



**BITS Pilani**

# **Software Architecture**

## **Module 5**

### **Ensuring conformance to architecture**

Harvinder S Jabbal  
SEZG651/SSZG653 Software Architectures

# Code may drift away from Architecture



Examples of drift:

- Not sticking to the discipline of layers – an object in one layer calling an object located in a layer beyond the adjacent layer
- Accessing data base directly without going through data access layer
- Notifying different modules one by one instead of using ‘publish – subscribe’ model,
- Not making use of a common logging mechanism, etc.

What other architecture violations have you come across?

# Techniques to keep the code and architecture consistent

---



- Embed design concepts in the code (Architecturally evident coding style)
- Use frameworks
- Use code templates
- Update architecture documentation

# Embed design in code



Follow 'Architecturally Evident Coding Style'

Indicate in the code, aspects of architecture being implemented. Example:

- If we are using a layered architecture, indicate to which layer the code belongs
- If we are using 'Publish – Subscribe' pattern, indicate in the code whether it belongs to Publisher or Subscriber
- If we are using Message queues (MQ) for communication between components, indicate what the component is doing - inserting message or retrieving message from the MQ.

# Use Frameworks



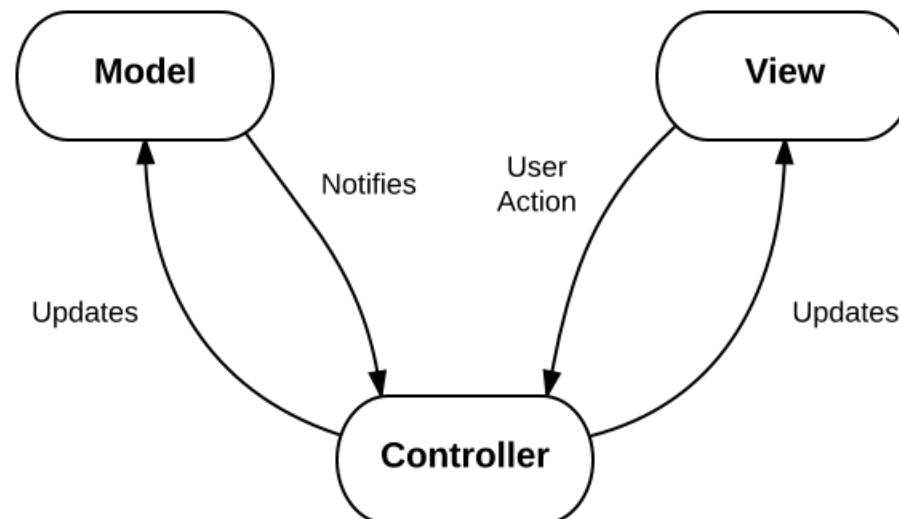
## Examples of frameworks:

- Spring
- Hibernate
- Autosar - AUTomotive Open System Architecture

# Spring MVC Framework



- Supports Model-View-Controller framework
- Here code belongs to one of the 3 component types - Model, View or Controller
  - **Model** encapsulates the application data
  - **View** renders screens on browser
  - **Controller** processes user requests, interacts with Model components and passes information to View components for rendering



# Experience sharing



Can you give examples of other frameworks that you have used?

# Other examples of frameworks

---



- Publish – Subscribe framework (JMS)
- Workflow framework of Salesforce.com
- Rules engine (DROOLS)
- Logging framework - Log4J (for logging events)



# Use Code templates



- A template provides a structure for developers to code
- For example a template for developing a code that needs to be fault tolerant will have the following sections in the template

Get event

Case (Event type)

**Normal:** // received by primary process

Process X; Send state to backup process;

Process Y; Send state to backup process;

**Update State data:** // received by Backup process

Update state data;

**Switch over:** // received by Backup process

Notify clients about change in Primary process;

End case;

Code needs to be accommodated in this template

# Update architecture documentation



- Changes to code should be accompanied with changes to architecture document when applicable
- At least mark parts of the arch document that is no longer applicable as 'No longer applicable'. This increases trust on remainder of the document
- Technique: At release time, synchronize the arch doc with code.

# Exercise



- What other techniques can we think of to ensure that code does not drift away from architecture?

# Exercise



What other techniques can we think of to ensure that code does not drift away from architecture?

- Educate new team members about the architecture
- Do Code reviews
- Create folders for each architectural aspect such as layer, service, UI, external interfaces, etc. and follow a discipline of storing the code in respective folders

# Appendix