SECURITY ENGINEERING METHODS AND APPROACHES USED IN SDLC

Mohammed AbuLamddi*

ABSTRACT

For the development and maintenance of system and software design and are compatible with operating systems. In the old it takes into account the increase in production as a major factor in the software industry, safety and ease of software engineering. In industry e.g. IEEE STD 1228-1994 [1] with the development and progress and openness we face a lot of difficulties, and surround us sudden risks and threats. And the more progress we the more complicated those risks. The protections of those risks and avoid a prerequisite everyone is seeking to achieve. Although one of the greatest dangers of our time are those that threaten the security of information and computers. Multiplied these risks and varied methods and different motives. If any facility for small and large seek to maintain the confidentiality of the information that is owned and safety. But with the large number and diversity of programs and the multiplicity of styles available now penetrating devices and vandalism and access to confidential information is increasingly easy with the days. This article assesses the threats and risks faced by the phases of the software development life and reviewed by firing so light on the importance of resolving the problem in the analysis and design phases of the program better than to take preventive and defensive means to avoid these risks.

Keywords: *Software Life-cycle Management, SDLC, security engineering, enterprise security architecture, secure information system, security risk analysis, security management.*

1 INTRODUCTION

Know the life cycle of the system (System Development Life Cycle) as the activities associated with the system, which includes the establishment of the system, development and implementation, operation and maintenance and surrender completely. Appeared this science because of the difficulties faced by the programmers and developers at the beginning of forties in terms of the lack of rules and regulations specify consecutive operations to build the systems in addition to the lack of policy specifies processes and divided the business on the team and most of this lack of references and the intent of this document the stages of building the system software within the rules prior to order refer to it in case I want to change or develop a functions due to the surrounding environment, these reasons in turn led to the emergence of technology life cycle systems in late 1960 so that appeared first spark the emergence of specialty Computer Engineering, which introduced to the scope of disciplines computing focuses this specialty technology life cycle systems in all its aspects. and there is a misconception when some that programming is a process for the manufacture of the system software, but is a stage of stages building system software and these stages are described steps shared to build and develop systems from the beginning of the feasibility study through built to maintain and finally delivered. over the years emerged precipitant methodologies which gave a new character to the life cycle of the regulations and also the owner of this evolution appears languages specialized uses in building stages system, due to the development of life-cycle systems to the system programmatic in nature needs to development and change to keep pace with emerging requirements and differences that appear in the environment surrounding it.

Life cycle systems mainly focus on building a system from the moment of receipt of the idea in mind to get out of the system to the customer, and service after the delivery of the system in terms of maintenance and provide the guide and teach the use of the system. The systems life cycle (SDLC) contain three main objectives: to make sure that the systems are delivered with high quality, providing sound administrative controls to manage the project, increasing the productivity of team rules. In order to achieve these goals must perform a lot of requirements that include: support for all technical activities and technical support of all administrative activities Table 1.

Table 1. Technical, administrative activities.

Technical activities	**Administrative activities**
Definition Team (analyst, designer, programmer, testing).	Identify priorities and define target
Installation of the system (training, such as writing the guide).	Project tracking and assess the situation
Support productivity (management problems and solutions).	Risk assessment and monitoring work step-by-step
Definition of updated versions and evaluate alternatives.	Analysis of costs and benefits
	Supplier management and stakeholders
	Ensure quality assurance

2 METHODOLOGIES

Several methodologies have been developed in order to be a guide to direct operations within the life cycle of systems, and methodologies is important for the development of regulations accidentally deliberate and systematic in the sense based on the ways and means working to find the best path to get to the processing and quality assurance system. These methodologies include the following:

Waterfall Model; Prototyping; Iterative, Increment Framework; Spiral Model; Rapid Application Development; Join Application Development; Agile Software Development.

The simplest methodology and model for the life cycle of systems is what is known model Waterfall Model, which focuses on developing the regulations sequentially any phase begins after the end of the stage that preceded. Composed form of several stages with each stage a group of activities must be completed within the time periods specified in advance after the

completion of all activities in the current phase can move on to the next stage began in the implementation of its activities. Figure 1

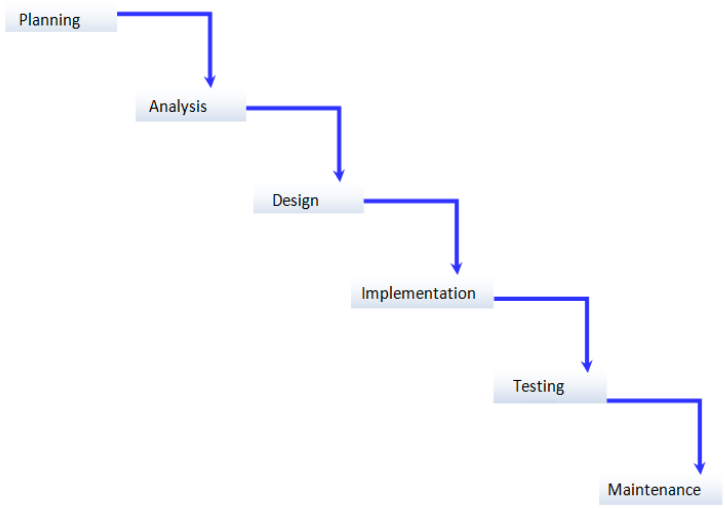

Figure 1. Phases of the development life cycle program

3 ANALYZE SECURITY REQUIREMENTS

Systems analyst must after identifying the problem and before starting the analysis of the current system that takes into account the constraints and limitations that may conflict with the goals of the system. It depends the success of an information system on its ability to achieve its objectives in the light of these limitations. One of the most important limitations as follows: Administration policy; Individuals; Resistance to change; Finance; Available technical; Environmental constraints.

Security requirements analysis is a critical task, and application teams and partly because either ignore or get this error function. All too many cases, they put "the cart before the horse", and design security solution before addressing the security needs of the real business level. Want to combine business managers and application development and security team executives to better understand the key sensitivities and the consequences of the risk of information security, including the breach of various scenarios that are accessed information illegally or damaged. Your objective is to collect the information needed to drive the next phase - architecture and design.

The analysis can be divided into security requirements in those subtasks:

- Evaluate security risks and consequences.
- Perform asset value analysis.

- Discuss the potential threats.
- Analyze potentially malicious or harmful activities.
- Analyze high-level vulnerabilities.
- Discuss the security goals.
- Review regulatory requirements and corporate Information Security policy.
- Review future business goals.

Directed discuss security requirements, and begin to identify what should be security risks involved for your business, and that is, what do you want to protect against. The sequence subtasks in such a way to help guide the team to express their needs, and thus provide the basis for the requirements and priority document. The priority document requirements logically leads you to the next stage, and design.

4 SECURITY IN DESIGN PHASE

At every stage we have to include enhanced security. Although it is very necessary for all the necessary security features stage depth starting from the design stage. The condition in the information gathering stage brought mostly occurs between the user and the developer because of this need of the security level will be less than that. Apart from that the user to collect reliable information that should also be taken the correct information for the development of the system. In the analysis phase, and what information we get from the requirements phase that would give to the analysis phase.

We will be analyzing the information collected using some materials valid documents, white papers, and existing methods. Information collected in the form of the structure of the model is structured. The same in the analysis phase we have to estimate what kind of security requirements need for our system. It should include elements of security and features in every aspect of the system, such as user data, unit, design, testing.

Developers need to know the principles of safe design software and how they are employed in the design of systems flexible and trustworthy. Two basic concepts of the design include abstraction and decomposition of the system by using architecture and constraints to achieve the security requirements that were obtained during the requirements phase. Most readers are probably familiar with these concepts. Abstraction is the process to reduce the complexity of the system by removing unnecessary details and isolate the most important elements to make the design more easily. Decomposition is the process to describe the generalizations that form an abstract idea.

One method, from the top to the bottom of decomposition, and involves the collapse of a large system into smaller parts. For object-oriented designs, the progress the application, module, class, and method. Further details safe principles of software design in many books and articles, and web portals, and articles. In this paper we present some techniques to improve the SDLC [2] Table 2.

Table 2. Techniques to improve the SDLC.

Element	**Explanation**
Maintainability	Code must be characterized by good maintenance easily and this means that it can easily evolve to meet changing requirements. This feature is very important because change is inevitable in a world that is changing every day.
Dependability	System must not cause any significant damage in the event of failure.
Efficiency	The system must not be overly consumed computing resources such as memory and processor. Effectiveness include a number of performance-related aspects such as: Response time, processing time, memory consumption.
Usability	Software must be easy to use. The ease of use includes two aspects: 1 - Design and appropriate interfaces 2 - Well documented code.

5 CONCLUSION

Even if security is considered throughout the SDLC when building software, and even if extensive testing is performed, vulnerabilities will likely still exist in the software after deployment, simply because no useful software is 100 percent secure [3]. Software can be designed and developed to be extremely secure, but if it is deployed and operated in an insecure fashion, many vulnerabilities can be introduced. For example, a piece of software might provide strong encryption and proper authentication before allowing access to encrypted data, but if an attacker can obtain valid authentication credentials, he or she can subvert the software's security. Nothing is 100 percent secure, which means that the environment must always be secured and monitored to thwart attacks. Although security requirements engineering is an area of active research, many promising techniques have already emerged that you can use to identify the requirements needed to improve your software products. It seems obvious that systematic and thorough identification of security requirements in the early stages of the SDLC will result in more secure software and reduced security costs later on [4].

REFERENCES

[1] Software Engineering Standards Committee, IEEE Standard for Software Reviews, SH94592,4 March 1998.

[2] Patrick McBride, & Edward P. Moser. Secure System Development Life Cycle (SDLC), METASeS, 2000.

[3] Viega, John, & McGraw, Gary. Building Secure Software: How to Avoid Security Problems the Right Way. Boston, MA: Addison-Wesley, 2001.

[4] P.Mahizharuvi, K.Alagarsamy, A Security Approach in System Development Life Cycle, Int. J. Comp. Tech. Appl., Vol 2 (2), 253-257.