Security requirements in Software Requirements Engineering

[(1)]P.Mahizharuvi,

Research Scholar,
Dept of MCA, Computer Center,
Madurai Kamaraj University, Madurai.

[(2)] Dr.K.Alagarsamy,

Associate Professor
Dept of MCA, Computer Center,
Madurai Kamaraj University, Madurai.

Abstract

In the last few decades, software projects have encountered major difficulties. Most software engineering projects tend to be late and over budget. Several of the causes of these failures are related to requirements engineering issues such as requirements creep, poorly documented requirements, requirements that were impossible to satisfy, and requirements that failed to meet the needs of the user. Good requirements management practices help improve customer satisfaction, lower the system development costs, and increase the chance of having successful project which in turn give rise to security issue or vulnerable to hackers. Requirements metrics, when incorporated in requirements management, assist in analyzing the quality of requirements and identifying the reasons for software reengineering. In this paper requirements metrics define the output measures of the software process to security of the system. Some recent reports indicate that the project success rate has slightly increased over the last years. This success is a result of defining a process and use of tools like requirements management tools. There are several requirements management tools available in the market. These tools focus on information management aspects of requirements management namely traceability and organization. However, they also offer extended capabilities of collecting requirements metrics. Hence of, we likely to concentrate on the each and every phase of analysis in the life cycle of development of the software in context to security by incorporating to the requirement.

Keywords: Requirements metrics, Analysis phase, Tracing, Requirements.

1. Introduction

In the modern of era technology like Software industry that requirements engineering is critical to the success of any major development project. Security requirements are often identified during the system life cycle. However, the requirements tend to be general mechanisms such as password protection, firewalls, and virus detection tools. Often the security requirements are developed independently

of the rest of the requirements engineering activity and hence are not integrated into the mainstream of the requirements activities. As a result, security requirements that are specific to the system and that provide for protection of essential services and assets are often neglected. The requirements elicitation and analysis that is needed to get a better set of security requirements seldom takes place.

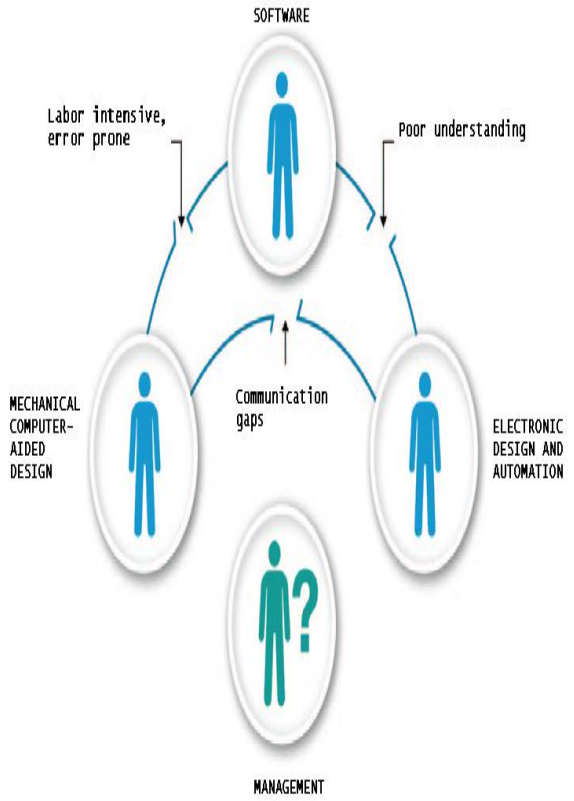

Fig.1.1 Requirement of various domain specific understanding in the development of software.

We rely on smarter products each day, from cell phones and home appliances to cars and airplanes. The software embedded in these products delivers greatly increased functionality, but it also introduces a complex new set of development challenges. The reality is that product manufacturers are now also becoming software companies, leveraging the flexibility of software to introduce new functionality and variation in response to changing client demands. At the same time, these manufacturers are encountering increasing complexity; as they infuse software-based innovation into the development process, they are struggling to understand the effect of the dynamic software development process on the more traditional electrical and mechanical engineering domains. Unfortunately, many companies simply do not have the skill sets, resources or development platforms necessary to build and integrate the intelligent software that is needed, which give rise to complexity and in turn security.

2. Related Work

The reality is that product manufacturers are now also becoming software companies, leveraging the flexibility of software to introduce new functionality and variation in response to changing client demands. At the same time, these manufacturers are encountering increasing complexity; as they infuse software-based innovation into the development process, they are struggling to understand the effect of the dynamic software development process on the more traditional electrical and mechanical engineering domains. Unfortunately, many companies simply do not have the skill sets, resources or development platforms

necessary to build and integrate the intelligent software that is needed.

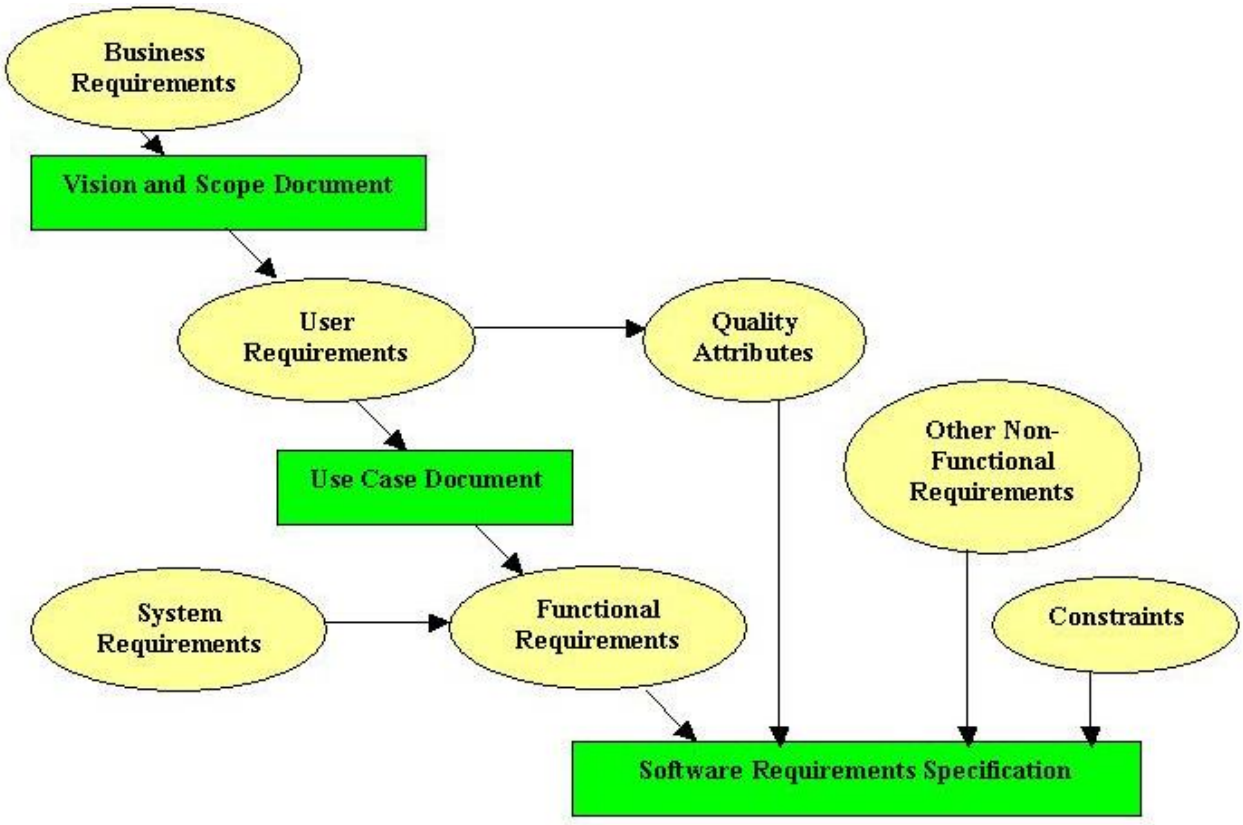

Fig.2.1 Steps involved in the process of making the requirement to the end of quality and security

Requirements should have unique identifiers that can be continuously used throughout the software development process. These are known as the attributes of a requirement. Attributes yield significant information about the state of the system. Based on the attributes of requirements, queries can be made on the status of requirements. These attributes can help to plan, communicate, track the project's activities, and collect requirements metrics throughout the project lifecycle. Some of the requirements attributes are: requirement source, requirement authors, requirement rationale, requirement version number, requirement relative importance, an assignee (to whom the requirement is assigned in the organization), comments, status, time it was created, due date by which the requirement must be provided, method of verification (qualification type to be used to verify that a requirement has been met), relationships to other requirements, and test number (identification number of method of verification).

Software life-cycle models describe phases of the software cycle and the order of execution of those phases. Many models are being adopted by software companies, but most of them have similar patterns. Typically each phase produces deliverables required by the next phase in the life cycle. Requirements are translated into design. Code is produced during the implementation phase and is driven by the design. Code is

finally tested against requirements to ensure quality.

3. Methods

In this method we need for eliciting and prioritizing security requirements in software development projects. Requirements management has evolved as an important aspect of the software development process and has been an area of research in recent years. In broad perspective, requirements management involves information storage, organization, traceability, analysis, visualization, change management and documentation. The agreement forms the basis for estimating, planning, performing, and tracking service level delivery and project progress. Good requirements management practices can lead to higher customer satisfaction, lower system development costs, and increase the chance of having successful software with the highly secure and quality product or software.

People and process: one size doesn't fit all

As of the Embedded software (firmware) is turning up in products across every durable goods sector. The level of intelligence contained in products is growing to the point that it is becoming impossible to imagine them without embedded software. This applies not only to obvious examples such as smart phones, but also to other product areas that are not associated with consumer electronics such as automobiles, aircraft, biomedical devices, or heavy machinery. For instance, today most automobiles are "driven by wire." When the driver steps on the brakes or pushes down on the gas pedal, he or she is interacting with electronic control units (ECUs) containing embedded software that run the power train. Software also governs how avionic controls are organized, or how biomedical devices operate, and the list goes on. It has become virtually impossible to separate the software from the product or vice versa software defines what the product is and how it performs. Hence it assumes to the requirement through the every phase of people with the requirement to end level of the software.

In some sectors, such as aerospace and defense, embedded software development is managed as a systems engineering activity, whereas in other sectors such as automotives, it has only more recently evolved from a self-contained, peripheral activity to a core product development activity that is managed in concert with mechanical, electrical, and controls engineering. As these systems have grown more connected and interdependent, software designers are adopting the same “system-of-systems” design disciplines that their mechanical and electrical engineering counterparts have practiced in systems-intensive sectors. Taking advantage of the rapid innovation associated with software and smart chip development, even for products that have extended, multi-year lead times and complex multi-tiered supply chains, such as any large, complex item of machinery

If we consider the requirements engineering team can be thought of as external consultants, though often the team is composed of one or more internal developers of the project. Which in turn give rise to the stakeholders can expect it to result in the identification, documentation, and inspection of relevant security requirements for the system or software that is being developed.

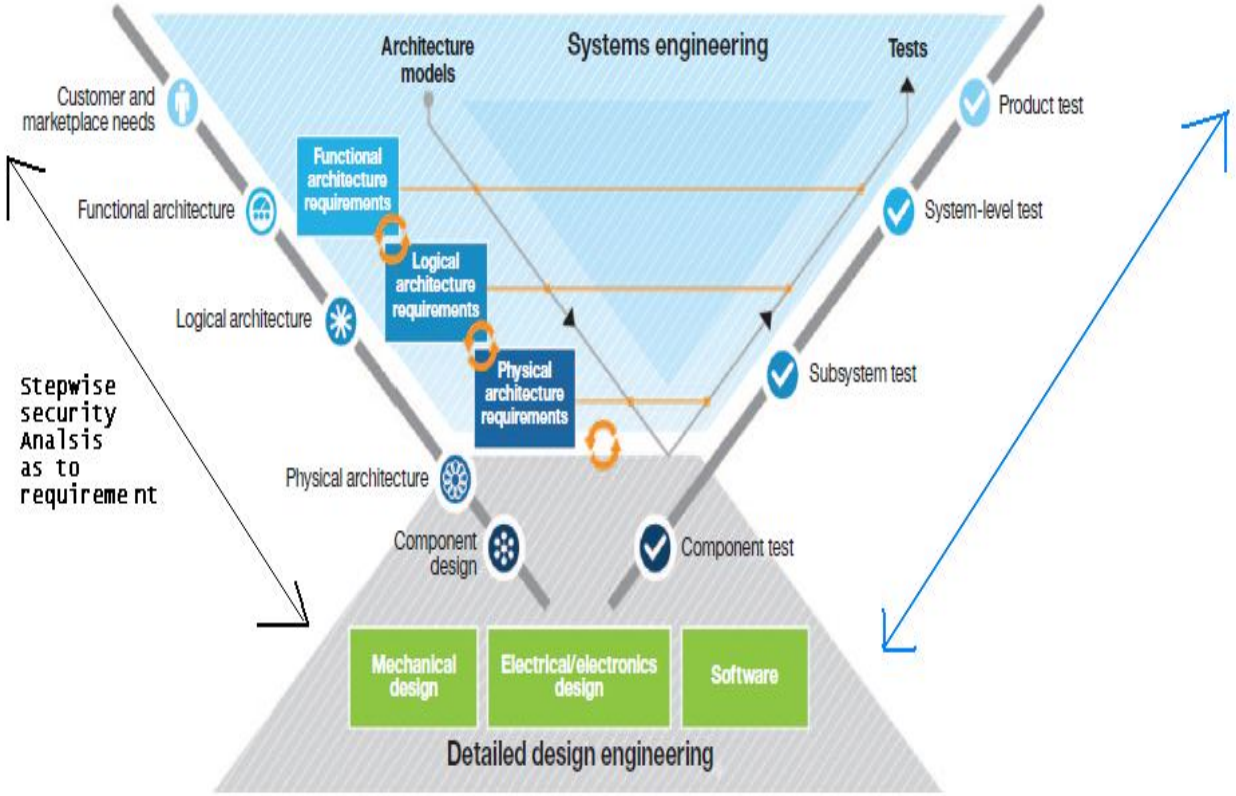

Fig.3.1 Process of analyzing the requirements for ensuring security during the cycle of software.

In the above fig.3.1 the development of today's smarter products involves a complexity of interdependencies. Managing the linkage of interdependent pieces of information including requirements, architecture models, tests and change requests with each other and with detailed product design data throughout the development cycle which rises to the concept of co-operating the requirement through the security phase. In addition to helping ensure that each product requirement is properly implemented, traceability facilitates compliance with regulations and standards, keeps teams focused on project goals, and helps engineers avoid overlooking stakeholder demands and give a quality product with all the security. In short, it enables product developers to make sure that they build the product they set out to build not just a product that "works

In this paper, to succeed in today's fast-paced and competitive marketplace, companies must cultivate efficient and effective ways for their different engineering and business domains to communicate and collaborate. Software is not only changing the nature of products, but also the nature of systems engineering. Across many product industry sectors, software is arguably accounting for the lion's share of innovation. Embedded software provides the intelligence in today's smart products. However, it also adds more design

interdependencies, thereby increasing product complexity. The challenge in overcoming complexity is that most product organizations are highly productive by engineering discipline and tool adoption.

Performance Analysis w.r.t Existing System

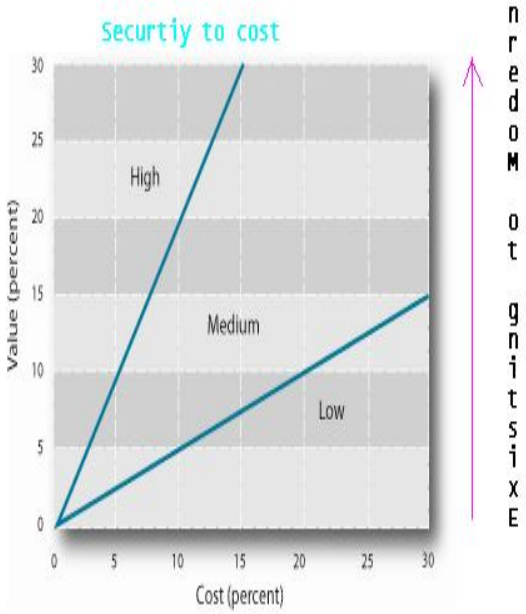

Fig.3.2 Graph showing the security with cost and Classical to modern system.

In the above fig.3.2, the graph is showing the nature of the low, medium and high with the cost in the horizontal i.e. X-axis while in the vertical axis showing the percentage of security value. Hence the performance increase with the cost is directly proportional to security and vice versa where as it also shows the existing where cost was less as compared to the modern software system, where cline wants both quality and security at a time. Hence, we need quality irrespective of cost is the main theme of the modern software.

The systems engineering discipline helps to overcome these by focusing on the big picture, ensuring that the design is properly supported through the lifecycle of the project giving to security analysis at each phase. Yet all too often, software engineering workflows, reinforced by the proliferation of disconnected software tooling, compounds the challenge that systems engineers face. For many product engineering organizations, the growth of software content is raising the urgency to ingrain systems engineering mentality and practice across all engineering teams.

4. Conclusion

Today's electric, gas, water utilities and day to day need software applications and technology to serve as a robust springboard from which to meet the challenges of the future. Every industry is different. Products across different industries are different in size, complexity, and degree of functional interdependency. The reality is that most product engineering organizations are highly diverse, not only in technical disciplines, but also processes and tool preferences. But, all where we arise a question mark of security, which gives rise to quality of the software. Hence in this paper; we have concentrated on the concept of security through the analysis of requirements. In the last phase of conclusion it although a complex, multifaceted issue, achieving best-in-class results can be broken down into key challenges that can be addressed through formalized processes portfolio, product and project and performance management.

5. Reference

[1] N. R. Mead, E. Hough, and T. Stehney, Security Quality Requirements Engineering(SQUARE) Methodology, Software Engineering Institute, Carnegie MellonUniv., Pittsburgh,PA,Tech.Rep.CMU/SEI-2005-TR-009,2005. http://www.sei.cmu.edu/publications/documents/05.reports/05tr009.html

[2] AnalystPro Software Development, Requirements Management (2000). Retrieved from the World Wide Web http://www.analysttool.com/requirements.htm.

[3] Waterfall model,‖ Wikipedia. http://en.wikipedia.org/wiki/Waterfall_Model

[4] B. Boehm, ― A Spiral Model of Software Development and Enhancement,‖ IEEE Computer, vol. 21, no. 5, pp. 61-72, May 1988.

[5] P. Kruchten, the Rational Unified Process: An Introduction, 3rd ed. Boston: Addison-Wesley, 2003.

[6] Rational Software, ―Rational Unified Process: Best Practices for Software Development Teams,‖ Rational Software, White Paper TP026B, Rev 11/01, 2001. http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_best practices_TP026B.pdf

[7] S. W. Ambler, ―A Manager's Introduction to Rational Unified Process, 2005. http://www.ambysoft.com/downloads/managersIntroToRUP.pdf

[8] Principles behind the Agile Manifesto,‖ 2001. http://agilemanifesto.org/principles.html

[9] Asbrand D. (1998). IT Metrics For Success, Information Week Online, News in Review issue of August 17, 1998.

[10] ASEA Report (1994). Analysis of Automated Requirements Management Capabilities developed in support of Advanced System Engineering Automation (ASEA), CSC-2.7 Requirements/Design Manager (Contract N0. F30602-93-C-0123), Prepared for Rome Laboratory, Air Force Materiel Command C3CB 525 Brooks Rd. Griffiss AFB, NY 13441 by Software Productivity Solutions, Inc. 122 4th Avenue Indalantic, FL. 32903.

[11] Bahill, A.T. and Dean, F.(1999). Discovering system requirements, Chapter 4 in the Handbook of Systems Engineering and Management, A.P. Sage and W.B. Rouse (Eds), John Wiley & Sons, 175-220.