ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 4, pp: (30-35), Month: October - December 2015, Available at: www.researchpublish.com

Linking Security Activities with Agile Methodologies during Software Development

[1]Bhawna Arora, [2]Dr. PK Suri

Chairman & Professor, HCTM Technical Campus, Kaithal, Haryana, India

Abstract: **Quick change in the necessities and increment in the advancement of software prompts the utilization of agile systems. Lessening in time prompts vast benefit to the organization. On account of a few vulnerabilities in software products and high measure of harm brought on by them, software developers are authorized to create more secure products. Some of the methodologies like SCRUM, Extreme programming (XP), Feature driven Development (FDD), Adaptive software development (ASD) etc. are being used by the developers. This paper concentrates on agile methods keeping in mind the end goal to equip them with security exercises. In this paper different agile methodologies are reviewed moreover security activities are integrated with agile methodologies, Dependability of the system get increase by adding security feature to it.**

Keywords: **Agility, Security, Agile Methodologies, SDLC.**

I. INTRODUCTION

Software goes from many different stages during its development. Software development process consist of Analysis, design, implementation, test, deploy, production and also product's death iterations should be armed with security activities to reach more secure. Software is the combination of methods, practices, activities that are that are utilized to obtain and insist software and its related products. Using system in most critical environment may lead to huge amount of risks and damages if security is not considered. From last numerous decades, this has been an issue that how it can be created speedier, less expensive and in better way. Many tools, techniques and methodologies are implemented to increase security in software systems. Many recommendations are gathered and some trials are done, this prompts expansion of security agile software advancement. There exist a wide range of strategies for agile yet because of lack of learning and skill, these routines are hard to complete. Subsequently, it gets to be hard to know its advantages and disadvantages. As of late, a few impacts of agile methodologies are being seen during software development process.

This paper will focus on methods to add security activities to agile software methodologies with different parameters and in addition, paper reviews different methodologies in agile development process.

II. RELATED WORKS

Efforts are being made to develop secure software products because of great losses and damages to organizations. All these practices led to design and development of risk management tools and lots of different methods to analyze security.

AEGIS: - is a methodology to develop secure systems and which was introduced by “Flechais”. It has been designed based on asset modeling, security requirements identification, risk analysis, and usability context. It starts after design phase of software development life cycle (SDLC) and tries to reduce vulnerabilities by means of risk analysis and mitigation, but it does not represent any clear procedure for the development team so as to perform these activities. It is not a full-lifecycle software development process.

Agile: - The word agile means ability to move with quick easy grace. According to Boehm, agile methods are “*an outgrowth of rapid prototyping and rapid development experience as well as the resurgence of a philosophy that* *programming is a craft rather than an industrial process*". **Alistair Cockburn defines agile as** "*agile implies being effective and manoeuvrable. An agile process is both light and sufficient. The lightness is a mean of staying manoeuvrable. The sufficiency is a matter of staying in the game*". **Who is one of the founders of the agile movement in software development.**

Software Security: - to design and develop secure systems. It is very important to include security into every phase of the Software Development life cycle. It is a system-wide issue that involves both building in security mechanisms and designing the system to be robust.

III. AGILE METHODOLOGIES

Apart from traditional and object oriented processes, new method comes with some different and exciting features. These are agile methodologies with the motive to develop with high speed and full satisfaction of client. Some of the examples of agile methods are: - . XP, Scrum, FDD and DSDM

Features of agile methods:

1) Proper communication between the team members.
2) Small team size.
3) Capture requirements at a high level and
4) Develop small builds following incremental approach.
5) Frequent delivery of products.
6) Dynamic client association

DSDM: Is the prime agile development method. DSDM was around before the term 'agile' was discovered, yet is completely in view of the considerable number of standards we've come to know as agile.

SCRUM: Which focuses especially on the most proficient method to oversee undertakings inside of a group based improvement environment. Scrum is the most prevalent and broadly embraced technique. it is moderately easy to implement and addresses a considerable lot of the management issues that have tormented IT development teams for quite a long time.

XP (Extreme Programming): Is a more radical method, concentrating all the more on the product designing process and tending to the analyze, development and test stages with novel methodologies that have a significant effect to the quality of the end product.

IV. SECURITY TO AGILE METHODOLOGIES

Many different techniques are there to produce secure software products.

From the preexisting processes,

1) Security related activities are filtered
2) Agility degree is computed
3) Agile and Security activities are linked
4) Activity-Process linking Algorithm
5) Agility Reduction Tolerance

Filter Security Activities:

With a study on works done in this field, we accomplish a rich rundown of exercises that can be performed during a product development procedure to reach more secure products. In our study on a few scholastic and mechanical assets, many exercises and sub-exercises going from initiation to production and item's demise are gained. We name these exercises as security exercises and utilize this rundown as a premise for next strides in our system

Ordering security exercises will help us to see every action and its execution connection better, and to see related exercises in the same suitable classification too.

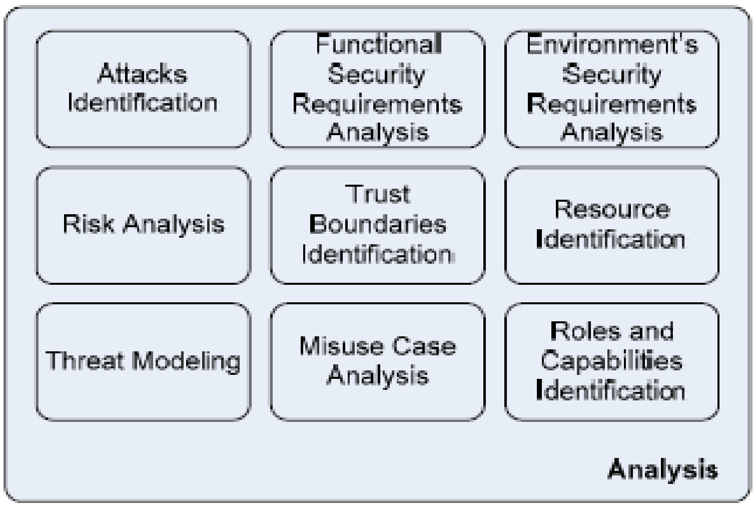

Attacks Identification
Functional Security Requirements Analysis
Environment's Security Requirements Analysis
Risk Analysis
Trust Boundaries Identification
Resource Identification
Threat Modeling
Misuse Case Analysis
Roles and Capabilities Identification
Analysis

Figure 1 – Extracted Security Activities for Analysis Phase of Generic SDLC

Computing Agility Degree:

With respect to agile manifesto, agile strategies have regular components that distinct them from other sorts of software development procedures. Adding security exercises to agile techniques may prompt a substantial procedure that won't be executed as expected.

By characterizing agility degree for every action, we can quantify its agile behavior. Agility degree represents level of activity's compatibility with agile methodologies and is calculated on the basis of agility features such as simplicity, rapidity, and people orientation. Agile degree is aggregate of values in ADVect of an action. This straightforward meaning of agility degree gives a decent sense about activity's admiration to the agile manifesto and agile methodologies.

Agility degree vector (ADVect) of an activity is not extraordinary for all development teams and associations.

$$\text{ADVect} = \begin{vmatrix} \text{Simplicity} \\ \text{Customer Interaction} \\ \text{Free of Modeling and Documentation} \\ \text{Change Tolerate} \\ \text{Speed of Execution} \\ \text{Informality} \\ \text{People Orientation} \\ \text{Iterative} \\ \text{Flexibility} \end{vmatrix}$$

For example, "Threat Modeling" is an activity in our security activities list a convoluted, semi-formal and substantial weight action, which is in light of displaying and requires top to bottom security learning. Detailed analysis of this activity results in estimation for ADVect of it as [1 1 0 3 2 1 1 4 1]. It is an activity that is not as simple as agile methods does not have incredible communications with the client, obliges complete demonstrating and documentation endeavors, with medium-level resistance on changes, low speed, formal and strategy based rather than people-oriented, Table 1 shows results of analyzing some security activities, their ADVect and agility degree as well.

Linking of Agile and Security Activities:

Each agile methodology has its own core procedure engine that security exercises ought to be go through this engine. To infuse a security action into agile method, we ought to coordinate this activity with one of the original method's activities.

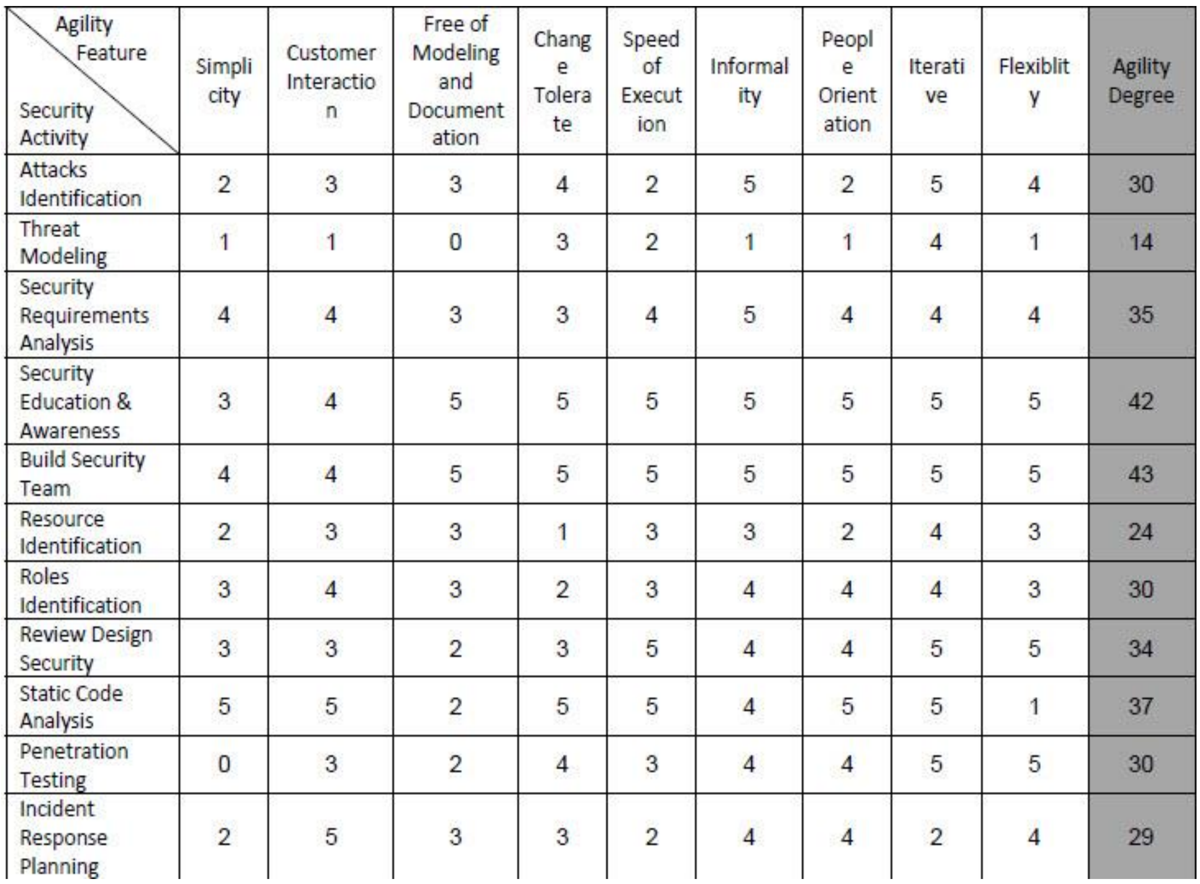
1) The prime step is to analyze agile methodologies and identify their core engine activities.

Combination of external activities with an agile methodology and its activities may prompt a substantial non-agile new activity. Agility degree attribute has been characterized to gauge activities' agile nature. And also security exercises, we ought to compute this worth for filtered agile activities. We can look at agility components of these two sorts of exercises furthermore estimate agility degree of a integrated activity.

Combination of two activities with agility degree of x and y comes about a resulting activity with minimum(x, y) as its degree. It is additionally valid about ADVect of them. Moreover, average agility degree of activities is defined as process's agility degree.

It is difficult to coordinate a security activity with all agile activities. Some activities could be consolidated to shape a new activity. yet it is not valid about every pair of activities. For example "Vulnerability Test" is a security activity that its coordination with "Final Test" activity of dX agile method, comes about a security outfitted test activity. Consequently, these two exercises are integrable. As another illustration, vulnerability test can't be consolidated with "Design" or even "Test-Driven Implementation" activities. We characterize another matrix named activity integration compatibility matrix (AICM).

Table 1 – Security activities and the grade of their agility features. Agility degree is calculated by summation of grades



Agility Feature / Security Activity	Simplicity	Customer Interaction	Free of Modeling and Documentation	Change Tolerate	Speed of Execution	Informality	People Orientation	Iterative	Flexiblity	Agility Degree
Attacks Identification	2	3	3	4	2	5	2	5	4	30
Threat Modeling	1	1	0	3	2	1	1	4	1	14
Security Requirements Analysis	4	4	3	3	4	5	4	4	4	35
Security Education & Awareness	3	4	5	5	5	5	5	5	5	42
Build Security Team	4	4	5	5	5	5	5	5	5	43
Resource Identification	2	3	3	1	3	3	2	4	3	24
Roles Identification	3	4	3	2	3	4	4	4	3	30
Review Design Security	3	3	2	3	5	4	4	5	5	34
Static Code Analysis	5	5	2	5	5	4	5	5	1	37
Penetration Testing	0	3	2	4	3	4	4	5	5	30
Incident Response Planning	2	5	3	3	2	4	4	2	4	29

Agility reduction tolerance (ART) is another parameter utilized as a part of our calculation which controls embeddings of low agile security activities to current agile software development process. ART is a decimal number greater than 0 and lower than 5. A high estimation of ART demonstrates that organization could acknowledge substantial weight security exercises in their software development process to create more secure systems. We will clarify this parameter more and systems to choose it later.

The Activity-Process Linking Algorithm:

Here is an algorithm to choose security activities keeping in mind the end goal to be incorporated with perfect existing organization's agile process activities and enhance security of software development life cycle. It comprises of six stages and controls agility attribute of the new process by method for ART parameter as mentioned before.

Step 1: Select an activity from security activities list with highest agility degree. Assuming *SecActsList* and *ad(x)* as list of security activities and agility degree of activity *x*, we have:

$\text{secAct} = x \mid x \in \text{SecActsList} \wedge$

$\forall_{y \in SecActsList} .\ ad(x) \geq ad(y)$

Step 2: Using filled AICM matrix, form a list of agile activities that are integrable with selected security activity in step 1. Then from this list, select the heaviest one (lower agility degree) to integrate with:

$\text{aglAct} = = x \mid x \in \text{AglActsList} \ \wedge$

$\text{AICM}\,[x, \text{secAct}] \neq 0 \ \wedge$

$\forall_{y \in AglActsList} .\text{ad}(x) \leq \text{ad}(y)$

where aglActsList is the list of activities extracted from current agile process. If there is no compatible agile activity to *secAct*, remove it from the list of security activities and go to step 6.

Step 3: Integrate selected agile and security activities and generate a new activity as *newAct* with new agility degree:

$ADVect_{newAct} = \min(ADVect_{secAct}, ADVect_{aglAct})$

$\Rightarrow \text{ad}\ (\text{new}Act) = \min(ad(secAct), ad(aglAct))$

Step 4: Replace new activity with original one in the agile process and recalculate its agility degree. If original process's agility degree plus *ART* parameter is greater than or equal to new process's agility degree, this integration will be acceptable. Otherwise, selected security activity should not be inserted into current agile software development life cycle.

Step 5: Remove selected security activity (secAct) from the list of security activities (*SecActsList*).

Step 6: If there are some activities in the list of security activities, go to step 1. If not, algorithm terminates. This algorithm leads to a new software development process for project's team that some security activities are integrated with it and agile nature of the method is controlled by means of ART parameter as well.

Agility Reduction Tolerance (ART):

Activities to expand security in software life cycle are typically alongside high measure of effort. They may oblige documentation, formal and semi-formal operations furthermore utilization of some modeling procedures to foresight, prevent, detect and remove security defects from the software. This conduct fluctuates starting with one security activity then onto the next, yet there is a type of weight in a large portion of them. Accordingly, blind joining of activities with agile methodologies may prompt inadmissible decrement in agility of software development team, forcing too cost to the project and development speed lessening also. Then again, security in produced system is a critical issue and to accomplish it, we ought to endure measures of cost and agility level reduction; it is a tradeoff in the middle of security and the expense.

As said before, agility reduction tolerance (ART) parameter defined as a criterion to control agility degree of existing agile process of the project. In the above mentioned algorithm, security activities have an attribute named "agility degree". To incorporate these activities with agile methodology, we begin from the most agile activities to the heaviest one. In this manner we have least rate of diminishment in agile nature of the procedure through method execution.

Tuning ART parameter is SMET's craft to keep a harmony in the middle of security and weight of the product improvement process. SMET ought to have the capacity to quantify expense brought about by expanding the estimation of this parameter. In one hand and measure of harms from security surrenders in the product then again. These harms can be wiped out by executing some security activities being developed life cycle of the product.

Utilizing risk analysis methods to recognize security chances, their potential harm and event probability is a standard approach to evaluate expenses of vulnerable software. Additionally taking into account project's team and current agile process, system designer can measure cost of agility reduction in his development process and lastly with an expense/advantage analysis, calculate perfect quality for ART parameter to infuse some security exercises in existing procedure.

V. SYSTEM ENHANCEMENTS AND FUTURE WORK

We can upgrade the system presented in this paper with a few progressions to it and new ideas. In figuring agility degree of the agile process, we can dole out activities a weight in view of their frequency in software development life cycle and have an all the more genuine agility degree of the process. Analyzing security activities and unwind them to pick up a superior agility degree is another effort that may be done. Making systems reliable, utilization of fuzzy values for AICM matrix and lastly, dependency check among security activities are different upgrades that can be connected. This work is in progress.

VI. CONCLUSION

Security is a vital quality part of software systems and to accomplish this objective, we ought to take care of it during Software development life cycle. Utilizing the introduced method, security exercises can be coordinated to agile strategies to upgrade security of software products. Secure method engineering team (SMET) can tune agility reduction tolerance (ART) parameter to make a harmony between expenses of diminished level of agility as a result of security exercises and advantages from growing more secure systems.

REFERENCES

[1] Howard, M., Lipner, S., *"The Security Development Lifecycle - SDL: A Process for Developing Demonstrably More Secure Software"*, Microsoft Press, 2006.

[2] Gregoire, J., Buyens, K., Win, B. D., Scandarioto, R.,Joosen, W., *"On the Secure Software Development Proecss: CLASP and SDL Compared"*, In Proceedings of the Third International Workshop on Software Engineering for Secure Systems, 2007.

[3] Beznosov, K., Kruchten, P., *“Towards Agile Security Assurance”* In Proceedings of the 2004 Workshop on New Security Paradigms, 2005.

[4] Beznosov, K., *"Extreme Security Engineering: On Employing XP Practices to Achieve ‘Good Enough Security’ without Defining It."*, First ACM Workshop on Business Driven Security Engineering, 2003.

[5] Flechais, I., Sasse, M. A., Hailes, S. M. V., *“A process for developing secure and usable systems”*, In Proceedings of the 2003 Workshop on New Security Paradigms, 2003.

[6] Grance, T., Hash, J., Stevens, M., *"Security Considerations in the Information System Development Life Cycle"*, NIST, Computer Security Division, NIST Special Publication 800-64, REV. 1, 2004.

[7] Swanson, M., etc, *"Security Metrics Guide for Information Technology Systems"*, NIST, Computer Security Division, NIST Special Publication 800-55, 2003.

[8] Agile Alliance, *"Manifesto for Agile Software Development"*, 2005, http://www.agilealliance.org.

[9] Secure Software Inc., *"CLASP: Comprehensive Lightweight Application Security Process"*, Version 2.0,2006, http://www.securesoftware.com/process.

[10] US-CERT, Software Engineering Institute, *"Build Security In"*, 2006, https://buildsecurityin.us cert.gov.

[11] Jürjens, J., *"Developing Secure Systems with UMLsec From Business Processes to Implementation"*, UMLsec homepage, 2002, http://www4.in.tum.de/~umlsec.

[12] The Common Criteria Portal, 2007, http://www.commoncriteriaportal.org.

[13] Systems Security Engineering – Capability Maturity Model (SSE-CMM) official web site, 2007, http://www.sse-cmm.org.

[14] TSP for Secure Systems Development – Presentation.