$$CPSNR = 10\log_{10}\left(\frac{255^2}{CMSE}\right)$$

Where

Color Mean Square Error (CMSE) is calculated as

$$CMSE = \left(\frac{1}{3HW}\right)\sum_{i=r,g,b}\sum_{y=1}^{H}\sum_{x=1}^{W}\left(I_o(x,y,i) - I_r(x,y,i)\right)^2$$

I_o and I_r represents the original and the reconstructed (interpolated) images of sizes H(height) x W (width) each.

Table 1: CPSNR Comparisons

Image	BI [14]	OR [4]	ECI [13]	ESSC [3]	AHD [8]	SA [15]	VCD [7]	Proposed method
1	34.8	41.3	41.7	42.9	41.9	41.8	42.9	43.1
2	27.7	37.9	35.0	38.1	37.6	39.1	40.0	40.2
3	33.4	41.4	40.5	42.7	40.9	41.5	42.2	42.1
4	23.4	31.0	30.5	35.2	33.8	35.9	36.4	37.4
5	32.2	41.3	39.6	42.6	41.1	41.9	43.0	43.0
6	29.0	37.4	36.2	39.5	37.5	38.9	39.9	40.1
7	31.1	41.8	37.9	41.2	41.4	41.6	43.6	42.7
8	27.8	38.1	35.3	40.1	38.5	40.0	41.0	40.8
9	31.5	39.1	38.7	41.3	39.3	40.5	41.1	41.4
10	28.4	36.2	35.7	39.0	36.6	38.9	39.1	39.6
11	30.1	36.5	36.5	38.3	36.5	37.7	38.0	39.5
12	26.8	32.5	33.8	34.8	33.6	34.8	35.0	38.7

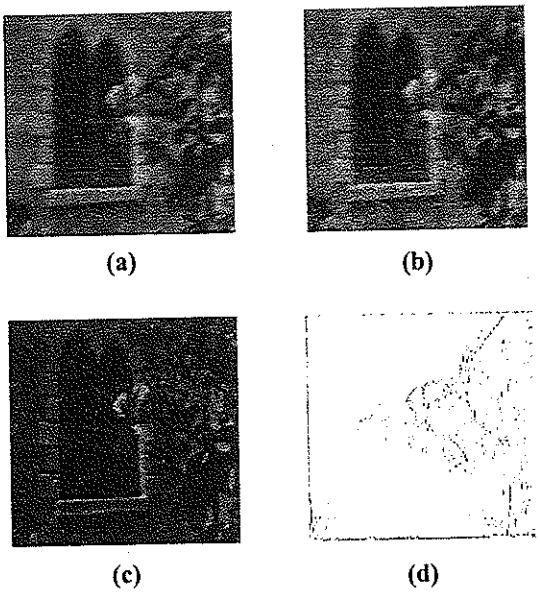

Figure 2: a, Blurred Image of Window;
b, Mosaic Image of window
c, Demosaiced Image of Window;
d, Error During Demosaicing and Sharpening.

Table 1 shows the comparison of CPSNR with seven existing methods BI [14], OR [4], ECI [13], ESSC [3], AHD [8], SA [15], VCD [7].

4.2 Structural Similarity Comparison

For image quality assessment, the Structural Similarity (SSIM) index is a highly useful metric, since image statistical features are highly spatial and non-stationary in general [16]. And at a typical viewing distance only, a local area in the image can be perceived with high resolution by the human observer at the instance. The SSIM index is defined as the function of luminance, contrast and structure. Figure 3 shows the SSIM index values comparison between the color interpolation using variances of color differences and the proposed method.

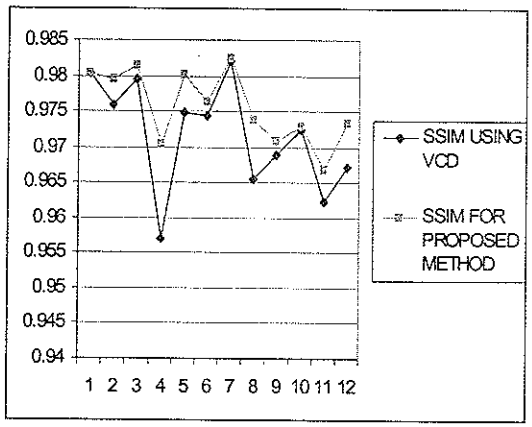

Figure 3: SSIM Comparison

5. Conclusion

A novel highly edge preserving, adaptive weighted color interpolation algorithm along with adaptive weighted sharpening for single sensor digital still cameras equipped with Bayer color filter array is presented. This proposed adaptive weighted sharpening method effectively enhance the image with the help of edge details. The proposed weighted color interpolation algorithm makes use of the variance of color difference to estimate the interpolation direction and weight value according to the edge value for interpolating missing samples. The refinement of red and blue pixel values using weighted color difference effectively exploited the spectral correlation and resulted in consistent and optimal image quality. Simulation results show that the proposed interpolation algorithm is able to produce subjectively and objectively better results as compared with a number of existing algorithms. By introducing adaptive weighted method for texture region, the performance of the algorithm can be improved.

References

[1] B.Bayer, *"Color imaging array"*, U.S. Patent No. 3 971 065, Jul 1976.

[2] B. K. Gunturk, J. Glotzbach, Y. Altunbask, R. W. Schafer and R. M. Mersereau, *"Demosaicing: Color filter array interpolation"*, IEEE Signal Process. Mag., Vol. 22, No. 1, PP. 44-54, Jan 2005.

[3] Chang and Y P Tan, *"Effective Use of Spatial and Spectral Correlations for Color Filter Array Demosaicking"*, IEEE Trans. Image Process., Vol. 14, Feb 2004.

[4] D. D. Muresan and T. W. Parks, *"Optimal recovery demosaicing"*, IEEE Trans. Image Prs., Vol. 14, No. 2, PP 267-268, Feb 2005.

[5] H.J. Trussell and R.E. Hartwig, *"Mathematics for demosaicking"*, IEEE Transaction on Image Processing, Vol. 11, No. 4, PP 485-492, April 2002.

[6] J. F. Hamilton and J. E. Adams, *"Adaptive Color Plane Interpolation in Single Sensor Color Electronic Camera"*, U.S. Patent 5 629-734, 1997.

[7] KH Chung, YH Chan, *"Color Demosaicing Using Variance of Color Differences"*, IEEE Transaction on Image Processing., Vol. 14, Oct 2006.

[8] K. Hirakawa and T. W. Parks, *"Adaptive homogeneity - directed demosaicing algorithm,"* IEEE Transaction on Image Processing, Vol. 14, No. 3, PP 360-369, Mar. 2005.

[9] N. Kehtarnavaz, H.J Oh and Y. Yoo, *"Color filter array interpolation using color correlations and directional derivatives"*, Journal of Electronic Imaging, 2003.

[10] N.Krishnan, S.S.Vinsley, et al, *"Effective iterative Demosaicing"*, IEEE and SIPCICOM Sponsored International Conference, Proceedings PP 335-338 India, Feb 2007.

[11] N.Krishnan, S.S.Vinsley, et al, *"A noval adaptive color interpolation algorithm for single sensor digital camera images"*, Geospatial Conference, India 2007.

[12] R. Lukac, K. Martin and K. N. Plataniotis, *"Demosaicked image postprocessing using local color ratios"*, IEEE Trans.CircuitsSyst. VideoTec., Vol.14, No.6, PP 914-920, Jun. 2004.

[13] S.C.Pei and I.K.Tam, *"Effective color interpolation in CCD color filter arrays using signal correlation"*, IEEE Trans. Circuits Syst. Video Tec., Vol.13, No.6, PP 503-513,Jun. 2003.

[14] T. Sakamoto, C. Nakanishi and T. Hase, *"Software pixel interpolation for digital still camera suitable for a 32-bit MCU,"* IEEE Trans. Consum. Electron., Vol. 44, No. 4, PP 1342-1352, Nov. 1998.

[15] X. Li, *"Demosaicing by successive approximation"*, IEEE Trans. Image Process., Vol. 14, No. 3, PP 370-379, Mar. 2005.

[16] Z Wang, A C Bovik, H R Sheikh, E P Simoncelli, *"Image Quality Assessment: From Error Visibility to Structural Similarity"*, IEEE Trans. on Image Processing, Vol. 13, No. 4, April 2004.

[17] R. Lukac and K. N. Plataniotis, *"Cost effective sharpening of single sensor camera images"*, ICME, IEE Explore 2006.

Author's Biography

Prof. Dr. Nallaperumal Krishnan received M.Sc. Degree in Mathematics from Madurai Kamaraj University, Madurai, India in 1985, M.Tech Degree in Computer and Information Sciences from Cochin University of Science and Technology, Kochi, India in 1988 and Ph.D. degree in Computer Science & Engineering from Manonmaniam Sundaranar University, Tirunelveli. Currently, He is Heading Center for Information Technology and Engineering of Manonmaniam Sundaranar University. His research interests include Signal and Image Processing, Remote Sensing, Visual Perception, and mathematical morphology fuzzy logic and pattern recognition. He has authored three books, edited 18 volumes and published 30 scientific papers in Journals. He is a Senior Member of the IEEE.

S.S.Vinsley received the B.E Degree in Electronics and Communication Engineering and M.E Degree in Communication Systems from Madurai Kamaraj University, Madurai. He is currently pursuing the Ph.D degree at Manonmaniam Sundaranar University, Tirunelveli.

C.Seldev Christopher received the B.E Degree in Electronics and Communication Engineering from Manonmaniam Sundaranar University and M.E Degree in Communication Systems from Madurai Kamaraj University, Madurai. He is currently pursuing the Ph.D degree at Manonmaniam Sundaranar University, Tirunelveli.

Specifying Secured Recuirement

R. A. Khan[1], K. Mustafa[2]

ABSTRACT

Generally, software engineers are poorly trained to elicit, analyze, and specify security requirements, often confusing them with the architectural security mechanisms that are traditionally used to fulfill them. There is a great demand to apply a continuous security mechanism when developing the system. One of the most ignored parts of a security-enhanced software development lifecycle is the security requirements engineering process. This study presents a structured approach for security requirement specification. A prescriptive framework, Secured Requirement Specification Framework (SRSF), has been proposed as a major contribution in this paper.

Keywords : Software Security, Security Requirement, Risk Analysis, Use Cases, Abuse Cases, Secure Software Development Life Cycle.

1. INTRODUCTION

Software is said to be secure if it can function properly despite of malicious attacks and threats. Security is important in all aspects of life, and the increasing pervasiveness and capability of information technology makes IT infrastructure security increasingly so [1]. The continual and increasing publicity given to failures of IT security demonstrate the importance of developing and assuring software to appropriate levels of security. An article by C. Mann 'Why is Software So Bad', concludes that bad habits and inadequate software life cycle processes have led to the development of poor software [2]. No doubt, there is advancement in the software engineering process and tools, but the literature survey reveals that the progress in improving the quality of software is still lagging [3]. This assessment can be made with respect to security by answering the question why is software so insecure and vulnerable?

Secure software is software that is able to resist most attacks, tolerate the majority of attacks it cannot resist, and recover quickly with a minimum of damage from the very few attacks it cannot tolerate. Secure software cannot be intentionally subverted or forced to fail. It remains dependable in spite of intentional efforts to compromise that dependability [4]. The lack of rigor and discipline in the software development process, driven by the focus on short time-to-market, performance and functionality, has produced rampant security vulnerabilities that gravely affect a large range of computing environments, from small deeply embedded safety applications to large enterprise software platforms [6].

In the traditional software development lifecycle (SDLC), security is often an afterthought, and security estimation and prediction efforts are delayed until after the software has been developed. Vulnerabilities are an emergent property of software which appears throughout the development phases. Therefore, it is highly desirable to adopt a 'before, during, and after' approach of software security to software development process [7]. A

[1]Department of IT, Babasaheb Bhimrao Ambedkar University, Lucknow, UP India e-mail : khanraees@yahoo.com
[2]Department of Computer Science, Jamia Millia Islamia, New Delhi-India e-mail : kmfarooki@yahoo.com

life cycle process that includes security assurance is needed for improving the overall security of software [3].

Requirements engineering is critical to the success of any major development project. Several efforts have been made to prove that requirements engineering defects cost 10 to 200 times as much to correct once fielded than if they were detected during requirements development [4]. It is also proven by the researchers and industry personals that reworking requirements defects on most software development projects costs 40 to 50 percent of total project effort, and the percentage of defects originating during requirements engineering is estimated at more than 50 percent. The total percentage of project budget due to requirements defects is 25 to 40 percent [4]. The need to consider security from the ground up is a fundamental tenet of secure software development. While many development projects produce next versions that build on previous releases, the requirements phase offers the best opportunity to build secure software. Therefore, it is highly desirable to define security requirements during software requirement specification.

2. Security Life Cycle

Applications developed with security in mind are safer than those where security is an afterthought [9]. Researchers and practitioners working in the area of Software Security Engineering have focused on using so-called best practices in the software lifecycle. These are the security-enhanced software development methodology which provides an integrated framework, or in some instances, phase-by-phase guidance for promoting security-enhanced development of software throughout the life cycle phases.

Secure software development is the term largely associated with the process of producing reliable, stable, bug and vulnerability free software. There are a number of ways that this can be undertaken within traditional application development, but the most common procedures involve phased security assessments and reviews that encompass knowledge share; design and implementation assessment and regular security health checks. There are several reasons why organizations choose to follow a secure software development program [27].

Therefore, a Secure Development Process should be integrated with all phases of the software development lifecycle. It ensures that security is a consideration at all stages of software development lifecycle, from requirement analysis through design and implementation to deployment in production environments.

Literature survey reveals that much work has been done in developing such a methodology [14-21]. Following section describes security enhanced software development methodologies proposed by various researchers and practitioners.

2.1 Microsoft's Framework (SDL)

Microsoft developed a trustworthy computing Security Development Life Cycle (SDL) in 2002 during its security pushes. The framework encompasses the addition of a series of security-focused activities and deliverables to each of the phases of Microsoft's software development process. The entire product team focuses on updating the product's threat models, performing code reviews and security testing, and revising documentation. The major objective of the proposed framework was to confirm the validity of the product's security architecture documentation through a focused, intensive effort, uncovering any deviation of the product from that architecture, and identify and remediate any residual security vulnerabilities.

2.2 Oracle's Framework (OSSA)

Oracle Corporation has made an extensive effort in developing a framework to secure software development. Its product development and maintenance process includes a comprehensive set of security assurance mechanisms and processes. The goals of these processes are to improve the strength of security mechanisms and reduce the likelihood of security flaws in products. Collectively these assurance mechanisms and processes are known as Oracle Software Security Assurance (OSSA).

2.3 Comprehensive, Lightweight Application Security Process (CLASP)

John Viega, chief security architect and vice president of McAfee, Inc, made an effort in developing a framework for secured software development in 2004, and developed a Comprehensive, Lightweight Application Security Process (CLASP) to insert security methodologies into each phase of SDLC [14]. CLASP provides a well-organized and structured approach to moving security concerns into the early stages of software development life cycle, whenever possible. CLASP consists of a set of 30 process pieces that can be integrated into any software development process.

2.4 McGraw's Approach

Gary McGraw describes Seven Touch points for Software Security in his book Software Security: Building Security In [22]. This set of software security best practices referred to as touch points. Putting software security into practice requires making some changes to the way organizations build software. These security best practices have their basis in good software engineering and involve explicitly pondering the security situation throughout the software life cycle.

2.5 Team Software Process Approach(TSP)

The SEI's Team Software Process (TSP) provides a framework, a set of processes, and disciplined methods for applying software engineering principles at the team and individual level [23]. TSP for Secure Software Development (TSP-Secure) extends the TSP to focus more directly on the security of software applications. The TSP-Secure framework is a joint effort of the SEI's TSP initiative and CERT program. The principal goal of this framework is to develop a TSP- based method that can predictably produce secure software.

2.6 Secure Software Development Model (SSDM)

It has been observed that producing secure software requires integrating Software Engineering (SE) process with Security Engineering [17]. Simon Adesina Sodiya, a researcher at the Nigerian University of Agriculture developed a Secure Software Development Model (SSDM), which integrates security engineering with software engineering so as to ensure effective production of secure software products[2][12].

2.7 Rational Unified Process-Secure(RUP)

The Rational Unified Process (RUP) is one of the most popular and complete process models being used by developers in recent years. This process model is extended to be used in developing secure software systems by researchers at Amirkabir University of Technology (Tehran Polytechnic) [24][25], and named as RUPSec. The major objective of the RUPSec is to define a software process model in which security requirements are considered in all development phases of a computer-

based system: business modeling, requirements, analysis and design, implementation, and testing

2.8 Security Extension to MBASE

Model-Based Architecting and Software Engineering (MBASE) is a set of guidelines that describe software engineering techniques for the creation and integration of development models for a software project. The models to be integrated extend beyond Product (development) models such as object oriented analysis and design models and traditional requirements models, to include Process models such as lifecycle and risk models, Property models such as cost and schedule, and most notably success models such as business-case analysis and stakeholder win-win.

3. Secure Requirement

Software engineering research has recently focused on improving the modeling abilities in terms of non-functional requirements such as stability [10], performance [11], fault tolerance [12] and security [13]. Unfortunately, security is assumed to be a technical issue and therefore best handled during architecture and design or, better still, during implementation. Since software requirements are often written by non-technical business analysts, this is a common conclusion [8]. An extensive literature survey reveals that a lot of work has already been done on how to effectively elicit, validate, and document software requirements, which may be extended to include security at requirement specification [8].

Security should begin at the requirements level, and must cover both overt functional security and emergent characteristics. One way to cover the emergent security space is to build abuse cases. Similar to use cases, abuse cases describe a system's behavior under attack, providing explicit coverage of what should be protected, from whom, and for how long. Table 1 describes the activities proposed by various researchers and practitioners in the requirement phase of the software development life cycle to come up with the secured requirement.

Table 1: Activities to be carried out for Securing the Requirement Phase

Microsoft SDL	Oracle Secure Software Assurance	CLASP	McGraw's 7 Touch points	TSP-Secure
• Review, recommend and ensures for security team plans; • Identifies critical objectives; • Identify security feature requirements; • Conduct risk analysis of requirements	• Ensure developers are security aware; • Ensure security standards exist and documented; • Ensure security tools and libraries are available.	• Specify operational environment • Perform security analysis of requirements • Detail misuse cases	• Develop security requirement specification • Build abuse cases	• Design security specifications • Identify assets • Develop use cases and abuse cases

4. The Framework

Literature survey reveals that security mechanism should be implemented at the user interface level as well as at the application-under-development level. At the user interface level security mechanism started with an analysis of user's security requirements. In order to accomplish the goal of the theme on security at requirement phase, following objectives are set forth:

- To ensure that users and client applications are identified and identities are properly verified.
- To ensure that users and client applications can only access data and services for which they have been properly authorized.
- To detect intrusion attempts by unauthorized users and client applications.
- To ensure that the unauthorized malicious programs (e.g., viruses) do not infect the application or component.
- To ensure that communications and data are not intentionally corrupted.
- To ensure that parties to interactions with the application or component cannot later repudiate those interactions.
- To ensure that centers and their components and personnel are protected against destruction, damage, theft, or surreptitious replacement (e.g., due to vandalism, sabotage, or terrorism).
- To ensure that system maintenance does not unintentionally disrupt the security mechanisms of the application, component, or center.

Taking into account the objectives discussed above a roadmap or framework for developing secured software specification, an integrated and prescriptive framework SRSF is hereby proposed. SRSF has been attempted to be highly implementable and prescriptive in nature. It has been structured into a hierarchical description including premises, generic guidelines and secured requirement specification process to be followed in order as follows.

4.1 Premises

The following premises have been considered when the proposed framework is being used to develop a secured software requirement specification:

- There is no universally agreed-upon definition for each of high-level security requirement attributes.
- The set of security attributes used in the development of the framework has been defined operationally in the context.
- A common set of features for the desired requirement specification may be used to form the basis for its development.
- The recourse optimization in SDLC depends on the early use of procedure for requirement specification and uncovering of vulnerabilities as far as possible.
- The approach to risk estimate should be more applicable to identifying low security software than the highly secured code.

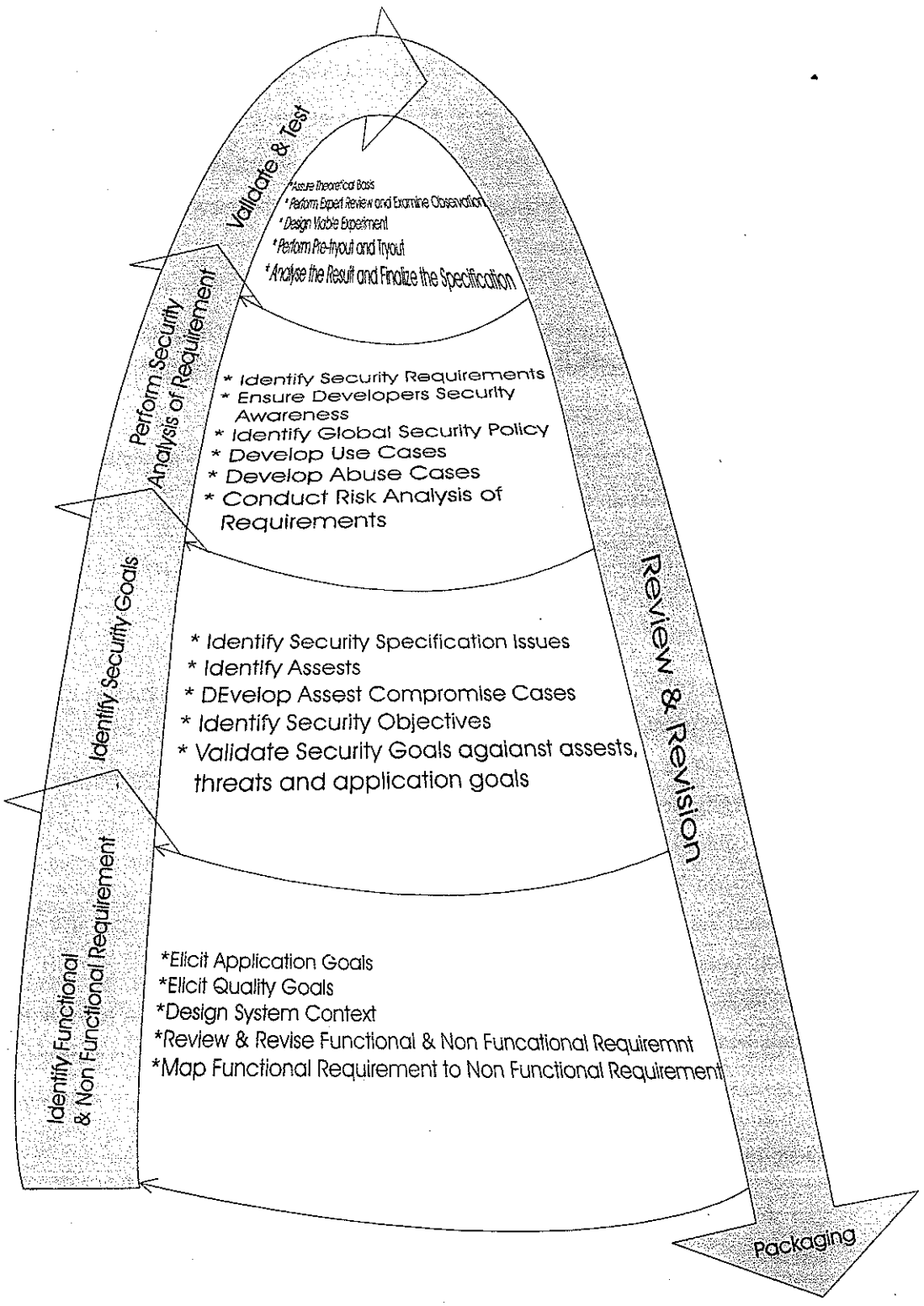

Figure 1 : Secured Requirement Specification Framework (SRSF)

4.2 Guidelines

The guidelines before following the process to develop the secured software specification may be listed as follows;

- Assure compliance/ adherence to collect a generally-accepted set of characteristics that good requirements possess.
- Identify and persist with all the security-specific issues involved in requirements engineering.
- Identify policies and standards as a source of software security requirement.
- Assure to control somehow all the extraneous and intervening factors that may affect the outcome based prediction.

4.3 The Process

The development process of the security requirement is comprised of five phases together with prescriptive steps for each and has been depicted pictorially in SRSF, Fig. 1. Such a framework has been proposed on the basis of integral and basic components for designing secured requirement specification. The first phase starts with the identifying functional and non-functional requirements. Identifying security goals for the desired specification is treated as an important task and has been putforth as a second phase, followed by the phases termed as perform security analysis of requirement, validation and testing, review and revision and packaging. An attempt has been made to symbolically represent the spirit of developing the secured requirement specification make the framework prescriptive in nature followed by a brief description of each of the phases comprising the depicted steps in the special reference to development of the same.

4.3.1 Identify Functional and Non-Functional Requirement

One of the foremost tasks of this comprehensive problem-solving activity is to identify the functional and non-functional requirements. This phase will elicit the application goals and quality goals. System context will be designed. Revision of the identified functional and non-functional requirement will be based on the review of the same. Importance of this phase lies in the fact it serves as the basis for evolving initial set of specifications to subsequent phases of development.

4.3.2 Identify Security Goals

There are five general steps required to identify the security goals including identification of security specification issues, identification of the assets, development of asset compromise cases, identification of the security objectives, and validation of security goals against assets, threats and application goals. The result is a set of security goals, which are validated by ensuring that the business goals remain satisfied.

4.3.3 Perform Security Analysis of Requirement

Before performing a security analysis, one must understand what is to be built. This task should involve reviewing all existing high-level system documentation. If other documentation such as user manuals and architectural documentation exists, it is advisable to review that material as well. This phase comprises of the sub activities including identification of security requirements, ensuring developers security awareness, identification of global security policy, conducting risk analysis of requirement.

4.3.4 Validate & Test

Common wisdom, intuition, speculation and proof of concepts may not be reliable sources of credible knowledge, hence it is necessary to place the specified requirement under testing. Testing is one of the best empirical research strategies, performed through quantitative analysis of experimental data on implementation. Testing is crucial for the success of any software measurement project. This phase comprises of assuring theoretical basis, performing expert review and examination observation, designing viable experiment, performing pre-tryout and tryout, and analyzing the result and finalizing the specification.

4.3.5 Review and Revision

This phase is informal and has been placed as the fifth phase with free-to-enter at any of the earlier phases. Basic idea of such a prescription is to have adequate enough exposure and then turn back for better review, in the light of all the previous phases. However, informal reviews and revisions may be carried out at any of the stages in the requirement specification development process.

4.3.6 Packaging

This phase is the last and conclusive phase of the specification development process. During this phase the developed requirement specification is prepared with the needed accessories to become a ready-to-use product, like any other usable product.

5. Findings And Future Work

A key verification step for the framework described in this paper is the ability to show that the system can satisfy the security requirements. An experimental tryouts and statistical analyses at a large scale with typical representative samples may be needed to standardize the framework, and will be carried out in next phase. It may assist other researchers and practitioners for more developmental activities. This framework may form the basis for the development of better-refined roadmap.

6. Conclusion

Application designed with security in mind is safer than those where security is an afterthought. Traditionally, security issues are first considered during the Design phase of the software development life cycle once the software requirement specification has been frozen. This paper has presented a prescriptive framework for security requirement specification comprising of six steps including identification of functional and non-functional requirement, identifying security goals, performing security analysis of requirement, validation and testing, review and revision, and packaging. The developed framework may be used to ensure the software requirement specification contains the security specifications which helps improve the security of application and reduce the cost of re-work later.

References

[1] U.S. Department of Homeland Security, The National Strategy to Secure Cyberspace, February 2003.

[2] C. Mann, Why Software Is so Bad, Technology Review (July-August 2002).

[3] David P. Gilliam, Thomas L. Wolfe, Josef S. Sherif, *"Software Security Checklist for the Software Life Cycle"*, Proceedings of the Twelfth IEEE International Workshops on Enabling Technologies, Infrastructure for Collaborative Enterprises WETICE'03 IEEE, 2003.

[4] Nancy R. Mead, *"How to Compare the Security Quality Requirements Engineering (SQUARE) Method with Other Methods"*, Technical Note, CMU/SEI-2007-TN-021, August 2007.

[5] Security In The Software Lifecycle, Making Software Development Processes-and Software Produced by Them-More Secure, Department of Homeland Security, DRAFT Version 1.1 - July 2006.

[6] Irvine C. E., Levin. T. E, Nguyen. T. D. and Dinolt. G. W, *"The Trusted Computing Exemplar Project"*, Proceedings of the 2004 IEEE Systems, Man and Cybernetics Information Assurance Workshop, West Point, NY, June 2004, PP 109-115.

[7] Elfriede Dustin, The Secure Software Development Lifecycle, November 11, 2006. Available at: http://www.devsource.com/article2/0,1895,2055993,00.asp

[8] Rudolph Araujo, Security Requirements Engineering: A Road Map, Security Feature (July 2007) Available at : http://www.softwaremag.com/

[9] Roshan Chandran, Security at Software Require ment Specification, August 2004. Available at: http://Palisade.Plynt.Com/Issues/2004aug/Security-Requirements/

[10] Jazayeri. M. *"On Architectural Stability and Evolution"*, Reliable Software Technologies - Ada-Europe 2002, Vienna, Austria, June 17-21 2002. Available at: http://www.infosys.tuwien.ac.at/ Staff/mj/papers/archstab.pdf

[11] *"W. Performance Testing of Distributed Compo nent Architectures"*. S.Beydeda and V.Gruhn (eds), *"Building Quality into COTS Components - Te sting and Debugging"*, 2004. Springer. Denaro, G, P olini, A., & Emmerich. Available at: http://www.cs.ucl.ac.uk/staff/w.emmerich/publications/BeyadaGruhn/PerformanceTesting.pdf.

[12] Guerra. P. A. D. C, Rubira. C. & De Lemos. R. *"A Fault-Tolerant Software Architecture for Component-Based Systems"*, Lecture Notes in Computer Science 2003, 2677, PP 129-149, Springer.

[13] Jurjens, J. *"UMLsec: Extending UML for Secure Systems Development"*, LNCS 2003.

[14] *"Oracle Software Security Assurance"* [web page] (Redwood Shores, CA: Oracle Corporation). Available from: http://www.oracle.com/security/software-security-assurance.html

[15]James W. Over (CMU SEI), *"TSP for Secure Systems Development"* (presentation at CMU SEI, Pittsburgh, PA). Available from : http://www.sei.cmu.edu/tsp/tsp-secure-presentation/

[16]Ivan Flechais, Cecilia Mascolo and M. Angela Sasse, *"Integrating Security and Usability into the Requirements and Design Process"*, Proceedings of the Second International Conference on Global E-Security, London, UK, April 2006. Available at: http://www.softeng.ox.ac.uk/personal/Ivan.Flechais/downloads/icges.pdf

[17] Sodiya, Adesina Simon; Onashoga, Sadia Adebukola, Ajayi & Olutayo Bamidele, *"Towards building secure software systems"*, Proceedings of Issues in Informing Science and Information Technology, Salford, Greater Manchester, England, Vol. 3, PP 25-28, June 2006 Available at : http://informingscience.org/proceedings/InSITE2006/IISITSodi143.pdf.

[18] Mohammad Zulkernine and Sheikh Iqbal Ahamed, *"Software Security Engineering: Toward Unifying Software Engineering and Security Engineering, chap"*, XIV in Enterprise Information Systems Assurance and System Security: Managerial and Technical Issues, Merrill Warkentin and Rayford B. Vaughn, eds., Hershey, PA: Idea Group Publishing, 2006.

[19] Royce, Managing the Development of Large Software Systems, op cit.

[20] Dan Wu, Ivana Naeymi-Rad and Ed Colbert (University of Southern California), *"Extending MBASE to Support the Development of Secure Systems"*, in Proceedings of the Software Process Workshop, Beijing, China, PP 25-27, May 2005. Available at: www.cnsqa.com/cnsqa/jsp/html/spw/download Copy%20of%20MBASE_Sec_Ext_danwu_abstract%5B1%5D.v1.revisedv2.1.pdf

[21] Secure Software Engineering portal.Available at: http://www.secure-software-engineering.com/

[22] Gary McGraw, *"Software Security: Building Security In"*, Addison Wesley, 2006.

[23] Humphrey, Watts. S, *"Winning with Software: An Executive Strategy"* Boston, MA: Addison Wesley, 2002 (ISBN 0201776391, 2002

[24] M. Reza A. Shirazi, P. Jaferian, G. Elahi, H. Baghi and B. Sadeghian, *"RUPSec: An Extension on RUP for Developing Secure Systems-Requirements Discipline"*, Proceedings of World Academy of Science, Engineering and Technology ISSN 1307 - 6884, Vol. 4, PP 208-212, Feb 2005.

[25] *"Software Security Assurance"*, State-of-the-Art Report (SOAR) Information Assurance Technology Analysis Center (IATAC) Data and Analysis Center for Software (DACS) Joint endeavor by IATAC with DACS July 31, 2007.

[26] Thorsten Schneider, *"Secure Software Engineering Processes: Improving the Software Development Life Cycle to Combat Vulnerability", sqp* Vol. *9*, No. *1/© ASQ, PP 4-13, 2006*

[27] Glyn Geoghegan, Secure Development Framework, A Corsaire White Paper, 05 April 2004

Author's Biography

Dr. R. A. Khan is currently working as a Reader in the Department of Information Technology, Babasaheb Bhimrao Ambedkar University (A Central University), Lucknow, UP. His area of interest is Software Security, Software Quality and Software Testing. He has authored two books on software quality and software testing.

Dr. K. Mustafa is currently working as a Reader in the Department of Computer Science, Jamia Millia Islamia New Delhi-India.He has published several papers and articles in Internationals and National Journals. He is the author of books "Software Quality: Concepts and Practices" and "Software Testing: Concepts and Practices".