Requirement Analysis and Specification

Ruchita Shah

Software Requirements

- Functional & Non-functional Requirements
- User Requirements
- System Requirements
- The Software Requirements Documents

Software Requirements

https://dilbert.com/strip/2006-01-29

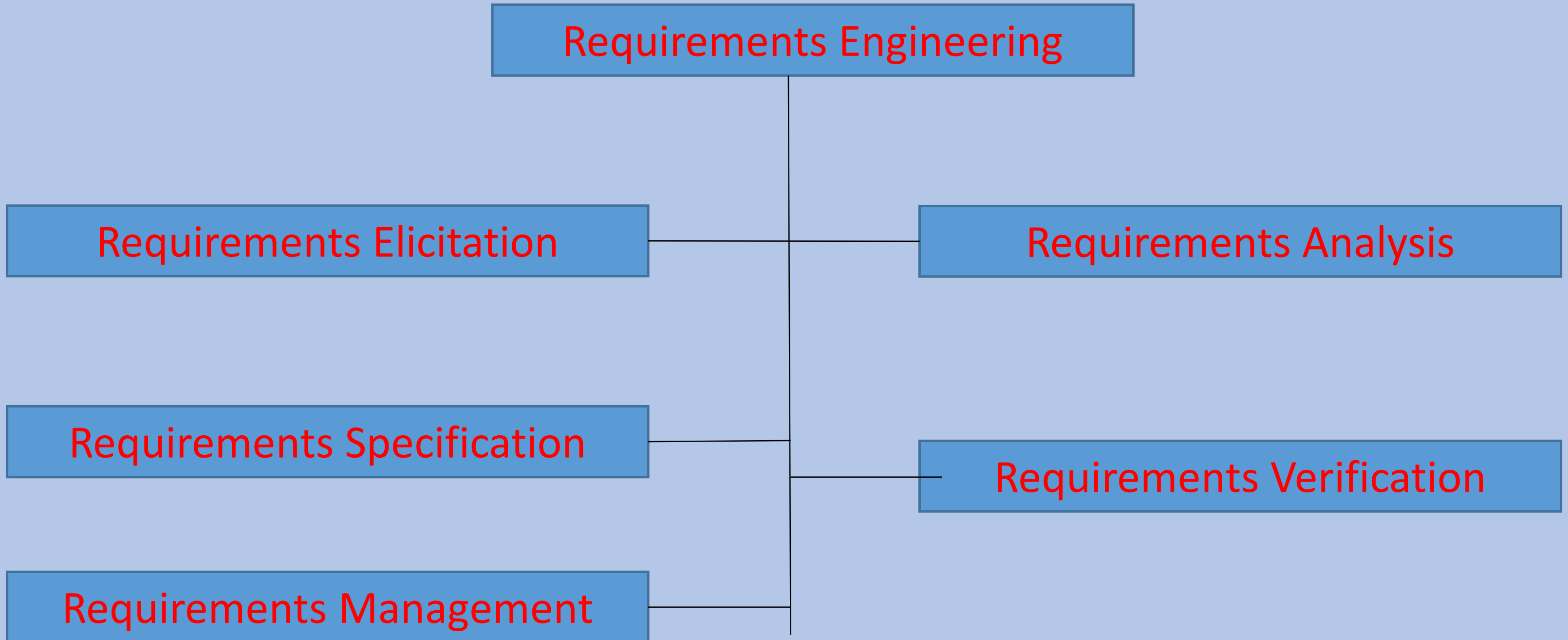
Requirements Engineering

- The process of establishing services that the customer requires from a system and the constraints under which it operates and is developed
- The requirements themselves are the descriptions of the system services and constraints that are generated during the requirements engineering process

What is a Requirement?

- It may range from a high level abstract statement of a service or of a system constraint to a detailed mathematical functional specification
- This is inevitable as requirements may serve a dual function
  - May be the bases for a bid for a contract – therefore must be open to interpretation
  - May be the basis for contract itself – Therefore must be defined in detail
  - Both these statements may be called requirements

Requirements Engineering Components



Requirement Elicitation

- The task of communication with customers and users
- To determine their requirements
- Also called as requirements gathering

Requirement Analysis

- To find if stated requirements are
  - Unclear
  - Incomplete
  - Ambiguous
  - Contradictory
- To resolve the issues

Requirement Specifications

- Documenting the system requirements in a formal or semi-formal manner to ensure
  - Clarity
  - Consistency and
  - Completeness
- Might be documented in various forms such as
  - Natural language documents
  - Use-cases
  - User stories
  - Process specification

Eliciting Requirements

- Helps customers to define what is required
  - What is to be accomplished
  - How the system will fit into the needs of the business
  - How the system will be used in day-to-day basis
- Analyst can employ several techniques to elicit the requirements from the customers
  - interviews
  - Focus-groups (requirements workshops) and creating requirements list
  - Prototyping and Use-cases
  - Combination of above all

Requirement Analysis

- Process of studying and analyzing customers’ and users’ needs to arrive at a definition to software requirements
- Requirements must be actionable, measureable, testable, related to identified business needs or opportunities, and defined to a level of details that are sufficient for system design
- Requirements can be functional and non-functional

Requirement Analysis

- Defining stake-holder and user profiles

Description – brief description of stake-holder and user type

Type : qualify users' expertise, technical background and degree od sophistication

Responsibilities – list users' key responsibility with regard to system being developed, why a stakeholder?

  - Stack-holder : a person, group or company that is directly or indirectly involved in the project and who may affect or get affected by the outcome of the project

Requirement Analysis

- Defining stake-holder and user profiles

Success criteria – how does user/stake holder define success? How rewarded?

Involvement : In what way involved in project? What role? Ex. Requirements reviewer, system tester etc

Deliverables – Are any deliverables being produced by user? For whom? Any required by stake holders?

Comments/issues - problems that interfere with success etc., This includes trends that make users’ job easier or harder

Requirement Analysis

- Defining user work environment

Number of people involved in this currently? Changing?

How long is a task cycle currently? Changing?

Any unique environmental constraints – mobile, out-doors, in-flight etc.?

Which system platforms are in use currently? Future ?

What other applications are in use? need for integration?

Requirement Analysis

- Product overview

Put the product in perspective to other related products and the users’ environment

Independent?

Component of a larger system?

How do the subsystems interact with this component?

Known interfaces with them and this component?

Block diagram?

Requirement Analysis

- Other Product requirements

Hardware platform requirements

System Requirements - supported host OS, Peripherals, companion software

Environmental requirements – temperature, shock, humidity, radiation, usage conditions, resource availability, maintenance issues, type of error recovery

Applicable standards – legal, regulatory, communications

Types of Requirements

- User requirements : statements in natural language and diagrams of services the system provides and its operational constraints, written for customers
- System requirements : A structures document setting out detailed description of the system’s functions, services and operational constraints. Defines what should be implemented so may be part of a contract between client and developer

Requirements Vs Design

Requirements	**Design**
Describe what will be delivered	Describe how it will be done
Primary goal : **Understanding**	Primary goal : **Optimization**
There is more than one solution	There is only one solution
Customer is interested	Customers mostly not interested except for external

Functional Requirements

- Describes system behavior
- Statements of services to be provided by the system, how system will react to particular input, system’s behavior in particular situations etc
- Priority – order of the feature in importance
- Criticality – how essential is each requirement to the overall system
- Risks – when might a requirement not be satisfied? What steps to reduce this risk?

Non-functional Requirements

- Describes other desirable attribute of the system
- Constraints on the services or other functions offered by the system such as timing constraints, constraints on development process, standards etc
- Product cost – how to measure cost
- Performance – efficiency, response time, startup time
- Portability – target platforms, binary or byte-code compatibility
- Availability – how much down time is acceptable

Non-functional Requirements

- Security – prevention against intrusion, attacks
- Safety – damage to people, environment, monetary
- Maintainability – reusability, extensibility etc
- Non-functional requirements are more critical than functional requirements. If these are not delivered, system stands to useless

FURPS + model

- Functional – features, capability security
- Usability – human factors, help, documentation
- Reliability – frequency of failure, recoverability, predictability
- Performance – response time, throughput, accuracy, availability, recourse usage
- Supportability – adaptability, maintainability, internationalization, configurability
- Developed by Grady in 1992

FURPS + model

Not to forget

- Implementation - resource limitations, language & tools, hardware
- Interface – constraints with external systems
- Operations – system management in its operational settings
- Packaging
- Legal - licensing

Example

- A LIBRARY SYSTEM that provides a single interface to a number of databases of articles in different libraries
- Users can search, down load and print articles for their personal study

Example – functional requirements

- The user must be able to all of initial set of databases or select a subset from it
- System shall provide appropriate view so user can read document from document store
- Every order shall be allocated a unique identifier which user shall be able to copy to account’s permanent storage area

Requirements imprecision

- Problems arise when requirements are not precisely stated
- Ambiguous requirements may be interpreted differently by developers and customer
- Consider the term “appropriate view”
  - User interpretation- special purpose view for each different document types
  - Developer interpretation – provide a text view to display content of document

Requirements completeness and consistency

- Requirements should be both complete and consistence
- Complete – include description of all facilities required
- Consistent – no conflicts or contradiction in description of system facilities

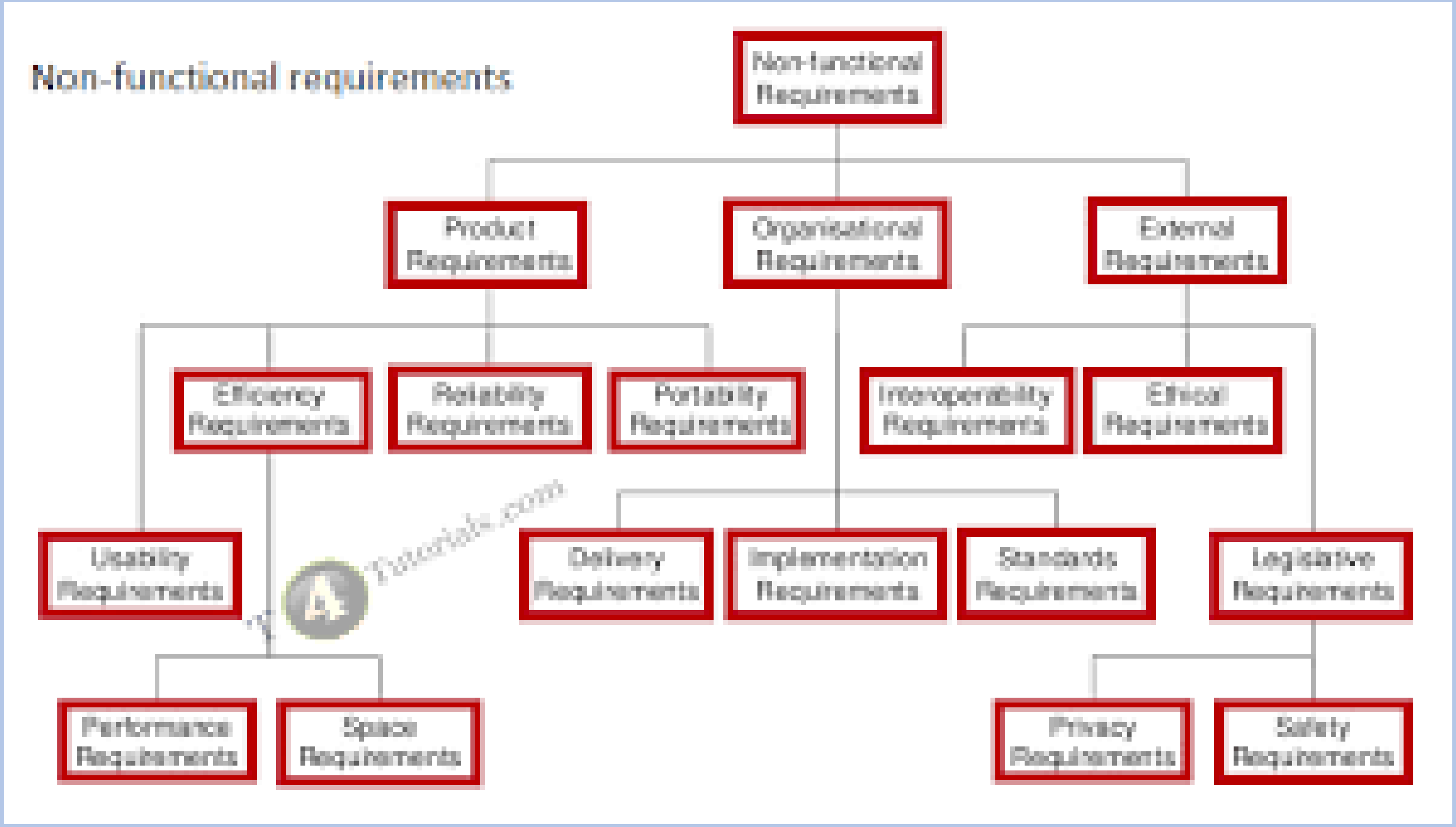
Non-functional Requirements

- Product Requirements – which specify that delivered product must behave in a particular way for ex. – execution speed, reliability etc
- Organizational requirements – which are a consequence of organizational policies and procedures for ex. –process standards used
- External requirements – which arise from the factors which are external to the system and its development process for ex. -interoperability requirements, legislative requirements etc.

Non-functional Requirements

- Product Requirements – which specify that delivered product must behave in a particular way for ex. – execution speed, reliability etc
- Organizational requirements – which are a consequence of organizational policies and procedures for ex. –process standards used
- External requirements – which arise from the factors which are external to the system and its development process for ex. -interoperability requirements, legislative requirements etc.

Non-functional Requirements



Non-functional Requirements examples

- Product Requirements
  - User i/f for libsys shall be implemented as simple HTML without frames or JAVA applets
- Organizational requirements
  - The system development process and deliverable document shall conform to the process and deliverables defined
- External requirements
  - The system shall not disclose any personal information about customers to the operators of the system, only their name & reference number

Non-functional Requirements examples

- System goal
  - The system should be easy to use by experienced users and should be organized in such a way that user error should be minimized
- A verifiable non-functional requirement
  - Experienced users shall be able to use all the system function after a total 2 hours of training. After the training average numbers of errors made by experience user shall be more than 2

Requirements Analysis

- Bridge the gap between requirements & software design
- Provides models
    - System information
    - Function
    - Behaviour
- Model can be translated into Architect, data and component design
- Expect to do a little bit design during analysis and a little analysis during design

Requirements Analysis Objectives

- Identify customer’s needs
- Evaluate system for feasibility
- Perform economic & technical analysis
- Allocate functions to system elements
- Establish schedule & constraints
- Create system definitions

Requirements Engineering Tasks

- Inception
- Elicitation
- Elaboration
- Negotiation
- Specification
- Validation
- Requirements Management

Requirements Engineering- Inception

- How does software project get started?
- Catalyst – single event or need evolves over time
- A casual conversation may lead to SE
- Sometimes when a business need is identified or new market/service is discovered
- Ask questions to establish a basic understanding of problem, people who want solution, nature of desired solution

Requirements Engineering- Elicitation

- Its not as simple as ask the customer & users
- Problem of scope : boundary of the system, ill defined or unnecessary technical detail lead to confusion
- Problem of understanding : not sure about what is needed, don’t have full understanding of problem domain, trouble in communication, ambiguous /unstable requirements
- Problem of volatility : requirements change over time
- Requirements gathering activity in organized manner

Requirements Engineering- Elaboration

- Information gathered are expanded & refined
- Focus on developing refined technical model of s/w functions, features & constraints
- User scenarios are created
- User scenarios are converted into analysis class
- At the end, an analysis model defining informational, functional & behavioral domain

Requirements Engineering- Negotiation

- Customers & user must rank requirements & prioritize
- Risk associated with each requirements identified & analyzed
- Iteratively requirements are eliminated. Combined, modified

Requirements Engineering- Specification

- Can be
  - A written document
  - Set of graphical models
  - Formal mathematical model
  - Usage scenarios
  - Combination of any of these
- For large systems : a written document using natural language combining graphical model
- For small system : usage scenario may suffice
- Final product of requirements engineering

Requirements Engineering- Validation

- Examine specifications to ensure all s/w requirements stated clearly
- Detect inconsistency, omissions, errors & correct
- Formal technical review are conducted
- Include S/w engineers, customers, users & other stakeholders

Requirements Engineering- Management

- A set of activities that help project team to identify, control & track requirements
- Also changes in requirements are handled
- Begins with assigning unique id to each requirement
- Traceability tables are created

Initiating the Requirements Engineering Process

- Conducting meaningful conversations with customers
- Identify Stakeholders: create a list of people who contribute input as requirement
- Recognize multiple view points : requirements are explored with from many different viewpoints
- Working towards collaboration : identify areas of commonality & areas of conflict

Initiating the Requirements Engineering Process

- Asking first questions
    - Context free questions such as
    - Who is behind the request for this work?
    - Who will use solution?
    - What will be the economic benefit?
    - Is there another solution?
    - What is a characteristic of a “good” output”?
    - Which problems will be addressed by the solution?
    - Are my questions relevant to the problem

Software Requirements Elicitation

- Meeting with customers is the most common technique
- Use context free question to find out
  - Customers’ goals & benefits
  - Identify stakeholders
  - Gain understanding of problem
  - Determine customer reactions to proposed solutions
  - Assess effectiveness of meeting
- Interview cross section of users when many users are anticipated

Management Concerns

- How much effort for analysis?
- Who will do analysis?
- Why is it so difficult?
- Who will pay for it

Feasibility study

- Economic Feasibility
  - Cost/benefit analysis
- Technical feasibility
  - h/w, s/w, people etc.
- Legal feasibility
- Alternatives
  - Here is always more than one way to do it

System Specifications

- Introduction
- Functional Data description
- Sub-system description
- System modelling & simulation results
- Products
- Appendix

System Requirements

- Requirements
  - Features of system or system function used to fulfill system purpose
- Focus on customers’ needs and problem; not on solution
  - Requirements specification documents for customer
  - Requirements specification documents for developers

Types of Requirements

- Functional Requirements
  - Input/output
  - Processing
  - Error handling
- Non-functional requirements
  - Physical environment (equipment location, multiple sites etc)
  - Interfaces
  - User & human factors (who are the users, their skill level etc)

Types of Requirements

- Non-functional requirements
  - Performance (System function speed etc)
  - Documents
  - Data
  - Resources
  - Security (backup, firewall etc)
  - Quality assurance

Requirements Validation

- Correct?
- Consistent?
- Complete?
  - Externally – all desired properties are present
  - Internally – no undefined references
- Each requirement describes something actually needed by the customer
- Requirements are verifiable (testable)
- Requirements are traceable

Requirements Definition Document

- General purpose of the document
- System background and objectives
- Description of approach
- Detailed characteristics of proposed system (data & functionality)
- Description of operating environment

Facilitated Application Specification technique

- FAST
- Meeting between customers & developers at a neutral site
- Goals
  - Identify the problem
  - Propose elements of solutions
  - Negotiate different approaches
  - Specify preliminary set of requirements

Facilitated Application Specification technique

- FAST
- Rules of participation & preparation established ahead of time
- Agenda suggested
  - Brainstorming encouraged
- Facilitator appointed
- Definition mechanisms
  - Spreadsheets, wallboards, stickers etc.

Quality Function Deployment (QFD)

- Customers’ needs imply technical requirements
  - Normal Requirements : minimal functional 7 performance
  - Expected requirements : important implicit requirements
  - Exciting requirements : may become normal requirement in future
- Function deployment : determines value of required function
- Information deployment : focuses on data objects & events produced or consumed by the system
- Task deployment : product behavior & implied operating environment

Quality Function Deployment (QFD)

- User Scenario
  - An overall vision of system function & features materialized
  - Software team must understand different classes of end-users
  - Developers & users create a set of scenarios that identify type of usage of the system
  - Also known as use cases

Quality Function Deployment (QFD)

- Value analysis makes use of
  - Customer interviews
  - Observations
  - Surveys
  - Historical data
- To create
  - Customer voice table
  - Extract expected requirements
  - Derive exciting requirements

User profile example

- Full control (Administrator)
- Read/write/modify All (managers)
- Read/write/modify own (user)
- Read only (general public)

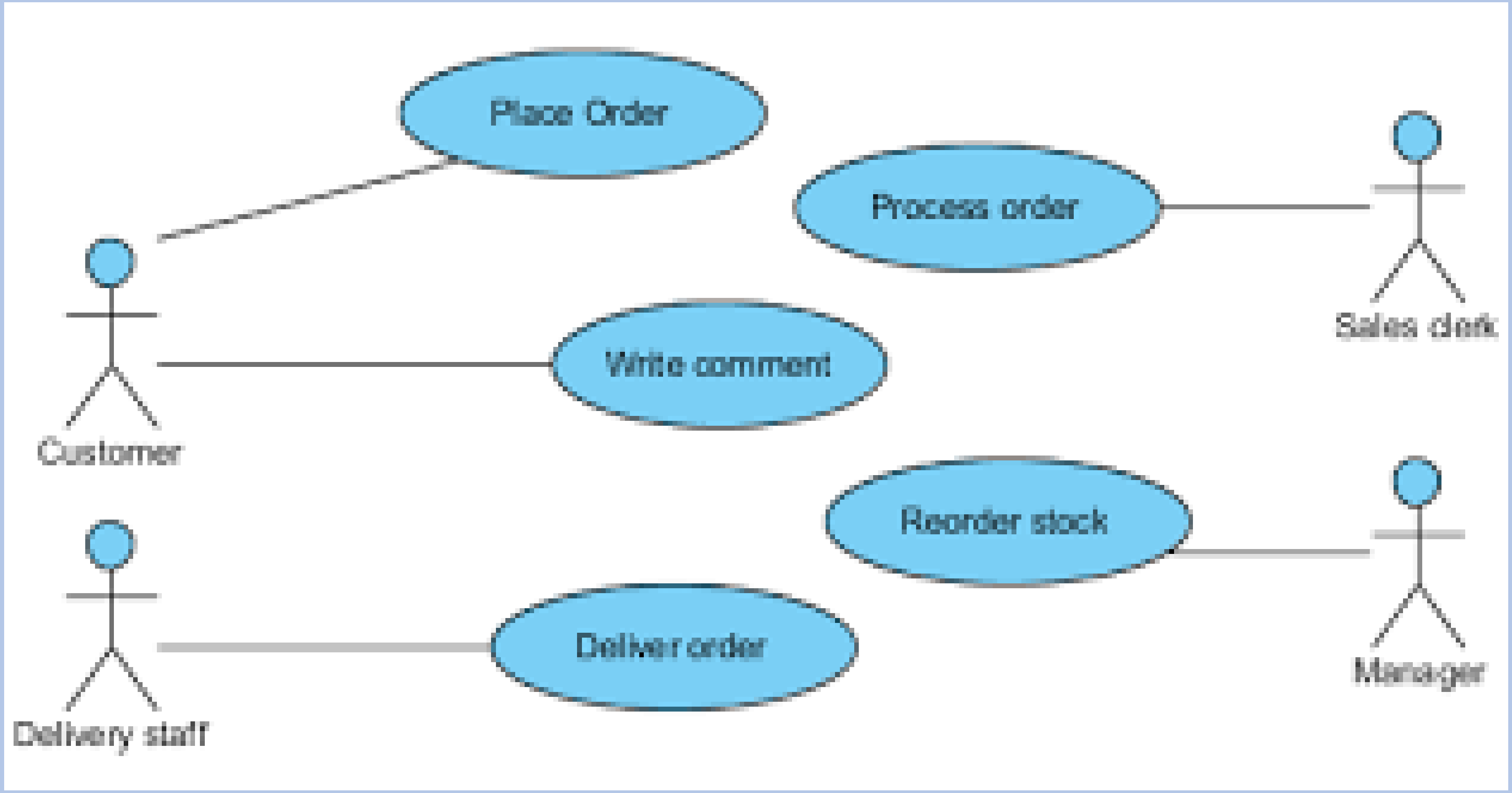
User profile example

- Read only uses
    - Will only read the database and can not insert, delete or update any record
- Read/write/modify own users
    - Will be able to insert new records, delete them & modify information inserted in past

User profile example

- Read/write/modify All users
  - Will be able to do record maintenance tasks, can modify, delete any record created by any user
- Full control users
  - System administrators, capable to change system settings and maintaining user profiles

User profile example



Analysis principles

- Information domain of problem must be presented & understood
- Models depicting system information, functions & behavior should be developed
- Models & problems must be partitioned in a manner that uncovers details in layers
- Analysis proceeds from information to implementation details
- Must be traceable

Information Domain

- Encompass all data objects that contains numbers, texts, images, audio, video etc
- Information content or data model : shows the relation between data & control objects that make up the system
- Information flow : represents manner in which data & control object change as each moves through system
- Information structure : represent internal organization of various data & control items

Modelling

- Data Model
  - Shows relationship among system objects
- Functional Model
  - Description of functions that enables the transformation of system objects
- Behavioral Model
  - Manner in which software responds to events from outside world

Partitioning

- Process that results in elaboration of data function & behavior
- Horizontal partitioning
  - Breadth first decomposing of the system functions, behavior or information; one at a time
- Vertical partitioning
  - Depth first elaboration of the system function, behavior or information; one subsystem at a time

Requirements views

- Essential view
  - Presents the functions to be accomplished & the information to be processed while ignoring implementation
- Implementation view
  - Presents the real world realization of processing functions & information structures
- Avoid the temptation of directly moving to implementation view & assuming that the essence of the problem is obvious

Requirements views

- Essential view
  - Presents the functions to be accomplished & the information to be processed while ignoring implementation
- Implementation view
  - Presents the real world realization of processing functions & information structures
- Avoid the temptation of directly moving to implementation view & assuming that the essence of the problem is obvious

Use cases

- Scenarios that describe how processes are performed in specific situations
- Written narratives that describe the role of an actor (user or device) as it interacts with the system
- Use cases are designed from actors’ point of view
- Not all actors can be identified during first iteration of requirements elicitation
- It is important to identify important actors before developing use-cases

Developing Use cases

- Define set of actors that are involved with the system
  - different people that use system or product within context of function
  - they assume some role
  - they communicate with system
  - actor & end-user may not be same

Developing Use cases

- Ex. Machine operator
  - interacts with control computer for a manufacturing cell
  - s/w for control computer requires 4 roles: programming mode, test mode, monitoring mode and troubleshoot mode
  - Machine operator plays roles of programmer, tester, monitor & troubleshot
  - There might be different people for all 4 roles
- Primary actors – interact directly with the system
- Secondary actors – supports system so primary actor can use it

Developing Use cases

- Identify primary & secondary actors
- Actor’s goals
- Preconditions before story begins
- Main tasks or functions being performed by the actor
- Exceptions
- Variations in actor’s interaction
- Which system information actor will acquire, produce or change
- What information does actor desire from system
- Does actor wish to be informed about unexpected changes

Developing Use cases

- Ex. SafeHome software