL)
Identify Attack Surface (CLASP)		Fuzzy Testing (SDL)
		Code Review (CLASP, SDL)
Design		Security Testing (CLASP)
Risk Analyses (CT, CC)		
Assumption Documentation (CT)		*Release*
Critical Assets (CC)		External Review (CT)
UMLSec (CC)		Repository Improvement (CC)
Quality Gates (SDL)		Incident Response Planning (SDL
Cost Analysis (SDL)		Signing the Code (CLASP)
Attack Surface Reduction (SDL)		Operational Planning and Readiness
Security Architecture (CLASP)		Final Security Review (SDL)
Secure Design Principles (CLASP)		
Security Activities		
Countermeasure Graphs (O)		
Requirements Inspection (CC)		
Threat Modeling (CLASP, SDL)		

Integration of Security Activities with Agile Development Methods

As mentioned in earlier section, there are some guidelines, best practices, methods and other materials in Security engineering (SE) that can be used by project's team to produce secure software products [18]. To arm agile methods with security features, it is acceptable to use these experienced and proposed activities for secure software development. On the other hand, integrating some heavy weight activities with agile processes may lead to a process that cannot be named agile and possibly will be unacceptable for project's team. In order to restrain reduction of agility nature, a proper method has to be used. First security activities are extracted from Security engineering (SE) processes, and then agility degree of activities is defined to measure their nimbleness. Integration issues of agile and security activities are handled and a flowchart to integrate security activities with organization's agile process is introduced.

Integration Method

To integrate security activities selected from Security engineering (SE) processes the following steps shown in the figure 5 have to be followed. This flowchart provides a method through which security activities can be integrated with agile activities without compromising the agility of the process.


START
Security Activity List (SAL), Agile Activity List (AAL), Fuzzy Integration Matrix (FIM)
Select Lightest Security Activity from Security Activity List
Fill Agile Activity List with agile activities that are integrable with above selected Security Activity using Fuzzy Integration Matrix (FIM)
Select Heaviest Agile Activity from the above Agile Activity List and Integrate it with Selected Security Activity to create New Activity
If AgilityDegree (Security Activity) >= AgilityDegree (New Activity)
No
Integration Unsuccessful
Yes
Integration Successful
Integrate Selected Agile Activity with Security Activity to create New Activity
Is Security Activity List Empty
Yes
STOP

IV.CONCLUSION

This work provides a comprehensive approach to select agile development and also the best agile method suited according to the project requirement. We hope that this preliminary roadmap serves as a starting point for creating a comprehensive agile selection approach and enables the generation of more fruitful research results from the field. Using introduced method in this paper, security activities can be integrated to agile methodologies to enhance security of software product without compromising the overall agility of the project.

In addition, since the selected security activities are originally developed for waterfall development approach, some of the security activities might need modification in order to adapt with an agile process. We have not investigated new or pure agile SE-processes (but a selection of existing/modified security activities as a base for the agile development). Therefore, the directions for future work primarily include evaluating these security activities that are selected as compatible and beneficial to an agile model in a real agile industry setting. These steps will add value to the findings and gain acceptance in the real agile industry.

REFERENCES

[1] Beck K., et al. Manifesto for Agile Software Development, February 2001.
[2] Blitz, D.C., and van Vliet, P., "Global Tactical Cross-Asset Allocation", Journal of Portfolio Management, Vol. 35, No. 1, pp. 23-38, 2008.
[3] Boehm, B. "A Spiral Model of Software Development and Enhancement", Journal: Computer, Vol. 21, No. 5, pp. 61-72, 1988.
[4] Beck, Kent, Andres, Cynthia, Extreme Programiming: Embrace Change (2nd ed.). Addison Wesley Professional, Boston, 2004.
[5] N. Ramasubbu and R. K. Balan, "Globally distributed software development project performance: an empirical analysis", in Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering - ESEC-FSE 07, 2007, p. 125.
[6] Concas, G., Francesco, M., Marchesi, M., Quaresima, R., and Pinna, S. "An agile development process and its assessment using quantitative object-oriented metrics" Agile Processes in Software Engineering and Extreme Programming, 83- 93, 2008.
[7] Ramasubbu, N. & Balan, R.K., "The impact of process choice in high maturity environments: An empirical analysis" In the proceedings of the 31st IEEE International Conferences on Software Engineering (ICSE 2009), Vancouver, British Columbia, Canada. pp. 529–539, May 16–24, 2009.
[8] Begel , A. , and Nagappan , N. "Usage and perceptions of agile software development in an industrial context: An exploratory study" In ESEM '07: First International Symposium on Empirical Software Engineering and Measurement, 255–264. Washington, DC: IEEE, 2007 .
[9] Parsons, D., H. Ryu and R. Lal, " The Impact of Methods and Techniques on Outcomes from Agile Software Development Projects" IFIP International Federation for Information Processing, Springer Boston: 235- 249, 2007.
[10] Fitzgerald, B., Harnett, G., and Conboy, K. "Customizing Agile Methods to SoftwarePractices", European Journal of Information Systems, Vol 15, No. 2, 2006.
[11] Kniberg, H., and Skarin, M. "Kanban and Scrum - making the most of both," C4Media Inc., USA, 2010.
[12] Highsmith, J. "Agile software development ecosystems", Boston, M.A., Pearson Education, 2002.
[13] Ken Schwaber and Mike Beedle, Agile Software Development with Scrum (Prentice Hall, 2001).
[14] Poppendieck, M and Poppendieck T "Lean Software Development An Agile Toolkit" Boston: Addison Wesley, 2003.
[15] J. Stapleton, DSDM: The Method in Practice, Second ed: Addison Wesley Longman, 2003.
[16] S. R. Palmer and J. M. Felsing, "A Practical Guide to Feature-Driven Development" Upper Saddle River, NJ: Prentice Hall PTR, 2002.
[17] Abrahamsson, P., Warsta, J., Siponen, M., and Ronkainen, J., "New directions in agile methods: Comparative analysis". In Proceedings of the 25th International Conference on Software Engineering, 244–254, 2003.
[18] Keramati, H., Hassan, S., Hosseinabadi, M. " Integrating software development security activities with agile methodologies ", pg. 749-754, IEEE/ACS International Conference on Computer Systems and Applications, AICCSA 2008, March 31-April 4 2008.
[19] Baca D., Carlsson B., "Agile Development with Security Engineering Activities ", ACM International Conference on Software Engineering ICSE '11, May 21–28, 2011.
[20] Nasr-Azadani B., MohammadDoost R., "Estimation of Agile Functionality in Software Development ",Proceedings of the International MultiConference of Engineers and Computer Scientists 2008 Vol I IMECS 2008, 19-21 March, 2008.