UNIT II

Requirements Analysis and Specification

S.No. | Topics | Pages
--- | --- | ---
1. | SW Requirements | 47
2. | SW Req. Document (SRD) | 53
4. | Feasibility Study | 56
5. | Requirement Elicitation & Analysis | 57
6. | Requirement Validation | 59
7. | Requirement Mgmt | 60
8. | Petri Nets | 64
9. | Data Dictionaries | 67
Software Engineering

1. **S/w Requirements**

S/w req are classified as:

- **Functional** or **Non-functional** requirements

  ➔ **Functional** req - These are the services the sys should provide.
  ➔ **Non-functional** req - The constraints imposed by sys.

Functional Requirements:

- The final req for a sys describe what the sys should do.
- These req depend on the type of s/w being developed, the expected and annual.
Final req are usually described in an abstract way that can be understood by users. More specific details are included in the final req, describing the sys for its input and output, exceptions, etc. in detail.

E.g.: The final req for a mental health care patient mgmt sys (mhc-pms) used to maintain info about patients receiving treatment for mental health problems, may be:

1. The appointment lists for all clinics should be able to search by name.
2. For each clinic, a list of patients who are expected to attend appointments that day will be provided.
3. Specific facilities to be provided by the sys.
4. Final req should be both consistent and specific.
Non-functional Requirements:

- The requirements that are not directly concerned with the specific service delivered by the sys to its user.
- It relates to emergent sys properties such as reliability, response time, and storage capacity.
- Also, they may define constraints on the sysimple such as capabilities of I/O devices or the data representations used in the interface with other systems.
- Non-functional requirements such as performance, security, or availability, usually specify or constrain characteristics of the system as a whole.
- Failing to meet non-functional req means the whole sys is unusable.

For eg: the aircraft sys does not meet its reliability req it will not be certified as safe.
The simple in these requirements may be diffused throughout the system. There are two reasons for this:

1) non-final req may affect overall arch of a sys more than the indivi component.

2) A single non-final req such as security req may grow into a no of related fin-al req, that define new sys services that are required.

Classification of non-final req:

- Non-final req
  - Product req
    - Feature req
      - Operational req
      - Security req
      - Usability req
      - Reliability req
  - Original req
    - Environment req
      - Operational req
      - Dev req
      - Leg req
    - Reg req
      - Accounting req
1. Product Requirement

The requirement specifies the behavior of the system.

- It includes performance requirements on how fast the system must execute and how much memory it requires.
- Reliability requirements that set out the acceptable failure rates.
- Security requirements and usability requirements.

2. Organizational Requirements

These requirements are broad.

Use requirements derived from policies and procedures in the customer's and developer's environment.

- Operational process requirements define how the system will be used.
- Development requirements that specify the programming language.
- Environment requirements that specify the operational environment of the system.
**External Requirements**

This broad heading covers all requirements that are derived from factors external to the sys and its dev proc. Regulations req that set by law & enforc are what must be done for the sys to be approved for use by a reg. Legislative req that must follow the laws & enforc req.

The non-functional sys proper is an in below table.

<table>
<thead>
<tr>
<th>Property</th>
<th>Measures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed</td>
<td>processed trans/sec, use/mbps/mbits</td>
</tr>
<tr>
<td>Size</td>
<td>M bytes, no. of Rom chips</td>
</tr>
<tr>
<td>Ease of use</td>
<td>Training time, no. of help reqd</td>
</tr>
<tr>
<td>Reliability</td>
<td>mean time to failure prob of unavailability, Rate of failure occurrence, availability</td>
</tr>
<tr>
<td>Robustness</td>
<td>time to restart after failure, no. of events, causing failure prob of data corruptions, probability of target dependant stop</td>
</tr>
</tbody>
</table>
The S/W req doc (SRS) is an official statement of what the sys developers should implement.

- It should include both user & system reqs. Sometimes, they are integrated into a single description.
- If there are no user reqs, then detailed sys reqs may be separate.

- When a new contractor is developing, the sys reqs remain essential. When req change, the doc is out of date, so the effort is largely wasted.

- Extreme prog approaches collect the user reqs incrementally & write these on a new req. Then use card as user stories. Then use prioritized req for imp of the next increment of the sys. This is suitable for business sys, where req are not stable.

- Req docs have diverse set of users, ranging from senior mgmt to org, to the engineers responsible for
specify the req. & check whether we meet their needs. auto.
specify changes to req.

use req. doc to plan a
bid for the sys & to plan
the sys.

test
use the req. to understand
what sys is to be developed

use the req. to develop
validation tests for the sys.

use the req. to understand
the sys & the relationships
between its parts.

IEEE std. for req. document.

Chapter

Preface

expected readability of docu &
describe its versions

Intro

need for the sys. briefly describes the
sys's fn. & explain how it works.

Glossary

technical terms used in the docu.

curr req

define service provider for the user.

defn.

define non-final req.

sys

high level overview of the sys.
1. Define the requirements document.

2. Sys. customer - specify the requirements and check whether we meet their needs. Also, specify changes to req.

3. Sys. engg - use the req. to understand what sys. is to be developed.

4. Sys. test - use the req. to develop validation tests for the sys.

5. Sys. maintenance - use the req. to understand the sys. and its relationships from its parts.

<table>
<thead>
<tr>
<th>IEEE Std for req. document</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter</td>
</tr>
<tr>
<td>Preface</td>
</tr>
<tr>
<td>Intro</td>
</tr>
<tr>
<td>Glossary</td>
</tr>
<tr>
<td>Core req.</td>
</tr>
<tr>
<td>Def. sys.</td>
</tr>
<tr>
<td>High level overview of the sys.</td>
</tr>
</tbody>
</table>
Requirement Engineering Process:

Requirements process may include 4 high-level activities:

(i) business (feasibility study)
(ii) discovering req (elicitation & analysis)
(iii) converting true req into some std (specifications)
(iv) checking that the req actually define the sys that the customer wants.

The req engg is an iterative process in which the activities are interleaved. The activities are organized as an iterative process around a spiral, with

Appendices: Detailed, specific info., that is related to apply developed.

Index: Index of diagram, table, fn etc.
Feasibility Study

A feasibility study should take place early in the project. The study should address the following questions:

1. Is there a need for the new product or service?
2. Can the project be completed within the required time frame?
3. Is the project technically feasible?
4. Can the project be completed within the budget constraints?

Feasibility studies are typically performed in the first phase of a project. They help determine whether the project is worth pursuing.

The study is conducted in a structured way, and it varies in detail from project to project. Often, the study is devoted to eliciting requirements for the new system.

The study should not be too refined; it should be sufficient to determine the feasibility of the project.

Once the feasibility study is complete, the project may be continued or abandoned.
Requirement Elicitation and Analysis (REA)

REA may involve a variety of different kinds of people in an org. A system stakeholder or anyone who must have direct or indirect influence on the sys req. Stakeholder includes end users & engineers, developers.

- Req. Discovery
- Req. Classification and Organisation
- Requirement Prioritisation & Negotiation

REA process activities are:

1. Requirement Discovery:
   - It is a process of interacting with stake holders of syg to discover their requirements. Domain req. from stakeholders and documentation are also discovered during this activity.

2. Requirement Classification and Organisation:
   - It takes constructed collection of req. groups, related req., & organizes things into clusters.
grouping requirements as use a map of sys architecture to identify sub systems and to associate req with each sub system.

3. Requirements Prioritization and Negotiation

When multiple stake holders are involved, req will conflict. This activity is concerned with prioritizing req and finding and resolving req conflict through negotiation.

4. Requirements Specification

The req are documented and i/p into the next round of the sys. The req are produced in a formal or informal document. The process of gathering info about the required sys and exist sys and distilling the info from this info with stakeholders thru interaction and observation and use scenario & prototype to help stakeholders understand what is to help stakeholders understand what is sys will be like.

Interviewing:

Interview may be of two types: closed interviews
Scenarios:

People usually find it easier to relate to real-life examples than abstract descriptions.

Use cases:

We cases are a map discovery techniques that were first introduced in the obbectory model.

Requirement Validation:

It is the process of checking that requirements actually define the sys that customer really wants.

Req validation is important because errors in a requirements document can lead to extensive rework costs when there problems are discovered during development or after the sys is in service.

The cost of fixing a req problem by making a sys change is usually much greater than repairing design or coding errors.

Different types of checks:

1) Validity checks: A user may think that a sys is needed to perform certain fns. The thought and analysis may identify additional or different fns that are required.
(vi) consistency checks:
Reg. in the document should not conflict. That is there should not be contradictory constraints of the same sys.

(ii) completeness checks.
The req. doc. should include req. that define all fn. and the constraints intended by the sys. user.

(iv) Realism checks:
Using knowledge of existing tech., the req. should be checked to ensure that they can actually be implemented.

(iv) verifiability:
To reduce the potential for change, the req. should always be written so that they can be verified.

This means that we should be able to write a set of tests that can determine that the delivered sys. meets each specified requirement.

Requirement Management:
The req. for large sys. are always changing. one reason is that sys. are usually developed to address the prob. that cannot be completed defined in time.
Once a sys has been installed and is regularly used, new reqs inevitably emerge. There are several reasons why change is inevitable.

1) The business and technical environments of the system always change after install, so new reqs may be introduced. It may be necessary to update the system with new

2) People who pay for a sys and use it are rarely the same people.

3) Large sys usually have a diverse community, with many users having different reqs & priorities that may be conflicting.

Req mgmt is the process of understanding and controlling changes to sys reqs.

Requirement mgmt planning:

Planning is an essential stage in the Req mgmt process.

0 Requirement identification:

Each req must be uniquely identified so that it can be cross-referenced with other reqs and used in trouble shooting.
(ii) A change mgmt process

This is the set of activities that
assess the impact and cost of changes.

(iii) Traceability policies:

These policies define the
relationships between each req. & btw the
design & the sys design that should be
recorded. The traceability policy should
also define how these records should
be maintained.

(iv) Tool support

Req. mgmt involves the
processing of large amount of info
about the req. and the tool

Req. mgmt needs automated support tool.

we need tool support for

Requirement Storage:

The req. should be maintained
in a secure, managed database store that
is accessible to everyone involved
in the req. engg process.

Change mgmt:

The process of change mgmt
is simplified if active
tool support is available.

Traceability mgmt:

Tool support for traceability
allows related reqs to be improved.
Req change mgnt should be applied to all proposed changes to a sys req, after the sys req doc has been approved.

Identified, prob analysis, change anal & costing, change implementation.

Revised

Change mgmt is essential because we need to decide if the benefits of impl new sys req are justified by the costs of impl.

3 principal stages to a change mgmt process:

1. Problem analysis and change specification

   The problem or change proposal is analysed to check that it is valid.

2. Change analysis and costing

   The cost of making the change is estimated in terms of modification to the req doc.

3. Change implementation

   When even a new sys req has to be urgently implemented we should not change the sys and modify the req doc as it leads to req req and sys impl getting out of step.

SriVidya College of Engineering & Technology

www.studentsfocus.com
A petri net is a form of finite state machine useful in modeling concurrency and asynchronous communication. Petri nets are used to describe and analyze the structure and information flow in systems. Petri nets consist of a set of places $P$, set of transitions $T$, an input function $\delta : P \to T$, and an output function $\gamma : T \to P$. Places represent storage for input or output. Transitions represent activities that transform input into output. An input mapping $\delta$ maps a transition to its input places, and an output mapping $\gamma$ maps a transition to its output places.

Notation for Petri nets:

- **Token (information)**

  ![Token](Image)

- **Place**

  ![Place](Image)

- Place marked with one token

- **Transition**

  ![Transition](Image)

- Transition marked with a place and connection by two places and transitions
whenever the activity represented by the transition \( t \) occurs, this activity is hidden from view. However, the transition completes the transformation of its \( p \) event occur. The transition is said to fire and \( \text{token} \) is deposited in its \( o/p \) place.

\( \rightarrow \) A \text{token} represents a piece of info either to be processed by one or more transitions or info resulting from the firing of one or more transitions.

The placement of \( p \) tokens in a Petri net is called a marking of the net.

Petri nets are governed by transition firing rules:

1. A transition is enabled if each \( p \) of its \( i/p \) places has the required no. of \( \text{tokens} \), one for each arc leading from a place to a transition.

2. A transition can only fire if it is enabled.

3. Whenever a transition fires, one or more \( \text{tokens} \) that enabled it is removed and transition \( t \) places one or more \( \text{tokens} \) in each of its \( o/p \) places, one for each arc leading from \( t \) to an \( o/p \) place.
The significance of the firing rule is that more than one transition can be enabled at the same time (concurrent process).

In second config (after transition $b_1$ fires), transitions $b_2$ and $b_3$ are enabled (the activities represented by these transitions overlap in time).

The behaviours represented by changing the petri net result from successive firings of transition $b_1$. Successive firings can be explained by writing out some mapping.

<table>
<thead>
<tr>
<th>Config (a)</th>
<th>$T(b_1) = {P_1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I(b_2) = {P_1, P_3}$</td>
<td></td>
</tr>
<tr>
<td>$I(b_3) = {P_2}$</td>
<td></td>
</tr>
</tbody>
</table>

With $b_1$ enabled.
Data Dictionaries:

A data dic stores info about data items found in a DFD.

A summary of typical info used to construct a data dic is given below.

<table>
<thead>
<tr>
<th>Information</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Identifies data item</td>
</tr>
<tr>
<td>Alias</td>
<td>Identifies other names, abbr,</td>
</tr>
<tr>
<td>Data Type (DS Type)</td>
<td>Used to identify a data item</td>
</tr>
<tr>
<td>Description</td>
<td>Indicates how (why) a data item is used</td>
</tr>
<tr>
<td>Duration (begin)</td>
<td>Life span of data (when created)</td>
</tr>
<tr>
<td>Accuracy</td>
<td>High, medium, low accuracy</td>
</tr>
<tr>
<td>Range of values</td>
<td>Allowable values of data item</td>
</tr>
</tbody>
</table>

conf (c) = 0 (t₁₂) = 0 (t₃₈) = f₄₅₃ (b)
A data dictionary supplies information on data typing, required accuracy of data, useful to designers and implementers. A name, alias, type and description indicates how to identify possible other names, properties or how the type of data and what is how it is used.

duration, accuracy and range

duration, accuracy and range

data values specify a life span, all possible data values of data items respectively.

Data flow specify processes that generate or receive data.

When data are derived from a real-time system, data items can have timing constraints specifying the time before the data becomes out-of-date. For example, traffic flow changes continuously, so traffic flow data needs to be refreshed, replacing old with new data.
A data dictionary can be used to check the completeness and consistency of DFDs. Whenever all bubbles, arrows, and db have labels and all arrows have sources and destinations, a DFD is considered complete.