Chapter -3 Architectural design:

-Sunil Lama

Architectural design is concerned with understanding how a system should be organized and designing the overall structure of that system. In the model of the software development process, architectural design is the first stage in the software design process. It is the critical link between design and requirement engineering, as it identifies the main structural components in a system and the relationships between them. The output of the architectural design process is an architectural model that describes how the system is organized as a set of communicating components.

3.1 Architectural design decisions:

Architectural design is a creative process where you design a system organization that will satisfy the functional and non-functional requirements of a system. Because it is a creative process the activities within the process depend on the type of system being developed, the background and experience of the system architect, and the specific requirements for the system. It is therefore useful to think of architectural design as a series of decisions to be made rather than a sequence of activities.

During the architectural design process, system architects have to make a number of structural decisions that profoundly affect the system and its development process. Based on their knowledge and experience, they have to consider the following fundamental questions about the system:

1. Is there generic application architecture that can act as a template for the system that is being designed?
2. What strategy will be used to control the operation of the components in the system?
3. What architectural organization is best for delivering the non-functional requirement of the system?
4. How will the architectural design be evaluated?
5. How should the architecture of the system be documented?

Although each software system is unique, systems in the same application domain often have similar architectures that reflect the fundamental concepts of the domain. For embedded systems and systems designed for personal computers, there is usually only a single processor and you will not have to design a distributed architecture for the system. However, most large systems are now distributed systems in which the system software is distributed across many different computers. The choice of distribution architecture is a key decision that affects the performance and reliability of the system.

Because of the close relationship between nonfunctional requirements and software architecture, the particular architectural style and structure that you choose for a system should depend on the nonfunctional system requirements:

1. Performance: If performance is a critical requirement, the architecture should be designed to localize critical operations within a small number of components, with these components all deployed on the same computer rather than distributed across the network.
2. Security: If security is critical requirement, a layered structure for the architecture should be used, with the most critical assets protected in the innermost layers, with a high level of security validation applied to these layers.
3. Safety: If safety is the critical requirement, the architecture should be designed so that safety related operations are all located in either a single component or in a small number of components. This reduces the costs and problems of safety validation and makes it possible to provide related protection systems that can safely shut down the system in the event of failure.
4. Availability: If availability is a critical requirement, the architecture should be designed to include redundant components so that it is possible to replace and update components without stopping the system.
5. Maintainability: If maintainability is a critical requirement, the system architecture should be designed using fine-grain, self-contained components that may readily be changed. Producers of data should be separated from consumers and shared data structures should be avoided.

3.2 System organization:

- Reflects the basic strategy that is used to structure a system.
- Three organizational styles are widely used:
  1. A shared data repository style,
  2. A shared services and servers style,
  3. An abstract machine or layered style.

The repository model

- Sub-systems must exchange data. This may be done in two ways:
  1. Shared data is held in a central database or repository and may be accessed by all sub-systems,
  2. Each sub-system maintains its own database and passes data explicitly to other sub-systems.
- When large amounts of data are to be shared, the repository model of sharing is most commonly used.

Repository model characteristics:

- Advantages:
  1. Efficient way to share large amount of data.
  2. Sub-systems need not be concerned with how data is produced centralized management e.g. backup, security etc.
  3. Sharing model is published as the repository schema.
- Disadvantages
  1. Sub-systems must agree on a repository data model. Inevitably a compromise.
  2. Data evolution is difficult and expensive.
  3. No scope for specific management policies,
  4. Difficult to distribute efficiently.

3.3 Modular decomposition styles:

- Styles of decomposing sub-systems into modules.
- No rigid distinction between system organization and modular decomposition.

Sub-systems and modules

- A sub-system is a system in its own right whose operation is independent of the services provided by other sub-systems.
- A module is a system component that provides services to other components but would not normally be considered as a separate system.

Modular decomposition

- Another structural level where sub-systems are decomposed into modules.
- Two modular decomposition models covered
    I. An object model where the system is decomposed into interacting objects,
    II. A pipeline or data-flow model where the systems is decomposed into functional modules which transform inputs to outputs.
- If possible, decisions about concurrency should be delayed until modules are implemented.

3.4 Control styles

- Are concerned with the control flow between sub-systems. Distinct from the system decomposition model.
- Centralized control
    1. One sub-system has overall responsibility for control and starts and stops other sub-systems.
- Event-based control
    1. Each sub-system can respond to externally generated events from other sub-systems or the system's environment.

Centralized control:

- A control sub-system takes responsibility for managing the execution of other sub-systems.
- Call-return model:

1. Top-down subroutine model where control starts at the top of a subroutine hierarchy and moves downwards. Applicable to sequential systems.

- Manager model:
    1. Applicable to concurrent systems. One system component controls the stopping, starting and coordination of other system processes. Can be implemented in sequential systems as a case statement.

Event – driven systems:

- Driven by externally generated events where the timing of the event is out with the control of the sub-systems which process the event.
- Two principal event – driven models
    1. Broadcast models: An event is broadcast to all sub-systems. Any sub-system which can handle the event may do so.
    2. Interrupt- driven models: Used in real-time systems where interrupts are detected by an interrupt handler and passed to some other component for processing.
- Other event driven models include spreadsheets and production systems.

3.5 Reference Architecture:

Reference architecture captures important features of system architectures in a domain. Essentially, they include everything that might be in application architecture although, in reality, it is very unlikely that any individual application would include all the features shown in a reference architecture. The main purpose of reference architectures is to evaluate and compare design proposal, and educate people about architectural characteristics in that domain.

- Architectural models may be applicable to some application domain:
    1. Generic model
    2. Reference model

  (Generic are bottom-top model whereas reference models follow top-down approach.)

- Reference models are derived from study of application domain rather than from existing system- maybe used as a basic system implementation or to compare different system.
- It acts as a standard against which system can be evaluated.

OSI reference model is a layered model of communication system

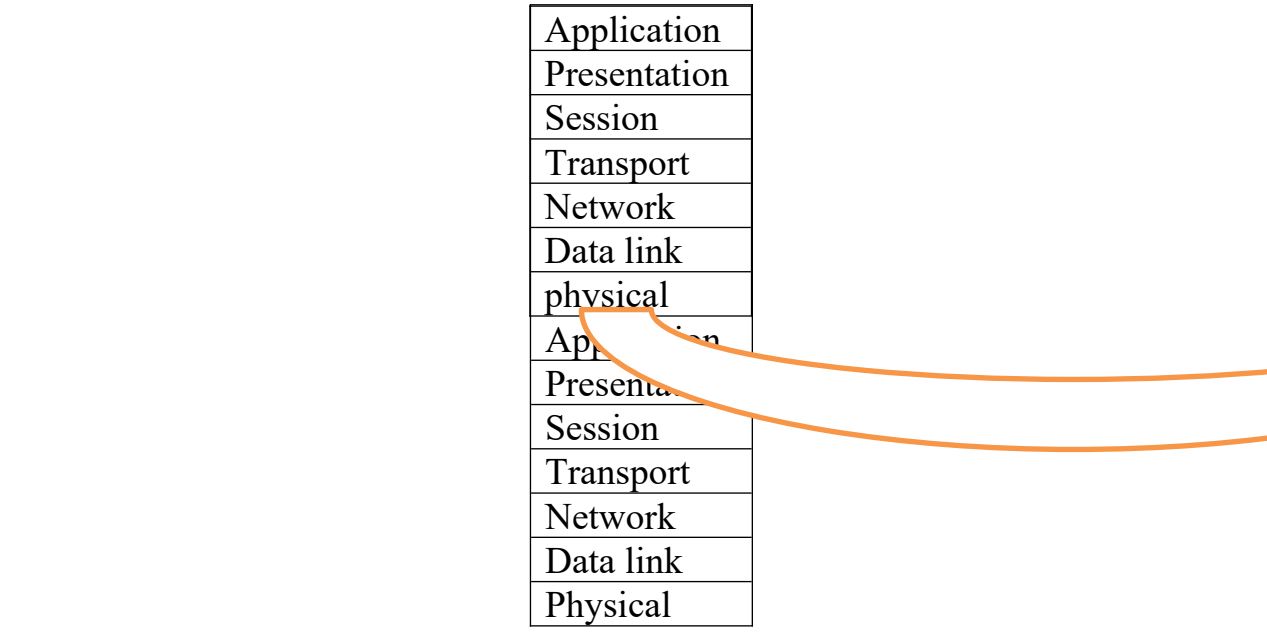
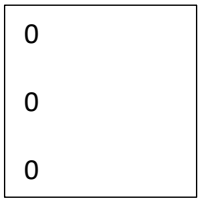


Fig: OSI reference model

3.6 Multiprocessor architecture:

Traffic flow traffic light control



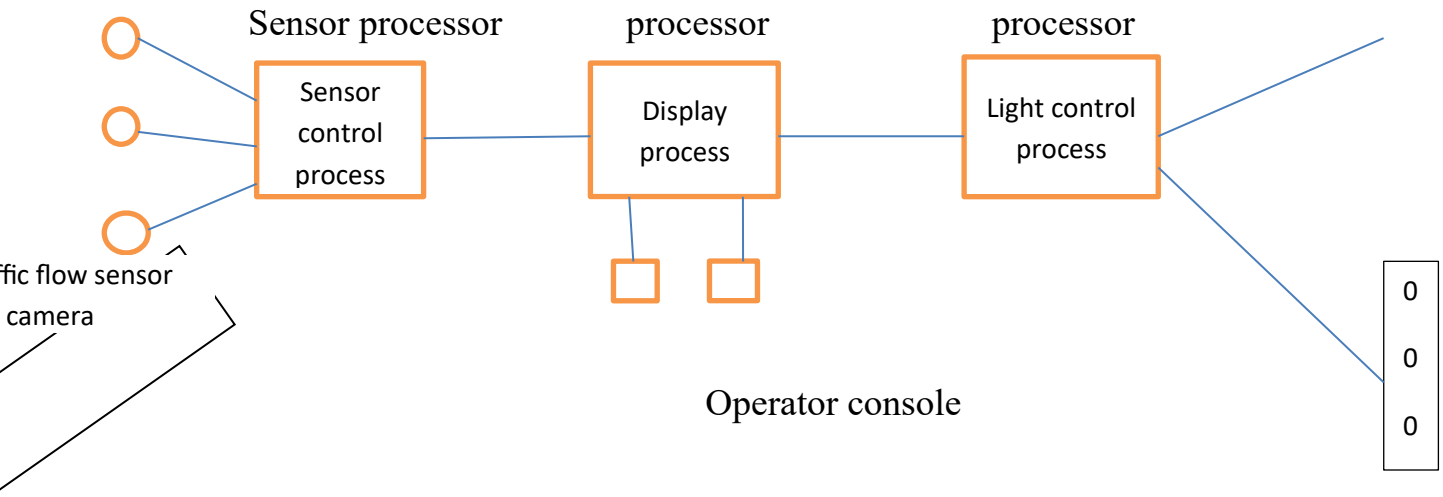


Fig: Simple multiprocessor architecture for traffic light system

It is the simplest distributed system model. The system is composed of multiple processor processes which may execute on different processor. It is Architectural model of many large real time systems. In this architecture, distribution of process to processor may be preordered or may be under the control of the dispatcher.

3.7 Client – server architecture:

- This architectural model is used for the set of services that are provided by server and a set of clients that use this service.
- Client should know about the server but the server need not know about clients in client-server architecture.
- Mapping of processor to process is not necessarily 1:1 in case of client-server architecture.
- Client and server are logical processors in client-server architecture.

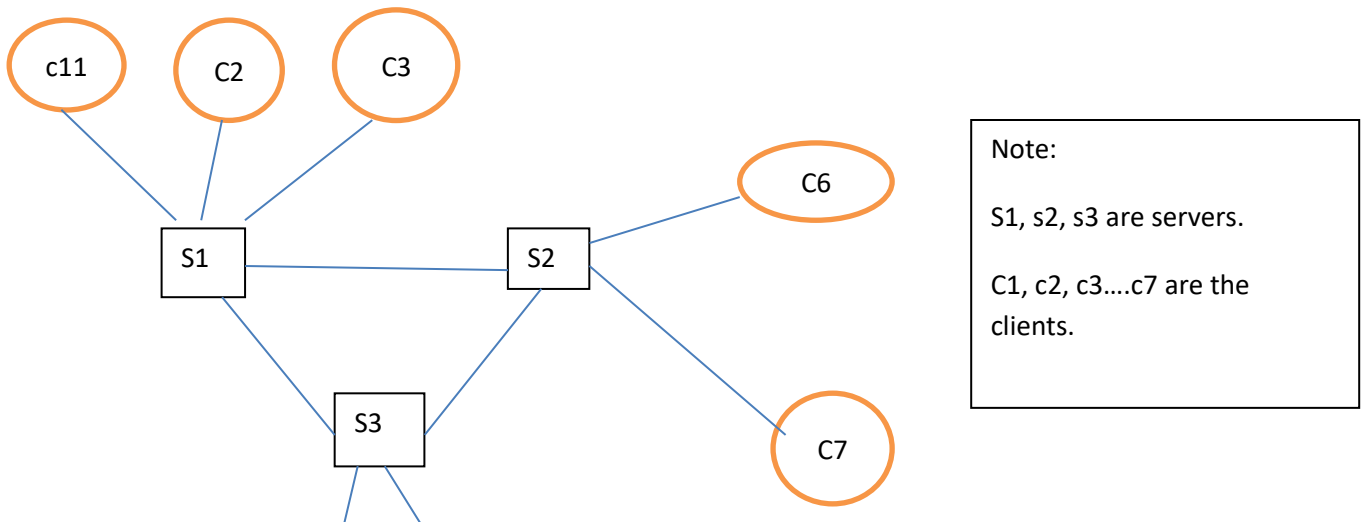


Fig: client-server architecture

There are different application layers in client-server architecture:

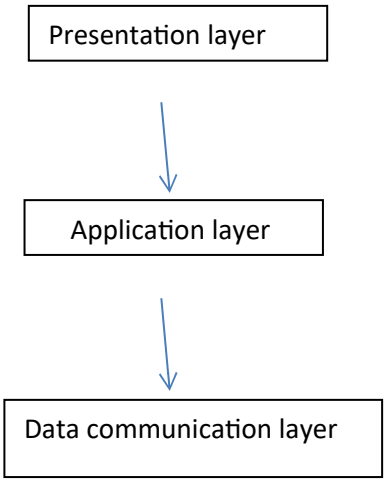


Fig: layer of application

- Presentation layer is concerned with presenting the result of a computation to system users and with collecting user inputs.
- Application layer is concerned with providing application specific functionality.
- Data management layer concerned with managing the system.

3.8 Distributed object architecture:

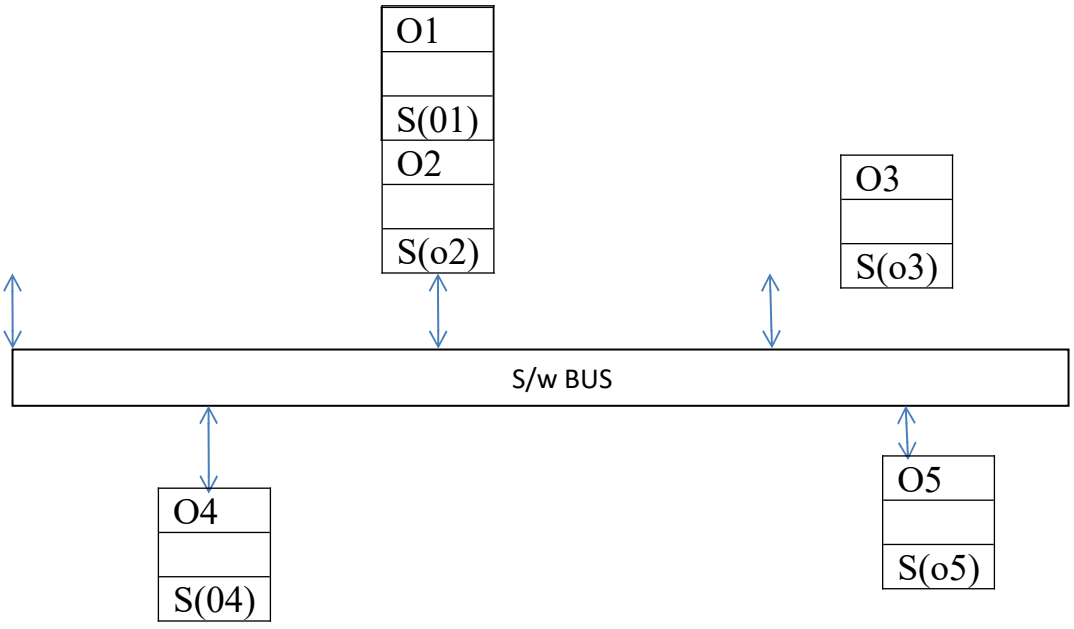


Fig: Distributed object architecture.

Features:

- No distinction
- Each distributed entity is objects that provide service other object and receive from other objects.
- Object communication is through middleware system called an object request booker or software bus.
- However, more complexes to design the client server architecture.

Advantages:

- It allows system designer to delay decision on where and how services should be provided.
- It is very open system architecture that allows to new resources to be as required.
- System is flexible and scalable.

3.9 Inter-organizational distributed computing

Features:

- Used for security and interoperability reason.
- Local standards management and operational processes apply for such inter-organizational distributed computing.

- Newer model of distributed computing have been designed to support inter-organizational computing where different node server are located at different organization.