**UNIT 4.5**

**SOFTWARE ENGINEERING**

**TUTOR NAME: LORAINE**

**SUBMITTED BY FATIMA EJAZ BARRI**

Contents

[**UNIT 4.5** 1](#_Toc60247012)

[**SOFTWARE ENGINEERING** 1](#_Toc60247013)

[Contents 2](#_Toc60247014)

[TASK1:- 4](#_Toc60247015)

[Software engineering 4](#_Toc60247016)

[1.1Software engineering principles 4](#_Toc60247017)

[Rigor and formality: 5](#_Toc60247018)

[Separation of concerns: 5](#_Toc60247019)

[Modularity and decomposition: 5](#_Toc60247020)

[Abstraction: 5](#_Toc60247021)

[Anticipation of change: 6](#_Toc60247022)

[Generality: 6](#_Toc60247023)

[Incremental Development: 6](#_Toc60247024)

[The Software Development Life Cycle (SDLC) 6](#_Toc60247025)

[Phase 1: Planning 6](#_Toc60247026)

[Phase 2: Analysis 7](#_Toc60247027)

[Phase 3: Design 7](#_Toc60247028)

[Phase 4: Development 7](#_Toc60247029)

[Phase 5: Testing 7](#_Toc60247030)

[Phase 6: Deployment 7](#_Toc60247031)

[Phase 7: Maintenance 7](#_Toc60247032)

[1.2 Software engineering methods and technique 8](#_Toc60247033)

[Diagram of waterfall model 8](#_Toc60247034)

[Phases of the waterfall model 8](#_Toc60247035)

[TASK 2 14](#_Toc60247036)

[**2.1) Software engineering practices** 14](#_Toc60247037)

[2.2) multidisciplinary nature of software development. 14](#_Toc60247038)

[2.3) the team structure of software engineering 16](#_Toc60247039)

[**Entity relationship diagram** 16](#_Toc60247040)

[Two types of DFDs are: 23](#_Toc60247041)

[1- 26](#_Toc60247042)

[Task 3: 27](#_Toc60247043)

[3.1) apply a software engineering approach to software development for information management. 27](#_Toc60247044)

[3.2) Use software engineering methods in system development 27](#_Toc60247045)

## TASK1:-

Software engineering

The research is called software engineering, which helps one to design, manage and improve a software system.

Software engineering is the process by which end-user software is developed, constructed, and tested using programming languages that will fulfil user needs. As compared to basic programming, computer technology is used to build larger and more sophisticated software systems.

**NEEDS OF SOFTWARE ENGNEERING**

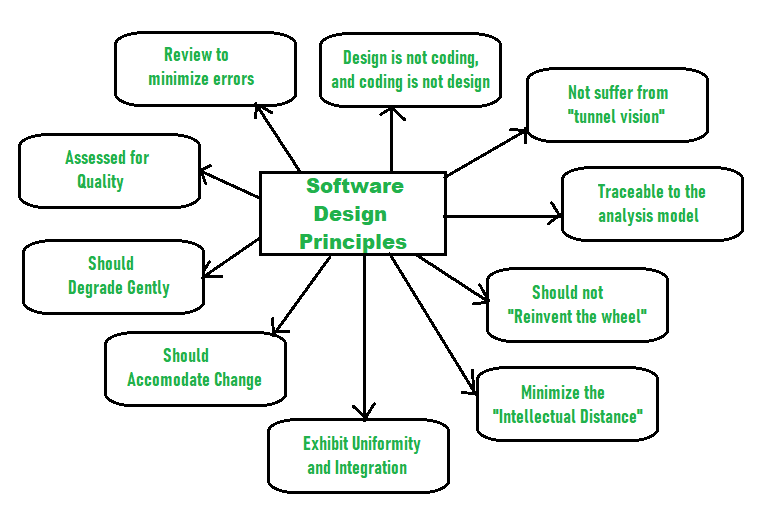
1. Since software development is costly, appropriate steps are required to make efficient and effective use of resources.
2. The need for software engineering is another factor in the cost and time considerations.
3. Factors of reliability.

## 1.1 Software engineering principles

Principles are the basic concepts to ensure the making of the best software system. Principle are practiced through methods. The principles I am going to discuss next have helped the production of great software systems and the product depends on these principles to come out perfect. Following these principles precisely ensures that the system will have all the desired qualities expected from a good system. Although the principles control the process but they cannot guarantee the quality of the product so it is important to focus on both quality of the process and the following of the principles of software engineering.

The basic principles of software engineering are:

* Rigor and formality
* Separation of concerns
* Modularity and decomposition
* Abstraction
* Anticipation of change
* Generality
* Incremental Development

F.G- 1.1- Software design principles

Rigor and formality:

Rigor means to follow a certain plan exactly and correctly. Following this principle enables us to develop a system with a certain attitude because we follow the plan made before starting the project. It enables the team to come up with a consistent plan and makes them plan everything thoroughly before rushing in the production process of the system. Rigor helps us to be more certain about the system being in accordance with the client’s expectations.

Formality, on the other hand, means to support rigor with mathematical and analytical methods. It means to support your plan with mathematical work so that you can be very sure that your plan will work out perfectly.

Rigor and formality help us to be sure about our product and make the process be less stressful because working out the process on paper makes the team be more certain about what they are working with and how it will work out. This also helps us to be more confident about our system.

Separation of concerns:

This principle states that a problem or concern should be divided in many small concerns which are making the problem be so big so that the team can divide and work on the small problems contributing to the big one which will eventually lead to the solution. This process can save a lot of time and help the process be less stressful because working out the problem as a whole can be very confusing and wasteful.

Modularity and decomposition:

This principle works like separation of concerns. It calls for breaking down the process of producing the system into different steps or modules. This helps to understand the product down to its very roots, enabling to pinpoint any problem which could not have been pointed out otherwise.

The components which make up the system are interconnected in one or another so fixing the problem in one part of the system will help the whole system to work in a better and efficient way. It is important that the modules should not be too dependent on each other because then it will make the module be less usable as an individual unit.

Abstraction:

Abstraction helps you to work on the parts of the product that are more crucial and important to the system and ignore the parts that can be ignored. This principle is also like to separation of concerns but is slightly different because the parts or modules are separated according to their function in the product.

Abstraction is important to build a software system because it helps highlight and perfect the main purposes and uses of the system and cover the imperfections of the extra features.

Anticipation of change:

It is important to make your system flexible enough so that it can incorporate future changes and updates easily without having to disturb the perfected parts of the system. This principle tells us to guess what changes might be needed in the system in the near future so that we can keep the possible changes in mind while proceeding and make the system be flexible towards the changes.

The changes can occur due to clients demanding a difference in the functioning of the system or making the system up to date with the current technology. The changes can be major or minor so it is also important to test the system after modifying it to ensure that it works properly.

Generality:

This principles tells us to work on the in a general way in addition to focusing on it individually which means that instead of just focusing on the individual parts of the system, we should focus on the general idea and purpose of the system as a whole. Generality describes the system according to the main features while relating them to the smaller parts.

This principle is also used in solving problems in the system. The problem should be looked at in a general way so that the solution does not only make the individual part of the system better but helps the whole system to work in a better way.

Incremental Development:

This principle encourages us to develop the system in a stepwise process, test the system frequently so any changes or problems can be detected early on, and make it easier to complete the system.

It tells us to send a finished part of the system to the customer to get their opinion and to know if it works according to their expectations and if there is any lack in its functions, it can be fixed before proceeding. This helps to build a connection and communication between the customer and company.

It is important to follow this principle because it helps us to develop the system and perfect it stepwise so that we do not have to fix the problems in the system at the end when it is more complicated. It also helps to perfect the functions of the system and helps the company to understand the requirements and expectations of the customers.

The Software Development Life Cycle (SDLC)

The software development life cycle SDLC is a process used to develop software systems. The SDLC’s main goal is to develop software that meets and exceeds customer expectations.

The SDLC consists of the following steps to meet their main goal, develop a good software system.

Phase 1: Planning

The planning phase consists of determining the main goal of the project and the software system. This phase also consists of planning of how to reach said goals. The two primary activates involved in the planning phase are as follows:

1. Identification of the system for development
2. Feasibility assessment

Phase 2: Analysis

In this phase, the requirements to complete the project and the plan to build the project are revised. The two steps used in this process are:

**PLANNING**

ANALYSIS

**DESIGN**

**DEVELOPMENT**

TEXTING

**DEPLOYMENT**

MAINTENANCE

1. Gathering business requirement
2. Performing a detailed analysis

Phase 3: Design

Here, the overall look of the project is planned and every detail related to the design is discussed. The two steps involved are:

1. Design of IT infrastructure
2. Designing the system model

For a long-lasting system, the base of the project should be a strong one.

Phase 4: Development

In this phase, the planning is put into work and all the discussions are kept in view while building the system. This includes:

1. Development of IT infrastructure
2. Development of database and code

In this phase the previous steps are made practical.

Phase 5: Testing

In testing, the final project is assembled and is gone through some tests to check if there are any problems with the system. This is done to avoid any backlash after the release of the system. It generally includes:

1. Writing test cases
2. Execution of test cases

This phase is very important to build a good system as we can fix any issues with the project before its release. The testers assume what problems might occur and then compare them to the actual issues.

Phase 6: Deployment

This is the part where the system is put out to the world and is open to the public for use.

Phase 7: Maintenance

This phase comes after the release of the project where the feedback from users is taken into account and the system is updated to fit the users’ needs. It depends how intricately the system is built, if it goes through a lot of testing it may not need a lot of tweaks. This phase typically has the following three steps:

1. Helping users
2. Maintaining the system
3. Updating system

1.2 Software engineering methods and technique

#### Waterfall Model with Diagram and All the Phase

The waterfall model is another model used for the step by step development of a software system. For this classical model the term waterfall is used because it evolves in a downward fashion systematically from one step to another.

Diagram of waterfall model

Maintenance

Requirements

System Design

Implementation

Integration and Testing

Deployment of System

FIG-1.2: Waterfall model

Phases of the waterfall model

Requirements:The first step is to plan out what the system is about, why it’s being built and what is required for it to be made. It is important to plan out these specifications as it will help in getting a clear goal for the system.

System design: In this step, all the requirements are looked into and a design is planned for the system. This entails all the steps to be followed in order to get the final desired product.

Implementation:From the system design, prepared individual units are integrated in the next system and the whole system is made.

Integration and testing:in the phases of integration & testing, the company puts in the use after the system has been successfully tested. Testing is done so that the client does not have any problem during the installation of the software. If there are any flaws or errors. Every unit is tested for its functionality. If there are many errors then they will surface at this point in the process.

Deployment of the system:After the setting up of the system and testing, the system is released in the market and is available for customer’s use.

Maintenance:This step is done after the release of the system, this is done to solve any problems faced by the clients after live use of the system and to update it according to modern upgrades so that the system can be used in the future.

#### Advantages of the waterfall model:

* It is a basic and simple model.
* It is easy to apply due to its inflexible structure.
* The steps are independent and are finished before proceeding to the next step.
* Tests are done to ensure correction in every phase.
* It helps to find errors earlier

#### Disadvantages of the waterfall model:

* If something with the design does not work out, many difficulties will be faced in the implementation step.
* There are many uncertainties and risks associated with the steps.
* It is not the best model to follow for complicated projects.
* It does not help in long and continuously updated projects.
* It is not a suitable project for projects which require updates after short periods.

#### Incremental model

An Incremental model is a process of software development where requirements divided into multiple standalone modules of the software development cycle.

#### Incremental lifecycle model in software testingDiagram of the incremental model:

FIG-1.3: INCREMENTAL MODEL

tryqa.com, 2019

#### Phases of the incremental model

#### Requirements analysis:

The criteria are defined by the product development experience in the first process of the incremental model. Moreover, the design analysis team understands the technical requirements of the program. This phase plays an important role designing the program under the incremental model.

##### Design & development:

In this step of the SDLC, incremental model the design of the interface of the system and the process of implementation are successfully completed. The incremental model uses design and development process as software develops new practicality.

##### Testing:

##### The testing stage tests the output of each existing function in the incremental model as well as new functionality. The different methods are used in the testing stage to test the actions of each assignment.

##### Implementation:

##### The coding process of the system under development is allowed by an incremental model. This stage involves the final coding of the device designed during the design and development process and the features evaluated during the test phase.

##### Advantages of the incremental model:

* Less costly
* Easy to debug
* More flexible
* The Very low initial cost

##### Disadvantages of the incremental model:

* Needs for good planning
* It is pricey.

##### When to use the incremental model:

* It is used when the requirements of the system are highlighted and understood.
* It is also used when there is a strict time on which the software is to be released.

##### Spiral model

Spiral model is the combination of iterative model and waterfall model. Each step has a design to be implemented and after the implementation, the step is finished by showing the product to the customer. The development team for spiral-SDLC model goes over the set of requirements in one step and goes over the steps needed in the fulfilling of these requirements. The teams adds onto the system to fulfil the ever-increasing requirements until the applications are ready for production phase.

##### Diagram of the spiral model:

##### spiral model

FIG-1.4: SPIRAL MODEL

guru99.com, 2019

##### Phases of the spiral model

Planning

In this step, the whole plan of the process to reach the goals is laid out. Every aspect is considered and all the complications are sorted out. The clients’ expectations and requirements are put into perspective.

##### Risk analysis

This step includes the software analyst to work out any possible complications in the process and how to solve these problems.

Engineering

This step includes the implementation of the plan and includes building of the software system which is then tested for any errors and is released to the public.

##### Evaluation:

In this phase, the customers critic the software system and share their reviews about it, according to which the next phase is planned.

##### Advantages of the spiral model

* It is suitable for high quality and objective oriented projects.
* The system is produced more quickly and efficiently.

##### Disadvantages of the spiral model

* Spiral may continue indefinitely.
* It has a possibility of errors and risks.

##### When to use the spiral model

* It is used where the project needs regular evaluation for possible risks.
* When changes are required at any time.
* When the requirements are not clear.

##### Prototyping model

Prototype methodology is characterized as a model for software development in which a prototype is developed, tested and then reworked as required until an effective prototype is achieved. It also establishes a basis for the final system to be produced. The model of software prototyping works best in situations where the function of the project is not defined. It is an iterative, trial and error process between the developer and the customer.

##### Phases of the prototyping model:

##### Requirements:

A prototyping model starts with the analysis of requirements. During this phase, the system requirements are defined in detail. During the process, system users are approached to know which their system requirements are.

##### Quick design:

The second phase is a simple design or a prototype design. A simple system design is created at the stage that is not a complete design however. This helps the user with a quick idea of the system. The quick design helps the prototype to develop built.

Build prototype:

A prototype is designed in this phase based on the information gathered from quick design.

##### Initial user evaluation:

The proposed system would be presented to the client for an initial evaluation at this stage. That is helps to find out the working models strength and weakness. The customer gathers suggestions and sends it to the developer.

##### Refining prototype:

If the user is not pleased with, the current prototype should be refined according to the input and suggestions of the user.

##### Implement product and maintain:

It is thoroughly tested and deployed for production once the final system is developed based on the final prototype. To reduce downtime and prevent large-scale failure, the system undergoes routine maintenance.

Diagram of the prototype model:

ANALYSIS

QUICK DESIGN

BUILD PROTOTYPE

USER EVALUATION

FIG-1.5: PROTOYPE MODEL

##### Advantages of the prototyping model

* The errors can be detected.
* The user’s opinion is understood completely which helps in the proper building of the system.
* Features that are complicated to integrate in the system are identified and better understood.

##### Disadvantages of the prototyping model

* It leads to disrupting the process of building the actual system.
* Customers can get confused between the prototype and the actual system.

##### When to use the prototyping model

Prototypes are made to develop a system, which consists of human and system interaction because it helps to understand the customer’s expectations. It is not easy to develop a system, which can be easily used. They make designing a user interface system easier and better.

##### Modelling tools

Modelling tools are testing tools that are in accordance to the structure and features of the model. They help in the testing of the software system model and helps validate it so that there are no errors and problems found in the end product. The errors can be difficult to omit as you can fix the error in one part of the project and find out that there is another part with conflicting information about the error. Modelling tools are used by developers to perfect the design of the software system.

## TASK 2

## **2.1) Software engineering practices**

##### Practice 1: Develop Iteratively

Develop iteratively is a procedure that is used to convey the man goal of the system in a proper manner so that the requirement is understood. Each iteration is based around going through, classifying, analysing the requirements of the system, designing and testing the software system in accordance to the goals of the system.

##### Practice 2: Manage Requirements

Managing requirements are the transformation of the stakeholder’s requests into the system’s needs and functions. These are then detailed into functional and non-functional requirements. They are divided into design, user documentation and tests. The requirements are very important in the process of testing. You have to test if the features of the system are according to the requirements and this helps in developing a system in accordance with the needs.

##### Practice 3: Use Component Architectures

Architecture is an aspect of design but it is not all of the design of the system. It is just the main design of the system, which does not change any function, or feature of the system. The architecture of a system is very important in the development of the system; the changes made throughout a system’s lifecycle are mainly done by changing the architecture. The most important feature of the design of the system is that it should be able to accommodate any changes made in the future.

##### Practice 4: Model Visually

A model is made of the system to be designed so that its features and components can be better understood. A complicated system is understood by making a visual model of the system to be made. Modelling is important because it helps the organization to understand the features of the system and helps them to understand how to execute the project. It helps in studying the details of the system and helps to change them according to needs.

##### Practice 5: Verify Quality

Quality does not mean to achieve the needs and meeting requirements but it also means to measure the implementation of a process to ensure that the system is of good quality. Sometimes, some parts of the testing is not done or is left to be done at the end. For example, black box testing. This can be dangerous as the errors found at the end would not be able to be fixed at the right time resulting in the compromise of the quality off the system. This means, to ensure the quality of the system, every step should be applied properly without any compromise.

##### Practice 6: Control Changes

Changes are inevitable in the development of a system but if they are not done in a proper and controlled manner, they can result into chaos. Because many people are working on the development of a system that means the introduction of different ideas on the system from each person so, it is important that these chaos should be taken into consideration to ensure the smooth development and working of the system

## 2.2) multidisciplinary nature of software development.

Project management is the discipline of resource organization and management to complete a project successfully, it requires project constraints within specified scope, quality, time and cost. Project management software discipline of planning monitoring and controlling software projects preparation identify the scope, estimate the work involved and create a project schedule project monitoring and control keep the team up-to-date on the progress of the project and address problems.

Executive sponsor:

* Has ultimate authority and responsibility for a project.
* Approves changes in scope
* Provides additional funding for changes to scope

Project sponsor:

* Make business decisions for the project
* Participate in one or more projects on a day-to-day basis
* Make product resources available

Project manager:

* Reports to and receives management from executive sponsor
* Participate in and approve project plan and deliverables
* Manages, evaluate and prioritize project work plans with the intention of remaining on time and on budget
* Manage project resources
* May manage and oversee the following resources senior technical project manager

Project team lead:

* Assigned full or part-time to participate in project team activities
* Responsible for contributing to overall project expectations and individual team deliverables
* Manage project plan activities and contribute to project plan creation in partnership with project manager
* Coordinates project plan documentation, testing, training efforts

Project team member:

* Survivability policy issues to team manager for transfer to relevant policymaking bodies
* This function involves all the different resources needed to implement the project plan

Software development activities

The development of software also includes the following activities:

* Analysis of requirements software
* Implementation
* Testing

Analysis of requirements software:

The software specifications determine the systems functionality. Defines system constraints

Two types of requirements

* Functional requirements
* Non-functional requirements

Software architecture and software design

Software design is a technical description of the implementation of the requirements by the system.

The system architecture describes:

* How to decompose the system into subsystems (modules)
* Responsibilities of each unit
* Interaction between platforms and technologies modules

## 2.3) the team structure of software engineering

The team structure highlights the problems of the organization with regards to individual project teams. There are ways to organize the teams.

These are mainly three formal team structures:

##### Chief programmer team

* Can lead to low morale in the team
* It can lead to decrease in original ideas
* It is efficient in well understood problems.
* It is made when the completion of the project is more important than the team’s excitement about the project, the team’s member individual development, etc.

##### Democratic team

* This kind of team leads to more excitement regarding your job and also results in satisfaction with your work.
* This type of team focuses on no ego between the team members.
* They lack in locating errors in the system.
* The programmer can be too proud.
* They should think of the project as a team work rather than a member’s individual work.
* Due to no authority, the team members can pick up small fights.

##### Mixed control team organization

* Consists of ideas from both democratic and chief programmer teams.
* It consists of a democratic setup.
* It is a suitable type of organization for a large team.
* The tasks are divided among groups.
* It is the mostly used type of organization.

## **Entity relationship diagram**

To put it in simple terms, an Entity Relationship Diagram also known as an ERD is a technique that is commonly used by software engineers to represent data in a way that would make comprehending it easier and hence make the software design and production easier. An ERD is mainly tasked with the role of providing the different entities and showing the relationship between them in a way that anyone can understand.

These are four basics of ERDs.

* Entity
* Attributes
* Relationship
* Cardinality

##### Entity

An entity is a thing indistinguishable from all other objects. It can be merely identifiable as a rectangle in an ERD. All entities have some properties or information that define them.

**Example:**

**CUSTOMER**

**EMPLOYEE**

Table-1.1: entity

##### Entity set

An entity set is a set of same type of entities. They should be interrelating. For example, a customer having a bank account at a bank. The customer, account and bank are entities, which are relating and thus make an entity set.

**WORKS FOR**

**DEPARTMENT**

**EMPLOYEE**

Table-1.2: entity set

##### Attributes

The information or properties, which define an entity are called attributes of that entity. For example, an employee is an entity and their name, age and mobile number can be the attributes.

##### Relationship

A relationship is the connection between two entities. For example, a customer has an account in a bank so the relationship between the entities (customer and the account) is formed.

**CUSTOMER**

**HAS**

**ACCOUNT**

Table-1.3: relationship

##### Relationship set

A relationship set is a set of interrelating relationships. They are of the same type. Relationships can also have some attributes defining them like entities.

ACC\_NUM

ADDRESS

NAME

ACC\_TYPE

ACC\_ID

DOB

HAS

**ACCOUNT**

**CUSTOMER**

Table-1.4: relationship set

When the set of entities having a relationship set are not distinct, it is called a recursive relationship.

E\_ID

E\_NAME

CONTACT

**WORKS FOR**

**EMPLOYEE**

Table-1.5: relationship set

The Binary relationship involves two entity sets.

LOAN\_TYPE

AMOUNT

DOB

LOAN\_NUM

NAME

ADDRESS

**BARROW**

**CUSTOMER**

**LOAN**

Table-1.6: relationship set

Ternary relations involves three entity sets.

**BRANCH**

**E\_NAME**

**CODE**

**ADDRESS**

**E\_ID**

**CONTACT**

**BANK**

WORKS IN

**EMPLOYEE**

Table-1.7: relationship set

##### Make an entity diagram a bank management system

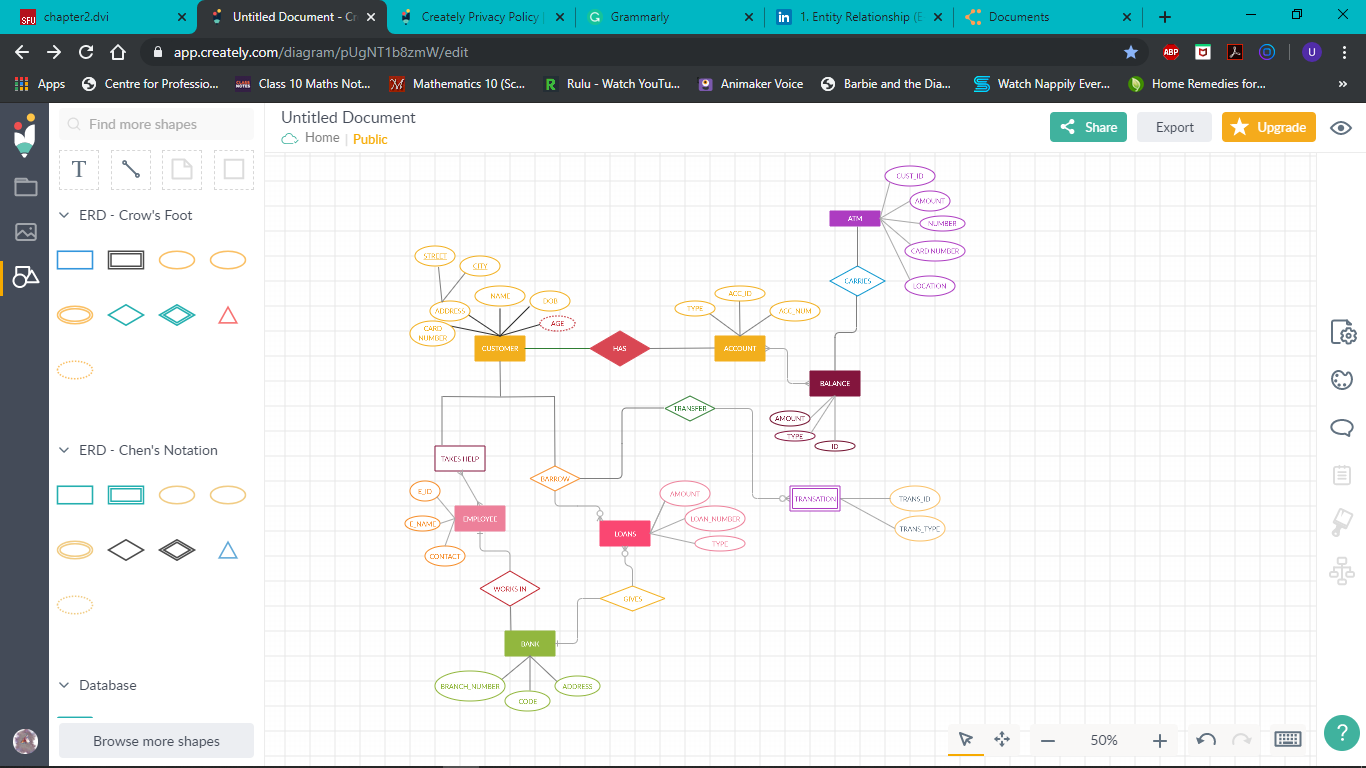


Fig: 1.8: ERD BANK MANAGEMENT SYSTEM

##### Mapping cardinalities

Mapping cardinalities shows how many entities can be associated to another entity.

##### Types of cardinalities:

* Many to many: In this type, more than one entity from a group is associated with more than one entity from another group.
* One to one relationship: In this relationship, two entities only have one relationship connecting them.
* One-to-many: In this type of relationship, one entity is associated with many other entities or many different instances of the same entities to form a relationship.

##### Keys

Different entities can be identified by means of their attributes. Keys are the set of attributes that specialise an entity.

##### Super key

This key includes the set of attributes that identify a specific entity in an entity set.

##### Candidate key

A branch of super key is known as candidate key. Candidate keys are frequent in an entity set i.e. there are many of them.

##### Primary key

One of the candidate keys chosen by the database designer to identify an entity is called a primary key.

##### Unified modelling languages

Unified Modelling Languages (UML) is the common, standard and graphical modelling language of the software system.

These are two types of the unified modelling languages

1. Structured modelling diagram
2. Behavioural modelling diagram

##### Unified modelling languages diagram

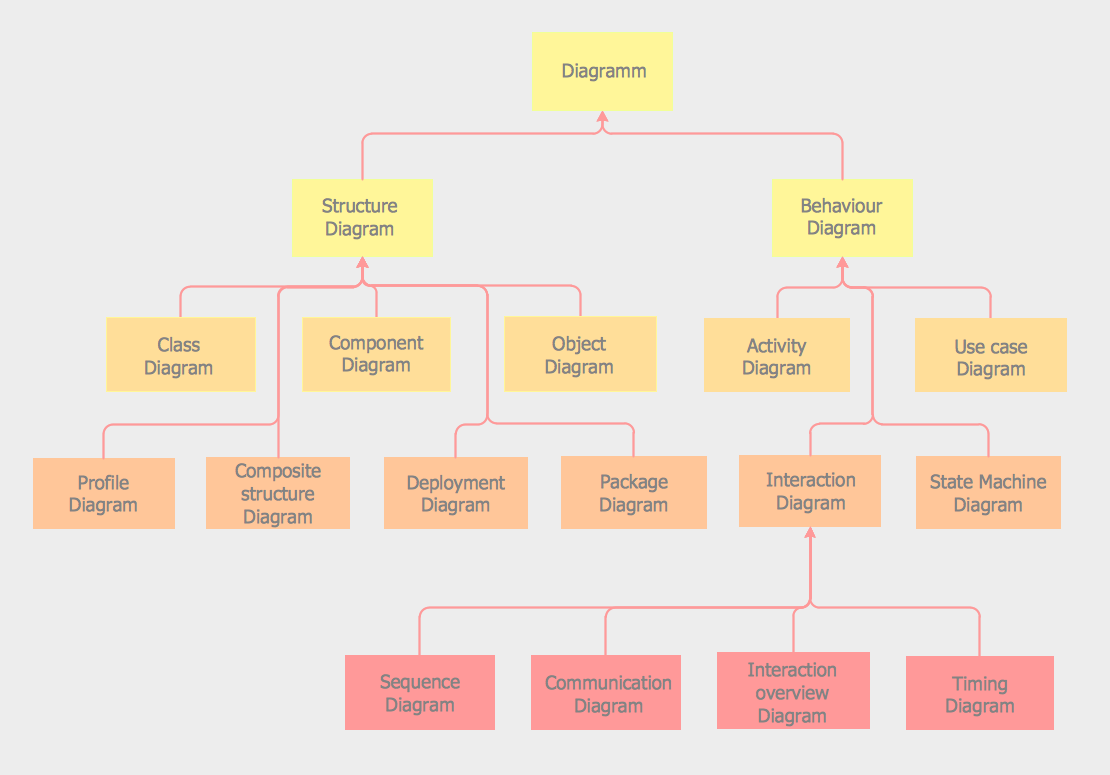


Fig-1.9: UML Diagram

guru99.com, 2019

##### Structure diagram:

A structure diagram is a development tool used to model the different components of a system, from the description of how the individual components interact to construct the whole, to modelling the specifics of the smallest components themselves, such as the different objects and classes used to program the system.

These are two kinds of structure diagrams:

* Data structure diagram:

A conceptual information model used to describe the entities in databases, their relationship and their constraints.

* Composite structure diagram:

A type of static structure diagram showing a class’s internal structure and collaboration with other classes and objects.

##### Class diagram:

A diagram of class is a static diagram. It represents an applications static view. Class diagram is used not only to illustrate, describe and document different aspects of a system but also to construct the software applications executable code. Class diagram describes the class’s attributes and operations as well as the systems constraints. Because they are the only UML diagrams that can be mapped directly with object –oriented languages, class diagrams are commonly used in modelling object-oriented systems. That is shows a set of groups, interfaces, relations and constraints. It is also known as a diagram of structure.

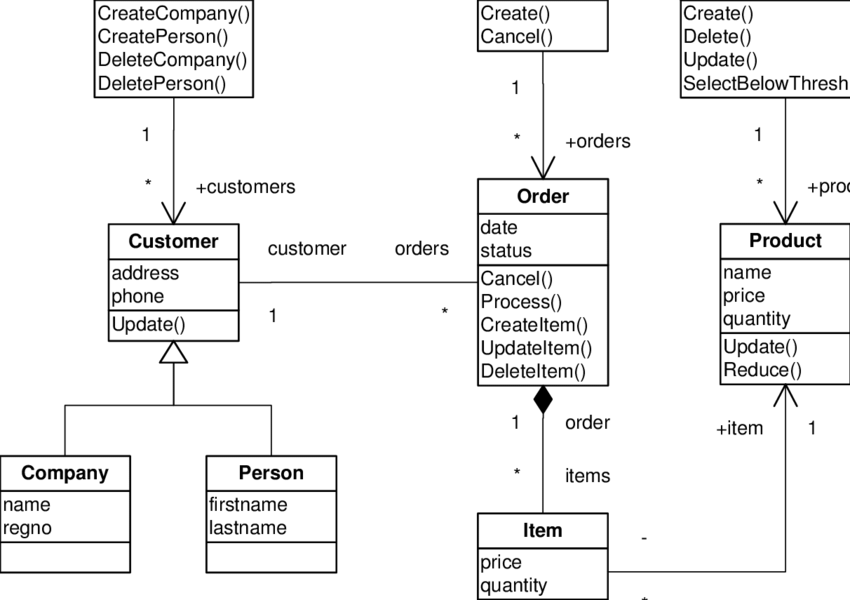


Fig: 2.0 – class diagram

##### Behavioural modelling diagram

UML behavioural diagrams describe the dynamic aspects of a system. The Behavioural diagrams are categorized as follows:

* Use case diagrams
* Interaction diagrams
* State-chart diagrams
* Activity diagrams
* Sequence diagram
* Communication diagram
* Interaction overview diagram
* Timing diagram

##### Use case diagram:

The most important aspect of modelling a system is capturing the complex Behaviour. Dynamic Behaviour means the systems actions when it runs/ operates. Even static Behaviour is not enough to model a dynamic behaviour rather than static Behaviour is more important. Five diagrams are accessible in UML to model the dynamic nature and one of them is the use case diagram. Now that we have to discuss the dynamic nature in the use case diagram, some internal or external variable should be involved to make the interaction.

The common points used in a use case diagram are as follows:

##### Use case:

Use cases are used to describe the functionality of the high level and how the user will use the system. A use case reflects the distinct functionality of the system component, package, or class.

Use case diagram

The use cases diagrams provide a view of the Behaviour of the elements in a system and how to use them in the context.

Comprise:

* Uses cases
* Actors
* Relationships

Use case diagrams are used:

To model the context of a system by enclosing all the activates of system within a rectangle and focusing on the actors outside the system by interacting with it.

To model the requirements of a system from the outside point of view.

##### Example:

Let’s us consider an automated trading house system. We assume the following features of the system.

* The trading house has two types of customer transactions individual customers and corporate customers.
* Once the customer places an order, it is processed by sales department and the bill is given to the customer.
* The system allows the manager to manage customer accounts and respond to any customer queries.

##### Applications of UML

* UML is a very powerful modelling language.
* It has many applications like it is used in embedding systems, web applications, etc.
* The UML tools can also generate some program language code from UML.
* It can be used to design the whole system ignoring the platform language.

##### Advantages of UML

* It can be used to design and process any type of application.
* It is very useful in developing complicated projects.

##### Disadvantages of UML

* The sequence diagram cannot represent all conditions so the state diagram should only represent basic facts not every outcome.
* UML costs a lot of money and it takes experience to perfect it.

##### Data flow diagram

A technique used to express system requirements in graphical form is the data flow diagram abbreviated as data flow diagram. This led to a modular design, known as a bubble chart too. The aim of DFD would be to clarify requirements and important data transformation. DFD is an initial design process that divides the requirement to the lowest level of detail in a practical way. DFD is a sequence of bubbles, which is joined by lines.

## Two types of DFDs are:

1. Logical data flow diagram:

This type of DFD focuses on the system process and data flow. For example, how data is moved between different entities in a banking software system.

1. Physical data flow diagram

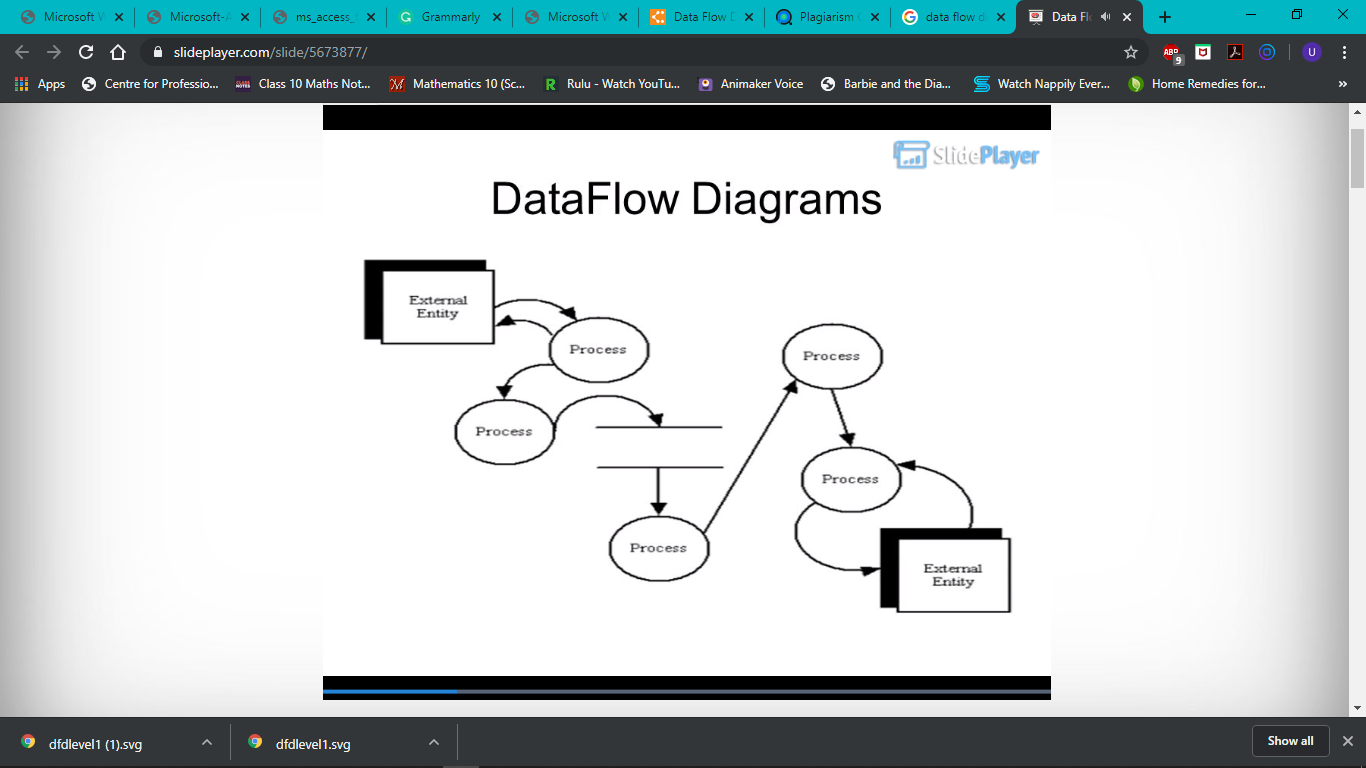
A type of DFD shows the actual implementation of the data flow in the system. It is more specific and close to being implemented.

##### Symbols of data flow diagram

These four basic symbols are used to represent the data flow diagram.

1. Process: This symbol represents the activity that changes the data in any way.
2. Data store: This symbol represents the data that is not moving or flowing through the system.
3. External entity: This symbol represents the source of the data and the place the data is flowing to.
4. Data Flow: This symbol represents the moving or flow of data.

The basic notation used in data flow diagrams are following:



##### Examples of Data flow diagram

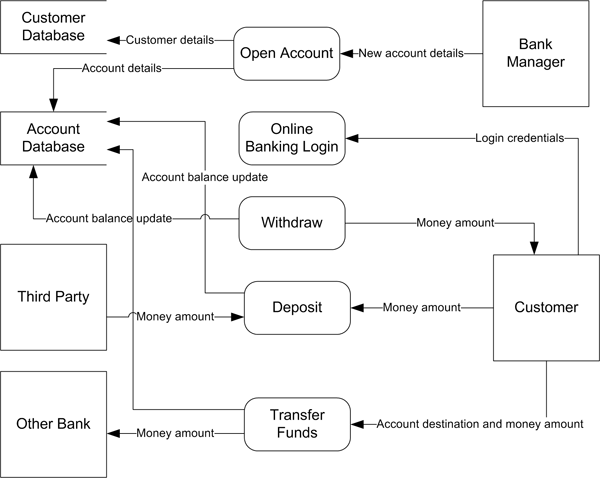


Fig: 2.1

##### Diagram of the data flow diagram

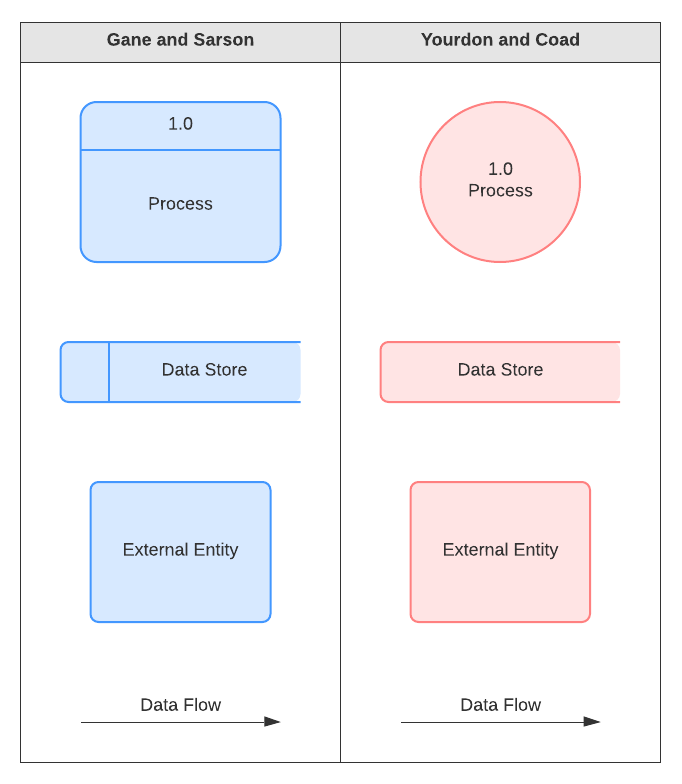


Fig: 2.2

##### Levels of the data flow diagram

The DFD can be used to perform any system. The DFDs are divided into different levels, which represent the increasing data flow. The levels are numbered as 0, 1, and 2 and so on.

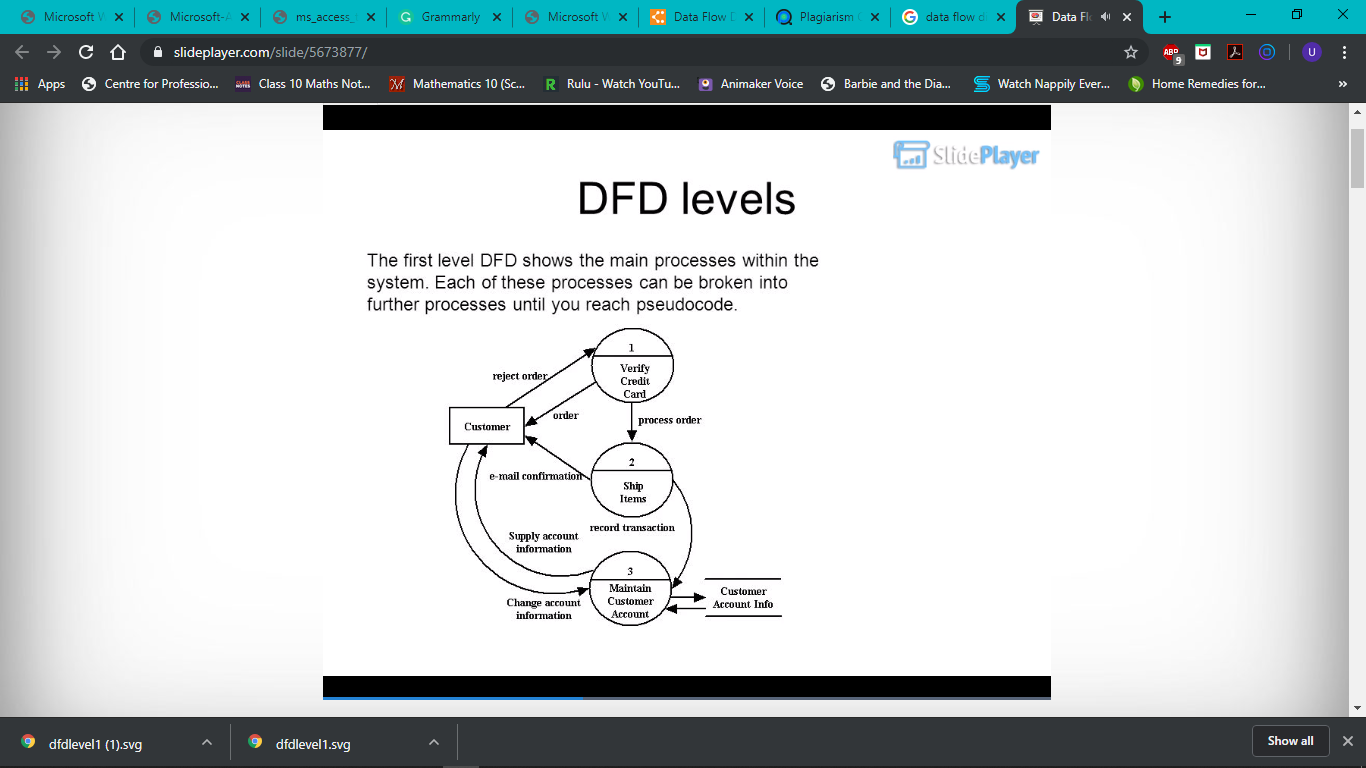


Fig: 2.3

##### 0-level DFD ATM

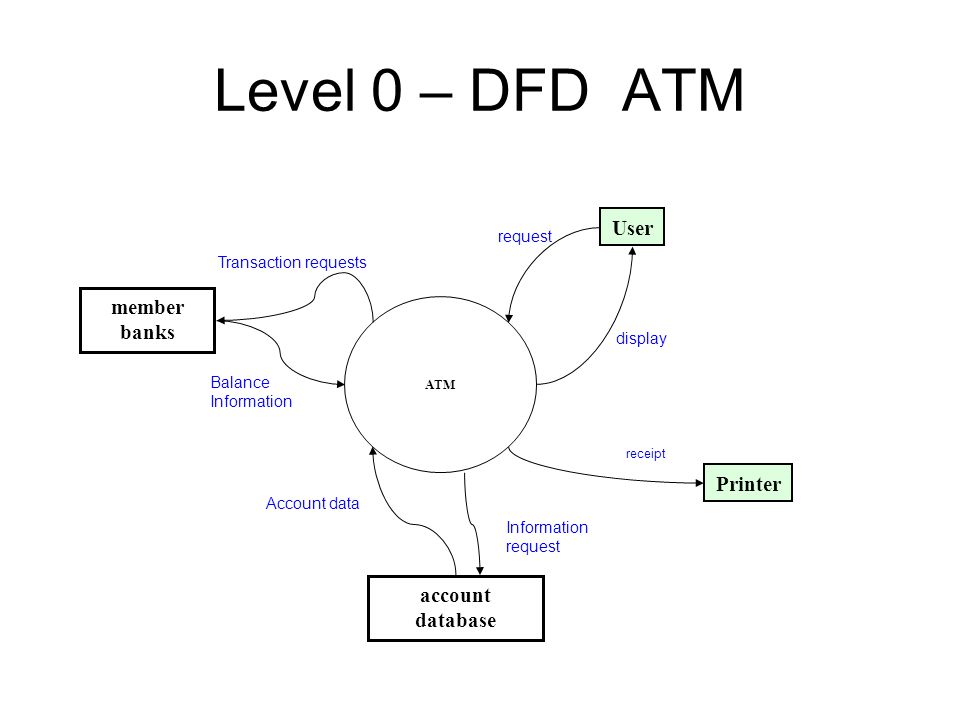
****It is known as the main system model. The full requirement is represented by a single bubble in this DFD and the input and output are represented by arrows with specific directions. Then it is decomposed into smaller bubbles which are represented in a more detailed DFD and it focuses on its part in the system. This decomposing of the system can be done multiple times to understand the system completely. It is also important to represent the number of inputs and outputs in a continuous manner, this is called levelling. For example if a supposed bubble ‘A’ has one input x and two outputs y1 and y2 then the extended DFD will also have one input and two outputs.

Fig: 2.4

## 1-Level DFD

In 1-Level DFD, the DFD is divided into multiple processes or bubbles. This level comprises of the highlighting of main goals of the system and it also includes breakdown of complex processes of 0-Level DFD into sub processes.

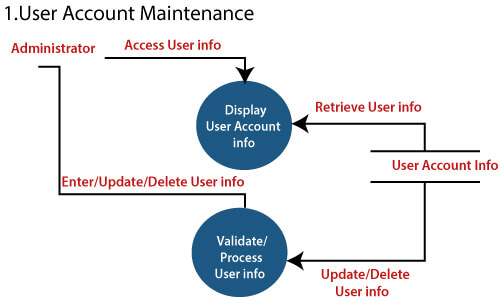


Fig: 2.5

**Software implementation**

During the development phase of the software engineers, build the operating systems that is sometimes referred to as the construction phase.

##### 3.3) explain the role of data Verification and validation in systems development

Verification is the process by which a system is checked thoroughly to ensure that the system is in accordance to the customer’s requirements and expectations.

Validation is the assurance of the customer that the system and its components, functions, design, etc. are how they are supposed to be.

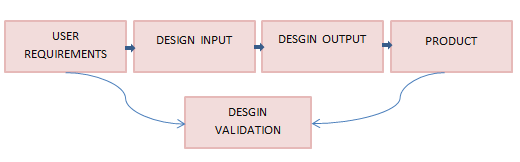


Fig: 2.6 – (guru99.com)

##### Template for test case

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **STEP SR.** | **STEP DESCRIPTION** | **PATH & ACTION** | **TEST DATA** | **EXPECTED RESULTS** | **ACTUAL RESULT** | **PASS/ FAIL** | **COMMENTS** |
| **1** | Insertion of data in account | Add a row at last | Customer ID:  123456789  Account type: Corporate  Account no:  123456789 | New row should be added to br table | New row is added | Pass |  |
| **2** | Deleting a row from Acc.table | Select a row with Account ID  11100 | No data | Row with Account ID should be deleted | Row with Account ID  Is deleted | Pass |  |
| **3** | Update of data in “Account” table | Select a row with Account ID  11100 | Account\_num: 123456789  Customer ID:  123456789 | the data in the row with Account ID  11100 is should got updated | the data in the row with account ID 11100 is updated | Pass |  |
| **4** | Leaving any column empty which is mandatory to fil | Add a row at last but with incomplete data in any of the mandatory column | No data | An error message should be displayed | An error message is displayed | Pass |  |
| **5** | Inserting data other than specified in validation rule of Account type | Add a row at last | Account no: 123456789  Customer ID:  123456789  Account type  savings | an error message should be displayed | An error message is displayed | Pass |  |
| **6** | Instering data in customer\_ID which is not from the customer table | Add a row at last | Account\_num: 123456789  Customer\_ID 123456789  Account type: current | An error message should be displayed | An error message should be displayed | Pass |  |

|  |  |
| --- | --- |
| Date: 05-01-2020  Version/Release: 1  Objective: Testing Customer Table | Tested By: Fatima Ejaz Barri  Environment: Test Environment  Test ID: |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Step | Description | Path & Action | Test Data | Expected Result | Actual result | Pass/Fail | Comments |
| 1 | Insertion of data in “Customer” table | Add a row at last | Customer ID: 123456789  DOB: 10/18/2018  Customer Name: Amal  Customer Address:  Dubai  Customer Card No: 31415953 | New row should be added | New row is added | Pass |  |
| 2 | Insertion of wrong data type in Customer table | Add a row at last | Customer ID: 321475262  DOB: 5/5/2005  Customer Name: Sana  Customer Address: USA  Customer Card No: 45632178 | error message should be displayed | An error message are displayed | Pass |  |
| 3 | Inserting data in column employee which is not from the customer table | Add a row at last | Customer Id: 789245613  DOB: 12/14/2015  Customer Name: Aisha  Customer address: London  Customer Card No: 65478920 | an error message should be displayed | an error message should be displayed | Pass |  |
| 4 | Inserting data in column Customer card no which is not from the customer table | Add a row at last | Customer Id: 985462123  DOB: 9/19/2019 Customer name: Areej  Customer address: TORONTO  Customer card no: 56478952 | An error message should be displayed | An error message should be displayed | Pass |  |

REFERENCE

Dinesh Thakur. Software engineering. [Online] available at: <http://ecomputernotes.com/software-engineering/what-is-software-engineering-write-basic-objective-and-need-for-software-engineering> [ accessed November 25, 2019].

Ukessays. (2013). Software Engineering principle. [Online] available at: <https://www.ukessays.com/essays/it-research/software-engineering-principles.php>[accessed November 25, 2019].

Mithat konar. (July 25, 2012). Principle of software engineering. [Online] available at: <https://mithatkonar.com/wiki/doku.php/century/principles_of_software_engineering> [accessed November 25, 2019].

Principle of software of engineering. [online] available at: <https://www.d.umn.edu/~gshute/softeng/principles.html> [accessed November 25, 2019].

Lakshay Sharma, (2013). Waterfall model. [Online] available at: <https://www.toolsqa.com/software-testing/waterfall-model/> [accessed November 25, 2019].

Career ride, (2016). When to use the waterfall model. [Online] available at: <https://www.careerride.com/pmp-when-to-use-waterfall-model.aspx> [accessed November 25, 2019].

Tryqa. Incremental model. [Online] available at: <http://tryqa.com/what-is-incremental-model-advantages-disadvantages-and-when-to-use-it/> [accessed November 25, 2019].

Java point. (2011). Phase of the incremental model. [online] available at: <https://www.javatpoint.com/software-engineering-incremental-model> [accessed November 25, 2019].

Tutorials point. SDLC. [Online] available at: <https://www.tutorialspoint.com/sdlc/sdlc_software_prototyping.htm> [accessed November 25, 2019].

Guru. Spiral model. [Online] available at: <https://www.guru99.com/what-is-spiral-model-when-to-use-advantages-disadvantages.html> [accessed November 25, 2019].

Melsatar. (March 25, 2012). Software development life cycle. [Online] available at: <https://melsatar/2012/03/15/software-development-life-cycle-models-and-methodologies/> [accessed November 25, 2019].

Tryqa. Modelling. [Online] available at: <http://tryqa.com/what-is-modelling-tools-in-software-testing/> [accessed November 25, 2019].

Guru. Prototyping model. [Online] available at: <https://www.guru99.com/software-engineering-prototyping-model.html> [accessed November 25, 2019].

Java point. (2011). Entity relationship. [Online] available at: <https://www.javatpoint.com/software-engineering-entity-relationship-diagrams> [accessed November 25, 2019].

IT definitions. Unified modelling languages. [Online] available at: <https://www.defit.org/uml/> [accessed November 25, 2019].

Wikipedia. Unified modelling languages. [Online] available at: <https://en.wikipedia.org/wiki/Unified_Modeling_Language> [accessed November 25, 2019].

Guru. UML relationship. [Online] available at: <https://www.guru99.com/uml-relationships-with-example.html> [accessed November 25, 2019].

Techopedia. (December 29, 2014). Structure diagram. [Online] available at: <https://www.techopedia.com/definition/22358/structure-diagram> [accessed November 25, 2019].

Tutorials point. UML – class diagram. [Online] available at: <https://www.tutorialspoint.com/uml/uml_class_diagram.htm> [accessed November 25, 2019].

Tutorials point. UML – Use Case diagram. [Online] available at: <https://www.tutorialspoint.com/uml/uml_use_case_diagram.htm> [accessed November 25, 2019].

Visual. DFD diagram. [Online] available at: <https://www.visual-paradigm.com/guide/data-flow-diagram/logical-vs-physical-data-flow-diagrams/> [accessed November 25, 2019].

Kenneth A. kozar. (1997). DFD diagram. [online] available at: <https://spot.colorado.edu/~kozar/DFDtechnique.html> [accessed November 25, 2019].

Guru. UML diagram. [Online] available at: <https://www.guru99.com/uml-class-diagram.html> [accessed November 25, 2019].

Java point. (2011). Levels in DFD. [Online] available at: <https://www.javatpoint.com/software-engineering-data-flow-diagrams> [accessed November 25, 2019].

Guru. Verification and validation. [Online] available at: <https://www.guru99.com/design-verification-process.html> [accessed January 13, 2020].