What is it? The process of studying customer and user needs to arrive at a definition of system, hardware, or software requirements (i.e., a property that the software must have)

Importance Many defects can be traced to the (requirements) specification. These can be very hard to fix, since they are very structural to the project and, if discovered late, may be spread throughout the project

Problems (mis)communication & (mis)understanding

Evolving requirements – requirements creep: uncontrolled changes or continuous growth in a project's requirements

Levels of software requirements

- Business requirements/needs - high-level objectives vision and scope document
- User requirements/needs - goals or tasks that the user must be able to perform with the product use case models
- System requirements - requirements for the system as a whole (HW/SW) system requirements specification document
- Software requirements - derived from system requirements software requirements specification (SRS) document

0.1 Types of software requirements

Functional requirements (capabilities) Functions that the software is to execute

Nonfunctional requirements Act to constrain the solution

- Mostly quality requirements – Example: The maximum system down-time should be 8 hours per year
- Can also include development process requirements (such as programming languages, etc.)

Quality requirements Quality characteristics, sub-characteristics and metrics

ISO/IEC 25010 standard

- Functionality suitability – degree to which provides functions that meet stated and implied needs
- Performance efficiency – performance relative to execution conditions
- Reliability – degree to which specified functions are performed under specified conditions for a specified period of time
- Usability - effectiveness, efficiency and satisfaction in a specified context of use
- Compatibility - degree to which information can be exchanged with other products, systems or components
- Maintainability – degree of effectiveness and efficiency during modification
- Portability - degree of effectiveness and efficiency during hardware or software transfer
- Security – degree of information and data protection

Requirements Engineering and Agile processes Requirements Engineering mainly goes against the agile methodology to not overplan before development

Solution: user stories - Lightweight way to record a software need, with just enough information

0.2 Requirements engineering process

Requirements elicitation (or discovery) Interact with stakeholders and other sources (documents, existing systems, etc.) to collect/discover their requirements

Problems

- Problems of scope – ill-defined boundaries, unnecessary information
- Problems of understanding
- Problems of volatility - Requirements evolve over time.

Requirements analysis (& negotiation) Detect and resolve problems with the requirements

- Completeness
- Consistency

- Unambiguity
- Verifiability
- Necessity
- Feasibility

Requirements specification Production a Software Requirements Specification (SRS) document, as well as other docs, Prototypes, Models, etc..

Requirements validation Demonstrate that the requirements define the system that the customer really wants. Attempts to prevent the costly errors (as said before) that can happen during requirements

0.3 Requirements elicitation techniques

Interviews - Most widely used requirements elicitation technique

- Open/unstructured - various issues are explored with stakeholders
- Closed/structured - based on a pre-determined list of questions
- Mixed

Brainstorming Useful to elicit new and innovative requirements

- Moderator and 4-8 people
- Generate new ideas and discuss, review and organize them

Questionnaires (surveys) Well-suited for confirming/prioritizing previously identified candidate requirements Set of questions

Goal analysis Hierarchical decomposition of stakeholder goals to derive system requirements

Social Observation and Analysis Requirements can be derived from the external observation of the routine way and tactics of work

Prototyping Initial/primitive version of a system (cheaper, faster to develop, limited in functionality)

- Throw-away prototypes - focus on requirements rather than implementation constraints
- Evolutionary prototypes - Appropriate for rapid, iterative, application development

0.4 System models in requirements engineering

A simplified representation of a system (as-is or to-be) from a certain perspective - tackle complexity through abstraction. Also helps removing the ambiguity and lack of structure inherent to natural language descriptions. In requirements engineering:

- Use case model - for organizing functional requirements
- Domain model – for organizing the vocabulary and information requirements

0.4.1 Use case model

Use case diagram(s) + associated documentation

Purpose Show the system purpose and usefulness, capture functional requirements (through the use cases), specify the system context (actors)

Actors

- User role or external system

Use cases

- Functionality or service as perceived by users
- Type of interaction between actors and the system
- Sequence of actions, including variants, resulting in an observable result with value for an actor

Relationships

- Generalization
- Extension
- Include

0.4.2 Domain Modeling

- Used to organize the vocabulary of the problem domain or to capture information requirements
- Represented through UML class diagrams
- Can use integrity constraints (or invariants) associated with classes to restrict valid object states