Topic 4: Software Requirements

Topic 4 Notes

The software requirements are description of features and functionalities of the target system. Requirements convey the expectations of users from the software product. The requirements can be obvious or hidden, known or unknown, expected or unexpected from client's point of view.

Requirement Engineering

The process to gather the software requirements from client, analyze, and document them is known as requirement engineering.
The goal of requirement engineering is to develop and maintain sophisticated and descriptive 'System Requirements Specification' document.

Requirement Engineering Process

It is a four-step process, which includes –

i. Feasibility Study
ii. Requirement Gathering
iii. Software Requirement Specification
iv. Software Requirement Validation

Feasibility study

When the client approaches the organization for getting the desired product developed, it comes up with a rough idea about what all functions the software must perform and which all features are expected from the software.
Referencing to this information, the analysts do a detailed study about whether the desired system and its functionality are feasible to develop.
This feasibility study is focused towards goal of the organization. This study analyzes whether the software product can be practically materialized in terms of implementation, contribution of project to organization, cost constraints, and as per values and objectives of the organization. It explores technical aspects of the project and product such as usability, maintainability, productivity, and integration ability.
The output of this phase should be a feasibility study report that should contain adequate comments and recommendations for management about whether or not the project should be undertaken.

Requirement Gathering

If the feasibility report is positive towards undertaking the project, next phase starts with gathering requirements from the user. Analysts and engineers communicate with the client and end-users to know their ideas on what the software should provide and which features they want the software to include.

Software Requirement Specification (SRS)

SRS is a document created by system analyst after the requirements are collected from various stakeholders.

SRS defines how the intended software will interact with hardware, external interfaces, speed of operation, response time of system, portability of software across various platforms, maintainability, speed of recovery after crashing, Security, Quality, Limitations etc.

The requirements received from client are written in natural language. It is the responsibility of the system analyst to document the requirements in technical language so that they can be comprehended and used by the software development team.

SRS should come up with the following features:

i. User Requirements are expressed in natural language.

ii. Technical requirements are expressed in structured language, which is used

iii. inside the organization.

iv. Design description should be written in Pseudo code.

v. Format of Forms and GUI screen prints.

vi. Conditional and mathematical notations for DFDs etc.

Software Requirement Validation

After requirement specifications are developed, the requirements mentioned in this document are validated. User might ask for illegal, impractical solution or experts may interpret the requirements inaccurately. This results in huge increase in cost if not nipped in the bud. Requirements can be checked against following conditions -

i. If they can be practically implemented

ii. If they are valid and as per functionality and domain of software

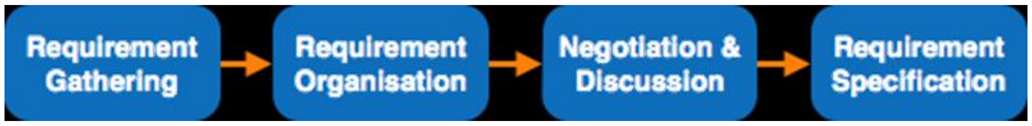
iii. If there are any ambiguities

iv. If they are complete

v. If they can be demonstrated

Requirement Elicitation Process

Requirement elicitation process can be depicted using the following diagram:



i. **Requirements gathering -** The developers discuss with the client and end users and know their expectations from the software.

ii. **Organizing Requirements -** The developers prioritize and arrange the requirements in order of importance, urgency and convenience.

iii. **Negotiation & discussion -** If requirements are ambiguous or there are some conflicts in requirements of various stakeholders, it is then negotiated and discussed with the stakeholders. Requirements may then be prioritized and reasonably compromised.
The requirements come from various stakeholders. To remove the ambiguity and conflicts, they are discussed for clarity and correctness. Unrealistic requirements are compromised reasonably.

iv. **Documentation -** All formal and informal, functional and non-functional requirements are documented and made available for next phase processing.

Requirement Elicitation Techniques

Requirements Elicitation is the process to find out the requirements for an intended software system by communicating with client, end users, system users, and others who have a stake in the software system development.
There are various ways to discover requirements.

Interviews

Interviews are strong medium to collect requirements. Organization may conduct several types of interviews such as:

i. Structured (closed) interviews, where every single information to gather is decided in advance, they follow pattern and matter of discussion firmly.
ii. Non-structured (open) interviews, where information to gather is not decided in advance, more flexible and less biased.
iii. One-to-one interviews which are held between two persons across the table.
iv. Group interviews which are held between groups of participants. They help to uncover any missing requirement as numerous people are involved.

Questionnaires

A document with pre-defined set of objective questions and respective options is handed over to all stakeholders to answer, which are collected and compiled.
A shortcoming of this technique is, if an option for some issue is not mentioned in the questionnaire, the issue might be left unattended.

Task analysis

Team of engineers and developers may analyze the operation for which the new system is required. If the client already has some software to perform certain operation, it is studied and requirements of proposed system are collected.

Domain Analysis

Every software falls into some domain category. The expert people in the domain can be a great help to analyze general and specific requirements.

Brainstorming
An informal debate is held among various stakeholders and all their inputs are recorded for further requirements analysis.

Prototyping
Prototyping is building user interface without adding detail functionality for user to interpret the features of intended software product. It helps giving better idea of requirements. If there is no software installed at client's end for developer's reference and the client is not aware of its own requirements, the developer creates a prototype based on initially mentioned requirements. The prototype is shown to the client and the feedback is noted. The client feedback serves as an input for requirement gathering.

Observation
Team of experts visit the client's organization or workplace. They observe the actual working of the existing installed systems. They observe the workflow at the client's end and how execution problems are dealt. The team itself draws some conclusions which aid to form requirements expected from the software.

Software Requirements Characteristics
Gathering software requirements is the foundation of the entire software development project. Hence, they must be clear, correct, and well-defined.
A complete Software Requirement Specifications must be:

i. Clear
ii. Correct
iii. Consistent
iv. Coherent
v. Comprehensible
vi. Modifiable
vii. Verifiable
viii. Prioritized
ix. Unambiguous
x. Traceable
xi. Credible source

Software Requirements
We should try to understand what sort of requirements may arise in the requirement elicitation phase and what kinds of requirement are expected from the software system.

Broadly software requirements should be categorized in two categories:

Functional Requirements

Requirements, which are related to functional aspect of software fall into this category. They define functions and functionality within and from the software system. Examples -

i. Search option given to user to search from various invoices.
ii. User should be able to mail any report to management.
iii. Users can be divided into groups and groups can be given separate rights.
iv. Should comply business rules and administrative functions.
v. Software is developed keeping downward compatibility intact.

Non-Functional Requirements

Requirements, which are not related to functional aspect of software, fall into this category. They are implicit or expected characteristics of software, which users make assumption of.

Non-functional requirements include -

i. Security
ii. Logging
iii. Storage
iv. Configuration
v. Performance
vi. Cost
vii. Interoperability
viii. Flexibility
ix. Disaster recovery
x. Accessibility

Requirements are categorized logically as:

i. **Must Have**: Software cannot be said operational without them.
ii. **Should have**: Enhancing the functionality of software.
iii. **Could have**: Software can still properly function with these requirements.
iv. **Wish list**: These requirements do not map to any objectives of software.

While developing software, ‘Must have’ must be implemented, ‘Should have’ is a matter of debate with stakeholders and negation, whereas ‘Could have’ and ‘Wish list’ can be kept for software updates.

User Interface requirements

User Interface (UI) is an important part of any software or hardware or hybrid system. A software is widely accepted if it is –

i. easy to operate
ii. quick in response

iii. effectively handling operational errors
iv. providing simple yet consistent user interface

User acceptance majorly depends upon how user can use the software. UI is the only way for users to perceive the system. A well performing software system must also be equipped with attractive, clear, consistent, and responsive user interface. Otherwise the functionalities of software system cannot be used in convenient way. A system is said to be good if it provides means to use it efficiently. User interface requirements are briefly mentioned below –

i. Content presentation
ii. Easy Navigation
iii. Simple interface
iv. Responsive
v. Consistent UI elements
vi. Feedback mechanism
vii. Default settings
viii. Purposeful layout
ix. Strategical use of color and texture.
x. Provide help information
xi. User centric approach
xii. Group based view settings.

Software System Analyst

System analyst in an IT organization is a person, who analyzes the requirement of proposed system and ensures that requirements are conceived and documented properly and accurately. Role of an analyst starts during Software Analysis Phase of SDLC. It is the responsibility of analyst to make sure that the developed software meets the requirements of the client.

System Analysts have the following responsibilities:

i. Analyzing and understanding requirements of intended software
ii. Understanding how the project will contribute to the organizational objectives
iii. Identify sources of requirement
iv. Validation of requirement
v. Develop and implement requirement management plan
vi. Documentation of business, technical, process, and product requirements
vii. Coordination with clients to prioritize requirements and remove ambiguity
viii. Finalizing acceptance criteria with client and other stakeholders

Software Metrics and Measures

Software Measures can be understood as a process of quantifying and symbolizing various

attributes and aspects of software.
Software Metrics provide measures for various aspects of software process and software product.
Software measures are fundamental requirements of software engineering. They not only help to control the software development process but also aid to keep the quality of ultimate product excellent.
According to Tom DeMarco, a (Software Engineer), “You cannot control what you cannot measure.” By his saying, it is very clear how important software measures are.
Let us see some software metrics:

i. **Size Metrics -** Lines of Code (LOC), mostly calculated in thousands of delivered source code lines, denoted as KLOC.
ii. **Function Point Count** is a measure of the functionality provided by the software. Function Point count defines the size of functional aspect of the software.
iii. **Complexity Metrics -** McCabe’s Cyclomatic complexity quantifies the upper bound of the number of independent paths in a program, which is perceived as complexity of the program or its modules. It is represented in terms of graph theory concepts by using control flow graph.
iv. **Quality Metrics -** Defects, their types and causes, consequence, intensity of severity and their implications define the quality of the product. The number of defects found in development process and number of defects reported by the client after the product is installed or delivered at client- end, define quality of the product.
v. **Process Metrics -** In various phases of SDLC, the methods and tools used, the company standards and the performance of development are software process metrics.
vi. **Resource Metrics -** Effort, time, and various resources used, represents metrics for resource measurement.