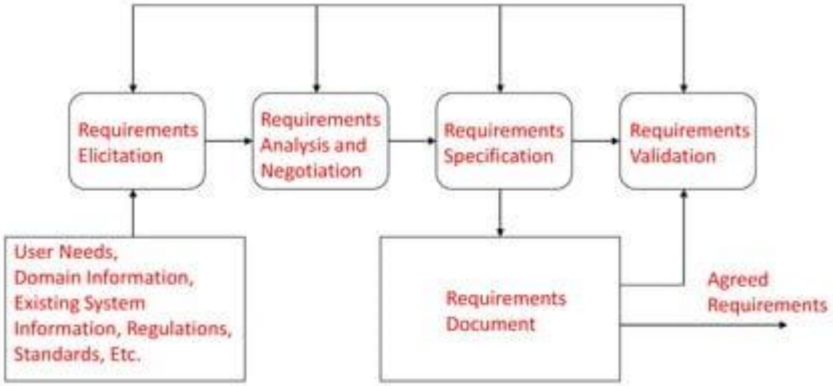
Requirements Analysis and Modeling

Syllabus:

Lecture no	Content	Duration (Hr)	Self-Study (Hrs)
1	Requirement Elicitation, Software requirement specification (SRS)	2	1
2	Data Flow Diagram(DFD), Feasibility Analysis, Cost- Benefit Analysis	2	1
3	Developing Use Cases (UML), Requirement Model – Scenario-based model	2	1
4	Class-based model, Behavioral model.	2	1

Requirements Engineering Activities


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

Requirements Elicitation

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

- Requirements acquisition or requirements discovery

Requirements Analysis and Negotiation

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

Requirements Analysis and Negotiation

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

Requirements Specification

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

Requirements Document

- Detailed descriptions of the required software system in form of requirements is captured in the requirements document

- Software designers, developers and testers are the primary users of the document

Requirements Validation

- It involves reviewing the requirements model for consistency and completeness

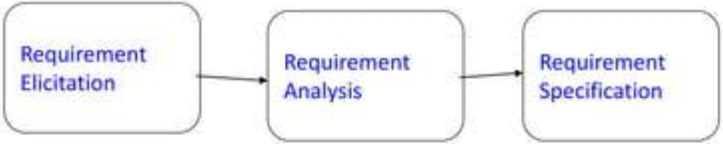
- This process is intended to detect problems in the requirements document, before they are used as a basis for the system development

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

Requirement Analysis

- Requirement Analysis bridges the gap b/w Elicitation and requirement specification

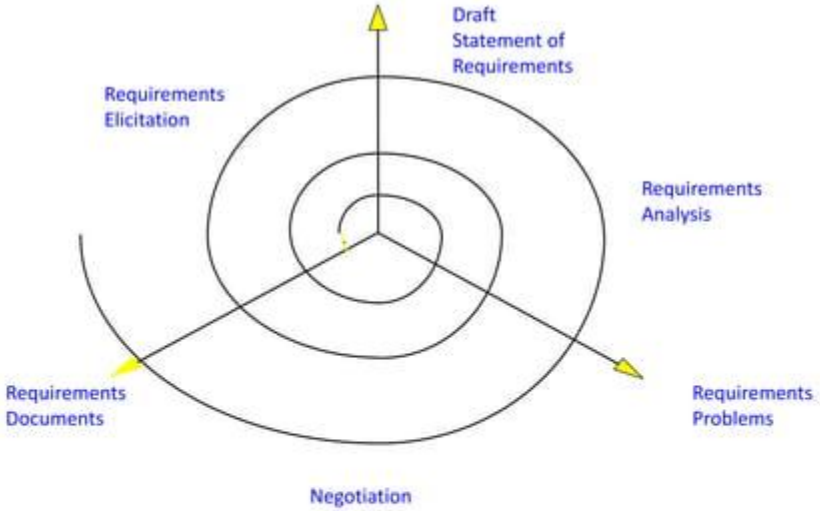


Requirements Analysis

- What is it?
  - The process by which customer needs are understood and documented.
  - Expresses "what" is to be built and NOT "how" it is to be built.

- Example 1:
  - The system shall allow users to withdraw cash. [*What*?]
- Example 2:
  - A sale item's name and other attributes will be stored in a hash table and updated each time any attribute changes. [*How*?]

Iterative Aspects of Elicitation, Analysis, and Negotiation



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

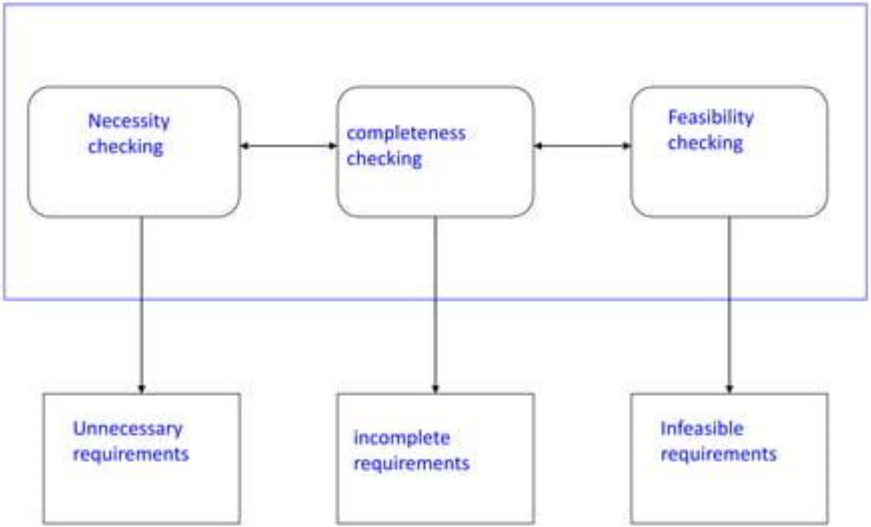
Requirements Analysis

- 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

Requirements Analysis

- 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

Requirements Analysis Process



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

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

Analysis Checklists

- 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

Checklist Items

- Premature design
- Combined requirements
- Unnecessary requirements
- Use of non-standard hardware

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?

Checklist Items Description

- 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

Checklist Items

- Conformance with business goals
- Requirements ambiguity
- Requirements realism
- Requirements testability

Checklist Items Description

- 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?

Checklist Items Description

- 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?

Up-till Now

- Discussed requirements analysis, which is an iterative activity and checks for incomplete and inconsistent requirements
- Studied analysis checklists, and will continue our discussion of requirements analysis.
- We'll talk about requirements negotiation.

Requirements Negotiations

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

Requirements Negotiation

- In planning a requirements engineering process, it is important to leave enough time for negotiation. Finding an acceptable compromise can be time-consuming

Requirements Negotiation

- 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 prioritization
- Requirements agreement

Requirements Discussion

- Requirements which have been highlighted as problematic are discussed and the stakeholders involved present their views about the requirements

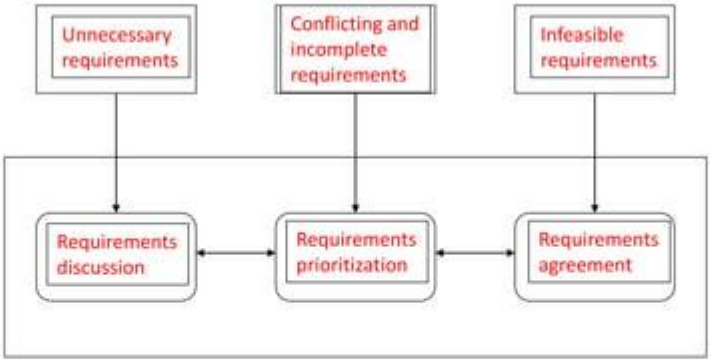
Requirements Prioritization

- 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



Comments on Requirements Negotiation

- In principle, requirements negotiation should be an objective process
- The requirements for the system should be based on technical and organizational needs

Requirements Analysis

- C- and D-Requirements
  - C-: Customer wants and needs; expressed in language understood by the customer.
  - D-: For the developers; may be more formal.

Requirements Analysis

Why document requirements?

- Serves as a contract between the customer and the developer.
- Serves as a source of test plans.
- Serves to specify project goals and plan development cycles and increments.

Requirements Analysis

- Roadmap:
  - Identify the customer.
  - Interview customer representatives.
  - Write C-requirements, review with customer, and update when necessary.
  - Write D-requirements; check to make sure that there is no inconsistency between the C- and the D-requirements.

SRS (Software Requirement Specification)

What is an SRS ?

- SRS is the official statement of what the system developers should implement.
- SRS is a complete description of the behaviour of the system to be developed.
- SRS should include both a definition of user requirements and a specification of the system requirements.
- The SRS fully describes what the software will do and how it will be expected to perform.

Purpose of SRS

- The SRS precisely defines the software product that will be built.
- SRS used to know all the requirements for the software development and thus that will help in designing the software.
- It provides feedback to the customer.

SRS Format

Introduction

- Purpose
- Document Conventions
- Product Scope
- Reference

Overall Description

- Product Perspective
- Product Functions
- User Classes and Characteristics
- Operating Environment
- Design and Implementation Constraints
- User Documentation
- Assumptions and Dependencies

External Interface Requirements

- User Interfaces
- Hardware Interfaces
- Software Interfaces
- Communications Interfaces

Continue…

System Features

- System Feature 1
- System Feature 2 (and so on)

Other Nonfunctional Requirements

- Performance Requirements
- Safety Requirements
- Security Requirements
- Software Quality Attributes
- Business Rules

Other Requirements

- Appendix A: Glossary
- Appendix B: Analysis Models

Types of reader for requirement specification

System Customers

Specify the requirements and read them to check that they meet their needs. Customer specify changes to the requirements .

Managers

Use the Requirements Document to plan a bid for the system and to plan system development process.

System Engineers

Use the requirements to understand what system is to be deployed.

Continue…

System test engineers

- Use the requirements to develop validation tests for the system

System Maintains engineers

- Use the requirements to understand the system and the relationship between its parts

Characteristics of SRS

- **Correct :** Every requirement given in SRS is a requirement of the software.
- **Unambiguous:** Every requirement has exactly one interpretation.
- **Complete:** Includes all functional, performance, design, external interface requirements; definition of the response of the software to all inputs.
- **Consistent:** Internal consistency.
- **Ranked importance:** Essential vs. desirable.

Continue...

- **Verifiable:** A requirement is verifiable if and only if there exists some finite cost effective process with which a person or machine can check that the SW meets the requirement.
- **Modifiable:** SRS must be structured to permit effective modifications (e.g. don't be redundant, keep requirements separate)
- **Traceable:** Origin of each requirement is clear.

Benefits

The IEEE 830 standard defines the benefits of a good SRS:

- Establish the basis for agreement between the customers and the suppliers on what the software product is to do.
- Reduce the development effort.
- Provide a basis for estimating costs and schedules.
- Provide a baseline for validation and verification.
- Serve as a basis for enhancement.

Introduction

- SRS: Software Requirements Specification

1. Software Requirements Specification (SRS) is a document that lays out the description of the software that is to be developed as well as the intention of the software under development.
2. Software requirements specification shows what the software is supposed to do as well as how it is supposed to perform. It is written down before the actual software development work starts.
3. An SRS contains the goals and measurable outcomes of the development process.
4. With a single glance at an SRS, any stakeholder understands what the finished product should be.

Why SRS is important?

Advantages

Software requirements specification(SRS) is important for developers because it minimizes the amount of time and effort developers have to expend to achieve desired software goals. It thus reduces development cost.

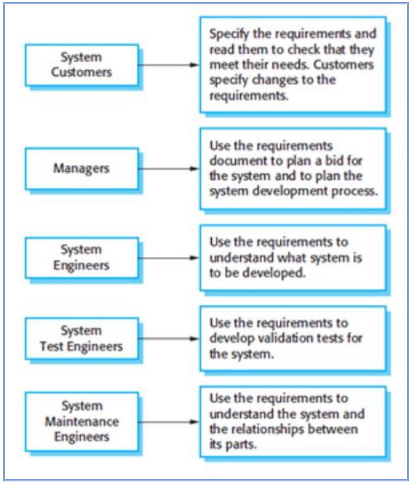
benefits the client company because the lesser the development cost, the lesser the developers will charge from the client.

If composed properly, an SRS ensures that there is less possibility of future redesigns as there is less chance of mistake on the part of developers as they have a clear idea on the functionalities and externalities of the software.

It also helps clear any communication problems between the client and the developer. Furthermore, an SRS serves to form a foundation of mutual agreement between the client and the developer (supplier).

It also serves as the document to verify the testing processes.

Who uses the SRS document?



System Customers
Specify the requirements and read them to check that they meet their needs. Customers specify changes to the requirements.
Managers
Use the requirements document to plan a bid for the system and to plan the system development process.
System Engineers
Use the requirements to understand what system is to be developed.
System Test Engineers
Use the requirements to develop validation tests for the system.
System Maintenance Engineers
Use the requirements to understand the system and the relationships between its parts.

Fig 1. Users of a requirements document [SOMMERVILLE2010]

When is an SRS said to be good?

If it tries to address the following:

Functionality. What is the software supposed to do?

External interfaces. How does the software interact with people, the system's hardware, other hardware, and other software?

Performance. What is the speed, availability, response time, recovery time of various software functions, etc.?

Attributes. What are the portability, correctness, maintainability, security, etc. considerations?

Design constraints imposed on an implementation. Are there any required standards in effect, implementation language, policies for database integrity, resource limits, operating environment(s) etc.?

8 Characteristics of Good SRS: https://docs.google.com/document/d/1NeDqK4w0GGRO5zFPeKviqkzR65iCWQZkif7z5hGAP4A/edit?usp=sharing

Template of an SRS Document

Format by IEEE (Institute of Electrical and Electronics Engineers)

1. Introduction

1.1 Purpose
a)This gives the purpose of the SRS document, not the software itself.
b)It also states how much of the software is covered by the document, particularly saying whether it describes the entire software system or only a part of it.
c)It also states the intended readers of the document.

1.2 Document conventions
This covers whether the documentation follows any particular format.

Continued..

1.3 Intended audience and reading suggestions

a)This segment lists the intended audience and provides reading suggestions.
b)For example, a document might be for both developers and project managers.
c)This part suggests how developers should read and also how project managers should read this document.
d)The sequence and importance of different segments might be different for developers and project managers.

Continued..

1.4 Product Scope

a)This segment describes the software in brief. Its purpose, objectives etc.
b)It further states how the software ties into corporate aims and objectives.

1.5 References

This segment lists any and all websites and documents that were referred to in this document. The reference should be clear enough that the reader can access the referenced document or site after reading the reference list.

Continued..

2. Overall description
2.1 Product Perspective

a)This part describes the context within which the product (software) is being built.
b)It also shows if the software is part of a product family, a replacement for an already existing member, or a completely new and unique product.
c)Also, if the SRS only describes a part of a larger system, then this part will lay out the requirements from the larger system for this part to operate effectively.
A diagram should be drawn here to show the chief components of the entire system, the interlinking connections, and also the interfaces.

Continued..

2.2 Product Functions

a)A list of the major functions that product is to perform, or lets the users perform.
b)This should be readable and easily understandable by all the intended readers of the SRS.

2.3 User classes and characteristics

a)This part identifies the user classes that will use the product.
b)The classes may be categorized based on which functions they use, usage frequency, technical expertise, experience etc.
c)The relevant characteristics of each class should be given here. Also the most important user classes should be identified here.

Continued..

2.4 Operating Environment

a)This segment describes the environment in which the product will operate, such as the platform (hardware) operating system version etc.
b)Also any other components the application will cohabit.

2.5 Design and Implementation Constraints

a)This segment lays out any constraints or issues that will limit options to developers.
b)Some of these might be hardware limitations, interfaces, particular technologies, parallel operations, language requirements, regulatory policies, corporate policies, security provisions etc.

Continued..

2.6 User Documentation

This part shows the user manuals, tutorials etc that will be provided along with the software.

2.7 Assumptions and Dependencies

This part lists any assumptions that could affect the requirements stated in the SRS. The software would not work to the desired level if these assumptions are incorrect or change. Also listed here is any dependency the product will have on external factors.

Continued..

External Interface Requirements
3.1 User Interfaces
a)This part describes the logical characteristics of each interface between the software and the user.
b)This might include screen image samples, Graphics User Interface standards, screen layout constraints, buttons and functions, keyboard shortcuts, message displays etc.
c)The software components for which user interface is need will also be defined.

3.2 Hardware Interfaces

This segment describes the logical and physical characteristics of each interface between the software and hardware components of the system.

Continued..

3.3 Software Interfaces
a)This segment describes the connections between the product and any other specific software components, be it operating systems, databases libraries, etc.
b)Data items and messages going into the system and going is identified and the purpose of each is described.

3.4 Communications Interfaces
a)This part describes any communications functions required by the software, such as e-mail, web browser, network server communications protocols, electronic forms etc. Any communication standards to be used is also identified.
b)Communication security and/or encryption issues are specified.

4. System Features
This part lists the major services the product will provide. This part may be organized by use case, mode of operation, user class, object class, functional hierarchy, a combination of these etc.
An example might be:

4.1 System Feature 1 (basically here the name of the feature will be stated)

4.1.1 Description and Priority

4.1.2 Stimulus/Response Sequences

4.1.3 Functional Requirements

4.2 System Feature 2 (and so and so)

5. Other Non-functional Requirements
5.1 Performance Requirements

Requirements such as RAM requirements, CPU speed etc. Basically, these will ensure the software performs smoothly and without any problems

5.2 Safety Requirements

Requirements concerned with possible loss, damage or harm that could result from the use of the product are stated here. Any safeguards that must be implemented should also be defined. External policies or regulations stating safety issues that affect the product’s design or use should be referred. Any safety certifications that must be satisfied is defined.

5.3 Security requirements

This part specifies any requirements regarding privacy and safety, as well as protection of data created or used by the product. Any user identity authentication requirements are stated here. Any safety certifications that needs to be satisfied is also defined.

5.4 Software Quality Attributes

Any additional quality characteristics of the product that might be important are specified here. Such as: adaptability, interoperability, availability, correctness, maintainability, robustness, usability, testability etc. These are written in a specific, verifiable and quantitative way.

Continued..

5.5 Business Rules

Operating principles of the product are stated here. Such as which individuals can perform which functions under different circumstances.

6. Other requirements
Any requirements not stated in the SRS elsewhere are stated here. The requirements may include database requirements, legal requirements etc.

Example of SRS Document

https://krazytech.com/projects/sample-software-requirements-specificationsrs-report-airline-database

Creating Data Flow Diagram

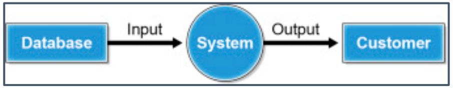
Table of Contents

- What are Data Flow Diagrams?
- Why they are useful?
- Symbols used in a DFD
  - Process
  - Data Flow
  - External entities
  - Data stores
- Rules for creating DFDs
- Levels of DFD
- Common mistakes while drawing DFDs
- A simple example of DFD (context diagram)
- Example of DFD (till level 1)

What are Data Flow Diagrams?

- Data Flow Diagrams model events and processes i.e. activities which transform data within a system.

- A Data Flow Diagram, is a pictorial representation of data that flows from one process to another process inside a system(generally an information system) along with mentioning how data flows into the system and out of the system.



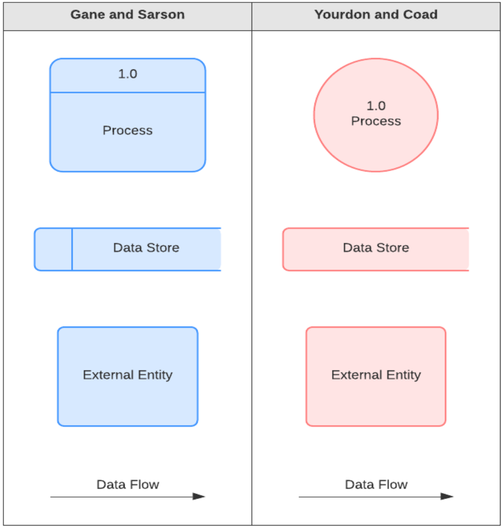
Why they are useful?

- A DFD shows an abstract or functional view of the system to be developed.

- The graphical representation easily overcomes any gap between ’user and system analyst’ and ‘analyst and system designer’ in understanding a system.

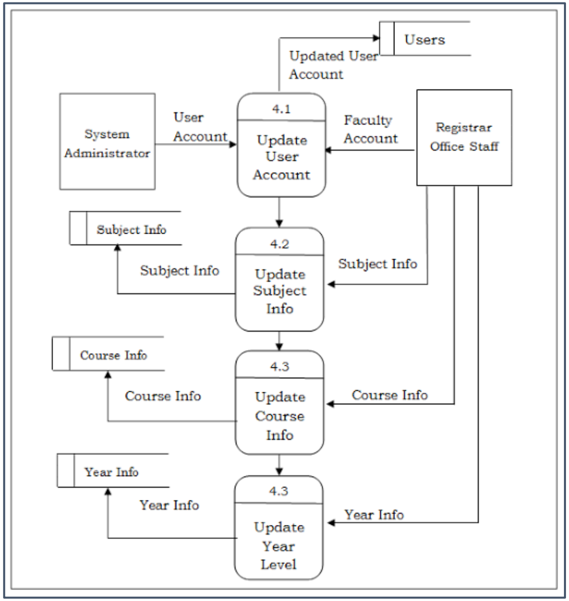
Symbols used in a DFD

- Process/es: denoted by a process box.
- Data flow: denoted by a labelled arrow.
- Data store: denoted by an open rectangle or two parallel lines.
- Entity: denoted by a rectangle



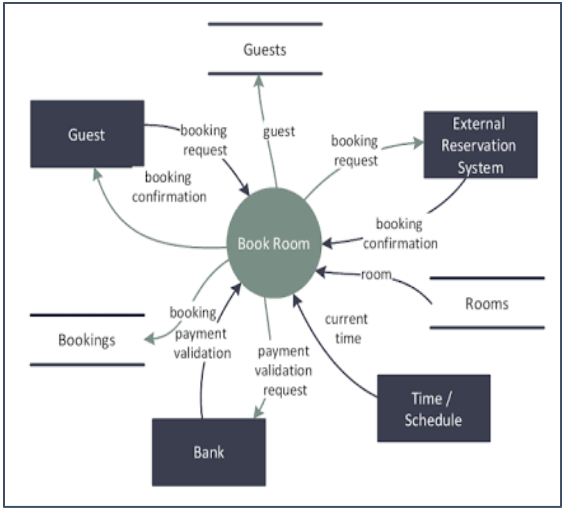
Processes

- Processes transform or manipulate data.
- Processes are 'black boxes' - we don't know what is in them until they are decomposed.
- The name of the process is usually given in such a way that represents the functionality of the process.



Data Flows

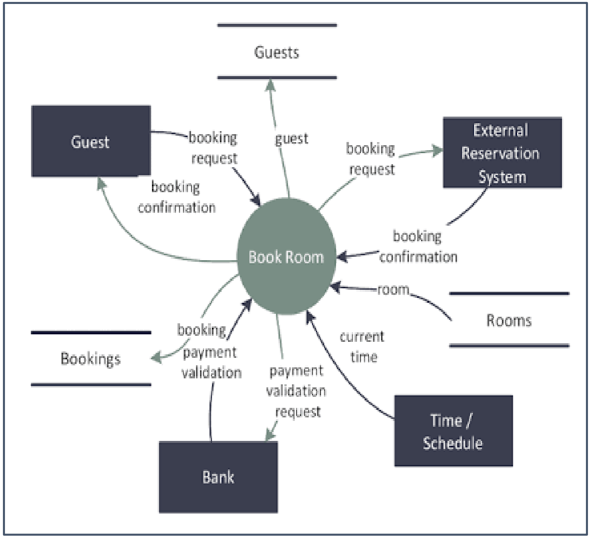
- Depicts data/information flowing to or from a process.
- The arrows must either start and/or end at a process box.
- It is impossible for data to flow from data store to data store except via a process, and
- External entities are not allowed to access data stores directly.
- Arrows must be named.



External Entities

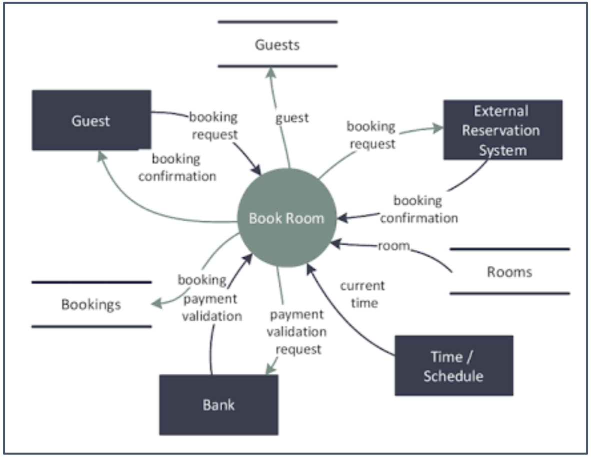
- Also known as 'External sources/recipients, are things (eg: people, machines, organisations etc.) which contribute data or information to the system or which receive data/information from it.

- The name given to an external entity represents a Type not a specific instance of the type.



Data Stores

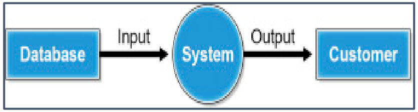
- Are some location where data is held temporarily or permanently.
- In physical DFDs there can be 4 types.
  - D = computerised Data
  - M = Manual, e.g. filing cabinet.
  - T = Transient data file, e.g. temporary program file
  - T(M) = Transient Manual, e.g. in-tray, mail box.



Rules for creating DFDs

- Entities are either 'sources of' or 'sinks' for data input and outputs - i.e. they are the originators or terminators for data flows.
- Data flows from Entities must flow into Processes.
- Data flows to Entities must come from Processes.
- Processes and Data Stores must have both inputs and outputs (What goes in must come out!)
- Inputs to Data Stores only come from Processes.
- Outputs from Data Stores only go to Processes.
- External Entities only come at level 0 DFD.

Levels of DFD



- Level 0 or context diagram: represents broad overview of a system.
- The entire system is shown as single process and also the interactions of external entities with the system are represented in context diagram.
- Further we split the process in next levels into several numbers of processes to represent the detailed functionalities performed by the system.(level 2, level 2 etc.)

Numbering of processes

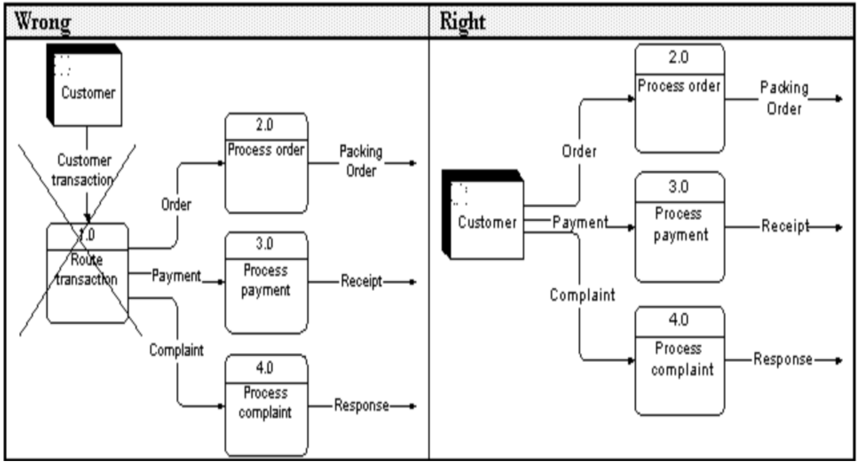
Numbering of processes :

If process ‘p’ in context diagram is split into 3 processes ‘p1’, ‘p2’and ‘p3’ in next level then these are labeled as 0.1, 0.2 and 0.3 in level 1 respectively.

Let the process ‘p3’ is again split into three processes ‘p31’, ‘p32’ and ‘p33’ in level 2, so, these are labeled as 0.3.1, 0.3.2 and 0.3.3 respectively and so on.

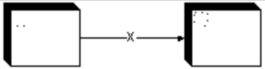
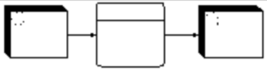
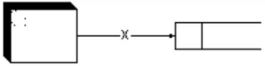
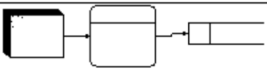
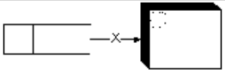
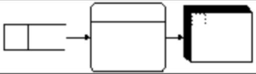
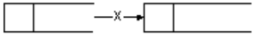
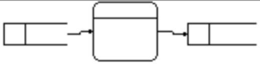
Common mistakes 1

Processing steps that do not change data, do not belong to DFDs.



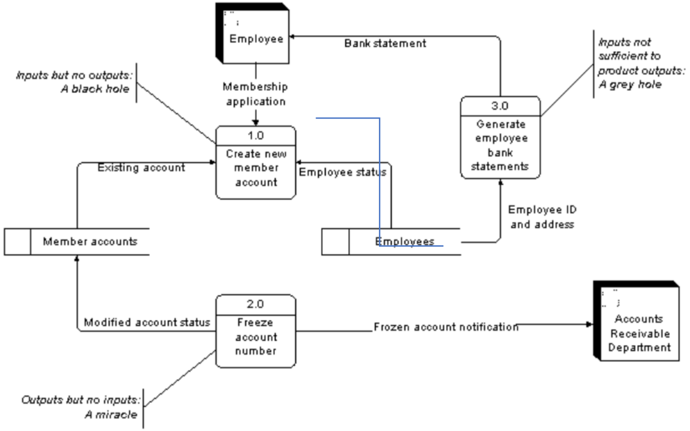
Common mistake 2

Common Data flow
diagram mistakes

Wrong	Right	Description
		A source or a sink cannot provide data to another source or sink without some processing occurring.
		Data cannot move directly from a source to a data store without being processed.
		Data cannot move directly from a data store to a sink without being processed.
		Data cannot move directly from one data store to another without being processed.

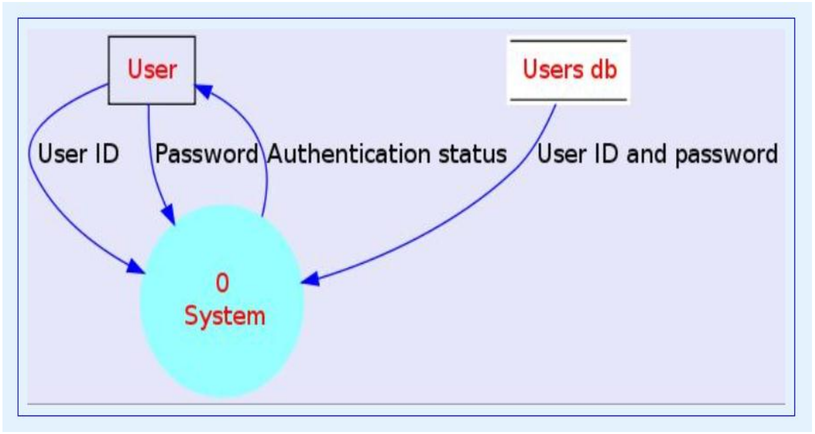
Common mistake 3

Black holes, grey holes and miracles


Employee
Bank statement
Inputs not sufficient to product outputs: A grey hole
Inputs but no outputs: A black hole
Membership application
3.0
Generate employee bank statements
1.0
Create new member account
Existing account
Employee status
Employee ID and address
Member accounts
Employees
2.0
Freeze account number
Modified account status
Frozen account notification
Accounts Receivable Department
Outputs but no inputs: A miracle

A simple example of DFD

Draw a context-level DFD to depict the typical user authentication process used by any system. An user gives two inputs -- user name and password.


User
Users db
User ID
Password
Authentication status
User ID and password
0
System

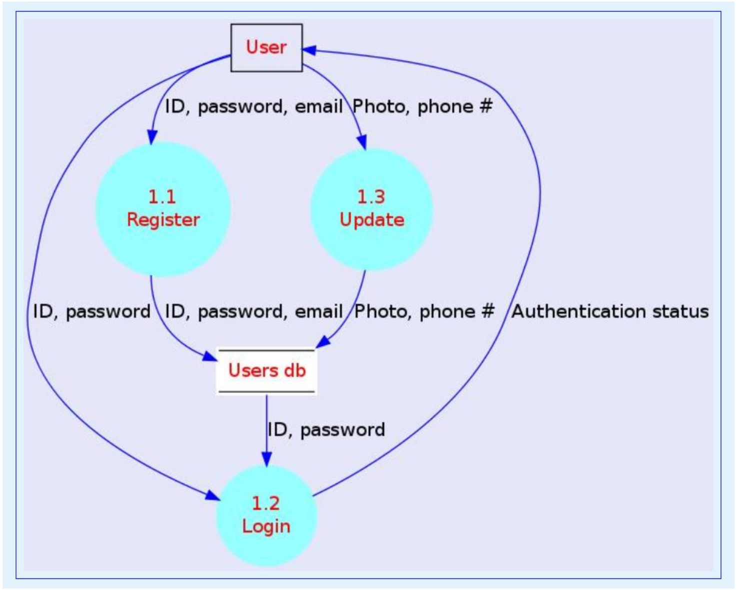
Example 2 DFD

DFD for a Social Networking site

The Absolute Beginners Inc. is planning to launch a revolutionary social networking site, EyeCopy. You have been entrusted with designing a DFD for the proposed application. In particular, you have been asked to show the following scenarios:

- User registration
- User login
- Profile update

Draw a Level 1 DFD to depict the above data flow and the corresponding processes. Should there be any data store in the DFD?


User
ID, password, email
Photo, phone #
1.1 Register
1.3 Update
ID, password
ID, password, email
Photo, phone #
Authentication status
Users db
ID, password
1.2 Login

Feasibility Analysis :

- The objective of the feasibility study is to establish the reasons for developing the software that is acceptable to users, adaptable to change and conformable to established standards.
- Feasibility Study in Software Engineering is a study to evaluate feasibility of proposed project or system.
- **Types of Feasibility :**
- Technical Feasibility
- Operational Feasibility
- Economic Feasibility
- Legal Feasibility
- Schedule Feasibility

- **Technical Feasibility :** Technical feasibility study gives report whether there exists correct required resources and technologies which will be used for project development. Along with this, feasibility study also analyzes technical skills and capabilities of technical team, existing technology can be used or not, maintenance and up-gradation is easy or not for chosen technology etc.
- **Operational Feasibility :** Operational feasibility is operational scopes are determining usability of product, Determining suggested solution by software development team
- **Economic Feasibility –**
  In Economic Feasibility study cost and benefit of the project is analyzed. Means under this feasibility study a detail analysis is carried out what will be cost of the project for development which includes all required cost for final development like hardware and software resource required, design and development cost and operational cost.

- **Legal Feasibility –**
  In Legal Feasibility study project is analyzed in legality point of view. This includes analyzing barriers of legal implementation of project, data protection acts or social media laws, project certificate, license, copyright etc

- **Schedule Feasibility –**
  In Schedule Feasibility Study mainly timelines/deadlines is analyzed for proposed project which includes how many times teams will take to complete final project which has a great impact on the organization as purpose of project may fail if it can’t be completed on time.

Cost-Benefit Analysis :

- Cost-benefit analysis compares the expected financial gain derived from a particular set of actions with the expected cost of providing each action to determine the most profitable option. The projected benefits of a plan or program are divided by its estimated total long-term cost.
- **Costs**
- Direct costs
- Indirect costs
- Intangible costs
- Opportunity costs
- Costs of potential risks

- **Benefits**
- Direct
- Indirect
- Total benefits
- Net benefits

- The determination of costs and benefit entails following steps :

1. Identify the costs and benefits pertaining to given project.
2. Categorize the various costs and benefits for analysis.
3. Select a method of evaluation.
4. Interpret the results of the analysis.
5. Take action.

1. **Hardware cost –**
   Hardware cost includes actual purchase and peripherals (external devices) that are connected to computer. For example, printer, disk drive etc. Actually, finding actual cost of hardware is generally more difficult especially, when system is shared by various users so as to compared to a system which dedicated stand alone . In some case, best way is to treat it as operating cost.
2. **Personnel costs –**
   Personnel costs includes EDP staff salaries and benefits as well as pay for those who are involved in process of development of system. Cost occurred during development of system which are one time costs and are also called development cost. Once system is installed, cost of operating and maintaining system becomes recurring cost that one has to pay very frequently based on requirement.

3.Facility cost –
Facility cost is amount of money that is spent in preparation of a site that is physical where application or computer will be in operation. This includes wiring, flooring, lighting and air conditioning. These costs are treated as one- time costs and are included into overall cost estimate of candidate system.

4.Operating costs –
These includes all costs associated with day-to-day(everyday) operation of system and amount depends on number of shifts, nature of applications. There are various ways of covering operating costs. One approach is to treat operating costs as an overhead. Another approach is to charge money from each authorized user for amount of processing they require from system. Amount charged is based on computer time or time they spend on system, staff time ad volume of output produced .

5.Supply costs –
Supply cost are variable costs that increase with increased use of paper, disks and like. They should be estimated and included in overall cost of system.

- **Requirements modeling** is the process of identifying the requirements this software solution must meet in order to be successful. Requirements modeling contains several sub-stages, typically: scenario-based modeling, flow-oriented modeling, data modeling, class-based modeling, and behavioral modeling.
- While technically there is no right way to go through the stages of requirements modeling, it typically begins with scenario-based modeling.

Scenario–Based Modelling

- For the building of analysis and design models, it is essential for software engineers to understand how end users and other actors want to interact with the system
- Analysis Modeling with UML begins with the creation of scenarios in the form of use-cases, activity diagrams and swim-lane diagrams
- Use cases can be represented as a text narrative or as a ordered sequence of user actions or by using a template
- Example, consider the use case Access Camera Surveillance - Display Camera Views (ACS - DCV) of the Safe Home Security System

- **Scenario-based Model :**
  Using a scenario-based approach, system is described from user’s point of view. **For example**, basic use cases and their corresponding use-case diagrams evolve into more elaborate template-based use cases.

Class-based model :

- The section present a series of informal guidelines as how do we go about developing the class-based elements of an analysis model:
- - classes and objects
- - attributes
- - operations
- - packages
- - CRC models, and
- - collaboration diagrams.

Class-based model :

- Class-based modeling begins by identifying the classes in the use case.
- A collection of things that have similar attributes and common behaviors i.e., objects are categorized into classes. In addition to class diagrams, other analysis modeling elements depict manner in which classes collaborate with one another and relationships and interactions between classes.

Behavioral elements :

- Behavioral Modeling
- The behavioral model indicates how software will respond to external events or stimuli. To create the model, the analyst must perform the following steps:
- • Evaluate all use-cases to fully understand the sequence of interaction within the system.
- • Identify events that drive the interaction sequence and
- understand how these events relate to specific objects.
- . Create a sequence for each use-case.
- . Build a state diagram for the system.
- Review the behavioral model to verify accuracy and consistency.

Behavioral elements :

Effect of behavior of computer-based system can be seen on design that is chosen and implementation approach that is applied. Modeling elements that depict behavior must be provided by requirements model.