International Journal of Research in Engineering & Technology (IJRET)
ISSN 2321-8843
Vol. 1, Issue 2, July 2013, 155-160
© Impact Journals

REQUIREMENT ENGINEERING: AN OVERVIEW

SHIPRA GUPTA[1] & MANOJ WADHWA[2]

[1]Assistant Professor, World Institute of Technology, Sohna, Haryana, India

[2]Professor, Echleon Institute of Technology, Faridabad, Haryana, India

ABSTRACT

Software engineering as a discipline is still evolving and not yet stable. The people associated with computer field often face problems with the software. Software Requirements and estimation provides software professionals the information they need to address requirements engineering and estimation. Requirements form the basis of the initial estimates and plans on which the software product is built and validated. This paper recapitulates the definition of software requirement engineering. This paper discusses the basic concepts and provocations for requirement engineering and estimation.

KEYWORDS: Requirements, SRS, Functional Requirements, Non-Functional Requirements, DFD

INTRODUCTION

Software projects are subject to a multitude of problems that lead to schedule and cost overruns and poor quality of delivered software. This becomes a hurdle in the process of software development for the software professionals.

The basic definition of software engineering was given by Fritz Bauer as, 'the establishment and use of sound engineering principles in order to obtain, economically, software that is reliable and works efficiently on real machines'.IEEE in its standard 610.12-1990, defines software engineering as, 'the application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of approach; that is, the application of engineering to software.[7]

IEEE in its standard 610.12-1990, defined requirements as, 'a condition or capability that must be met or possessed by a system or a system component to satisfy a contract, standard, specification or other formally imposed documents.[3]

Zave[9] provides one of the clearest definitions of RE:

"Requirements engineering is the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behavior, and to their evolution over time and across software families."

CLASSIFICATION OF REQUIREMENTS

Requirements can be classified as

- **Functional Requirements**: Functional requirements describe an interaction between the system and its environment. They focus on the "what" of the system and identify the "functions" and "data" related requirements.
- **Non-Functional Requirements**: These requirements focus on "how well" aspects of the system and identify attributes under which the system needs to operate.

Non-functional requirements can be further classified as

- o **Product Requirement:** Requirements which specify that the delivered product must behave in a particular way e.g. execution speed, reliability
- o **Organizational Requirements:** Requirements which are a consequence of organizational policies and procedures e.g. process standards used, implementation requirements
- o **External Requirements:** Requirements which arise from factors which are external to the system and its development process e.g. interoperability requirements, legislative requirements.

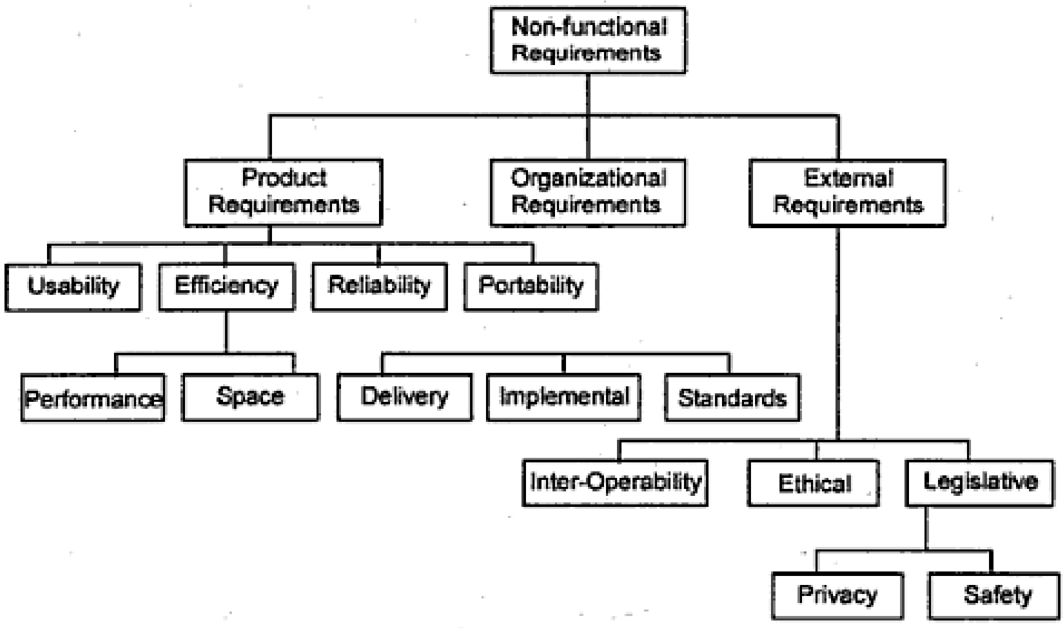

Figure 1: Non-Functional Requirements

- **Behavioral Requirements:** These requirements include any and all information necessary to determine if the runtime behavior of a given implementation is acceptable. The behavioral requirements define all constraints on the system outputs (e.g. value, accuracy, timing) and resulting system state for all possible inputs and current system state.
- **Developmental Quality Attributes:** Developmental Quality Attributes include any constraints on the attributes of the system's static construction. These include properties like testability, changeability, maintainability and reusability.[8]

ACTIVITIES OF REQUIREMENT ENGINEERING

Requirement can be divided into two main set of activities: Requirement Definition and Requirement Management.

Requirement Definition Consists of

- **Requirement Elicitation or Gathering of Requirements:** Requirement elicitation portends discovery of all possible requirements.[1]

- **Requirement Analysis or Modeling:** Requirement analysis starts in parallel with requirement elicitation and involves refining and modeling the requirements to diagnose inconsistencies, errors and other defects.
- **Requirement Documentation:** The requirements that are gathered and modeled are put together in a document known as software requirement specification document.
- **Requirement Review:** The SRS is reviewed by all stake holders.

Requirement Management Consists of

- **Requirement Change Management:** This involves systematic handling of changes to agreed requirements (SRS) during the course of the project.
- **Requirement Traceability:** Requirement traceability is a process of ensuring that one or more test cases address each requirement.[5]

A sign off by the representatives of the customer, users and development team marks the end of the requirements definition activities and the start of the requirement management activities.

REQUIREMENT DOCUMENTATION

The goal of requirement definition and management phase is to closely understand the customer requirements and to systematically organize the requirements into a specification document. [4]

The requirement document is known as Software Requirement Specification Document (SRS). SRS forms the basis for development as well as testing activities.

The SRS has Three Main Parts: the introduction to the document, the overall description of the system for which the SRS is written and the specific requirements.

Introduction includes statement of the purpose of the document, a statement of the system’s scope, a list of definitions and acronyms used in the document.

The overall description provides a complete abstract view. Software lifecycle is used to describe the period of time that starts.

SRS Document Should Have the Following Characteristics

- **Completeness:** For a software requirements specification to be complete, it must have the following properties:
    - Description of all major requirements relating to functionality, performance, design constraints and external interfaces.
    - Definition of the response of the software system to all reasonable situations.
    - Conformity to any software standards, detailing any sections which are not appropriate.
    - Have full labelling and references of all tables and references, definitions of all terms and units of measure.
- Be fully defined, if there are sections in the software requirements specification still to be defined, the software requirements specification is not complete.

- **Consistency:** A software requirement specification is consistent if none of the requirements conflict. There are a number of different types of confliction:
  - **Multiple Descriptors** - This is where two or more words are used to reference the same item, i.e. where the term cue and prompt are used interchangeably.
  - **Opposing Physical Requirements** - This is where the description of real world objects clash, e.g. one requirement states that the warning indicator is orange, and another states that the indicator is red.
  - **Opposing Functional Requirements** - This is where functional characteristics conflict, e.g. perform function X after both A and B has occurred, or perform function X after A or B has occurred.
- **Traceability:** A software requirement specification is traceable if both the origins and the references of the requirements are available. Traceability of the origin or a requirement can help understand who asked for the requirement and also what modifications have been made to the requirement to bring the requirement to its current state. Traceability of references are used to aid the modification of future documents by stating where a requirement has been referenced. By having foreword traceability, consistency can be more easily contained.
- **Unambiguous:** One way of removing ambiguity is to use a formal requirements specification language. The advantage to using a formal language is the relative ease of detecting errors by using lexical syntactic analyzers to detect ambiguity. The disadvantage of using a formal requirements specification language is the learning time and loss of understanding of the system by the client.
- **Verifiable:** A software requirement specification is verifiable if all of the requirements contained within the specification are verifiable. A requirement is verifiable if there exists a finite cost-effective method by which a person or machine can check that the software product meets the requirement. Non-verifiable requirements include "The system should have a good user interface" or "the software must work well under most conditions" because the performance words of good, well and most are subjective and open to interpretation. If a method cannot be devised to determine whether the software meets a requirement, then the requirement should be removed or revised.

Requirement Review: The requirement review is a manual process that involves people from both client and contractor organization. They check the requirements document for anomalies and omissions.[6] Requirement verification is the process of checking at each stage whether the output conforms to requirements for that stage. The basis of verification is a approach involving tests, inspections and analysis. For maximum effectiveness, review and verification should not be treated as a discrete activity to be done only at the end of the preparation of the SRS. Review should be treated as a continuous activity.[7]

- **Continuous Review:** Requirements cannot be captured correctly at the first time, so several iterations are required to define requirements accurately. This can be achieved by reviewing the requirements on a continuous basis. This repeated review is a part of an incremental approach to elicitation.

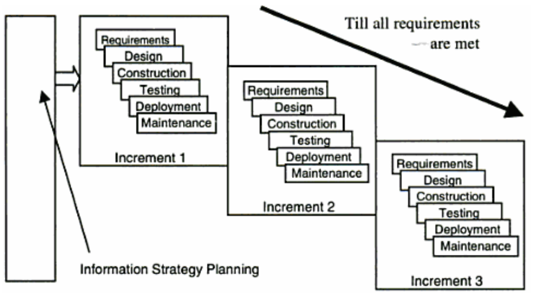

Figure 2: Incremental Life Cycle Model

- Phase End Review: After the SRS document is complete according to the analysts, the document undergoes a formal review. Without proper review, the requirements can be incomplete or incorrect. Phase end review involves reading the SRS, verifying that the SRS is complete, clear, consistent, modifiable, traceable, feasible and testable.

PROBLEMS RELATED TO SRS

According to Ivy Hooks [2] the following are the problems in SRS

- Writing Bad Assumptions: Bad assumptions typically occur either because authors of the SRS do not have access to sufficient information or the information does not exists.
- Writing Implementations instead of Requirements: SRS should contain what is needed and not how it is to be provided. Stating implementation details forces a design approach when not intended.
- Using incorrect terms: In a specification, there are terms to be avoided and terms that must be used in a very specific manner.
- Using wrong Language: Requirements should be easy to read and understand. Requirement statements should not be complicated by long winded explanation of operations, design or other related information.
- Unverifiable Requirements: Every requirement must be verified when writing the requirements. Requirements may be unverifiable due to the use of ambiguous terms.
- Missing Requirements: Detailed Requirement analysis is necessary to assure that all requirements are covered.
- Over-specifying: Over specification is the primary cause of cost overruns on their programs. Over-specifying usually arises from stating something that is unnecessary or from stating overly stringent requirements.

CONCLUSIONS

This paper defined the basic concepts of software engineering. Selection of appropriate life cycle model is one of the first step in any software project. Projects are essentially non-routine, planned set of activities, set up to achieve a well defined objective in a specified time frame using a set of allocated resources. Any software project starts with requirement gathering. For this the broad scope of the proposed system should be identified. According to the study of the system, the requirements are gathered. Requirement gathering is basically in terms of what is to be built. The gathered requirements are documented in a software requirement specification (SRS) document. The requirements are then evaluated by the end-users. According to the gathered requirements a project team is assigned. The project manager assigns the responsibilities to the team members according to their skills. Another responsibility of the project manager is to estimate the cost of development and the time frame within which the development will be completed.

REFERENCES

1. A.A Puntambaker, Software Engineering, Technical Publications Pune, Edition-I.
2. Hooks, I, writing Good Requirements, Proceedings of the Third International Symposium of the NCSE- Volume 2, 1993.
3. KARL E. Weigers, Software Requirements, Edition -2, Microsoft
4. Rajib Mall, Fundamentals of Software Engineering, Edition-III, PHI
5. Rick D. Craig, Stefan P. Jaskiel, Systematic Software Testing, Edition-II, Artech House.
6. Sommerville, Software Engineering, Pearson Edition
7. Swapna Kishore, Rajesh Naik, Software Requirements and Estimation, Tata Mc-Graw Hill Education
8. Software Engineering Project Management, Wiley India Edition
9. Zave, P. (1997). Classification of Research Efforts in Requirements Engineering. ACM Computing Surveys, 29(4): 315-321.