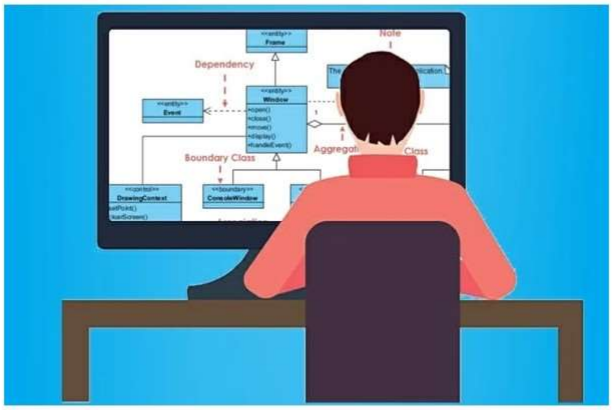
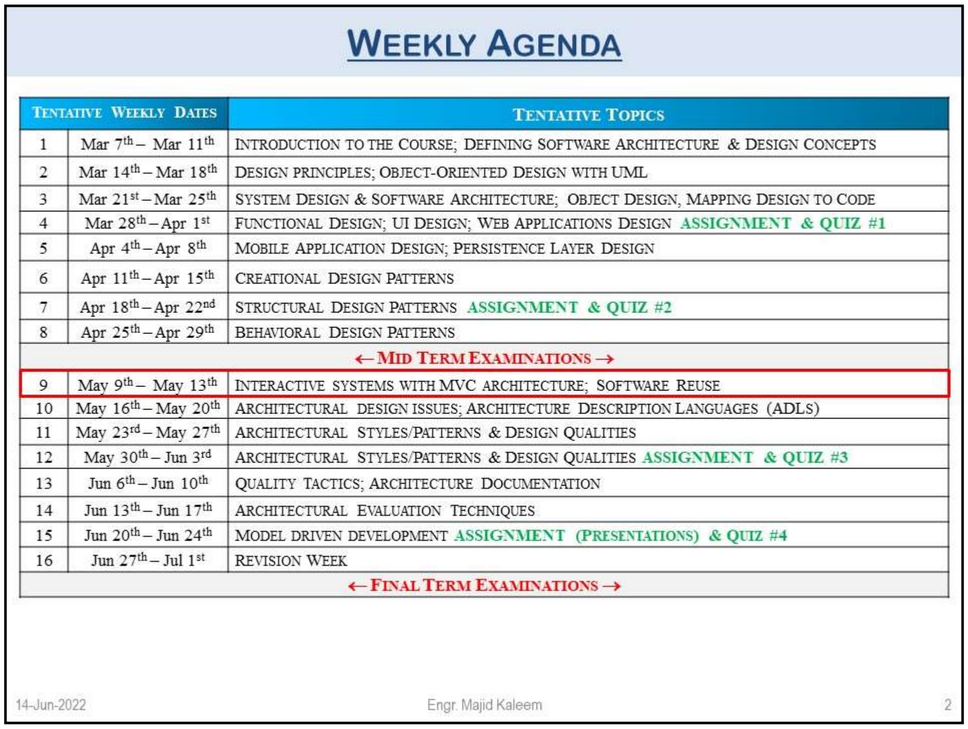
Software Design & Architecture
Spring 2022 - Week-09



مدرس: مہندس ماجد کلیم

جامعہ بحریہ، واقعگاہ کراچی

Engr. Majid Kaleem



Weekly Agenda

Tentative Weekly Dates		Tentative Topics
1	Mar 7th – Mar 11th	Introduction to the Course; Defining Software Architecture & Design Concepts
2	Mar 14th – Mar 18th	Design principles; Object-Oriented Design with UML
3	Mar 21st – Mar 25th	System Design & Software Architecture; Object Design, Mapping Design to Code
4	Mar 28th – Apr 1st	Functional Design; UI Design; Web Applications Design **Assignment & Quiz #1**
5	Apr 4th – Apr 8th	Mobile Application Design; Persistence Layer Design
6	Apr 11th – Apr 15th	Creational Design Patterns
7	Apr 18th – Apr 22nd	Structural Design Patterns **Assignment & Quiz #2**
8	Apr 25th – Apr 29th	Behavioral Design Patterns
		← Mid Term Examinations →
9	May 9th – May 13th	Interactive systems with MVC architecture; Software Reuse
10	May 16th – May 20th	Architectural design issues; Architecture Description Languages (ADLs)
11	May 23rd – May 27th	Architectural Styles/Patterns & Design Qualities
12	May 30th – Jun 3rd	Architectural Styles/Patterns & Design Qualities **Assignment & Quiz #3**
13	Jun 6th – Jun 10th	Quality Tactics; Architecture Documentation
14	Jun 13th – Jun 17th	Architectural Evaluation Techniques
15	Jun 20th – Jun 24th	Model driven development **Assignment (Presentations) & Quiz #4**
16	Jun 27th – Jul 1st	Revision Week
		← Final Term Examinations →

14-Jun-2022 Engr. Majid Kaleem 2

WHAT ARE INTERACTIVE SYSTEMS?

- Interactive systems are simply the computer systems that support interaction between humans and machines (human machine interaction) by showing specific response to specific human actions.

INTERACTIVE SYSTEMS AND THE MVC ARCHITECTURE

- Interaction-oriented architecture has two major styles:
  1. Model-View-Controller (MVC)
  2. Presentation-Abstraction-Control (PAC)

- Both **MVC** and **PAC** propose ***three*** components decomposition and are used for interactive applications such as web applications.

- They are different in their flow of control and organization.

- **PAC** is an agent-based hierarchical architecture but **MVC** does not have a clear hierarchical structure.

Interactive Systems and the MVC architecture

- The interaction-oriented software architecture decomposes the system into three major partitions:

1. **Data module** − Data module provides the data abstraction and all business logic.
2. **Control module** − Control module identifies the flow of control and system configuration actions.
3. **View presentation module** − View presentation module is responsible for visual or audio presentation of data output and it also provides an interface for user input.

MVC architecture

- MVC decomposes a given software application into three interconnected parts that help in separating the internal representations of information from the information presented to or accepted from the user.

Module	Function
Model	*Encapsulation the underlying data and business logic* *Classes that represent data of the application* *Use validation logic to enforce business rules on the data*
Controller	*Respond to user action and direct the application flow* *Classes that handle request to the application by the user* *Get data from database or objects* *Specify views t hat return a repose to the client*
View	*Formats and present the data from model to user* *Templates (presentation)file that your application uses to dynamically generate HTML*

USEFUL WEB RESOURCE

- https://thedotnetguide.com/mvc-design-pattern/
- **Self-Study:**
  1. **MVP** – **Model-View-Presenter**
  2. **MVVM** – **Model-View-View-Model**

MODEL

- Model is a central component of MVC that directly manages the ***data***, ***logic***, and ***constraints*** of an application.
- It consists of data components, which maintain the raw application data and application logic for interface.
- It is an independent user interface and captures the behavior of application problem domain.
- It is the domain-specific software simulation or implementation of the application's central structure.
- When there has been change in its state, it gives notification to its associated view to produce updated output and the controller to change the available set of commands.

MODEL

- They are responsible for processing the data, managing database connections, implementing business rules and querying database.
- It processes data and passes it on to the view with out worrying about the final cosmetic looks.
- Model gets requests from the controller and they notify the corresponding views regarding the data.
- *In Microsoft technologies they are .NET DLL while in Java they are Java beans.*

9

VIEW

- View can be used to represent any ***output*** of information in graphical form such as diagram or chart.
- It consists of ***presentation*** components which provide the visual representations of data
- Views request information from their model and generate an output representation to the user.
- Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants.

10

VIEW

- View represents the look and feel of an application; *in one line they represent the GUI of a system*.
- So view gets data and put in cosmetic formatting before displaying on the UI.
- *It can be HTML, JAVAApplets, Windows form, XSL etc.*

CONTROLLER

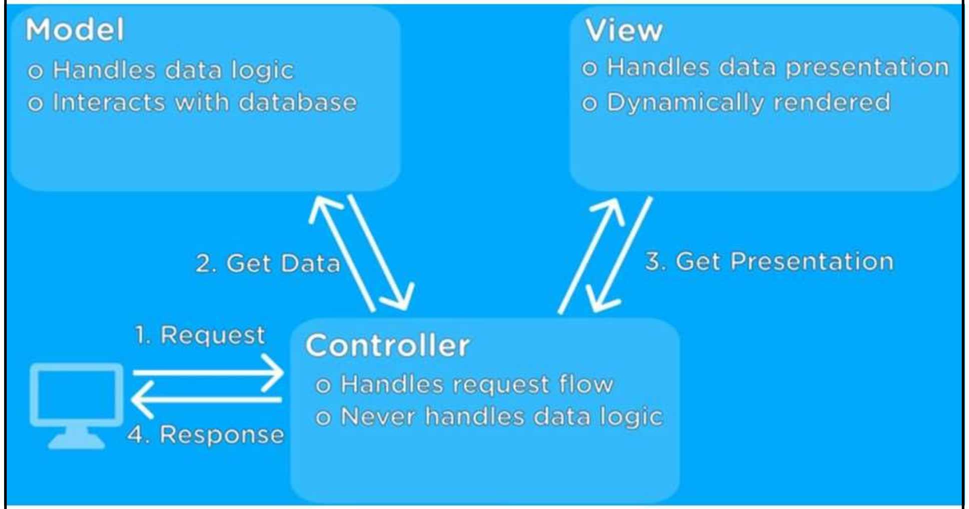
- A controller accepts an input and converts it to commands for the model or view.
- It consists of input processing components which handle input from the user by modifying the model.
- It acts as an interface between the associated models and views and the input devices.
- It can send commands to the model to update the model's state and to its associated view to change the view's presentation of the model.

CONTROLLER

- Controllers accept user events like mouse click, button enter etc and reacts accordingly.
- Controllers get these events from views and they trigger events to change the model to update their state.
- Once models have updated their states they communicate the same to the corresponding views to refresh the display.
- *In .NET technologies they behind code while in Java it's the Service method of the servlet.*

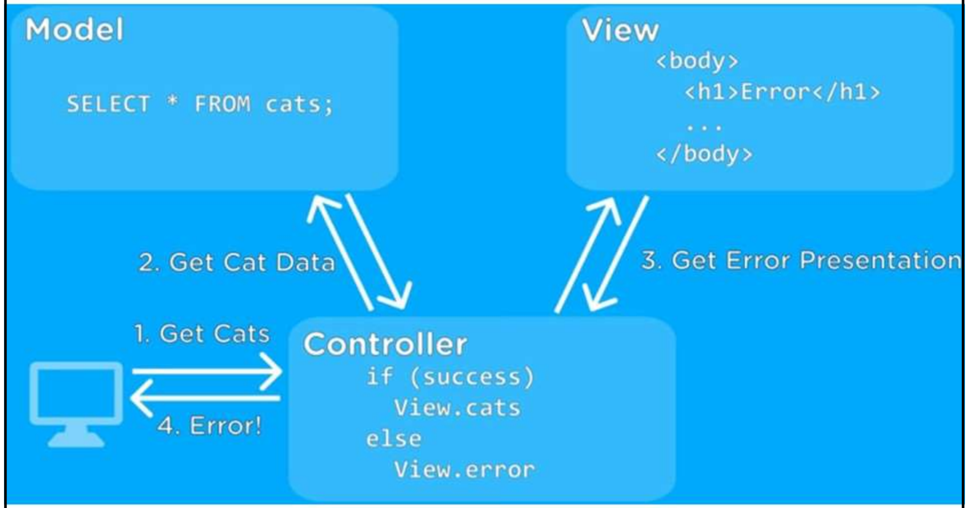
13

MVC



14-Jun-2022 Engr. Majid Kaleem 14

MVC

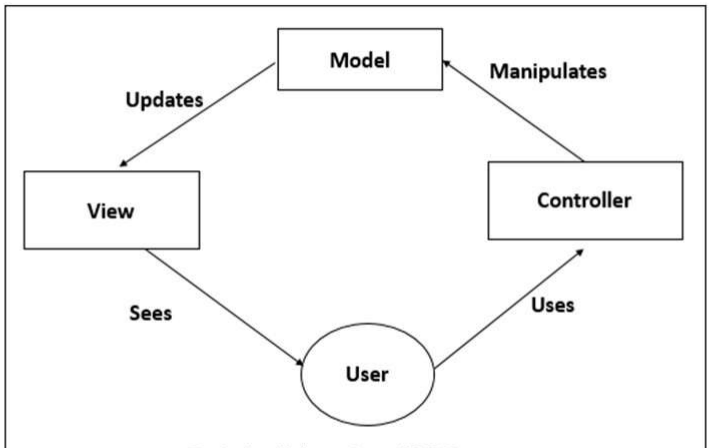


```
SELECT * FROM cats;
```

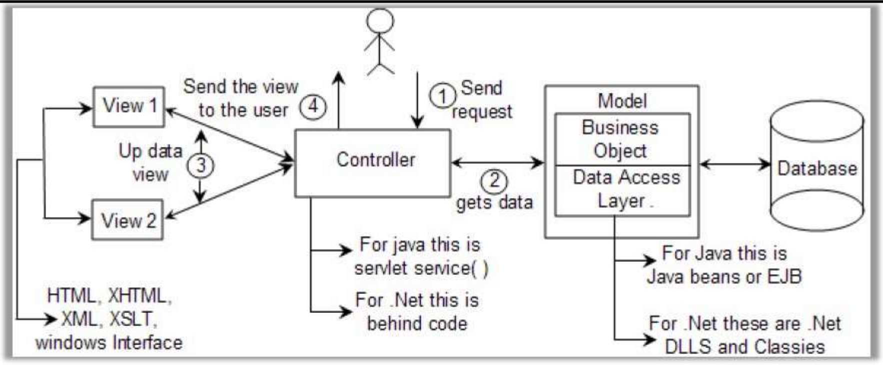
```
<body>
  <h1>Error</h1>
  ...
</body>
```

```
if (success)
  View.cats
else
  View.error
```

TYPICAL MVC



Typical collaboration of MVC component



View 1
View 2
Send the view to the user 4
Up data view 3
Controller
1 Send request
2 gets data
Model
Business Object
Data Access Layer .
Database
For java this is servlet service( )
For .Net this is behind code
For Java this is Java beans or EJB
For .Net these are .Net DLLS and Classies
HTML, XHTML, XML, XSLT, windows Interface

- So looking at the above figure we can say there are four main steps by which MVC works:-

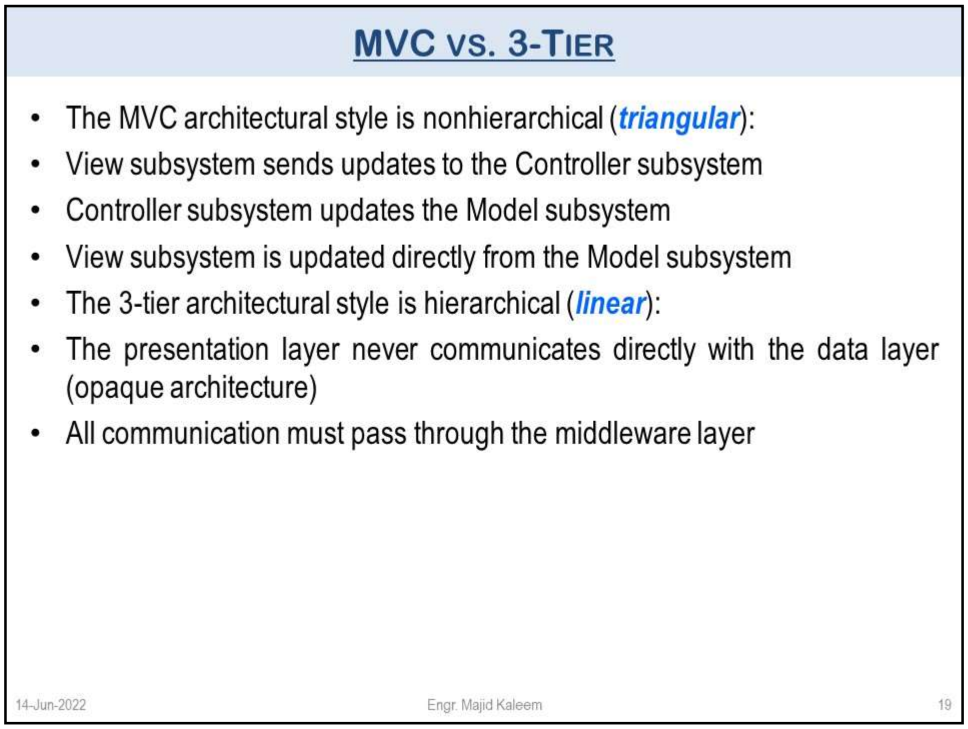
1. User sends an event like keyboard, button click or enter event to the controller.
2. Controller sends this event to the model who in turn updates himself and sends the data to the controller.
3. Controller updates the corresponding view depending on the event.
4. This view is then viewed by the end user.

17

MVC VS. 3-TIER

- **In MVC** : MVC architecture is ***triangular***: the view sends updates to the controller, the controller updates the model, and the view gets updated directly from the model
- **In Three Tier** : A three tier architecture is the client tier never communicates directly with the data tier
- In a Three-tier model all communication must pass through the middle tier
- 3 tier divides the whole app in: UI, logic and data
- MVC divides the UI part in: view (kind of UI of the UI), model (data) and controller (logic)

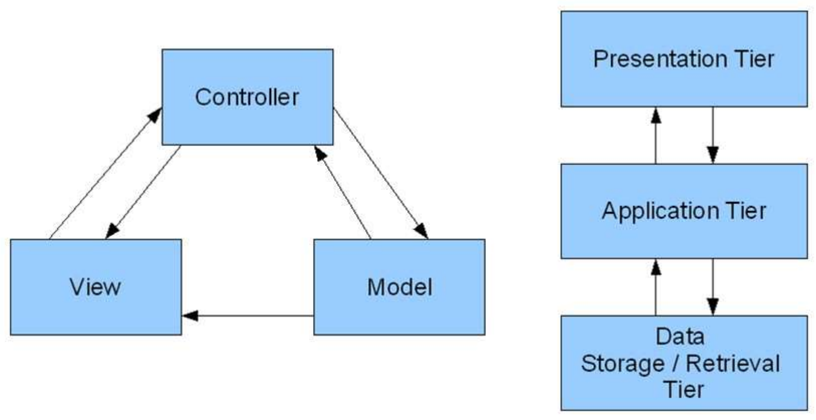
14-Jun-2022 Engr. Majid Kaleem 18



MVC VS. 3-TIER

- The MVC architectural style is nonhierarchical (***triangular***):
- View subsystem sends updates to the Controller subsystem
- Controller subsystem updates the Model subsystem
- View subsystem is updated directly from the Model subsystem
- The 3-tier architectural style is hierarchical (***linear***):
- The presentation layer never communicates directly with the data layer (opaque architecture)
- All communication must pass through the middleware layer

MVC VS. 3-TIER



Software Reuse

- Software reuse is the use of existing software or software knowledge to build new software for an individual or an organization.
- Software reuse is also called as "***Code Reuse***".
- Systematic software reuse is still the most promising strategy for increasing productivity and improving quality in the software industry.
- Example of software reuse is software ***library***.
- *Component-based architecture & web service-oriented architecture (SOA) supports software reusability.*

Purpose of Reuse

- CHEAPER PRODUCTS: It includes shorter development time ,easier maintenance
- BETTER QUALITY PRODUCTS: Code that was written for reuse should has better specifications and should be thoroughly tested.

Types of Reuse

- Concerning motivation and driving factors , reuse can be:
- Opportunistic - While getting ready to begin a project, the team realizes that there are existing components that they can be reused.
- Planned - A team strategically designs components so that they'll be reusable in future projects.

23

Types of Software Reuse

- Application System Reuse:
- It is concerned with reusing an entire application inside another. For example : MS-Office
- Component Reuse:
- It is concerned with components of one application reused in another application.

24

APPROACHES THAT SUPPORT REUSE

- Architectural patterns
- Design patterns
- Component-based development
- Application frameworks
- Legacy system wrapping
- Service-oriented systems
- ERP Systems.

25

ARCHITECTURAL PATTERNS

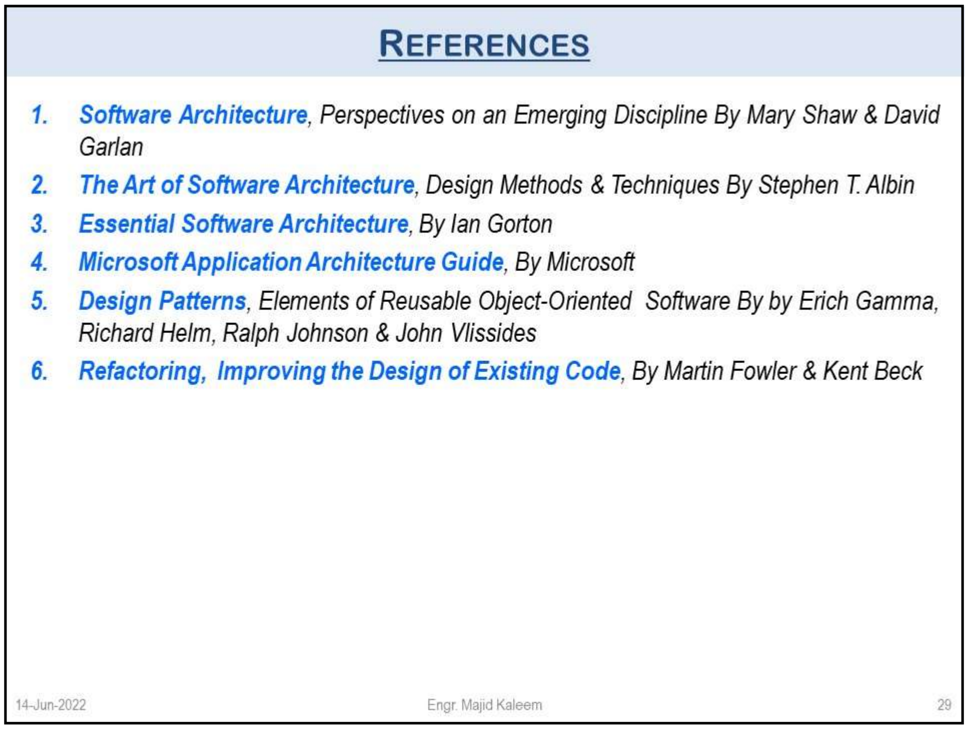
- An architectural pattern is a widely recognized and reused solution to a recurring design problem in the field of software architecture .
- The architectural patterns addresses various issues in software engineering such as computer hardware performance limitations, high availability and minimization of a business risks..

26

CONCLUSION

- A good software reuse process facilitates the increase of productivity, quality and reliability and decreases the costs and implementation time.
- By far the most important part of the reuse process is the people.
- If the people in the organization do not understand the concepts behind reuse and do not see the benefits, reuse won't happen
- Reuse processes and procedures must be incorporated into the existing software development process.

```
If(anyQuestions)
{
 askNow();
}
else
{
 thankYou();
 submitAttendance();
 endClass();
}
```



References

1. ***Software Architecture***, *Perspectives on an Emerging Discipline By Mary Shaw & David Garlan*
2. ***The Art of Software Architecture***, *Design Methods & Techniques By Stephen T. Albin*
3. ***Essential Software Architecture***, *By Ian Gorton*
4. ***Microsoft Application Architecture Guide***, *By Microsoft*
5. ***Design Patterns***, *Elements of Reusable Object-Oriented Software By by Erich Gamma, Richard Helm, Ralph Johnson & John Vlissides*
6. ***Refactoring, Improving the Design of Existing Code***, *By Martin Fowler & Kent Beck*