Logical Architecture Refinement

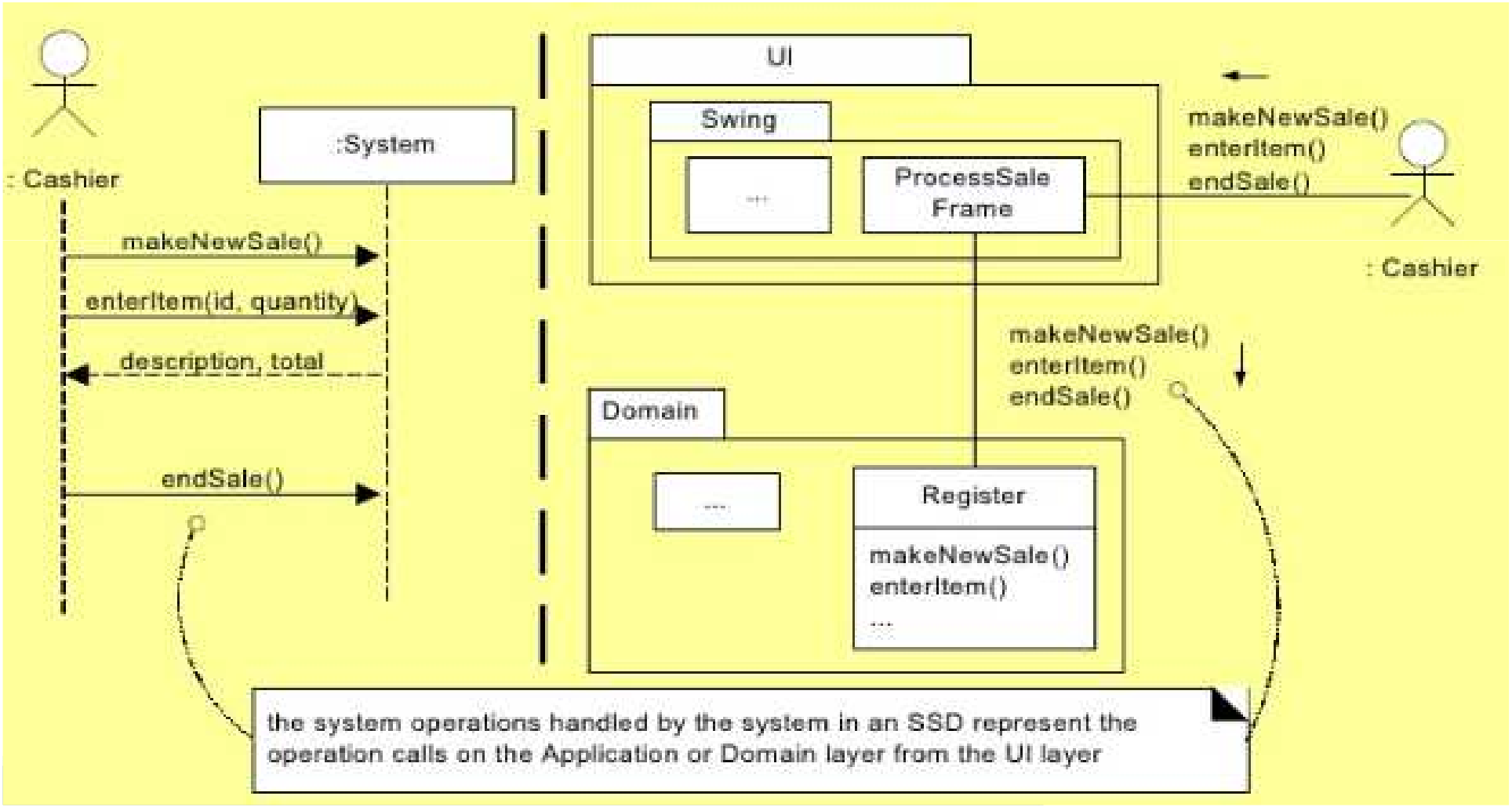
Presentation by:
Ms. J.K. Josephine Julina
Assistant Professor
Department of Information Technology
SSN College of Engineering

Introduction

Logical architecture is the large scale organization of the software classes into packages, subsystems and layers.
Higher layers get services from lower layers.
UML package diagrams are often used to illustrate the logical architecture of a system.
Layers exhibit cohesive separation of responsibility and concerns.
Lower representation gap.
Logical - because there's no decision about how these elements are deployed across different operating system processes or across physical computers in a network (deployment architecture)

SSD and Layers

The system operation requests should be forwarded by the UI layer to the domain layer for handling.



Refinement

Show only relevant dependency lines between packages or elements in the packages.

Package could be

conceptual group of things /

subsystem as well

Façade and observer are used commonly for the design connections between layers and packages.

Façade for downward collaboration; Observer for upward collaboration.

Controllers can be either a single object or a façade class.

Proxy has the same interface with the object being represented whereas adapter has different interface.

Façade pattern

Should expose small number of high level operations.

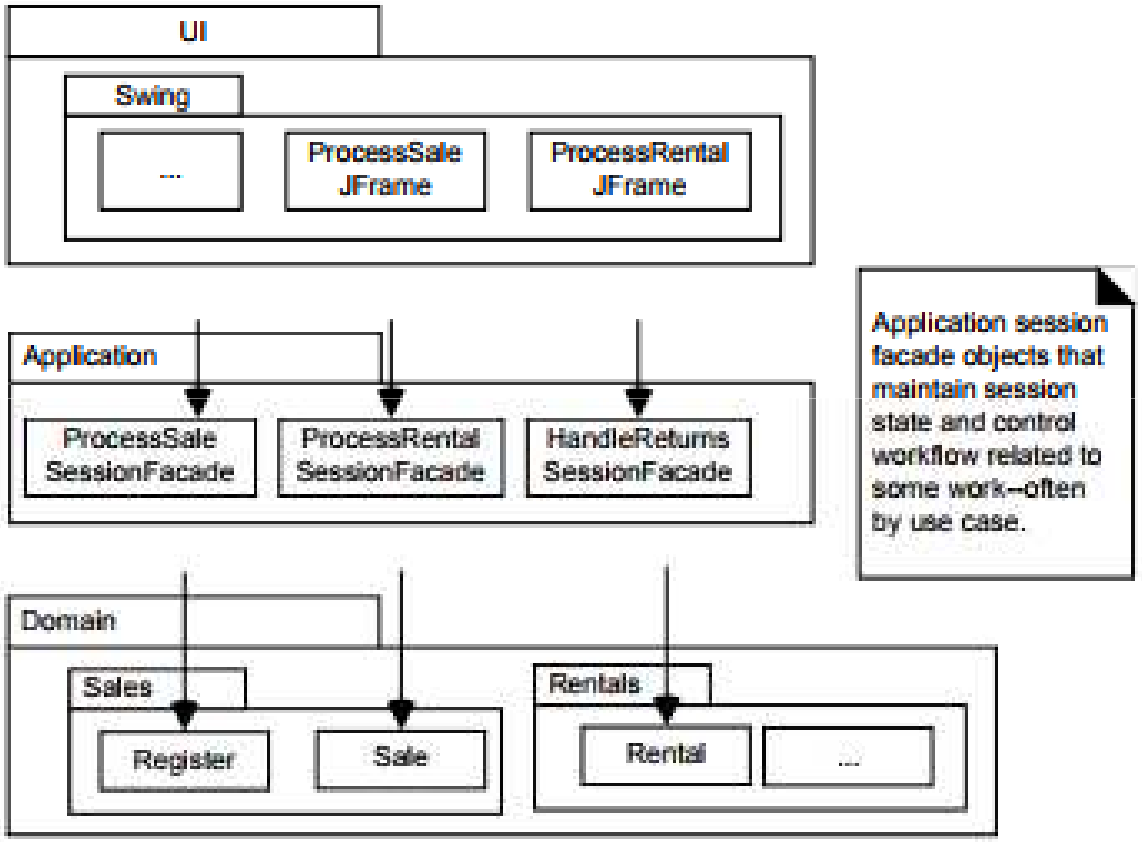
Does not do its own work, just acts as mediator or consolidator.

If there are many subsystems, then one façade exists for each subsystem.

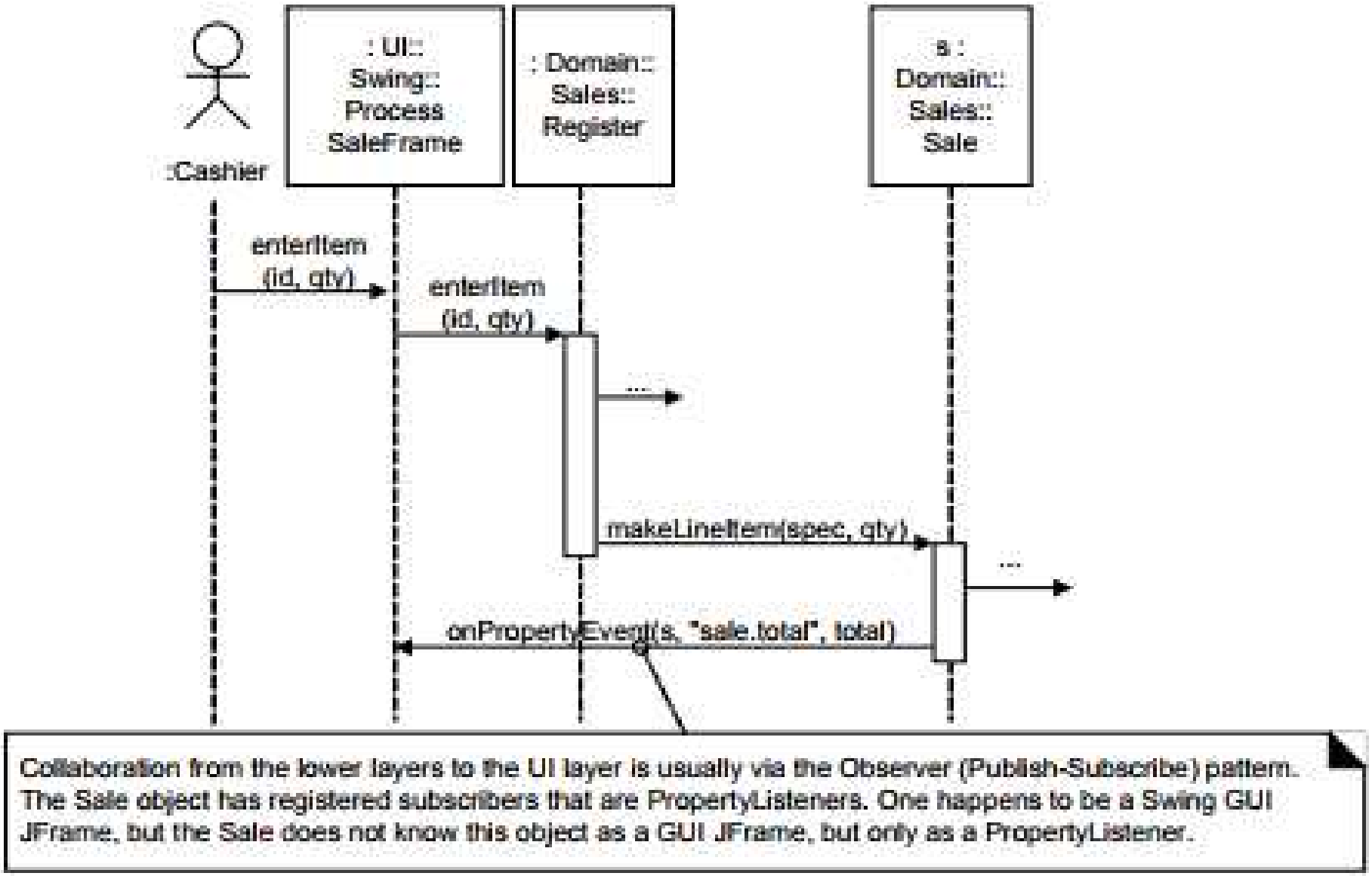
Façade is a single point of contact.

Façade pattern

Layers – UI, Domain and Technical services. If the domain layer has many use cases and system operations, it is common to introduce an application layer of objects to maintain session state for the operation of a use case.



Observer pattern



Coupling

All higher layers have dependencies on the Technical Services and Foundations layer

It is primarily the domain layer that has dependency on the Business Infrastructure layer
UI depends on application layer which in turn depends on Domain layer; UI does not call on the Domain layer, unless there is no Application layer

For a single-process “desktop” application, software objects in the domain layer are directly visible to, or passed between, UI, Application, and Technical Services

Layer issues

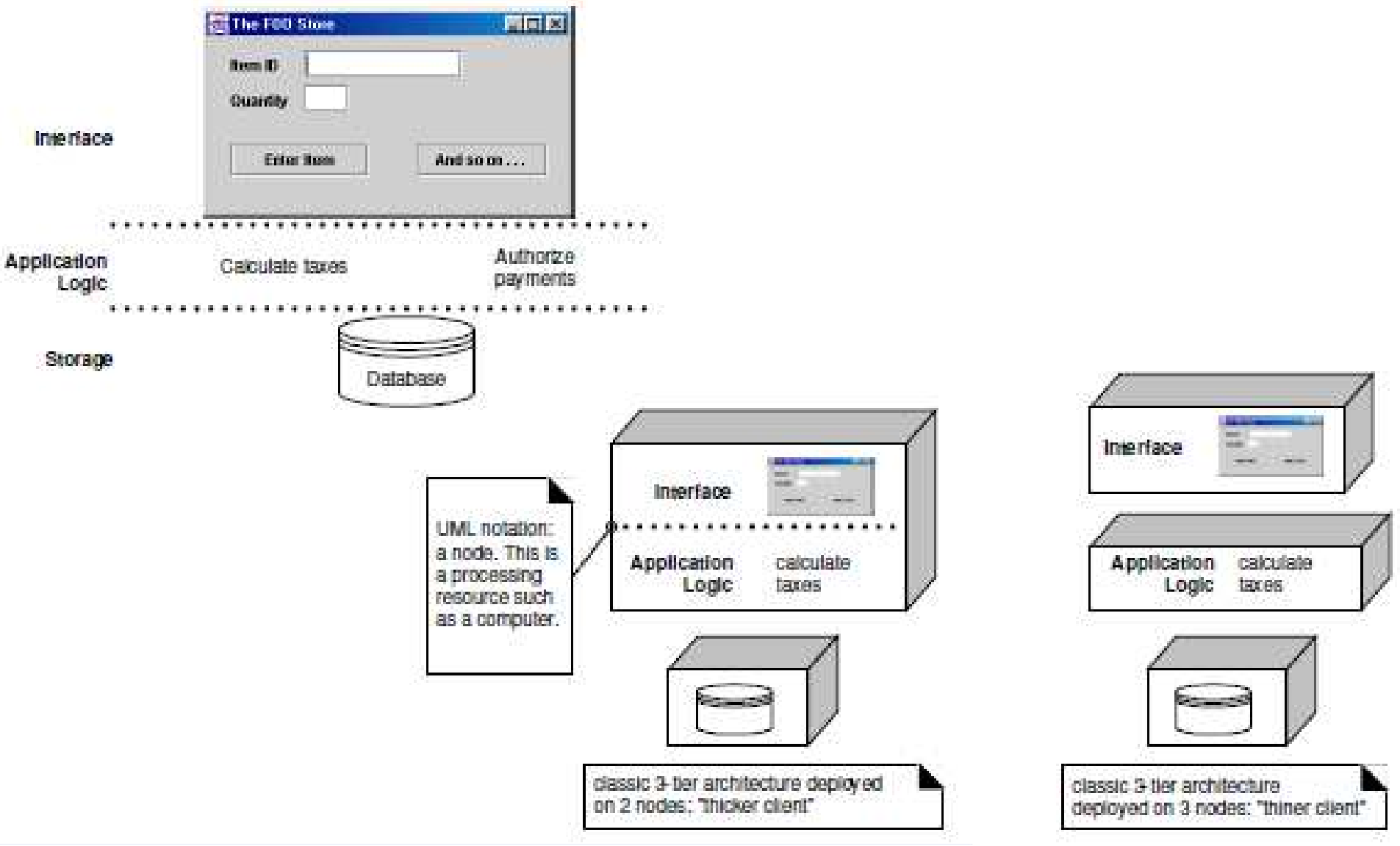
Architecture layers provide logical view; do not indicate deployment to nodes. Can degrade performance.

Handling exceptions is becomes difficult. The original exception is converted into one that is meaningful at the level of subsystem.

A core pattern is always required.

Application layer is introduced when multiple interfaces are used, distributed system with domain layer on a different server than the UI layer, domain layer cannot handle session state.

3 tier architecture



MVC architecture

Promotes reuse of components

The View is the UI layer – it contains all the GUI objects, and has very little application logic

The Model – this is the layer that contains all of the logical functionality of the system – what it actually does

The Controller – this is the “glue” layer between the View and the Model.

Takes requests from the View, passes to the Model

Takes information from the Model, passes to the View for display