CBCS:2020-21

CSDP234C:Project and Project Related Assignments

1.Introduction

1.1 Motivation
1.2 Problem statement
1.3 Purpose/objective and goals
1.4 Literature /Survey
1.5 Project scope and limitations

2. System analysis

2.1 Existing systems
2.2 Scope and limitations of existing systems
2.3 Project perspective, features
2.4 Stakeholders
2.5 Requirement analysis
2.6 Functional requirements, performance requirements, security requirements etc.

3. System Design

3.1 Design constraints
3.2 System Model: UML diagrams
3.3 Data Model
3.4 User interfaces

4. Implementation details

4.1 Software/hardware specifications

5. Outputs and Reports

6. Testing

6.1 Test Plan, Black Box Testing or Data Validation Test Cases, White Box Testing
or Functional Validation Test cases and results

7. Conclusion and Recommendations Future Scope

8. Bibliography and References

A suggested list of assignments is given below.

1. Project Time management: plan (schedule table), Gantt chart, Roles and responsibilities, data collection, Implementation
2. Simple assignments to evaluate choice of technology
3. Assignments on Ul elements in chosen technology
4. Assignments on User interfaces in the project
5. Assignments on event handling in chosen technology
6. Assignments on Data handling in chosen technology
7. Online and offline connectivity
8. Report generation
9. Deployment considerations
10. Test cases

1.Introduction

Project work is an activity which aims to give students learning experience the chance to synthesize their knowledge from different areas of learning, which is critically and creatively applied to current life situations.

It progresses under the guidance and monitoring of an advisor or a mentor.

The introduction of the Project is not just the beginning of the project, but it gives the outlines of the objectives of the project. It relates to the aims that you had for accomplishing the project and the introduction gives an idea of all the objectives that you wish to achieve through the project work.

In the introduction you have to mention why you are interested in doing the project, and you have to give the whole background i.e the foundation from which the idea behind the project emerged. Giving background information is vital as it tells the long back history behind the context of the project work.

The introduction is a must for all sorts of the project as it acts as a mirror to what is written in the project.

1.1 Motivation

Motivation is the word derived from the word 'motive' which means needs, desires, wants or drives within the individuals. It is the process of stimulating people to actions to accomplish the goals. In the work goal context the psychological factors stimulating the people's behaviour can be - desire for Success.

Unlike most tangible project management functions, motivation is not designated by the project manager to a team member, instead motivation is internal to each team member and derived from a team member's desire to achieve a goal, accomplish a task, or work toward expectations.

What should we write in Project motivation?

It is about to explain what your ultimate goal is and what you want to accomplish in that capacity. Then explaining your role in the project to reach the goal.

Here are four different types of motivation that can compel people to act

Extrinsic Motivation

Extrinsic motivation comes from outside us. We do it because we are impelled to, for example because we are told to by someone who has power over us.

Many employment motivation systems work on the principle of extrinsic reward, because this is effective for simple activities but it is less useful when you want a person to be self-driven.

Intrinsic Motivation

Intrinsic motivation is done for internal reasons, for example to align with values or simply for the hedonistic pleasure of doing something.

In work, people are intrinsically motivated by working for an inspiring leader or in areas where they have a personal interest.

Introjected Motivation

Introjected motivation is similar to intrinsic motivation in that it is internalized. The distinctive aspect of this is that if it is not done, then the person feels the tension of guilt.

Identified Motivation

Identified motivation is where a person knows that something needs doing but has not yet decided to do anything about it.

http://vlabs.iitkgp.ernet.in/se/

1.2 Problem statement

A simple and well-defined problem statement is used by the project team to understand the problem and work toward developing a solution.
Problem statements often have three elements: the **problem** itself, stated clearly and with enough contextual detail to establish why it is **important;** the method of solving the problem, often stated as a claim or a working thesis; the **purpose**, statement of objective and scope.

1.3 Purpose/objective and goals

The first step in all projects of business, home, or education, is to define goals and objectives. This step defines the project's outcome and the steps required to achieve that outcome. Goals and objectives must be clear statements of purpose. Each with its own purpose that drives the end result of the project.

The purpose of determining the work Goals and Objectives is fundamental steps of project management. Setting them allows one to plan, organize and control the sequence, time duration, resources and cost of activities required to ensure project success

Project management objectives are the successful development of the project's procedures of initiation, planning, execution, regulation and closure. Also it is about the guidance of the project team's operations towards achieving the goals within the set scope, time, quality and budget standards.

Goals and objectives are statements that describe what the project will accomplish, or the business value the project will achieve.

Goals are high level statements that provide overall context for what the project is trying to achieve, and should align to business goals.

Objectives are lower level statements that describe the specific, tangible products and deliverables that the project will deliver.

The definition of goals and objectives is more of an art than a science, and it can be difficult to define them and align them correctly.

Goals are high-level statements that provide the overall context for what the project is trying to accomplish.

Because the goal is at a high-level, It may take many projects over a long period of time to achieve the goal.

The goal should reference the business benefit in terms of cost, speed and / or quality.

For instance, an IT infrastructure project to install new web servers may ultimately allow faster client response, better price performance, or other business benefit. If there is no business value to the project, the project should not be started.

Goal generally non-measurable: If you can measure the achievement of your goal, it is probably at too low a level and is probably more of an objective.

If your goal is not achievable through any combination of projects, it is probably written at too high a level.

It is important to understand business and project goal statements, Goals are most important from a business perspective. The project manager needs to understand the business goals that the project is trying to contribute to. However, you do not need to define specific project goals. On the other hand, objectives definitely are important.

Objectives are concrete statements describing what the project is trying to achieve. The objective should be written at a lower level, so that it can be evaluated at the conclusion of a project to see whether it was achieved or not. Goal statements are designed to be vague. Objectives should not be vague. A well-worded objective will be Specific, Measurable, Attainable/Achievable, Realistic and Time-bound (SMART).

Note that the objective is much more concrete and specific than the goal statement.

We must assume that the objective is achievable,realistic and time-bound.

Objectives should refer to the deliverables of the project.

1.4 Literature /Survey

Definition. A literature review is a comprehensive summary of previous research on the similar topic. The literature review surveys scholarly articles, books, and other sources relevant to a particular area of research. The review should enumerate, describe, summarize, objectively evaluate and clarify this previous research.

Steps in the literature review process

Read and evaluate the sources and to determine their suitability to the understanding of the topic. Analyse, interpret and discuss the findings and conclusions of the sources you selected.

How do you write a literature survey?

Literature Review: Conducting & Writing

1.Choose a topic.

2.Decide on the scope of your review.
3. Select the databases you will use to conduct your searches.
4.Conduct your searches and find the literature.
5.Review the literature.

The purpose of a literature review is to:
Provide a foundation of knowledge on the topic. Identify inconsistencies: gaps in research, conflicts in previous studies, open questions left from other research.

A literature review surveys books, scholarly articles, and any other sources relevant to a particular issue, area of research, or theory, and by so doing, provides a description, summary, and critical evaluation of these works in relation to the research problem being investigated.

1.5 Project scope and limitations

What is in the scope and limitations?

Delimitations aim to narrow the scope of a study. For example, the scope may focus on specific variables, specific participants, specific sites, or narrowed to one type of research design (e.g., ethnography or experimental research). Limitations, however, aim to identify potential weaknesses of the study.

The purpose of a scope and limitations statement is to provide an outline of what your project will address and what it won't address.A timeline with major tasks to be accomplished for the project and which team member will take a leadership role and responsibility for the task.

8 Key Steps to Developing a Project Scope Statement

1. Understand why the project was initiated.
2. Define the key objectives of the project.
3. Outline the project statement of work.
4. Identify major deliverables.
5. Select key milestones.
6. Identify major constraints.
7. List **scope** exclusions.
8. Obtain sign-off.

2. System analysis

2.1 Existing systems

Procedure of Analyzing the Existing System

System analyst while analyzing the existing system should:

1. Carry out the analysis of the system at a place where the system is functioning. This step will ensure that the analyst is accepted as one of those operating the system.
2. Note down the key personnel in the system besides the head of the department. The key personnel are those who contribute towards the system operations.
3. Spend some time with the operating personnel and observe the system to understand the finer details of the system.
4. Define the scope of the system and its objective.
5. Collect all the documents which are raised in the system. These documents carry data from one point to another. The documents could be printed or handwritten.
6. Collect separately the outputs such as statements, reports, memos, etc. made in the system to throw more light on the information it generates.
7. Make a list of rules, formulae, guidelines, policies, etc., which are used in running the system.
8. Note down the checkpoints and the controls used in the system to ensure that the dataflow is complete, processing of the data is correct and the analysis is precise.
9. Study the flow of data in the system in units, summary and aggregates from document to document from one stage to the other.
10. Make a small system note as a base document and seek an appointment with each head of the department to discuss the system.
11. Examine whether the achievement of the system's objectives is feasible in the present system.
12. If there are problems in the feasibility of implementation, then examine whether the present system can be modified by introduction of documents, controls, procedures and systems. If this is not possible, redefine the scope of the system and objectives in the light of the study.

13. Draw a revised system flow chart to indicate how the system runs the major steps of processing the information. This chart should include all the modifications which had been suggested and accepted.
14. Discuss the flow chart with the personnel operating the system so that they understand the system.
15. Make a list of the outputs (statements, reports) containing information. Get the contents of the reports approved by the head of the department.
16. Analyze the requirements of the information and reports from the utility point of view.
17. Compare the costs of the old and the new system, and benefits offered.
18. Obtain approval of the new system from the users and the top management.
19. Write a system manual for use of the people in the department for reference to the other users of the system.

2.2 Scope and limitations of existing systems

The purpose of a scope and limitations statement is to provide an outline of your project and timeline with major tasks to be accomplished for the project and which team member will take a leadership role and responsibility for the task.
These project management limitations occur through variations in collected data. The mere disagreement among team members can affect a project's control as well as past project failures. Good managers will develop control limit paths and set standards when issues such as these arise.
Scope identifies all of the things which the project can complete over; limitations identify the boundaries, i.e., those things which are beyond scope.

2.3 Project perspective, features

There are seven characteristics of project:

- A single definable purpose, end-item or result.
- Every project is unique.
- Projects are temporary activities.
- Projects cut across organizational lines.
- Projects involve unfamiliarity.
- The organization usually has something at stake when undertaking a project.

Four Important Project Characteristics

There are several important characteristics of any project that will help to determine where on the SDLC continuum a project is best located. Let's consider the four most important: complexity, reoccurrence, predictability, and plannability.

Complexity

Required domain knowledge, necessary learning and exploration, amount of innovation, and levels of customer and stakeholder involvement are just some of the characteristics that determine the complexity of a project. For example, high required domain knowledge increases complexity because more domain knowledge means more risk of error when transferring that knowledge to a developer – requiring more, and more frequent, feedback, prototyping, innovation, and so on.

Reoccurrence

Reoccurrence is a factor of both how many times a project has been conducted, and the level of variance between projects. Annual or semi-annual projects with little deviation between projects will have a high reoccurrence factor. Conversely, projects which are being conducted for the first time, or which have large variability from one time to the next, will have a low recurrence factor.

Predictability

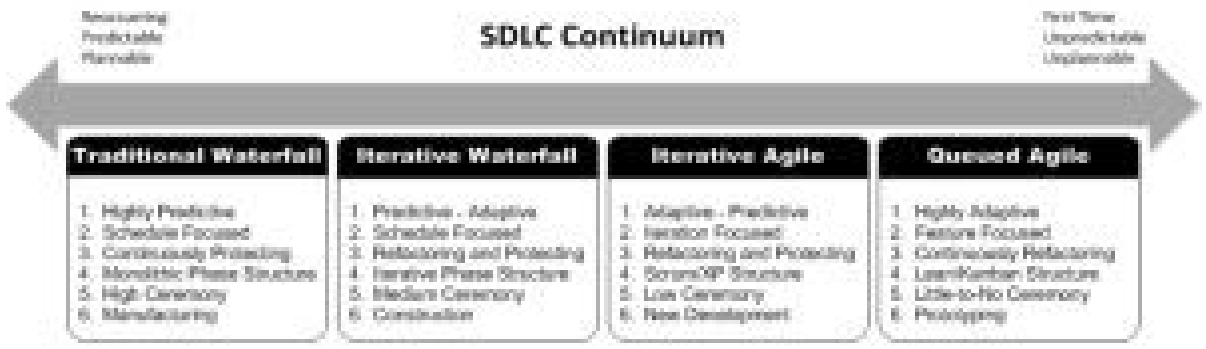
Predictability refers to the statistical probability that a task will occur as predicted. Different project types lend themselves to various levels of predictability, primarily due to the amount of performance information and the task control mechanisms in place that can improve the ability to peer into the future. Thus, a stable task which has been repeated often – such as running a financial report every Monday morning – will have a higher predictability than the first time attempting to run the same report.

Plannability

Plannability refers to the probability that project resources will be available when planned, and that tasks will occur when planned. Projects with insulated teams tend to have higher plannability than projects that require interaction between teams or other agencies. Also, a task may have high predictability, but low plannability. For example, the estimated time to produce the reoccurring financial report discussed above may have a high predictability once all of the data has been acquired, but low plannability if the data has to be acquired from different sources with different priorities for providing the information in a timely manner.

Expanded SDLC Continuum

The high or low state for each of these characteristics will determine where a project falls on the SDLC Continuum, shown below

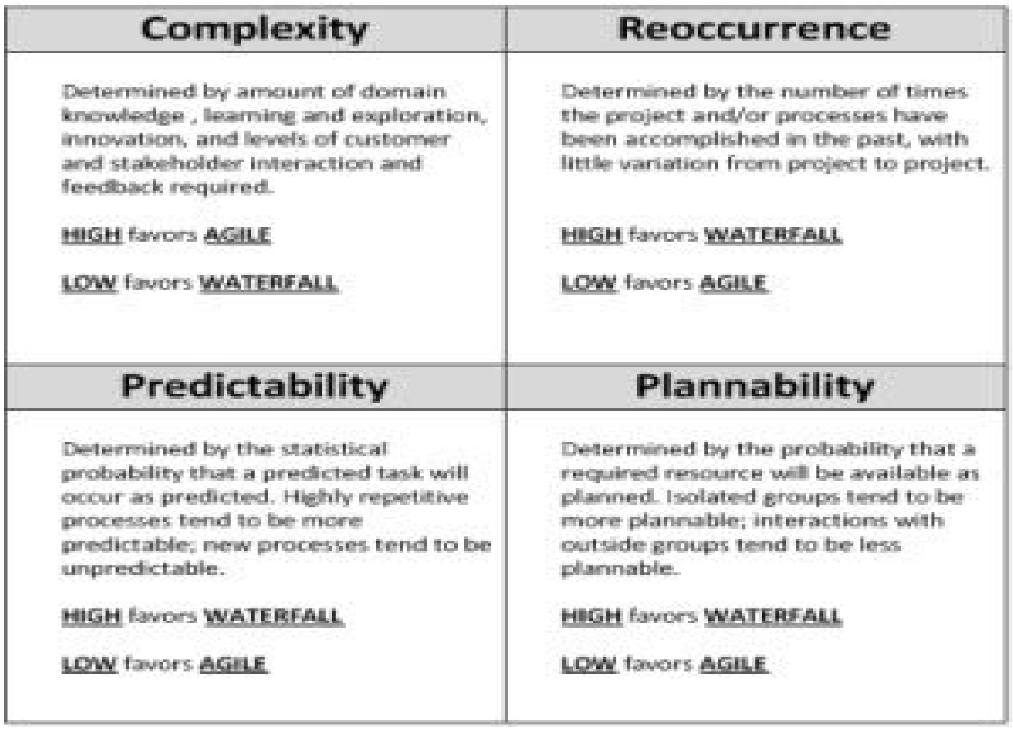


As can be seen, projects that have higher predictability and plannability, and which have a higher reoccurrence, tend to favor the waterfall approach, while those on the low side of these characteristics tend to favor the agile approach. This is primarily due to the fact that projects with high predictability, plannability, and reoccurrence tend that also have long, stable planning windows that justify the upfront planning and requirements gathering that are part and parcel for Waterfall projects. Projects with low predictability, etc., tend to only support short planning windows that require more adaptive control methods.

Complexity is not as straight-forward as the other three characteristics, as there have been successful, complex projects on both sides of the continuum. However, complexity tends to have an inverse relationship with plannability and predictability and, thus, high complexity tends to favor the more adaptive methods. Complex projects with high reoccurrence, however, may offset some of this.

Summary

The below chart summarizes the four project characteristics discussed in this post (click to enlarge):



Complexity	Reoccurrence
Determined by amount of domain knowledge , learning and exploration, innovation, and levels of customer and stakeholder interaction and feedback required. **HIGH** favors **AGILE** **LOW** favors **WATERFALL**	Determined by the number of times the project and/or processes have been accomplished in the past, with little variation from project to project. **HIGH** favors **WATERFALL** **LOW** favors **AGILE**
Predictability	**Plannability**
Determined by the statistical probability that a predicted task will occur as predicted. Highly repetitive processes tend to be more predictable; new processes tend to be unpredictable. **HIGH** favors **WATERFALL** **LOW** favors **AGILE**	Determined by the probability that a required resource will be available as planned. Isolated groups tend to be more plannable; interactions with outside groups tend to be less plannable. **HIGH** favors **WATERFALL** **LOW** favors **AGILE**

Utilizing this chart can help provide some guidance for determining which of the main project management types to propose and utilize for any type of project.

2.4 Stakeholders
Stakeholders are individuals who either care about or have a vested interest in your project. The customer, subcontractors, suppliers, and sometimes even the government are stakeholders. The project manager, project team members, and the managers from other departments in the organization are stakeholders as well.

Four types of stakeholders: users, governance, influencers and providers, which all together go by the acronym UPIG.
Key stakeholders can provide requirements or constraints based on information from their industry that will be important to have when understanding project constraints and risks. The more you engage and involve stakeholders, the more you will reduce and uncover risks on your project.

2.5 Requirement analysis

Requirements analysis is nearly self-explanatory: it's the process of defining the expectations of stakeholders on a project. It analyzes, documents, validates and manages all of the identified requirements—while considering the possibility that there are conflicting requirements among stakeholders.

What is a requirement analysis document?

The results of the requirements elicitation and the analysis activities are documented in the Requirements Analysis Document (RAD). This document completely describes the system in terms of functional and nonfunctional requirements and serves as a contractual basis between the customer and the developer.

Requirements analysis includes three types of activities: Eliciting requirements: (e.g. the project charter or definition), business process documentation, and stakeholder interviews.

2.6 Functional requirements, performance requirements, security requirements etc.

In software engineering, a functional requirement defines a system or its activities.

Non-Functional Testing like Performance, Stress, Usability, Security testing, etc.

Functional requirements may involve calculations, technical details, data manipulation and processing, and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describe all the cases where the system uses the functional requirements, these are captured in use cases.

Functional Vs Non-Functional Requirements: The Comparison Table

The difference between functional and non-functional requirements given below :

Parameters	**Functional Requirement**	**Non-Functional Requirement**

Requirement	It is mandatory	It is non-mandatory
Capturing type	It is captured in use case.	It is captured as a quality attribute.
End-result	Product feature	Product properties
Capturing	Easy to capture	Hard to capture
Objective	Helps you verify the functionality of the software.	Helps you to verify the performance of the software.

Area of focus	Focuses on user requirement	Concentrates on the user's expectation and experience.
Documentation	Describe what the product does	Describes how the product works
Product Info	Product Features	Product Properties

What are security functional requirements?

Functional security requirements describe functional behavior that enforces security. Functional requirements can be directly tested and observed. Requirements related to access control, data integrity, authentication, and wrong password lockouts fall under functional requirements.

What are performance requirements?

Performance requirements define how well the system performs certain functions under specific conditions. Examples are speed of response, throughput, execution time and storage capacity. Like most quality attributes, performance requirements are key elements when designing and testing the product.

3. System Design

3.1 Design constraints

The three primary constraints are time, scope and cost. These are frequently known as the triple constraints or the project management triangle.

The following are common types of design constraint.

- Commercial Constraints. Basic commercial constraints such as time and budget.
- Requirements. Functional requirements such as specifications of features for a website.
- Non-Functional Requirements.
- Compliance.
- Style.
- Sensory Design.
- Usability.
- Principles.

3.2 System Model: UML diagrams

A UML diagram is a diagram based on the UML (Unified Modeling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system.

List of UML Diagram Types

what are the different UML diagram types?

There are two main categories; **structure diagrams** and **behavioral diagrams**.

1. Structure Diagram

    a. Class Diagram

    a. Component Diagram

    b. Deployment Diagram

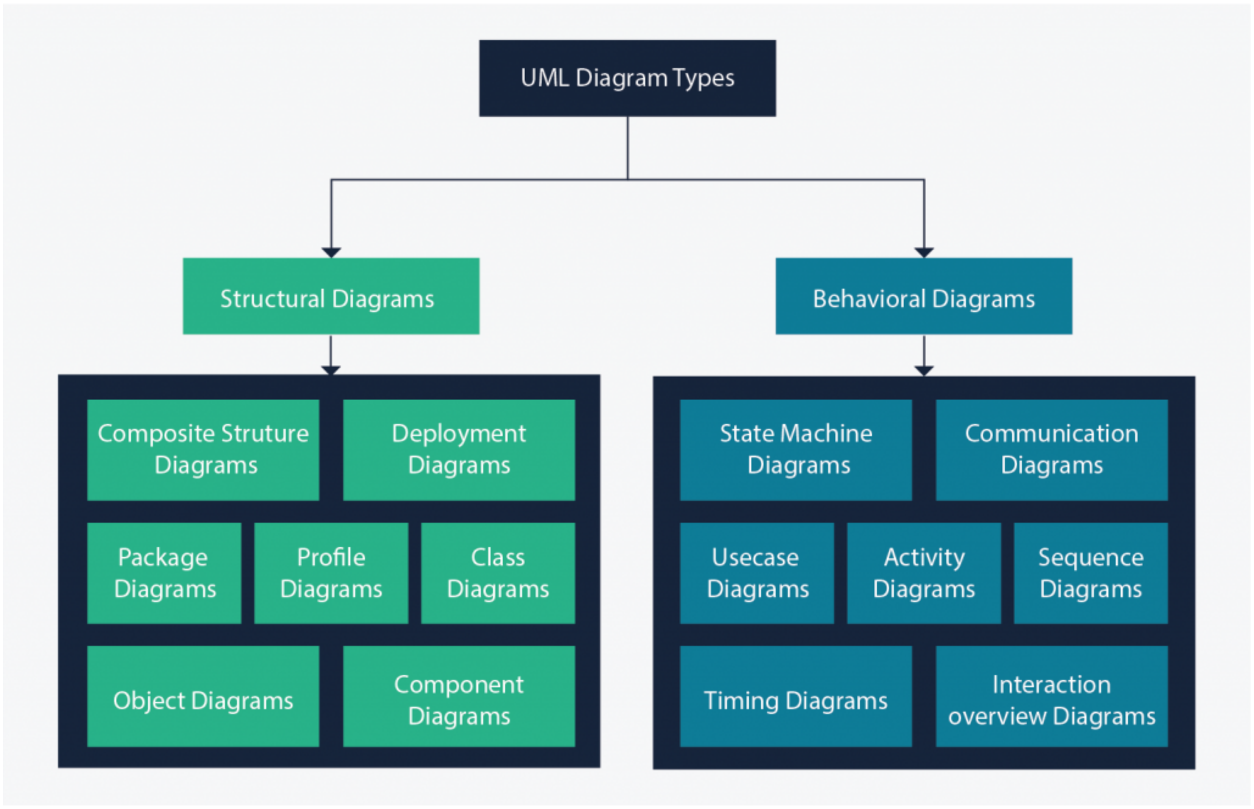
    c. Object Diagram

    d. Package Diagram

    e. Profile Diagram

    f. Composite Structure Diagram

2. Behavioral Diagrams
    a. Use Case Diagram
    b. Activity Diagram
    c. State Machine Diagram
    d. Sequence Diagram
    e. Communication Diagram
    f. Interaction Overview Diagram
    g. Timing Diagram



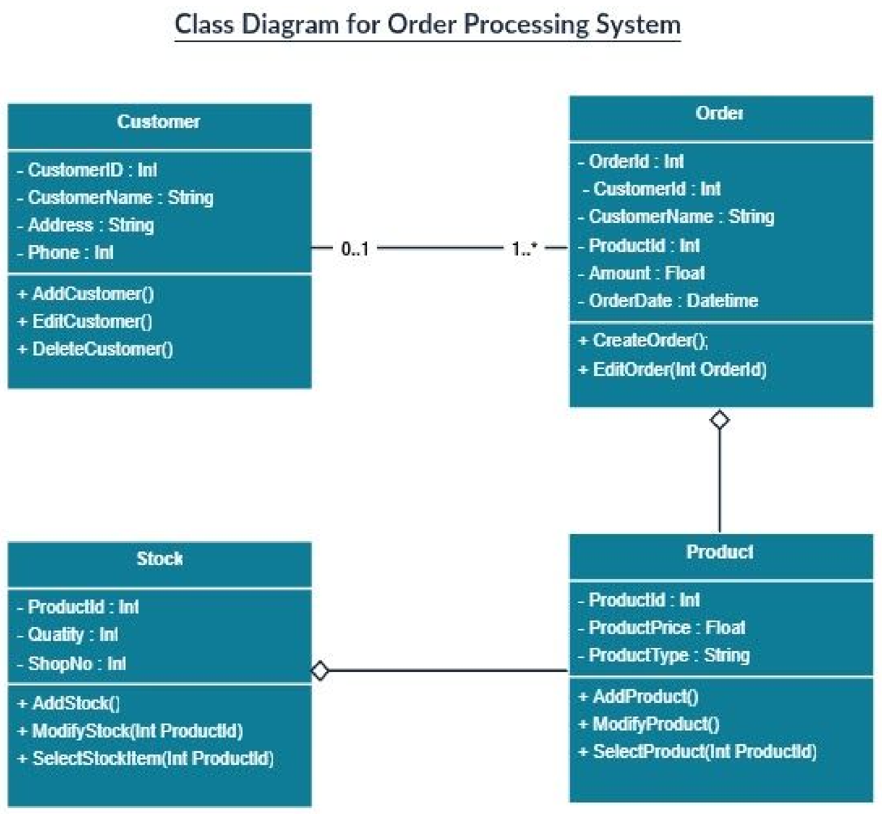
Structure diagrams show the things in the modeled system. In a more technical term, they show different objects in a system. **Behavioral diagrams** show what should happen in a system. They describe how the objects interact with each other to create a functioning system.

Class Diagram

Class diagrams are the main building block of any object-oriented solution. It shows the classes in a system, attributes, and operations of each class and the relationship between each class.

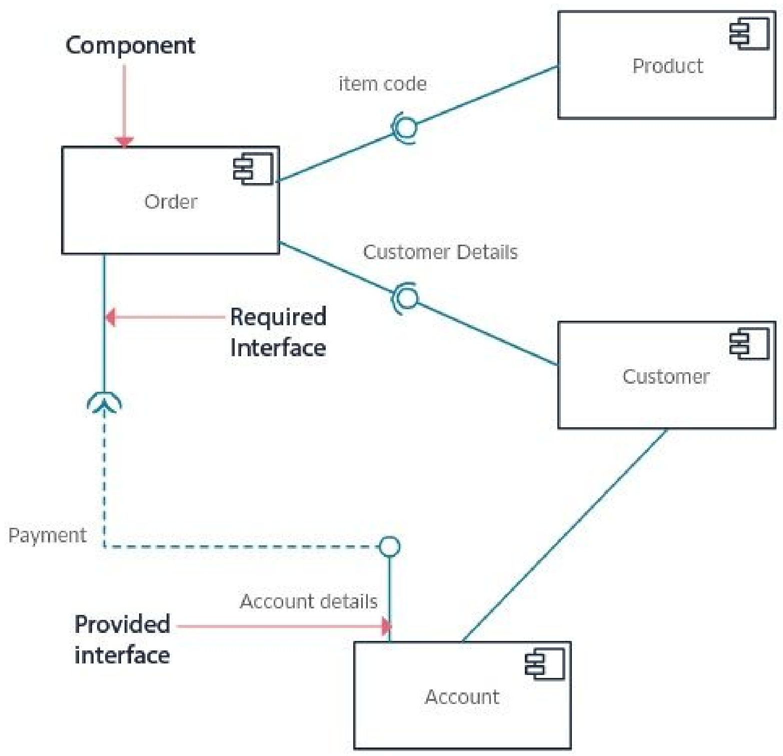
In most modeling tools, a class has three parts. Name at the top, attributes in the middle and operations or methods at the bottom. In a large system with many related classes, classes are grouped together to create class diagrams. Different relationships between classes are shown by different types of arrows.

Below is an image of a class diagram.



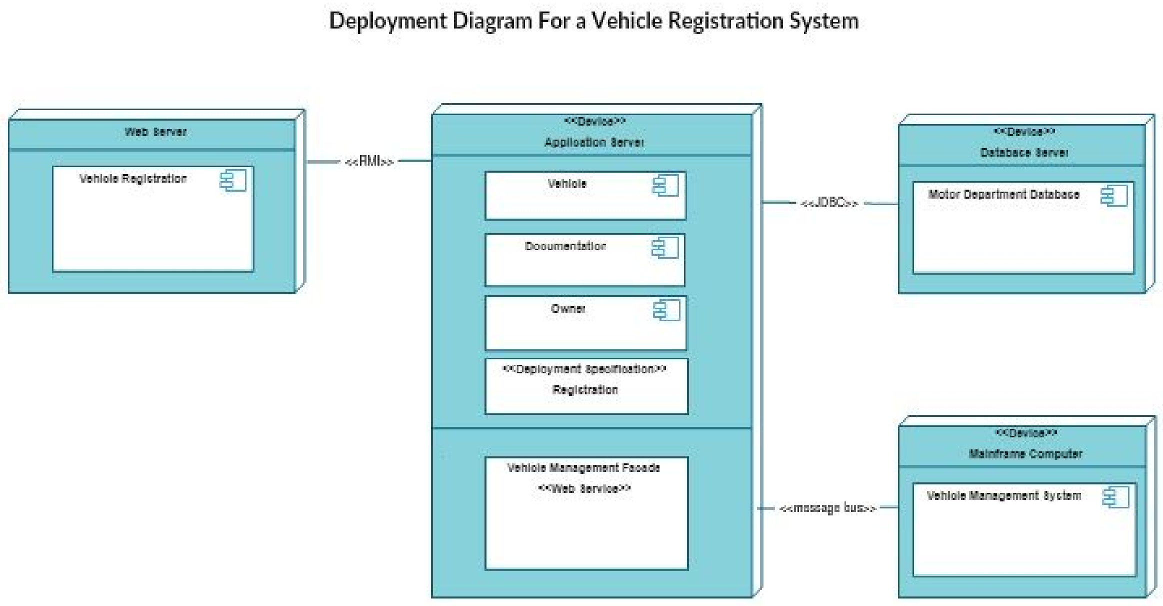
Component Diagram

A component diagram displays the structural relationship of components of a software system. These are mostly used when working with complex systems with many components. Components communicate with each other using interfaces.. The interfaces are linked using connectors. Th\e image below shows a component diagram.



Deployment Diagram

A deployment diagram shows the hardware of your system and the software in that hardware. Deployment diagrams are useful when your software solution is deployed across multiple machines with each having a unique configuration. Below is an example deployment diagram.



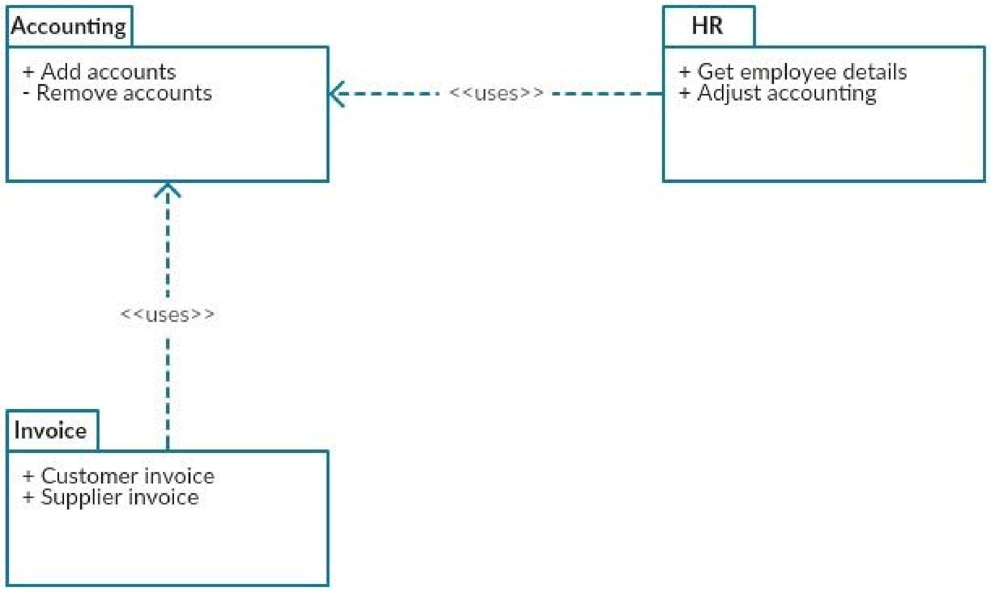
Object Diagram

Object Diagrams, sometimes referred to as Instance diagrams are very similar to class diagrams. Like class diagrams, they also show the relationship between objects but they use real-world examples.

They show how a system will look like at a given time. Because there is data available in the objects, they are used to explain complex relationships between objects.

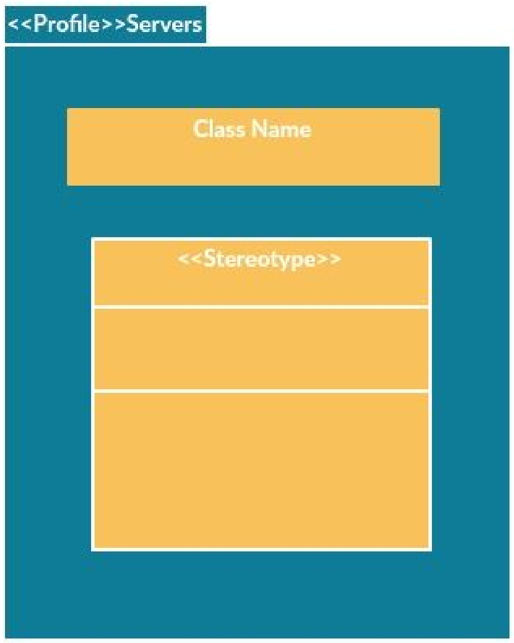
Package Diagram

As the name suggests, a package diagram shows the dependencies between different packages in a system.



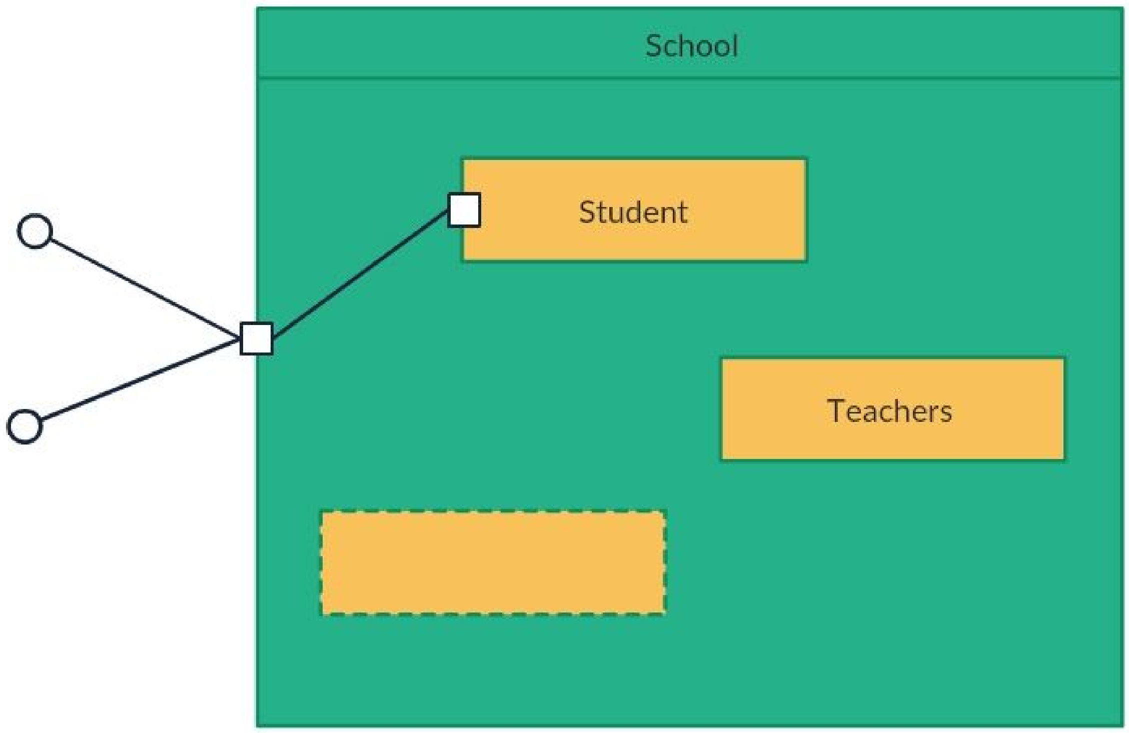
Profile Diagram

Profile diagram is a new diagram type introduced in UML 2. This is a diagram type that is very rarely used in any specification.



Composite Structure Diagram

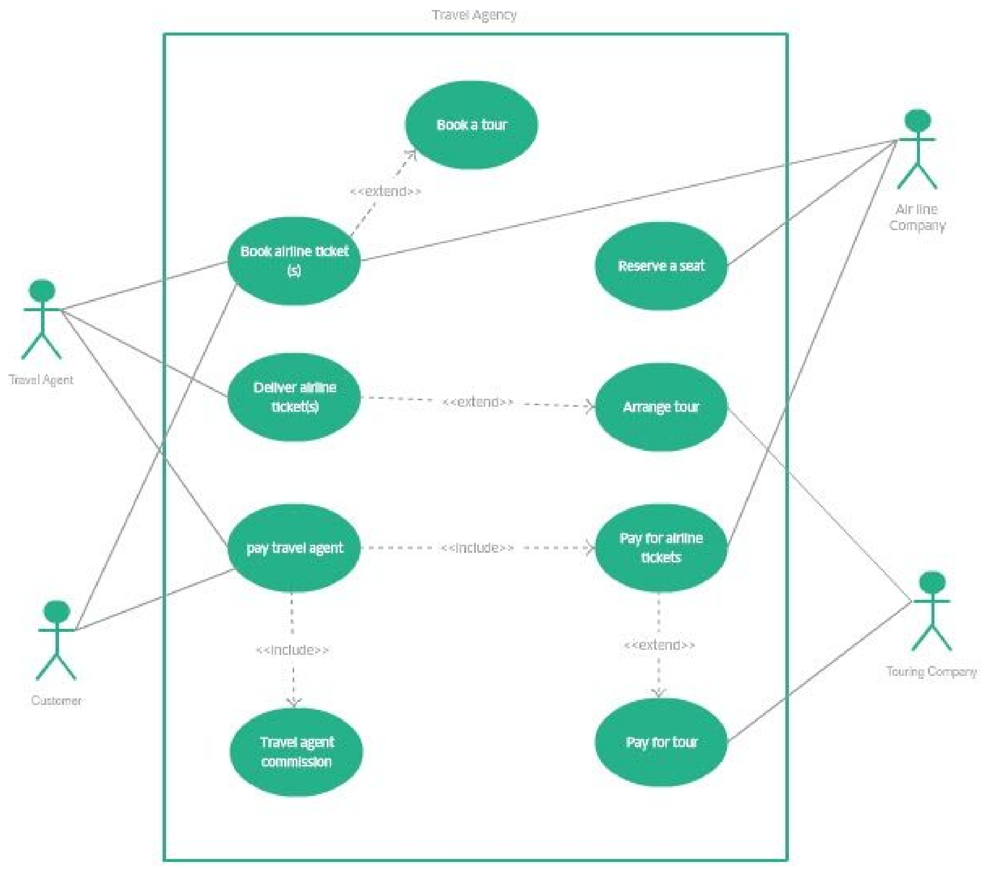
Composite structure diagrams are used to show the internal structure of a class.



Use Case Diagram

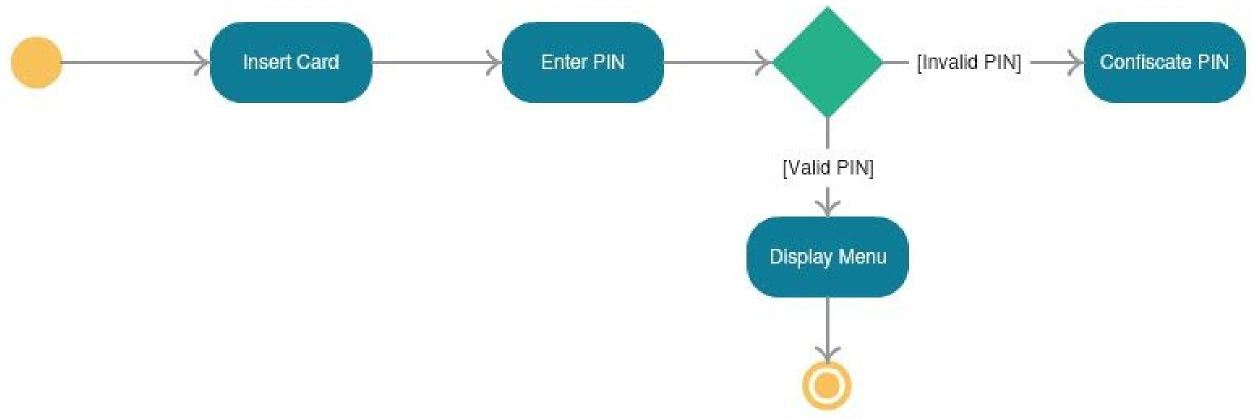
As the most known diagram type of the behavioral UML types, Use case diagrams give a graphic overview of the actors involved in a system, different functions needed by those actors and how these different functions interact.

It’s a great starting point for any project discussion because you can easily identify the main actors involved and the main processes of the system.



Activity Diagram

Activity diagrams represent workflows in a graphical way. They can be used to describe the business workflow or the operational workflow of any component in a system. Sometimes activity diagrams are used as an alternative to State machine diagrams.

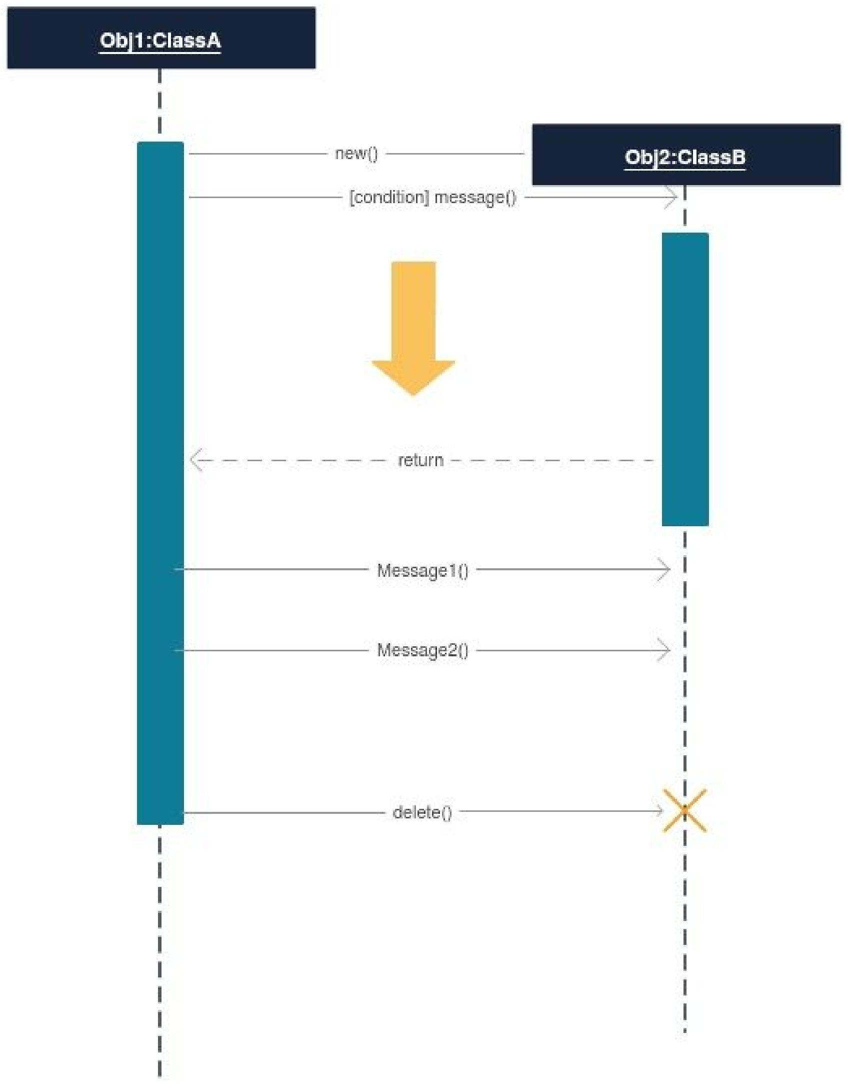


State Machine Diagram

State machine diagrams are similar to activity diagrams, although notations and usage change a bit. They are sometimes known as state diagrams or state chart diagrams as well. These are very useful to describe the behavior of objects that act differently according to the state they are in at the moment. The State machine diagram below shows the basic states and actions.

Sequence Diagram

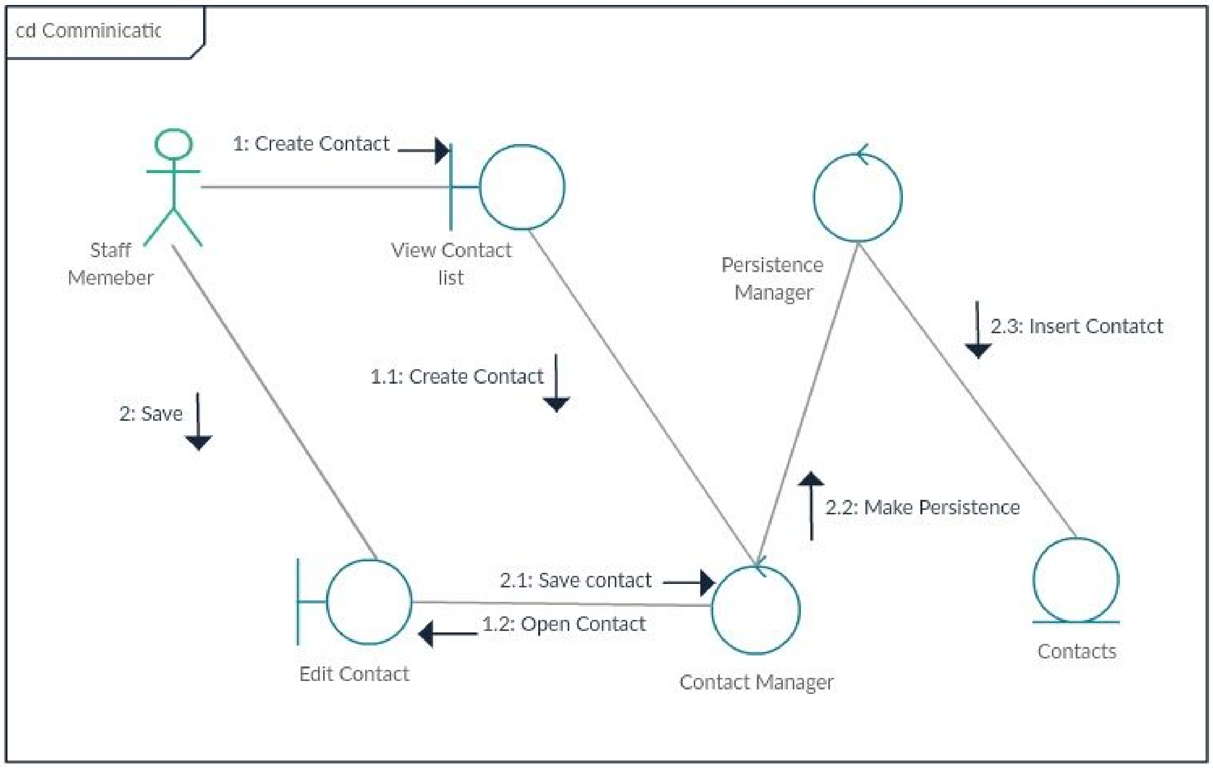
Sequence diagrams in UML show how objects interact with each other and the order those interactions occur. It's important to note that they show the interactions for a particular scenario. The processes are represented vertically and interactions are shown as arrows. This article explains the purpose and the basics of Sequence diagrams.



Sequence diagram drawn using Creately

Communication Diagram

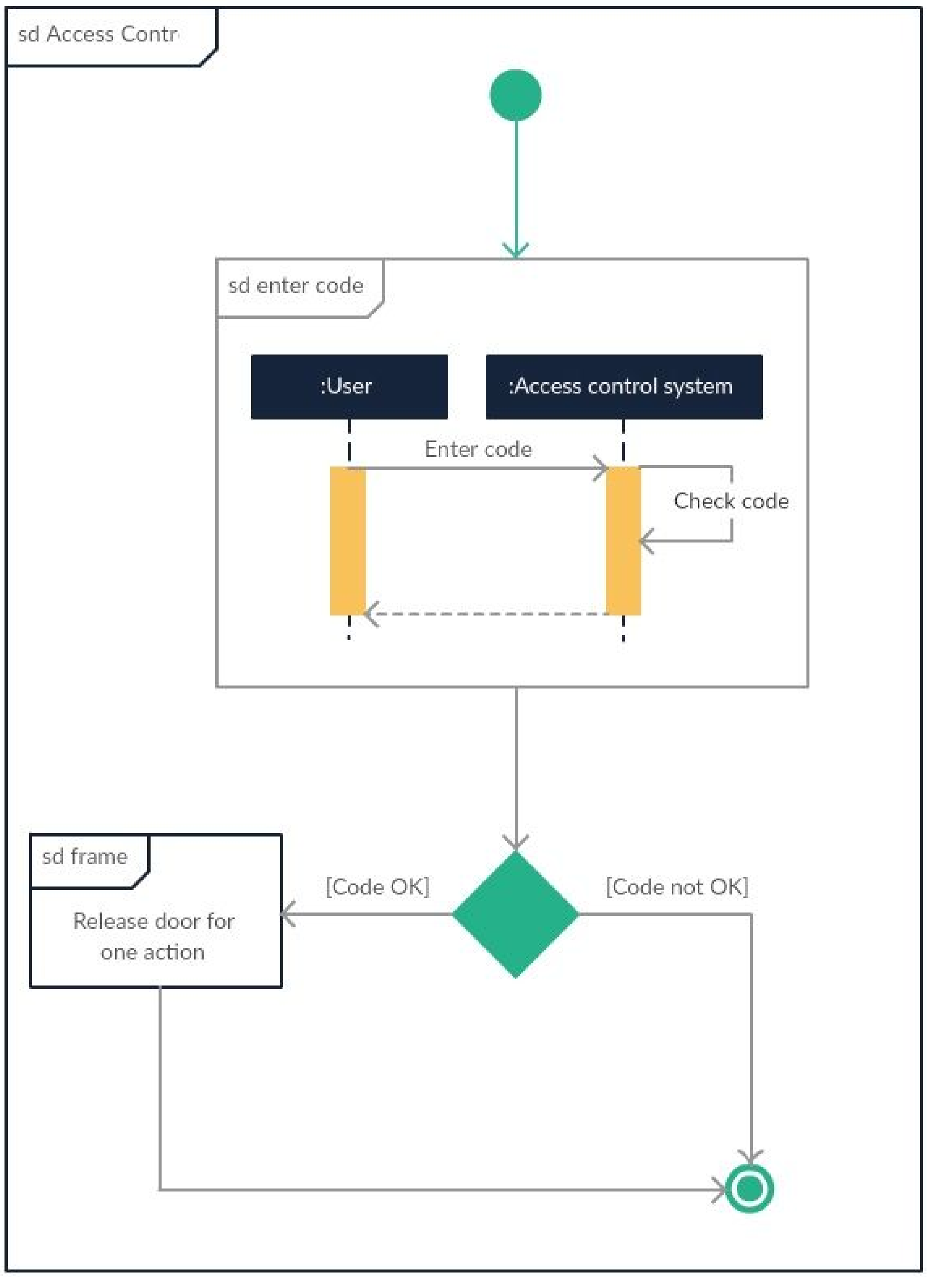
In UML 1 they were called collaboration diagrams. Communication diagrams are similar to sequence diagrams, but the focus is on messages passed between objects. The same information can be represented using a sequence diagram and different objects.



Interaction Overview Diagram

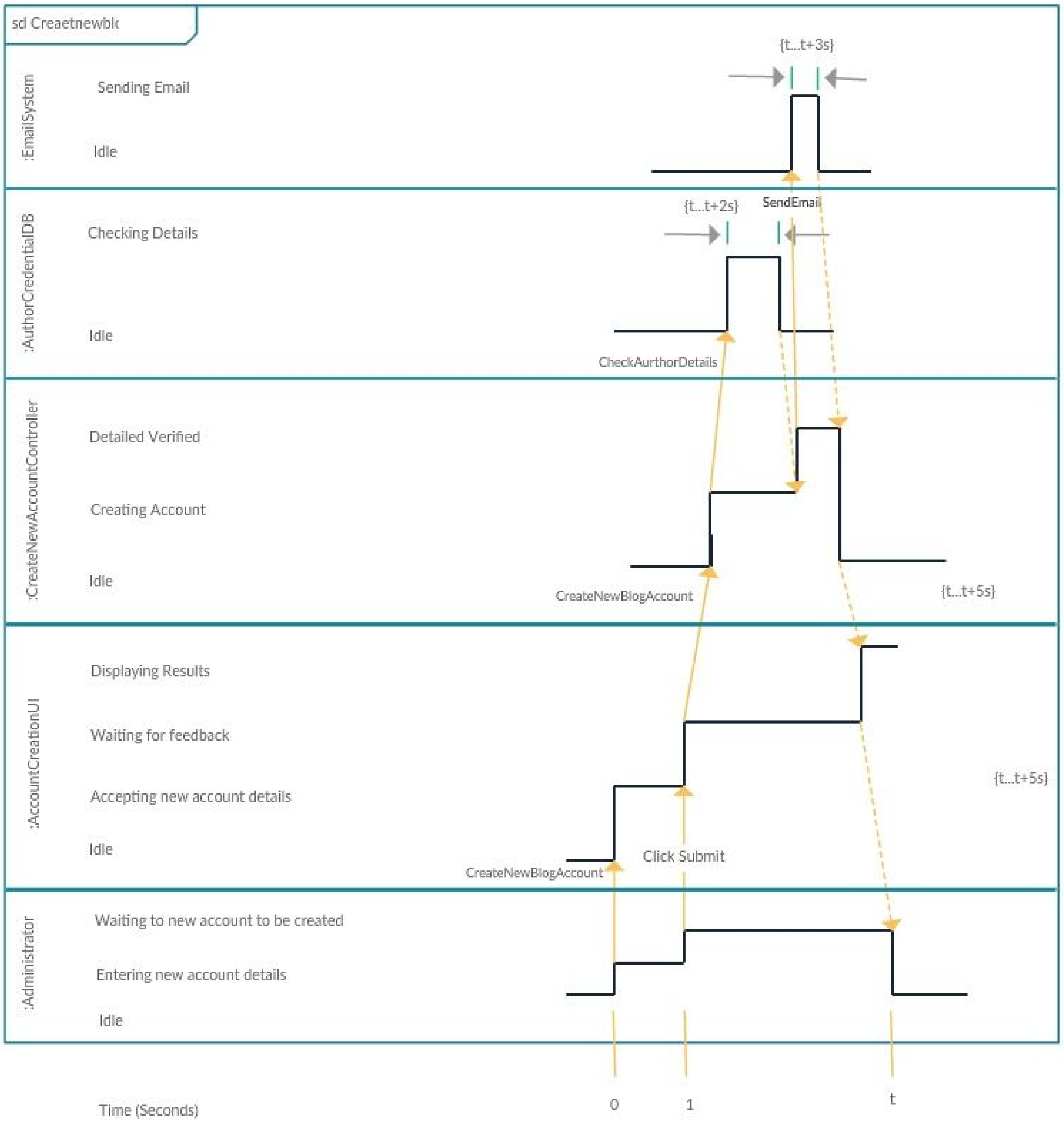
Interaction overview diagrams are very similar to activity diagrams. While activity diagrams show a sequence of processes, Interaction overview diagrams show a sequence of interaction diagrams.

They are a collection of interaction diagrams and the order they happen. As mentioned before, there are seven types of interaction diagrams, so any one of them can be a node in an interaction overview diagram.



Timing Diagram

Timing diagrams are very similar to sequence diagrams. They represent the behavior of objects in a given time frame. If it’s only one object, the diagram is straightforward. But, if there is more than one object is involved, a Timing diagram is used to show interactions between objects during that time frame.



Mentioned above are all the UML diagram types. UML offers many diagram types, and sometimes two diagrams can explain the same thing using different notations.

3.3 Data Model

A data model (or data model) is an abstract model that organizes elements of data and standardizes how they relate to one another and to the properties of real-world entities.

What are the types of data models?

There are three different types of data models: conceptual, logical and physical, and each has a specific purpose.

- Conceptual Data Models: High-level, static business structures and concepts.
- Logical Data Models: Entity types, data attributes and relationships between entities.
- Physical Data Models: The internal schema database design

Data models define how the logical structure of a database is modeled. Data Models are fundamental entities to introduce abstraction in a DBMS. Data models define how data is connected to each other and how they are processed and stored inside the system.

Entity-Relationship Model

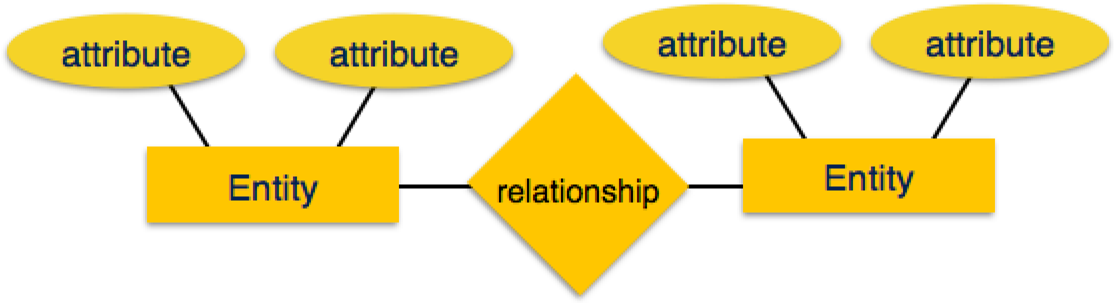
Entity-Relationship (ER) Model is based on the notion of real-world entities and relationships among them. While formulating real-world scenarios into the database model, the ER Model creates entity set, relationship set, general attributes and constraints.

ER Model is best used for the conceptual design of a database.

ER Model is based on −

- Entities and their *attributes*.
- Relationships among entities.

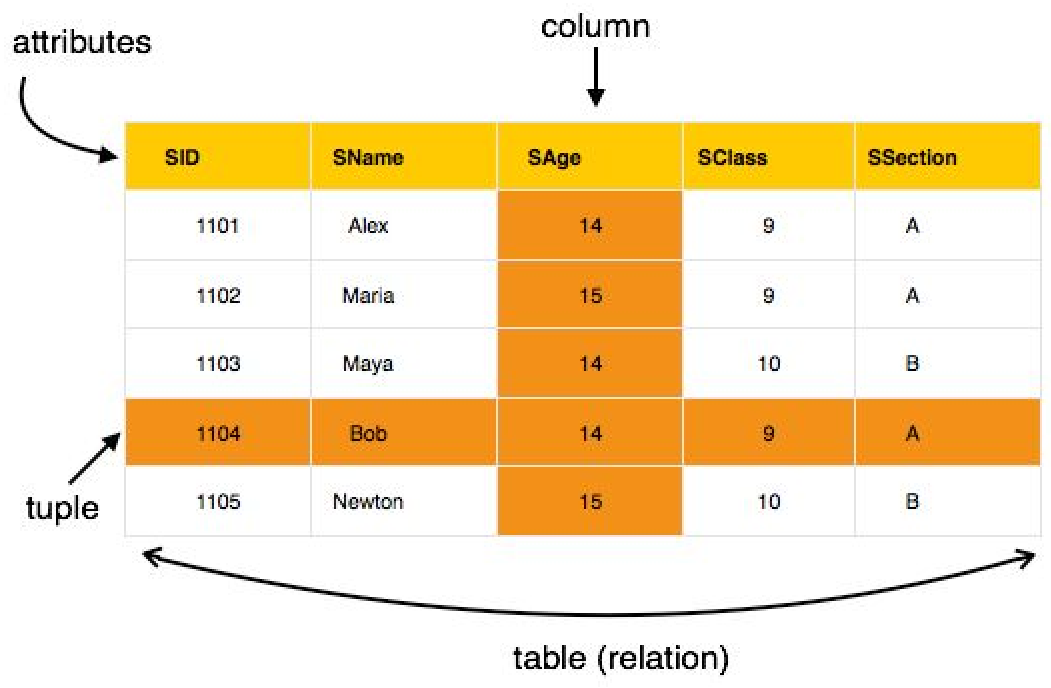
These concepts are explained below.



- Entity − An entity in an ER Model is a real-world entity having properties called attributes. Every attribute is defined by its set of values called domain. For example, in a school database, a student is considered as an entity. Student has various attributes like name, age, class, etc.
- Relationship − The logical association among entities is called *relationship*. Relationships are mapped with entities in various ways. Mapping cardinalities define the number of association between two entities.
  Mapping cardinalities −
  - one to one
  - one to many
  - many to one
  - many to many

Relational Model

The most popular data model in DBMS is the Relational Model. It is more scientific a model than others. This model is based on first-order predicate logic and defines a table as an n-ary relation.



SID	SName	SAge	SClass	SSection
1101	Alex	14	9	A
1102	Maria	15	9	A
1103	Maya	14	10	B
1104	Bob	14	9	A
1105	Newton	15	10	B

The main highlights of this model are −

- Data is stored in tables called relations.

- Relations can be normalized.
- In normalized relations, values saved are atomic values.
- Each row in a relation contains a unique value.
- Each column in a relation contains values from a same domain.

3.4 User interfaces

User interface helps the developer/designer to create the end product in a logical and sequential way. User Interface (UI) design has four main elements: Usability, Visualization, Functionality and Accessibility.
Elements of Interface Design.
Interface elements include but are not limited to:

- Input Controls: checkboxes, radio buttons, dropdown lists, list boxes, buttons, toggles, text fields, date fields..
- Navigational Components: breadcrumb, slider, search field, pagination, slider, tags, icons.

Input Controls

Checkboxes

Checkboxes allow the user to select one or more options from a set. It is usually best to present checkboxes in a vertical list. More than one column is acceptable as well if the list is long enough that it might require scrolling or if the comparison of terms might be necessary.

Radio buttons

Radio buttons are used to allow users to select one item at a time.

Dropdown lists

Dropdown lists allow users to select one item at a time, similarly to radio buttons but are more compact allowing you to save space. Consider adding text to the field, such as ‘Select one’ to help the user recognize the necessary action. Some people consider them rather irritating than useful. But this really depends on a purpose and context.

List boxes

List boxes, like checkboxes, allow users to select multiple items at a time, but are more compact and can support a long list of options if needed.

Buttons

A button indicates an action upon touch and is typically labeled using text, an icon, or both. The **dropdown button** consists of a button that when clicked displays a drop-down list of mutually exclusive items.

Toggles

A toggle button allows the user to change a setting between two states. They are most effective when the on/off states are visually distinct.

Text fields

Text fields allow users to enter text. It can allow either a single line or multiple lines of text.

Date field

A date picker allows users to select a date and/or time. By using the picker, the information is consistently formatted and input into the system. The same picker is used for time.

Navigational Components

Breadcrumb

Breadcrumbs allow users to identify their current location within the system by providing a clickable trail of proceeding pages to navigate by.

Search field

Basically, this is a search box that allows users to enter a keyword or phrase (query) and submit it to search the index with the intention of getting back the most relevant results. Typically search fields are single-line text boxes and are often accompanied by a search button.

Pagination

Pagination divides content up between pages and allows users to skip between pages or go in order through the content.

Slider

A slider, also known as a track bar, allows users to set or adjust a value. When the user changes the value, it does not change the format of the interface or other info on the screen.

Tags

Tags allow users to find content in the same category. Some tagging systems also allow users to apply their own tags to content by entering them into the system.

Icons

An icon is a simplified image serving as an intuitive symbol that is used to help users to navigate the system. Typically, icons are hyperlinked.

Image carousel

Image carousels allow users to browse through a set of items and make a selection of one if they so choose. Typically, the images are hyperlinked. Make sure they do not automatically rotate very fast.

Informational Components

Tooltips

A tooltip allows a user to see hints when they hover over an item indicating the name or purpose of the item.

Icons

Here icons serve the purpose of providing the users with the information about what's hidden behind and what might happen if the user will press an icon.

Progress bar

A progress bar indicates where a user is as they advance through a series of steps in a process. Typically, progress bars are not clickable.

Notifications

A notification is an update message that announces something new for the user to see. Notifications are typically used to indicate items such as the successful completion of a task, or an error or warning message.

Message boxes

A message box is a small window that provides information to users and requires them to take an action before they can move forward.

Modal windows (Pop-up)

A modal window requires users to interact with it in some way before they can return to the system.

Containers

Accordion

An accordion is a vertically stacked list of items that utilize show/ hide functionality. When a label is clicked, it expands the section showing the content within. There can have one or more items showing at a time and may have default states that reveal one or more sections without the user clicking.

4. Implementation details

4.1 Software/hardware specifications

Software requirements specification report describes:

1The intended purpose

2 System Architecture

3 Client-Server Architecture
4 Design and Implementation
5Server details, software installation procedures etc.
For Example:

Hardware Requirements

- Pentium IV or higher, (PIV-300GHz recommended)
- 256 MB RAM
- 1 Gb hard free drive space

Software Requirements

- PHP (front end)
- HTML
- JavaScript
- MS Word 97 or later
- Web Browser: Microsoft Internet Explorer, Mozilla, Google Chrome or later
- MySQL Server (back-end)
- Operating System: Windows XP / Windows7/ Windows Vista

5. Outputs and Reports

Outputs

These are the first level of results associated with a project. Often confused with “activities”, outputs are the direct immediate term results associated with a project. In other words, they are usually what the project has achieved in the short term. An easy way to think about outputs is to quantify the project activities that have a direct link on the project goal.

An output can be a document, a product, a service, or a result. An input to a process can be an output from another process, or things such as documents, company policies, and templates created outside of the project management processes.

What are project reports?

Project Reports are used when you need to communicate the status of a project. You can complete Project Reports on a weekly basis and ensure that they are sent to all of your project stakeholders.

Types of project management reports are as follows:

- Team availability report.
- Status report.
- Project health report.
- Risk assessment.
- Time tracking report.
- Baseline reports.

6. Testing

6.1 Test Plan, Black Box Testing or Data Validation Test Cases, White Box Testing or Functional Validation Test cases and results

Software Testing is evaluation of the software against requirements gathered from users and system specifications. Testing is conducted at the phase level in software development life cycle or at module level in program code. Software testing comprises Validation and Verification.

Software Validation

Validation is the process of examining whether or not the software satisfies the user requirements. It is carried out at the end of the SDLC(System Development Life Cycle). If the software matches requirements for which it was made, it is validated.

- Validation ensures the product under development is as per the user requirements.
- Validation answers the question – "Are we developing the product which fulfills all user needs from this software ?".
- Validation emphasizes on user requirements.

Software Verification

Verification is the process of confirming if the software is meeting the business requirements, and is developed with the proper specifications and methodologies.

- Verification ensures the product being developed is according to design specifications.
- Verification answers the question– "Are we developing this product by our design specifications ?"

- Verifications concentrate on the design and system specifications.

Target of the test are -

- **Errors** - These are actual coding mistakes made by developers. In addition, there is a difference in output of software and desired output is considered as an error.
- **Fault** - When error exists fault occurs. A fault, also known as a bug, is a result of an error which can cause a system to fail.
- **Failure** - failure is said to be the inability of the system to perform the desired task. Failure occurs when fault exists in the system.

Manual Vs Automated Testing

Testing can either be done manually or using an automated testing tool:

- **Manual** - This testing is performed without taking help of automated testing tools. The software tester prepares test cases for different sections and levels of the code, executes the tests and reports the result to the manager.
  Manual testing is time and resource consuming. The tester needs to confirm whether or not right test cases are used. Major portion of testing involves manual testing.
- **Automated-** This testing is a testing procedure done with aid of automated testing tools. The limitations with manual testing can be overcome using automated test tools.

A test needs to check if a webpage can be opened in Internet Explorer. This can be easily done with manual testing. But to check if the web-server can take the load of 1 million users, it is quite impossible to test manually.

There are software and hardware tools which help testers in conducting load testing, stress testing, regression testing.

Testing Approaches

Tests can be conducted based on two approaches –

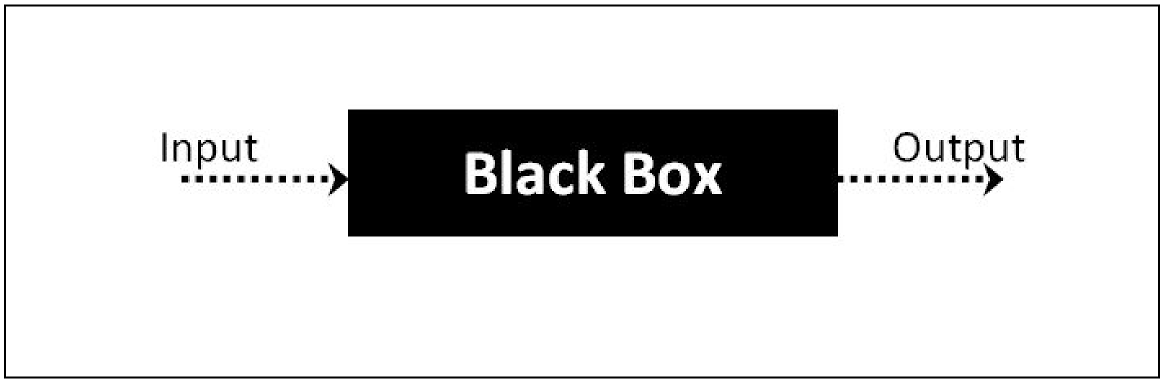
- Functionality testing
- Implementation testing

When functionality is being tested without taking the actual implementation in concern it is known as black-box testing. The other side is known as white-box testing where not only functionality is tested but the way it is implemented is also analyzed.

Exhaustive tests are the best-desired method for perfect testing. Every single possible value in the range of the input and output values is tested. It is not possible to test each and every value in a real world scenario if the range of values is large.

Black-box testing

It is carried out to test functionality of the program. It is also called ‘Behavioral’ testing. The tester in this case, has a set of input values and respective desired results. On providing input, if the output matches with the desired results, the program is tested ‘ok’, and problematic otherwise.



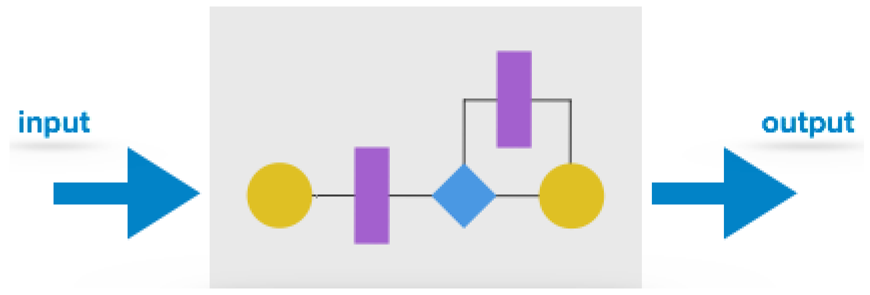
In this testing method, the design and structure of the code are not known to the tester, and testing engineers and end users conduct this test on the software.

Black-box testing techniques:

- Equivalence class - The input is divided into similar classes. If one element of a class passes the test, it is assumed that all the class is passed.
- Boundary values - The input is divided into higher and lower end values. If these values pass the test, it is assumed that all values in between may pass too.
- Cause-effect graphing - In both previous methods, only one input value at a time is tested. Cause (input) – Effect (output) is a testing technique where combinations of input values are tested in a systematic way.
- Pair-wise Testing - The behavior of software depends on multiple parameters. In pairwise testing, the multiple parameters are tested pair-wise for their different values.
- State-based testing - The system changes state on provision of input. These systems are tested based on their states and input.

White-box testing

It is conducted to test programs and its implementation, in order to improve code efficiency or structure. It is also known as ‘Structural’ testing.



In this testing method, the design and structure of the code are known to the tester. Programmers of the code conduct this test on the code.

The below are some White-box testing techniques:

- Control-flow testing - The purpose of the control-flow testing to set up test cases which covers all statements and branch conditions. The branch conditions are tested for both being true and false, so that all statements can be covered.
- Data-flow testing - This testing technique emphasizes to cover all the data variables included in the program. It tests where the variables were declared and defined and where they were used or changed.

Testing Levels

Testing itself may be defined at various levels of SDLC. The testing process runs parallel to software development. Before jumping on the next stage, a stage is tested, validated and verified.

Testing separately is done just to make sure that there are no hidden bugs or issues left in the software. Software is tested on various levels -

Unit Testing

While coding, the programmer performs some tests on that unit of program to know if it is error free. Testing is performed under white-box testing approach. Unit testing helps developers decide that individual units of the program are working as per requirement and are error free.

Integration Testing

Even if the units of software are working fine individually, there is a need to find out if the units if integrated together would also work without errors. For example, argument passing and data updation etc.

System Testing

The software is compiled as a product and then it is tested as a whole. This can be accomplished using one or more of the following tests:

- Functionality testing - Tests all functionalities of the software against the requirement.
- Performance testing - This test proves how efficient the software is. It tests the effectiveness and average time taken by the software to do the desired task. Performance testing is done by means of load testing and stress testing where the software is put under high user and data load under various environment conditions.
- Security & Portability - These tests are done when the software is meant to work on various platforms and accessed by number of persons.

Acceptance Testing

When the software is ready to hand over to the customer it has to go through last phase of testing where it is tested for user-interaction and response. This is important because even if the software matches all user requirements and if the user does not like the way it appears or works, it may be rejected.

- Alpha testing - The team of developers themselves perform alpha testing by using the system as if it is being used in the work environment. They try to find out how users would react to some action in software and how the system should respond to inputs.
- Beta testing - After the software is tested internally, it is handed over to the users to use it under their production environment only for testing purposes. This is not as yet the delivered product. Developers expect that users at this stage will bring minute problems, which were skipped to attend.

Regression Testing

Whenever a software product is updated with new code, feature or functionality, it is tested thoroughly to detect if there is any negative impact of the added code. This is known as regression testing.

Testing Documentation

Testing documents are prepared at different stages -

Before Testing

Testing starts with test cases generation. Following documents are needed for reference –

- FRS document - Functional Requirements document
- Test Policy document - This describes how far testing should take place before releasing the product.
- Test Strategy document - This mentions detailed aspects of the test team, responsibility

matrix and rights/responsibility of test manager and test engineer.

- Traceability Matrix document - This is the SDLC document, which is related to the requirement gathering process. As new requirements come, they are added to this matrix. These matrices help testers know the source of requirement. They can be traced forward and backward.

While Being Tested

The following documents may be required while testing is started and is being done:

- Test Case document - This document contains a list of tests required to be conducted. It includes Unit test plan, Integration test plan, System test plan and Acceptance test plan.
- Test description - This document is a detailed description of all test cases and procedures to execute them.
- Test case report - This document contains a test case report as a result of the test.
- Test logs - This document contains test logs for every test case report.

After Testing

The following documents may be generated after testing :

- Test summary - This test summary is collective analysis of all test reports and logs. It summarizes and concludes if the software is ready to be launched. The software is released under the version control system if it is ready to launch.

7. Conclusion and Recommendations Future Scope

Conclusion

The most important thing in conclusion is no new material is included but repeat the objectives you stated in the beginning and then make the conclusion on it ,whether it is fully matched,partially matched or no matched.It is also ok if objectives are not fully matched,then you have to state that in which part is matched, how it is matched.

Limitations

Possible reasons for limitations in your research/project may include:

- Time – time to write the report and collect data
- Budget – inability to collect data first hand
- Access to appropriate interviewees
- Limited to the number of questionnaires returned
- Limited in the scope of the work achieved

- Language
- Cultural reasons

Recommendations

These will be recommendations made as a direct result of the research/project. The recommendations may be for a client/customer, particularly if your research/project is suggested by your sponsoring company,then following points can be asked.

- Suggestions for change
- Recommendations for new implementations of tools/processes/methods
- Advice for further communication of the research/project findings

All of these are as a result of your completed research/project work. It is good practise to make a note of where the recommendations came from within the dissertation, so you should refer back to the appropriate place in the literature review, or data analysis, or discussion chapters. Help the reader by indicating the source of your recommendation.

Future scope

- Future scope that is focused solely on study findings.
- Future scope that is focused on the theory or theoretical model misused.
- Future scope from lack of literary support.
- Future scope on geographical outreach.
- Future scope on testing methods and statistics.

8. Bibliography and References

Bibliography

It means listing of works used and/or considered by an author while doing the research/project.

In bibliography we include :

Citation- all intake citations and other researcher's citations also.

Consulted - work that is consulted or read text found useful.

Useful part/important - certain things are important as part of research/project and they are useful

References

It is referenced within your article/book/report. Bibliography is not directly included in your text, but references to those sources are directly included.Means if there are in text citation for a source there is a reference list entry.A reference can be used to support an argument.

Both bibliography and references are arranged alphabetically.