Software Requirement Engineering

Course Code – SE- 208

Reference Books:

1. Software Quality: Analysis and Guidelines for Success by Capers Jones
2. Software Assessments, Benchmarks, and Best Practices by Capers Jones
3. Customer-oriented Software Quality Assurance by Frank P. Ginac
4. Software Engineering by Sommerville
5. Software Engineering: A Practitioner's Approach by Roger S. Pressman
6. Requirements Engineering: Processes and Techniques by Kotonya and Sommerville
7. Inroads to Software Quality by Alka Jarvis and Vern Crandell
8. Software Requirements: Objects, States, and Functions by Alan M. Davis
9. Software Engineering Quality Practices by Ronald K. Kandt
10. High Quality Low Cost Software Inspections by Ronald A. Radice

Software Requirements

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
- There is a huge difference between wanted and needed and it should be kept in mind all the time

Software Requirements

- A complete description of what the software system will do without describing how it will do it is represented by the software requirements
- Software requirements are complete specification of the desired external behavior of the software system to be built
- They also represent External behavior of the system
- Software requirements may be:
  - ✓ Abstract statements of services and/or constraints
  - ✓ Detailed mathematical functions
- Software requirements may be:
  - ✓ 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

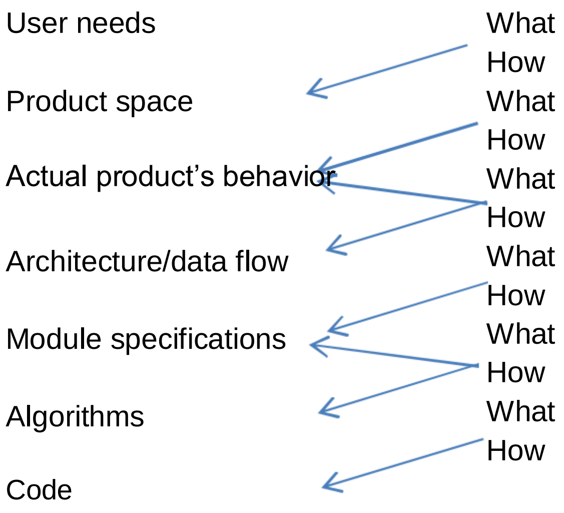
Sources of Requirements

- Stakeholders
  - People affected in some way by the system
- Documents
- Existing system
- Domain/business area

Levels of Software Requirements

- Stakeholders describe requirements at different levels of detail
  - "What versus How"
  - "One person's floor is another person's ceiling"

What versus How



User needs
Product space
Actual product’s behavior
Architecture/data flow
Module specifications
Algorithms
Code

What How What How What How What How What How What How

Importance of Software 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

Examples of Requirements

- 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
- 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.
- The system shall allow users to search for an item by title, author, or by International Standard Book Number
- The system’s user interface shall be implemented using a web browser
- The system shall support at least twenty transactions per second
- The system facilities which are available to public users shall be demonstrable in ten minutes or less

Kinds of Software Requirements

1. Functional requirements
2. Non-functional requirements
3. Domain requirements
4. Inverse requirements
5. Design and implementation constraints

1: Functional Requirements

- 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
- Sequencing and parallelism are also captured by functional requirements
- Abnormal behavior is also documented as functional requirements in the form of exception handling
- Functional requirements should be complete and consistent
- Customers and developers usually focus all their attention on functional requirements

Functional Requirements Examples:

- The system shall solve a quadratic equation using the following formula
    - ✓ $x = (-b \pm sqrt(b^2 - 4*a*c))/2*a$
- The user shall be able to search either the entire database of patients or select a subset from it (admitted patients, or patients with asthma, etc.)
- The system shall provide appropriate viewers for the user to read documents in the document store
- Every order shall be allocated a unique identifier (ORDER_ID) which the user shall use to access that order
- The system shall allow customers to return non-perishable items within fifteen days of the purchase. A customer must present the original sale receipt to return an item

Comments on Examples

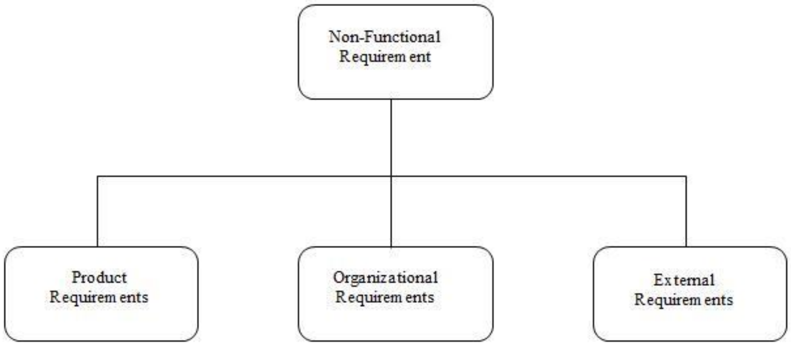
- Notice the level of detail in different requirements described above. Some are very detailed compared to others
- Notice the ambiguity in the requirement, which uses the term “appropriate viewers‟
- This requirement does not mention the formats of documents and types of viewers, which can be used
- Notice the ambiguity in the requirement for solving the quadratic equation. The requirement does **not speak about the possibility when the value of „a‟ is zero**
    - ✓ $x = (-b \pm sqrt(b^2 - 4*a*c)) / 2*a$
- Incomplete and ambiguous requirements are open to multiple interpretations and assumptions
- This can lead to the development of poor quality, or faulty, software products

2: Non-Functional Requirements

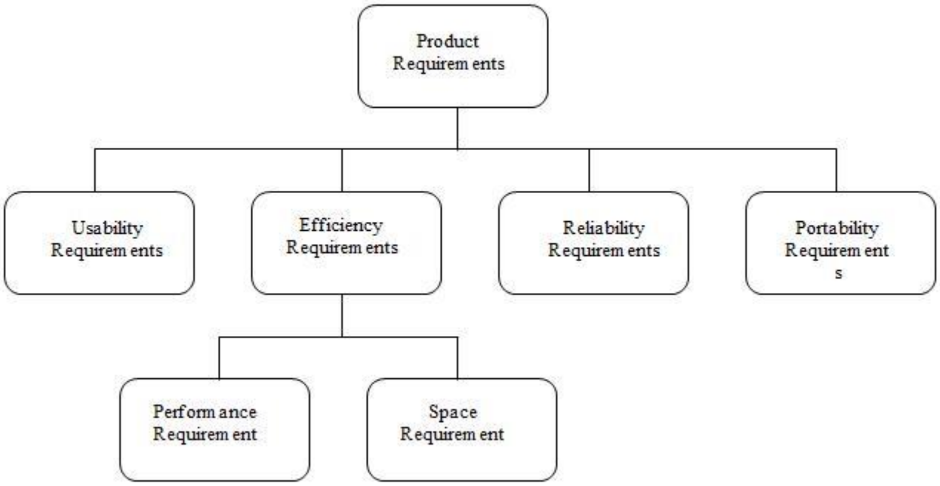
- Most non-functional requirements relate to the system as a whole. They include constraints on timing, performance, reliability, security, maintainability, accuracy, the development process, standards, etc.
- They are often more critical than individual functional requirements
- Capture the emergent behavior of the system, that is they relate to system as a whole
- Must be built into the framework of the software product
- Failure to meet a non-functional system requirement may make the whole system

unusable

- 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 arise through user needs, because of budget constraints, because of organizational policies, because of the need of interoperability with other software and hardware systems, or because of external factors such as safety regulations, privacy legislation, etc.



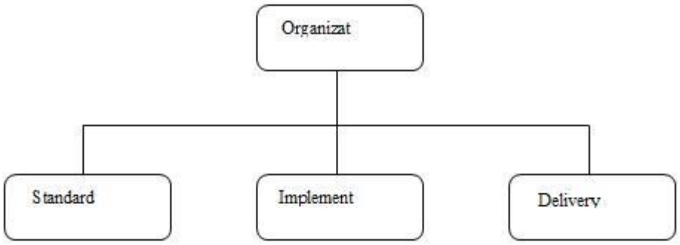
Product Requirements:



Product Requirements Examples

- 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

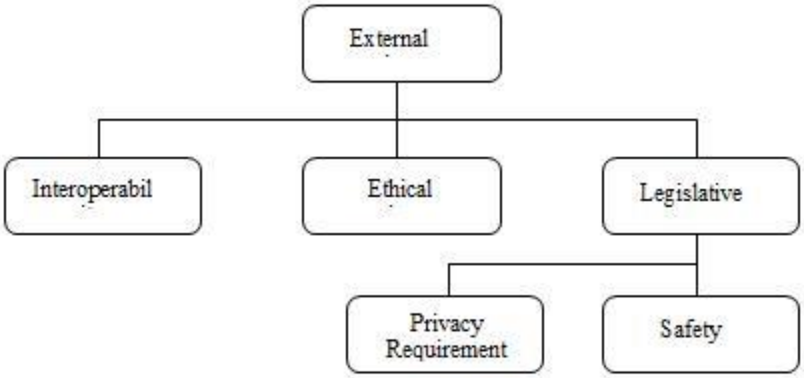
Organizational Requirements:



Organizational Requirements Examples

- The system development process and deliverable documents shall conform to the MIL-STD-2167A
- Any development work sub-contracted by the development organization shall be carried out in accordance with Capability Maturity Model

External Requirements:



External Requirements Examples

- 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

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

Non-Functional Requirements Discussion

- NFRs are very important to capture the emergent behavior of the system in these there major dimensions
- Product
  - Usability, reliability, portability, efficiency (performance, space)
- Organizational
  - Standards, implementation, delivery
- External
  - Interoperability, ethical, legislative (privacy, safety)

NFRs as Goals

- Non-functional requirements are sometimes written as general goals, which are difficult to verify
- They should be expressed quantitatively using metrics (measures) that can be objectively tested

Example: Goal converted into an NFR

- Goal (unverifiable)
  - The system should be easy to use by experienced controllers and should be organized in such a way that user errors are minimized

- Non-functional requirement (verifiable)
  - Experienced controllers shall be able to use all the system functions after a total of two hours' training. After this training, the average number of errors made by experienced users shall not exceed two per day

Software Metrics for Non-Functional Requirements (NFRs)

	Property	Measure
1	Speed	• Processed transactions/second • Response time • Screen refresh time
2	Size	• K bytes • Number of function points
3	Ease of use	• Training time • Number of help frames
4	Reliability	• Mean time to failure • Probability of unavailability • Rate of failure occurrence • Availability
5	Robustness	• Time to restart after failure • Percentage of events causing failure • Probability of data corruption on failure
6	Portability	• Percentage of target-dependent statements • Number of target systems

- With the help of these measures the NFRs can be verified quantitatively
- It should also be noted that the cost of quantitatively verifying each NFR may be very high

3: Domain Requirements

- Requirements that come from the application domain and reflect fundamental characteristics of that application domain
- These can be both the functional or non-functional requirements
- These requirements, sometimes, are not explicitly mentioned
- Domain experts find it difficult to convey domain requirements
- Their absence can cause significant dissatisfaction
- Domain requirements can impose strict constraints on solutions. This is particularly true for scientific and engineering domains
- Domain-specific terminology can also cause confusion
- Example: In a commission-based sales businesses, there is no concept of negative commission. However, if care is not taken novice developers can be lured into developing systems, which calculate negative commission
- Banking domain has its own specific constraints, for example, most banks do not allow over-draw on most accounts, however, most banks allow some accounts to be over-drawn

4: 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
- Example: The system shall not use red color in the user interface, whenever it is asking for inputs from the end-user

5: Design and Implementation Constraints

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

Design and Implementation Constraints Examples

- The system shall be developed using the Microsoft .Net platform
- The system shall be developed using open source tools and shall run on Linux operating system

Another view

- There also exists another view of requirements apart from different kinds of requirements we have studied so far.
  - Another view of requirements
  - There are some problems which occur in requirements, that are necessary to be identified and properly attended.
  - Problems in requirements

Another View of Requirements

- In general requirements can be viewed as
  - User/customer requirements OR System contract requirements

User/Customer Requirements

- Functional and non-functional requirements should be stated in natural language with the help of forms or simple diagrams describing the expected services of a system by the User under certain constraints
- These are understandable by users, who have no, or little, technical knowledge
- System design characteristics should be avoided as much as possible
- It is a good practice to separate user requirements from more detailed system requirements in a requirements document
- Including too much information in user requirements, constraints the system designers from coming up with creative solutions
- The rationale associated with requirements is very important. It helps in managing changes to requirements

System Contract Requirements

- Sets out the system services and constraints in detail
- May serve as the basis of contract for implementation of the system
- Should be complete and consistent
- They are used by the designers and developers as the starting point for system design
- They should be understood by technical staff of the customer organization and the development team
- In principle, these requirements should also state 'what' the system does, rather than 'how' it is implemented
- However, with the level of details needed to specify the system completely, it is not possible to exclude all design information
- An initial architecture of the system may be defined to help structure the requirements specification
- In most cases, systems interoperate with other systems
- Use of specific design may be included as an external requirement
- Natural language is often used to describe system requirements
- Some specification languages may be used with natural language, which add structure to specifications and reduce ambiguity
- Unified Modeling Language (UML) is a specification language, which has become the de-facto standard for modeling requirements

Requirements Problems

- 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
- There are misunderstandings between customers, those developing the system requirements, and software engineers developing or maintaining the system

Problems with Natural Languages

- Requirement specification in natural language pose some problems which include
  - ✓ Lack of clarity
  - ✓ Requirements confusion
  - ✓ Requirements amalgamation
- 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"
- 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

Impact of Wrong Requirements

- When requirements are wrong, systems are late, unreliable and don't meet customers needs
- his results in enormous loss of time, revenue, market share, and trust of customers

Processes and Process Models

Process

- A process is an organized set of activities, which transforms inputs to outputs
- We can use synonyms of process such as: procedure, method, course of action, etc.
- Processes are essential for dealing with complexity in real world
- Processes document the steps in solving a certain problem
- They allow knowledge to be reused
- They Allow people to apply the process in their peculiar but similar problems
- Examples of Processes
  - ✓ An instruction manual for operating a microwave oven
  - ✓ An instruction manual for assembling a computer or its parts
  - ✓ A procedure manual for operating a motor vehicle radio and CD player
  - ✓ A quality manual for software development.
  - ✓ Such a manual describes the processes, which should be used to assure the quality of the software

Software Processes

- Software engineering, as a discipline, has many processes
- These processes help in performing different software engineering activities in an organized manner
  1. Requires creativity
  2. Provides interactions between a wide range of different people
  3. Helps in engineering judgment
  4. Requires background knowledge
  5. Experiences
- Examples of Software Processes
  - ✓ Software engineering development process (SDLC)
  - ✓ Requirements engineering process
  - ✓ Design process
  - ✓ Quality assurance process
  - ✓ Change management process

Software Requirements Engineering Process

- Before discussing different aspects of requirements engineering process, let us discuss the concept of process models

Process Models

- A process model is a simplified description of a process presented from a particular perspective
- There may be several different models of the same process
- No single model gives a complete understanding of the process being modeled

Variations in Process Models

- A process model is produced on the anticipated need for that model. We may need
  - A model to help explain how process information has been organized
  - A model to help understand and improve a process
  - A model to satisfy some quality management standard

Types of Process Model

- Coarse-grain activity models
- Fine-grain activity models
- Role-action models
- Entity-relation models

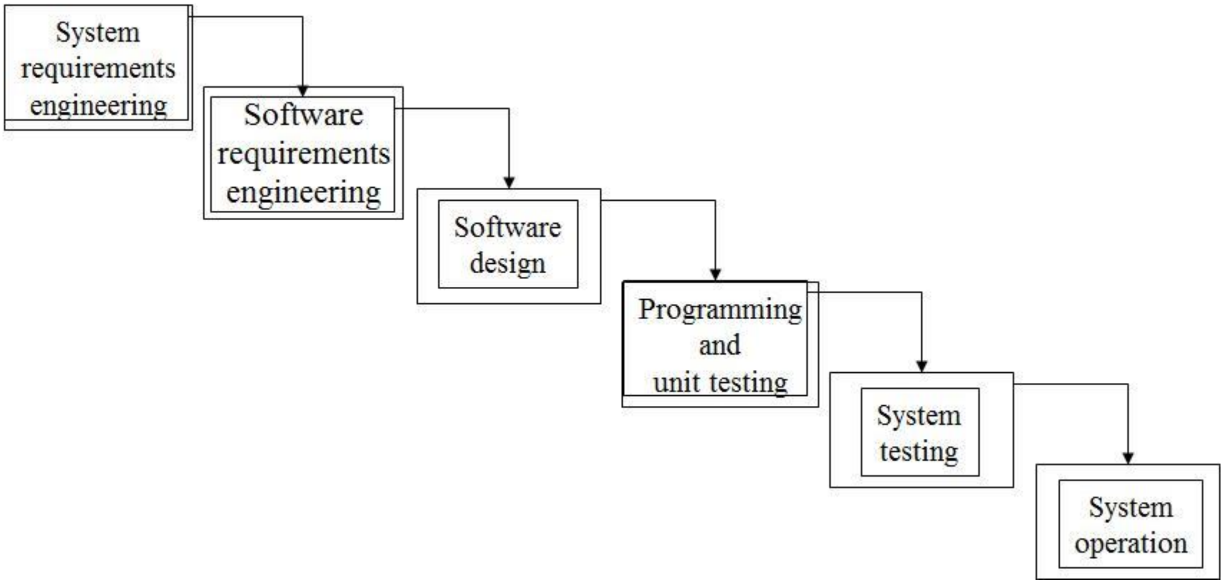
Coarse-grain Activity Model

- This type of model provides an overall picture of the process
- Describes the context of different activities in the process
- It doesn’t document how to enact a process

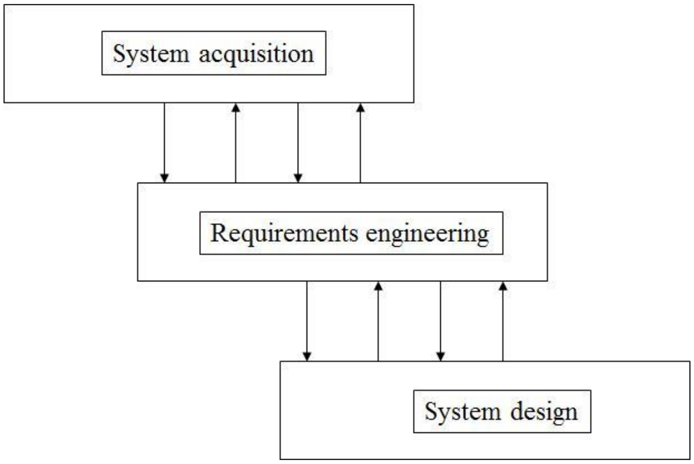
Context of Requirements Engineering

- Software requirements follow the “system requirements” and “system design”
- The primary goal is understanding
- Software requirements are followed by software design in a software development life cycle

Context of RE Process in Waterfall Model

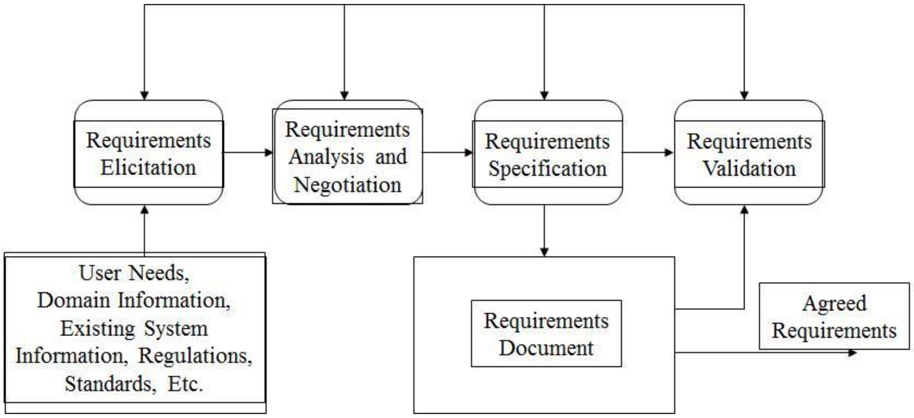

System requirements engineering
Software requirements engineering
Software design
Programming and unit testing
System testing
System operation

Another Perspective on Context of RE Process

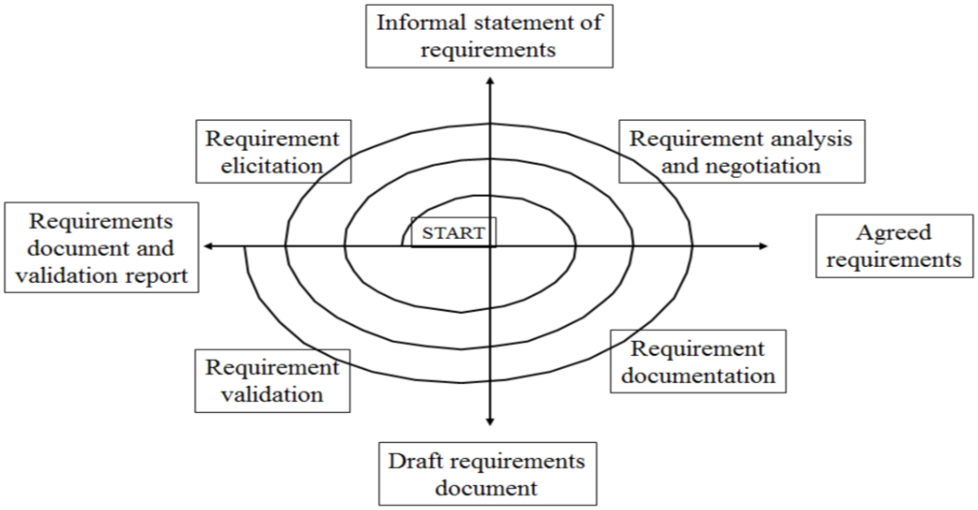


Coarse-grain Activity Model of the Requirements Engineering Process

- Requirements engineering process is an example of coarse-grain activity model


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

Spiral Model of RE Process


Informal statement of requirements
Requirement elicitation
Requirement analysis and negotiation
Requirements document and validation report
START
Agreed requirements
Requirement validation
Requirement documentation
Draft requirements document

Fine-grain Activity Models

- These are more detailed models of a specific process, which are used for understanding and improving existing processes
- We'll discuss some fine-grain processes within the general requirements engineering processes in later lectures

Role-action Models

- These are models, which show the roles of different people involved in the process and the actions which they take
- They are useful for process understanding and automation

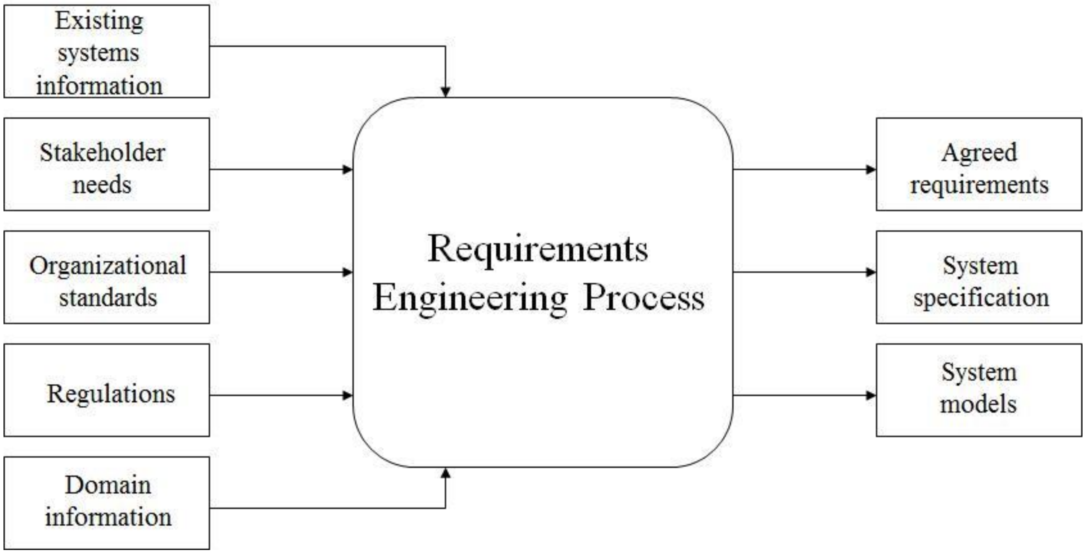
Entity-relation Models

- The models show the process inputs, outputs, and intermediate results and the relationships between them
- They are useful in quality management systems

Requirements Engineering Process

- The process(es) involved in developing system requirements is collectively known as Requirements Engineering Process

RE Process - Inputs and Outputs



RE Process – Inputs

It includes existing system information

- Information: It include the information about the functionality of systems to be replaced and the Information about other systems, which interact with the system being specified
- Stakeholder needs: Description of what system stakeholders need from the system to support their work

- Organizational standards: Standards used in an organization regarding system development practice, quality management, etc.
- Regulations: External regulations such as health and safety regulations, which apply to the system
- Domain information: General information about the application domain of the system
- Agreed requirements: A description of the system requirements, which is understandable by stakeholders and which has been agreed by them
- System specification: This is a more detailed specification of the system, which may be produced in some cases
- System models: A set of models such as a data-flow model, an object model, a process model, etc., which describes the system from different perspectives

RE Process Variability

- RE processes vary radically from one organization to another, and even within an organization in different projects
- Unstructured process rely heavily on the experience of the people, while systematic processes are based on application of some analysis methodology , but they still require human judgment

- RE processes vary radically from one organization to another, and even within an organization in different projects
- Unstructured process rely heavily on the experience of the people, while systematic processes are based on application of some analysis methodology , but they still require human judgment

Variability Factors

- There are four factors which count towards the variability of the Requirements Engineering Process
    1. Technical maturity: The technologies and methods used for requirements engineering vary from one organization to other
    2. Disciplinary involvement: The types of engineering and managerial disciplines involved in requirements vary from one organization to another
    3. Organizational culture: The culture of an organization has important effect on all business and technical processes
    4. Application domain: Different types of application system need different types of requirements engineering process.

RE Process

- Requirement Engineering Process has a formal starting and ending point in the overall software development life cycle.
- Begins
    - ✓ There is recognition that a problem exists and requires a solution
    - ✓ A new software idea arises
- Ends
    - ✓ With a complete description of the external behavior of the software to be built

- It is a continuous process in which the related activities are repeated until requirements are of acceptable quality
- It is one of the most critical processes of system development
- Based on the need of individual software projects and organizational needs, requirements engineering processes are tailored
- An important point to remember is that
- "There is no ideal requirements engineering process!"

Two Main Tasks of RE

- There are two main tasks which need to be performed in the requirements engineering process.
    1. Problem analysis: Analysis of a software problem
    2. Product description: Complete specification of the desired external behavior of the software system to be built. Also known as functional description, functional requirements, or specifications

Problem Analysis

- Problem analysis is the first and foremost task of requirements engineering process. It includes:
    - ✓ Brainstorming, interviewing, eliciting requirements
    - ✓ Identifying all possible constraints
    - ✓ Expansion of information
- Trading off constraints and organizing information
- Complete understanding should be achieved

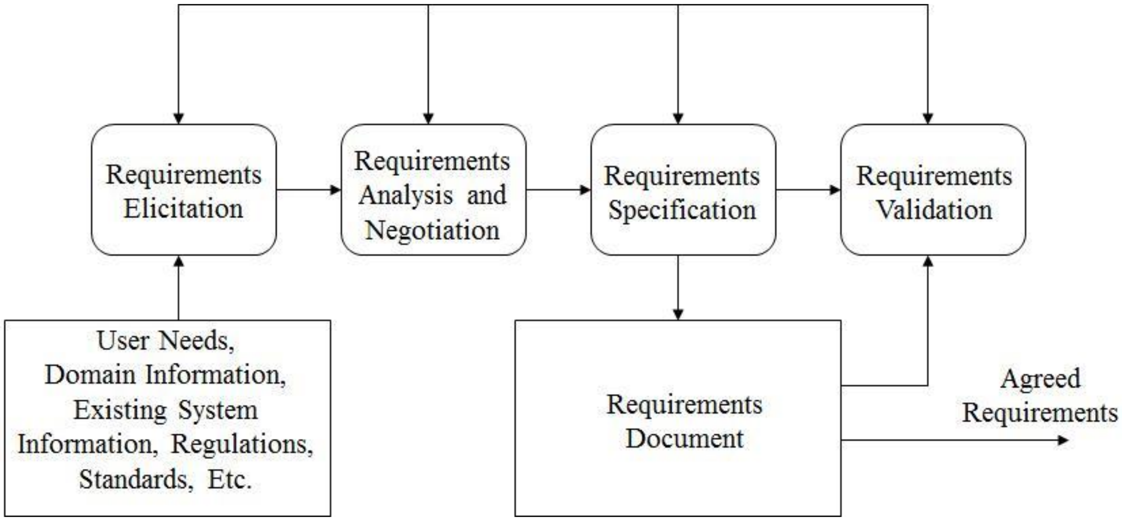
Product Description

- Product description is another task of requirements engineering process. In this task we:
    - ✓ Make decisions to define the external behavior of the software product
    - ✓ Organize ideas, resolve conflicting views, and eliminate inconsistencies and ambiguities

What Really Happens

- It should be kept in mind that: "Both problem analysis and product description run in parallel and iteratively throughout the 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

- Requirements elicitation activity is performed by
- 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

- Requirements analysis and negotiation activity is performed by
- Understanding the relationships among various customer requirements and shaping those relationships to achieve a successful result
- Negotiations among different stakeholders and requirements engineers
- Incomplete and inconsistent information needs to be tackled here
- Some analysis and negotiation needs to be done on account of budgetary constraints

Requirements Specification

- Requirements specification includes
- 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

- It involves reviewing the requirements model for consistency and completeness

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

Requirements Management

- Although, it is not shown as a separate activity in RE Process, it is performed through out the requirements engineering activities.
- Requirements management asks to identify, control and track requirements and the changes that will be made to them

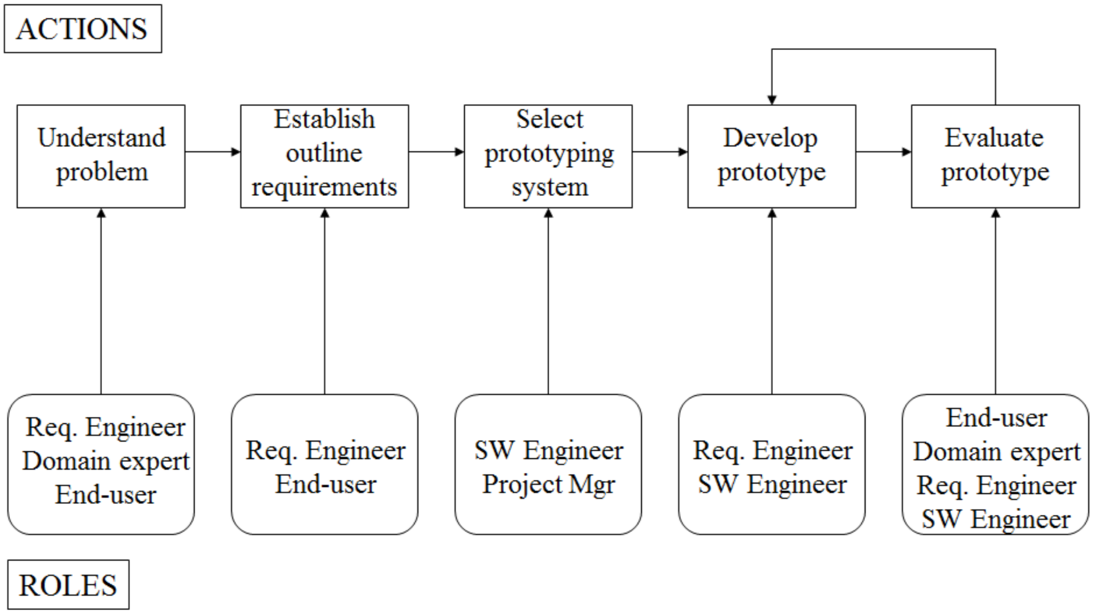
Who are Actors?

- Actors in a process are the people involved in the execution of that process
- Actors are normally identified by their roles rather than individually, e.g., project manager, purchasing director, and system engineer

Actors in the RE Process

- Requirements engineering involves people who are primarily interested in the problem to be solved (end-users, etc) as well as people interested in the solution (system designers, etc.)
- Another group of people, such as health & safety regulators, and maintenance engineers may be effected by the existence of the system
- Role-action diagrams are process models which show the actors associated with different process activities
- They document the information needs of different people involved in the process
- They use model of prototype software system as part of requirements elicitation process

Role-Action Diagram for Software Prototyping



Role Descriptions

Role	Description
Domain Expert	Responsible for proving information about the application domain and the specific problem in that domain, which is to be solved
System End-user	Responsible for using the system after delivery
Requirements Engineer	Responsible for eliciting and specifying the system requirements
Software Engineer	Responsible for developing the prototype software system
Project Manager	Responsible for planning and estimating the prototyping project

Human and Social Factors

- Requirements engineering processes are dominated by human, social and organizational factors because they always involve a range of stakeholders from different backgrounds and with different individual and organizational goals
- System stakeholders may come from a range of technical and non-technical background and from different disciplines

Stakeholder Types

- Software engineers
- System end-users
- Managers of system end-users
- External regulators
- Domain experts

Factors Influencing Requirements

- Personality and status of stakeholders
- The personal goals of individuals within an organization
- The degree of political influence of stakeholders within an organization

Process Support

- One way to minimize errors in the requirements engineering is to use process models and to use CASE tools
- The most mature CASE tools support well-understood activities such as programming and testing and the use of structured methods
- Support for requirements engineering is still limited because of the informality and the variability of the process

CASE Tools for RE

- Modeling and validation tools support the development of system models which can be used to specify the system and the checking of these models for completeness and consistency
- Management tools help manage a database of requirements and support the management of changes to these requirements

Process Improvement

- Process improvement is concerned with modifying processes in order to meet some improvement objectives
- Improvement objectives
  - ✓ Quality improvement
  - ✓ Resource reduction
  - ✓ Schedule reduction

Planning Process Improvement

- Some important questions arise:
- What are the problems with current processes?
- What are the improvement goals?
- How can process improvement be introduced to achieve these goals?
- How should process improvements be controlled and managed?

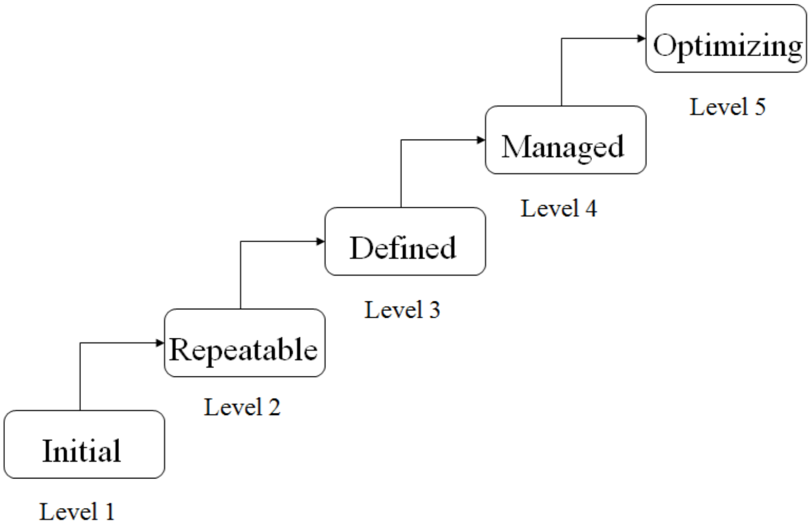
RE Process Problems

- Lack of stakeholder involvement
- Business needs not considered
- Lack of requirements management
- Lack of defined responsibilities
- Stakeholder communication problems
- Over-long schedules and poor quality requirements documents

Process Maturity

- Process maturity can be thought of as the extent that an organization has defined its processes, actively controls these processes and provides systematic human and computer-based support for them
- **The SEI"s Capability Maturity Model is a framework for** assessing software process maturity in development organizations

Capability Maturity Model



CMM Level 1: Initial

- Organizations have an undisciplined process and it is left to individuals that how to manage the process and which development techniques to use

CMM Level 2: Repeatable

- Organizations have basic cost and schedule management procedures in place. They are likely to be able to make consistent budget and schedule predictions for projects in the same application area

CMM Level 3: Defined

- The software process for both management and engineering activities is documented, standardized and integrated into a standard software process for the organization

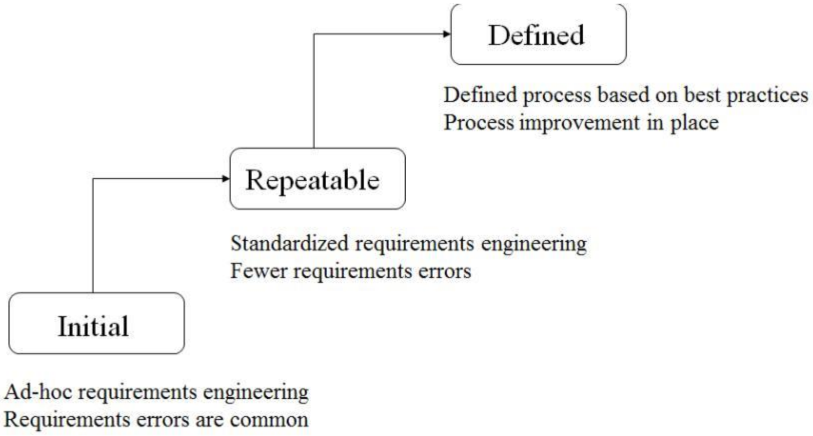
CMM Level 4: Managed

- Detailed measurements of both process and product quality are collected and used to control the process

CMM Level 5: Optimizing

- The organization has a continuous process improvement strategy, based on objective measurements, in place

RE Process Maturity Model


Defined
Defined process based on best practices
Process improvement in place
Repeatable
Standardized requirements engineering
Fewer requirements errors
Initial
Ad-hoc requirements engineering
Requirements errors are common

Initial RE Process Maturity Level

- There is no defined RE process.
- It suffer from requirements problems such as requirements volatility, unsatisfied stakeholders and high rework costs.
- It is dependent on individual skills and experience
- Defined standards for requirements documents, policies and procedures for requirements management
- Defined RE process based on good practices and techniques. Active process improvement process is in place

Best Practices for RE Process Improvement

- RE processes can be improved by the systematic introduction of best requirements engineering practices
- Each improvement cycle identifies best practice guidelines and works to introduce them in an organization
- Best practices will be discussed throughout the semester

Requirements Engineering Costs

- About fifteen percent (15%) of system development costs
- However, if the requirements engineering process is not executed properly, this cost can increase substantially

Social and Cultural Issues in Requirements Engineering

- Some aspects of the requirements engineering process deal with social and cultural issues
- What is the best way to deal with these issues?
- Some think that these issues fall outside the scope of requirements engineering process, and fall under management, interpersonal skills, or ethics

Social Issues in RE

- Requirements engineering is a social process, as it involves interaction among clients, engineers, and other systems
- Requirements engineering is not an entirely formal process, because it involves discovering client needs and reconciling them with technical possibilities

Stakeholders in RE Process

- At least three major groups participate in requirements engineering process;
  - ✓ The client organization
  - ✓ The requirements team
  - ✓ The development team
- There may be other interested parties, e.g., regulatory authorities

Six Areas of Social Issues

- Within the client organization
- Within the requirements team
- Between the client and the requirements team
- Between the development and requirements teams
- Within the development team
- Between the development team and the client

Issues within the Client Organization

- In a large organization, there are usually competing divisions or groups, so the notion of 'the client' is not obvious
- Intended users of the system may be different people from the ones who interact with

the requirements team

- The users of the system should be brought into the requirement engineering process, as they hold the key of the eventual success of the software engineering project
- The requirement process reveals the problems within the client organization, which must be addressed by facilitating communication among different stakeholders
- The problems within the client organization must not be buried, as they effect the implementation of the project
- The new automated system may have profound impact on how the business is conducted or how information is classified within the organization
- Success of the project requires that every group within the organization understand different aspects of the new system
- Problems of tacit knowledge
  - ✓ Say-do problem

Issues within the Requirements Team

- How work is organized?
- What methods and notations are used?
- What team members think about organization and how jelled requirement team is?

Issues between Client Organization and Requirements Team

- Financial arrangements
- Ethical obligations
- Legal safeguards
- Personal relationships
- Denial of information
- Management of changes

Issues between Development and Requirement Teams

- Development team needs to work very closely with the requirements team to resolve inconsistencies and to get details
- In some cases, requirements team may be disbanded or assigned other tasks

Issues of Development Team

- Team members may be demoralized
- There may be high turn over rate
- The deadlines may slip
- Developers dislike documentation
- Development teams may have to communicate with clients directly, to gain better understanding of the project’s possibilities and limitations, both for initial development and maintenance

Cultural Issues in RE

- Advances in the internet and communication technologies has enabled customers and developers to collaborate with each other in geographically and temporally dispersed environments, There may be
  - ✓ Time zones differences
  - ✓ Language and terminology differences

- ✓ Religious and racial differences
- ✓ Ethical issues
- ✓ Political differences
- ✓ Differences in business environment

Example: A Billion

- Scientific community and US consider the following number to be a
  billion 1,00,00,00,000
- For the rest of the world, a billion is
  10,00,00,00,00,000

Differences in Time Zones

- Working hours of clients and developers may differ by eight hours or more
- Arranging phone calls and video conferences become a hassle as one party has to come to office very early or stay very late
- Analysts start assuming requirements

Language and Terminology Differences

- Clients and developers may speak different languages or different dialects
- Requirements errors are introduced by not understanding other partner's language and terminology properly
- People and government in the US, and worldwide scientific community consider the following number to be a billion
  1,00,00,00,000
- For the rest of the world, a billion is
  10,00,00,00,00,000
- Globally, people communicate with fellow citizens using sports lingo to convey certain situations and concepts, even in the business environment
- This can cause misunderstandings
- Use of the word 'hockey' in Pakistan and US means two different sports: 'field hockey' and 'ice hockey' respectively

Religious and Racial Differences

- Insensitive comments on religious and racial backgrounds of people involved in software engineering projects can become a major hindrance in the successful execution of the requirements engineering process

Ethical Issues

- Access to confidential client information
- Possibility of elimination of jobs
- Differences of opinions with the client on the project

Political Differences

- Differences in political ideologies and personal convictions can also lead to

unprofessional environment in the execution of the requirements engineering process
- Some people do not want to work on military software program

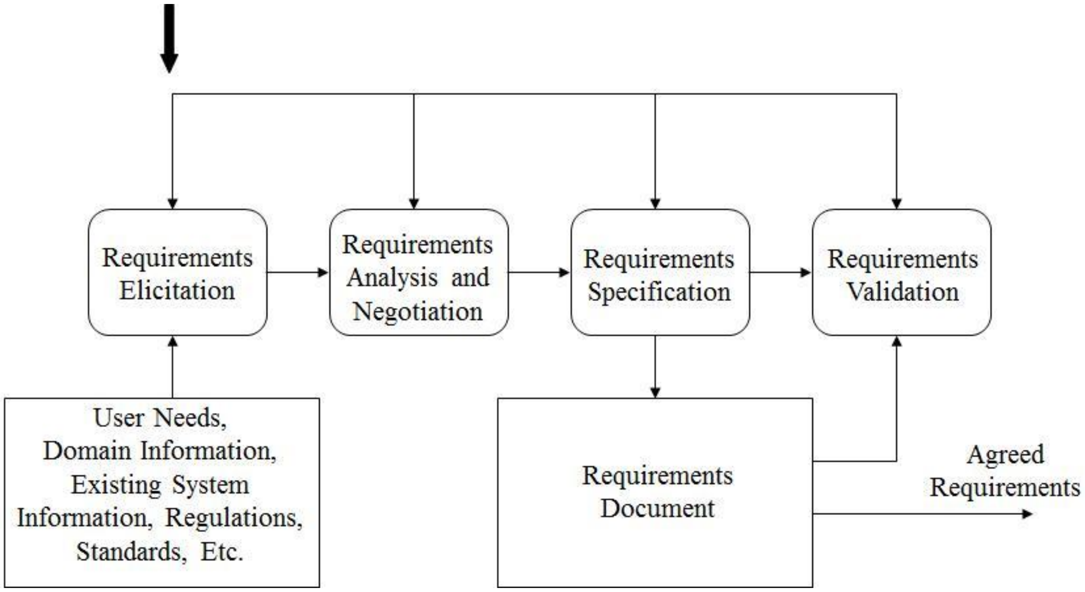
<u>Differences in Business Environments</u>

- Every society has its own culture within the business community, which must be understood for successful execution of the requirements engineering process

Addressing Social and Cultural Issues

- Understand social and cultural issues and differences
- Avoid judgmental comments and offensive remarks on un-related views and beliefs of others
- Create an environment of respect and professionalism
- Focus on discovering the needs of the customers
- Use state-of-the-art technology to facilitate activities in the requirements engineering process

Requirements Engineering Process


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

<u>RE Process 1</u>: Requirements Elicitation

- Elicit means to gather, acquire, extract, and obtain, etc.
- Requirements elicitation means gathering requirements or discovering requirements
- Activities involved in discovering the requirements for the system

Basics of Knowledge Acquisition

- These are the sources of knowledge acquisition
  - ✓ Reading
  - ✓ Listening
  - ✓ Asking
  - ✓ Observing

Requirements Elicitation Techniques

- Individual
- Group
- Modeling
- Cognitive

Problems in Requirements Elicitation

- Problems of scope
  - ✓ The boundary of the system is ill-defined
  - ✓ Unnecessary design information may be given
- Problems of understanding
  - ✓ Users have incomplete understanding of their needs
  - ✓ Users have poor understanding of computer capabilities and limitations
  - ✓ Analysts have poor knowledge of problem domain
  - ✓ User and analyst speak different languages
  - □ Ease of omitting “obvious” information
  - ✓ Conflicting views of different users
  - ✓ Requirements are often vague and untestable, e.g., “user-friendly” and “robust”
- Problems of volatility
  - ✓ Requirements evolve over time and hence there are some requirements which are bound to change during the system development process due to one reason or the other.

Contexts in Requirements Elicitation Process

- It is important to consider the context in which requirements are being elicited. Requirements elicitation process may be followed in the following contexts
  - ✓ Organization
  - ✓ Environment
  - ✓ Project
  - ✓ Constraints imposed by people
- Organization
  - ✓ Submitters of input
  - ✓ Users of output
  - ✓ Ways in which the new system change the business process
- Environment
  - ✓ Hardware and software
  - ✓ Maturity of the target system domain
  - ✓ Certainty of the target system’s interfaces to the larger system
  - ✓ The target system’s role in the larger system
- Project
  - ✓ The attributes of the different stakeholder communities, such as the end users, sponsors, developers, and requirements analysts. Examples of such attributes are:
    - ○ Management style
    - ○ Management hierarchy
    - ○ Domain experience
    - ○ Computer experience

- The constraints imposed by the people
  - ✓ They are involved in the elicitation process, e.g., managerial constraints concerning cost, time, and desired quality in the target system

Requirements Elicitation Guidelines

- Assess the business and technical feasibility for the proposed system
- Identify the people who will help specify requirements and understand their organizational bias
- Define the technical environment
- Identify "domain constraints" that limit the functionality or performance of the system
- Define one or more requirements elicitation methods (interviews, focus groups, team meetings)
- Solicit participation from many people so that requirements are defined from different points of view; be sure to identify the rationale for each requirement that is recorded
- Identify ambiguous requirements as candidates for prototyping
- Create usage scenarios to help customers/users better identify requirements

Ethnomethodology

- Looks for behaviors that may be different in a specific culture but which have the same underlying purpose or meaning
- Conversational analysis
- Measurement of body system functions
- Non-verbal behavior studies
- Detailed video analysis

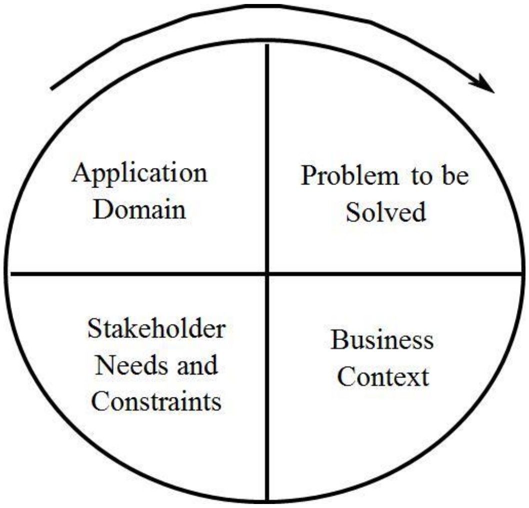
Requirements and Psychology

- Errors in statements can happen in two places
  - ✓ Perception of facts **–** reality
  - ✓ Linguistic representation of one of these perceptions **–** personal reality
- To remove these errors, requirements should be reviewed (during and after elicitation)

Use Case Modeling

- Define actors and black-box use cases
- The functional requirements of the system are defined in terms of use cases and actors
- The use case descriptions are a behavioral view

Components of Requirements Elicitation



Dimensions to Requirements Elicitation

- Application domain understanding
- Problem understanding
- Business understanding
- Understanding the needs and constraints of system stakeholders
- Application domain understanding
  - ✓ Knowledge of the general area where the system is applied
- Problem understanding
  - ✓ The details of the specific customer problem where the system will be applied must be understood
- Business understanding
  - ✓ Understand how systems interact and contribute to overall business goals
- Understanding the needs and constraints of system stakeholders
  - ✓ Understand, in detail, the specific needs of people who require system support in their work

Elicitation and Analysis Processes

- Requirements elicitation and requirements analysis are closely linked processes

Requirements Elicitation Stages

- Objective setting
- Background knowledge acquisition
- Knowledge organization
- Stakeholder requirements collection

Objective Setting

- Overall organizational objectives should be established at this stage
- These include general goals of business, an outline description of the problem to be

solved and why the system may be necessary, and the constraints on the system such as budget, schedule, and interoperability constraints

Background Knowledge Acquisition

- Requirements engineers gather and understand background information
- This includes information about the organization where the system is to be installed, information about the application domain of the system, and information about any existing systems which are in use and which may be replaced

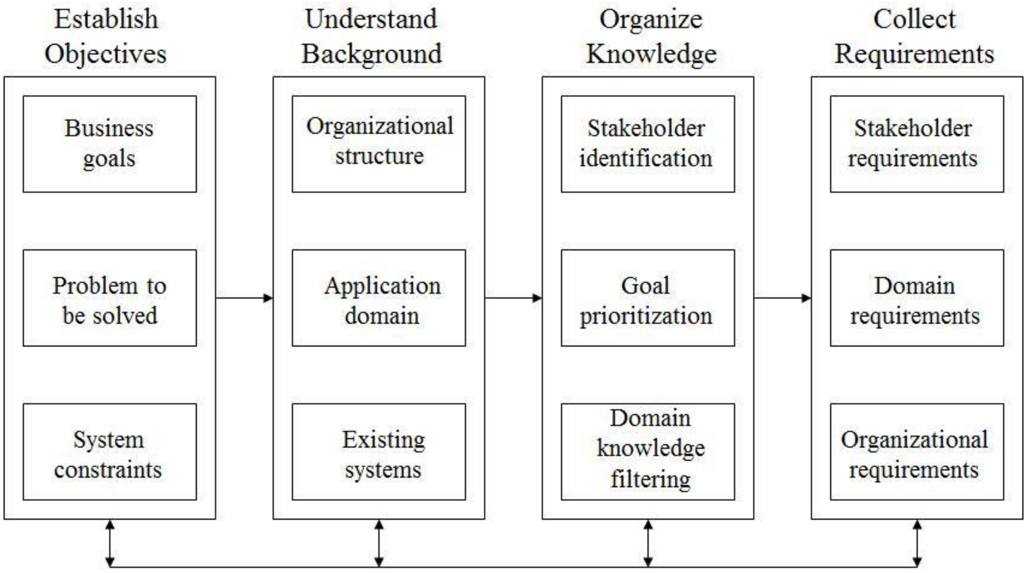
Knowledge Organization

- The large amount of knowledge which has been collected in previous stage must be organized and collated
- Identifying system stakeholders and their roles in the organization, prioritizing the goals of the organization and discarding domain knowledge which does not contribute directly to the system requirements

Stakeholder Requirements Collection

- It involves consulting system stakeholders to discover their requirements, and deriving requirements which come from the application domain and the organization which is acquiring the system

A General Requirements Elicitation Process



Comments on this Process

- It is an idealized process, while the reality of requirements elicitation tends to be much messier
- The activities are usually mixed up with each other

- If objective setting activities are not carried out, significant analysis problems occur, as no objective and business goals are available to prioritize requirements
- It is an idealized process, while the reality of requirements elicitation tends to be much messier
- The activities are usually mixed up with each other
- If objective setting activities are not carried out, significant analysis problems occur, as no objective and business goals are available to prioritize requirements
- The output from the requirements elicitation process should be a draft document which describes the system requirements, which is then analyzed to discover problems and conflicts in the requirements definition
- This process is followed by the requirements analysis process, which will be discussed in another lecture

Basics of Knowledge Acquisition

- Reading
- Listening
- Asking
- Observing
- Results in large volume of information, which must be organized to make it understandable

Knowledge Structuring Techniques

- Partitioning
  - ✓ Organization of knowledge into aggregation relationships, where requirements knowledge is described in terms of its parts
  - ✓ Booking system example: a booking record may be may be defined as a flight reference, source & destination of flight, the name & address of the passenger, fare, and date of travel
- Abstraction
  - ✓ Organization of knowledge according to general/specific relationships. Requirement knowledge is described by relating specific instances to abstract structures
  - ✓ Passenger abstraction may represent all classes of passengers (children, adults, full-fare paying, concessionary passengers, etc.)
- Projection
  - ✓ Organization of knowledge from several different perspectives or viewpoints
  - ✓ Booking system example: travel agents, airline management, check-in desk operators, passengers, a bookings database, etc.

Specific Elicitation Techniques

- Interviews
- Scenarios
- Observations and social analysis
- Requirements reuse

Interviews

- The requirements engineer or analyst discusses the system with different stakeholders and builds up an understanding of their requirements
- Interviews are less effective for understanding the application domain and the organizational issues due to terminology and political factors

Types of Interviews

- Closed interviews
  - ✓ The requirements engineer looks for answers to a pre-defined set of questions
- Open interviews
  - ✓ There is no predefined agenda and the requirements engineer discusses, in an open-ended way, what stakeholders want from the system

Interviewing Essentials

- Interviewers must be open-minded and should not approach the interview with pre-conceived notions about what is required
- Stakeholders must be given a starting point for discussion. This can be a question, a requirements proposal or an existing system
- Interviewers must be aware of organizational politics - many real requirements may not be discussed because of their political implications

Interview Steps

- Prepare
- Conduct
  - ✓ Opening
  - ✓ Body
  - ✓ Closing
- Follow through

<u>Prepare for the Interview</u>

- Before developing questions
  - ✓ Define the purpose and objectives
  - ✓ Determine whether the interview should be conducted by one person or a team (define roles for team members)
  - ✓ Contact interviewee to arrange time, place, and logistics of the interview and outline the purpose and format
  - ✓ Obtain background information
- After contacting the interviewee
  - ✓ Develop the interview guide
  - ✓ List name and title of interviewee and date of the interview
  - ✓ List questions in the order you will ask them
  - ✓ Move from general to specific
  - ✓ Include open questions to elicit essay type response (e.g., Describe…, Tell me…, How…)
  - ✓ Include closed questions to obtain specific information (e.g., Who? How much? Where?)

Conduct the Interview

- Opening
    - ✓ Establish rapport and build trust and credibility ○ Make eye contact
        - ○ Shake hands
        - ○ Introduce yourself (and your team); provide information about role(s) in the interview process
    - ✓ Clarify purpose, time frame, and key objectives
    - ✓ Transition to the core of the interview by leading into the first question
- Body
    - ✓ Follow your interview guide as you ask questions; use probes to follow up on a Response
    - ✓ Be flexible and open-minded
    - ✓ Listen actively
    - ✓ Monitor your voice and body Language
    - ✓ Identify Interviewee's main concerns
    - ✓ Maintain rapport
    - ✓ Take accurate notes
    - ✓ Use silence and pauses
    - ✓ Ask for and obtain relevant documentation
    - ✓ Ask "catch-all" question at the end
- Closing
    - ✓ Summarize findings and link to purpose
    - ✓ Answer any questions the interviewee has
    - ✓ Determine and agree on next steps
    - ✓ Set next meeting, if necessary
    - ✓ Thank the interviewee for his/her input and for taking the time to meet with you

Follow Through

- Immediately after the interview, fill in your notes; be sure to jot down impressions and important ideas
- Review any documentation received from the interviewee
- Write an interview report, if necessary
- Follow up on leads obtained during the interview
    - ✓ Contact other potential interviewees
    - ✓ Research other data sources
- Follow up in agreed-upon next steps
- Send a thank you note to the interviewee, if appropriate

Listening

- The art of listening is most important. You can best impress your client by listening and giving due attention to what the client or customer is saying
- This requires effort on part of the interviewer
- Listening Steps
    - ✓ Hear
        - ○ Listen to learn as much as you can so that you will know how to respond

- Give the speaker your undivided attention; don’t just wait for your turn to speak
- Concentrate on the message, not the person
- Don’t interrupt
- Tune out distractions such as interfering noises, wandering thoughts, and emotional reactions t**o** the speaker‟s message
- Suspend judgment about the message until you have heard all the facts
- Take notes on the speaker’s key points, if appropriate
- Learn to manage your own emotional filters, personal blinders, and biases, which can keep you from hearing what is really being said

✓ Interpret

- Observe the speaker’s nonverbal cues (gestures, facial expressions, and tone of voice) and factor them into your interpretation
- Listen for the attitudes and motives behind the words
- Listen for the speaker‟s needs and wants
- Put the message in a broader context
- Integrate what you’ve just heard into what you already know about the speaker or subject

✓ Respond

- Nonverbal Response to the Message
  - Make eye contact
  - Nod affirmatively
  - Use facial expressions and gestures to indicate that you are listening
- Verbal Response to the Message
  - Ask questions and probe to get more specific information and ensure understanding
  - Rephrase the message using different words to check the meaning
  - Make empathetic remarks that acknowledge you understand the speaker’s feelings, without offering opinions or judging him or her

✓ Evaluate

- Identify the main point of the message and its supporting evidence
- Clarify facts, perceptions, and opinions
- Distinguish between fact and opinion
- Group facts in like categories and logical order (importance, chronology)
- Base your opinion about the message on the facts
- Use the total message – the needs, the context, and the content – to follow through on what you hear

Brainstorming

- Facilitated application specification technique (FAST)
- Group activity
- All members are equal
- Off-site meeting location is preferred

Scenarios

- Scenarios are stories which explain how a system might be used. They should include
  - ✓ A description of the system state before entering the scenario
  - ✓ The normal flow of events in the scenario
  - ✓ Exceptions to the normal flow of events
  - ✓ Information about concurrent activities
  - ✓ A description of the system state at the end of the scenario
- Scenarios are examples of interaction sessions which describe how a user interacts with a system
- Discovering scenarios exposes possible system interactions and reveals system facilities which may be required

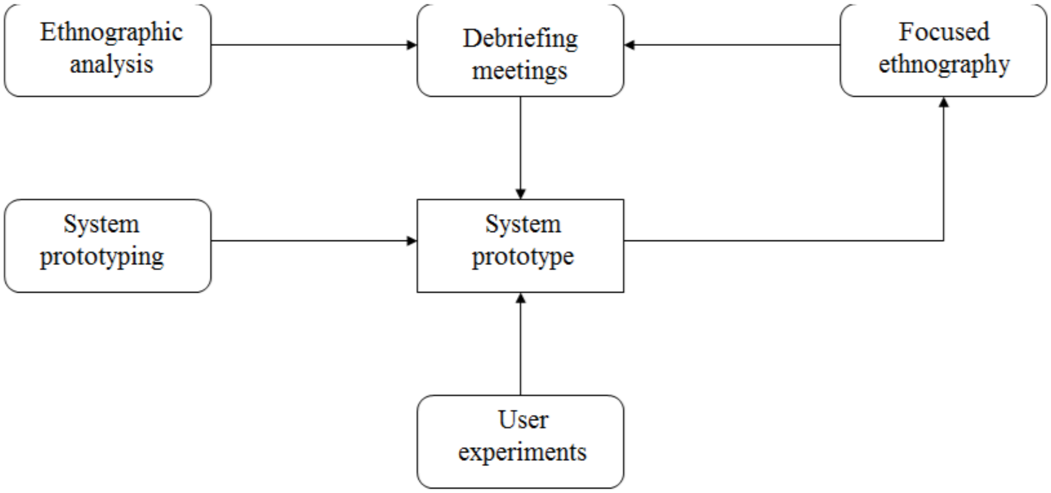
Scenarios and Use-Cases

- The term use-case (i.e., a specific case of system usage) is sometimes used to refer to a scenario
  - ✓ A use-case is a scenario
  - ✓ A scenario is a collection of use-cases. Therefore, each exceptional interaction is represented as a separate use-case
  - ✓ A use-case is a collection of scenarios

Observation and Social Analysis

- People often find it hard to describe what they do because it is so natural to them. Sometimes, the best way to understand it is to observe them at work
- Ethnography is a technique from the social sciences which has proved to be valuable in understanding actual work processes
- Actual work processes often differ from formal, prescribed processes
- An ethnographer spends an extended time observing people at work and building up a picture of how work is done

Ethnography in Requirements Elicitation



Ethnography Guidelines

- Assume that people are good at doing their job and look for non-standard ways of working
- Spend time getting to know the people and establish a trust relationship
- Keep detailed notes of all work practices. Analyze them and draw conclusions from them
- Combine observation with open-ended interviewing
- Organize regular de-briefing session where the ethnographer talks with people outside the process
- Combine ethnography with other elicitation techniques

Requirements Reuse

- Reuse involves taking the requirements which have been developed for one system and using them in a different system
- Requirements reuse saves time and effort as reused requirements have already been analyzed and validated in other systems
- Currently, requirements reuse is an informal process but more systematic reuse could lead to larger cost savings

Reuse Possibilities

- Where the requirement is concerned with providing application domain information
- Where the requirement is concerned with the style of information presentation. Reuse leads to a consistency of style across applications
- Where the requirement reflects company policies such as security policies

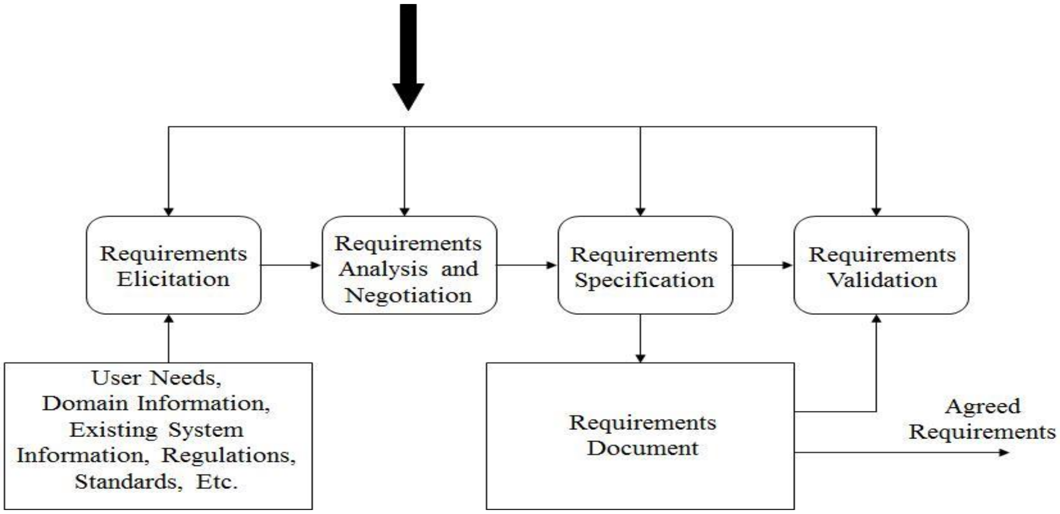
Prototyping

- A prototype is an initial version of a system which may be used for experimentation
- Prototypes are valuable for requirements elicitation because users can experiment with the system and point out its strengths and weaknesses. They have something concrete to criticize

Recap of Requirements Elicitation

- Requirements elicitation deals with discovering requirements for a software product
- It is an iterative process and consists of many activities including establishing objectives, understanding background, organizing knowledge, and collecting requirements
- Introduced the concept of elicitation and requirements elicitation process
- Basics of knowledge acquisition (reading, listening, asking, & observing)
- Knowledge acquisition techniques (individual, group, modeling, cognitive)
- Elicitation problems (scope, understandability, volatility)
- Context (organization, environment, project, constraints imposed by people)
- Guidelines for knowledge acquisition
- Discussed in detail some requirements elicitation techniques, especially interviews

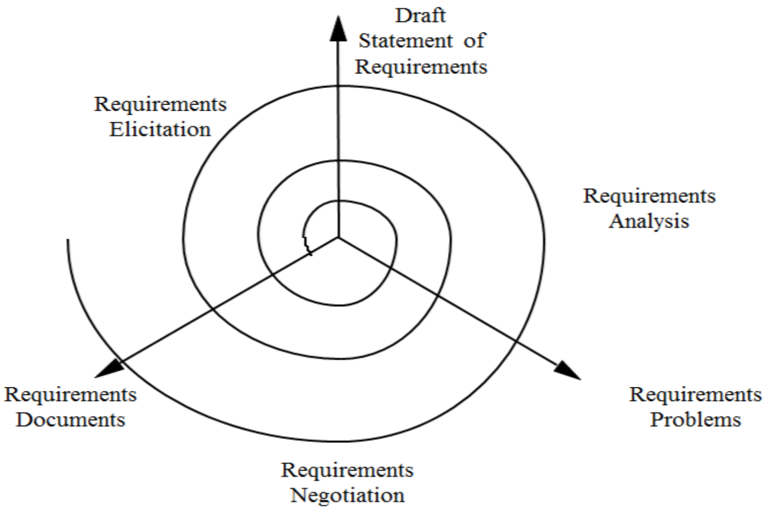
Requirements Engineering Process


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

RE Process 2: Requirements Analysis and Negotiation

- We'll discuss requirements analysis and negotiation separately, in order to understand them clearly and to appreciate that different skills are needed to perform them
- They are inter-leaved activities and join to form a major activity of the requirements engineering process
- The aim of requirements analysis is to discover problems with the system requirements, especially incompleteness and inconsistencies
- Some analysis is inter-leaved with requirements elicitation as problems are sometimes obvious as soon as a requirement is expressed
- Detailed analysis usually takes place after the initial draft of the requirements document is produced
- Analysis is concerned with incomplete set of requirements, which has not been discussed by stakeholders

Iterative Aspects of Elicitation, Analysis, and Negotiation



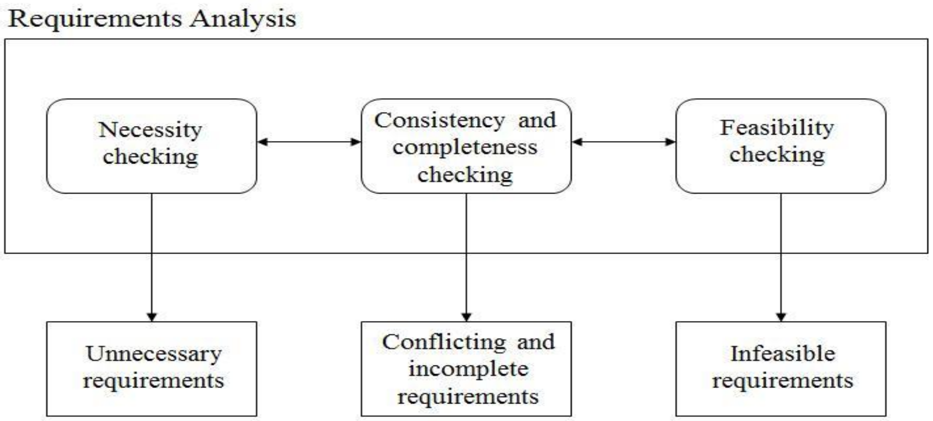
Comments on Requirements Analysis

- Analysts read the requirements, highlight problems, and discuss them in requirements review meetings
- This is a time-consuming and expensive activity
- Analysts have to think about implications of the draft statements of requirements
- People do not think in the same way and different analysts tackle the process in different ways
- It is not possible to make this activity a structured and systematic process
- It depends on the judgment and experience of process participants

Requirements Analysis Stages

- Necessity checking
  - ✓ The need for the requirement is analyzed. In some cases, requirements may be proposed which don't contribute to the business goals of the organization or to the specific problem to be addressed by the system
- Consistency and completeness checking
  - ✓ The requirements are cross-checked for consistency and completeness. Consistency means that no requirements should be contradictory; Completeness means that no services or constraints which are needed have been missed out
- Feasibility checking
  - ✓ The requirements are checked to ensure that they are feasible in the context of the budget and schedule available for the system development

Requirements Analysis Process



Analysis Techniques

- Analysis checklists
  - ✓ A checklist is a list of questions which analysts may use to assess each requirement
- Interaction matrices
  - ✓ Interaction matrices are used to discover interactions between requirements and to highlight conflicts and overlaps

Analysis Checklists

- Each requirement may be assessed against the checklist
- When potential problems are discovered, these should be noted carefully
- They can be implemented as a spreadsheet, where the rows are labeled with the requirements identifiers and columns are the checklist items
- The are useful as they provide a reminder of what to look for and reduce the chances that you will forget some requirements checks
- They must evolve with the experience of the requirements analysis process
- The questions should be general, rather than restrictive, which can be irrelevant for most systems
- Checklists should not include more than ten items, because people forget items on long checklists reading through a document. Example of analysis checklist

Checklist Items

- Premature design
- Combined requirements
- Unnecessary requirements
- Use of non-standard hardware
- Conformance with business goals
- Requirements ambiguity
- Requirements realism
- Requirements testability

Checklist Items Description

- Premature design
  - ✓ Does the requirement include premature design or implementation information?
- Combined requirements
  - ✓ Does the description of a requirement describe a single requirement or could it be broken down into several different requirements?
- Unnecessary requirements
  - ✓ **Is the requirement „gold plating‟? That is, is the** requirement a cosmetic addition to the system which is not really necessary
- Use of non-standard hardware
  - ✓ Does the requirement mean that non-standard hardware or software must be used? To make this decision, you need to know the computer platform requirements
- Conformance with business goals
  - ✓ Is the requirement consistent with the business goals defined in the introduction to the requirements document?
- Requirements ambiguity
  - ✓ Is the requirement ambiguous i.e., could it be read in different ways by different people? What are the possible interpretations of the requirement?
- Requirements realism
  - ✓ Is the requirement realistic given the technology which will be used to implement the system?
- Requirements testability
  - ✓ Is the requirement testable, that is, is it stated in such a way that test engineers can derive a test which can show if the system meets that requirement?

Requirements Interactions

- A very important objective of requirements analysis is to discover the interactions between requirements and to highlight requirements conflicts and overlaps
- A requirements interaction matrix shows how requirements interact with each other, which can be constructed using a spreadsheet
- Each requirement is compared with other requirements, and the matrix is filled as follows:
  - ✓ For requirements which conflict, fill in a 1
  - ✓ For requirements which overlap, fill in a 1000
  - ✓ For requirements which are independent, fill in a 0
- Consider an example
- An Interaction Matrix

Requirement	R1	R2	R3	R4	R5	R6
R1	0	0	1000	0	1	1
R2	0	0	0	0	0	0
R3	1000	0	0	1000	0	1000
R4	0	0	1000	0	1	1
R5	1	0	0	1	0	0
R6	1	0	1000	1	0	0

Comments on Interaction Matrices

- If you can’t decide whether requirements conflict, you should assume that a conflict exists. If an error is made it is usually fairly cheap to fix; it can be much more expensive to resolve undetected conflicts
- In the example, we are considering, we can see that R1 overlaps with R3 and conflicts with R5 and R6
- R2 is an independent requirement
- R3 overlaps with R1, R4, and R6
- The advantage of using numeric values for conflicts and overlaps is that you can sum each row and column to find the number of conflicts and the number of overlaps
- Requirements which have high values for one or both of these figures should be carefully examined
- A large number of conflicts or overlaps means that any changes to that requirement will probably have a major impact of the rest of the requirements
- Interaction matrices work only when there is relatively small number of requirements, as each requirement is compared with every other requirement
- The upper limit should be about 200 requirements
- These overlaps and conflicts have to be discussed and resolved during requirements negotiation, which we’ll discuss next

Requirements Negotiation

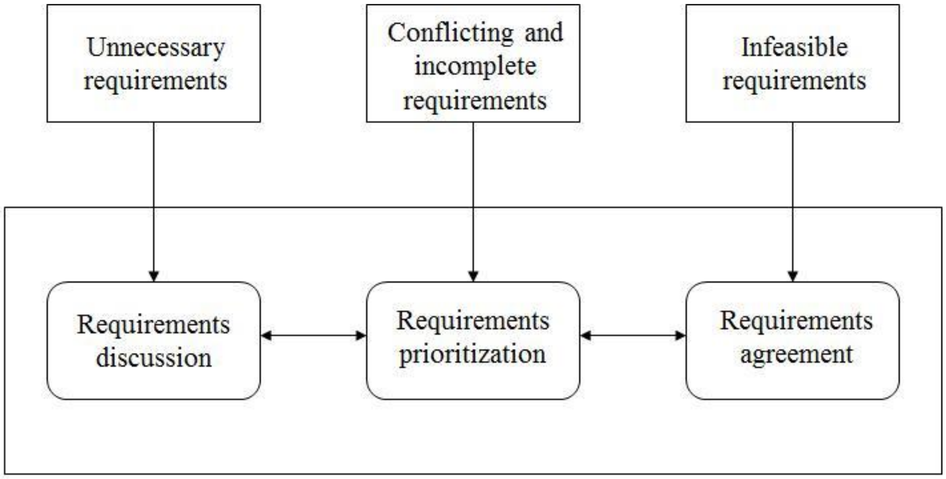
- Disagreements about requirements are inevitable when a system has many stakeholders. Conflicts are not “failures‟ but reflect different stakeholder needs and priorities

- Requirements negotiation is the process of discussing requirements conflicts and reaching a compromise that all stakeholders can agree to
- In planning a requirements engineering process, it is important to leave enough time for negotiation. Finding an acceptable compromise can be time-consuming
- The final requirements will always be a compromise which is governed by the needs of the organization in general, the specific requirements of different stakeholders, design and implementation constraints, and the budget and schedule for the system development

Requirements Negotiation Stages

- Requirements discussion
  - ✓ Requirements which have been highlighted as problematic are discussed and the stakeholders involved present their views about the requirements
- Requirements prioritization
  - ✓ Disputed requirements are prioritized to identify critical requirements and to help the decision making process
- Requirements agreement
  - ✓ Solutions to the requirements problems are identified and a compromised set of requirements are reached. Generally, this will involve making changes to some of the requirements

Requirements Negotiation Process



Requirements Negotiation

Comments on Requirements Negotiation

- In principle, requirements negotiation should be an objective process
- The judgments should be the requirements for the system should be based on technical and organizational needs
- Reality is, however, often different

- Negotiations are rarely conducted using only logical and technical arguments
- They are influenced by organizational and political considerations, and the personalities of the people involved
- A strong personality may force their priorities on other stakeholders
- Requirements may be accepted or rejected because they strengthen the political influence in the organization of some stakeholders
- End-users may be resistant to change and may block requirements, etc.
- The majority of time in requirements negotiation is usually spent resolving requirements conflicts. A requirement conflicts with other requirements if they ask for different things
- Example of access of data in a distributed system
- Even after years of experience, many companies do not allow enough time for resolution of conflicts in requirements
- Conflicts should not be viewed as ‘failures’, they are natural and inevitable – rather healthy

Resolution of Requirements Conflicts

- Meetings are the most effective way to negotiate requirements and resolve requirements conflicts
- All requirements which are in conflict should be discussed individually
- Negotiation meetings should be conducted in three stages

Stages of Negotiation Meetings

- Information stage
  - ✓ An information stage where the nature of the problems associated with a requirement is explained
- Discussion stage
  - ✓ A discussion stage where the stakeholders involved discuss how these problems might be resolved
    - o All stakeholders with an interest in the requirement should be given the opportunity to comment. Priorities may be assigned to requirements at this stage
- Resolution stage
  - ✓ A resolution stage where actions concerning the requirement are agreed
    - o These actions might be to delete the requirement, to suggest specific modifications to the requirement or to elicit further information about the requirement

Requirements Error/Defect

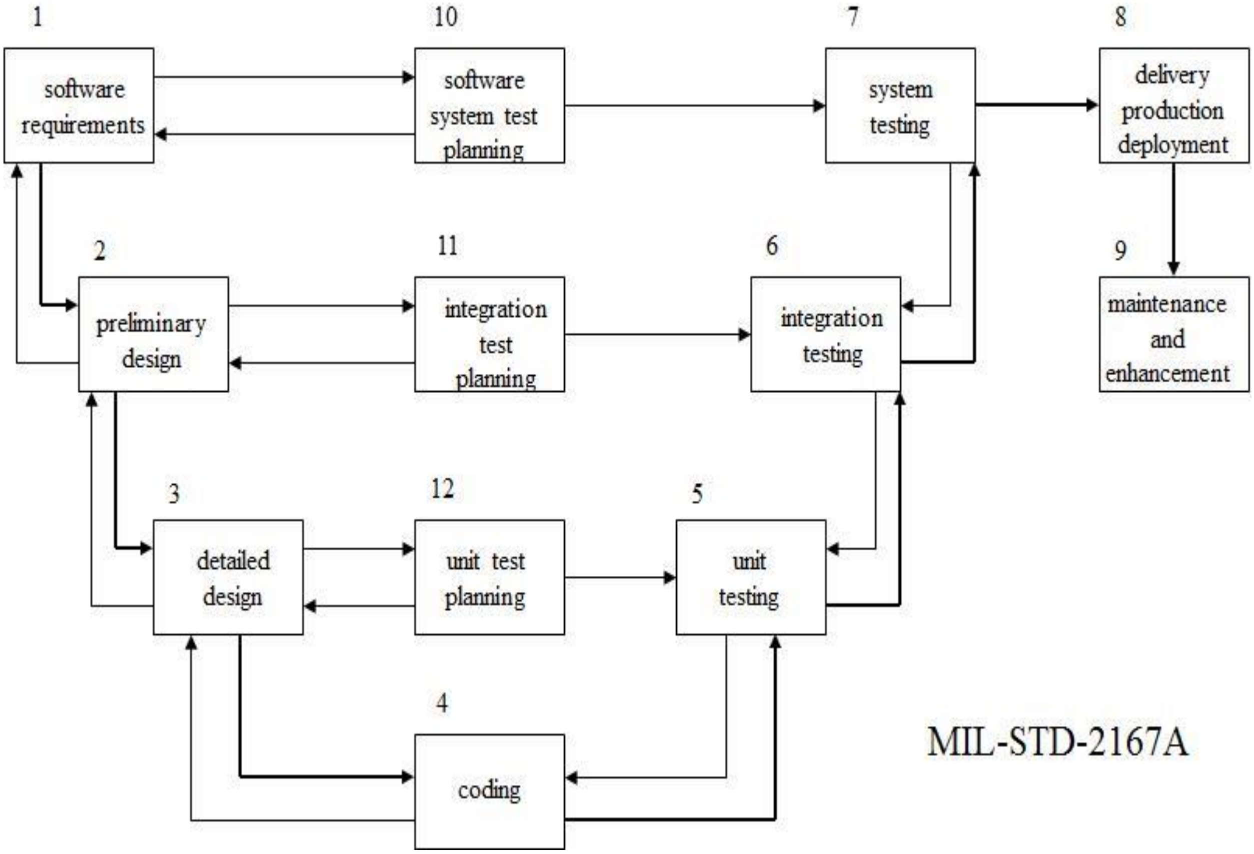
- A deficiency in the requirements quality that can hamper software development

Requirements Errors

- Errors and omissions find their way in different requirements documents
- If not removed, requirements errors usually flow downstream into design, code, and user manuals

- It is difficult to detect requirements errors once they flow downstream
- Requirements errors are most expensive to eliminate

Software Development Process



Types of Requirements Errors

- Errors of omission
  - ✓ Errors of omission are most common among requirements errors
  - ✓ Domain experts easily forget to convey domain knowledge to requirements engineers, because they consider that to be obvious and implicit
- Errors of clarity and ambiguity
  - ✓ 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
- Errors of commission
  - ✓ Errors of commission can also find their way into the requirements documents
- Errors of speed and capacity
  - ✓ Performance, that is errors of speed and capacity, are also found in requirements
  - ✓ Primarily, these occur due to conflicting understanding or competing needs of different stakeholders

Negative Impact of Requirements Errors

- **The resulting software may not satisfy user's real needs**
- Multiple interpretations of requirements may cause disagreements between customers and developers, wasting time and money, and perhaps resulting in lawsuits
- Negative impact on humans
  - ✓ Unsatisfied customers and developers
  - ✓ Lack of interest in automation of processes
  - ✓ Blame game

Addressing Requirements Errors

- Prevention
- Removal

Prevention vs. Removal

- For requirements errors, prevention is usually more effective than removal
- Joint application development (JAD), quality function deployment (QFD), and prototyping are more effective in defect prevention
- Requirements inspections and prototyping play an important role in defect removal

Defect Prevention

- Don't let defects/errors become part of the requirements document or requirements model in the first place
- How is it possible?
- Understanding application domain and business area is the first step in defect prevention
- Training in different requirements engineering activities (elicitation, analysis and negotiation, specification, and validation) is also very important for defect prevention
- Allocating enough time to conduct requirements engineering activities also is very important in this regard
- Willing and active participation of stakeholders in different activities of requirements engineering. That is why JAD is very useful in defect prevention as far as requirements errors are concerned
- An overall commitment to quality and emphasis on using documented processes is also a very important
- An overall commitment to process improvement

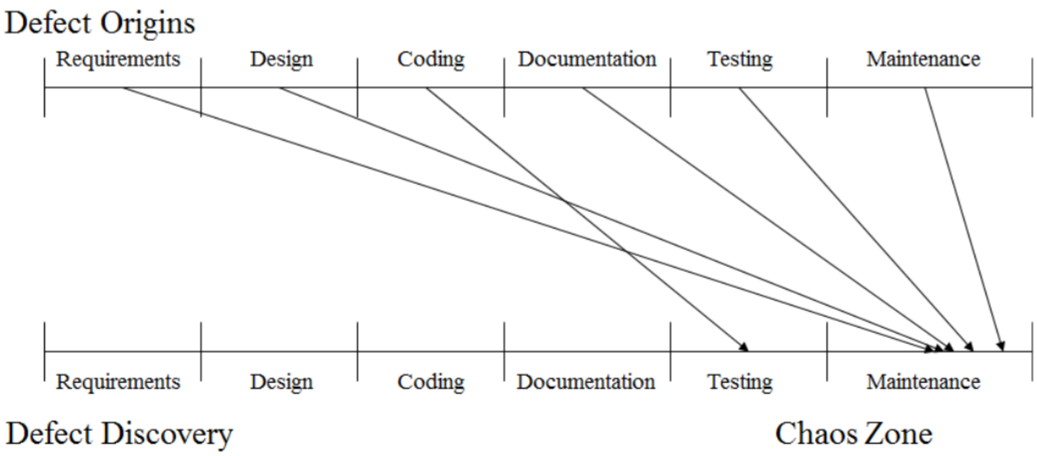
Inspections

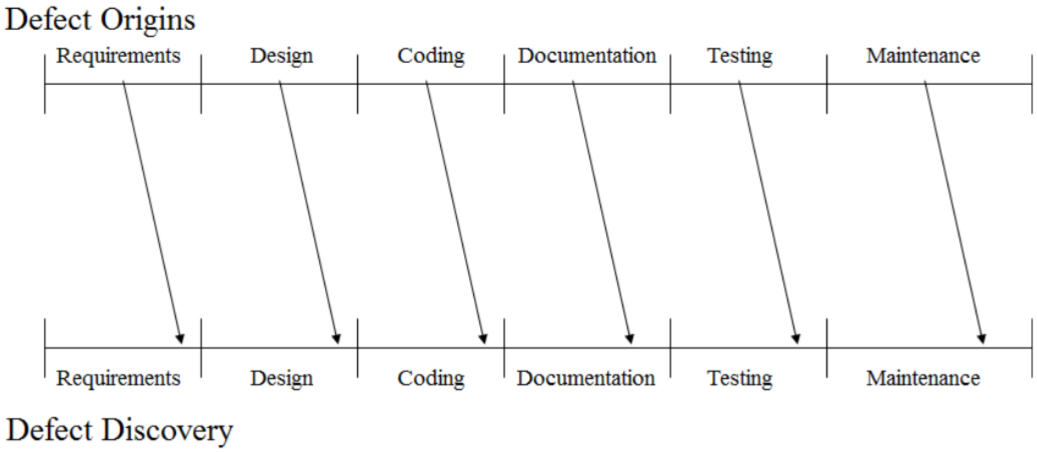
- Inspections, by all accounts, do a better job of error removal than any competing technology, and they do it at a lower cost
  - ✓ Robert Glass
- Inspections are conducted by a group of people working on the project, with the objective to remove defects or errors
- Every member of the inspection team has to read and evaluate requirements documents before coming to the meeting and a formal meeting is conducted to discuss

requirements errors

- Requirements errors detected during this inspections save lot of money and time as requirements errors do not flow into the design and development phases of software development process
- A complete description of inspections must address five dimensions:
  - ✓ Technical ✓ Assessment ✓ Managerial ✓ Tool support
  - ✓ Organizational

Defect Detection Without Inspections





Observations

- Requirements engineers are trained to write requirements documents, but have no training on reading/reviewing requirements documents
- Reviewers typically rely on ad hoc reading techniques, with no well-defined procedure, learning largely by doing

Techniques for Reading Requirements Documents

- Ad hoc review
  - ✓ A review with no formal, systematic procedure, based only individual experience
- Checklist review
  - ✓ A list of items is provided to reviewers, which makes this inspection process more focused
- Defect-based reading
  - ✓ Provides a set of systematic procedures that reviewers can follow, which are tailored to the formal software cost reduction notation
- Perspective-based reading
  - ✓ Researchers at Experimental Software Engineering Group at the University of Maryland, College Park, have created Perspective-Based Reading (PBR) to provide a set of software reading techniques for finding defects in English-language requirements documents

Different Perspectives

- PBR operates under the premise that different information in the requirements is more or less important for the different uses of the document
- Each user of the requirements document finds different aspects of the requirements important for accomplishing a particular task
- PBR provides a set of individual reviews, each from a particular requirements user's point of view, that collectively cover the document's relevant aspects
- This process is similar to constructing system use cases, which requires identifying who will use the system and in what way

Steps in PBR

- Selecting a set of perspectives for reviewing the requirements document
- Creating or tailoring procedures for each perspective usable for building a model of the relevant requirements information
- Augmenting each procedure with questions for finding defects while creating the model
- Applying procedures to review the document

Two Questions

- What information in these documents should they check?
- How do they identify defects in that information?

Benefits of Different Perspectives

- Systematic
  - ✓ Explicitly identifying the different uses for the requirements gives reviewers a definite procedure for verifying whether those uses are achievable
- Focused
  - ✓ PBR helps reviewers concentrate more effectively on certain types of defects, rather than having to look for all types

- Goal-oriented and customizable
    - ✓ Reviewers can tailor perspectives based on the current goals of the organization
- Transferable via training
    - ✓ PBR works from a definite procedure, and not the reviewer's own experience with recognizing defects, new reviewers can receive training in the procedures' steps

Identifying Defects

- A series of questions are used to identify different types of requirements defects
- Requirements that do not provide enough information to answer the questions usually do not provide enough information to support the user. Thus, reviewers can identify and fix defects beforehand

Requirements Defects that PBR Helps Detect

- Missing information
    - ✓ Any significant requirement related to functionality, performance, design constraints, attributes, or external interface not included
    - ✓ Undefined software responses to all realizable classes of input data in all realizable classes of situations
    - ✓ Sections of the requirements document
    - ✓ Figure labels and references, tables, and diagrams
    - ✓ Definitions of terms and units of measures
- Ambiguous information
    - ✓ Multiple interpretations caused by using multiple terms for the same characteristic or multiple meanings of a term in a particular context
- Inconsistent information
    - ✓ Two or more requirements that conflict with one another
- Incorrect fact
    - ✓ A requirement-asserted fact that cannot be true under the conditions specified for the system
- Extraneous information
    - ✓ Unnecessary or unused information (at best, it is irrelevant; at worst, it may confuse requirements users)
- Miscellaneous defects
    - ✓ Other errors, such as including a requirement in the wrong section

Benefits of PBR's Detailed Questions

- Allow controlled improvement
    - ✓ Reviewers can reword, add, or delete specific questions
- Allow training
    - ✓ Reviewers can train to better understand the parts of a representation or work product that correspond to particular questions

PBR General Questions

- Does the requirement make sense from what you know about the application or from what is specified in the general description?
- Do you have all the information necessary to identify the inputs to the requirement? Based on the general requirements and your domain knowledge, are these inputs correct for this requirement?
- Have any of the necessary inputs been omitted?
- Are any inputs specified that are not needed for this requirement?
- Is this requirement in the appropriate section of the document?

Results of PBR Experiments

- PBR provides a framework that represents an improved approach for conducting requirements reviews
- This approach will only be effective when an organization tailors the framework to its own needs and uses feedback from its reviewers to continually improve and refine the techniques
- PBR seems best suited for reviewers with a certain range of experience (not too little; not too much)
- Development teams that use PBR to inspect requirements documents tend to detect more defects than they do using other less- structured approaches
- Relatively novice reviewers can use PBR techniques to apply their expertise in other development tasks to defect detection
- Using PBR improves team meeting by helping team members build up expertise in different aspects of a requirements document
- It creates high-level representations of the software system, usable as a basis of work products in later stages of the development
- Each development organization can customize PBR’s set of perspectives, level of detail, and types of questions
- PBR facilitates controlled improvements, providing a definite procedure, alterable according to projects metrics and reviewers’ feedback

PBR General Questions

- Does the requirement makes sense from what you know about the application or from what is specified in the general description?
- Do you have all the information necessary to identify the inputs to the requirement? Based on the general requirements and your domain knowledge, are these inputs correct for this requirement?
- Have any of the necessary inputs been omitted?
- Are any inputs specified that are not needed for this requirement?
- Is this requirement in the appropriate section of the document?

Results of PBR Experiments

- PBR provides a framework that represents an improved approach for conducting requirements reviews
- This approach will only be effective when an organization tailors the framework to its own needs and uses feedback from its reviewers to continually improve and refine the techniques
- PBR seems best suited for reviewers with a certain range of experience (not too little; not too much)
- Development teams that use PBR to inspect requirements documents tend to detect more defects than they do using other less- structured approaches
- PBR seems best suited for reviewers with a certain range of experience (not too little; not too much)
- Development teams that use PBR to inspect requirements documents tend to detect more defects than they do using other less- structured approaches
- It creates high-level representations of the software system, usable as a basis of work products in later stages of the development
- Each development organization can customize PBR’s set of perspectives, level of detail and type of questions.
- PBR facilitates controlled improvements, providing a definite procedure, alterable according to projects metrics and reviewers’ feedback.