The Journal of International Scientific Researches
2018, 3(1)

Secure Software Development in Agile Development Processes of E-Government Applications

E-Devlet Uygulamalarının Çevik Geliştirme Süreçlerinde Güvenli Yazılım

Abstract

Agile software development process is found to be the most useful for software industry, since it provides flexibility over requirements and specifications that can change over time. For this reason, government departments and municipalities as well as private organizations can develop products in a faster way but with some disadvantages as well as advantages. One of the concerns is the security problem due to increasing sophisticated attacks and their incrementing costs for cyber defense. Considering the increasing attacks over e-government platforms, development of software requires more emphasis on the security aspect. Particularly for government institutions that mostly have to lean on third party providers for software development that will provide automation of public services via internet, secure software problem became one of the most crucial concerns. Because of some vulnerability that is caused by incremental model developers are enforced to make more secure products. In this paper, large amount of literature has been researched to specify the security issues in agile processes which is the most common and chosen methodology for its elasticity. There are some challenges to provide secure software in agile processes. We have tried to answer why we could not develop secure software because of challenges and what methods can be used to overcome challenges. Comparative security engineering processes have explained to have secure software.

Ahmet Efe
Dr., Ankara Kalkınma Ajansı, CISA, İç Denetçi, aefe@ankaraka.org.tr

Nisanur Mühürdaroğlu
muhurdaroglu.nisanur@gmail.com

Article Type / Makale Türü
Research Article / Araştırma Makalesi

Keywords
Agile Software Development Process, Secure Software, Security Engineering Processes, E-Government Security

Anahtar Kelimeler
Çevik Yazılım Geliştirme Süreci, Güvenli Yazılım, Güvenlik Mühendisliği Süreçleri, E-Devlet Güvenliği

JEL: H11, H83, L86

Submitted: 19 / 02 / 2018
Revised: -
Accepted: 30 / 03 / 2018

Öz

Çevik yazılım geliştirme süreci, zamanla değişebilen şartlar ve şartnameler üzerinde esneklik sağladığı için, yazılım endüstrisi için en yararlı olduğu bulunmuştur. Bu nedenle, devlet daireleri ve belediyeler ile özel kuruluşlar ürünlerin daha hızlı bir şekilde geliştirilebileceği gibi, bazı dezavantajları ve avantajları da beraberinde getirebilirler. Endişelerden biri, artan sofistike saldırılar ve bunların siber savunma için artan maliyetler nedeniyle güvenlik problemidir. E-devlet platformları üzerindeki artan saldırıları göz önüne alarak, yazılımın geliştirilmesi güvenlik yönüne daha fazla önem vermeyi gerektirir. Özellikle internet üzerinden kamu hizmetlerinin otomasyonu sağlayacak yazılım geliştirme için üçüncü parti sağlayıcılara ağırlık vermek zorunda olan devlet kurumları için güvenli yazılım sorunu en önemli endişelerden biri haline geldi. Geliştirici modelden kaynaklanan bazı güvenlik açığı nedeniyle, geliştiriciler daha güvenli ürünler üretmeye zorlanmaktadır. Bu yazıda esnekliği için en yaygın ve seçilmiş yöntem olan çevik süreçlerdeki güvenlik konularını belirlemek için çok sayıda literatür araştırılmıştır. Çevik süreçlerde güvenli yazılım sağlamak için bazı zorluklar vardır. Zorluklardan ve zorlukların üstesinden gelmek için hangi yöntemlerin kullanılabileceğinden dolayı güvenli yazılım geliştirilemediğimizi yanıtlamaya çalıştık. Karşılaştırmalı güvenlik mühendisliği süreçleri, güvenli bir yazılıma sahip olduklarını açıkladı.

Introduction

Security issues are very serious for computers and software. The numbers of attacks by attackers or malicious software are rapidly increasing. These attacks or vulnerabilities could cause to serious problems. Relying upon just functionality of software in critical systems could cause to important security problems.

It is known that e-government applications fail in many areas. However, the linkage of failures with the development method has been the subject of failure of e-health services known as Obama-care. One of the biggest debates about Healthcare.gov's collapse is the way software is developed. Experts think that agile approaches cannot prevent all risk, as war room notes use sprints and story

cards as part of the Agile development method. Reducing risks requires skill and experience. Then again, some media, such as CBS News, suggested that Healthcare.gov was not properly tested by shrinking the test month programs of Healthcare.gov for months (Heusser, 2013).

In the agile software development process, the terms "sprint" and "iteration" refer to the time when a new, complete software development stack is ultimately designed, coded, and fully tested. Normally two weeks is the standard length for teams to start repeatedly; Beyond that, healing means shortening of repetition.

Most teams only use "sprint" or "iteration" to add waterfall concepts. The language like "three architectural sprints, six coding sprints, two test sprints and two hardening sprints" is often interpreted as a clue that something is wrong. For this reason, in our study, it was accepted as a research problem to analyze agile method implementation in the software development processes of e-government applications.

Agile methods have an important impact in recent years for developing software (Ghani & Yasin, 2013). When we compare waterfall model and agile methodology, agile is more proper for customer satisfaction and product quality (Dyba & Dingsoyr). Some researchers have compared agile process with security engineering processes. For the last years, large parts of software companies have switched to more flexible agile methodology from rigid waterfall model.

Agile development successfully could be integrated to traditional development except security level (Wayrynen & M. Boden, 2004). Namely, agile software development makes difficult to develop secure software. There are some criticisms about agile SDLC that produce less secure software. (Alberts & Allen, 2011) (S. Bryan, 2010)

One of the reasons that makes challenging to develop secure software is changing requirement. It makes not possible to see all picture of product since we do not know all requirements of project. There are some literature studies that compare agile processes with security engineering (SE) (J. Wayrynen, 2004) (Bostrom & Jaana Wayrynen, 2006)(Hossein Keramati, 2008), the lack of complete requirements makes it harder to have common SE practices (Baca & Carlsson).

The aim of this study is to define challenges of agile methodology in terms of security and identify what practices from SE processes can be easily integrated to agile processes.

(Beznosov & Kruchten) has defined some mismatches between security assurance and agile processes. One of the most important mismatches is documentation. Security assurance methods are document-driven. But agile methodology does not rely on documentation. As a result of it, even a small part of the software takes too much time to complete. For example, in a study by (Wimmel & Wisspeintner, 2002). It took three months to complete just one iteration with 18 developers. The other mismatches are deeply test analysis and security evaluation in implementation phase.

1. Related Works

Agile processes have been deemed unsuitable for security sensitive software development as the rigors of assurance are seen to conflict with the lightweight and informal nature of agile processes (Peeters, 2017).

Most software developers aren't primarily interested in security. For decades, the focus has been on implementing as much functionality as possible before the deadline, and patching the inevitable bugs when it's time for the next release or hot fix. However, the software engineering community is slowly beginning to realize that information security is also important for software whose primary function isn't related to security. Security features or mechanisms typically aren't prominent in such software's user interface (Tondel, Jaatun, & Meland, 2008).

As threats to applications have increased, developers have begun including security in their software design. Secure development life cycles are methodologies for accomplishing this, it needs to be questioned by researchers, weather companies actually using SDLs (Geer, 2010).

Particularly for e-government software development project that are being provided via contracts with third parties, a solid foundation for acquisition includes not only the required technical and management activities but also the budget, schedule, and staff needed to carry them out. In a study (Creel, 2007) this is found to be challenging, in part due to pressures to reduce costs

and hasten delivery of new capabilities, but also because of historical attitudes toward software. At the policy level, this is beginning to change with software's growing role in implementing critical capabilities and interoperability requirements and with higher expectations for system dependability. But it is asserted that more work is needed for these changes to reach the core of the acquisition program office and impact the outcomes of major systems acquisitions

Previous works in this topic have focused on literature work and some of them have used in industry experience. (Hossein Keramati, 2008) identify security engineering activities in two process Microsoft SDL (Ambler, 2013) and Comprehensive Lightweight Application Security Process. Because of several vulnerabilities in software products and high amount of damage caused by them, software developers are enforced to produce more secure systems. Software grows up through its life cycle, so software development methodologies should pay special attention to security aspects of the product. It is argued that reduction of agile nature of organization's current process can be restrained by means of agility measurement and applying an efficient activity integration algorithm with a tunable parameter named agility reduction tolerance (ART). It is asserted by (Hossein Keramati, 2008) that using this approach, method engineer of the project can enhance his agile software development process with security features to increase product's trustworthiness. The paper introduces Agility degree algorithm for Agile processes. Anderson's book chapter defines what security enginnering is in a detailed way (Anderson, 2003). SE focuses on the tools, processes and methods that we need to design, implement and test complete systems.

(Baca & Carlsson) specify agile development with security engineering activities. This paper specifies security engineering processes for agile development. These SE processes Cigatel Touchpoints, Common Criteria and Microsoft SDL. (Ouslati & M.M. Rahman) paper is literature survey about challenges of agile development processes in terms of security. *SANS* (Integrating Security Into Development, 2006) defines integrating security into software development processes. It provides a methodology that can be used in development processes. (Beznosov & Kruchten) define mismatches between security assurance and agile methodology. It has a table for all phases for software development processes. We could see all the mismatches for all phases. The paper is based on literature studies and the authors to identify some techniques that fit well with agile methodology.

A secure system is one that is protected against specific undesired outcomes. Delivering a secure system, and particularly a secure web application, is not easy. Integrating general-purpose information systems development methods with security development activities could be a useful means to surmount these difficulties. Agile processes, such as Extreme Programming, are of increasing interest in software development. Most significantly for web applications, agile processes encourage and embrace requirements change, which is a desirable characteristic for web application development. In a study by (Ge, Richard F. Paige, Chivers, & Brooke, 2006), it is presented an agile process to deliver secure web applications. The contribution of the research is not the development of a new method or process that addresses security concerns. Rather, they investigated general-purpose information system development methods (e.g., Feature-Driven Development (FDD)) and mature security methods, namely risk analysis, and integrate them to address the development of secure web applications. It is asserted that the key features of their approach are (1) a process capable of dealing with the key challenges of web applications development, namely decreasing life-cycle times and frequently changing requirements; and (2) an iterative approach to risk analysis that integrates security design throughout the development process.

As the use of the Internet and networked systems become more pervasive, the importance of developing secure software increases. In another study (Davis, 2005) it is presented an overview information about existing processes, standards, life cycle models, frameworks, and methodologies that support or could support secure software development. Where applicable and possible, some evaluation or judgment is provided. The study aimed to reach software engineering process group (SEPG) members, software developers, and managers seeking information about existing software development life cycle (SDLO) processes that address security.

In a study by (Beznosov & Kruchten) it is defined mismatches in Agile software development processes with security requirements. At the requirements phase; Specification Analysis and Review, at the design phase; Formal validation, Informal validation, External review, at the implementation phase; Informal requirements traceability, Informal validation, Formal validation, Test depth analysis, Change authorization, External review and Security evaluation. Those are the areas that agile methodology found to be lacking in security.

2. Methodology

We have tried to form some questions related with the security software development. We are going to explain what questions need to be searched in this area:

1) What is agile development process?

Before starting this research, we needed to have detailed explanation about agile development processes.

2) How can we develop secure software?

This question is the starting point. Since, secure software is so important to protect our information and to avoid attacks.

3) How can we provide security assurance in agile development processes?

There are different security assurance techniques to provide security. We need to learn these assurance techniques before relating with agile methodology.

4) What challenges do we have for developing secure software in agile processes?

We have some challenges which prevent to provide security. We need to learn these challenges to provide some solutions.

5) What can we do to have secure product using security engineering processes?

Now, we can provide some solutions to our study after asking previous questions. We have some security engineering process methods to develop secure software.

6) What is security engineering?

We have mentioned about SE processes. Therefore, we need to find the answer to this question. After finding answers to the questions, we tried to compare solutions as far as we understand from literature as well as real software developing life. In this paper, we have written answers to those questions. As a result of answers, we have some questions about result.

7) Are there any problems which cannot be solved easily?

Even though we have solutions, there are some situations which cannot be solved. For example, we need to have documentation for security. But this process is so long. Therefore, it makes difficult to deliver the product up to deadline.

8) Does ISO, COBIT or CMMI standards provide a basis for security in agile?

CMMI has security engineering process just like COBIT-5 that handles, technical and administrative processes interlinked and intertwined according to stakeholder needs and organizational objectives.

3. Backgrounds

This section gives an overview of the ASD approach, secure software development and security assurance cases.

3.1. Agile Development Approach

The individual Agile Methods include Extreme Programming (the most well-known), Scrum, Lean Software Development, Crystal Methodologies, Feature Driven Development, and Dynamic Systems Development Methodology. While there are many differences between these methodologies, they are based on some common principles, such as short development iterations, minimal design up front, emergent design and architecture, collective code ownership and ability for anyone to change any part of the code, direct communication and minimal or no documentation (the code is the documentation), and gradual building of test cases. Some of these practices are in direct conflict with secure SDLC processes. For example, a design based on secure design principles that addresses security risks identified during an up-front activity such as Threat Modeling, is an

integral part of most secure SDLC processes, but conflicts with the emergent requirements and emergent design principles of Agile Methods (Davis, 2005).

The agile development approach has several advantages. Firstly, it decreases the possibility of project failure since it provides early detection of missing requirements thanks to customer participation. Secondly, it provides to learn customer needs rather than customer wishes. Thirdly, it enables to see technical obstacles before starting the project. (Othmane, Weffers, Angin, & Bhargava, 2014)

Note that there is a disadvantage about quality requirement. Since, it includes just software functionality not quality requirements like security requirements. (Ouslati & M.M. Rahman)

In Agile Software Development has no rigid rule like waterfall model or traditional models. We have continued iterations. Adding new features quickly and code changing requests are very important and priority. This can lead to conflicts in term of security. One of the most used methods to integrate security principles is integrating security team members with senior coders to software development team.

Table 1. Principles for Agile Software Development

Code Principle
P1. Our first priority is to meet customer needs and implement them continuously.
P2. Look with favor on changing requirements.
P3. Deliver working software frequently, with a timescale.
P4. Customers and developers should work together daily during the project.
P5. Working with senior developers is very important to build a satisfied project.
P6. One of the most effective method of communication with development team is face-to-face conversation.
P7. Working software is very important for continuity of progress.
P8. Agile processes support maintainable development. Therefore, the developers and users can maintain product indefinitely.
P9. Technical perfection and good design improves development process.
P10. The team makes meeting about how to become more effective on regular timescale.

These are code principles that are selected from (Ouslati & M.M. Rahman). In practice, traditional methods which have similar features can be integrated easily to agile methods. We will use this theory to implement secure software in agile methodology.

There is a similar process as named ASD, shown in figure 1. It has three phases. These phases are *inception*, *transition* and *construction*.

- *Inception phase*: defines goal of project and it is initial architecture
- *Construction phase*: implementing software in iterations. There is a requirement for iterations. Development team and customer should define scope of iteration.
- *Transition phase*: this is testing part during integration. This part is necessary to make it ready a new release.

3.2. Overview of developing secure software

There is a model for secure software engineering as named

System Security Engineering-Capability Maturity Model

(SSE-CMM) which has three processes: *risk process, engineering process*, and *assurance process*. The *risk process* identifies risks that the system has. It detects security vulnerabilities. The *security engineering process* determines solutions to the threats. The security assurance process enables confidence since the implemented security solutions decrease the security risks. (Ouslati & M.M. Rahman)

Table 2. Security Risks and Security Assurance Practices

Code	*Assurance practice*	*Description*
S1	Determine security policies	We need to determine the security needs to meet security policy. These policies become the security goals of project.
S2	Security training	We need to train developers about security techniques. We can use security coding standards using tools.
S3	Define vulnerabilities	We need to define vulnerabilities. They can be source code vulnerabilities or weaknesses in the software architecture, such as unprotected sending data in a public network
S4	Identify security risks	We need to identify the security threats to the software and define their probabilities when we have an impact.
S5	Verify and validate the security	We need to make security analysis of source code and show whether the customers security needs are met.

3.3. Challenges of Developing Secure Software Using Agile Methodology

We could divide into 4 parts as following: Software development life-cycle challenges, Incremental development challenges, Security assurance challenges, awareness and collaboration challenges. Now, we will mention about these challenges.

3.3.1. Software development life-cycle challenges

In software development life-cycle, we do not have security requirement phase and risk assessment for agile development processes. We cannot include security activities to agile. Since, much iteration are required, they lead to time consuming to develop secure software.

3.3.2. Incremental development challenges

In incremental development, we need to make refactoring to prevent unstructured code and change requirements frequently. These break security constraints. Because of changing requirements security assurance is difficult to success.

3.3.3. Security assurance challenges

To provide security assurance we need detailed documentation. In addition to that, tests are not sufficient to ensure whether implementation has security requirements. Therefore, we could not detect all vulnerability cases in test phase.

3.3.4. Awareness and collaboration challenges

Developers could neglect security requirements because of lacking experience on secure software. Therefore, it will be good to separate team as security and functionality implementation. In addition, customers could not aware of security.

3.4. Agile Scrum

Scrum is known as a simple framework used to organize teams and make them more productive by working better. Scrum is a "lean" approach to software development that allows teams to develop software to choose the amount of work done and decide how best to do it. This system is designed to adapt to the changing needs of the software development process in a short and regular intervals. It is based on fulfilling the customer and business requirements to develop the working product in real time according to the needs of the customer. In this way, Scrum provides what the customer wants at the time of delivery, and also removes the waste known as the business which is not highly valued by the customer (J. Sutherland, 2011).

3.5. Sprints

Sprints is one of the basic concepts of Agile, and in general it is the process of dividing the project into smaller pieces. For example, when the project is to design a word processor application; save, print, and divide labels into small workpieces or sprints. When these functions are combined with other functions, it is understood what the application is building.

3.6. Daily scrum

Daily scrum or stand-up meeting means daily meetings for team members and scrum masters and product owners (K. Schwaber, 2002). These are often referred to as "affiliated" members. Other members are "included" members; These may be sales directors, Chief Technology Officer, Chief Information Officer (CIO), or Chief Information Security Officer (CISO). The meeting is normally requested to be held at the same time and place, and each team member is required to answer the following questions to the meeting members:

a)What have I done since the last scrum meeting?

b)What I want to achieve at the next scrum meeting?

c)Which obstacle is slowing me down?

In the event of slippage or delays, the product owner immediately understands the situation and can take corrective action. Theoretically, the product owner should be aware of all problems that are not older than a day in general. As you can see, Agile is clearly focused on producing good code. In the minds of security advisors and accreditors, there are questions such as "where is safety at all in this rapid development" and "how do I provide accreditation of products".

3.7. Issues with Agile Scrum

Agile scrum has obvious advantages in favor of Agile when compared to waterfall methodology. For example, with the ability to change user requirements, four key values can be seen when looking at Agility values:

a) Communication and interaction on individuals, processes and tools

b) Software running on comprehensive documentation

c) Cooperation of the client in negotiating the contract

d) Ability to respond to change by following a plan

It is argued that there is a gap in agile methodology and that security is lacking in thought. (Howard, 2005) The growing tendency to use agile techniques to build Web applications is argued to mean that security engineering methods must be integrated into agile processes (Oueslati, 2016)

GAO has identified 14 difficult areas in addition to many conveniences and advantages in implementing the Agile method in the federal environment (GAO, 2012):

- Software teams were having difficulty working closely together.
- Procurement processes cannot support agile projects.
- The teams have difficulty in transitioning to self-directed work.
- Clients often do not trust replicated solutions.
- Employees have difficulty in providing more timely and frequent input.
- Teams have difficulty managing the iterative process requirements.
- Agencies confused employees.
- Compliance evaluations are difficult to implement within the iteration time.
- There are difficulties in getting new vehicles on time.
- The official reporting practices of government agencies are not compatible with Agile.
- There are difficulties in creating and maintaining technical environments.
- Traditional works of art do not fit Agile.
- Agile guidance was not clear.
- Traditional status monitoring is not compatible with Agile

4. Secure Software Developments That Can Be Embedded in an Agile Methodology

In this section we briefly describe generalization of the three examined SE processes. These are Cigatel Touchpoint, Common Criteria and Microsoft SDL. They are used in different phases of development process. In order, the project is expected to pass these phases: requirements (Rq), design(D), Implementation(I), Test (T) and Release(R).

4.1. Cigatel Touchpoints

Cigatel Touchpoints (N. Davis) has been defined as an SE process. Its processes can be integrated to existing development processes. In addition, it increases quality and security.

In requirement phase, we should include security requirements besides functional requirements. In addition, we should define abuse cases under attack. These cases can include following case; what should be protected from whom.

In design phase, risk analysis is very important. If we skip this part at early phases, it can cause to costly problems. Besides, designers should prepare assumption documentation and predict possible attacks.

In implementation phase, static code analyses are made by developers. But it is not sufficient for secure software.

In testing phase, penetration testing, red team testing and risk based testing should be made. Penetration testing simulates real world. We can identify clearly attack patterns. We need red team and risk based testing to make more secure software. Red team testing works with standard functional testing techniques. Risk based testing is based on attack patterns.

In release phase, the person who is outside the design team can analyze existing touchpoints.

4.2. Common Criteria

Common Criteria (Ouslati & M.M. Rahman) is well used SE principal that is ISO certified. In requirement phase, we need to define security requirements as documentation. Stakeholders should be included to requirement phase. It provides to define common security definitions.

In design phase, risk analyses should be made by designers. Risk is determined form application to application. Our aim is to reach 100% risk acceptance. This should be captured in risk assessment document.

In design phase, Security Readiness Review (SRR) is used. It has some vulnerable assets by requirements engineer. In addition, designers should review quality of team's work as well as security requirement assessment.

4.3. Microsoft Security Development Lifecycle Process

The Microsoft Security Development Lifecycle (SDL) (Integrating Security Into Development, 2006) is a software development process that is created by Microsoft. It increases software reliability about security threats. Spiral model taken into account which is risk-driven approach to design this process. Also, it is Agile friendly.

In requirement phase, we need to define security requirement for given software project and possible user roles as well as their access level to the software. Also, we should ensure whether technical design specifications are appropriate to security requirements.

In design phase, designers should make quality measures such as activities for completion of requirements. Besides, cost analyses should be made for different possible threats.

Designers validate existing threat models for correctness of product design. The threat modeling should define vulnerabilities. Additionally, they reduce attack surfaces by simplifying interfaces.

In implementation phase, security tools are used to assist project. Developers explain unsafe functions and specify some recommendations for these unsafe functions. Static code analyses must be made like other SE processes.

In testing phase, dynamic analysis should be achieved. Dynamic testing tools can be used to do this. Manual code review of risk assessment could be useful. Fuzzy testing should be made by using fuzzy test tools. After explaining results, developers should develop a mitigation strategy for given software.

In release phase, incident response planning that provides clear guidelines should be created. Finally, all threat models and security tool results should be reviewed.

4.4. COBIT-5

COBIT 5 includes a process reference model, defining and describing in detail a number of governance and management processes. It provides a process reference model that represents all of the processes normally found in an enterprise relating to IT activities, offering a common reference model understandable to operational IT and business managers. The proposed process model is a complete, comprehensive model, but it is not the only possible process model. Each enterprise must define its own process set, taking into account the specific situation. Incorporating an operational model and a common language for all parts of the enterprise involved in IT activities is one of the

most important and critical steps towards good governance. It also provides a framework for measuring and monitoring IT performance, communicating with service providers, and integrating best management practices.

Regarding the software development COBIT-5 requires an integrated and holistic way of development aligned with business objectives based on the enterprise risk profile. The focus is on identifying and analyzing how IT creates value for the enterprise in enabling business transformation in an agile way, in making the current business processes more efficient, in making the enterprise more effective, and in meeting governance-related requirements such as managing risk, ensuring security, and complying with legal and regulatory requirements.

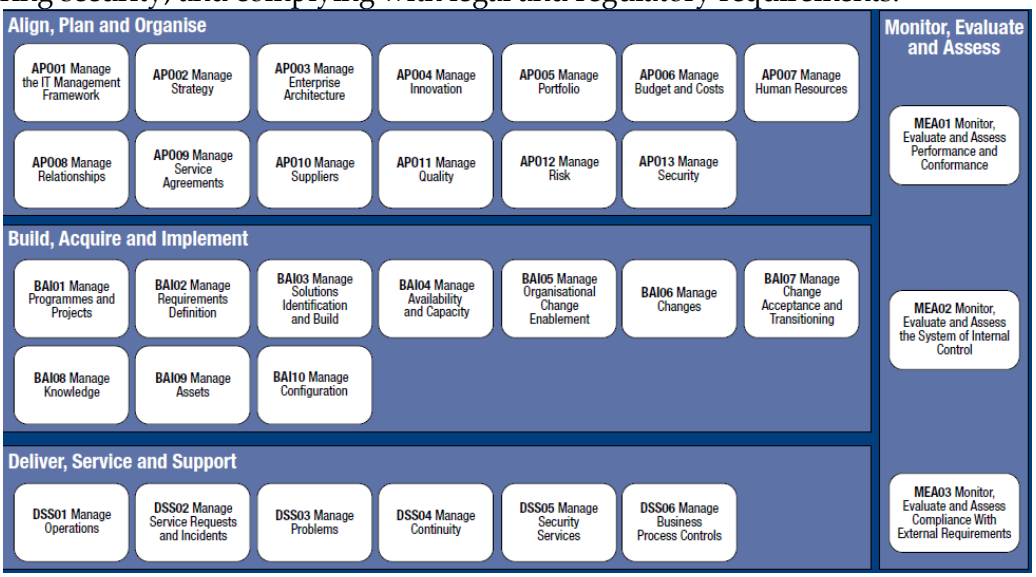

Figure 1. COBIT 5 Process Reference Model of Management

Source: (ISACA, 2012)

As is seen in the above Fig. 1. there are many processes to be implemented as part of management domain. Without an integrated methodology that takes into consideration all business and IT related processes while developing software. Here as it is shown in the Fig. 2. below, there are some risks that need to be mitigated by pertinent processes. At the right side of the table the number and names of the COBIT-5 processes are written.

Risk Scenario		COBIT 5 Process Capabilities
Architectural agility and flexibility	• Complex and inflexible IT architecture obstructing further evolution and expansion	• APO01 Efficient and defined business and IT-related processes • EDM04 Governance over resource optimisation • APO02 Responsive strategic planning • APO03 Maintenance of enterprise architecture • APO04 Innovation and initiation of change • APO05 Portfolio management decision taking • BAI02,03 Agile development life cycle methods • APO13 Maintaining security in an agile and flexible environment
Software implementation	• Operational glitches when new software is made operational • Users not prepared to use and exploit new application software	• APO11 Consistent and effective quality management activities • BAI01 Project management • BAI02 Requirements definitions • BAI03 Solution development • BAI05 Managing organisational changes with regards to software implementation • BAI06 Change management • BAI07 Extensive solution testing • BAI08 Knowledge support
Software integrity	• Intentional modification of software leading to wrong data or fraudulent actions • Unintentional modification of software leading to unexpected results • Unintentional configuration and change management errors	• BAI02 Definition of application control requirements • BAI06 Change management • BAI07 Testing and acceptance practices • BAI10 Configuration data • DSS05 Access controls • DSS06 Business process controls
Software performance	• Regular software malfunctioning of critical application software • Intermittent performance problems with important system software	• BAI03 Software development quality assurance • BAI04 Planning for and addressing capacity and performance issues • DSS03 Root cause analysis and problem resolution
Ageing of application software	• Application software that is old, poorly documented, expensive to maintain, difficult to extend or not integrated in current architecture	• EDM04 Resource management direction and/or oversight • APO02 Recognising and strategically addressing current IT capability issues • APO03 Maintaining enterprise architecture • APO04 Identifying new and important technology trends • BAI03 Maintaining applications • BAI09 Maintaining assets • DSS06 Business process controls

Figure 2. Risk Scenarios and COBIT 5 Process Capabilities

Source: (ISACA, 2012)

4.5. Others

SANS introduced a new methodology as called "*Scalable and Agile Lifecycle Security for Applications" (SALSA)*. Actually, it is not real development process. It provides to increase code analysis. SALSA defines the following policies:

1) Developers should be educated about how to analyze attacks and the potential threats.

2) Security practices should be integrated into application life cycle throughout implementation.

3) Security should be integrated into automated building process. This suggestion is perfect for Agile methodology, since automated processes are main part of rapid changes.

4) Developers should make security training as well as security professionals should make development training to create quality product.

5) Vulnerability assessment and penetration testing should be included to process whenever possible. This method will help us to find missed vulnerabilities.

6) More transparent code helps to have clear information of all stakeholders about project. (Integrating Security Into Development, 2006)

5. Discussions and Proposed Future Work

We have mentioned about some security engineering processes as named cigatel touchpoints, common criteria, Microsoft SDL and some other methodologies. These methods are generally similar to each other in terms of software development phases. Even though we have some solutions to have secure software in agile processes. It seems these solutions are not enough. Because we have some trade-offs result of the SE processes. Achieving security in agile processes will be a time-consuming process. If security is very important for us, we need to change our software development approach. There are some software development approaches special to secure

products. But if we need to compare these SE processes Microsoft SDL process would be more useful to us, since it has more detailed process than other SE processes. Requirement, design, implementation and testing phases have more specialized analysis for security.

Automation in the software development processes is also possible. As is known, the deep coding project that has been started by Microsoft in 2016, there will be some robotic software using AI and machine learning technology to provide required software programs. Further work can be done on the possibility, success and variation of the deep coding technology that can surely affect agile methodology (Bigs, 2017).

Conclusions

Project managers and business owners can often try to bypass the security team without informing them about decisions made in the project. This can lead to undesirable effects that can lead to security threats, weaknesses and security breaches. State units that provide public services electronically and develop software should be a formal process of providing evidence to businesses that a system or application is as secure as possible. Businesses that have developed e-government applications need to work with security at the beginning of the discovery phase; security is not late for the development process, not for the end, but for security to have any effect. Security should be seen as an activating part of the work, and it should certainly be buried in the process of the overhaul, subtracting it from what is being considered in the last minute (Harrison, 2016).

In this paper, we have mentioned about security challenges and the ways of developing secure software integrating SE processes and agile development processes. Namely, agile has been criticized for lacking security due to its incremental approach. The security activities were chosen to integrated security into existing development processes. These were Cigatel Touchpoints (N. Davis), Common Criteria (Ouslati & M.M. Rahman) and Microsoft SDL (Integrating Security Into Development, 2006). In a future work these proposed SE processes would be integrated to existing development processes.

We have concluded that in order for a reasonable security practice, an organization should implement an integrated and holistic approach to all processes including agile development process. Particularly for the government agencies and ministries that constantly develop new software for a broader range of e-services, it becomes more important to align security with organizational and administrative processes and policies. It is found that COBIT-5 framework that has been developed by ISACA which has acquired the CMMI, provides a good opportunity to integrate secure software development with other required management and governance processes.

References

Alberts, J., & Allen, R. (2011). *Risk based measurement and analysis: Application to software security.* Carneige Mellon University Pittsburg: Software Enginnering Instıtute.

Ambler, S. (2013). Retrieved from The Agile System Develpoment Lifecycle: http://www.ambysoft.com/agileLifecycle.html

Anderson, R. (2003). What is Security Engineering? In *Security Engineering.* New York: Wiley.

Baca, D., & Carlsson, B. (n.d.). Agile Development with Security Engineering Activities.

Beznosov, K., & Kruchten, P. (n.d.). Towards Agile Security Assurance.

Bigs, J. (2017, 02 25). *DeepCoder builds programs using code it finds lying around.* Retrieved from techcrunch: https://techcrunch.com/2017/02/23/deepcoder-builds-programs-using-code-it-finds-lying-around/

Bostrom, G., & Jaana Wayrynen, M. B. (2006). *Extending XP Practices to support Security Requirements Engineering* (pp. 11-17). ACM SESS 06.

Creel, R. (2007). Assuring Software Systems Security: Life Cycle Considerations for Government Acquisitions. *Carnegie Mellon University* , https://www.us-cert.gov/bsi/articles/best-practices/acquisition/assuring-software-systems-security---life-cycle-considerations-government-acquisitions.

Davis, N. (2005). *Secure Software Development Life Cycle Processes: A Technology Scouting Report.* http://www.dtic.mil/docs/citations/ADA447047: Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst.

Dyba, T., & Dingsoyr, T. (n.d.). Emprical Studies of Agile Software Development: A Systematic Review. *Information and Software Technology Elseiver*, 883-859.

Efe, A. (2013). COBIT-5 Framework As A Model For The Regional Development Agencies In TURKEY. *INTERNATIONAL JOURNAL OF eBUSINESS AND eGOVERNMENT STUDIES*, 33-43, http://dergipark.gov.tr/download/article-file/257103.

Efe, A. (2016). Kamu Yönetiminde COBIT-5 Çerçevesinde Risk Yönetimi: Türkiye'de Kalkınma Ajansları Özelinde Bir Analiz . *Uluslararası Eğitim Bilim ve Teknoloji Dergisi*, 1 - 18, http://dergipark.gov.tr/uebt/issue/21610/232109.

Efe, A. (2016). Unearthing and Enhancing Intelligence and Wisdom Within the COBIT 5 Governance of Information Model. *ISACA Journal*, http://www.isaca.org/Knowledge-Center/Research/Documents/COBIT-Focus-Unearthing-and-Enhancing-Intelligence-and-Wisdom-Within_nlt_Eng_0416.pdf.

Efe, A. (2017). A Model Proposal for Organizational Prudence and Wisdom Within Governance of Business and Enterprise IT. *ISACA Journal*, http://www.isaca.org/Knowledge-Center/Research/Documents/COBIT-Focus-A-Model-Proposal-for-Organizational-Prudence_nlt_Eng_0317.pdf.

Efe, A. (2017). Kamu Yönetiminde Cobit-5 Bilişim Yönetişiminin Kalkinma Ajanslari Özelinde Uygulanabilirliği. *Yönetim Bilişim Sistemleri Dergisi*, 1-26, http://dergipark.gov.tr/download/article-file/331323.

Ge, X., Richard F. Paige, F. A., Chivers, H., & Brooke, P. J. (2006). Agile development of secure web applications. *Proceeding ICWE '06 Proceedings of the 6th international conference on Web engineering* , (pp. 305-312). Palo Alto, California, USA .

Geer, D. (2010). Are Companies Actually Using Secure Development Life Cycles? *Computer, Volume: 43 Issue: 6*.

Ghani, I., & Yasin, I. (2013). *Software Security Engineering in Extreme Programming Methodology: A Sistematic Literature* (pp. 215-221). Science International Volume.

Hossein Keramati, S.-H. M. (2008). Computer Systems and Applications. *Integating Software Development Security Activities with Agile Methodologies* (pp. 749-754). ACS/IEEE International Conference on Computer Systems and Applications.

Howard, H., & Lipner, S. (2006). The Security Development Lifecycle. Microsoft Press.

J. Wayrynen, M. B. (August 15-18, 2004). *Security Engineering and Extreme Programmng and Agile Methods.* Calgary, Canada.

Othmane, L., Weffers, H., Angin, P., & Bhargava, B. (2014). *Extending the agile development process to develop acceptably secure software.* IEEE Transactions on dependable and secure computing.

Ouslati, H., & M.M. Rahman, L. O. (n.d.). Literature Review of the Challenges of Developing Secure Software Using the Agile Approach.

Peeters, J. (2017). *Secure Application Development*. Retrieved from Agile Security Requirements Engineering: https://handouts.secappdev.org/handouts/2008/abuser%20stories.pdf

S. Bryan, S. (2010). Security Practices for Agile Development. *MSDN Magazine*.

Tondel, I. A., Jaatun, M. G., & Meland, P. H. (2008). Security Requirements for the Rest of Us: A Survey. *IEEE Software*.

Wayrynen, J., & M. Boden, G. B. (2004). Security. *Engineering and Extreme Programming: An Impossible Marriage?* (p. 117). In Proceedings of the 4th Conference on Extreme Programming and Agile Methods.

Wimmel, M. V., & Wisspeintner, A. (2002). *Secure Systems Development Based on the Common Criteria: the palME Project.* South Carolina, USA: Tenth ACM SIGSOFT Symposium.