International Journal of Engineering Research and General Science Volume 3, Issue 3, May-June, 2015
ISSN 2091-2730

An Indispensable part of System Development: The Requirement Management

[1]Akshara B. Dave, [2]Dr. Sanjay M. Shah
[1]Research Scholar, CharuSAT University, India, akshara.dave@gmai.com, 9879108751

Abstract— Software development life Cycle (SDLC) is an essential activity in terms of Software development process. In this paper, we have elaborated the term "Requirement management". Different natures, types of stakeholders, and their form of documents given at time of discussion are the questions whose answers are tried to be given through this paper. We describe the specific behavior which shapes an individual's imagination towards a developed system. The major road map is management of requirements throughout system development. Generally, this paper focuses on such points that make the term "Requirement Management" a basic and in depth process hence, an essential portion.

Keywords— Software Engineering, Requirement specification, Requirement management, Requirement Traceability, Traceability matrix, Changes in requirement, Challanges

1. INTRODUCTION

The goal of the requirements engineering process is to create and maintain a system requirements document. The overall process includes four higher level requirement engineering sub-processes. There are concerned with assessing whether the system is useful to the business(Feasibility studies), discovering requirements(elicitation and analysis), converting these requirements into some standard form(Specification), and checking that requirements actually define the system that stack holder want(validation).

Requirement Traceability is valued as main task in an increasing number of methodologies for requirement engineering [RE][5]. This task is reflected by the various systems that have been developed in different area. [4]Though many advances, Requirement Traceability covers a huge questioning part from software industry. We call it as problem specification [3].

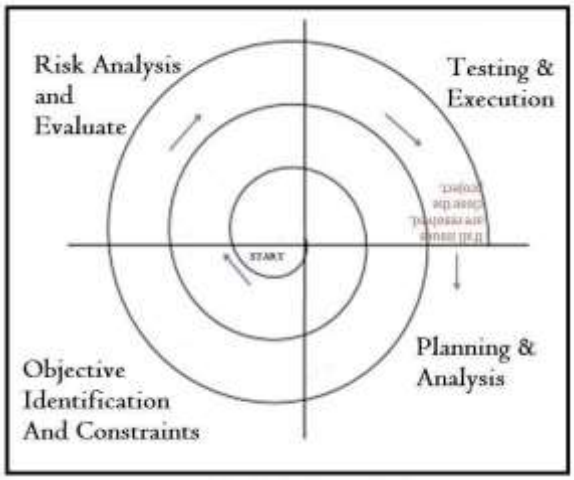

Figure 1: Spiral Model of requirement Engineering

2: CHALLENGES OF ROLES IN SE

In a knowledge industry like software, people play a crucial role and are considered as valuable assets. This requires a broader knowledge[2]

1.	***Not all roles with same title are similar:*** Several meanings for the same role in same/different organizations
2	***Competencies are not clear to both organizations as well as individuals:*** The knowledge, skills and attitude related to SE roles are often subjective and unclear
3	***Skills evolve and roles evolve too:*** Continuous evolution of technologies, skills makes roles a moving target
4	***Crosscutting and Implicit roles*** make it hard for people as they are not explicitly documented
5	***Changing perspectives:*** There are several continuously evolving perspectives of roles from people, organization and other views.
6	***Environment Factors:*** Implications of changes in culture, process, technology and organization on roles is not clear and often puts people under pressure
7	***C4:*** Global interaction between various stakeholders is difficult because of culture, language and other issues
8	***Non-SE Professionals and Diversity:*** There is no clear separation and mapping of activities for SE and non-SE professionals.

Table 1: Challenges of roles in SE

There are mainly two types of requirements:

Functional requirements. (2) Non functional requirements

3.1: Functional Requirement

The functional requirements for a system describe what the system should do. These requirements depend on the type of software being developed, the expected users of the software and the general approach taken by the organization when writing requirements. When expressed as user requirements, the requirements are usually described in a fairly abstract way. However, functional system requirements describe the system function in detail, its inputs and outputs, exceptions, and so on. Functional requirements for a software system may be expressed in a number of ways. For example, here are examples of functional requirements for a university library system(LIBONLINE), used by students and faculty to order books and documents from other libraries.

The user shall be able to search either all of the initial set of databases or select a subset from it.

1. The system shall provide appropriate viewers for the user to read documents in the document store.
2. Every order shall be allocated a unique identifier(ORDER_ID), which the user shall be able to copy to the account's permanent storage area.

These functional user requirements define specific facilities to be provided by the system. These have been taken from the user requirements document, and they illustrate that functional requirements may be written at different levels of detail.

LIBONLINE system is a single interface to a range of article databases. It allows users to download copies of published articles in magazines, news papers and scientific journals.

Imprecision in the requirements specification is the cause of many software engineering problems. It is natural for a system developer to interpret an ambiguous requirement to simplify its implementation. Often, however this is not the customer wants. New requirements have to be established and changes made to the system. Of course, this delays system delivery and increases costs.

3.2: Non-functional Requirements

Non functional requirements are requirements that are not directly concerned with the specific functions delivered by the system. They may relate to emergent system properties such as reliability, response time and store occupancy. Alternatively, they may define constraints on the system such as the capabilities of I/O devices and the data representations used in system interfaces.

Non-functional requirements specify the emergent properties of the system. Therefore, they may specify system performance, security, availability, and other emergent properties. This means that they are often more critical than individual functional requirements. Failing to meet a non-functional requirement can mean that the whole system is unusable.

Non-functional requirements arise through user needs, because of budget constraint, because of organizational policies, because of the need of interoperability with other software or hardware systems, or because of external factors such as safety regulations or privacy legislation. Such requirements may come from required characteristics of the software product requirements or from external sources.

Types of non functional requirements:

Fig. 2. Product requirements.
Fig. 3. Organizational requirements.
Fig. 4. External requirements.

Next theory shows examples taken from the library System LIBONLINE whose user requirements are already discussed in functional requirements section.

Product Requirement: The user interface for LIBONLINE shall be implemented as simple HTML without frames or java applets.

Organizational Requirement: the system development process and deliverable documents shall confirm to the process and deliverable defined in XYZCo-SP-STAN-95(Company standard process).

External requirement: The system shall not disclose any personal information about system users apart from their name and library reference number to the library stuff who use the system.

4: REQUIREMENT MANAGEMENT IN SOFTWARE DEVELOPMENT (SD) PROJECTS

The survey said there number of problems arising in requirements management in global software development projects is shown in Fig.1. The surveys gives 44% to 80% of all defects are found in the requirements phase [1].

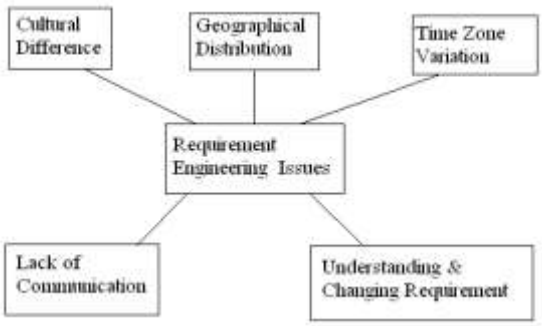

Figure: 2 Requirement Management issues in SD projects.

Many software projects fail due to problems with requirements. As requirement management is the first and essential step in SDLC, if it fails, it will lead to decrement the performance on further steps.

5. TASKS DONE BY REQUIREMENT ANALYSITS[6]

A Requirement analyst must answer the questions start from general words like Who, What, Where, When, Why, How and which.

Who have problem regarding system? The new system development can have impact upon many people. Often, they will have competing needs, and different perceptions of the problem. The requirements analyst is responsible for identifying and characterizing all the relevant Stakeholders.

What are the factors that prevents of solving? Requirements engineering is crucial for risk management – requirements engineers must balance the selection and scoping of the problem with the feasibility of implementing a solution within the given constraints. The requirements analyst is responsible for analyzing Feasibility and Risk.

When the user wants the solution? Providing a perfect solution to the problem a year after it was needed is unlikely to be acceptable. In the software industry, the delivery date is often set before anything else is agreed. Any constraints on schedule demanded by the application domain are valid requirements, as are other resource constraints such as cost, available staffing, and so on. The requirements analyst is responsible for identifying all the relevant Development constraints.

Where is the problem? The problem includes an investigation of both the physical and organizational context: locations and organizational units that are affected by the problem, or which will need to be involved in implementing a solution. The organizational context is especially important for understanding whether the right problem is being tackled. The physical context is especially important for systems that are to be used in harsh or demanding environments. Main task of requirement analyst is to be ensure about active participation of developers to understand the proper problem domain.

Why to solve? In order to make good design decisions, it is necessary to understand the motivations and rationale that the stakeholders have for wanting the problem solved. The requirements analyst is responsible for identifying and analyzing the stakeholders' goals.

6. REQUIREMENT MANAGEMENT

The requirements for the large software systems are always changing, One reason for this is that these systems are usually developed to address 'wicked' problems. Because the problem can not be fully defined, the software requirements are bound to be incomplete. During the software process, the stakeholders understanding of the problem is constantly changing. These requirements must then evolve to reflect this changed problem view.

Furthermore, once a system has been installed, new requirements inevitable emerge. It is hard for users and system customers to anticipate what effects the new system will have on the organization. Once end-users have experience of a system, they discover new needs and priorities.

[3] Large systems usually have a diverse user community where user have different requirements and priorities. These may be conflicting or contradictory. The final system requirements are inevitably a compromise between them and with experience, it is often discovered that the balance of support given to different users has to be changed.

[4] The people who pay for a system and the users of a system are rarely the same people. System customers impose requirements because of organizational and budgetary constraints. These may conflict with end-user requirements and, after delivery, new features may have to be added for user support if the system is to meet its goals.

[5] The business and technical environment of the system changes after installation, and theses changes must be reflected in the system. New hardware may be introduced, it may be necessary to interface the system with other systems, business priorities may change with consequent changes in the system support, and new legislation and regulations may be introduced which must be implemented b the system.

Requirement management is the process of understanding and controlling changes to system requirements. One need to keep track of individual requirements of requirement changes. One need to establish a formal process for making change proposals and linking these to system requirements. The process of requirements management should start as soon as a draft version of the requirement document is available.

6.1 Enduring and volatile requirements

Requirements evolution during the RE process and after a system has gone into service is inevitable. Developing software requirements focuses attention on software capabilities, business objectives and other business systems. As the requirements definition is developed, one normally develop a better understanding of users' needs. This fields information back to the user, who may then propose a change to the requirements.

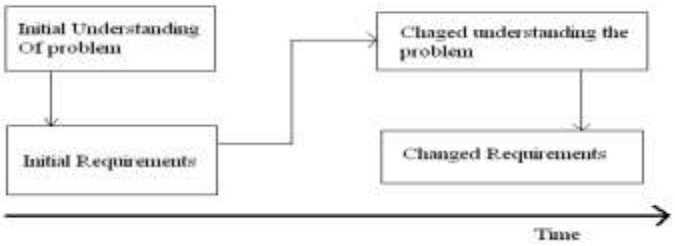

Figure 3: Requirement Evolution

Further more, it may take several years to specify and develop a large system. Over that time, the system's environment and the business objectives change, and the requirements evolve to reflect this.

From an evolution perspective, requirements fall into two classes:

3. *Enduring requirements:* These are relatively stable requirements that derive from the core activity of the organization and which relate directly to the domain of the system.
4. *Volatile requirements:* These are requirements that are likely to change during the system development process or after the system has been become operational. An example would be requirements resulting from government healthcare policies.

6.2 Requirement Management Planning

Planning is essential first stage in the requirements management process. Requirements management is very expensive. For each project, the planning stage establishes the level of requirements management detail that is required. During the requirements management stage, one has to decide on:

1. *Requirement identification:* Each requirement must be uniquely identified so that it can be cross- referenced by other requirements and so that it may be used in traceability assessments.
2. *A change management process:* This is the set of activities that assess the impact and cost of changes.
3. Traceability policies: these policies define the relationships between requirements and the system design that should be recorded and how these records should be maintained.
4. *CASE tool support:* Requirements management involves the processing of large amounts of information about the requirements. Tools that may be used range from specialist requirements management systems to spreadsheets and simple database systems.

There are many relationships among requirements and between the requirements and the system design. There are also links between requirements and the underlying reasons why theses requirements were proposed. When changes are proposed, one has to trace the impact of these changes on other requirements and the system design. Traceability is the property of a requirements specification that reflects the ease of finding related requirements.

There are three types of traceability information that may be maintained:

1. *Source traceability*: information links the requirements to the stakeholders who proposed the requirements and to the rationale for theses requirements. When a change is proposed, one use this information to find and consult the stakeholders about the change.
2. *Requirement Traceability*: Information links dependent requirements within the requirements document. User use this information to assess how many requirements are likely to be affected by a proposed change and the extent of consequential requirements changes that may be necessary.

3. *Design traceability*: Information links the requirements to the design modules where theses requirements are implemented. User use this information to assess the impact of proposed requirements changes on the system design and implementation.

Traceability information is often represented using traceability matrices, which relate requirements to stakeholders, each other or design modules. In a requirements traceability matrix, each requirement is entered in a row and in a column in the matrix. Where dependencies between different requirements exist, these are recorded in the cell at row/column intersection.

Table 2 shows a simple traceability matrix that records the dependencies between requirements. A 'D' in the row/column intersection illustrates that the requirement in the row depends on the requirement named in the column; an 'R' means that there is some other, weaker relationship between the requirements. For example, they may both define the requirements for parts of the same subsystem.

Req. ID	**1.1**	**1.2**	**1.3**	**2.1**	**2.2**	**2.3**	**3.1**	**3.2**
1.1		D	R					
1.2			D			R		D
1.3	R			R				
2.1			R		D			D
2.2								D
2.3		R		D				
3.1								R
3.2							R	

Table 2: A traceability Matrix

6.3 REQUIREMENTS CHANGE MANAGEMENT

Requirement change management should be applied to all proposed changes to the requirements. The advantage of using a formal process for change management is that all change proposals are treated consistently and that changes to the requirements document are made in a controlled way. There are three principal stages to a change management process:

- *Problem analysis and change specification:* The process starts with identified requirements problem or, sometimes, with a specific change proposal. During this stage, the problem or the change proposal is analyzed to check that it is valid. The results of the analysis are fed back to the change requester, and sometimes a more specific requirements change proposal is then made.
- *Change analysis and costing:* The effect of the proposed change is assessed using traceability information and general knowledge of the system requirements. The cost of making the change is estimated in terms of modifications to the required documents.
- *Change implementation:* the requirement document and, where necessary, the system design and implementation are modified. One should organize the requirements document so that one can make changes to it without extensive rewriting or reorganization.

CONCLUSION

As conclusion I describe the specific behavior which shapes an individual's imagination towards a developed system. New challenges of roles in formation of software engineering process as well as requirement traceability and change management is essential in today's era.

REFERENCES:

[1]Assawamekin, Nafon "An Ontology-Based Approach for Multiperspective Requirements Traceability between Analysis Models." Computer and Infromation Science(ICIS), 2010 IEEE/ACIS 9th International Conference on Computer and Information Science. IEEE, 2010.

[2] Chimalakonda, Sridhar, and Kesav V. Nori. "On the nature of roles in sofware engineering." *Proceeeding of the 7th International Workshop on Cooperative and Human Aspects of Software Engineering.* ACM, 2014

[3]Gotel, Orlena CZ, and Anthony CW Finkelstein. "an analysis of the reqirements traceability problem." *Requirements Engineering, 1994., Proceedings of the First International Conference on*. IEEE, 1994

[4] IEE. (1991). *Tools and Techniques for Maintaining Traceability During Design*, IEE Colloquium, Computing and Control Division,
Professional Group C1, Digest No.: 1991/180.

[5] Dorfman, M. & Thayer, R.H. (1990). *Standards, Guidelines and Examples on System and Software Requirements Engineering*, IEEE Computer Society Press Tutorial

[6]Easterbrook, Steve, et al. "Selecting empirical methods for software engineering research." *Guide to advanced empirical software engineering.* Springer London, 2008. 285-311. Chapter 1