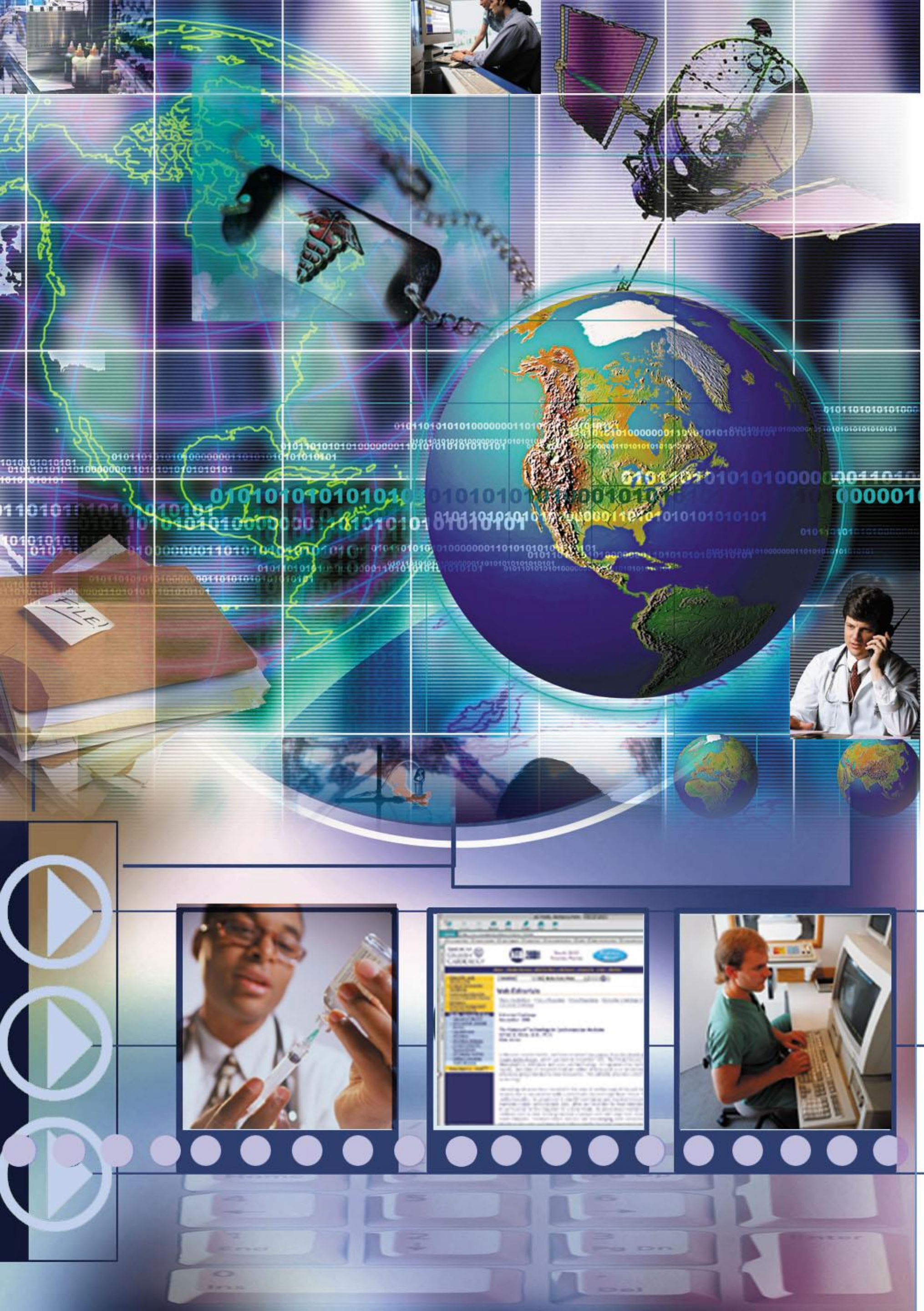


Chapter 6: ARCHITECTURE DESIGN

By:
Ts. Azaliza binti Zainal

ARCHITECTURE

Architects are trained to take your brief and can see the big picture – they look beyond your immediate requirements to design flexible buildings that will adapt with the changing needs of your business.

Architects solve problems creatively – when they are involved at the earliest planning stage, they gain more opportunities to understand your business, develop creative solutions, and propose ways to reduce costs.

RIBA (2005)

ARCHITECTURAL DESIGN

- *Architectural designs identify the necessary elements — and attributes of those elements — that support detailed design and construction efforts.*
- *These design elements are presented in context for examining how well the system as a whole collaborates to meet its intended functions.*

ARCHITECTURAL DESIGN GOAL

- ***Performance***

  - e.g. no extra communication

- ***Understandability***

  - *Developers and stakeholders understand*
  - *Keep It Simple*
  - *Should be familiar*

- **Maintainability**

  - *Separation of concerns*
  - *Changes are local*

- **Work division**

Impact of Architectural Decisions in Organization

- ***Impact on Customer Base.***
- ***Impact on Budget and Schedule.***
- ***Impact from Resource Availability.***

Key tasks that need to be performed during the software architectural design effort

- Identifying stakeholders concerns
- Identifying appropriate architectural views
- Identifying architectural styles and patterns
- Identifying influences of architectural decisions in organization
- Identifying the system's major components and interfaces
- Evaluating and validating the architecture
- Establish policies for ensuring architectural design synchronicity

SYSTEM ARCHITECTURE

- **System architects:**
  - –act on behalf of the client;
  - –address the big picture;
  - –ensure that the required qualities of the system are accounted for in the design;
  - –solve problems;
  - –ensure the required features are provided at the right cost.

SYSTEM ARCHITECTURE – KEY DEFINITIONS

- *System* is a set of components that accomplishes a specific function or set of functions.
- *Architecture* is the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.
- *Architectural Description* is a set of products that document the architecture.

SYSTEM ARCHITECTURE – KEY DEFINITIONS

- *Architectural View* is a representation of a particular system or part of a system from a particular perspective.
- *Architectural Viewpoint* is a template that describes how to create and use an architectural view. A viewpoint includes a name, stakeholders, concerns addressed by the viewpoint, and the modelling and analytic conventions.

(Garland & Anthony, 2003 & IEEE, 2000)

SYSTEM ARCHITECTURE – ARCHITECTURE TYPES

- **Soni et al.**

Type of architecture	Examples of elements	Examples of relationships
Conceptual	Components	Connectors
Module	Subsystems, modules	Exports, imports
Code	Files, directories, libraries	Includes, contains
Execution	Tasks, threads, object interactions	Uses, calls

Adapted from Weir and Daniels (1998)

SYSTEM ARCHITECTURE – ARCHITECTURE VIEWS

Kruchten - RUP’s 4 = 1 views

View	Explanation
Use case view	The important use cases in the system and scenarios that describe architecturally significant behaviour.
Logical view	Important design classes and interfaces in a package structure, with composite structure diagrams.
Implementation view	Architectural decisions made for the implementation in terms of subsystems and components and relationships among them.
Process view	A description of the processes (operating system processes and threads) and inter-process communications using stereotyped classes.
Deployment view	Physical nodes for the likely deployment platform, components deployed on the nodes and the communication channels between them, using deployment diagrams.

SYSTEM ARCHITECTURE – ADL

- **Architecture Description Language**
  - –UML 2.0 adds or changes features to support modelling architecture
  - –Package diagrams
  - –Component diagrams
  - –Composite structure diagrams
  - –Deployment diagrams

SYSTEM ARCHITECTURE – WHY MODEL IT?

- Architects use models to reason about the system and its ability to deliver the required quality attributes (reliability, performance, security etc.).
- Particular views help to reason about particular quality attributes.

SYSTEM ARCHITECTURE – WHY MODEL IT?

View	Contribution to assessing performance
Use case view	The use cases that require high performance can be identified and the scenarios used to walkthrough how the other views will affect the performance requirement.
Logical view	The logical view of classes will show whether techniques such as creating lightweight objects or value objects have been used to reduce the overheads associated with passing values around.
Implementation view	The more components or subsystems are involved, the more likely there are to be communication overheads, so the implementation view should show a small number of components used in the process.
Process view	The process view can be used to assess how many running processes will exist, and whether there will be multiple instances of the same process so that the work can be shared out by a special process that handles load-balancing. The kind of inter-process communication that is used will affect how efficiently data can be passed between processes.
Deployment view	The deployment view will show where different components are deployed, and whether data has to travel from machine to machine, or whether all the processes needed to deliver a high-performance use case are located on the same machine.

SYSTEM ARCHITECTURE INFLUENCES

- Existing systems
- Enterprise architectures
- Technical reference architectures

SYSTEM ARCHITECTURE INFLUENCES

Existing Systems

- Standard architectures

- **Heritage systems**
  - May be wrapped using adapters
- **Services**
  - Wrapping systems with adapters
- **Reverse-engineering and MDA**
  - Generate platform-specific models (PSMs) from platform-independent models (PIMs)
  - Reverse-engineer existing system logic to PIMs

SYSTEM ARCHITECTURE INFLUENCES

Enterprise Architectures

- Pressure in the United States
  - Clinger-Cohen Act 1996
  - Sarbanes-Oxley Act 2002
- Frameworks
  - Federal Enterprise Architecture Framework
  - Standards and Architectures for eGovernment Applications
  - Zachman Framework

SYSTEM ARCHITECTURE INFLUENCES

- **Technical Reference Architectures**
  - Standards for technologies to apply
  - Guidance on how to apply them
- **The Open Group Architecture Framework (TOGAF)**
  - Architecture Development Method
  - Enterprise Continuum
  - Resources

SYSTEM ARCHITECTURE
ARCHITECTURAL STYLES

- **Architectural styles apply to physical architecture and to software architecture**
- **Bass et al identify five main types:**
  - Independent components
  - Data flow
  - Data centered
  - Virtual machine
  - Call and return

SUBSYSTEMS

- A subsystem typically groups together elements of the system that share some common properties
- An object-oriented subsystem encapsulates a coherent set of responsibilities in order to ensure that it has integrity and can be maintained

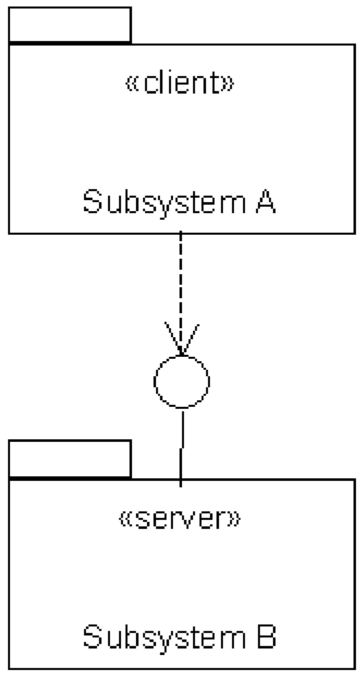
SUBSYSTEMS

- The subdivision of an information system into subsystems has the following advantages
  - It produces smaller units of development
  - It helps to maximize reuse at the component level
  - It helps the developers to cope with complexity
  - It improves maintainability
  - It aids portability

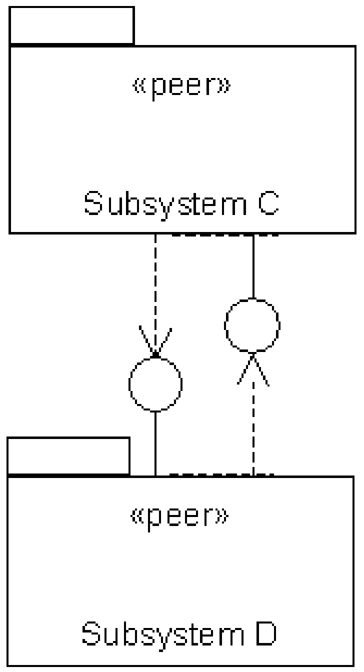
SUBSYSTEMS

- Each subsystem provides services for other subsystems, and there are two different styles of communication that make this possible
- These are known as *client–server* and *peer-to-peer* communication

STYLES OF COMMUNICATION BETWEEN SUBSYSTEMS



The server subsystem does not depend on the client subsystem and is not affected by changes to the client's interface.



Each peer subsystem depends on the other and each is affected by changes in the other's interface.

CLIENT-SERVER COMMUNICATION

- Client–server communication requires the client to know the interface of the server subsystem, but the communication is only in one direction
- The client subsystem requests services from the server subsystem and not vice versa

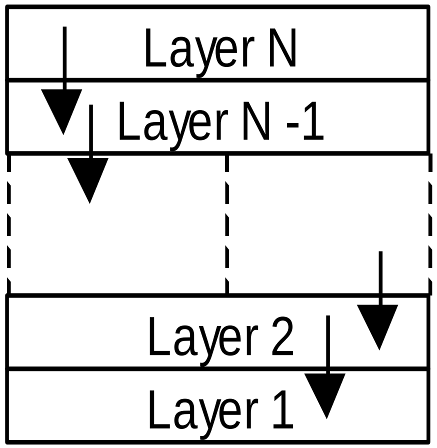
PEER-TO-PEER COMMUNICATION

- Peer-to-peer communication requires each subsystem to know the interface of the other, thus coupling them more tightly
- The communication is two way since either peer subsystem may request services from the other

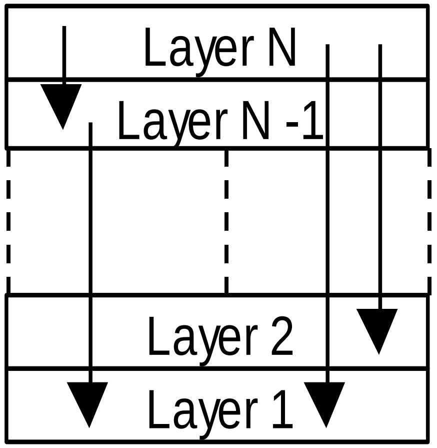
LAYERING AND PARTITIONING

- **Two general approaches to the division of a software system into subsystems**
  - *Layering*—so called because the different subsystems usually represent different levels of abstraction
  - *Partitioning*, which usually means that each subsystem focuses on a different aspect of the functionality of the system as a whole
- **Both approaches are often used together on one system**

SCHEMATIC OF A LAYERED ARCHITECTURE



Closed architecture—messages may only be sent to the adjacent lower layer.



Open architecture—messages can be sent to any lower layer.

LAYERED ARCHITECTURE

- A closed architecture minimizes dependencies between the layers and reduces the impact of a change to the interface of any one layer
- An open layered architecture produces more compact code, the services of all lower level layers can be accessed directly by any layer above them without the need for extra program code to pass messages through each intervening layer, but this breaks the encapsulation of the layers

OSI 7 LAYER MODEL

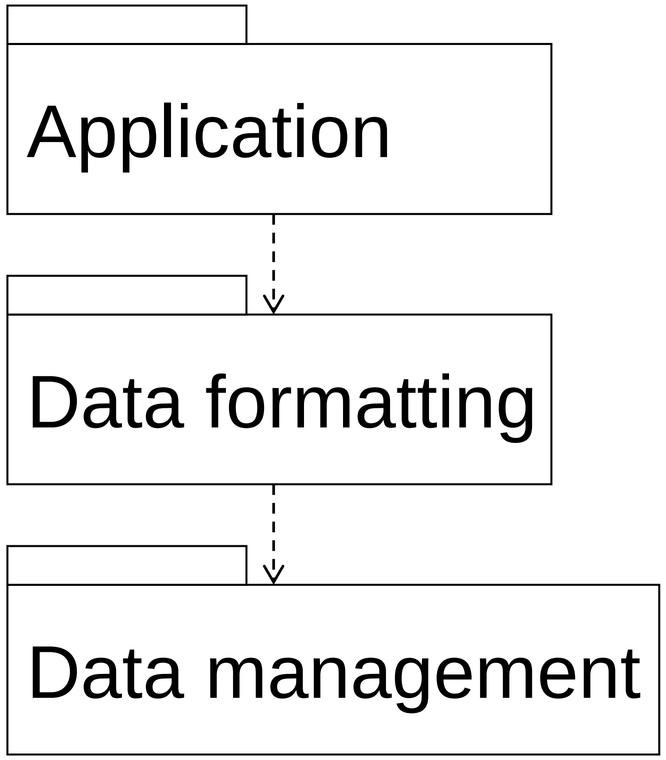
(Adapted from Buschmann et al., 1996)

Layer 7: Application Provides miscellaneous protocols for common activities.
Layer 6: Presentation Structures information and attaches semantics.
Layer 5: Session Provides dialogue control and synchronization facilities.
Layer 4: Transport Breaks messages into packets and ensures delivery.
Layer 3: Network Selects a route from sender to receiver.
Layer 2: Data Link Detects and corrects errors in bit sequences.
Layer 1: Physical Transmits bits: sets transmission rate (baud), bit-code, connection, etc.

APPLYING A LAYERED ARCHITECTURE

- **Issues that need to be addressed include:**
  - –maintaining the stability of the interfaces of each layer
  - –the construction of other systems using some of the lower layers
  - –variations in the appropriate level of granularity for subsystems
  - –the further sub-division of complex layers
  - –performance reductions due to a closed layered architecture

SIMPLE LAYERED ARCHITECTURE



DEVELOPING A LAYERED ARCHITECTURE

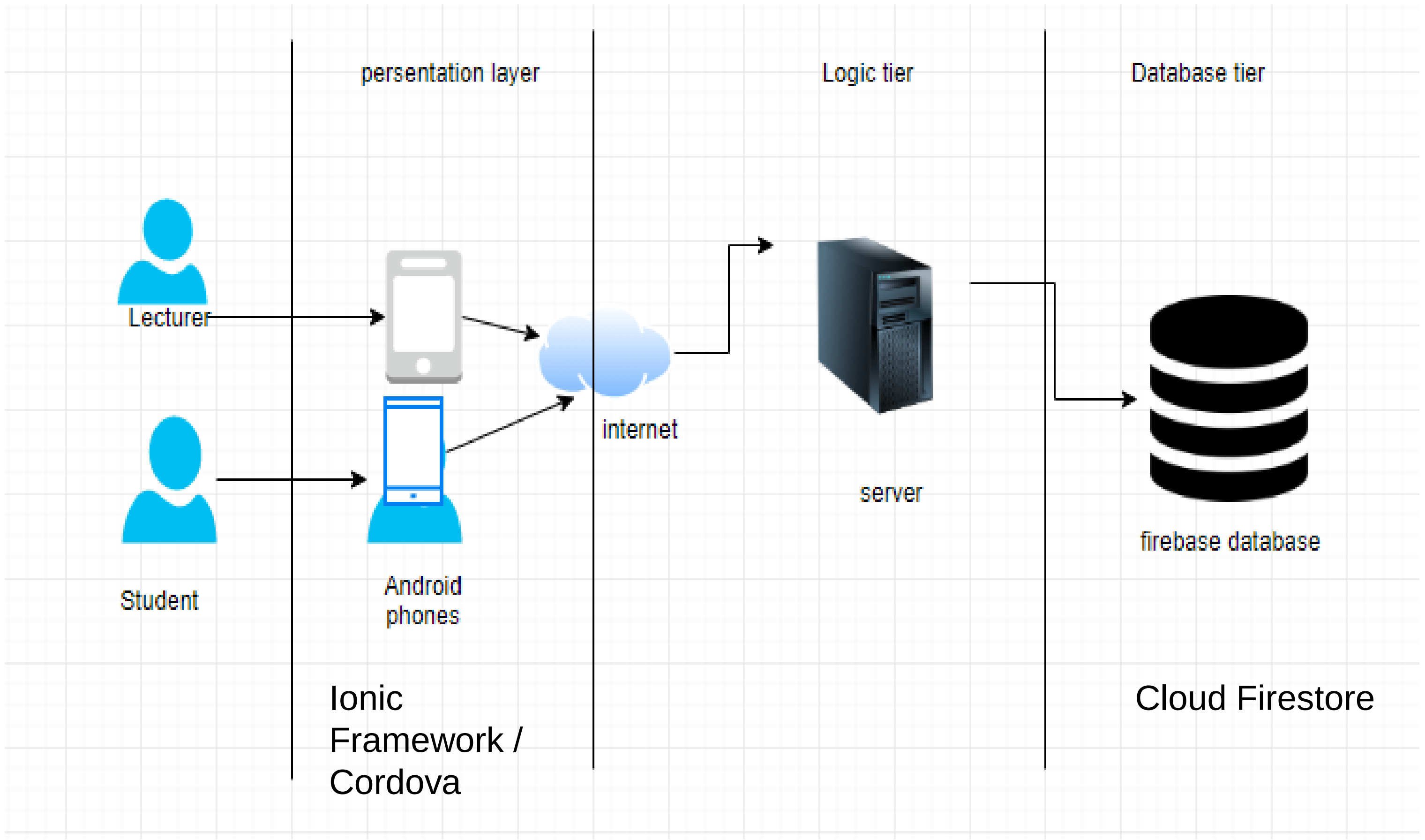
1. Define the criteria by which the application will be grouped into layers. A commonly used criterion is level of abstraction from the hardware.
2. Determine the number of layers.
3. Name the layers and assign functionality to them.
4. Specify the services for each layer.
5. Refine the layering by iterating through steps 1 to 4.

DEVELOPING A LAYERED ARCHITECTURE

6. Specify interfaces for each layer.
7. Specify the structure of each layer. This may involve partitioning within the layer.
8. Specify the communication between adjacent layers (this assumes that a closed layer architecture is intended).
9. Reduce the coupling between adjacent layers. This effectively means that each layer should be strongly encapsulated.

(Adapted from Buschmann et al., 1996)

EXAMPLE OF THREE LAYER ARCHITECTURES

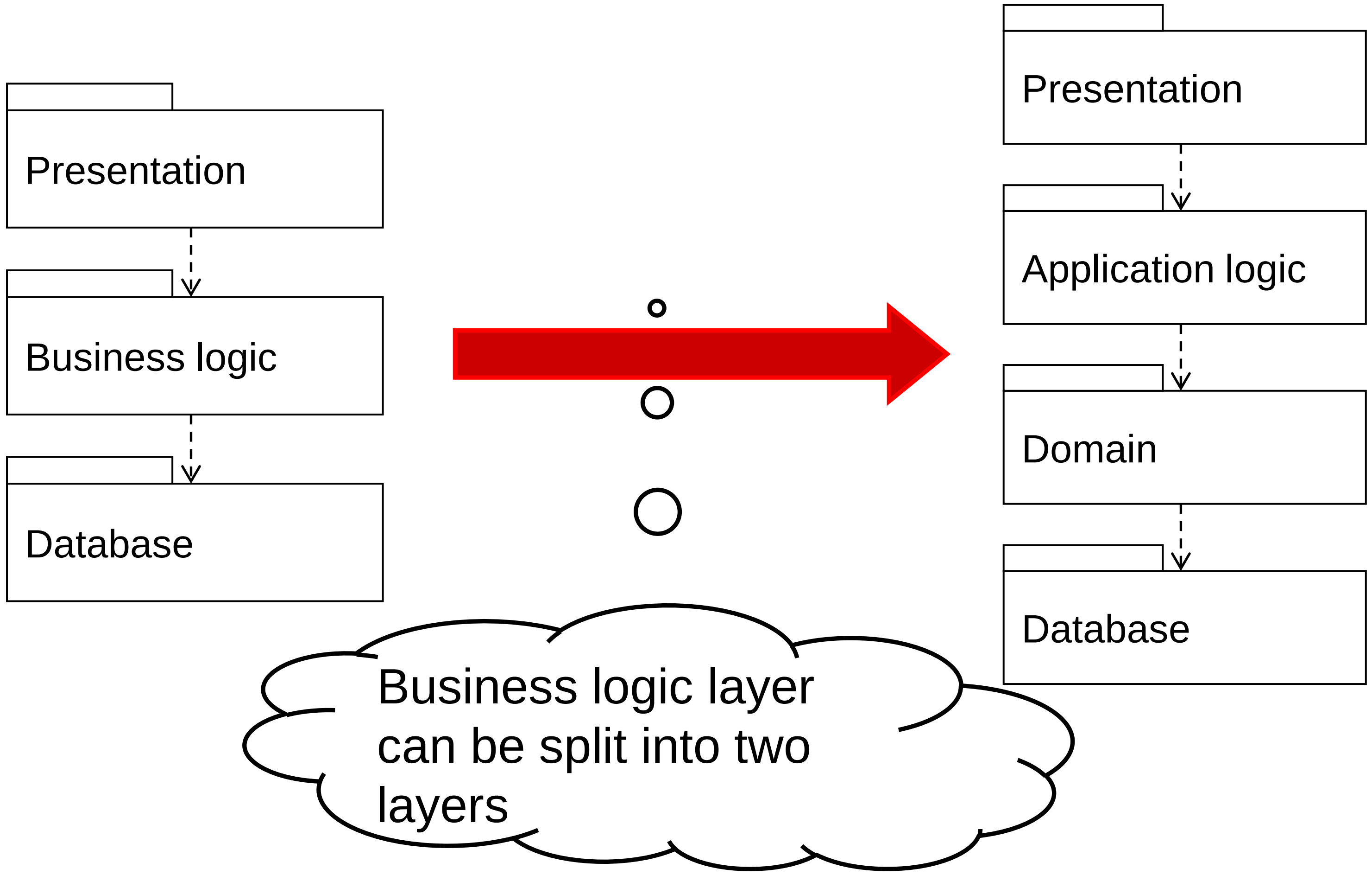


Ionic Framework / Cordova

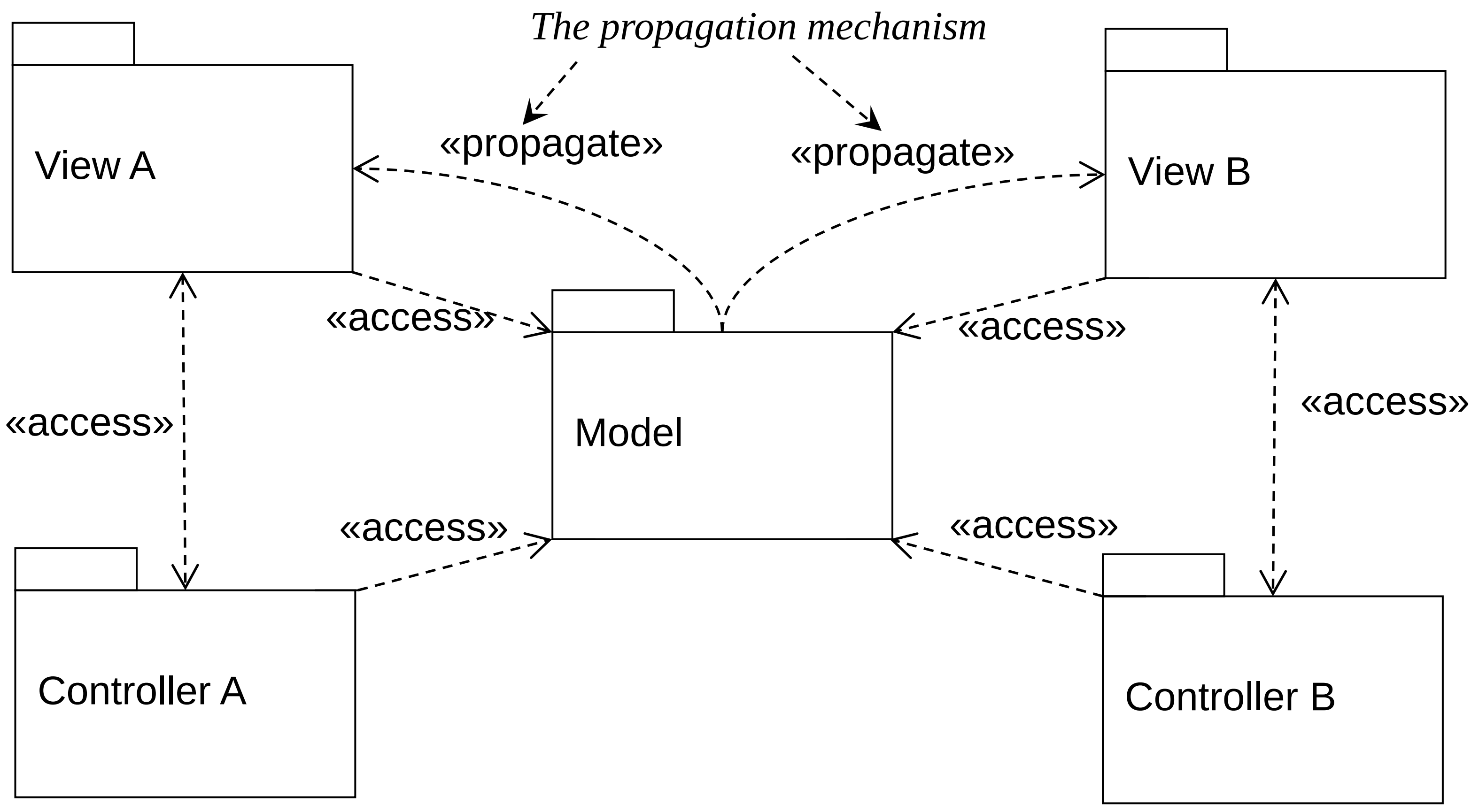
Cloud Firestore

Refer : https://ionicthemes.com/tutorials/about/building-a-ionic-firebase-app-step-by-step

THREE & FOUR LAYER ARCHITECTURES



MODEL-VIEW-CONTROLLER



(Adapted from Hopkins and Horan, 1995)

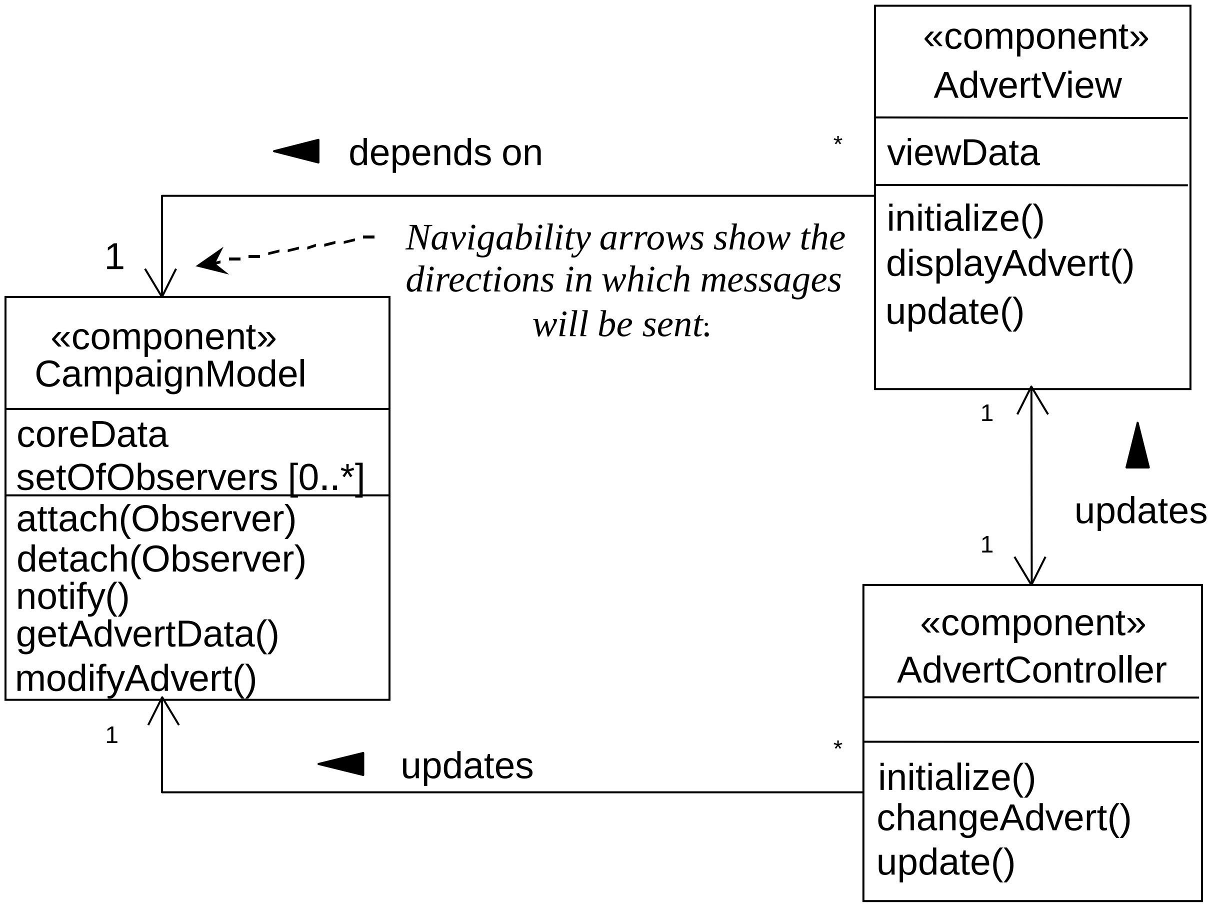
MODEL-VIEW-CONTROLLER

- *Model*—provides the central functionality of the application and is aware of each of its dependent view and controller components.
- *View*—corresponds to a particular style and format of presentation of information to the user. The view retrieves data from the model and updates its presentations when data has been changed in one of the other views. The view creates its associated controller.

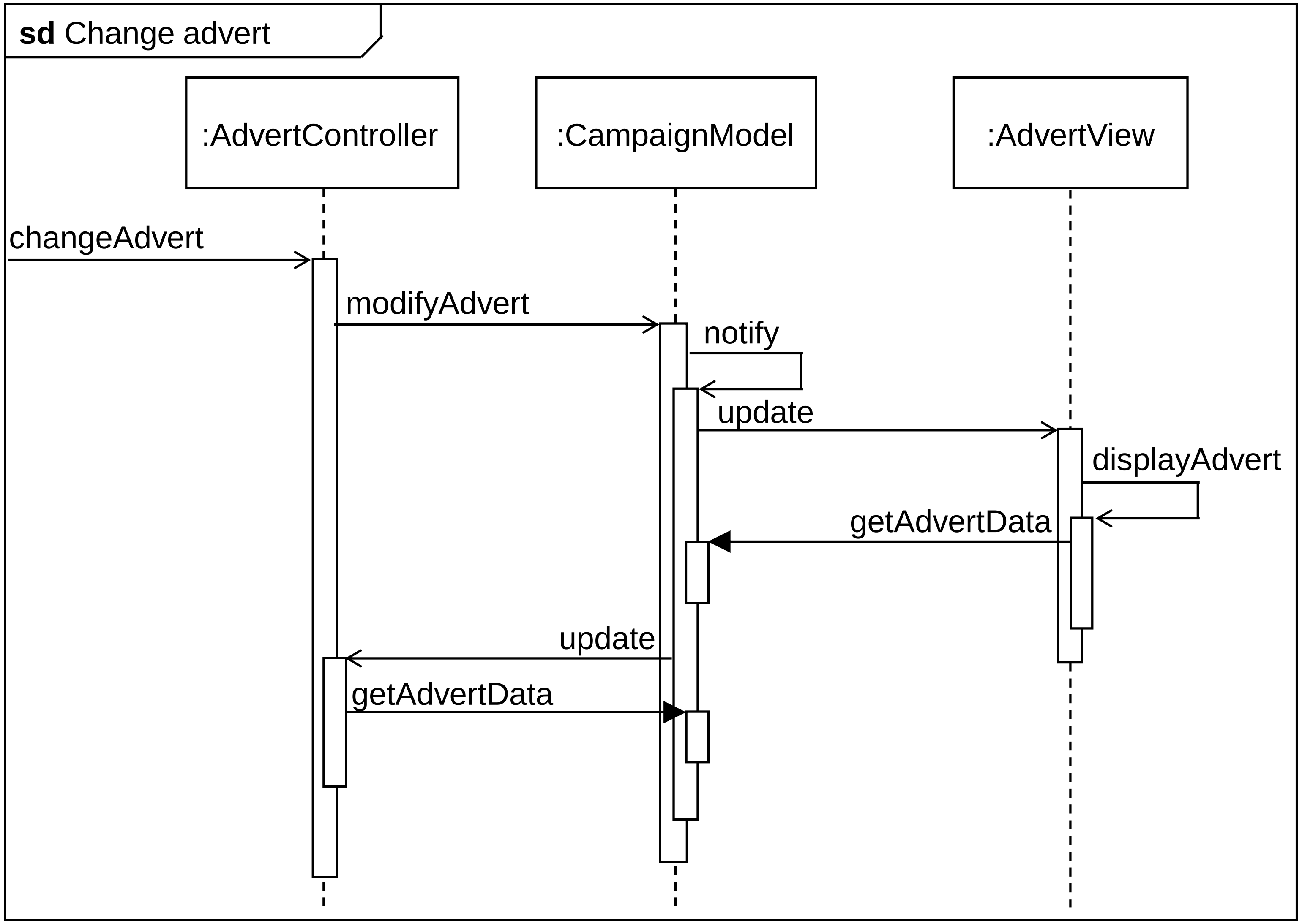
MODEL-VIEW-CONTROLLER

- *Controller*—accepts user input in the form of events that trigger the execution of operations within the model. These may cause changes to the information and in turn trigger updates in all the views ensuring that they are all up to date.
- *Propagation Mechanism*—enables the model to inform each view that the model data has changed and as a result the view must update itself. It is also often called the dependency mechanism.

MVC APPLIED TO AGATE



MVC COMPONENT INTERACTION



PLANNING FOR DESIGN

- Planning for when platform is known
- Setting standards
- Allowing time for training
- Agreeing objectives and planning tests
- Agree procedures to decide on trade-offs that significantly affect the system
- Planning time for different aspects of design

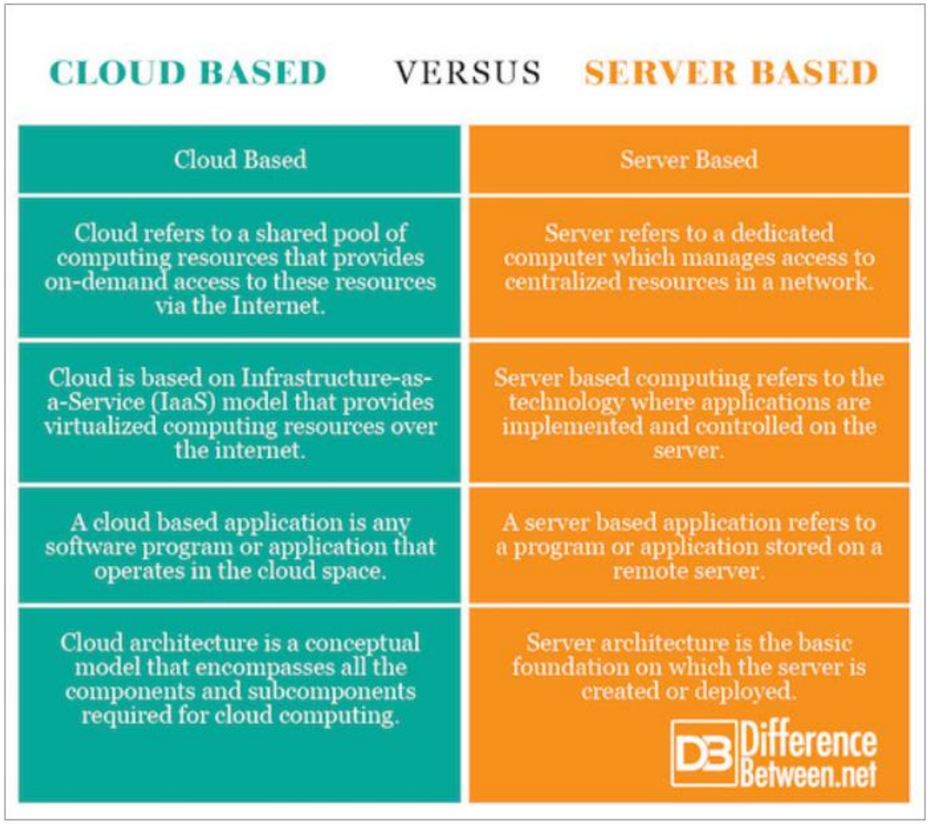
DEVELOPMENT STANDARDS

- HCI guidelines
- Input/output device guidelines
- Construction guidelines

CLOUD BASED vs SERVER BASED ARCHITECTURE

- Cloud computing architecture refers to the components and subcomponents required for cloud computing.

Cloud Based vs. Server Based: Comparison Chart



CLOUD BASED VERSUS SERVER BASED

Cloud Based	Server Based
Cloud refers to a shared pool of computing resources that provides on-demand access to these resources via the Internet.	Server refers to a dedicated computer which manages access to centralized resources in a network.
Cloud is based on Infrastructure-as-a-Service (IaaS) model that provides virtualized computing resources over the internet.	Server based computing refers to the technology where applications are implemented and controlled on the server.
A cloud based application is any software program or application that operates in the cloud space.	A server based application refers to a program or application stored on a remote server.
Cloud architecture is a conceptual model that encompasses all the components and subcomponents required for cloud computing.	Server architecture is the basic foundation on which the server is created or deployed.

DB Difference Between.net

CLOUD SERVICE MODELS

- **Software as a Service (SaaS):** Consumer uses applications running on a cloud.
- **Platform as a Service (PaaS):** Provides a development and deployment platform in the cloud
- **Infrastructure as a Service (IaaS):** Provides a virtual machine for developer or system administrator

Economical benefits of using cloud-based architecture

- Large data centers are cheaper (cost of power, infrastructure labor cost), security and reliably, hardware cost).
- Utilization of equipment (virtualization, adapt to time of day, year, usage patterns).
- Multi-tenancy: Single application used by multiple consumers saves costs related to user-support, simple upgrade, single version of software.

Q & A

THANK YOU FOR YOUR ATTENTION

☺