Quality, Category and Characteristics of Secured Requirements

Mohammad Ubaidullah Bokhari
Associate Professor,
Dept. of Computer Science
Aligarh Muslim University, India

Mahtab Alam
Research Scholar, Computer Science,
Dept. of Computer Science
Mewar University, Chitorgarh, India

ABSTRACT: ***The earliest system architectures have a significant impact on the quality of software and its adverse affects compounded in later stages. Despite the availability of many methodologies and models of SDLC, there is always a requirement creep, due to the increasing challenges of the vulnerabilities of a system. Managing requirements stability in a software project is a science and art with a number of problems to face for delivering good software within a stipulated time period and a sanctioned budget. Now a day a system protection is going to a big challenge for information system security engineer. The security requirements differ not one from project to project but also from user to user. Often, the number of assets potentially at risk outweighs the resources available to manage them. It is therefore extremely important to know where to apply available resources to mitigate risk in a cost-effective and efficient manner.***

KEYWORDS: ***Security, Usability, Requirement, Confidentiality, SDLC (Software Development Life Cycle)***

I. INTRODUCTION

It is considered that requirements are the foundation stone on which an entire software product is built. It is further presumed that verification and validation are needed to assure that the desired functionality, embodied in the total set of requirements, is ultimately and correctly delivered. The belief is that correct, complete, and testable software requirements are critical. The success of a project—whether measured in functional or financial terms—can be directly related to the quality of the requirements [20].

Collecting, analyzing and managing requirements are the major challenges for a project manager faces in a project. Healthy and rigorous requirement management process is one of the stepping-stones of success of a project. Almost all software projects arise out of a business problem. Requirements gathering and analysis try to identify the business problem to be solved and probable characteristic a software product needs to have as a solution to the business problem. In a recent survey it is observed that about 71% of the software is not completed due to poor requirement [2].

Perhaps 50% of all damage caused to information system comes from authorized personnel who are either untrained or incompetent. Another 25% or so of the damage seems to come from physical factors such as fire, water, and bad power. Maybe 20% of the damage comes from dishonest and disgruntled employees. Computer viruses cause another few percent, and may be 5 or 10% of the damage comes from external attacks [7].

Documenting requirements are vital throughout the requirements engineering process to ensure correct and complete representation of requirements, and an adequate basis for the future developments of the system. The requirements engineering is one of the challenging job to incorporate multiple viewpoints from different users and document communicated ideas in an unambiguous and comprehensive way. Two types of requirements representation are (1) natural language and (2) formal specifications. Expressing information in a natural language is easy to comprehend for all parties involved in the development project, but the information expressed can be ambiguous, resulting in multiple interpretations of a particular document. On the other hand, a formal specification overcomes the ambiguities and correctness verification.

However, formal techniques are complicated to apply for developers and difficult to understand for the customers [Sutcliffe 2002]. Given the two approaches of representing requirements, natural language is most commonly used. Building secure software systems, a lot has to be done. Security techniques have to be implemented in all the stages of the software engineering. Devanbu and Stubblebine

(2000) stated that security concerns must inform every phase of software development, from requirements to design, implementation, testing and deployment. This is necessary because software developer might unknowingly inject defects in all stages of the development [17].

In generally accepted software development practice, requirements analysis is supposed to happen before design, and design is supposed to happen before coding and testing. Some kind of document is supposed to come out of requirements analysis. The IEEE 830-1993 Recommended Practice for Software Requirements Specifications includes a list of quality attributes for requirements: correct, complete, unambiguous, consistent, and ranked for importance, verifiable, modifiable, and traceable. The specification is supposed to guide the engineering activities that follow [1].

In the past, the security of software or an information system has been treated as a qualitative attribute. Recently however, several researchers have suggested that security be treated as a quantifiable QoS (Quality-of-Software) attribute [10].

II. REQUIREMENTS

Requirement is a documentation of what a particular product or system should be or do. Requirements are used as inputs into the design phase of product development. Requirement phase is one of the major phases for the success of a software development life cycle (SDLC). Most of the software has not been completed just because of negligence of requirement phase. Due to increasing vulnerabilities of a system it is one of the challenging tasks for the software engineers to develop a secured system. Introducing security concept in the requirement phase curtails the efforts and cost of the software SDLC. Developing the system security context involves defining system premises and interfaces with SE, allocating security functions to target or external systems, and identifying data flows between the target and external systems and protection needs associated with those flows [19]. Requirement engineering mainly concerned with

- Requirement Elicitation
- Requirement Analysis
- Requirement Specification
- Requirement Verification
- Requirement Management

A. CATEGORY OF REQUIREMENTS

Problems that are not found until testing are at least 14 times more costly to fix than if the problem was found in the requirement phase [15]. Generally the requirements fit into one of three categories [13]:

a. BUSINESS REQUIREMENT

Generally, business requirements will focus on the demands from the customer and demands that are internal to the organization. As a result, business requirements may be somewhat unstructured.. Be aware that individual projects may have specific requirements that are not covered by the global policy or are in conflict with it. Since customers often are not adequately security-aware, one should not expect to derive an exemplary set of security requirements through customer interaction. It is recommended to explicitly bring up issues that may become important to system users after deployment, particularly [3]:

- Preferred authentication solutions;
- Preferred confidentiality solutions for network traffic;
- Preferred confidentiality solutions for long-term storage of key data; and
- Privacy concerns (particularly for personal data).

Business Requirements gathering represents the non-technical requirements of the system. These business requirements provide over-arching guidance to the SDLC team [5]. The business requirement is mainly concerned with the top level management of an organization whose main goal and objectives are as under:

- Abstract threat versus specific threat (e.g. Y2K)
- Return on Investment (ROI)
- Fuelled by compliance requirements (?)
- Must be a clear linkage to business AND security risk
- Prioritize the self assessment process according to critical assets
- Business Impact Analysis
- Primary assets versus Secondary assets

b. SECURITY REQUIREMENTS

The Security requirement is mainly concerned with the requirements which describe the job that user must perform, which can be thebest sub factored as follows:

- **Technical Requirements**

  Access Control
  Data Authentication
  Boundary Protection

- **Management Requirements**

  Security Policy & Organization

- **Operational Requirements**

  Security Procedures

- **Cross-cutting Requirements**

  Many of the above apply to all components within the system e.g. Security Policy, Configuration Management, Audit, etc

c. USABILITY REQUIREMENTS

Usability studies of security system deal not only with software aspects but also with systematic aspects. Moreover, they require the evaluation of interaction between different quality attributes like usability and security (an ISO sub characteristics of functionality) [14]. The usability requirement is concerned with the quality of the product which includes according to ISO 9126 standards following three attributes:

- **Learnability:** The effort required to learn the use of the security system.
- **Attractiveness:** The extent that a security system is attractive to the user.
- **Operability:** The effort required to properly carry out the assigned tasks.

The Operability is further broken down into following attributes:

- **Install ability:** The effort required to install the software.
- **User friendliness:** The effort required to use the system with ease.
- **Mobility:** The effort required to use the system for different computers.
- **Security interaction:** The effort required to enhance security without inducing dangerous error.

Application Designed with security in mind are safer than those where security is an afterthought. Generally, security issues are first considered during the design phase of the SDLC once the software requirement specification (SRS) has been frozen [21]. One of the major challenges the requirement engineer facing during requirements gathering is the testability of a requirement. The users are actually not familiar with the software process and even they are not aware what they actually needs as an end product. The use of natural language in requirements documentation seems to persist with increasingly important roles and in unlikely to be replaced. [16]. Requirements that user expects the developer to contractually satisfy must be understood by both parties and natural language seem to be the only mode of communication, since users are not trained to understood the logical notation. Seldom, a few customers come up with requirements that are not testable. To determine the testability of a requirement the requirement engineers have been suggested to ask the following questions at the time of questionnaire session [15]:

- Can it be further furcated into multiple requirements?
- If yes then the set of requirements are not testable.
- Can we define the acceptance criteria for this requirement?
- If the answer is no then this requirement is not testable.
- Clearly state the assumption you have made on this requirement. Check if the assumption is conflict with any other Assumption/requirement made so far.
- If yes then the set of requirements are not testable
- Is this requirement clashing with any other requirement?
- If yes then the set of requirements are not testable.

You will need to revisit the requirement again and again. Since most of the answers coming from the customer's side are against the testability of the

requirement phase that means it is still hard to assess the quality requirements. It is also advisable for a requirement engineers to keep the following tips in mind while gathering a requirement of a quality project.

B. TIPS OF GOOD REQUIREMENTS

The security engineering committee has introduced a number of techniques for managing, securing and protecting a information system, most of which have focused primarily on design and implementation issues. However, what the security committee has identified is important, but still lacking is a precise notion of security requirements. Security requirements are often specified in abstract statements for which the satisfaction criteria are not clear, or informal assertion that are too restrictive and intuitive to use [9]. To address these criteria the requirement engineers are given some tips to collect and analyze the requirement provided by the stakeholders [15].

- Number all your requirements uniquely. This will help considerably in establishing traceability.
- Identify the pre and post conditions for the requirement.This will help in test data setup and documenting the test results.
- Generate a Business workflow of the requirement. This will help the tester in understanding the system.
- Any information about existing functionality, which is not changing, should be clearly distinguished from the changing requirements. It is a good idea to put these in a new section called Regression test scenario section.
- Get an independent review done to confirm your understanding of the requirements.

List of all security functions, both Approved and non-Approved, and specification of all modes of operation, both Approved and non-Approved. Block diagram depicting all of the major hardware components including any microprocessors, input/output buffers, plaintext/cipher text buffers, control buffers, key storage, working memory, and program memory. Specification of all security-related information, including secret and private cryptographic keys (both plaintext and encrypted), authentication data (e.g., passwords, PINs), CSPs, and other protected information (e.g., audited events, audit data) whose disclosure or modification can compromise the security of the cryptographic module. Specification of a security policy including the rules derived from the requirements of this standard and the rules derived from any additional requirements imposed by the vendor).

The use of natural language to communicate between user and requirement engineers to describe the requirements of a complex system suffer from at least three discrepancies i) ambiguity and impreciseness, ii) in-accuracy, and iii) inconsistency and incompleteness.

C. IDEAL SOFTWARE REQUIREMENST ENGINEERING PROCESS

Experts' requirements engineers sort out recurring set of activities that characterize this engineering process. These activities, which are generally construed as necessary in order to produce a reliable, high quality, trustworthy system, include [11]:

a. REQUIRENMENTS ELICITATION

Identifies system stakeholders, stakeholder goals, needs, and expectations, and system boundaries. The technique to gather eliciting requirements may include questionnaire surveys, interviews, documentation review, or joint application development (JAD) team meetings etc. Lauesen identifies that the following intermediate documents need to be produced before eliciting requirements [Lauesen 2002]:

- Description of the present work domain business activities in the application domain.
- A list of problems in the domain identifying existing problems in the application domain.
- A list of business goals and critical issues – high level reasons for developing a new system. (For more information on goals, refer to [Cooper 1996].)
- Ideas for the large scale structure of the future system – a vision of how people will function when the software product is used in its intended environment.

b. REQUIREMENTS SPECIFICATION

Main attention of the specifying requirements for both functional and non- functional software

requirements are to firstly, specify the functional requirements of operational domain problems, which includes states, events (input events, output events, process execution flags or signals, error detection, and exception handling triggers), and system data (identification of data objects, data types, data sources, screen-display etc).

Functional requirements should also specify system data flow through system or subsystem states as controlled or synchronized by events. In contrast, non-functional requirements specify goals, capabilities, and constraints that situate the functional system within some context of operation. This can involve identifying an enterprise model, problem domain model, system model type, and data model type. Although the exact contents of a requirements specification vary from situation to situation, several types of information are included in most requirements specifications. In addition to functional and nonfunctional requirements statements, the SRS contains a variety of technical and non-technical text, such as [Hull 2002]:

- Description of the environment and objectives of the system
- Definition of the scope of the requirements
- Stakeholders description
- Supplementary models used in deriving the requirements, such as DFD and ER diagrams.

c. REQUIREMENTS ANALYZING

Focus on the internal consistency, completeness, or correctness of a specification. It does not check the requirements are externally correct or not. That determination may result from observing a visual animation of the specification during operational execution (a simulation). More sophisticated analyses may check for reachability, termination, live-lock and dead-lock, and safety in the modeled system. Requirements analysis representations can be divided into several categories – functions, data, and nonfunctional requirements [Lauesen 2002].

d. FUNCTIONS/FEATURES OF THE SYSTEM

- Dataflow diagram.
- Viewpoint form – A requirements representation based on the concept of viewpoints.
- Feature requirements – Statements of the functionalities saying "the product shall record/show/compute…"
- Scenario – A case story illustrating user tasks, or a specific case to be tested.
- Decision table – Lists all the possible conditions and appropriate actions in a tabular format.
- Decision tree – Provides a tree structure to lay out options and investigate the possible outcomes of choosing those options.
- State diagram (Finite state machine) – Depicts how a certain entity changes its states as a result of various events.
- Object-oriented requirements analysis formats – The conventional requirements generation approach (functional paradigm) models the system as a group of.

e. DATA REQUIREMENTS

- Data dictionary – A textual data description structured systematically.
- Data model (Entity Relationship diagram) – A block diagram describing data stored in the system and the relationships between the data [Hull 2002].
- Data expressions – Compact formulas that describe data sequences.

f. NON-FUNCTIONAL REQUIREMENTS

- Quality factors – Lists non-functional requirements (quality factors) and uses them as checklist in determining which non-functional requirements need to be addressed for the system.
- Quality grid – Represents non-functional requirements in the form of a grid.
- Goals – This approach treats non-functional requirements as goals that might conflict.

g. REQUIREMENTS VERIFICATION

Engages domain experts to assess feasibility of modeled system solution, as well as to identify realizable, plausible, and implausible system requirements. Systematic techniques for inspecting requirements to assess system usability and feasibility may also be employed. As a result of validation, the requirements engineer can better calibrate customer expectations about what can be developed.

h. CHARACTERISTICS OF GOOD REQUIREMENTS

As described above, a list of system requirements contains a complete description of the important requirements for a product design. From this list, subsequent design decisions can be based. Of course, if one is to place their trust in them, we must assume that all the requirements in such a list are good. This raises the question, 'How does one differentiate between good requirements and those that are not so good?' It turns out that good requirements have the following essential qualities:

- **A good requirement contains one idea**. If a requirement is found to contain more than one idea then it should be broken into two or more new requirements.
- **A good requirement is clear**; that is, the idea contained within it is not open to interpretation. If any aspects of a requirement are open to interpretation then the designer should consult the relevant parties and clarify the statement.
- **Requirements should remain as general as possible.** This ensures that the scope of the design is not unnecessarily limited.
- **A good requirement is easily verifiable,** that is, at the end of the design process it is possible to check whether the requirement has been met.

III. BASIC METRICS FOR REQUIREMENTS MANAGEMENT

The identified metrics are represent a simple set measurements that projects new to requirements, management can choose from, depending on their needs. Regardless of the software process being followed, every project documents requirements in some form using a variety of artifacts. However, many projects lack even the simplest requirements-related measurements to help manage the project to successful completion, avoid rework, control scope, or manage change during the project.

The identified metrics can be applied to virtually any software development effort. The identified metrics fall into three categories. First, there are metrics that help assess the goodness of the requirements process itself. Second, there are metrics that provide the project manager and project leaders with objective information to help guide the project to successful completion. Finally, there are metrics to help assess the impact requirements management is having on overall project costs and product quality. Some of these metrics can be gathered from requirements management tools; others need to be gathered from the tools used for managing change requests or tracking defects [1].

A. NUMBER OF REQUIREMENTS BY OWNER/RESPONSIBLE PERSON

These metrics indicate the workload of various people on the project. The information can be used by the project manager to determine whether the project could benefit by shifting some of the workload. It can also be used to determine whether the right people are assigned to specifying or implementing the most important requirements.

B. NUMBER OF REQUIREMENTS BY STATUS/TOTAL NUMBER OF REQUIREMENTS

Even when a project is running well, measurement is not only useful but necessary. "You cannot control what you cannot measure (Demarco 1982). Measurement is the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined rules [12]. The set of requirements for a project are constantly in flux, especially during the earlier phases of development. Some requirements may have been approved and will be incorporated into the product being developed.

Others may have been proposed by one or more stakeholders, but there is not yet agreement about whether they will be included in the product. Other requirements will be in various stages of development (e.g., being worked on, coding is complete, validated by testing, completed). Still others may be on hold pending clarification of certain issues.

Having a clear understanding of exactly what the state of each requirement is and where it is in the development process enables the project manager to effectively manage the project, avoid requirements and scope creep, and take corrective actions to deliver the project on time and within budget while assuring that all the critical business needs are satisfied.

C. FUNCTIONAL REQUIREMENTS ALLOCATED TO A PROJECT RELEASE OR ITERATION

Understanding exactly how many requirements, and which specific ones, are allocated to a release or iteration allows the project manager to successfully deliver the project on time with the most critical functionality. Making this information available to the team keeps everyone focused. It can also shorten development cycles by allowing the QA team to get an early start with test planning, test development, and establishing the appropriate test environment, while the code is being developed.

D. REQUIREMENTS GROWTH OVER TIME

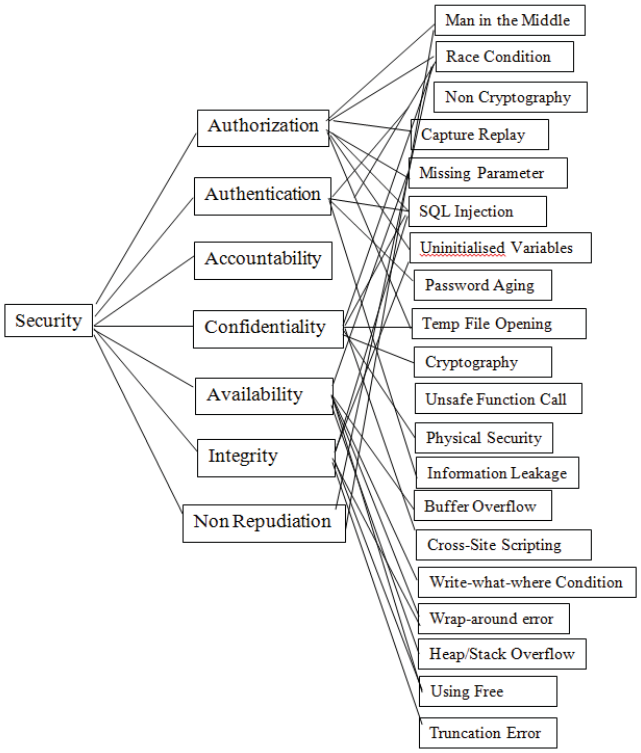
Early in the Lifecycle, this metric can help the project manager to determine whether adequate progress is being made gathering and specifying the requirements. As the project progresses, unusual growth can be an indicator of scope creep. It may also be an indicator that there are opportunities to improve the way in which requirements are elicited and documented.

E. NUMBER OF REQUIREMENTS COMPLETED

This is an objective indicator of the number of requirements implemented, tested, and validated to date. Trends of requirements completed over time can also help measure how quickly the project is moving toward completion (e.g., the velocity) and whether the project team can include more functionality or is potentially overcommitted.

IV. CONCLUSIONS AND FUTURE WORK

Software products are entirely depends upon software design which provides the blueprints on which the success of entire application based. A good design is always done with the help of quality requirement. Majority of the software fails due to poor requirement. In this paper we proposed some attributes of quality requirements and if the requirements are gain keeping this attributes in mind it definitely reduce the failure rate of the application. In future we will develop a security metrics for the requirements phase which will tell us the degree of security of the application.



V. REFERENCES

[1] Bach J. (1999) **"**Reframing Requirements Analysis", IEEE Computer Society.

[2] CEO, IT Magazine, November, 2005.

[3] Comprehensive Lightweight Application Security Process CLASP (2006), Version 2. Secure Software, Inc.

[4] DOD MIL-STD-490A, Specification Practices, June 4, 1985

[5] Hendrick (2003), Data Systems Analysts, Inc.

[6] IEEE Std (1993), Recommended Practice for Software Requirements Specifications, December 2.

[7] Kabay, M. E. (2002) "What's Important for Information Security: A Manager's Guide", Norwich University, Northfield.

[8] Kolde, C. (2004) "Basic metrics for requirements management", A Borland White Paper, March.

[9] Lin, L., Nuseibeh, B., Ince, D., Jackson, M. and Moffett, J. (2003) "Introducing Abuse Frames for Analysing Security Requirements", IEEE International Requirements Engineering Conference.

[10] Madan, B. B., Trivedi, K. S. (2002) "Security modeling and quantification of intrusion tolerant systems**",** Fast Abstract ISSRE, Chillarge Press.

[11] Mullins, J. (2001) Requirement Analysis – Characteristics of Good Requirements. The University of Queensland, MECH4551 – System Design Projects – Semester 2.

[12] Norman, E. F., Pfleeger, S. L. (2004) "Software Metrics A Rigorous and Practical Approach", Thomson Asia Pte Ltd.

[13] Patrica L. F (2004) "Software Requirement: The Requirement Set", Addison Wesley, 2004

[14] Piazzalunga, U., Salvaneschi, P. (2006) "How to Test Usability of Security Sensitive Systems", SQP Vo. 8, No. 3.

[15] Rosenberg, L., Hyatt, L., Hammer, T., Huffman, L. and Wilson, W. (1998) "Testing Metrics for Requirement Quality", NASA.

[16] Scacchi, W. (2001) "Understanding the Requirements for Developing Open Source Software Systems, IEE Proceedings – Software.

[17] Sodiya, A. S., Onashoga, S. A. and Ajayi, O. B. (2006) "Towards Building Secure Software Systems", Issues in Informing Science and Information Technology Volume 3.

[18] Sommerville, I. (1992) Software Engineering, Fourth Edition, Addison-Wesley Publishing Company, Wokingham, England.

[19] The information System Security Engineering Process, Chapter-3, IATF Release 3.1-September 2002.

[20] Wilson, W., Rosenberg, L., and Hyatt, L. (1996) Automated quality analysis of natural language requirements specifications. In Pacific Northwest Software Quality Conference, October 1996.

[21] www.palisade.plynt.com/issues/2004/aug/security-requuirements/8.