SOFTWARE ENGINEERING

CHAPTER 3

CONTENTS

Design Engineering :

- Design Engineering Definition
- Process and Quality
- Design Concepts
- The Design Model.

CONTENTS PART II

Software Architecture

Data Design

Architectural Styles and Patterns

Architectural Design

Conceptual Model

Basic Structural Modelling

Class Diagrams

Sequence diagrams

Collaboration diagrams

Use case diagrams

Component Diagrams

PART-I

DESIGN ENGINEERING

DESIGN ENGINEERING

The Design Phase of Software Engineering deals with transforming the customer requirements as described in the SRS.

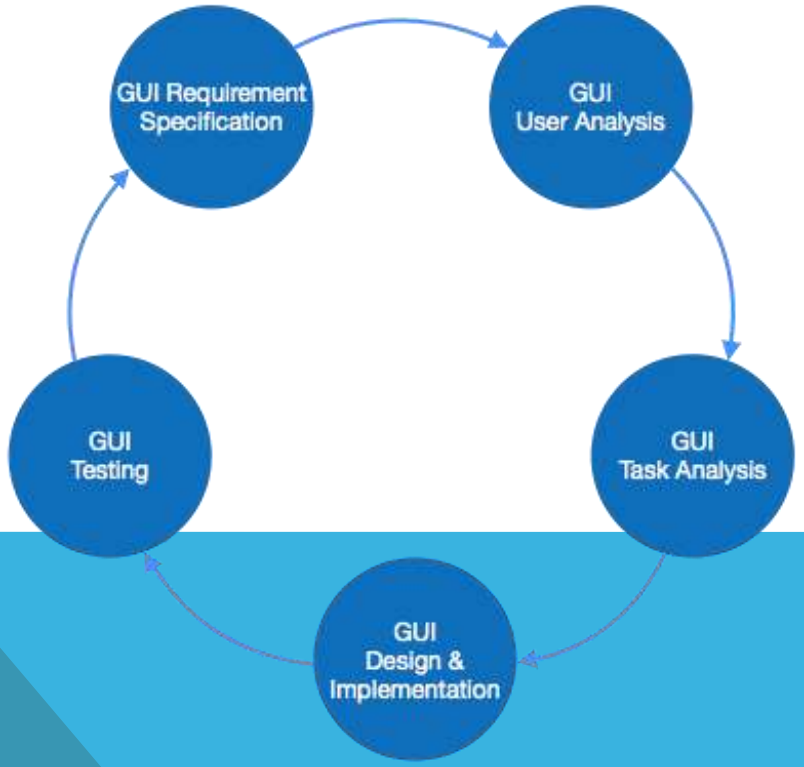
The Design Process can be divided into 3 parts :

- Interface Design
- Architectural Design
- Detailed Design

INTERFACE DESIGN

- *Interface design* is the specification of the interaction between a system and its environment. this phase proceeds at a high level of abstraction with respect to the inner workings of the system i.e, during interface design, the internal of the systems are completely ignored and the system is treated as a black box.
- Interface design should include the following details:
  - Precise description of events in the environment, or messages from agents to which the system must respond.
  - Precise description of the events or messages that the system must produce.
  - Specification on the data, and the formats of the data coming into and going out of the system.

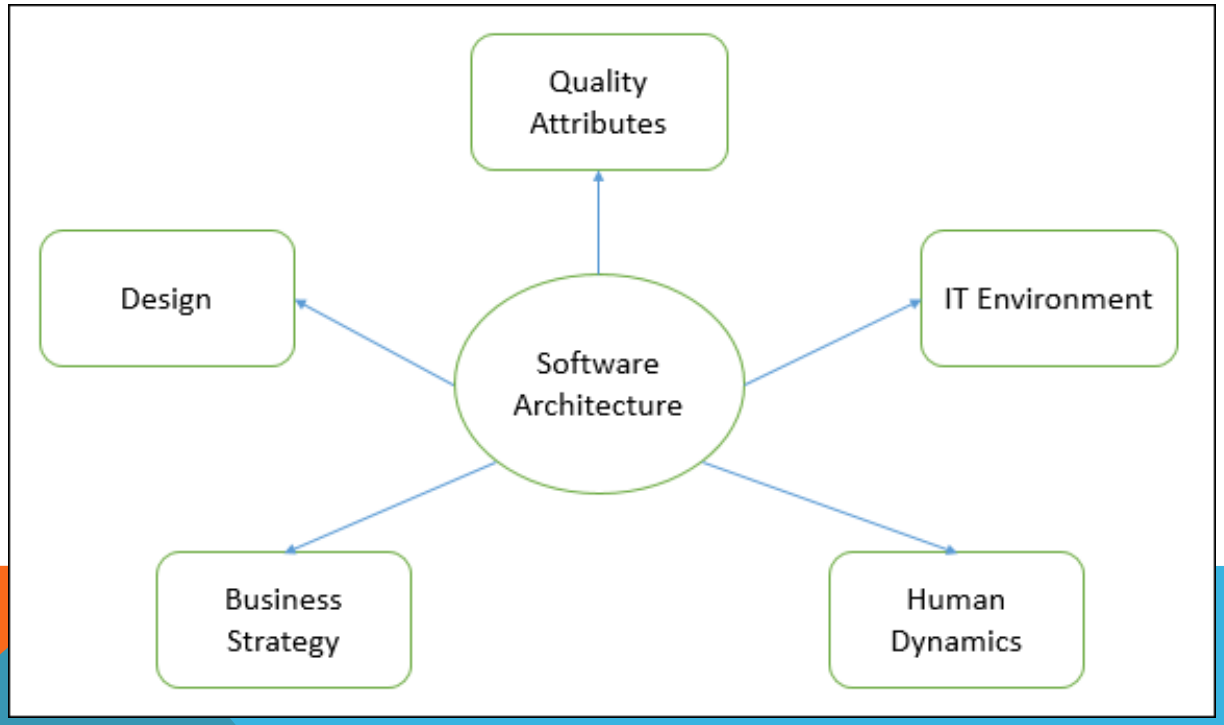
INTERFACE DESIGN



ARCHITECTURAL DESIGN

- *Architectural design* is the specification of the major components of a system, their responsibilities, properties, interfaces, and the relationships and interactions between them.
- In architectural design, the overall structure of the system is chosen, but the internal details of major components are ignored.

ARCHITECTURAL DESIGN



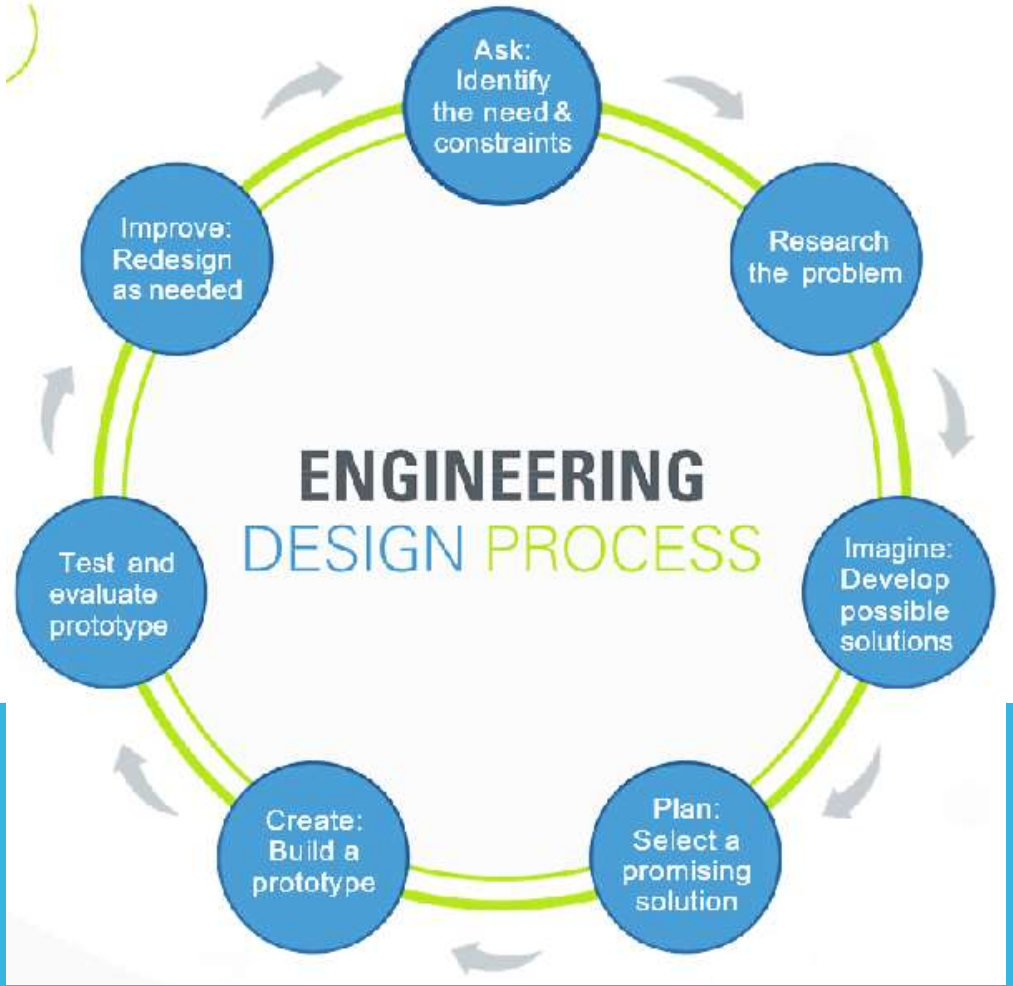
DETAILED DESIGN

- *Design* is the specification of the internal elements of all major system components, their properties, relationships, processing, and often their algorithms and the data structures.
- The detailed design may include: User interfaces, Unit states and state changes, Data and control interaction between units, Algorithms and data structures etc.

DESIGN ENGINEERING PROCESS

- The engineering design process is a series of steps that engineers follow to find a solution to a problem. The steps include problem solving processes such as, for example, determining your objectives and constraints, prototyping, testing and evaluation.
- While the design process is iterative it follows a predetermined set of steps, some of these may need to be repeated before moving to the next one.

DESIGN ENGINEERING PROCESS DIAGRAM



STEPS OF DESIGN ENGINEERING
STEP-1: DEFINE THE PROBLEM

- What is the problem or need?
- Who has the problem or need?
- Why is it important to solve?

STEPS OF DESIGN ENGINEERING
STEP 2: DO BACKGROUND RESEARCH

- Learn from the experiences of others — this can help you find out about existing solutions to similar problems, and avoid mistakes that were made in the past. So, for an engineering design project, do background research in two major areas:
- Users or customers
- Existing solutions

STEPS IN DESIGN ENGINEERING

STEP 3: SPECIFY REQUIREMENTS

Design requirements state the important characteristics that your solution must meet to succeed. One of the best ways to identify the design requirements for your solution is to analyze the concrete example of a similar, existing product, noting each of its key features.

STEPS IN DESIGN ENGINEERING
STEP-4: BRAINSTORM SOLUTIONS

There are always many good possibilities for solving design problems. If you focus on just one before looking at the alternatives, it is almost certain that you are overlooking a better solution. Good designers try to generate as many possible solutions as they can.

STEPS IN DESIGN ENGINEERING

STEP 5: CHOOSE THE BEST SOLUTION

Look at whether each possible solution meets your design requirements. Some solutions probably meet more requirements than others. Reject solutions that do not meet the requirements.

STEPS OF DESIGN ENGINEERING
STEP-6: DEVELOP THE SOLUTION

- Development involves the refinement and improvement of a solution, and it continues throughout the design process, often even after a product ships to customers.

STEPS OF DESIGN ENGINEERING
STEP-7:BUILD A PROTOTYPE

A prototype is an operating version of a solution.

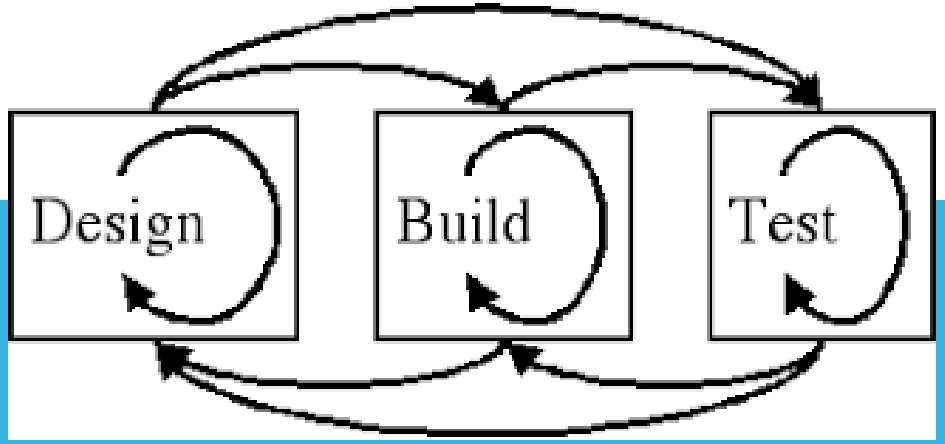
Often it is made with different materials than the final version, and generally it is not as polished.

Prototypes are a key step in the development of a final solution, allowing the designer to test how the solution will work.

STEPS OF DESIGN ENGINEERING
STEP-8: TEST AND RE-DESIGN

The design process involves multiple iterations and redesigns of your final solution.

You will likely test your solution, find new problems, make changes, and test new solutions before settling on a final design.



STEPS OF DESIGN ENGINEERING
STEP-COMMUNICATE RESULTS

- To complete your project, communicate your results to others in a final report and/or a display board.
- Professional engineers always do the same, thoroughly documenting their solutions so that they can be manufactured and supported.

DESIGN ENGINEERING QUALITY

- Quality engineering is the discipline of engineering concerned with the principles and practice of product and service quality assurance and control.
- In software development, it is the management, development, operation and maintenance of IT systems and enterprise architectures with a high quality standard.

DESIGN CONCEPTS

A set of the Concepts that go hand in hand along with the design engineering of the software system are known as Design Concepts

These Concepts are a key factor in analyzing the design of the software.

There are Few Concepts that are categorized as Design Concepts

1. ABSTRACTION

A solution is stated in large terms using the language of the problem environment at the highest level abstraction.

The lower level of abstraction provides a more detail description of the solution.

A sequence of instruction that contain a specific and limited function refers in a procedural abstraction.

A collection of data that describes a data object is a data abstraction.

2. ARCHITECTURE

The complete structure of the software is known as software architecture.

Structure provides conceptual integrity for a system in a number of ways.

The architecture is the structure of program modules where they interact with each other in a specialized way.

The components use the structure of data.

3. PATTERNS

A design pattern describes a design structure and that structure solves a particular design problem in a specified content.

4. MODULARITY

A software is separately divided into name and addressable components. Sometime they are called as modules which integrate to satisfy the problem requirements.

Modularity is the single attribute of a software that permits a program to be managed easily.

5. INFORMATION HIDING

Modules must be specified and designed so that the information like algorithm and data presented in a module is not accessible for other modules not requiring that information.

6. FUNCTIONAL INDEPENDENCE

- The functional independence is the concept of separation and related to the concept of modularity, abstraction and information hiding.
- The functional independence is accessed using two criteria i.e Cohesion and coupling.

6.1 COHESION

Cohesion is an extension of the information hiding concept.

A cohesive module performs a single task

It requires a small interaction with the other components in other parts of the program.

6.2 COUPLING

Coupling is an indication of interconnection between modules in a structure of software.

More precisely it is the interdependence between software modules.

Or it can be a measure of how closely connected two modules are.

7. REFINEMENT

- Refinement is a top-down design approach.
- It is a process of elaboration.
- A program is established for refining levels of procedural details.
- A hierarchy is established by decomposing a statement of function in a stepwise manner till the programming language statement are reached.

8. REFACTORING

It is a reorganization technique which simplifies the design of components without changing its function behavior.

Refactoring is the process of changing the software system in a way that it does not change the external behavior of the code still improves its internal structure.

9. DESIGN CLASSES

- The model of software is defined as a set of design classes.
- Every class describes the elements of problem domain and that focus on features of the problem which are user visible.

THE DESIGN MODEL

- Design modeling in software engineering represents the features of the software that helps engineer to develop it effectively, the architecture, the user interface, and the component level detail.
- Different methods like data-driven, pattern-driven, or object-oriented methods are used for constructing the design model.
- All these methods use set of design principles for designing a model.

DESIGN MODELLING

- The design model builds on the analysis model by describing, in greater detail, the structure of the system and how the system will be implemented.
- In the design model, packages contain the design elements of the system, such as design classes, interfaces, and design subsystems, that evolve from the analysis classes each package can contain any number of sub-packages that further partition the contained design elements.
- These architectural layers form the basis for a second-level organization of the elements that describe the specifications

SOFTWARE ENGINEERING CHAPTER 3

PART 2

SOFTWARE ARCHITECTURE

- Software architecture refers to the fundamental structures of a software system and the discipline of creating such structures and systems.
- The architecture of the system is a metaphor analogous to the architecture of a building.
- Functions as a blueprint for the development of the software.

SOFTWARE ARCHITECTURE TYPES

The most used software architectures are:

- Business Architecture
- Application Architecture
- Information Architecture
- Information Technology Architecture

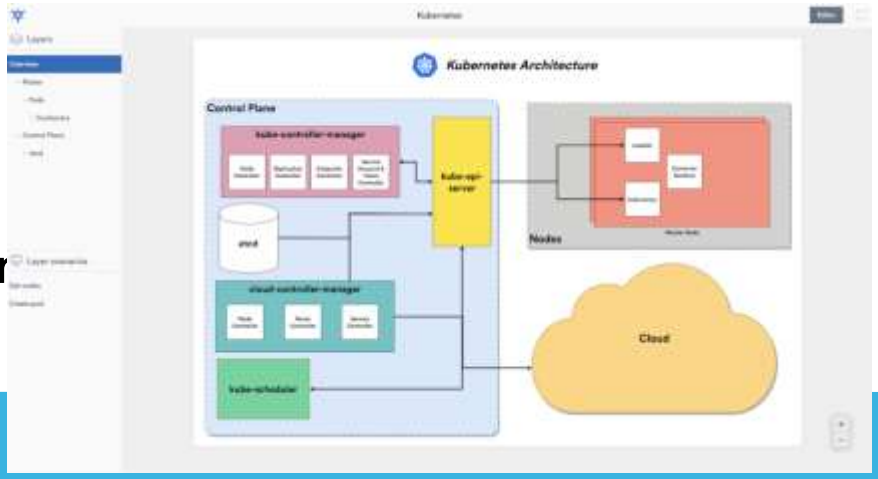
BUSINESS ARCHITECTURE

Business architecture defines the strategy of business, governance, organization and key business processes within an enterprise.

This type of architecture focuses on the analysis and design of business processes.

APPLICATION ARCHITECTURE

- It describes the patterns and techniques used to design and build an application.
- Gives a roadmap and best practices to follow when building an application.
- This diagram is a represer



INFORMATION ARCHITECTURE

- It is structural design of shared information environments; the art and science of organizing and labelling websites, intranets, online communities and software to support usability and findability.
- A discipline that focuses on the organization of information within digital products.

INFORMATION TECHNOLOGY ARCHITECTURE

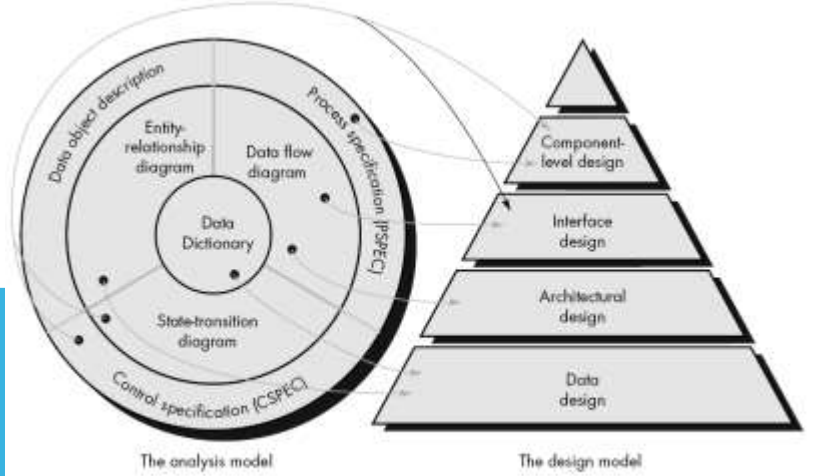
Process of development of methodical information technology specifications, models and guidelines.

It uses a variety of Information Technology notations for example UML within a coherent information architecture.

Focuses on three basic tiers within organization.

DATA DESIGN

- The first design activity resulting in a less complex, modular and efficient program.
- The information domain model developed during analysis phase is transformed into data structures needed for implementing the software



Translating the analysis model into a software design

ARCHITECTURAL STYLE

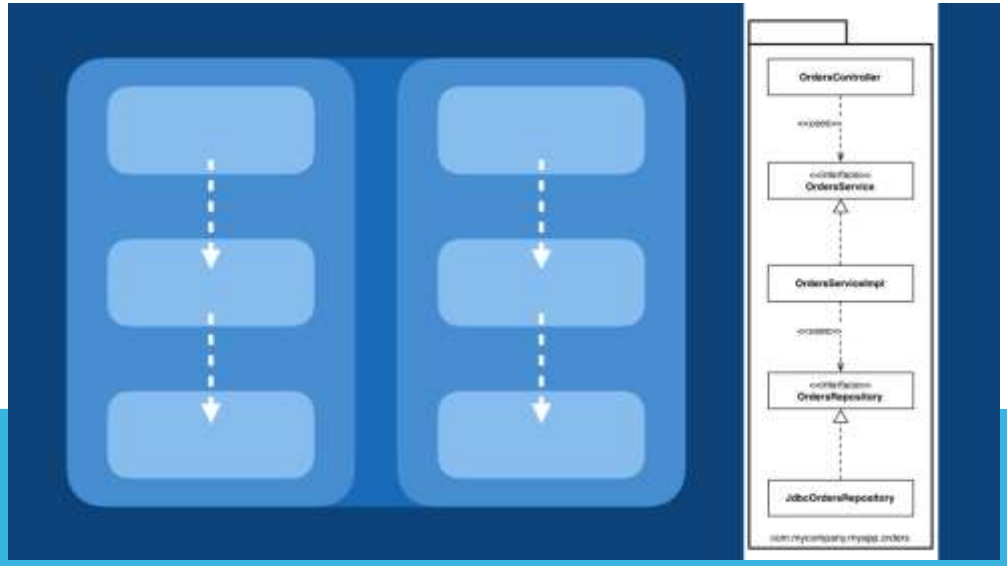
Shows how we organize our code or how the system will look like from an aerial view.

The list of style:

- Structure Architecture Style
- Messaging Styles
- Distributed Systems
- Shared memory Styles
- Adaptive System Style.

STRUCTURE ARCHITECTURE STYLE

- It consists of Several Component based Styles such as :
  - Layers
  - Pipes
  - Filters



MESSAGING STYLES

Messaging refers to various form s of communication information to users

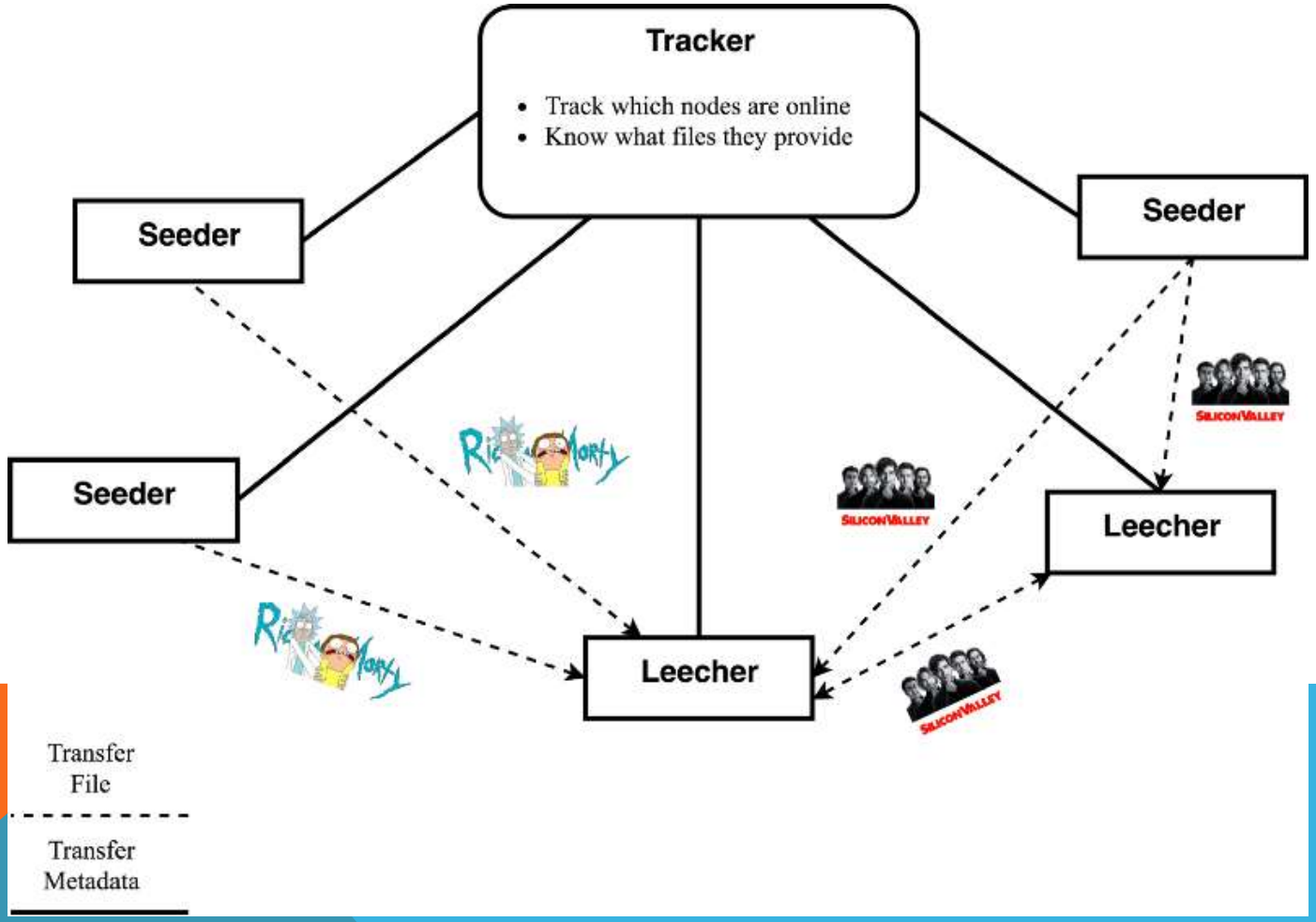
Some of the forms are : E-Mail , SMS, EMS, MMS , Instant Messaging , HDML Notifications , WAP Push

Based on the software requirement, developer can design the messaging system.

DISTRIBUTED SYSTEMS

- A distributed computer system consists of multiple software components that are on multiple computers, but run as a single system.
- The computers that are in a distributed system can be physically close together and connected by a local network, or they can be geographically distant and connected by a wide area network.

DISTRIBUTED SYSTEM



SHARED MEMORY STYLES

- Shared Memory Consists of three types:

Database Centric : Based on a DB

Blackboard: An AI Approach

Rule based: Applicable in most systems where automatic rule inference re executed.

SHARED MEMORY

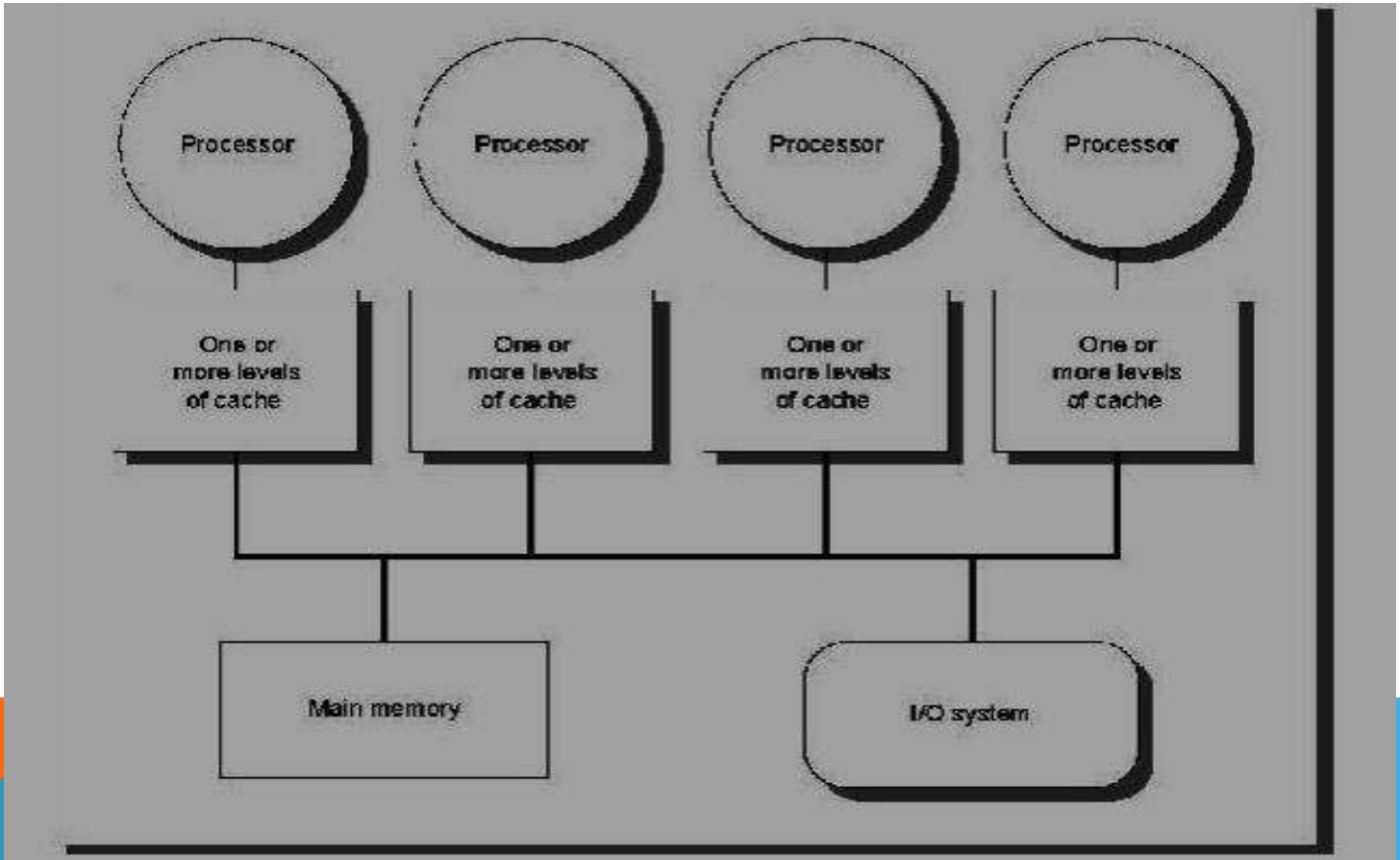
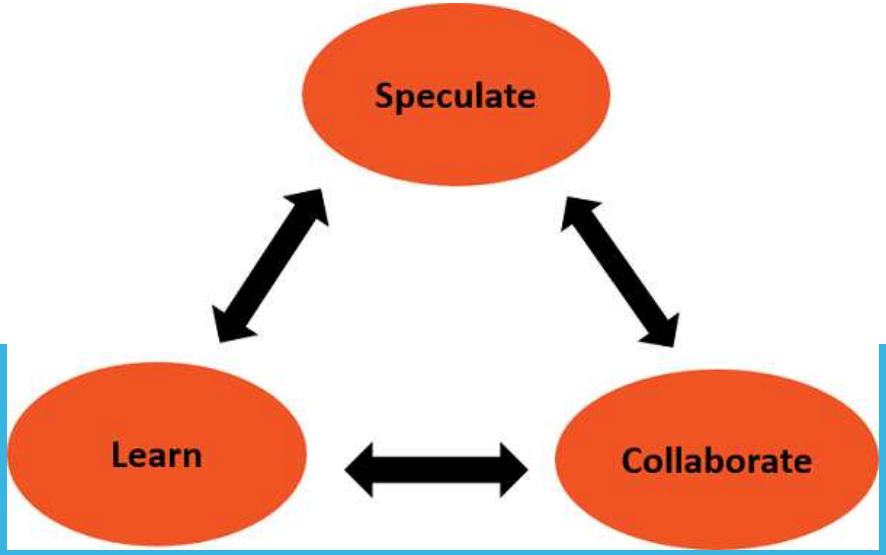


FIGURE 3.1 Symmetric Shared Memory Architecture

ADAPTIVE SYSTEM STYLES

These Styles consists of Microkernel Style , reflection , domain Specific language styles .

The System that changes its behavior in response to its environment.



ARCHITECTURAL PATTERNS

- Architectural Design Pattern are accumulative best practices and experiences that software professionals used over the years to solve the general problem by – trial and error – they faced during software development.
- Two main principles of object-oriented design:
  - Develop to an interface, not to an implementation.
  - Favor object composition over inheritance.

  They are Creational Patterns, Structural Patterns, Behavioral Patterns.

CREATIONAL DESIGN PATTERN

- Provide a way to create objects while hiding the creation logic. Thus, the object creation is to be done without instantiating objects directly with the “New” keyword to gives the flexibility to decide which objects need to be created for a given use case.
- Abstract Factory Pattern, Singleton Pattern, Builder Pattern and Prototype Pattern.

STRUCTURAL PATTERN

- These Patterns are concerned with class and object composition of the system
- Adapter, Bridge, Filter, Composite, Decorator, Façade, Flyweight and Proxy Come under these Structural Pattern.

BEHAVIORAL PATTERN

Behavioral Patterns are concerned with communications between objects.

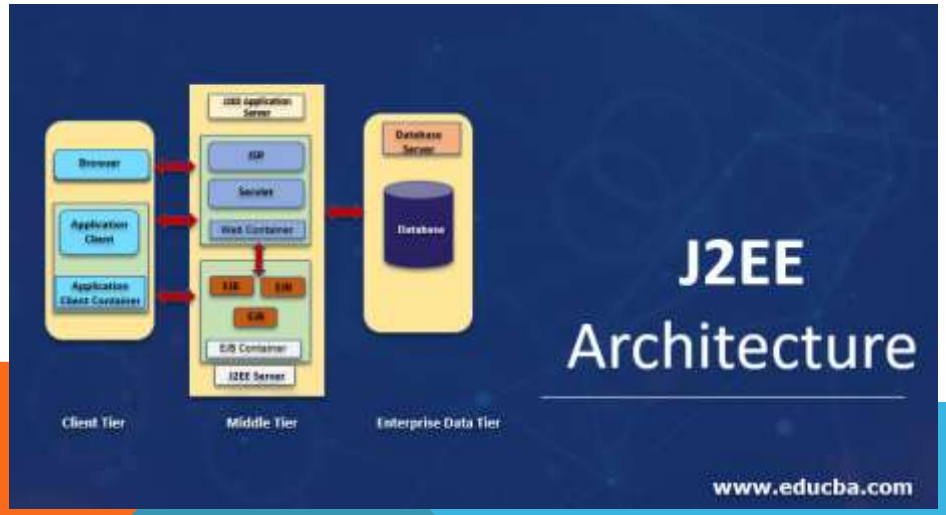
The communication can be of any type.

Responsibility, Command, Interpreter, Iterator, Mediator, Memento, Observer, State, Null, Strategy, Template and Visitor Come under Behavioral Pattern.

J2EE PATTERNS

These Patterns are specifically concerned with presentation tier

It was identified by Sun Java Center at Menlo Park in California USA.



ARCHITECTURAL DESIGN

The process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of a computer system.

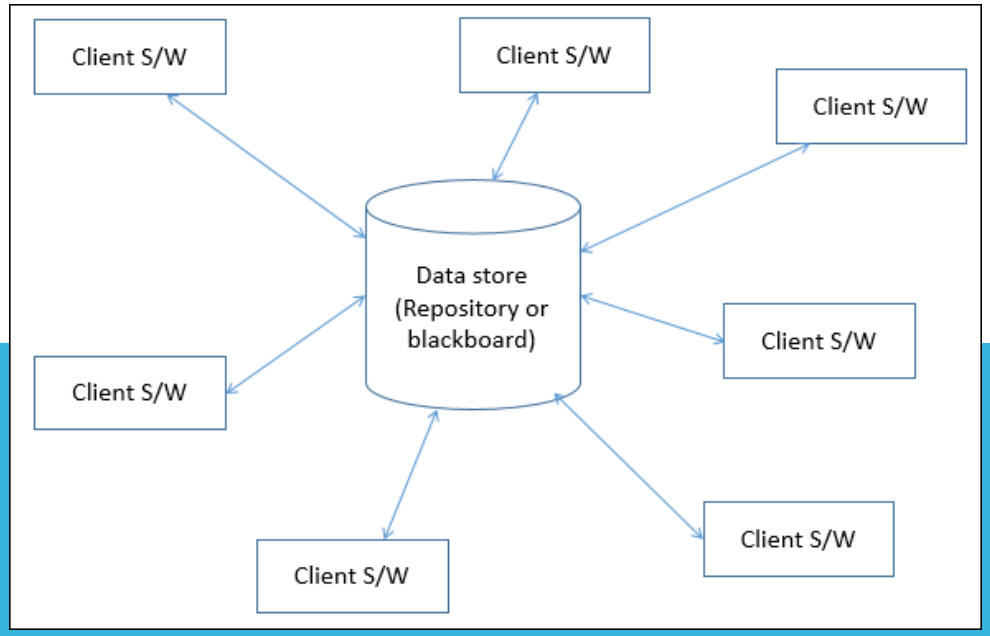
The Various Designs are

- Data centered
- Data Flow
- Call and Return
- Object Oriented
- Layered

DATA CENTERED

A data store will reside at the center of this architecture and is accessed frequently by the other components that update, add, delete or modify the data present within the store.

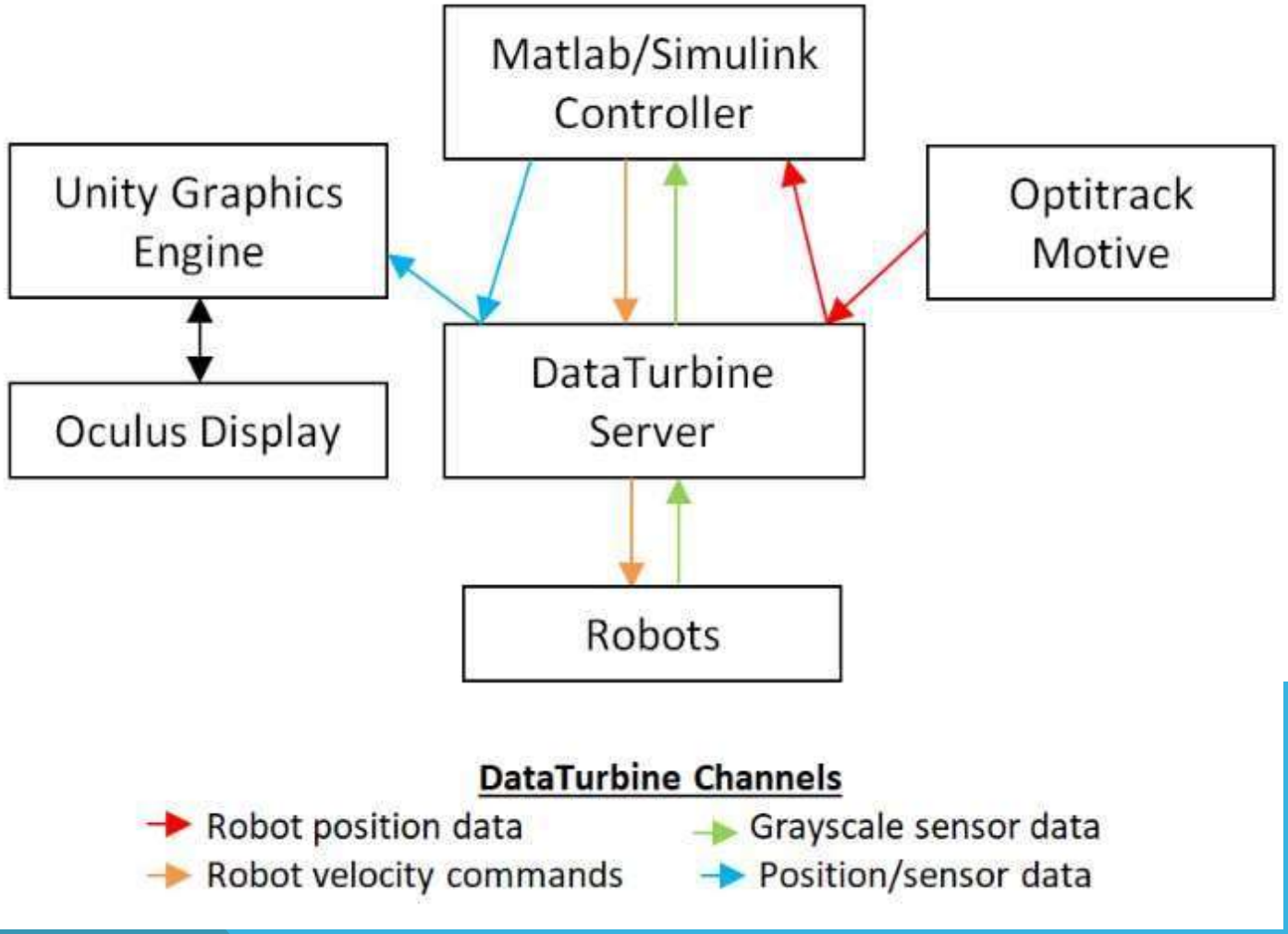
This data-centered architecture will promote integrity



DATA FLOW ARCHITECTURE

- This kind of architecture is used when input data to be transformed into output data through a series of computational manipulative components.
- Pipes are used to transmit data from one component to the next.
- Each filter will work independently and is designed to take data input of a certain form and produces data output to the next filter of a specified form. The filters don’t require any knowledge of the working of neighboring filters.

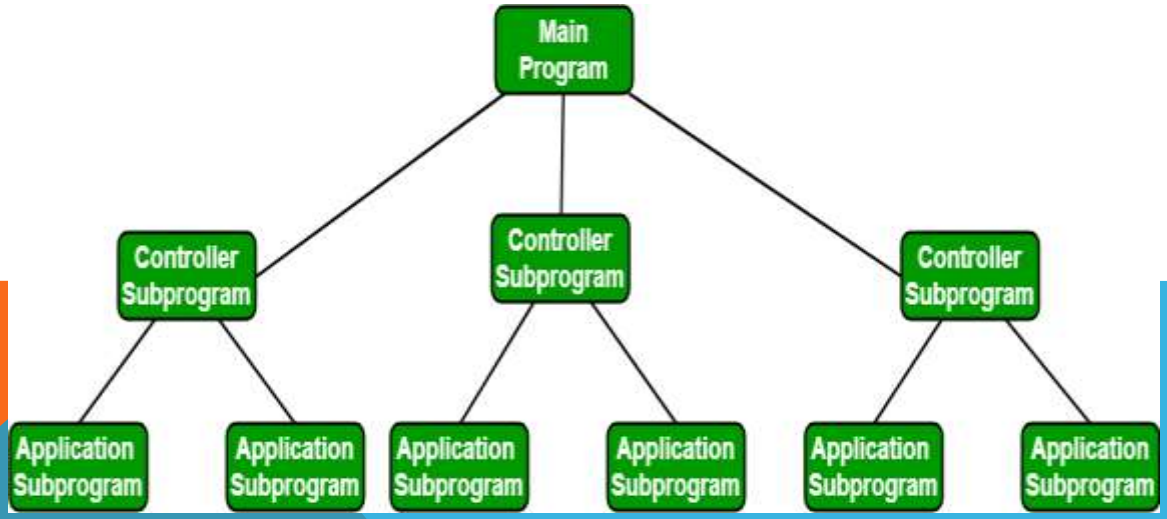
DATA FLOW ARCHITECTURE



CALL AND RETURN ARCHITECTURES

It is used to create a program that is easy to scale and modify. Many sub-styles exist within this category

Sub Categories Include: Remote Procedure call and Main program-Sub program Architectures.



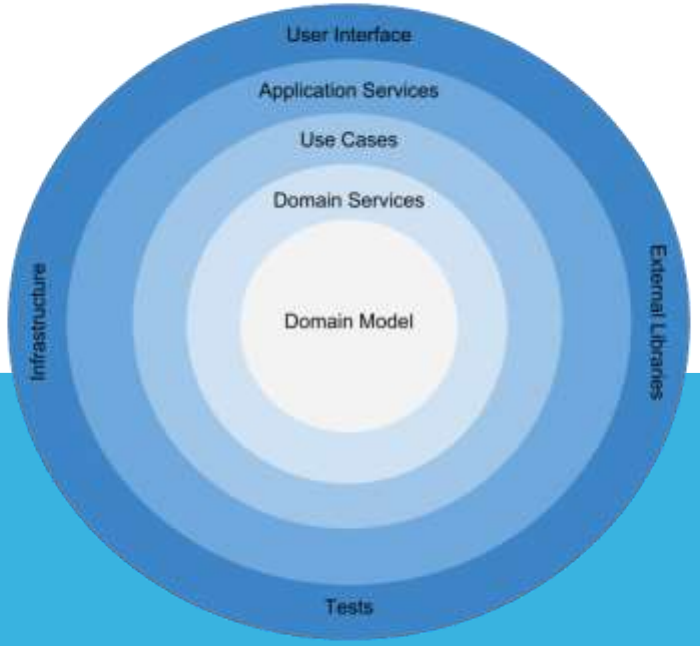
OBJECT ORIENTED ARCHITECTURE

- The components of a system encapsulate data and the operations that must be applied to manipulate the data.
- The coordination and communication between the components are established via the message passing.

Fig. Advantages of Object Oriented Architecture

LAYERED ARCHITECTURE

- A number of different layers are defined with each layer performing a well-defined set of operations.
- Each layer will do some operations that becomes closer to machine instruction set progressively.



CONCEPTUAL MODEL

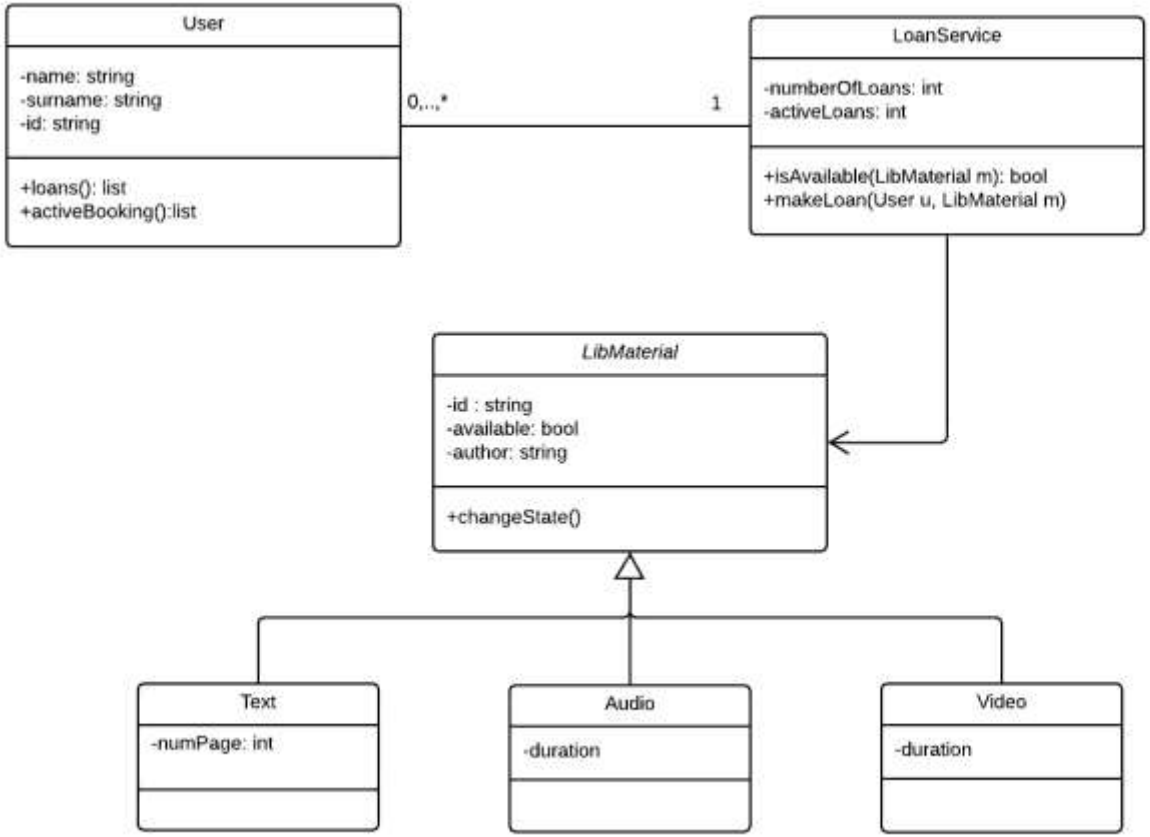
- It is a representation of a system that uses concepts and ideas to form said representation.
  Used across many fields ranging from sciences to socioeconomics to software development.
- These Models try to capture people's understanding of what is being modeled.

CLASS DIAGRAM

It is a Static diagram representing the static view of an application. It is not only used for visualizing , describing and documenting aspects but also for constructing executable code.

Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

CLASS DIAGRAMS



SEQUENCE DIAGRAMS-UML

UML is a modelling Language in the field of software engineering which aims to set standard ways to visualize the design of a system.

The Sequence Diagram is an Interaction Diagram.

It Simply depicts interaction between objects in a sequential order.

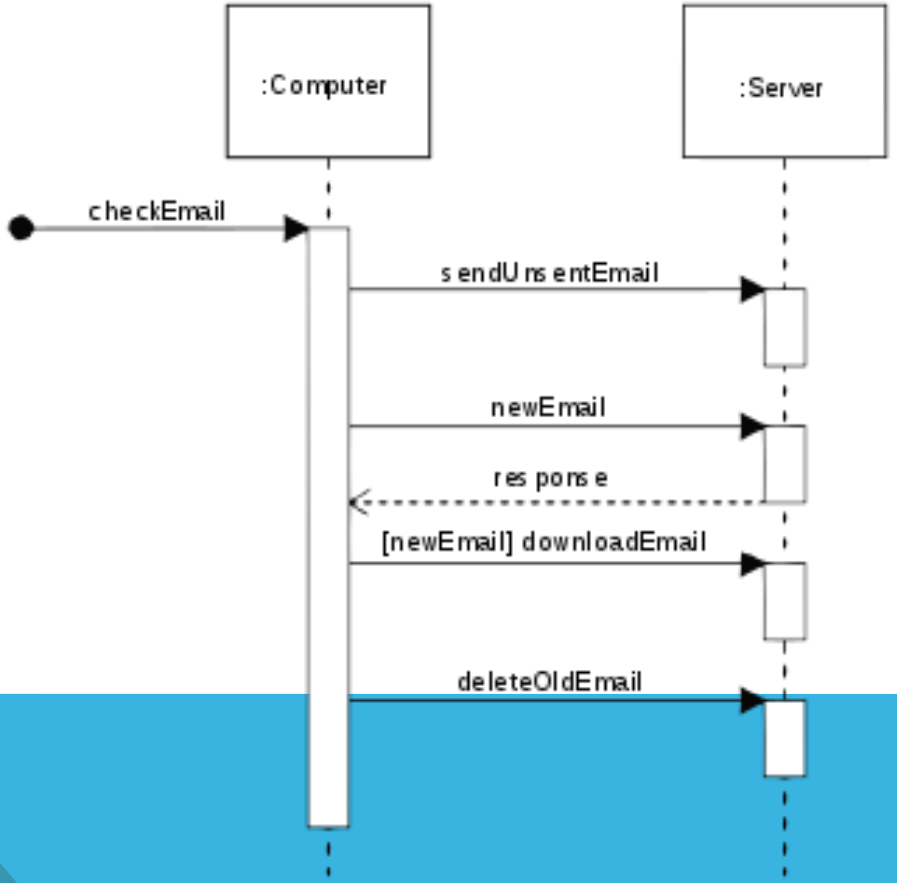
SEQUENCE DIAGRAMS

We also use the terms event diagrams or event scenarios to refer to the diagram.

Describe how and in what order the objects in a system function.

The Parts are : Actors, Lifelines, Messages, Guards,

SEQUENCE DIAGRAM



ADVANTAGE OF SEQUENCE DIAGRAM

Used to model and visualize the logic behind a sophisticated function, operation or procedure.

They are also used to show details of UML use case diagrams.

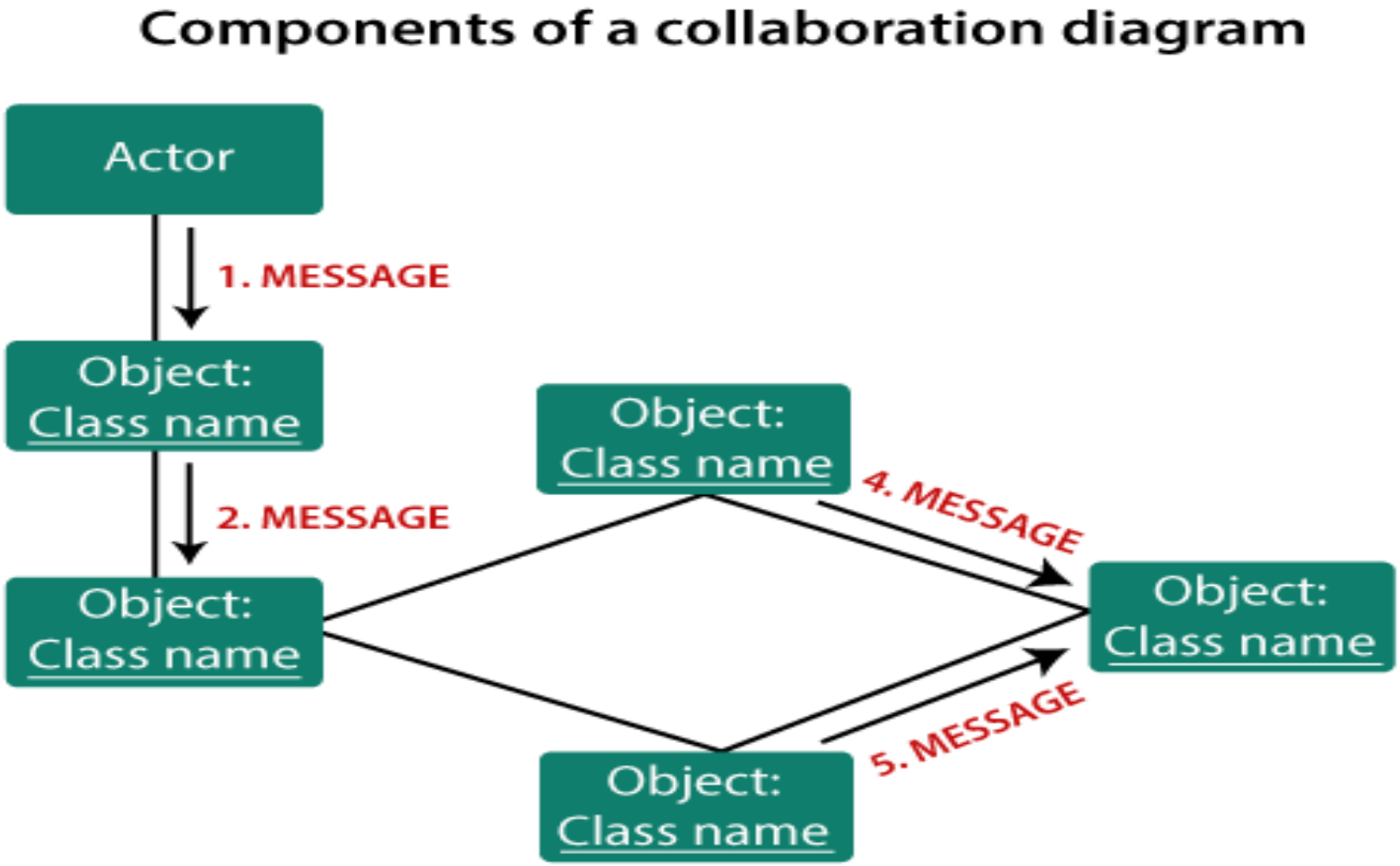
Used to understand the detailed functionality of current or future systems.

Visualize how messages and tasks move between objects or components in a system

COLLABORATION DIAGRAMS

- A collaboration diagram, also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). These diagrams can be used to portray the dynamic behavior of a particular use case and define the role of each object.

COLLABORATION DIAGRAM



ADVANTAGES OF COLLABORATION DIAGRAM

- The collaboration diagram is also known as Communication Diagram.
- It mainly puts emphasis on the structural aspect of an interaction diagram, i.e., how lifelines are connected.
- The syntax of a collaboration diagram is similar to the sequence diagram; just the difference is that the lifeline does not consist of tails.
- The special case of a collaboration diagram is the object diagram.
- It focuses on the elements and not the message flow, like sequence diagrams.

DISADVANTAGE OF COLLABORATION DIAGRAM

Multiple objects residing in the system can make a complex collaboration diagram, as it becomes quite hard to explore the objects.

It is a time-consuming diagram.

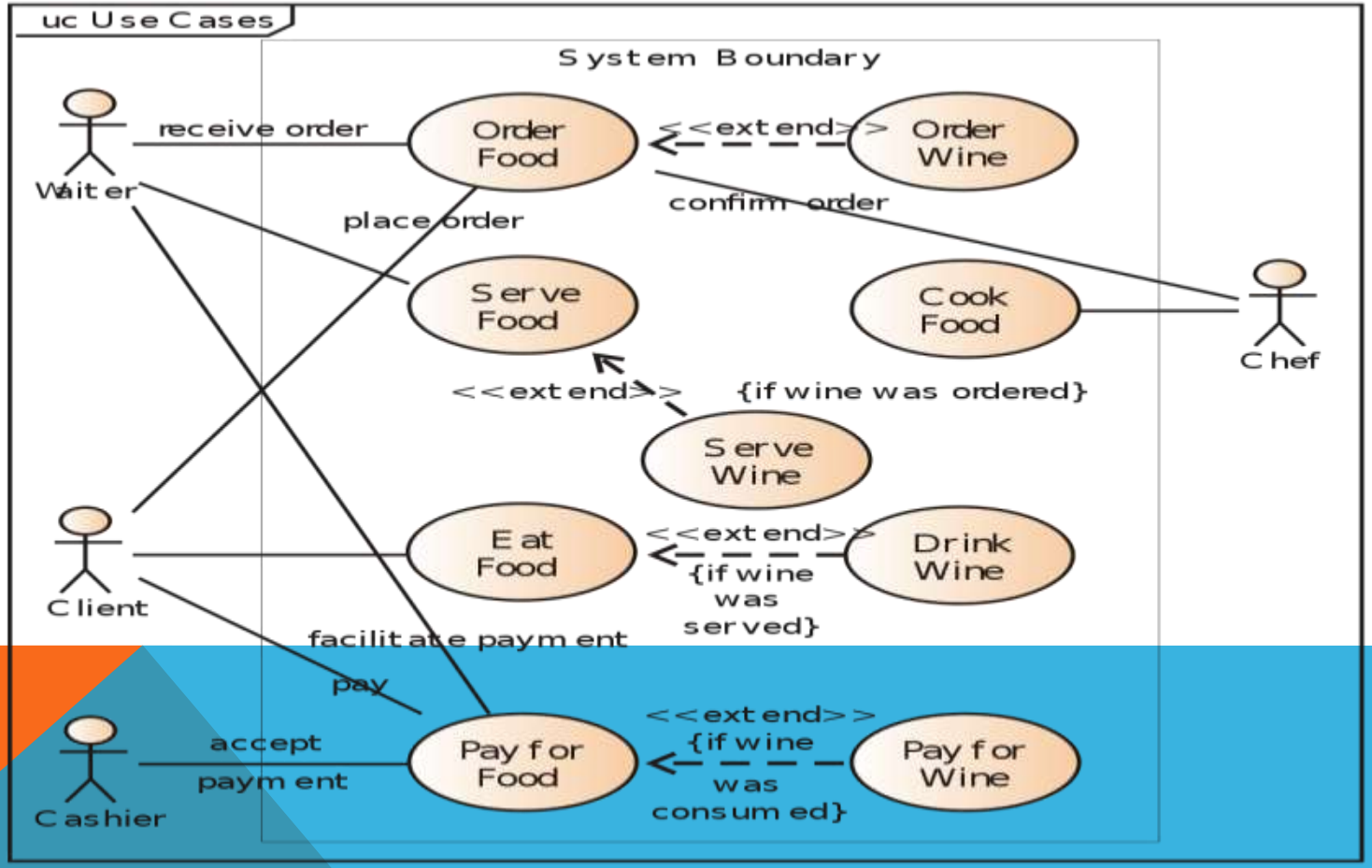
After the program terminates, the object is destroyed.

As the object state changes momentarily, it becomes difficult to keep an eye on every single that has occurred inside the object of a system.

UML USE CASE DIAGRAM

- A use case diagram is used to represent the dynamic behavior of a system.
- It encapsulates the system's functionality by incorporating use cases, actors, and their relationships.
- It models the tasks, services, and functions required by a system/subsystem of an application.
- It depicts the high-level functionality of a system and also tells how the user handles a system.
- Accumulates system’s requirement.

UML USE CASE DIAGRAM



UML USE CASE DIAGRAM BENEFITS

It gathers the system's needs.

It depicts the external view of the system.

It recognizes the internal as well as external factors that influence the system.

It represents the interaction between the actors.

COMPONENT DIAGRAM

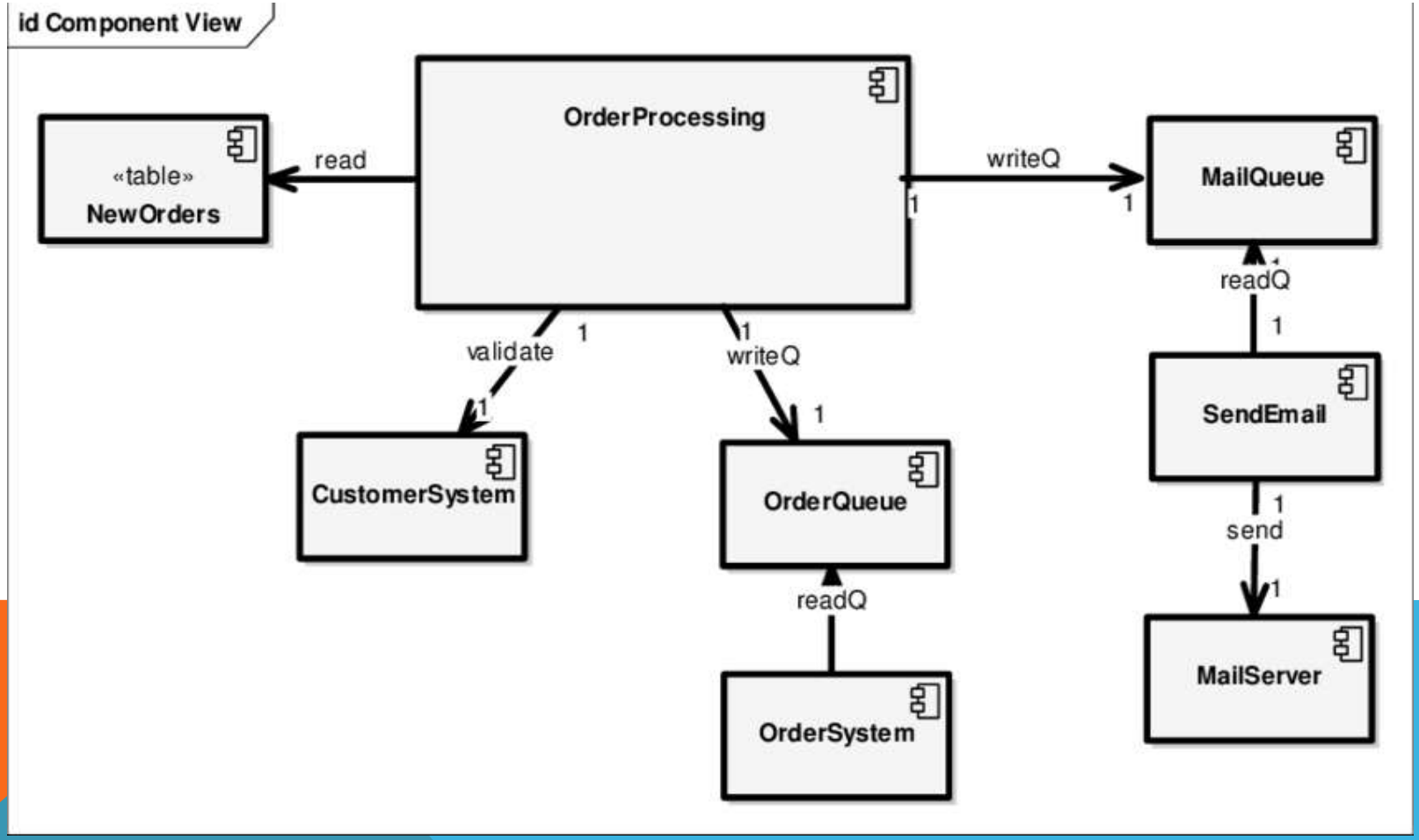
Component diagrams are different in terms of nature and behavior.

Component diagrams are used to model the physical aspects of a system.

Physical aspects are the elements such as executables, libraries, files, documents, etc. which reside in a node.

Component diagrams are used to visualize the organization and relationships among components in a system.

COMPONENT DIAGRAM



UML COMPONENT DIAGRAM ADVANTAGES

Component diagrams are very simple, standardized, and very easy to understand.

It is also useful in representing implementation of system.

These are very useful when you want to make a design of some device that contains an input-output socket.

Use of reusable components also helps in reducing overall development cost.

It is very easy to modify and update implementation without any causing any other side effects.