INSTITUTO SUPERIOR DE TECNOLOGIAS AVANÇADAS DE LISBOA

Software Engineering

João Caldeira

Invited Professor

Email. joaocarlos.caldeira@my.istec.pt

Mob. +351 917769544

December 14, 2022

INSTITUTO SUPERIOR DE TECNOLOGIAS AVANÇADAS DE LISBOA

Software Design

December 14, 2022

Software Design

Architectural Design

- **Definition**

1. Architectural design is concerned with understanding how a software system should be organized and designing the overall structure of that system

2. It is the critical link between design and requirements engineering, as it identifies the main structural components in a system and the relationships between them

3. The output of the architectural design process is an architectural model that describes how the system is organized as a set of communicating components

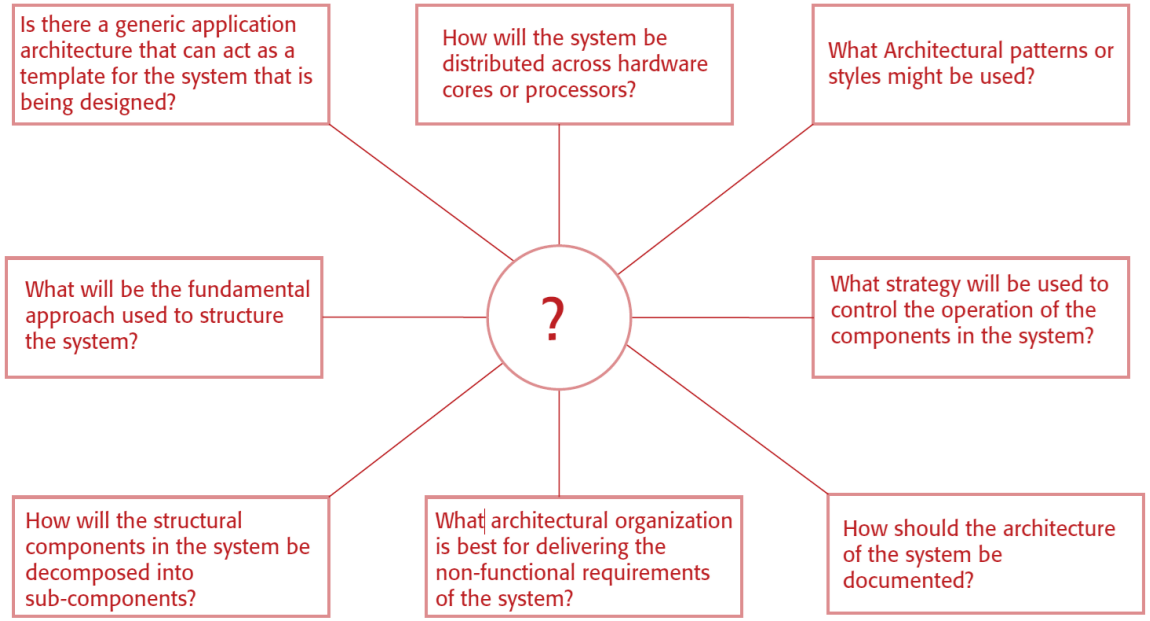
4. In agile processes, it is generally accepted that an early stage of an agile development process should focus on designing an overall system architecture

Software Design

Architectural Design Decisions

- **Concerns**

1. **Performance.** The architecture should be designed to localize critical operations within a small number of components
2. **Security.** A layered structure for the architecture should be used, with the most critical assets protected in the innermost layers
3. **Safety.** The architecture should be designed so that safety-related operations are co-located in a single or in a small number of components
4. **Availability.** The architecture should be designed to include redundant components so that it is possible to replace and update components without stopping the system



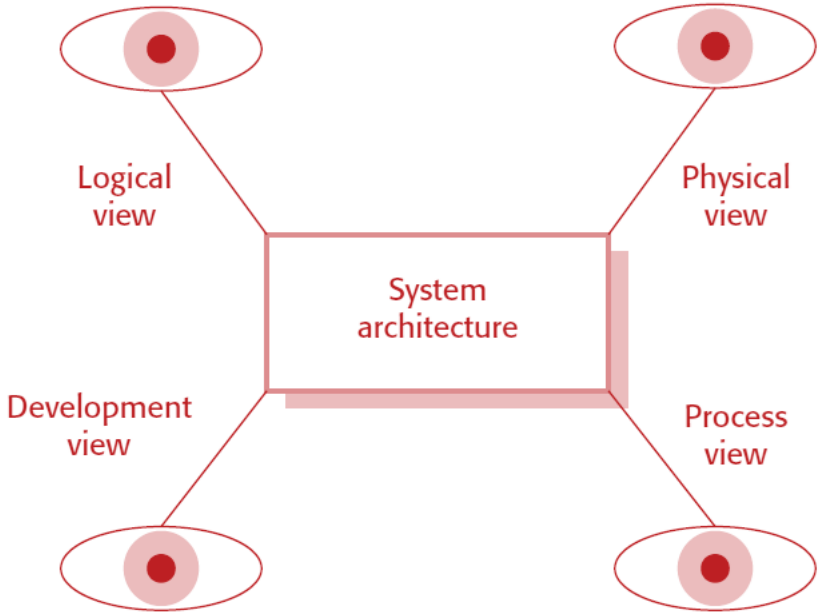
Software Design

Architectural Views

- **Options**

1. **Logical View.** Shows the key abstractions in the system as objects or object classes. **Useful to relate the system requirements to entities in this logical view**

2. **Process View.** Shows how, at runtime, the system is composed of interacting processes. **Useful for making judgments about non-functional system characteristics such as performance and availability**



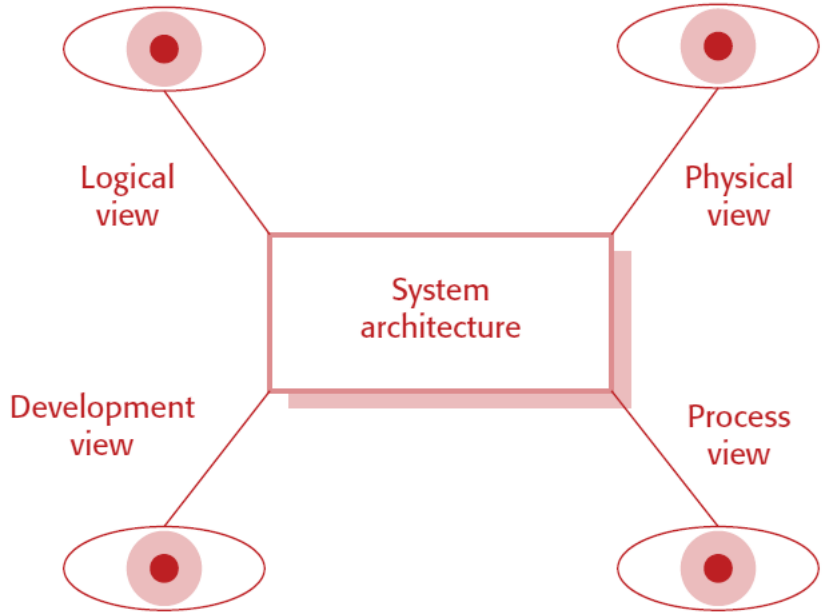
Software Design

Architectural Views

- **Options**

3. **Development View.** Shows how the software is decomposed for development; that is, it shows the breakdown of the software into components that are implemented by a single developer or development team. **Useful for software managers and programmers**

4. **Physical View.** Shows the system hardware and how software components are distributed across the processors in the system. **Useful for systems engineers planning a system deployment**



Software Design

Architectural Patterns

- The idea of patterns as a way of presenting, sharing, and reusing knowledge about software systems has been adopted in a number of areas of software engineering
  - **Architectural Pattern.** A stylized, abstract description of good practice, which has been tried and tested in different systems and environments
    - Should describe a system organization that has been successful in previous systems
    - It should include information on when it is and is not appropriate to use that pattern, and details on the pattern's strengths and weaknesses

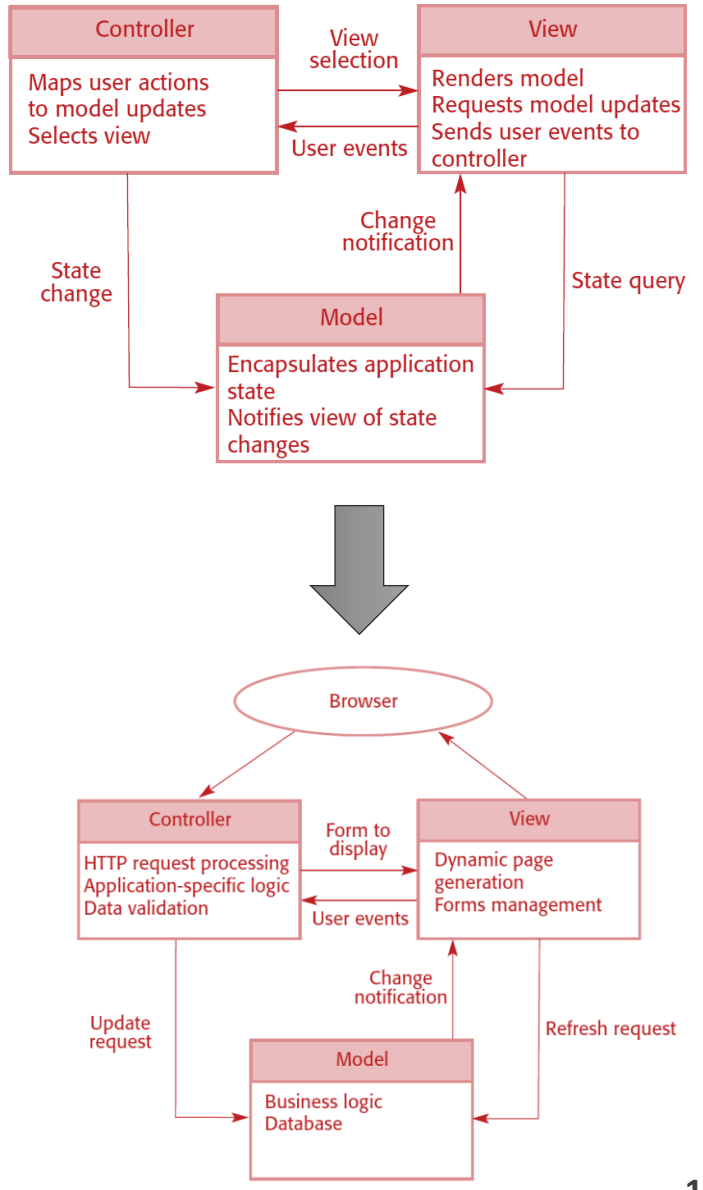
Name	MVC (Model-View-Controller)
Description	Separates presentation and interaction from the system data. The system is structured into three logical components that interact with each other. The Model component manages the system data and associated operations on that data. The View component defines and manages how the data is presented to the user. The Controller component manages user interaction (e.g., key presses, mouse clicks, etc.) and passes these interactions to the View and the Model. See Figure 6.5.
Example	Figure 6.6 shows the architecture of a web-based application system organized using the MVC pattern.
When used	Used when there are multiple ways to view and interact with data. Also used when the future requirements for interaction and presentation of data are unknown.
Advantages	Allows the data to change independently of its representation and vice versa. Supports presentation of the same data in different ways, with changes made in one representation shown in all of them.
Disadvantages	May involve additional code and code complexity when the data model and interactions are simple.

Software Design

Architectural Patterns

Model-View-Controller (MVC) Pattern

- This pattern is the basis of interaction management in many web-based systems and is supported by most language frameworks
- Separates elements of a system, allowing them to change independently.
  - Eg: Adding a new view or changing an existing view can be done without any changes to the underlying data in the model



Software Design

Architectural Patterns

Layered Architecture Pattern

- The notions of separation and independence are fundamental to architectural design because they allow changes to be localized

- The system functionality is organized into separate layers, and each layer only relies on the facilities and services offered by the layer immediately beneath it

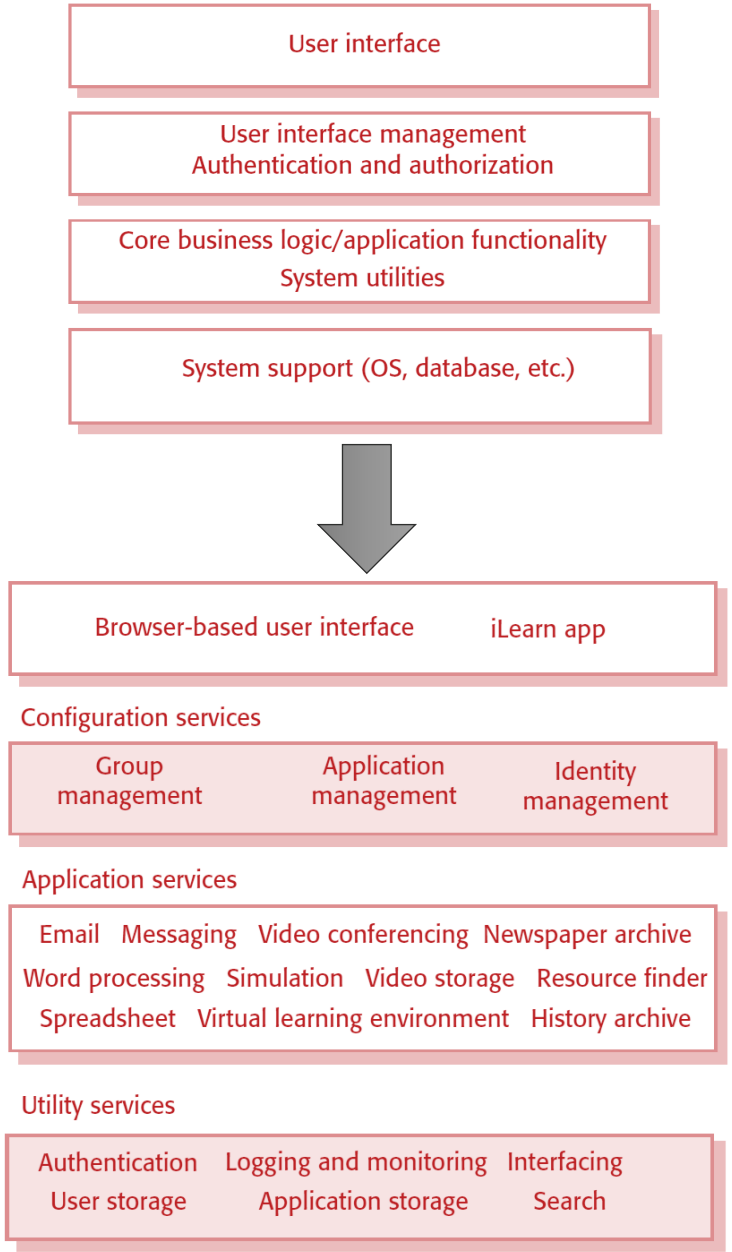
Name	Layered architecture
Description	Organizes the system into layers, with related functionality associated with each layer. A layer provides services to the layer above it, so the lowest level layers represent core services that are likely to be used throughout the system. See Figure 6.8.
Example	A layered model of a digital learning system to support learning of all subjects in schools (Figure 6.9).
When used	Used when building new facilities on top of existing systems; when the development is spread across several teams with each team responsibility for a layer of functionality; when there is a requirement for multilevel security.
Advantages	Allows replacement of entire layers as long as the interface is maintained. Redundant facilities (e.g., authentication) can be provided in each layer to increase the dependability of the system.
Disadvantages	In practice, providing a clean separation between layers is often difficult, and a high-level layer may have to interact directly with lower-level layers rather than through the layer immediately below it. Performance can be a problem because of multiple levels of interpretation of a service request as it is processed at each layer.

Software Design

Architectural Patterns

Layered Architecture Pattern

- This layered approach supports the incremental development of systems
- As a layer is developed, some of the services provided by that layer may be made available to users
- The architecture is also changeable and portable
- If its interface is unchanged, a new layer with extended functionality can replace an existing layer



Software Design

Architectural Patterns

Client-Server Architecture Pattern

Name	Client–server
Description	In a client–server architecture, the system is presented as a set of services, with each service delivered by a separate server. Clients are users of these services and access servers to make use of them.
Example	Figure 6.13 is an example of a film and video/DVD library organized as a client–server system.
When used	Used when data in a shared database has to be accessed from a range of locations. Because servers can be replicated, may also be used when the load on a system is variable.
Advantages	The principal advantage of this model is that servers can be distributed across a network. General functionality (e.g., a printing service) can be available to all clients and does not need to be implemented by all services.
Disadvantages	Each service is a single point of failure and so is susceptible to denial-of-service attacks or server failure. Performance may be unpredictable because it depends on the network as well as the system. Management problems may arise if servers are owned by different organizations.

- Organized as a set of services and associated servers, and clients that access/use the services

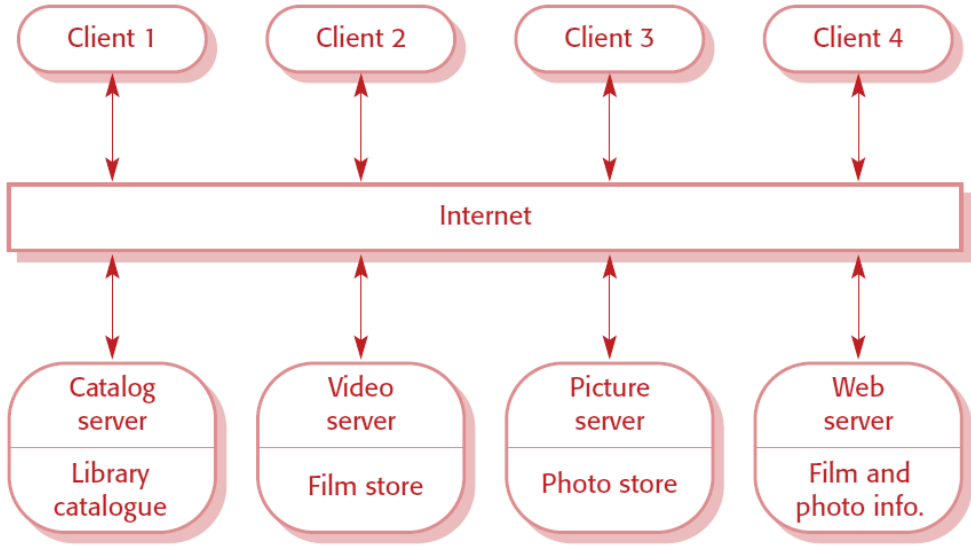
1. **A set of servers.** That offer services to other components.
   - **E.g.** Include print servers that offer printing services, file servers that offer file management services, and a compile server that offers programming language compilation services. **Servers are software components, and several servers may run on the same computer.**

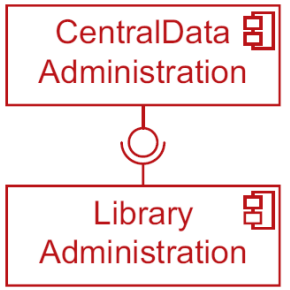
2. **A set of clients.** That call on the services offered by servers. There will normally be several instances of a client program executing concurrently on different computers

3. **A network** that allows the clients to access these services. **Client–server systems** are usually implemented as distributed systems, connected using Internet protocols

Client-Server Architecture Pattern

- The most important advantage of the client–server model is that it is a distributed architecture
- Effective use can be made of networked systems with many distributed processors
- It is easy to add a new server and integrate it with the rest of the system or to upgrade servers transparently without affecting other parts of the system

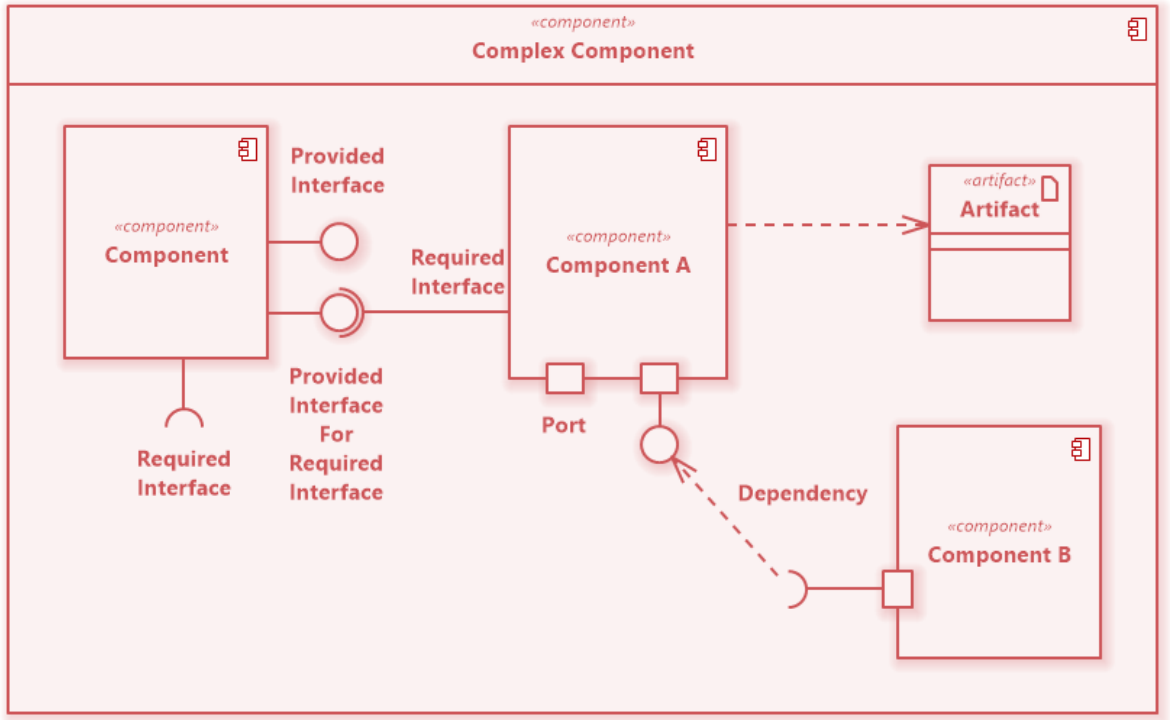




- **Components (Diagrams)**

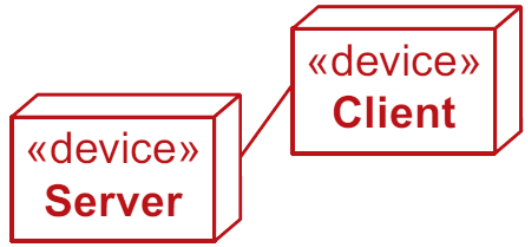
1. A component is an independent, executable unit that provides other components with services or uses the services of other components

2. When specifying a component, you can model two views explicitly:
   - **The external view.** Which represents the specification of the component
   - **The internal view.** Which defines the implementation of the component



Software Design

Structural Models



- **Deployment (Diagrams)**

1. Represents the hardware topology used and the runtime system assigned
2. The hardware encompasses processing units in the form of nodes as well as communication relationships between the nodes
3. A runtime system contains artifacts that are deployed to the nodes