UNIT-II
REQUIREMENTS ANALYSIS AND SPECIFICATION

The requirements for a system are the descriptions of what the system should do the services that it provides and the constraints on its operation.

- **User requirements** are statements, in a natural language plus diagrams, of what services the system is expected to provide to system users and the constraints under which it must operate.
- **System requirements** are more detailed descriptions of the software system's functions, services, and operational constraints. The system requirements document (sometimes called a functional specification) should define exactly what is to be implemented. It may be part of the contract between the system buyer and the software developers.

FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS

Software system requirements are often classified as functional requirements or nonfunctional requirements:

Functional requirements These are statements of services the system should provide, how the system should react to particular inputs, and how the system should behave in particular situations. In some cases, the functional requirements may also explicitly state what the system should not do.

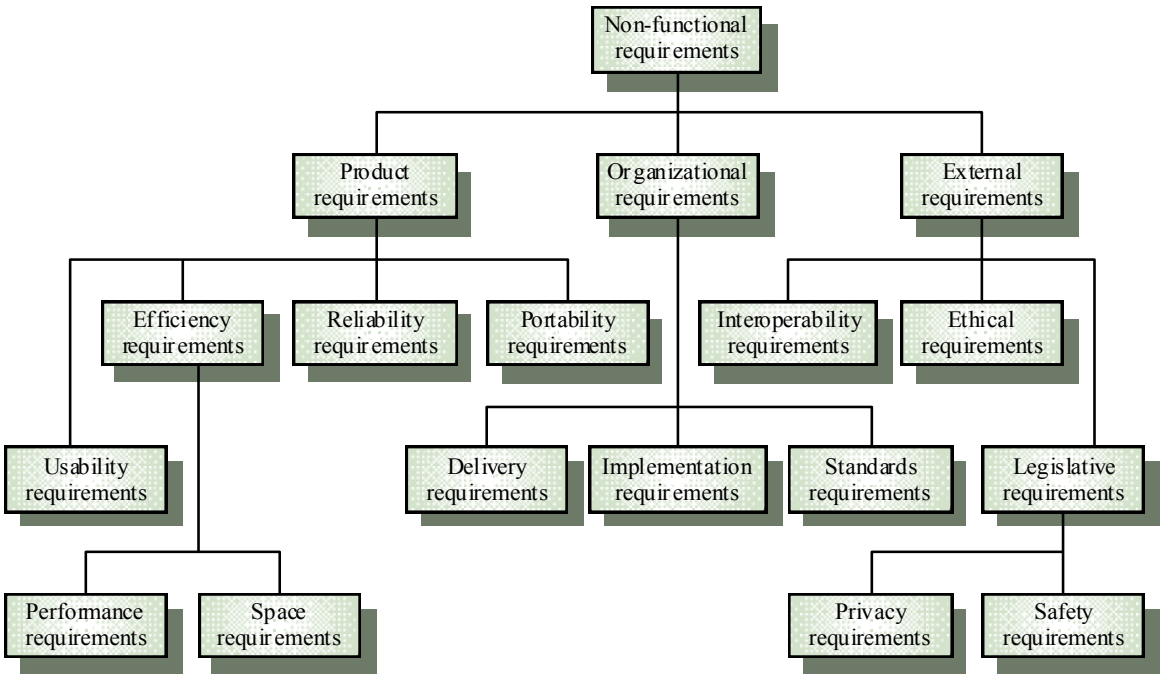
- Functional system requirements vary from general requirements covering what the system should do to very specific requirements reflecting local ways of working or an organization's existing systems.
- Imprecision in the requirements specification is the cause of many software engineering problems.
- The functional requirements specification of a system should be both complete and consistent. Completeness means that all services required by the user should be defined. Consistency means that requirements should not have contradictory definitions.

Non-functional requirements These are constraints on the services or functions offered by the system. They include timing constraints, constraints on the development process, and constraints imposed by standards. Non-functional requirements often apply to the system as a whole, rather than individual system features or services.

- These are requirements that are not directly concerned with the specific services delivered by the system to its users. They may relate to emergent system properties such as reliability, response time, and store occupancy.
- Although it is often possible to identify which system components implement specific functional requirements, it is often more difficult to relate components to non-functional requirements. The implementation of these requirements may be diffused throughout the system.

- There are two reasons for this:

  - ✓ Non-functional requirements may affect the overall architecture of a system rather than the individual components.
  - ✓ A single non-functional requirement, such as a security requirement, may generate a number of related functional requirements that define new system services that are required.



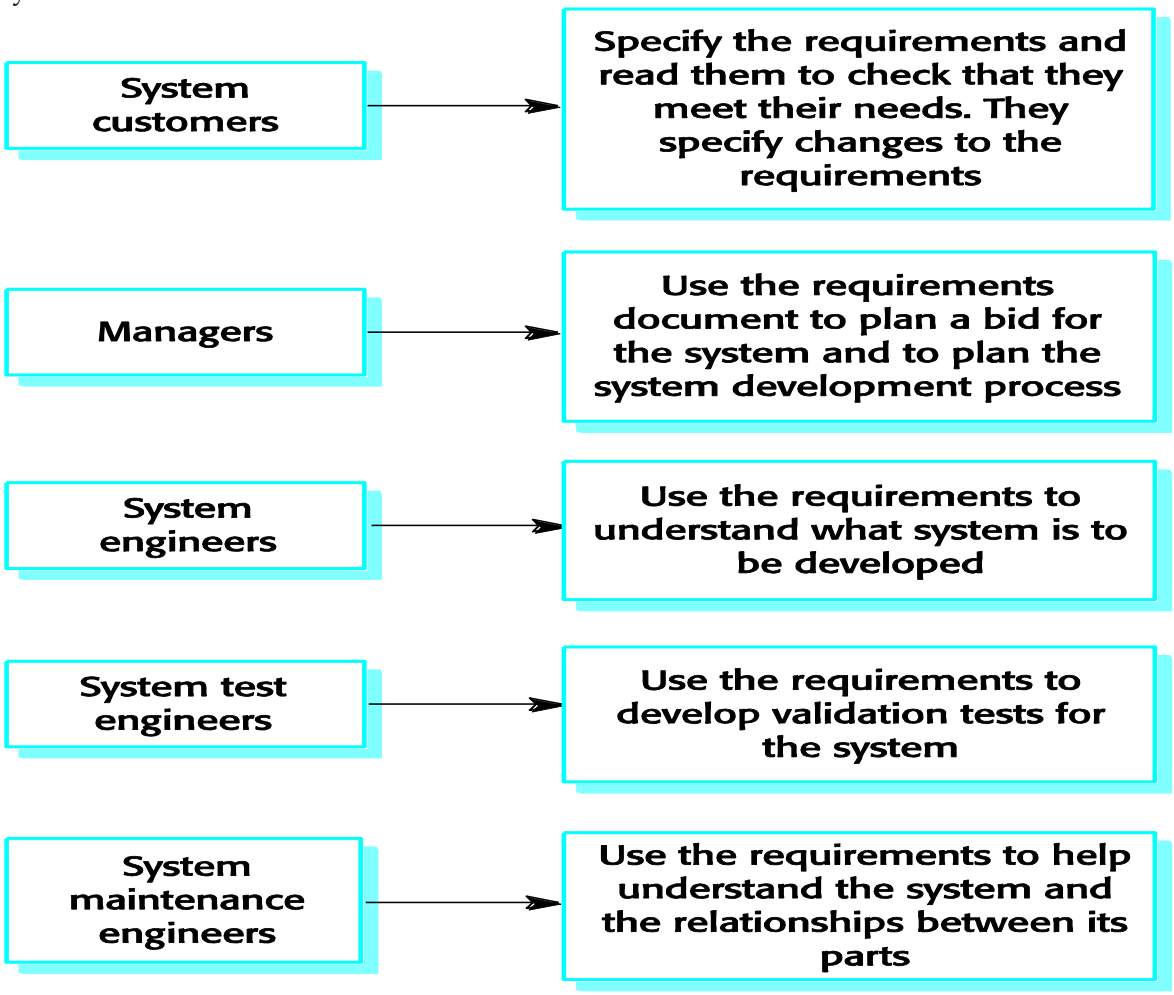
- **Product requirements** These requirements specify or constrain the behavior of the software.
- **Organizational requirements** These requirements are broad system requirements derived from policies and procedures in the customer's and developer's organization.
- **External requirements** This broad heading covers all requirements that are derived from factors external to the system and its development process.

Property	Measure
Speed	Processed trasactions/second user/Event response time Screen refresh time
Size	K Bytes Number of RAM chips
Ease of Use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence

	Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Probability	Percentage of target dependent statement Number of target systems

SOFTWARE REQUIREMENT SPECIFICATION(SRS)

- The requirements document is the official statement of what is required of the system developers.
- Should include both a definition of user requirements and a specification of the system requirements.
- It is NOT a design document. As far as possible, it should set out WHAT the system should do rather than HOW it should do it



- Defines a generic structure for a requirements document that must be instantiated for each specific system.
    - Introduction.
    - General description.
    - Specific requirements.
    - Appendices.
    - Index.
- Requirement Document Structure
    - Preface
    - Introduction
    - Glossary
    - User requirements definition
    - System architecture
    - System requirements specification
    - System models
    - System evolution
    - Appendices
    - Index

- The information that is included in a requirements document depends on the type of software being developed and the approach to development that is to be used.

REQUIREMENTS SPECIFICATION

- Requirements specification is the process of writing down the user and system requirements in a requirements document. Ideally, the user and system requirements should be clear, unambiguous, easy to understand, complete, and consistent.

- The user requirements for a system should describe the functional and nonfunctional requirements so that they are understandable by system users who don't have detailed technical knowledge.

- System requirements are expanded versions of the user requirements that are used by software engineers as the starting point for the system design. They add detail and explain how the user requirements should be provided by the system.

- The system requirements should simply describe the external behavior of the system and its operational constraints. They should not be concerned with how the system should be designed or implemented.

- At the level of detail required to completely specify a complex software system, it is practically impossible to exclude all design information. There are several reasons for this:

- ✓ The system requirements are organized according to the different sub-systems that make up the system.

- ✓ Systems must interoperate with existing systems, which constrain the design and impose requirements on the new system.

- ✓ The use of a specific architecture to satisfy non-functional requirements may be necessary.

Natural language specification

- To minimize misunderstandings when writing natural language requirements, the following guidelines are provided,
  - ✓ Invent a standard format and ensure that all requirement definitions adhere to that format.
  - ✓ Use language consistently to distinguish between mandatory and desirable requirements. Mandatory requirements are requirements that the system must support and are usually written using 'shall'. Desirable requirements are not essential and are written using 'should'.

  - ✓ Use text highlighting (bold, italic, or color) to pick out key parts of the requirement.
  - ✓ Do not assume that readers understand technical software engineering language. It is easy for words like 'architecture' and 'module' to be misunderstood. Therefore, use of jargon, abbreviations, and acronyms are avoided.
  - ✓ The rationale should explain why the requirement has been included. It is particularly useful when requirements are changed as it may help decide what changes would be undesirable.

Structured specifications

When a standard form is used for specifying functional requirements, the following information should be included:

- ✓ A description of the function or entity being specified.
- ✓ A description of its inputs and where these come from.
- ✓ A description of its outputs and where these go to.
- ✓ Information about the information that is needed for the computation or other entities in the system that are used (the 'requires' part).
- ✓ A description of the action to be taken.

- ✓ If a functional approach is used, a pre-condition setting out what must be true before the function is called, and a post-condition specifying what is true after the function is called.
- ✓ A description of the side effects (if any) of the operation.

REQUIREMENT ENGINEERING:

Requirement Engineering provides the appropriate mechanism for understanding what the customer wants, analyzing need, assessing feasibility, negotiating a reasonable solution, specifying the solution unambiguously, validating the specification and managing the requirements as they are transformed into an operational system.

Guidelines principles for requirement engineering:

- Understand the problem before beginning the analysis model.
- Develop prototypes that enable a user to understand how human/machine interaction will occur.
- Record the origin of and the reason for each and every requirements.
- Use multiple views of requirements.
- Rank the requirements and eliminate the ambiguity.

Requirement Engineering Process:

- **Inception**

  During inception, the requirements engineer asks a set of questions to establish…
  - ✓ A basic understanding of the problem
  - ✓ The people who want a solution
  - ✓ The nature of the solution that is desired
  - ✓ The effectiveness of preliminary communication and collaboration between the customer and the developer
- **Elicitation**

  Elicitation may be accomplished through two activities
  - ✓ Collaborative requirements gathering
  - ✓ Quality function deployment
- **Elaboration**
  - ✓ During elaboration, the software engineer takes the information obtained during inception and elicitation and begins to expand and refine it
  - ✓ Elaboration focuses on developing a refined technical model of software functions, features, and constraints
- **Negotiation**
  - ✓ During negotiation, the software engineer reconciles the conflicts between what the customer wants and what can be achieved given limited business resources
  - ✓ Requirements are ranked (i.e., prioritized) by the customers, users, and other stakeholders
  - ✓ Risks associated with each requirement are identified and analyzed
  - ✓

- **Specification**
  - ✓ A specification is the final work product produced by the requirements engineer
  - ✓ It is normally in the form of a software requirements specification
  - ✓ It serves as the foundation for subsequent software engineering activities
  - ✓ It describes the function and performance of a computer-based system and the constraints that will govern its development
- **Validation**
  - ✓ During validation, the work products produced as a result of requirements engineering are assessed for quality
  - ✓ The specification is examined to ensure that
    - all software requirements have been stated unambiguously
    - inconsistencies, omissions, and errors have been detected and corrected
    - the work products conform to the standards established for the process, the project, and the product
  - ✓ The formal technical review serves as the primary requirements validation mechanism
    - Members include software engineers, customers, users, and other stakeholders
- **Requirements Management**
  - ✓ During requirements management, the project team performs a set of activities to identify, control, and track requirements and changes to the requirements at any time as the project proceeds
  - ✓ Each requirement is assigned a unique identifier
  - ✓ The requirements are then placed into one or more traceability tables

FEASIBILITY STUDY:

- The aims of a feasibility study are to find out whether the system is worth implementing and if it can be implemented, given the existing budget and schedule.

- The purpose of feasibility study is not to solve the problem, but to determine whether the problem is worth solving. This helps to decide whether to proceed with the project or not.

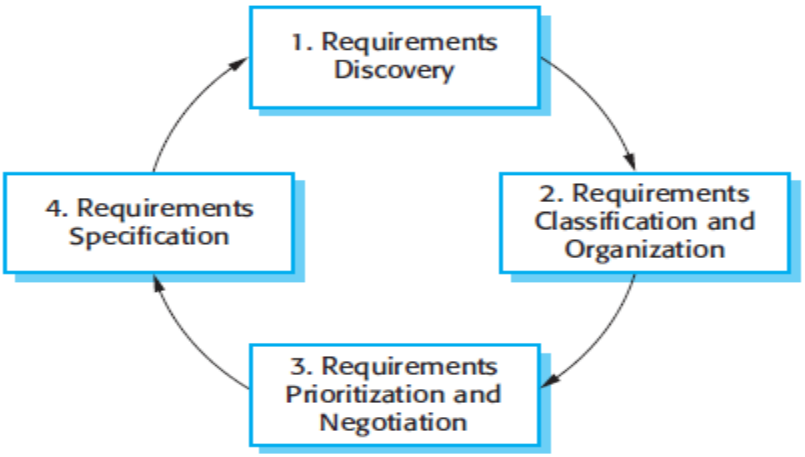
- The input to the feasibility study is a set of preliminary business requirements, an outline description of the system and how the system is intended to support business processes. The results of the feasibility study should be a report that recommends whether or not it is worth carrying on with the requirements engineering and system development process.

- Issues addressed by feasibility study

  - ✓ Gives focus to the project and outline alternatives.
  - ✓ Narrows business alternatives
  - ✓ Identifies new opportunities through the investigative process.
  - ✓ Identifies reasons not to proceed.
  - ✓ Enhances the probability of success by addressing and mitigating factors early on that could affect the project.
  - ✓ Provides quality information for decision making.
  - ✓ Provides documentation that the business venture was thoroughly investigated.
  - ✓ Helps in securing funding from lending institutions and other monetary sources.
  - ✓ Helps to attract equity investment.

- The feasibility study is a critical step in the business assessment process. If properly conducted, it may be the best investment you ever made Carrying out a feasibility study involves information assessment, information collection and report writing.

REQUIREMENTS ELICITATION AND ANALYSIS



The process activities are:

Requirements discovery: This is the process of interacting with stakeholders of the system to discover their requirements. Domain requirements from stakeholders and documentation are also discovered during this activity.

Requirements classification and organization: This activity takes the unstructured collection of requirements, groups related requirements, and organizes them into coherent clusters. The most common way of grouping requirements is to use a model of the system architecture to identify sub-systems and to associate requirements with each sub-system.

Requirements prioritization and negotiation: Inevitably, when multiple stakeholders are involved, requirements will conflict. This activity is concerned with prioritizing requirements and finding and resolving requirements conflicts through negotiation.

Requirements specification: The requirements are documented and input into the next round of the spiral.

Eliciting and understanding requirements from system stakeholders is a difficult process for several reasons:

- ✓ Stakeholders often don't know what they want from a computer system except in the most general terms; they may find it difficult to articulate what they want the system to do; they may make unrealistic demands because they don't know what is and isn't feasible.
- ✓ Stakeholders in a system naturally express requirements in their own terms and with implicit knowledge of their own work. Requirements engineers, without experience in the customer's domain, may not understand these requirements.
- ✓ Different stakeholders have different requirements and they may express these in different ways. Requirements engineers have to discover all potential sources of requirements and discover commonalities and conflict.
- ✓ Political factors may influence the requirements of a system. Managers may demand specific system requirements because these will allow them to increase their influence in the organization.
- ✓ The economic and business environment in which the analysis takes place is dynamic. It inevitably changes during the analysis process. The importance of particular requirements may change. New requirements may emerge from new stakeholders who were not originally consulted.

Requirements discovery

Requirements discovery (sometime called requirements elicitation) is the process of gathering information about the required system and existing systems, and distilling the user and system requirements from this information.

Sources of information during the requirements discovery phase include documentation, system stakeholders and specifications of similar systems.

Stakeholders range from end-users of a system through managers to external stakeholders
such as regulators, who certify the acceptability of the system.

For example, system stakeholders for the mental healthcare patient information system include:

- ✓ Patients whose information is recorded in the system.
- ✓ Doctors who are responsible for assessing and treating patients.
- ✓ Nurses who coordinate the consultations with doctors and administer some treatments.
- ✓ Medical receptionists who manage patients' appointments.
- ✓ IT staff who are responsible for installing and maintaining the system.
- ✓ A medical ethics manager who must ensure that the system meets current ethical guidelines for patient care.
- ✓ Healthcare managers who obtain management information from the system.
- ✓ Medical records staff who are responsible for ensuring that system information can be maintained and preserved, and that record keeping procedures have been
  properly implemented.

Interviewing

The requirements engineering team puts questions to stakeholders about the system that they currently use and the system to be developed. Requirements are derived from the answers to these questions.

Interviews may be of two types:

- ✓ Closed interviews, where the stakeholder answers a pre-defined set of questions.
- ✓ Open interviews, in which there is no pre-defined agenda. The requirements engineering team explores a range of issues with system stakeholders and hence develop a better understanding of their needs.

It can be difficult to elicit domain knowledge through interviews for two reasons:

- ✓ All application specialists use terminology and jargon that are specific to a domain. It is impossible for them to discuss domain requirements without using this terminology. They normally use terminology in a precise and subtle way that is easy for requirements engineers to misunderstand.
- ✓ Some domain knowledge is so familiar to stakeholders that they either find it difficult to explain or they think it is so fundamental that it isn't worth mentioning.

Effective interviewers have two characteristics:

1. They are open-minded, avoid pre-conceived ideas about the requirements, and are willing to listen to stakeholders. If the stakeholder comes up with surprising requirements, then they are willing to change their mind about the system.

2. They prompt the interviewee to get discussions going using a springboard question, a requirements proposal, or by working together on a prototype system. Saying to people 'tell me what you want' is unlikely to result in useful information. They find it much easier to talk in a defined context rather than in general terms.

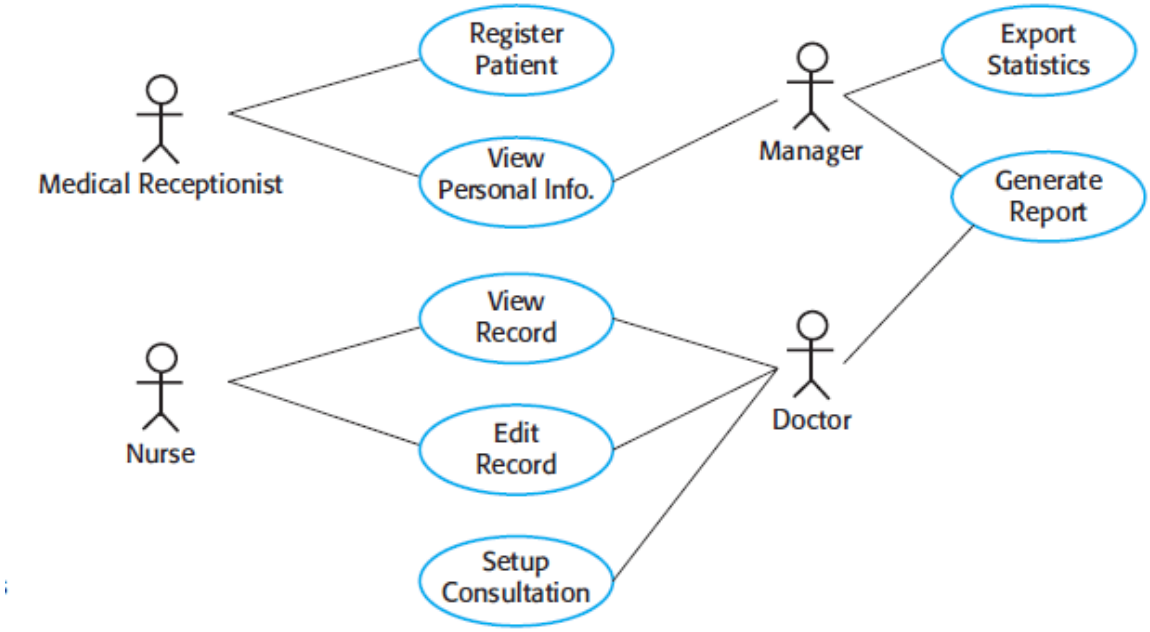
Scenarios

A scenario starts with an outline of the interaction. During the elicitation process, details are added to this to create a complete description of that interaction.

A scenario may include:

- ✓ A description of what the system and users expects when the scenario starts.
- ✓ A description of the normal flow of events in the scenario.
- ✓ A description of what can go wrong and how this is handled.
- ✓ Information about other activities that might be going on at the same time.
- ✓ A description of the system state when the scenario finishes.

Use cases



Use cases are documented using a high-level use case diagram. The set of use cases represents all of the possible interactions that will be described in the system requirements.
Actors in the process, who may be human or other systems, are represented as stick figures. Each class of interaction is represented as a named ellipse. Lines link the actors with the interaction. Optionally, arrowheads may be added to lines to show how the interaction is initiated.
Use cases identify the individual interactions between the system and its users or other systems. Each use case should be documented with a textual description. These can then be linked to other models in the UML that will develop the scenario in more
detail.

Ethnography

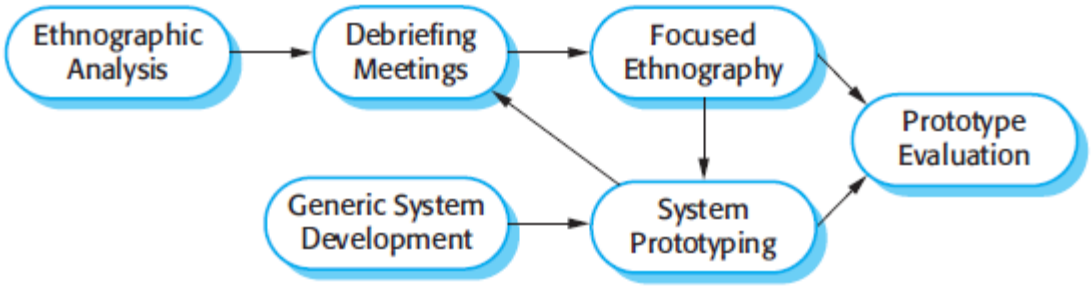
Ethnography is an observational technique that can be used to understand operational processes and help derive support requirements for these processes. An analyst immerses himself or herself in the working environment where the system will be used. The day-to-day work is observed and notes made of the actual tasks in which participants are involved. The value of ethnography is that it helps discover implicit system requirements that reflect the actual ways that people work, rather than the formal processes defined by the organization.

Ethnography is particularly effective for discovering two types of requirements:

Requirements that are derived from the way in which people actually work, rather than the way in which process definitions say they ought to work.
Requirements that are derived from cooperation and awareness of other people's activities.

Ethnography can be combined with prototyping

The ethnography informs the development of the prototype so that fewer prototype refinement cycles are required. Furthermore, the prototyping focuses the ethnography by identifying problems and questions that can then be discussed with the ethnographer.



REQUIREMENTS VALIDATION

Requirements validation is the process of checking that requirements actually define the system that the customer really wants.

The cost of fixing a requirements problem by making a system change is usually much greater than repairing design or coding errors. The reason for this is that a change to the requirements usually means that the system design and implementation must also be changed.

During the requirements validation process, different types of checks should be carried out on the requirements in the requirements document. These checks include:

- **Validity checks** A user may think that a system is needed to perform certain functions. However, further thought and analysis may identify additional or different functions that are required. Systems have diverse stakeholders with different needs and any set of requirements is inevitably a compromise across the stakeholder community.
- **Consistency checks** Requirements in the document should not conflict. That is, there should not be contradictory constraints or different descriptions of the same system function.

- **Completeness checks** The requirements document should include requirements that define all functions and the constraints intended by the system user.
- **Realism checks** Using knowledge of existing technology, the requirements should be checked to ensure that they can actually be implemented. These checks should also take account of the budget and schedule for the system development.
- **Verifiability** To reduce the potential for dispute between customer and contractor, system requirements should always be written so that they are verifiable. This means that you should be able to write a set of tests that can demonstrate that the delivered system meets each specified requirement.

There are a number of requirements validation techniques that can be used individually or in conjunction with one another:
1. Requirements reviews: The requirements are analyzed systematically by a team of reviewers who check for errors and inconsistencies.
2. Prototyping: In this approach to validation, an executable model of the system in question is demonstrated to end-users and customers. They can experiment with this model to see if it meets their real needs.
3. Test-case generation: Requirements should be testable. If the tests for the requirements are devised as part of the validation process, this often reveals requirements problems. If a test is difficult or impossible to design, this usually means that the requirements will be difficult to implement and should be reconsidered. Developing tests from the user requirements before any code is written is an integral part of extreme programming.

REQUIREMENTS MANAGEMENT

Once a system has been installed and is regularly used, new requirements inevitably emerge. It is hard for users and system customers to anticipate what effects the new system will have on their business processes and the way that work is done.

There are several reasons why change is inevitable:

1. The business and technical environment of the system always changes after installation. New hardware may be introduced, it may be necessary to interface the system with other systems, business priorities may change (with consequent changes in the system support required), and new legislation and regulations may be introduced that the system must necessarily abide by.

2. The people who pay for a system and the users of that system are rarely the same people. System customers impose requirements because of organizational and budgetary constraints. These may conflict with end-user requirements and, after

delivery, new features may have to be added for user support if the system is to meet its goals.

3. Large systems usually have a diverse user community, with many users having different requirements and priorities that may be conflicting or contradictory. The final system requirements are inevitably a compromise between them and, with experience, it is often discovered that the balance of support given to different users has to be changed.

Requirements management planning

Planning is an essential first stage in the requirements management process.

During the requirements management stage, the following is decided

Requirements identification: Each requirement must be uniquely identified so that it can be cross-referenced with other requirements and used in traceability assessments.
Change management process : This is the set of activities that assess the impact and cost of changes. I discuss this process in more detail in the following section.
Traceability policies: These policies define the relationships between each requirement
and between the requirements and the system design that should be recorded. The traceability policy should also define how these records should be maintained.
Tool support Requirements management: involves the processing of large amounts of information about the requirements. Tools that may be used range from specialist requirements management systems to spreadsheets and simple database systems.

Requirements management needs automated support and the software tools for this should be chosen during the planning phase.

Requirements storage: The requirements should be maintained in a secure, managed data store that is accessible to everyone involved in the requirements engineering process.
Change management: The process of change management is simplified if active tool support is available.
Traceability management: Tool support for traceability allows related requirements to be discovered. Some tools are available which use natural language processing techniques to help discover possible relationships between requirements.

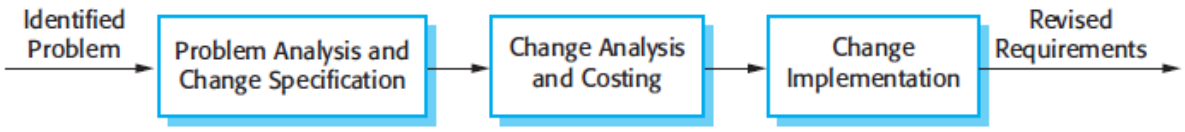
Requirements change management

Requirements change management should be applied to all proposed changes to a system's requirements after the requirements document has been approved.

There are three principal stages to a change management process:
1. Problem analysis and change specification: The process starts with an identified requirements problem or, sometimes, with a specific change proposal. During this stage, the problem or the change proposal is analyzed to check that it is valid. This analysis is fed back to the change requestor who may respond with a more specific requirements change proposal, or decide to withdraw the request.

2. Change analysis and costing: The effect of the proposed change is assessed using traceability information and general knowledge of the system requirements. The cost of making the change is estimated both in terms of modifications to the requirements document and, if appropriate, to the system design and implementation. Once this analysis is completed, a decision is made whether or not to proceed with the requirements change.

3. Change implementation: The requirements document and, where necessary, the system design and implementation, are modified. You should organize the requirements document so that you can make changes to it without extensive rewriting or reorganization. As with programs, changeability in documents is achieved by minimizing external references and making the document sections as modular as possible. Thus, individual sections can be changed and replaced without affecting other parts of the document.



DATA DICTIONARY:

A data dictionary stores information about data items found in a DFD.

Data dictionary features

Name: Identifies the data item
Alias: Identifies other names, abbreviations used to identify the data item
Data structure (type): Type of data (eg: int,char)
Description: Indicates how (why) a data item is used
Duration : (begins) Life span of data (when created)

Accuracy: High, medium and low accuracy
Range of values: Allowable values of data item
Data flows: Identifies process that generate/receive data

A data dictionary supplies information such as data typing, required accuracy of data useful to designers and implementers. Name, alias, type and description indicate how to identify, possible other names, types of data and what and how the data are used respectively. Duration, accuracy and range of values specify life span, required precision in measurement and all possible values of data items respectively. Data flows specify processes that generate or receive the data. This provides another useful tool in validating a design (making sure that the requirements for a data item are satisfied by a design and by code).
In the case where data are derived from real-time system, data items can have timing constraints
specifying the length of time before the data become out-of-date. also be included depending on the type of model being developed.

Advantages of data dictionary:

- It is a mechanism for name management
- It serves as a store of organizational information

Analysis model:
The analysis model is a set of models, technically represent a system in order to validate software requirements. Analysis modeling is the process of representing the requirements for data, function and behaviour in the combined form of text and diagrams for the purpose of easy
understanding and reviews.
Objectives of analysis model

- To describe and define the customer requirements
- To establish a basis for the creation of a software design
- To define a set of requirements that can be validated once the software is built.

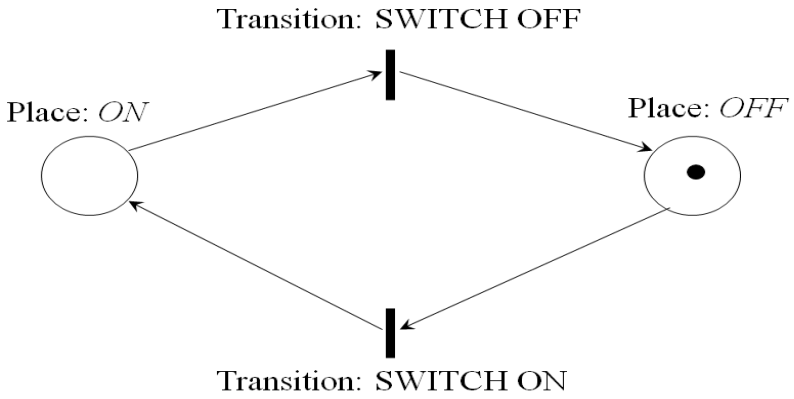
Analysis modeling is not a single step, but it is a collective activity which is executed step by step. The diagram depicts the functional view of analysis modeling

- Data modeling: defines data objects, attributes and relationships
- Functional modeling: defines data flow and its transformations within a system
- Behavioural modeling: depicts the impact of every event

Once the preliminary models are created, they are refined and analyzed to assess their clarity,
completeness and consistency.
PETRI NETS

- A Petri Nets (PN) comprises places, transitions, and arcs
  - Places are system states
  - Transitions describe events that may modify the system state

  - Arcs specify the relationship between places
- Tokens reside in places, and are used to specify the state of a PN

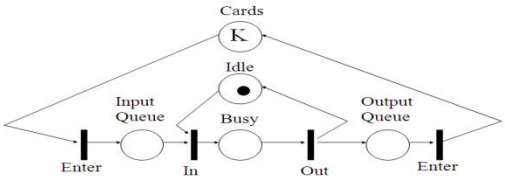


- Two places: Off and On
- Two transitions: Switch Off and Switch On
- Four arcs
- The off condition is true
- A transition can fire if an input token exists
  - One token is moved from the input place to the output place.

PN properties

- 8-tuple mathematical model
  - M={P,T,I,O,H,PAR,PRED,MP}
  - P - the set of places
  - T - the set of transitions
  - I,O,H - Input, output, inhibition function
  - PAR - the set of parameters
  - PRED - Predicates restricting parameter range
  - PM - Parameter value
- From this linear algebra can be used to analyze a network

Manufacturing Example



- Very rich modeling
- Easily capable of modeling software project, requirements, architectures, and processes

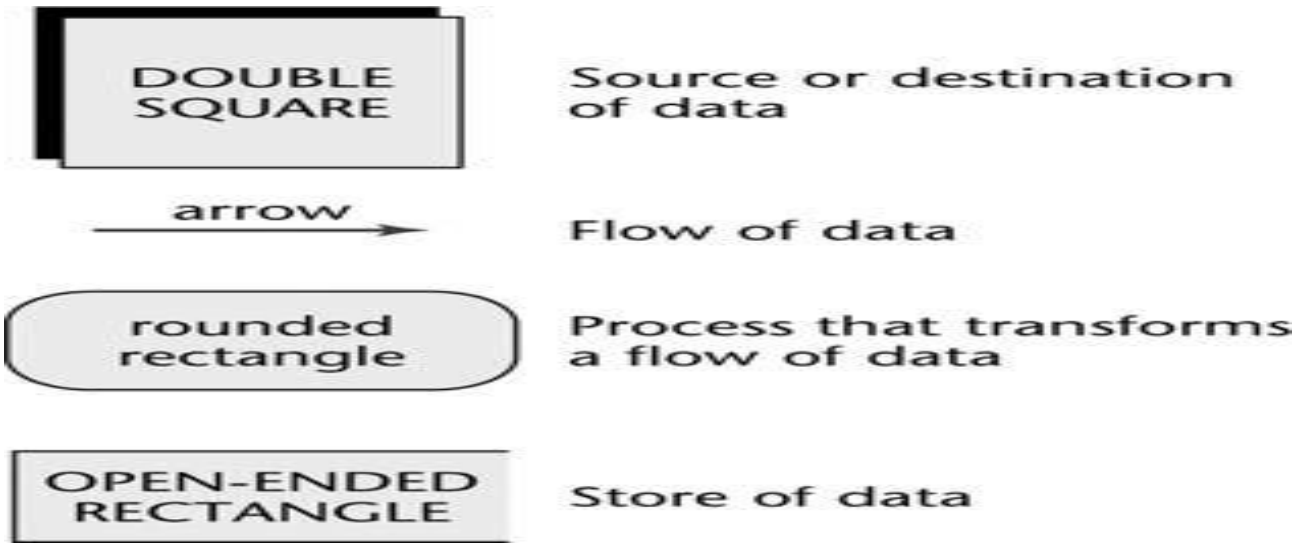
- Drawbacks
  - Complex rules
  - Analysis quite complex

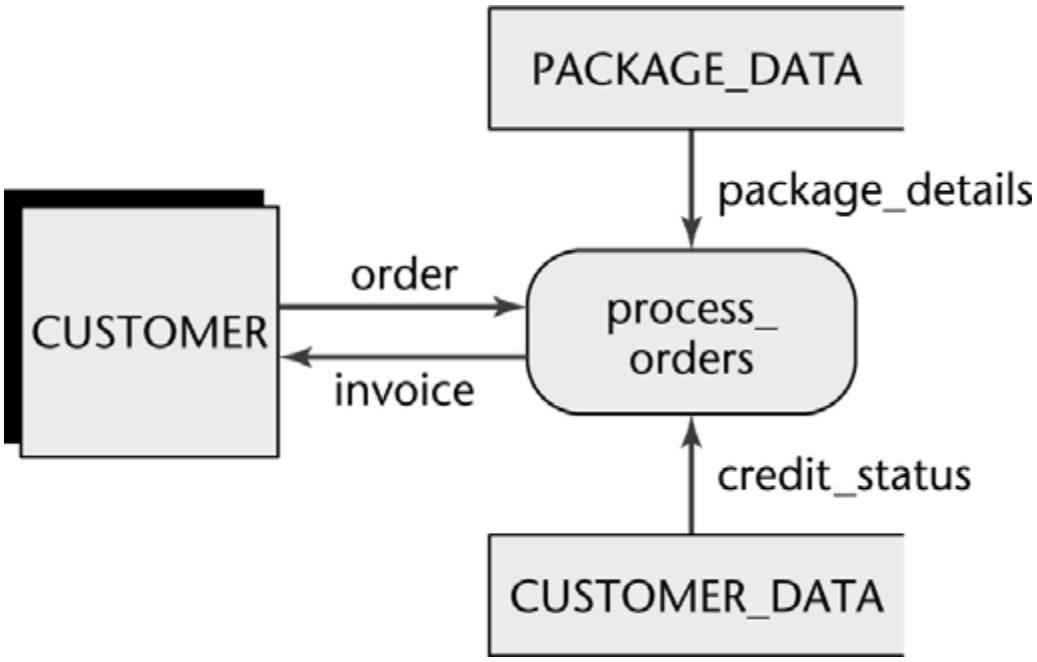
STRUCTURED SYSTEMS ANALYSIS

● Structures system analysis — A nine-step technique to analyze client's needs
● Step-wise refinement is used in many of steps
● Step 1: Draw the Data Flow Diagram (DFD)
  - ✓ A pictorial representation of all aspects of the logical data flow
  - ✓ Logical data flow — What happens
  - ✓ Physical data flow — How it happens
  - ✓ Any non-trivial product contains many elements
  - ✓ DFD is developed by stepwise refinement
  - ✓ For large products a hierarchy of DFDs instead of one DFD
  - ✓ Constructed by identifying data flows: Within requirements
  - ✓ document or rapid prototype

- Four basic symbols
  - ✓ Source or destination of data
  - ✓ (double-square box)
  - ✓ Data flow (arrow)
  - ✓ Process (rounded rectangle)
  - ✓ Data store (open-ended
  - ✓ rectangle)

● Step 2: Decide what sections to computerize and how (batch or online)
  - ✓ □ Depending on client's needs and budget limitations
  - ✓ □ Cost-benefit analysis is applied





- Step 3: Determine details of data flows
  - ✓ Decide what data items must go into various data flows
  - ✓ Stepwise refinement of each flow
  - ✓ For larger products, a data dictionary is generated
- Step 4: Define logic of processes
  - ✓ Determine what happens within each process
  - ✓ Use of decision trees to consider all cases
- Step 5: Define data stores
  - ✓ Exact contents of each store and its representation (format)
- Step 6: Define physical resources
  - ✓ File names, organization (sequential, indexed, etc.), storage medium, and records
  - ✓ If a database management system (DBMS) used: Relevant information for each table
- Step 7: Determine input-output specifications
  - ✓ Input forms and screens
  - ✓ Printed outputs
- Step 8: Determine sizing
  - ✓ Computing numerical data to determine hardware requirements
  - ✓ Volume of input (daily or hourly)
  - ✓ Frequency of each printed report and its deadline
  - ✓ Size and number of records of each type to pass between CPU
  - ✓ and mass storage
  - ✓ Size of each file

- Step 9: Determine hardware requirements
  - ✓ Use of sizing information to determine mass storage
  - ✓ requirements
  - ✓ Mass storage for backup
  - ✓ Determine if client's current hardware system is adequate
- After approval by client: Specification document is handed to design team, and software process continues