Requirements Elicitation

Requirements

Definition/Description of the system in terms understood by the customer, in natural language.

Functional requirements: describe what the system should do, described in an abstract way that can be understood by end users
Nonfunctional requirements: aspects not directly related to functional behavior. Ex: user interface look and feel, documentation, hardware considerations, error handling, security issues

Constraints: Imposed by the client or the environment Ex: use of specific tools

Validation:

- **Correctness:** The requirements represent the client's view
- **Completeness:** All possible scenarios, in which the system can be used, are described
- **Consistency:** There are no requirements that contradict each other
- **Clarity:** Requirements can only be interpreted in one way
- **Realism:** Requirements can be implemented and delivered
- **Traceability:** Each system behavior can be traced to a set of functional requirements

Prioritizing requirements:

- **High priority**: a high-priority feature must be demonstrated successfully during client acceptance
- **Medium priority:** usually implemented and demonstrated in the second iteration of the system development
- **Low priority :** illustrates how the system is going to be used in the future if not yet available technology enablers are available

Minimum viable product (MVP): is the release of a new product that includes the minimum features required and is used to validate customer needs and demands, before developing the full product.

Requirements Elicitation

Use - cases

Use cases - are abstractions describing all possible cases.

Structure of a use-case:

Name: indicates what the user is trying to accomplish (starts with verb)
Participating actors: generalized (ex: "Traveler" not "John"), named with noun
Entry condition:
Flow of events : uses informal natural language
Exit condition
Exceptions: describe what happens if things go wrong
Special requirements: nonfunctional requirements, constraints

Use case associations: are dependencies between use cases and are used to reduce complexity. (decompose a long use case into shorter <<include>>, extend an initial problem statement <<extend>>, separate alternate flows of events)

How do we validate a use-case:

- It should describe a complete user transaction,
- Is initiated by an actor and may interact with other actors
- the boundary of the system should be clear
- It should not describe the UI of the system
- Exceptions should be described separately
- It should not exceed two or three pages in length, otherwise decompose it in smaller use cases

Requirements Elicitation

Requirements Analysis Document

RAD - describes the system in terms of functional and nonfunctional requirements. Who is this for: the client, the users, the project management, the business analyst, the devteam;

The structure of RAD should contain:

1. **Introduction** - provide a brief overview of the function of the system and the reasons for its development, its scope, objectives and success criteria
2. **Current system** - describes the current state
3. **Proposed system**
    - 3.1 **Overview** - presents a functional overview of the system
    - 3.2 **Functional requirements**
    - 3.3 **Nonfunctional requirements**
    - 3.4 **Constraints ("Pseudo requirements")**
    - 3.5 **System models**
        - 3.5.2 **Use case model**
        - 3.5.3 **Object model**
        - 3.5.4 **Dynamic models**
        - 3.5.5 **User interface**
4. **Glossary** - to establish a clear terminology. Documenting important terms and their definitions

TASK

- validate the requirements using the specified criteria and adjust them if necessary
- prioritize the requirements and build the MVP for your project
- refine use-cases, identify relationships between actors and use cases
- start creating the RAD following the structure above