Software quality assurance

Lecture 7

(3-April-2013)

Instructor: Mr. Natash Ali Mian

Department of CS and IT

The University of Lahore

Switch off mobile phones during lectures, or put them into silent mode

CONTRIBUTE TO SOCIETY

Contents

- Requirement Engineering and Software Quality

Term Paper

Finalize Group Members	26-Feb-2013
Finalize Topic	12-Mar-2013
Search Papers and Sort Selected (TODAY)	20-Mar-2013
Go Through the Abstract and Introduction of Selected Papers	27-Mar-2013
Submit a Summary and Comments on related papers	**09-Apr-2013**
Submit Initial Draft	30-Apr-2013
Final Paper Submission	21-May-2013
Feedback on Final Submission + Plagiarism Report	28-May-2013
Final Presentation	4-June-2013

Please note that Every Phase has Marks

INTRODUCTION

- Requirements form the basis for all software products

- Requirements engineering is the process, which enables us to systematically determine the requirements for a software product

Requirement

- Something required, something wanted or needed
  - Webster's dictionary
- Difference between *wanted* and *needed*

IMPORTANT

- We are discussing requirements with reference to quality assurance and management

Importance of Requirements

- The hardest single part of building a software system is deciding what to build…No other part of the work so cripples the resulting system if done wrong. No other part is difficult to rectify later
  - Fred Brooks

Software Requirements - 1

- A complete description of *what* the software system will do without describing *how* it will do it

- External behavior

Software Requirements - 2

- Software requirements may be:
  - Abstract statements of services and/or constraints
  - Detailed mathematical functions
  - Part of the bid of contract
  - The contract itself
  - Part of the technical document, which describes a product

IEEE Definition

- A condition or capability that must be met or possessed by a system…to satisfy a contract, standard, specification, or other formally imposed document
  - IEEE Std 729

Sources of Requirements

- Stakeholders
  - People affected in some way by the system
  - Stakeholders describe requirements at different levels of detail
- Documents
- Existing system
- Domain/business area

Examples of Requirements - 1

- The system shall maintain records of all payments made to employees on accounts of salaries, bonuses, travel/daily allowances, medical allowances, etc.
- The system shall interface with the central computer to send daily sales and inventory data from every retail store

Examples of Requirements - 2

- The system shall maintain records of all library materials including books, serials, newspapers and magazines, video and audio tapes, reports, collections of transparencies, CD-ROMs, DVDs, etc.

Kinds of Software Requirements

- Functional requirements
- Non-functional requirements
- Domain requirements
- Inverse requirements
- Design and implementation constraints

Functional Requirements - 1

- Statements describing what the system does
- Functionality of the system
- Statements of services the system should provide
  - Reaction to particular inputs
  - Behavior in particular situations

Functional Requirements - 2

- Sequencing and parallelism are also captured by functional requirements
- Abnormal behavior is also documented as functional requirements in the form of exception handling

Non-Functional Requirements - 1

- Most non-functional requirements relate to the system as a whole. They include constraints on timing, performance, reliability, security, maintainability, accuracy, the development process, standards, etc.

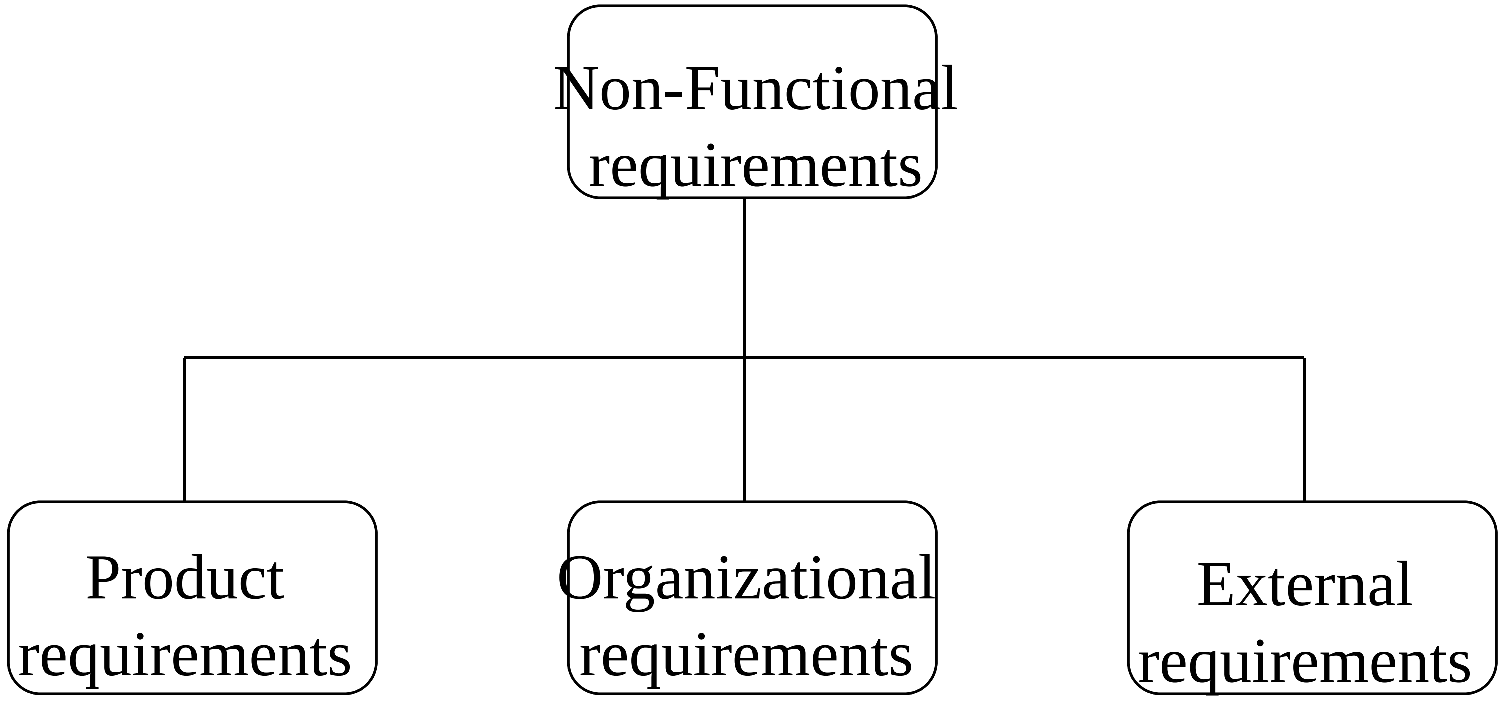
Non-Functional Requirements - 2

- They are often more critical than individual functional requirements
- Capture the emergent behavior of the system, that is they relate to system as a whole
- Many non-functional requirements describe the quality attributes of the software product

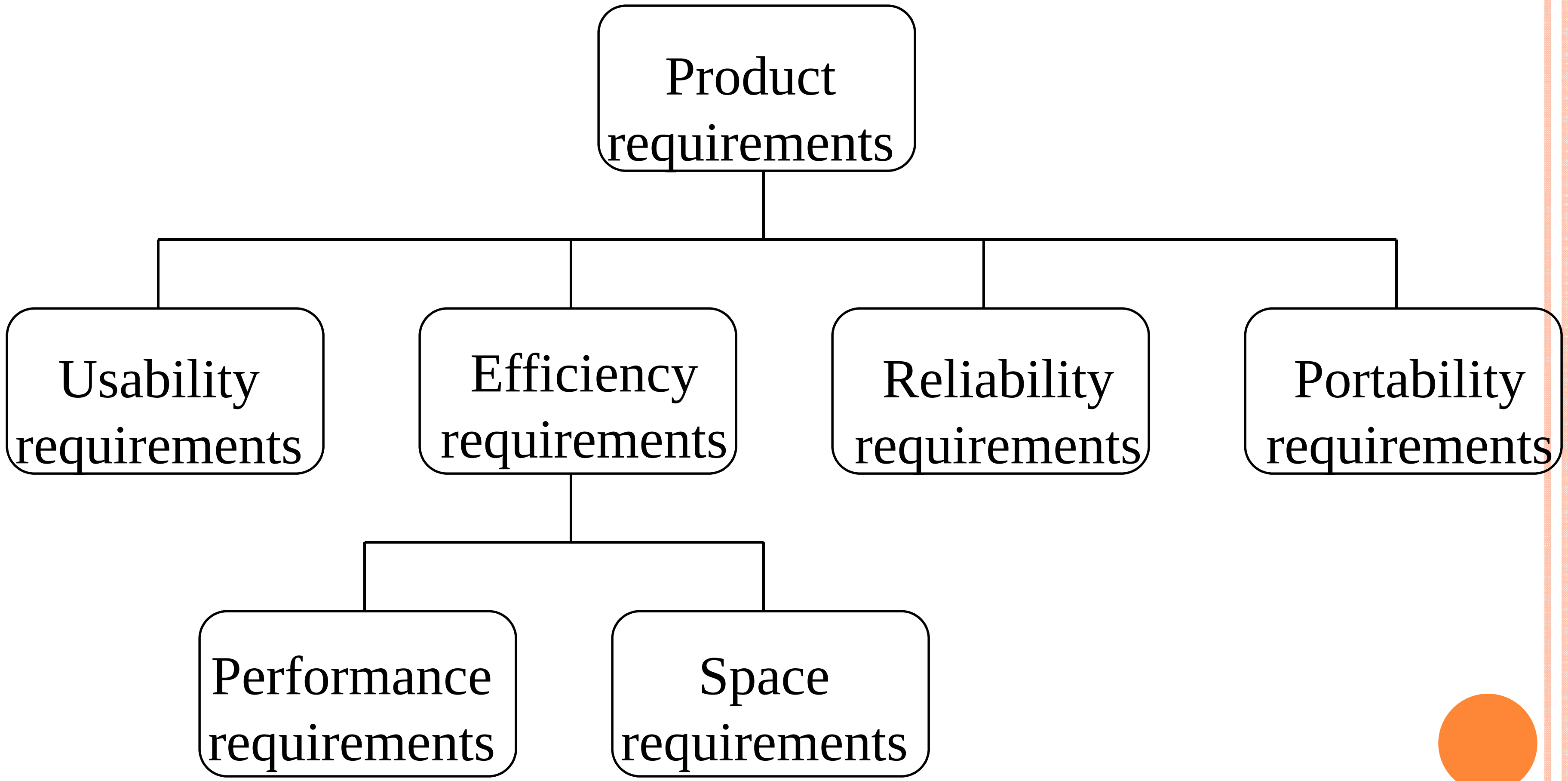
Non-Functional Requirements - 3

- For example, if an aircraft system does not meet reliability requirements, it will not be certified as 'safe'
- If a real-time control system fails to meet its performance requirements, the control functions will not operate correctly

Non-Functional Requirements - 4



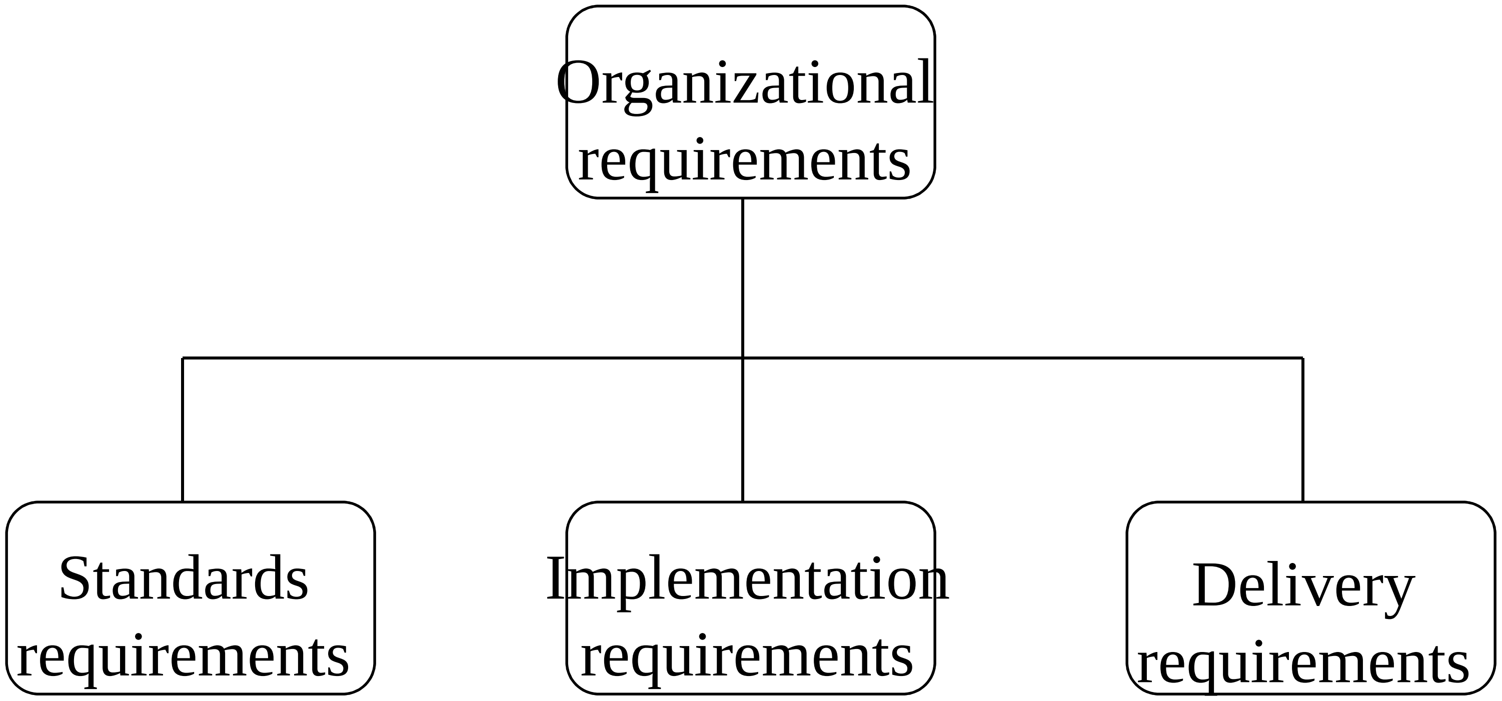
Product Requirements - 1



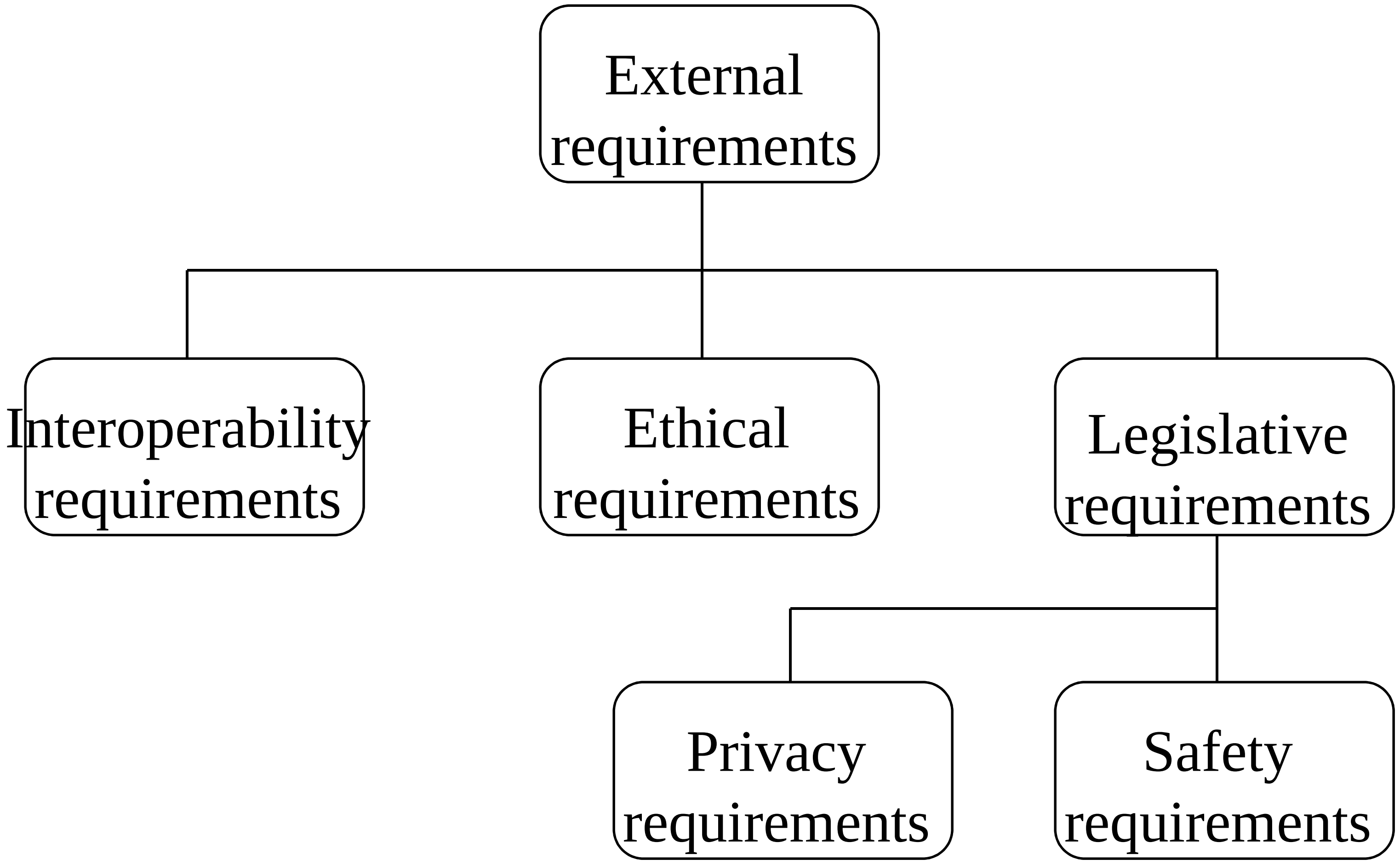
Product Requirements - 2

- The system shall allow one hundred thousand hits per minute on the website
- The system shall not have down time of more than one second for continuous execution of one thousand hours

Organizational Requirements - 1



External Requirements - 1



External Requirements - 2

- The system shall not disclose any personal information about members of the library system to other members except system administrators
- The system shall comply with the local and national laws regarding the use of software tools

Observations on Non-Functional Requirements - 1

- Non-functional requirements are written to reflect general goals for the system. Examples include:
  - **Ease of use**
  - **Recovery from failure**
  - **Rapid user response**

Observations on Non-Functional Requirements - 2

- Goals are open to misinterpretation
- Objective verification is difficult
- Distinction between functional and non-functional is not always very clear
- Non-functional requirements should be written in a quantitative manner as much as possible, which is not always easy for customers
- For some goals, there are no quantitative measures, e.g., maintainability
- Goals can be useful to designers and developers, as they give clues to them about priorities of the customers

Observations on Non-Functional Requirements - 3

- Chances of conflicts within non-functional requirements are fairly high, because information is coming from different stakeholders. For example, different stakeholders can give different response times or failure tolerance levels, etc.
- Some negotiations must be done among different stakeholders, to achieve an agreement in these situations
- Non-functional requirements should be highlighted in the requirements document, so that they can be used to build the architecture of the software product

Domain Requirements - 1

- Requirements that come from the application domain and reflect fundamental characteristics of that application domain
- Can be functional or non-functional
- These requirements, sometimes, are not explicitly mentioned, as domain experts find it difficult to convey them. However, their absence can cause significant dissatisfaction

Domain Requirements - 2

- Domain requirements can impose strict constraints on solutions. This is particularly true for scientific and engineering domains
- Domain-specific terminology can also cause confusion

Inverse Requirements

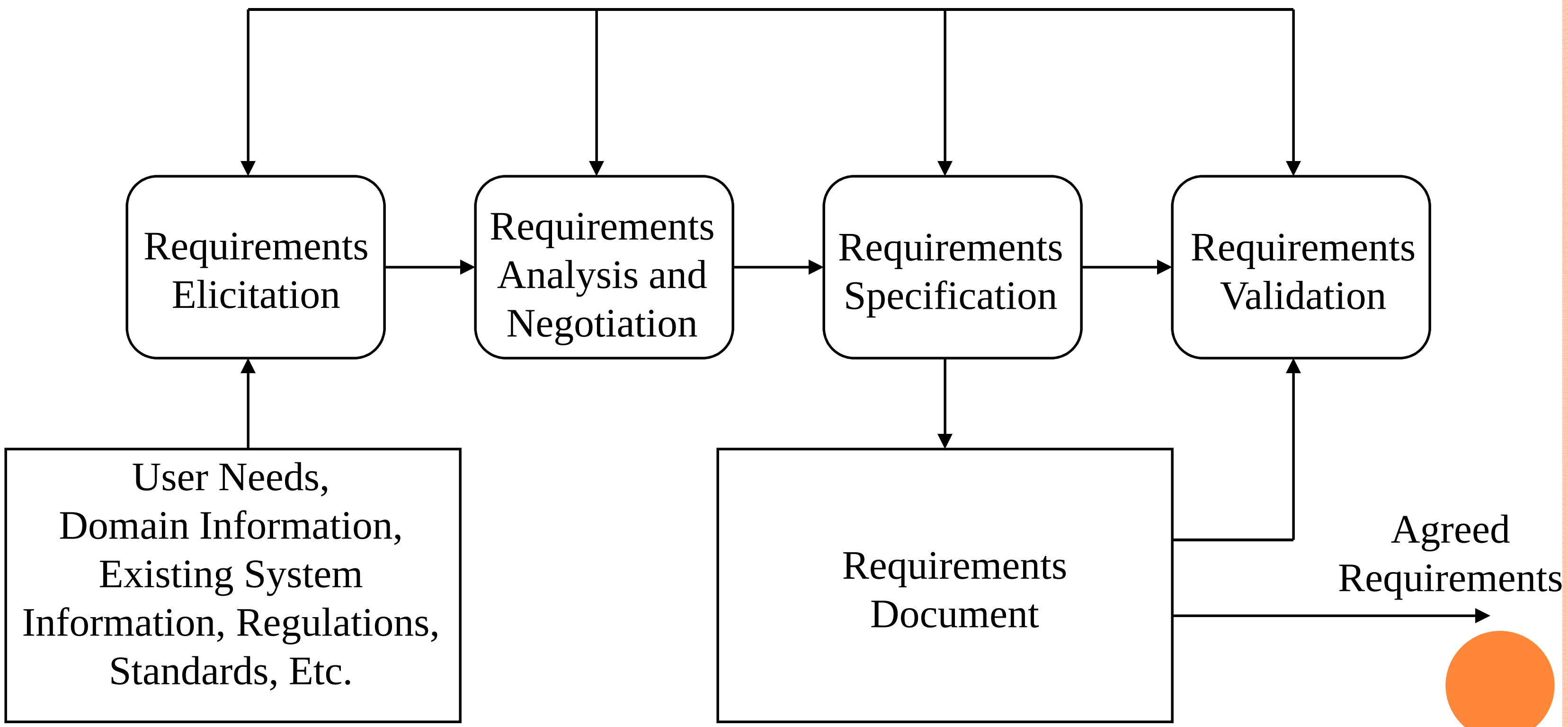
- They explain what the system shall **not** do. Many people find it convenient to describe their needs in this manner
- These requirements indicate the indecisive nature of customers about certain aspects of a new software product

Design and Implementation Constraints

- They are development guidelines within which the designer must work, which can seriously limit design and implementation options
- Can also have impact on human resources

Requirements Engineering Process

Requirements Engineering Activities


Requirements Elicitation
Requirements Analysis and Negotiation
Requirements Specification
Requirements Validation
User Needs, Domain Information, Existing System Information, Regulations, Standards, Etc.
Requirements Document
Agreed Requirements

Requirements Elicitation

- Determining the system requirements through consultation with stakeholders, from system documents, domain knowledge, and market studies
- Requirements acquisition or requirements discovery

Requirements Analysis and Negotiation - 1

- Understanding the relationships among various customer requirements and shaping those relationships to achieve a successful result
- Negotiations among different stakeholders and requirements engineers

Requirements Analysis and Negotiation - 2

- Incomplete and inconsistent information needs to be tackled here
- Some analysis and negotiation needs to be done on account of budgetary constraints

Requirements Specification

- Building a tangible model of requirements using natural language and diagrams
- Building a representation of requirements that can be assessed for correctness, completeness, and consistency

Requirements Document

- Detailed descriptions of the required software system in form of requirements is captured in the requirements document
- Software designers, developers and testers are the primary users of the document

Requirements Validation - 1

- Reviewing the requirements model for consistency and completeness
- This process is intended to detect problems in the requirements document, before they are used to as a basis for the system development

Requirements Validation - 2

- Requirements need to be validated for
  - consistency
  - testability
  - performance issues
  - conformance to local/national laws
  - conformance to ethical issues
  - conformance to company policies
  - availability of technology

Requirements Management

- Identify, control and track requirements and the changes that will be made to them
- It is important to trace requirements both ways
  - origin of a requirement
  - how is it implemented
- This is a continuous process

Requirements Problems - 1

- The requirements don’t reflect the real needs of the customer for the system
- Requirements are inconsistent and/or incomplete
- It is expensive to make changes to requirements after they have been agreed upon

Requirements Problems - 2

- There are misunderstandings between customers, those developing the system requirements, and software engineers developing or maintaining the system
- The requirements are written using complex conditional clauses (if A then B then C…), which are confusing

Requirements Problems - 3

- Terminology is used in a sloppy and inconsistent way
- The writers of the requirement assume that the reader has a specific knowledge of the domain or the system and they leave essential information out of the requirements document

Impact of Wrong Requirements - 1

- Difficult to check the requirements for errors and omissions
- Different interpretations of the requirements may lead to contractual disagreements between customer and the system developer

Impact of Wrong Requirements - 2

- When requirements are wrong, systems are late, unreliable and don’t meet customers needs
- This results in enormous loss of time, revenue, market share, and trust of customers
- Conformance to wrong requirements will result in a wrong system

Requirements Defects - 1

- All four categories of defects are found in requirements
  - Errors of commission
  - Errors of omission
  - Errors of clarity and ambiguity
  - Errors of speed and capacity
- If not prevented or removed, requirements defects usually flow downstream into design, code, and user manuals

Requirements Defects - 2

- Historically, requirements defects are most expensive and troublesome to eliminate
- Every effort should be made to eliminate requirements defects
- For requirements errors, prevention usually works better than removal

Requirements Defects - 3

- Errors of omission are most common among requirements defects
- Famous example is the Y2K problem

Requirements Defects - 4

- Second most common errors are those of clarity and ambiguity
- Primarily, because natural languages (like English) are used to state requirements, while such languages are themselves ambiguous
- For example: object

Requirements Defects - 5

- Errors of commission can also find their way into the requirements documents
- Performance errors can also be found in requirements documents

Prevention Vs Removal

- For requirements errors, prevention is usually more effective than removal
- Requirements inspections and prototyping play an important role defect removal

Changing/Creeping Requirements - 1

- Requirements will change, no matter what
- A major issue in requirements engineering is the rate at which requirements change once the requirements phase has "officially" ended
- This rate is on average 3% per month in the subsequent design phase, and will go down after that

Changing/Creeping Requirements - 2

- This rate should come down to 1% per month during coding
- Ideally, this should come down to no changes in testing

Defects and Creeping Requirements

- Studies have shown that very significant percentage of delivered defects can be traced back to creeping user requirements
- This realization can only be made, if defect tracking, requirements traceability, defect removal efficiency, and defect rates are all monitored for software projects

Damage Control of Creeping Requirements

- Following quality assurance mechanisms can limit the damage done by creeping requirements
  - Formal change management procedures
  - State-of-the-art configuration control tools
  - Formal design and code inspections

Problems with Natural Languages - 1

- Lack of clarity
- Requirements confusion
- Requirements amalgamation

PROBLEMS WITH NATURAL LANGUAGES - 2

- Natural language understanding relies on the specification readers and writers using the same words for same concept
- A natural language requirements specification is over-flexible. You can say the same thing in completely different ways

Problems with Natural Languages - 3

- It is not possible to modularize natural language requirements. It may be difficult to find all related requirements
  - To discover the impact of a change, every requirement have to be examined

Guidelines for Writing Requirements

Writing Requirements - 1

- Requirements specification should establish an understanding between customers and suppliers about what a system is supposed to do, and provide a basis for validation and verification

Writing Requirements - 2

- Typically, requirements documents are written in natural languages (like, English, Japanese, French, etc.)
- Natural languages are ambiguous
- Structured languages can be used with the natural languages to specify requirements
  - These languages cannot completely define requirements
  - They are not understandable by all stakeholders

Essentials for Writing Requirements - 1

- Requirements are read more often than they are written. Investing effort in writing requirements, which are easy to read and understand is almost always cost-effective
- Readers of requirements come from diverse backgrounds. Requirements writers should not assume that readers have the same background and knowledge as them

Essentials for Writing Requirements - 2

- Writing clearly and concisely is not easy. If you don’t allow sufficient time for requirements descriptions to be drafted, reviewed and improved, you will inevitably end up with poorly written requirements

Essentials for Writing Requirements - 3

- Different organizations write requirements at different levels of abstraction from deliberately vague product specifications to detailed and precise descriptions of all aspects of a system

Essentials for Writing Requirements - 4

- Level of detail needed is dependent on
  - Type of requirements (stakeholder or process requirements)
  - Customer expectations
  - Organizational procedures
  - External standards or regulations

Essentials for Writing Requirements - 5

- Writing good requirements requires a lot of analytic thought
- Specifying rationale of requirement is one way to encourage such thought

Guidelines for Writing Requirements - 1

- Define standard templates for describing requirements
- Use language simply, consistently, and concisely
- Use diagrams appropriately

Use of Standard Templates

- Define a set of standard format for different types of requirements and ensure that all requirement definitions adhere to that format
- Standardization means that omissions are less likely and makes requirements easier to read and check

Using Simple Language - 1

- Use language consistently. In particular, distinguish between mandatory and desirable requirements. It is usual practice to define mandatory requirements using '*shall*' and desirable requirements using '*should*'. Use '*will*' to state facts or declare purpose

Using Simple Language - 2

- Use short sentences and paragraphs, using lists and table
- Use text highlighting to pick out key parts of the requirements

Using Appropriate Diagrams

- Use diagrams to present broad overviews and show relationships between entities
- Avoid complex diagrams

Guidelines for Writing Requirements - 2

- Supplement natural language with other descriptions of requirements
- Specify requirements quantitatively

Using Other Descriptions of Requirements

- If readers are familiar with other types of descriptions of requirements (like equations, etc.) then use those
- Particularly applicable to scientific and engineering domains
- Don't try to write everything in natural language

Specify Requirements Quantitatively

- Specify requirements quantitatively wherever possible
- This is applicable to properties of system, such as reliability or performance
- Recollect our discussion on metrics for non-functional requirements

Additional Guidelines for Writing Requirements - 1

- State only requirement per requirement statement
- State requirements as active sentences
- Always use a noun or a definite pronoun when referring to a thing
- Do not use more than one conjunction when writing requirements statements

Additional Guidelines for Writing Requirements - 2

- Avoid using weak words and phrases. Such words and phrases re generally imprecise and allow the expansion or contraction of requirements beyond their intent

Examples of Words to be Avoided

- About, adequate, and/or, appropriate, as applicable, as appropriate, desirable, efficient, etc., if practical, suitable, timely, typical, when necessary

Additional Guidelines for Writing Requirements - 3

- State the needed requirements without specifying how to fulfill them
- Write complete statements
- Write statements that clearly convey intent

Requirements Document - 1

- The requirements document is a formal document used to communicate the requirements to customers, engineers and managers
- It is also known as software requirements specifications or SRS
- Requirements documents are usually written in natural languages (like, English, Japanese, French, etc.), which are ambiguous in themselves

Requirements Document - 2

- Functional requirements
- Non-functional requirements
  - Some of these are quality attributes of a software product
- Definitions of other systems which the system must integrate with

Requirements Document - 3

- Information about the application domain of the system, e.g., how to carry out particular types of computation
- Constraints on the process used to develop the system

Requirements Document - 4

- It should include both the user requirements for a system and a detailed specification of the system requirements
- In some cases, the user and system requirements may be integrated into one description, while in other cases user requirements are described before (as introduction to) system requirements

Requirements Document - 5

- For software systems, the requirements document may include a description of the hardware on which the system is to run
- The document should always include an introductory chapter which provides an overview of the system and the business needs

Requirements Document - 7

- A glossary should also be included to document technical terms
- And because multiple stakeholders will be reading documents and they need to understand meanings of different terms
- Also because stakeholders have different educational backgrounds

Requirements Document - 8

- Structure of requirements document is also very important and is developed on the basis of following information
  - Type of the system
  - Level of detail included in requirements
  - Organizational practice
  - Budget and schedule for RE process

What Should Not be Included in SRS?

- Project requirements (for example, staffing, schedules, costs, milestones, activities, phases, reporting procedures)
- Designs
- Product assurance plans (for example, configuration management plans, verification and validation plans, test plans, quality assurance plans)

Users of Requirements Documents

- System customers
- Managers
- System engineers
- System test engineers
- System maintenance engineers

Requirements for Requirements Document - 1

- It should specify only external behavior
- It should specify constraints on the implementation
- It should be easy to change
- It should serve as a reference tool for system maintainers
- It should record forethought about the lifecycle of the system
- It should characterize acceptable responses to undesired events

IEEE/ANSI Standard 830-1993

1. Introduction
2. General description
3. Specific requirements
4. Appendices
5. Index

Quality Attributes of Requirements Document - 1

- Correct
- Unambiguous
- Complete
- Verifiable
- Consistent
- Understandable by customer
- Modifiable
- Traced
- Traceable
- Design independent
- Annotated
- Concise
- Organized

Correct

- An SRS is correct if and only if every requirement stated therein represents something required of the system to be built

Unambiguous

- An SRS is unambiguous if and only if every requirement stated therein has only one interpretation
- At a minimum all terms with multiple meanings must appear in a glossary
- All natural languages invite ambiguity

Example of Ambiguity

- "Aircraft that are nonfriendly and have an unknown mission or the potential to enter restricted airspace within 5 minutes shall raise an alert"
- Combination of "and" and "or" make this an ambiguous requirement

Complete - 1

- An SRS is complete if it possesses the following four qualities
  - Everything that the software is supposed to do is included in the SRS
  - Definitions of the responses of the software to all realizable classes of input data in all realizable classes of situations is included

Complete - 2

- All pages are numbered; all figures and tables are numbered, named, and referenced; all terms and units of measure are provided; and all referenced material and sections are present
- No sections are marked “To Be Determined (TBD)

VERIFIABLE

- An SRS is verifiable if and only if every requirement stated therein is verifiable.
- A requirement is verifiable if and only if there exists some finite cost effective process with which a person or machine can check that the actual as-built software product meets the requirement

Consistent - 1

- An SRS is consistent if and only if:
  - No requirement stated therein is in conflict with other preceding documents, such as specification or a statement of work
  - No subset of individual requirements stated therein conflict.
- Conflicts can be any of the following
  - Conflicting behavior
  - Conflicting terms
  - Conflicting characteristics
  - Temporal inconsistency

Understandable by Customers

- Primary readers of SRS in many cases are customers or users, who tend to be experts in an application area but are not necessarily trained in computer science

Modifiable

- An SRS is modifiable if its structure and style are such that any necessary changes to the requirements can be made easily, completely, and consistently
- Existence of index, table of contents, cross-referencing, and appropriate page-numbering
- This attribute deals with format and style of SRS

Traced

- An SRS is traced if the origin of its requirements is clear.
- That means that the SRS includes references to earlier supportive documents

Traceable

- An SRS is traceable if it written in a manner that facilitates the referencing of each individual requirement stated therein

Techniques for Traceability

- Number every paragraph hierarchically
- Number every paragraph hierarchically and never include more than one requirement in any paragraph
- Number every requirement with a unique number in parentheses immediately after the requirement appears in the SRS
- Use a convention for indicating a requirement, e.g., use *shall* statement

Traced and Traceability - 1

- Backward-from-requirements traceability implies that we know why every requirement in the SRS exists.
- Forward-from-requirements traceability implies that we understand which components of the software satisfy each requirement

Traced and Traceability - 2

- Backward-to-requirements traceability implies that every software component explicitly references those requirements that it helps to satisfy
- Forward-to-requirements traceability implies that all documents that preceded the SRS can reference the SRS

Design Independent

- An SRS is design independent if it does not imply a specific software architecture or algorithm

Annotated

- The purpose of annotating requirements contained in an SRS is to provide guidance to the development organization

CONCISE

- The SRS that is shorter is better, given that it meets all characteristics

ORGANIZED

- An SRS is organized if requirements contained therein are easy to locate.
- This implies that requirements are arranged so that requirements that are related are co-related

Phrases to Look for in an SRS - 1

- Always, Every, All, None, Never
- Certainly, Therefore, Clearly, Obviously, Evidently
- Some, Sometimes, Often, Usually, Ordinarily, Customarily, Most, Mostly
- Etc., And So Forth, And So On, Such As

Phrases to Look for in an SRS - 2

- Good, Fast, Cheap, Efficient, Small, Stable
- Handled, Processed, Rejected, Skipped, Eliminated
- If…Then…(but missing Else)

The Balancing Act

- Achieving all the preceding attributes in an SRS is impossible
- Once you become involved in writing an SRS, you will gain insight and experience necessary to do the balancing act
- There is no such thing as a perfect SRS

References

- ‘Requirements Engineering: Processes and Techniques’ by G. Kotonya and I. Sommerville, John Wiley & Sons, 1998
- ‘Software Requirements: Objects, Functions, and States’ by A. Davis, PH, 1993
- ‘Software Engineering Quality Practices’ by R. K. Kandt, Auerbach Publications, 2006

References

- Software Quality: Analysis and Guidelines for Success by Capers Jones
- Requirements Analysis and Specification by Alan M. Davis
- A Practitioner's Approach to Software Engineering by Roger Pressman
- Software Engineering 6th Edition, by I. Sommerville, 2000
- 'Software Engineering Quality Practices' by R. K. Kandt, Auerbach Publications, 2006

Thanks!