Multilevel Security Spiral (MSS) Model: NOVEL Approach

Shams Tabrez Siddiqui
Research Scholar, Department of Computer Science, Aligarh Muslim University, Aligarh

Hatem S. A. Hamatta
Department of Applied Sciences, Al-Balqa Applied University, Aqaba, Jordan

M.U.Bokhari
Associate Prof, Department of Computer Science, Aligarh Muslim University, Aligarh

ABSTRACT
In software system development, security is a very crucial issue and is the demand of time in this connected world. But often engineers think after software development. Security should be given higher priority in software development life cycle. Considering security from early stages of life indicates good research and development. This paper mainly focuses on security activities involve in developing secure software's. Identifying security risks and managing those risks based on spiral model. A new spiral model with Multilevel Security Spiral (MSS) has been proposed with security activities.

General Terms
Software system development, security, priority, spiral model.

Keywords
Software, security risks, multilevel security spiral, software development life cycle.

1. INTRODUCTION
Security for a software system has always inverted and address solely within the production environment through perimeter security like firewall, proxy, antivirus, platform security, and intrusion prevention system [1, 21]. This is the reason for considering security as a non-functional requirement.

The non functional requirements are separate from procedural requirements that includes; maintainability, reusability, reliability portability, and security. Others do neither have any exact specification nor have any metrics for specifying objectives of requirements [24]. The security criteria which include authentication, privacy, authorization, and integrity of some applications exhibit to less variation.

The system is concerned with an entity that interacts with the environment such as hardware, software, human, and physical world with its natural phenomena. Systems fundamental characteristics are computation and communication and this may be characterized by fundamental properties as performance, behavior, dependability, and security. Performance means how long the system performs, satisfies the user needs. Behavior includes what the system does to implement its function and is described by a sequence of states. A secure system does what it is supposed to do and what is not supposed to do. Dependability means to avoid service failures and measure availability, reliability, and maintenance of the system that are more frequent [6, 17]. The three main aspects of security are Confidentiality, Integrity, and Availability. The given aspect of security is also referred as CIA [28]. Confidentiality is all about maintaining privacy from unauthorized users. Integrity is about ensuring the accuracy and completeness of information. Availability is ensuring about the information available to authorized users [17]. A software which enlists the above three aspects is considered as secure software. Security usually considered at post development activity but it should be considered during pre-development and development phases [1]. After software development, organizations try to incorporate security as a patch, but security is not a feature it is an emergent property of a complete system [1]. Organizations spend a lot of money in purchasing good firewalls and antivirus programs but there software's are not secure. Due to exploitation of security flaws they incur heavy losses of information.

This paper mainly focuses on issues of security activities of software development life cycle that requires a careful consideration that includes security in pre-requirements, requirements, design, implementation, testing, deployment, and maintenance phase [29]. The main focus of security activity is to identify security risks and managing those risks from the early stage of each phase. Based on spiral model and its risk perception, a new model has been proposed and named as Multilevel Security Spiral (MSS).

2. SECURITY ACTIVITIES FOR SECURE SOFTWARE DEVELOPMENT
While developing software developers never ever try to find out the root cause of unsecured software. From requirement gathering to software development, from software testing to maintenance, the whole project team from starting of the project puts their best effort into the Software development Lifecycle to ensure quality software but even then the software is produced with a number of security flaws [25]. Figure 1 shows the status of security according its positions in the developed system. It shows security is completely ignored in a way that there exists no such thing as security. Also, if security requirements were there in first place, then the steps following the requirements gathering phase which include design, development and testing would have taken care of security. Finally, if security incorporated in every phase of the software development life cycle, then software developed through it would have been more secure [5].

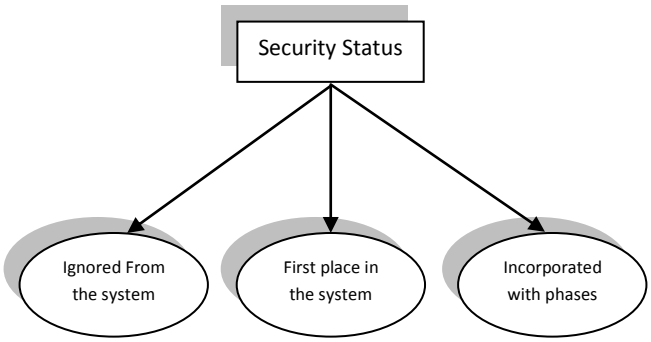

Fig 1: Security Status in Developed Systems

Security activities are divided into different phases (figure 2) of the life cycle starting from the requirements Phase but there are a few activities that need to be performed before the actual project start of.

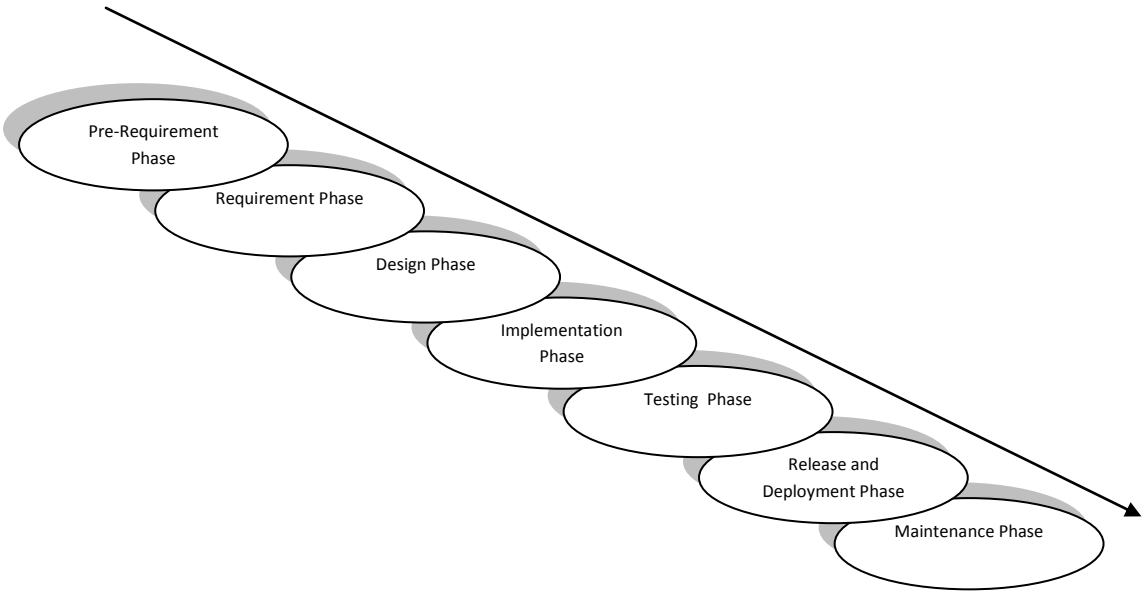

Fig 2: Security Activities in Software Development

2.1 Pre-requirements Phase

Security activities performed in this phase set the foundation for all the activities starting from the requirements phase right up to the maintenance phase. The following activities are performed during this phase:

2.1.1 Security Training

Many organizations are unaware of the importance of security and those who are aware of security are unaware of how to improve it. If the top management is not aware of the security, then they would not be willing to spend time and money on it. Security training should be divided into two parts i.e. general training and specific training. Fig 3 shows the scope of Security Training and their Impacts.

General training is for the whole organization. Its purpose is to create awareness of security in an organization's environment and it prove to be successful if everyone in an organization starts thinking about security in their plans, scheduling and acts (if needed).

Specific training is limited for the project team only. It varies from project to project. The goal of specific training is to identify the flaws; threats associated with them, and identify security measures that should be taken to secure software from these threats [19].

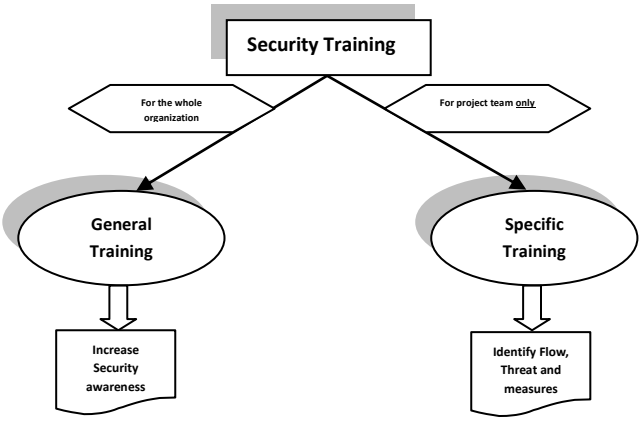

Fig 3: Scope of Security Training and their Impacts

General and specific training is shown in fig 3 develop a security culture into the organization with everyone taking security seriously [18]. Security trainings are successful if each individual is aware of security importance and knows what role he or she has to play towards the development of secure software [18].

2.1.2 Plan and Develop Framework for Risk Management

The plan is to identify risk items early in the development life cycle and develop some strategies to solve risk items. Set down some agenda to resolve new risk items that are highlighted [13]. Risk management framework should be incorporated in the beginning, so that top management and owners get educated about the risks and their possible impact. Risk management framework should work as reminder of security importance over a period of time.

2.2 Requirements Phase

Software development life cycle starts form requirements phase. During requirements phase, goal of the project team is to identify security requirements in a clear and understandable form from security point of view. Following activities are performed in requirements phase.

2.2.1 Identify Security Requirements

Requirements are the bases of all the phases in software development and describe what the system should do. If requirements are ambiguous, incorrect and incomplete, then the whole project is at risk and if they are clear, correct and complete, then the whole project can be a success (ignoring other constraints).

Most of the security requirements are identified at the start of the project during the requirements gathering phase, and some come from threat models, industry standards, and certification processes later on [7].

2.2.2 Develop Misuse Cases

The focus is on 'what' and not on 'how' during requirement gathering. The next step is move towards 'how' on abstract level without going into the design and implementation details when functional, non-functional and security requirements are identified, gathered, analyzed, agreed upon and documented. Security requirements describe what the system should not do, while functional requirements describe what the system should do [4]. Use cases are designed to explain functional requirements. Misuse cases are very useful in eliciting security requirements and in understanding possible attacks, actors, relation with allowed functionalities, estimating amount of loss, and determining system's behavior before and after the attacks [4].

2.3 Design Phase

This phase describes 'how' the features identified during the requirements phase, will be implemented. Design phase acts as a bridge between the requirement and implementation phases therefore it has to be complete and concise to serve as a starting point for software implementation.

The design phase holds a key importance for security purposes because large numbers of security problems have their origin in this phase. According to Howard and LeBlanc, 50% of security bugs are found during the design phase [9].

2.3.1 Build Security Architecture

Security architecture defines design techniques, use of strong typed language, application of least privilege, and minimization of attack surface [7].

Security architecture identifies important components whose correct functionality is important for security along with their properties and interaction between them [7, 20]. It provides an overall picture of security design. It does not contain a detailed design or any development details. Security architecture does not remain the same though out the life cycle therefore identification of new threats, changes in priorities of threats, change in software architecture etc cause security architecture to be revised and updated regularly[26].

2.3.2 Identify Interaction Points, Assets and their Access Points

Almost all software applications interact with other software applications in one way or another. Software architecture and design show interaction of components within the software and with other software. These interactions are very important because all of them are not safe [21].

Identification of assets that software manages and their access points is also very important step towards secure software [8]. Assets may include data, information, piece of code etc.

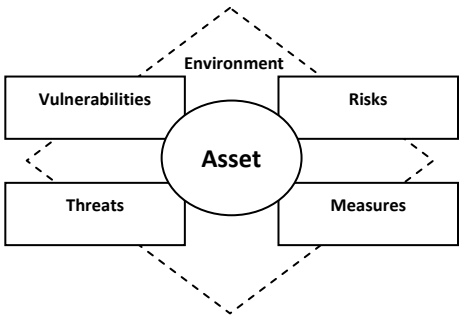

Fig 4: Assets relationship with environments and software component

Identification of these assets shown in fig 4, when and how they are stored and accessed is important for identifying security threats associated to these assets .

2.3.3 Minimize Software Attack Surface

Since no software can be 100% secure, therefore minimizing or reducing software attack surface is a technique used to minimize the attacks possible on the software system.

The design phase should specify the features with the default settings and in which circumstances the software will have more features for users [27]. It should also specify how software design deals with these circumstances [22].

2.4 Implementation Phase

Most of the security activities described so far is design centric. Design centric activities focus on architectural flaws built into software design but overlook implementation level flaws that a software developer might introduce into the software during coding [10, 11]. Like requirements and design phases, securing an implementation phase is also important because some flaws that are born with code and in earlier phases are impossible to detect.

Implementation phase is the last phase where security flaws can enter into the system. If secured, then the chances of flaws entering into the software are minimized. Implementation level flaws are those flaws which the programmer introduces into the application due to poor coding, lack of knowledge of security or use of un-safe methods etc. It is important to know that the use of security tools will not make the software secure, it only helps in making software secure [23].

2.4.1 Write Secure Code

To avoid security flaws in the implementation phase, the first and the most important thing the development team can do is to write secure code. Development team should understand the nature of implementation level security flaws, how they occur and what impact they can have on the software [13, 16].

2.5 Testing Phase

The focus has been on making secure software by incorporating security into the software development activities and by preventing security flaws chipping into the software. When the software enters into the testing phase, it is functionally complete [7].We can identify flaws that already exist in the software but we cannot make software secure by preventing flaws entering into the system.

Security testing is considered as a difficult job than the usual functional testing. It is different from usual software testing because understanding the mindset of an attacker is totally different from that of normal user [14]. Security testing demands testers to have expertise in security concepts, security flaws and attack methods.

2.5.1 Security Test Planning

In security testing, test planning can be considered as the most important phase. During test planning, testers make use of misuse cases and threat models to identify possible attacks, how they can be carried out and their impact. Testers then prepare test scripts for testing. These scripts are designed to attack software successfully.

During test planning, testing schedule is created. Priorities are assigned to test cases. Execution pattern of test cases is identified. Testing depth is identified for different areas. High risk or security critical areas may need more detailed testing than other areas. Bug severities are revised according to

security testing and tool selection is done. Briefly security test planning is not only about designing test cases but is about organizing the whole security testing activity efficiently [33].

2.5.2 Security Testing

Security testing is difficult as compared to the usual functional testing. In security testing, the tester needs to think like an attacker [12]. Security testing must involve two approaches [14], performing functional testing of security mechanisms and performing risk based testing to simulate attacker's behavior. Functional testing of security mechanisms makes use of use cases while risk based testing make use of misuse cases. Risk based testing requires understanding of the attacker's mindset [15].

Automated tools should be used for testing. Many automated tools are available in the market but most of them are for functional testing, but there are few which are for security testing. The use of automated tools can cut down manual testing efforts and testing time, hence providing more time for the testing team to focus on risky areas.

2.6 Release and Deployment Phase

When the software reaches the Release phase, it is assumed that most of the security flaws have been identified in preceding phases and the software is stable and is ready for deployment. But before deployment, the software is subjected to security review and audit. The purpose of this security review and audit is to identify any remaining security flaws and gain confidence over the software.

2.6.1 Security Review

The software is subjected to security review, after the testing phase. The main purpose of this review is to find out any remaining security flaws. Security review has to be well planned so that it does not take much time and resources. Security Review can be considered as the last activity where software is analyzed for security.

2.6.2 Security Audit

Security audit is final words on software security and nothing more than a question answer session. Based on the checklist, the security audit seeks to identify whether security practices have been followed during the software development or not. Top management look at results of the security audit and get an overall view for software from the security point of view.

2.6.3 Security Deployment

Security deployment is deploying software with default settings and configurations. The goal of these settings is to reduce possible attacks by minimizing the attack surface and it is identified during the design phase.

2.7 Maintenance Phase

After deployment the software goes into the hands of the end user. The software cannot be 100% secure [7]. Due to time constraints or any other reason there might be some identified security flaws present that were not fixed during software development. These flaws are looked again, prioritized, fixed and the software is monitored for new threats. New threats keep on coming and efforts are required to protect the software against them. Maintenance is an ongoing process and does not end until the software is completely out of use or taken over by a new software.

3. SPIRAL MODEL

Waterfall model and spiral model are used commonly in developing system. There are many weaknesses in the waterfall model. The major weakness is that once the requirement gathered in the first phase there is no formal way to change the requirement after going to the next phase and there is no risk analysis. The most important feature of the spiral model is that each cycle in spiral completed and covers all products that developed during cycle [2, 8]. It also includes all plans for the next cycle. The model is shown in fig.5.

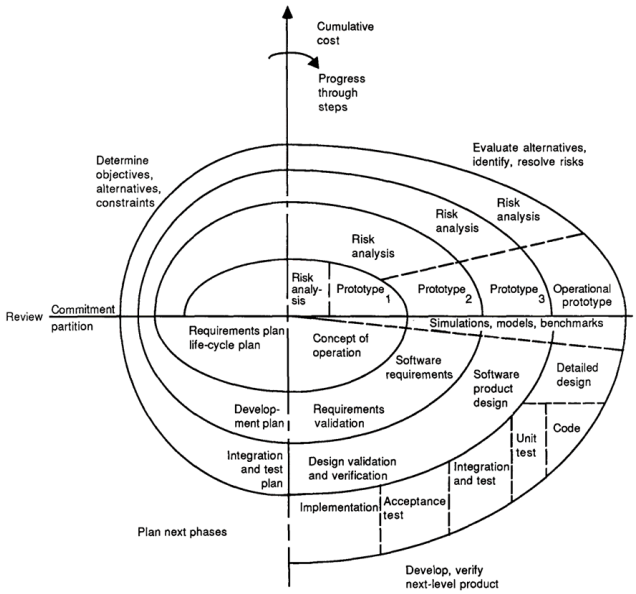

Fig 5: Spiral model [13].

The advantage of spiral model over the waterfall model is that the iterative approach that allows the developer even all the system requirement is not known to them. Spiral model considers security from the very beginning of the software development life cycle [2]. Good for large, expensive, complicated, and mission-critical projects. Software is produced early in the software development life cycle [30]. There is another model knows as extreme programming but the weakness is its dependency of code.

The four steps which involve in the spiral model are [31].

1. Determine objectives, alternatives, constraints: Find out the purpose of software and identify the measures to serves all requirements that should be taken to achieve the software.
2. Identify, resolve risks: Calculate all the possible risks that you may encounter during software development life cycle. Figure out how you will resolve those risks.
3. Develop next level product: Create software and test it for its functionality, reliability and security.
4. Plan next iteration: after you have completed one cycle and got your first prototype1, then re-pan and repeat the cycle so that in, you may proceed to get your next prototype2.

4. MULTILEVEL SECURITY SPIRAL (MSS) MODEL

Security activities for development of secure software models should be followed right from the beginning. A careful consideration for security is required in pre-requirements, requirements, design, implementation, testing, deployment

and maintenance phase. The main focus of security spiral is to identify security risks and managing those risks [8]. Multilevel Security Spiral (MSS) considers security from the very beginning of the software development life cycle . Spiral model is as organized approach for developing software. Spiral model is based on risk perception for project [3].

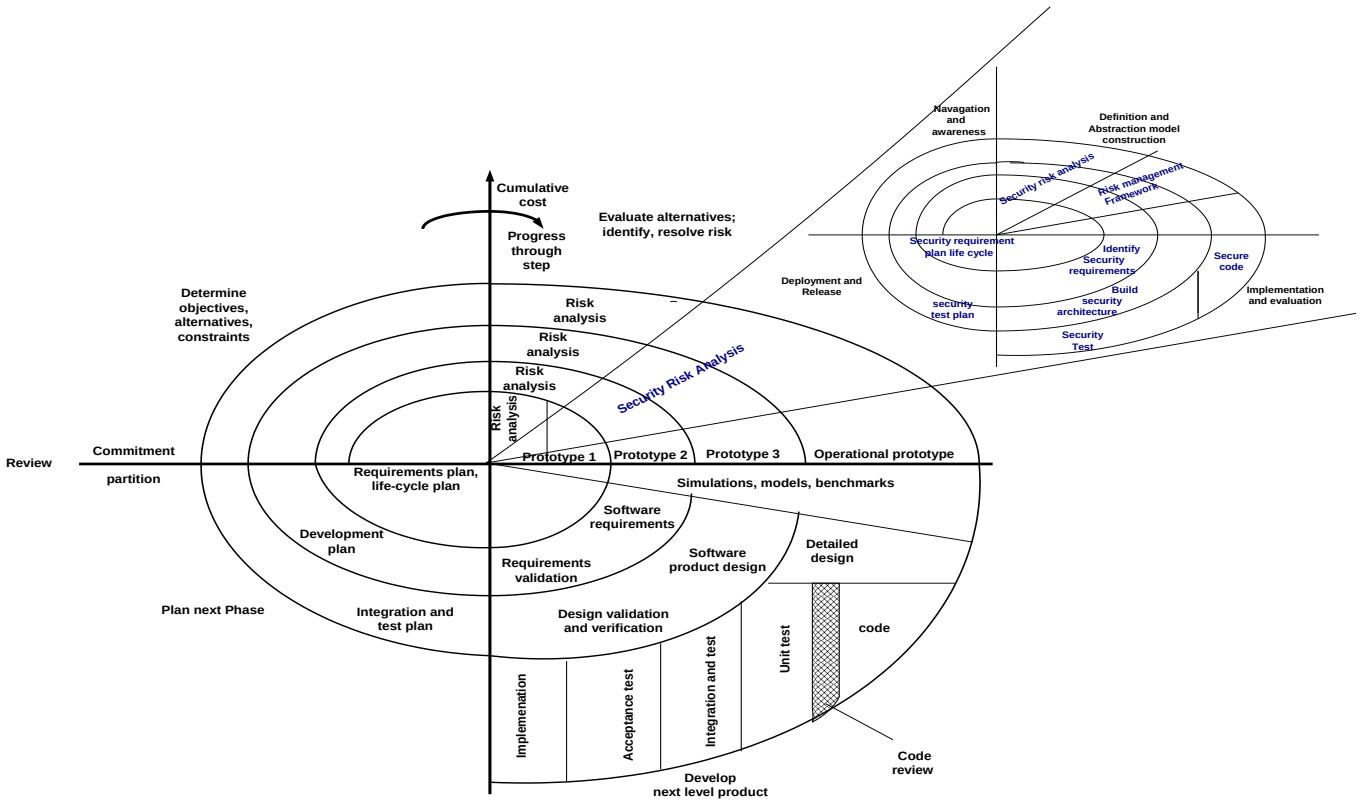

Fig 6: Security Spiral model

Each cycle of the Multilevel spiral model begin with identification of objectives, constraints, alternatives doable for achieving software. Fig 6 shows the security risk analysis in beginning of each cycle in spiral. After identifying the risk, next step is to develop strategy to resolve the uncertainty & risks [8]. This can be done by security spiral model which is included in the security risk analysis area. The activities in secure spiral model are organized very much like a traditional spiral that has many cycles. In the beginning of cycle, develop framework for risk management. After resolve risk, next step is to develop next product in which we create the software.

Including security in requirement phase and in design phase, after every step it must be revised by the programmer considering the risks. If any possible risk is encountered then it should be resolve in that step. At each step, extensions and design modification should be made. Secure code should written, code must be reviewed by the programmer himself considering the risks. Security test should be there after integration test before implementation. So that the Multilevel Security Spiral (MSS) model should be risk free and highly secure. Multilevel Security Spiral (MSS) model is better in all prospective from the other models (Table 1).

Table 1. Strengths and Weaknesses of Models.

Strengths and Weaknesses of Models.	**Waterfall**	**Incremental**	**Spiral**	**MSS**
Provide work force for Security specialization	X	√	√	√
Operating in Organized environment	√	√	√	√
Dynamic reporting for Risks and Risk Assessment	X	√	√	√
Allocation of resources and facilitates [32]	√	√	√	√
Early functionality	X	√	√	√
Complete set of requirements is not required at the onset [32]	X	√	√	√
Through prototyping controls cost and risk [32]	X	X	√	√
Provides Security Awareness	X	X	X	√

5. CONCLUSION

This paper mainly focuses on the activities that involve in developing secure life cycle of software. It requires a careful consideration of security in pre-requirements, requirements, design, implementation, testing, deployment and maintenance phase. The main focus of security spiral is to identify security risks and managing those risks. A new model based on spiral model and its risk perception, has been proposed and named as Multilevel Security Spiral (MSS). In MSS, security spiral model is included within the spiral model right from the beginning of the software development life cycle.

The software should be developed keeping in mind the secure software development activities and risks. After every step it must be revised by the programmer considering the risks. Secure code should written, code must be reviewed by the programmer himself considering the risks before implementation. So that the Multilevel Security Spiral (MSS) model should be risk free and highly secure.

6. REFERENCES

[1] G.McGraw, "Managing Software Security Risks",IEEE Security & Privacy, Volume 2, Issue 2, Mar- Apr 2004, Page(s): 80-83.

[2] M.I.Daud,"Secure Software Development Model: A Guide for Secure Software Life Cycle", Proceedings of the international MultiConference of Engineers and Computer Scientists 2010, Hong Kong.

[3] P. Jalote, "An Integrated Approach to software Engineering" 2nd Edn, Springer,1997.

[4] Guttom Sindre, Andreas L. Opdahl, "Eliciting Security Requirements by Misuse Cases", IEEE Explore, 2000.

[5] H.M.Shirazi, "A new model for secure software development", Int J. Intellig. Inform, Technol. Appl, 2009.

[6] Algirdas Avi_zienis, Fellow, IEEE, Jean-Claude Laprie, Brian Randell, and Carl Landwehr, "Basic Concept and Taxonomy of Dependable and Secure Computing", IEEE Transactions on Dependable and Secure Computing, Vol. 1,No. 1, January-March 2004.

[7] Steve Lipner, Michael Howard, "The Trustworthy computing Security Development Lifecycle", Microsoft Corporation. March 2005.

[8] Daljit kaur, Parminder kaur and Hardeep Singh, "Secure Spiral: A Secure Software development Model", Journal of Software Engineering 6(1): 10-15, 2012.

[9] Michael Howard and David LeBlanc, "Writing Secure Code", Second Edition, Microsoft Press, 2002.

[10] Kenneth R, "Bridging the Gap between Software Development and Information Security", Security & Privacy Magazine, IEEE, volume 3, issue 5, Sep- Oct, 2005, Page(s):75-79.

[11] Elfriede Dustin, "The Secure Software Development Lifecycle", Dev Source (sponsored by Microsoft), 2006.

[12] Bruce Potter, "Software Security Testing", IEEE Security & Privacy Magazine, volume 2, issue 5, Sep-Oct, 2004, Page(s) 81-85.

[13] Barry W. Boehm, "A Spiral Model of Software Development and Enhancement", TRW Defense Systems Group. Volume 21, Issue-5, 1988

[14] J. Whittaker, "Why Secure Applications Are Difficult to write", IEEE Security & Privacy Magazine, volume 1, issue 2, 2003, Page(s) 81-83.

[15] H.H. Thompson, "Why Security Testing is Hard", IEEE Security & Privacy, vol. 1, no. 4, 2003, pp. 83–86.

[16] Michael Howard, "A Process of Performing Security Code Reviews", Security & Privacy Magazine, IEEE, volume 4, issue 4, Jul- Aug, 2006, Page(s): 74-79.

[17] M.U.Bokhari and Shams T. Siddiqui, "A Comparative study of software requirements tools for secure software Development" BVICAM'S International Journal of IT(BIJIT), 2010.

[18] G. McGraw, "Adopting an Enterprise Software Security Framework", Security & Privacy Magazine, IEEE, volume 4, issue 2, Mar-Apr, 2006, Page(s): 84-87.

[19] A.S Sodiya, S.A.Onashoga, and O.B.Ajayi, "Towards Building Secure Software Systems", Volume 3, 2006.

[20] Len Bass, Paul Clements, and Rick Kazman, "Software Architecture in Practice",Second Edition. Addison Wesley 2003.

[21] Asoke K Talukder, "Security-aware Software Developemnt Life Cycle(SaSDLC)- Processes and Tools", IWOCON 2009, Cairo,Egypt, 28-30 April 2009.

[22] Gunar Peterson, "Collboration in Secure Development Process, Part 2", Information Security Bullettin, Volume 9, Page 210, June 2004.

[23] Michael Howard, "A Look Inside the Security Development Lifecycle at Microsoft", MSDN Magazine, November 2005.

[24] M.U.Bokhari and Shams T. Siddiqui, "Metrics for Requirement Engineering and Automated Requirement Tools", Proceedings of the 5th National Conference; INDIACom-2011, New Delhi.

[25] K.Hans, "Cutting edge practices for secure software engineering. Int.J.Comput Sci Security", 2010 4: 403-408.

[26] Peter Eeles, Senior IT Architect, IBM, "What is software architecture", 15th Feb 2006.

[27] Michael Howard, "Mitigate Security Risks by Minimizing the Code You Expose to Untrusted Users". MSDN Magazine, November 2004.

[28] M.U.Bokhari, Shams T. Siddiqui and Hatem S.A. Hamatta ," Object Oriented Software Security in Design Phase", IJCST Vol. 3, Issue 4, Oct- Dec 2012.

[29] William tozier.com, "Notional Slurry | Pontification without all the gritty gravitas - Bill Tozier's".

[30] Nabil Mohammed Ali Munassar and A. Govardhan, "A Comparison between Five Models of Software Engineering", International Journal of Computer Science Issues, Vol. 7, Issue 5, September 2010.

[31] Zeepedia.com, Online book .Com [Online]. See on link, http://www.zeepedia.com/read.php?spiral_model_determine_objectives_alternatives_and_constraints_prototyping_information_systems&b=14&c=21.

[32] Reed Sorensen, "A Comparison of Software Development Methodologies" , Software Technology Support Center, 1995.