CSE 470 - Software Architecture

BRAC University

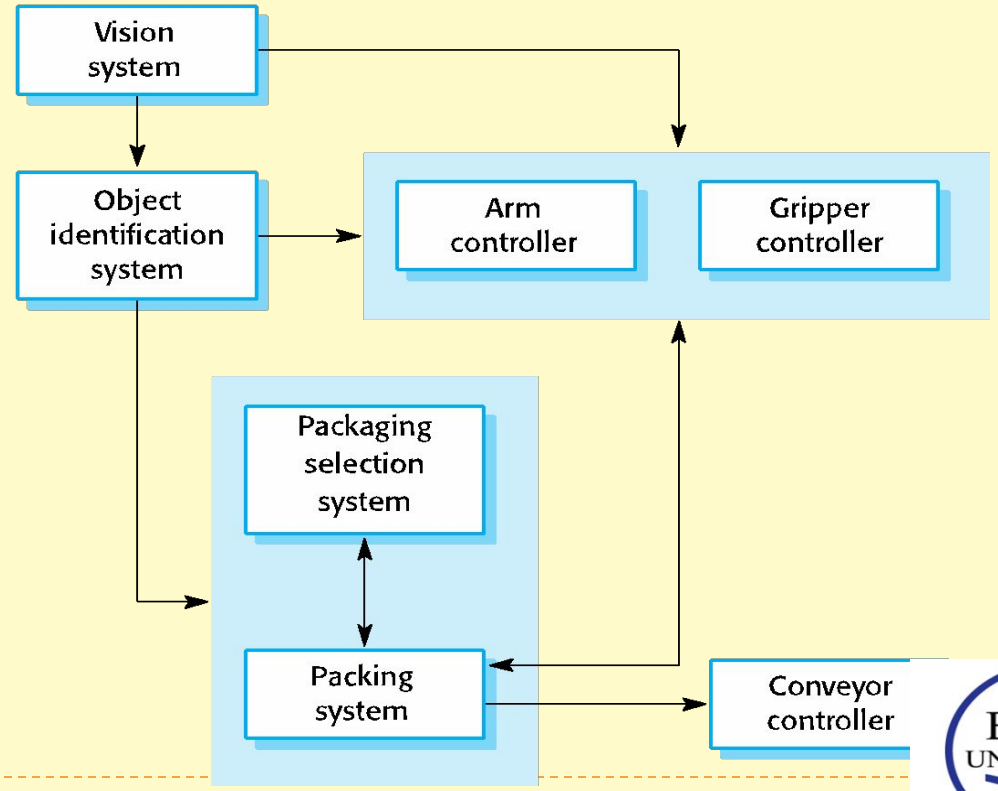
BRAC UNIVERSITY
Inspiring Excellence

Software Architecture

- Architectural design is concerned with understanding how a software system should be organized and designing the overall structure of that system.
- Software Architecture is how the defining components of a software system are **organized and assembled.** How they **communicate** each other. And how **the constraints of the whole system is ruled** by.
- **The architectural mod**
  the development phas

Architectural Representation

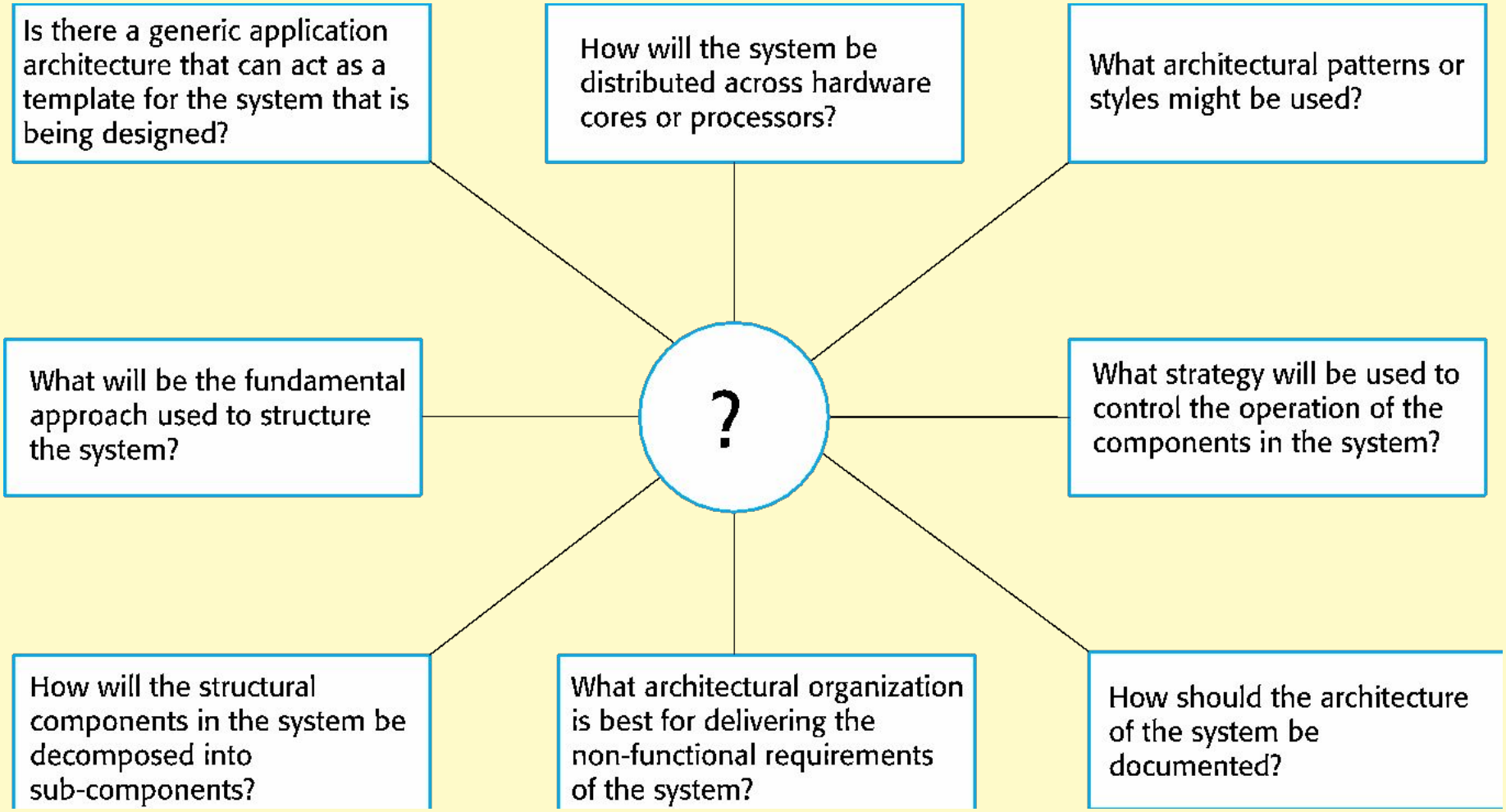
- Simple, informal block diagrams showing entities and relationships are the most frequently used method for documenting software architectures.



Advantages of explicit architecture

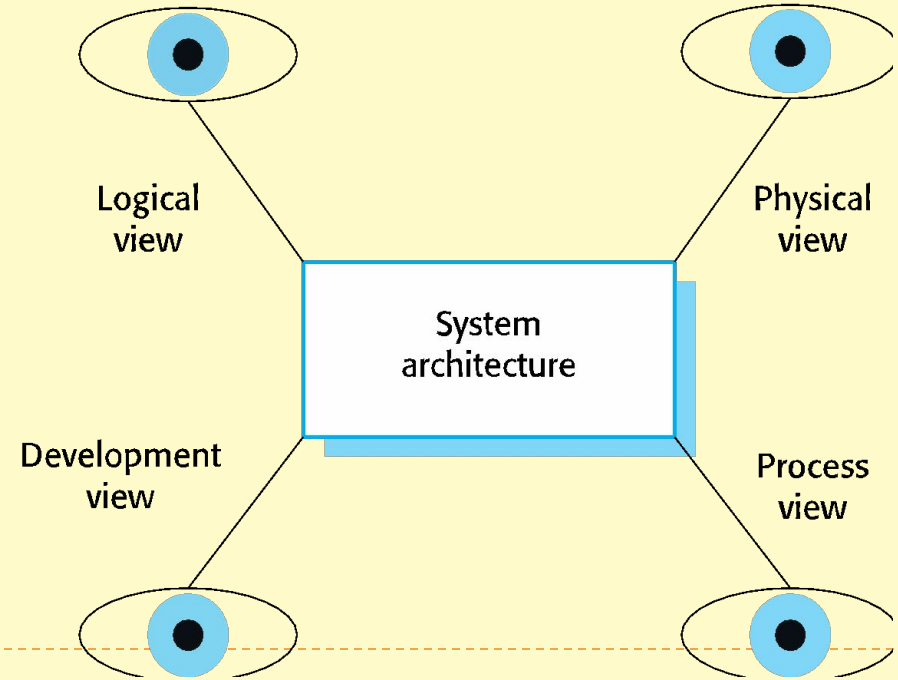
- Stakeholder communication
  - Architecture may be used as a focus of discussion by system stakeholders.
- System analysis
  - Means that analysis of whether the system can meet its non-functional requirements is possible.
- Large-scale reuse
  - The architecture may be reusable across a range of systems
  - Product-line architectures may be developed. (Softwares with common features like e-commerce)

Architecture Design Decisions



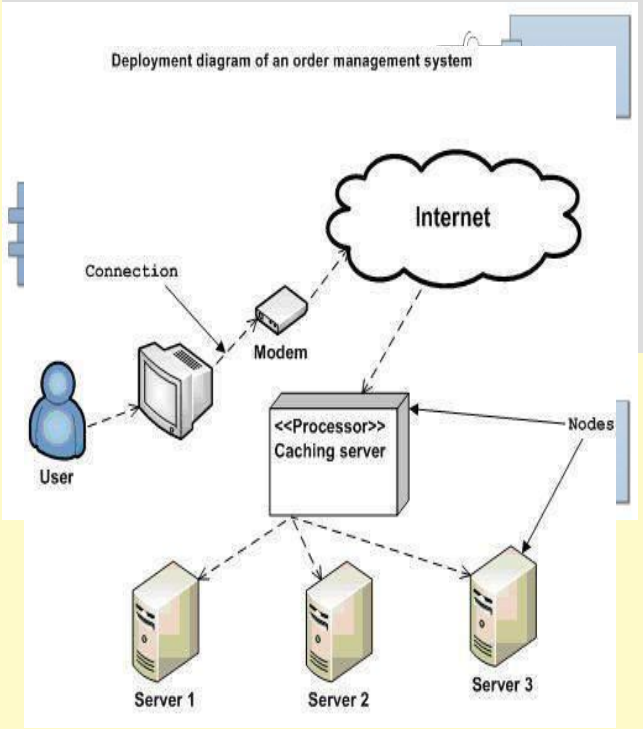
Architectural View

- There are four views from which the architecture of a software can be observed
- Each architectural diagram only shows one view or perspective of the system.

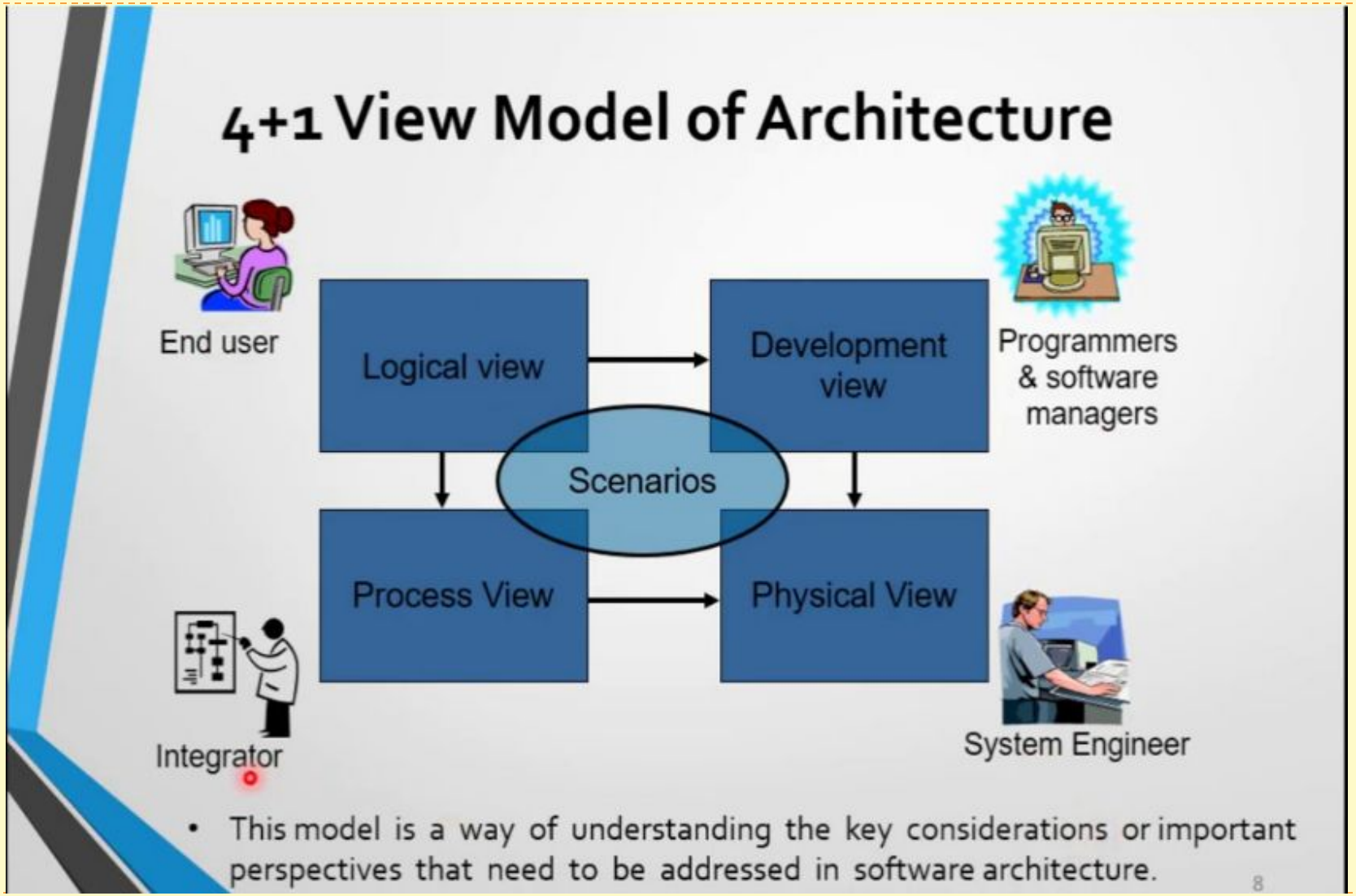


4 + 1 view model of software architecture

- A logical view, which shows the key abstractions in the system as objects or object classes. (class/state diagrams)
- A process view, which shows how, at run-time, the system is composed of interacting processes. (activity diagram)
- A development view, which shows how the software is decomposed for development. (Component/package diagram)
- A physical view, which shows the system hardware and how software components are distributed across the processors in the system. (Deployment diagram.)
- Related using use cases or scenarios (+1)



4 + 1 view model of software architecture



Architectural Pattern

- Let's think of some problems -

BRAC UNIVERSITY
Inspiring Excellence

Architectural Pattern

- An architectural pattern is a stylized description of good design practice, which has been tried and tested in different environments.
- It's a solution to a existing and commonly occurring problem.
- Patterns include information about when they are and when the are not useful.

MVC Pattern

- MVC goes for Model-View-Controller Pattern
- Separates presentation and interaction from the data handling logic.
- The system is structured into three logical components that interact with each other- Model, View and Controller

BRAC UNIVERSITY
Inspiring Excellence

Controller

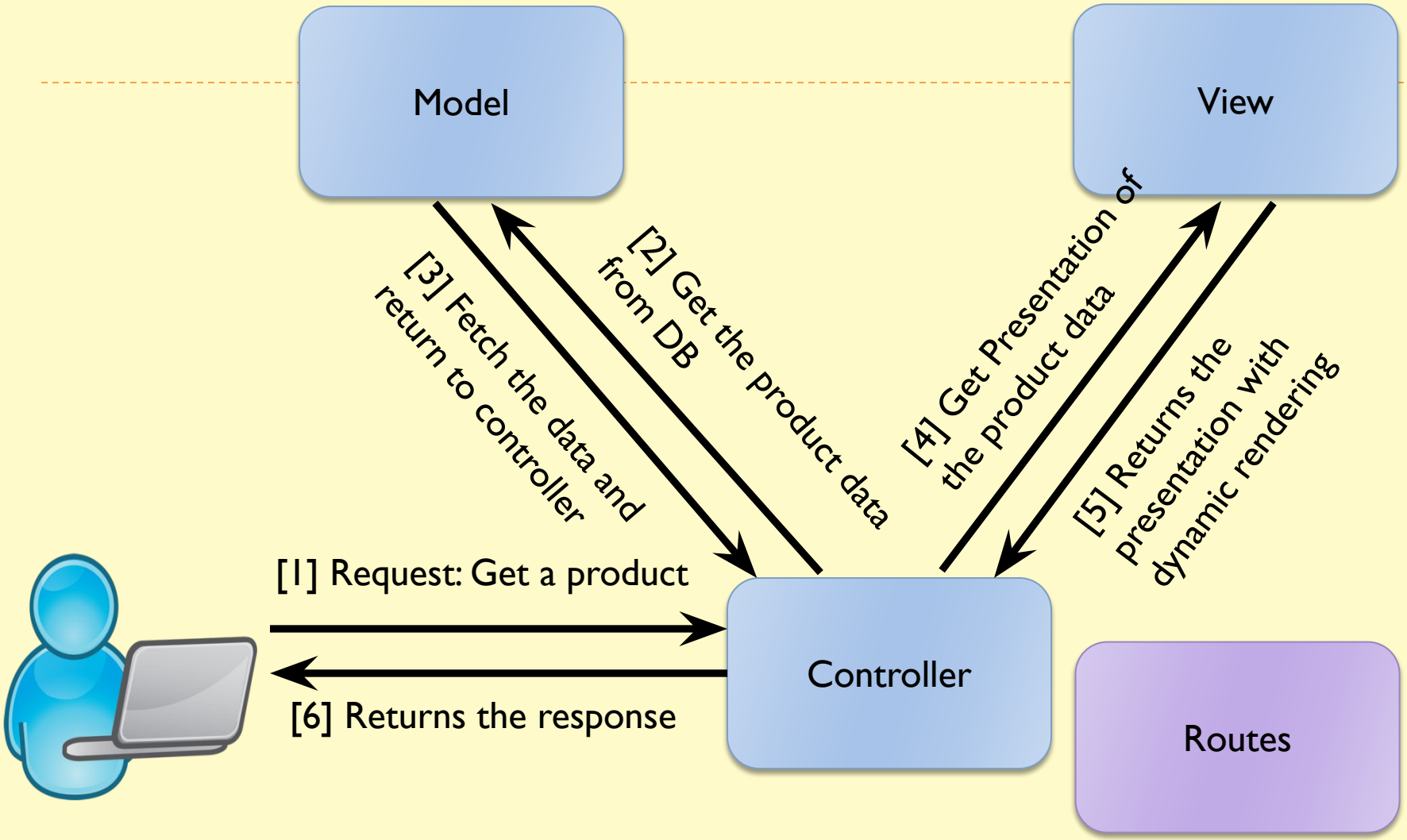
- Handles all the user inputs through url
- The interaction to an application starts here by the user interactions – mouse click, key press etc
- Process http url requests (*GET, POST, PUT, DELETE*)
  - *GET*: for getting a data
  - *POST*: for posting / inserting a data
  - *PUT*: for updating a data
  - *DELETE*: for removing a data
- Communicates with both Model and View
- Contains all server side logic
- In the example – ProductController, UserController, AccountController etc.

Model

- It refers to the Data Related Logic
- Interaction with database (such as *SELECT, INSERT, UPDATE, DELETE*)
- It communicates with the controllers
- Can sometimes update or collaborate with the view (Depends on framework)
- In the example – Product, User, Transaction, Cart etc are model classes.

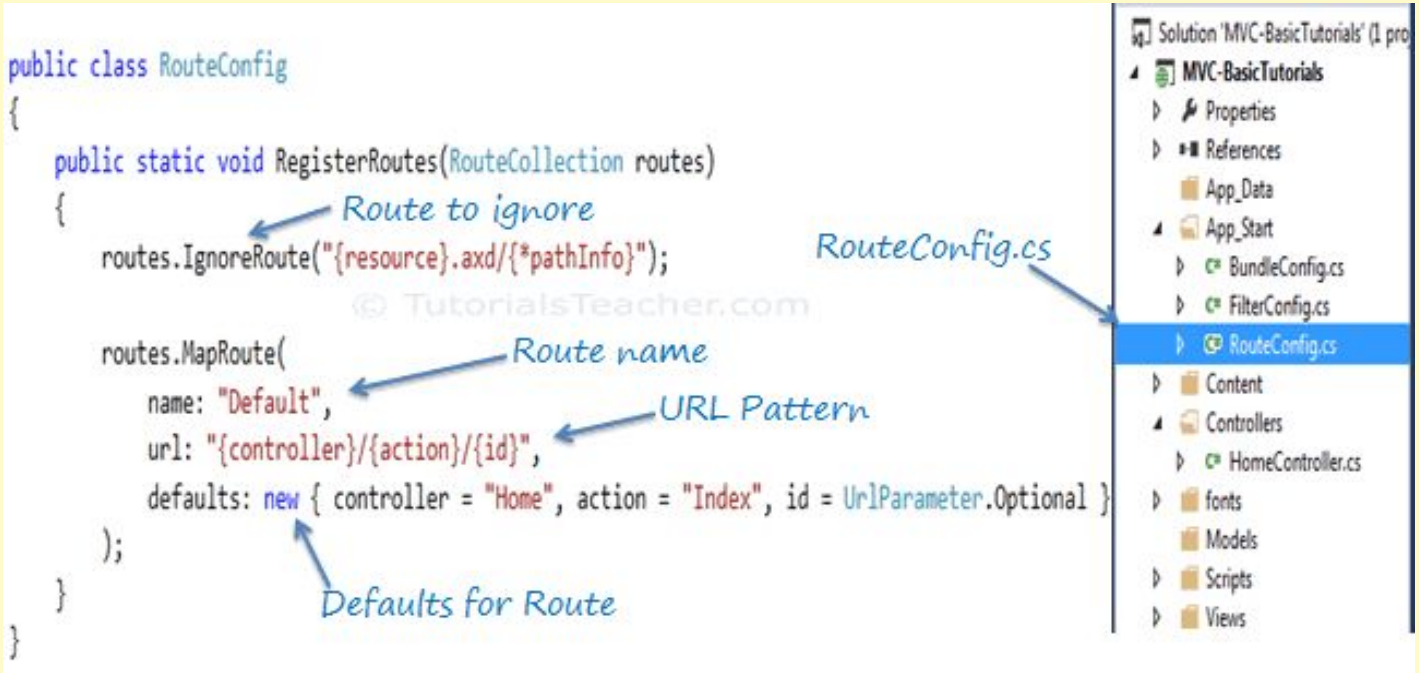
View

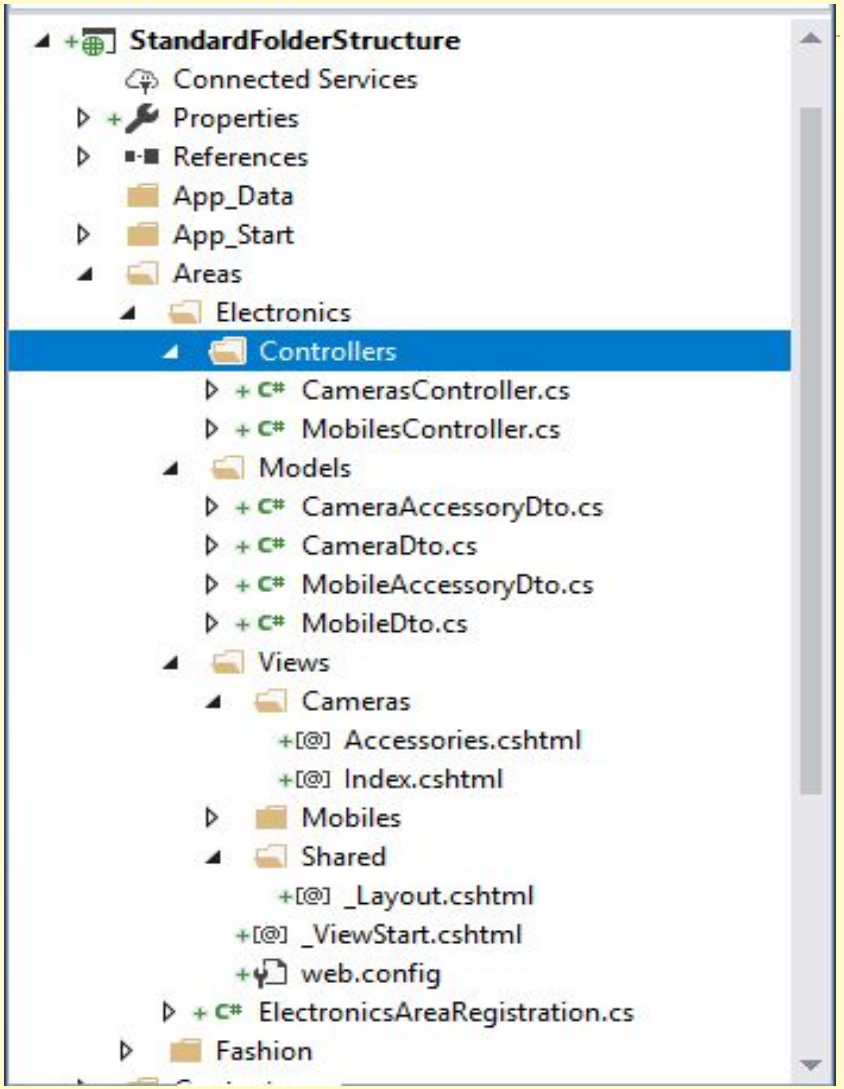
- What the end users see (UI)
- Usually Consists of *html/css*
- Communicates with the controller
- While coding are passed as dynamic values from the controller
- A controller can have multiple associated views.
- In the example – product.html, user.html to view a html file



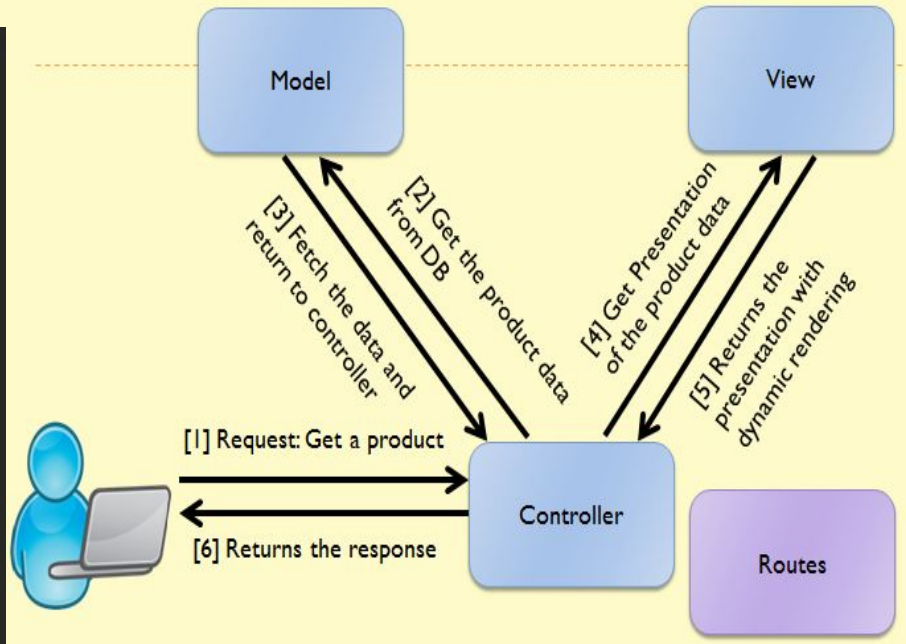
Routes

- Are urls to access a resource kept under App_Start folder
- General structure - http://domainName/{controller}/{action}/{id}
- http://YourApp.com/Users/Profile/25





```
http://yourapp.com/users/profile/1

/routes
    users/profile/:id = Users.getProfile(id)

/controllers
    class Users{
        function getProfile(id){
            profile = this.UserModel.getProfile(id)

            renderView('users/profile', profile)
    }

/models
    Class UserModel{
        function getProfile(id){
            data = this.db.get('SELECT * FROM users WHERE id = id')
            return data;
    }

/views
    /users
        /profile
        <h1>{{profile.name}}</h1>
        <ul>
            <li>Email: {{profile.email}}</li>
            <li>Phone: {{profile.phone}}</li>
```



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. Other advantages are: parallel working, improved testability, code reusability

Disadvantages:

Can involve additional code and code complexity when the data model and interactions are simple.

Summary

- MVC is one of the most used architectural patterns.
- It divides the system into three parts – model, view and controller
- It provides a extensible separation of concern (SOC) from presentation view to data processing logic.