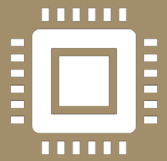
SOFTWARE DESIGN
COSC 4353/6353

Dr. Raj Singh

ARCHITECTURAL PATTERNS

An Architectural pattern is a general, reusable solution to a commonly occurring problem in software architecture within a given context.



Often documented as software design patterns.

The concept of an architectural pattern has a broader scope than the concept of design pattern.

AP - FOCUS

SOFTWARE AND HARDWARE LIMITATIONS

PERFORMANCE

HIGH AVAILABILITY

MINIMIZATION OF A BUSINESS RISK

SECURITY

DATA ACCESS AND MODELING

ARCHITECTURE PATTERNS VS. DESIGN PATTERNS

DPs are usually associated with code level commonalities while APs are seen as commonality at higher level.

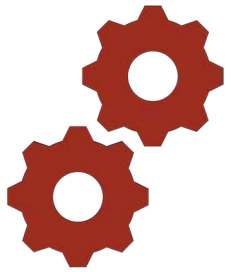
DPs provide various schemes for refining and building smaller subsystems. Aps cover the fundamental organization of the system.

DPs are usually influenced by programming language whereas APs are influenced by the system.

DPs are medium-scale tactics that flesh out some of the structure and behavior of entities and their relationships.

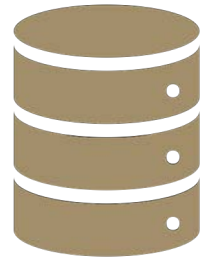
APs are high-level strategies that concerns large-scale components, the global properties and mechanisms of a system.

TYPES



Application Architecture Patterns

Science and art of ensuring the suite of applications being used by an organization to create the composite architecture is scalable, reliable, available and manageable.



Data Architecture Patterns

Composed of models, policies, rules or standards that govern which data is collected, and how it is stored, arranged, integrated, and put to use in data systems and in organizations

APPLICATION PATTERNS

Multitier / n-tier Architecture

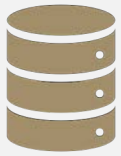
Model-View-Controller

Authentication Patterns

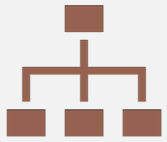
Authorization Patterns

MULTITIER/N-TIER ARCHITECTURE

N-tier application architecture provides a model by which developers can create flexible and reusable applications.



By segregating an application into tiers, developers acquire the option of modifying or adding a specific layer, instead of reworking the entire application.



A Three-tier architecture is typically composed of a presentation tier, a business or data access tier, and a data tier

THREE-TIER ARCHITECTURE

Presentation tier

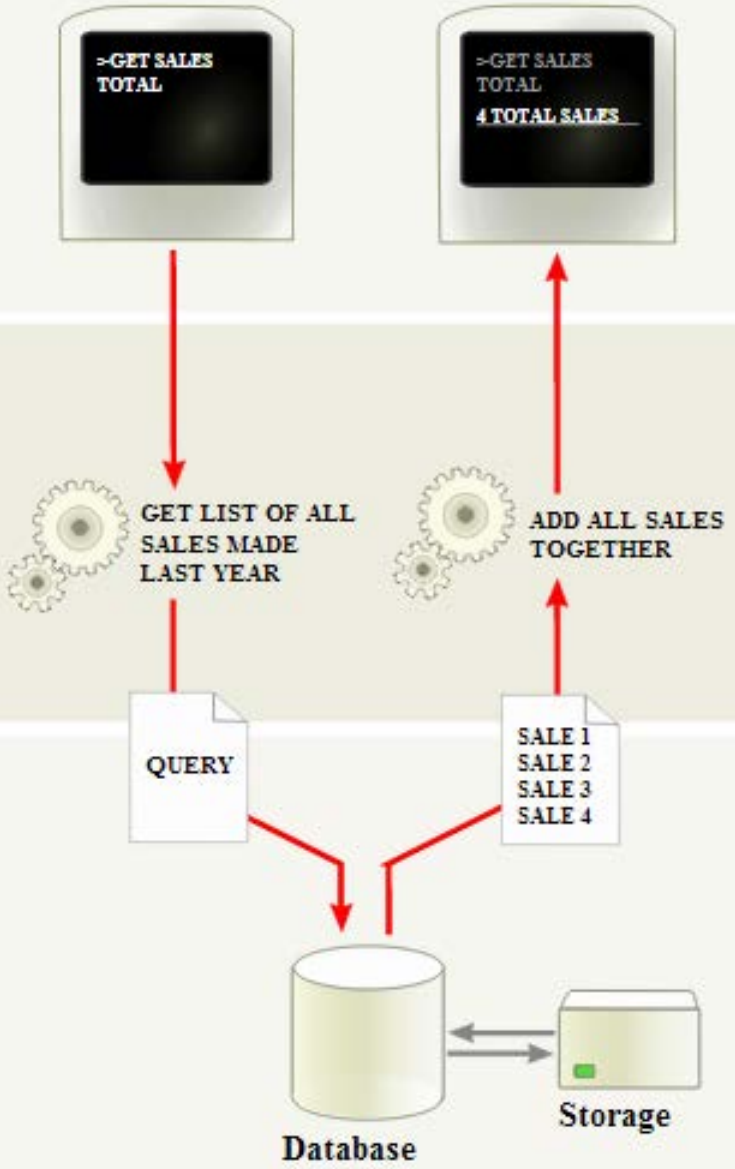
The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.

Logic tier

This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.

Data tier

Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.



MVC

Model–view–controller (MVC) is a software architecture pattern which separates the representation of information from the user's interaction with it.

The model consists of application data, business rules, logic, and functions.

A view can be any output representation of data, such as a chart or a diagram. Multiple views of the same data are possible, such as a bar chart for management and a tabular view for accountants.

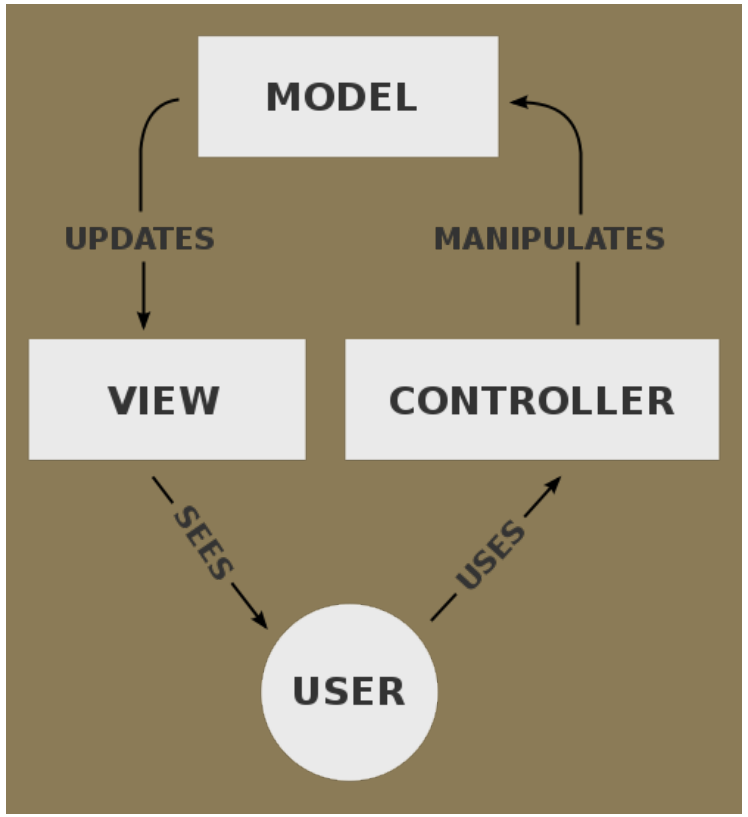
The controller mediates input, converting it to commands for the model or view

MVC

A **controller** can send commands to the model to update the model's state. It can also send commands to its associated view to change the view's presentation of the model

A **model** notifies its associated views and controllers when there has been a change in its state. This notification allows the views to produce updated output, and the controllers to change the available set of commands.

A **view** requests information from the model that it needs for generating an output representation to the user.



MULTITIER VS. MVC

At first glance, the three tiers may seem similar to MVC but fundamentally they are different.

The client tier never communicates directly with the data tier in a three-tier model.

All communication must pass through the middle tier. Conceptually the three-tier architecture is linear.

However, the 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.

AUTHENTICATION PATTERNS

Authentication is the process of identifying an individual using the credentials of that individual.

As computer systems have increased in complexity, the challenge of authenticating users has also increased.

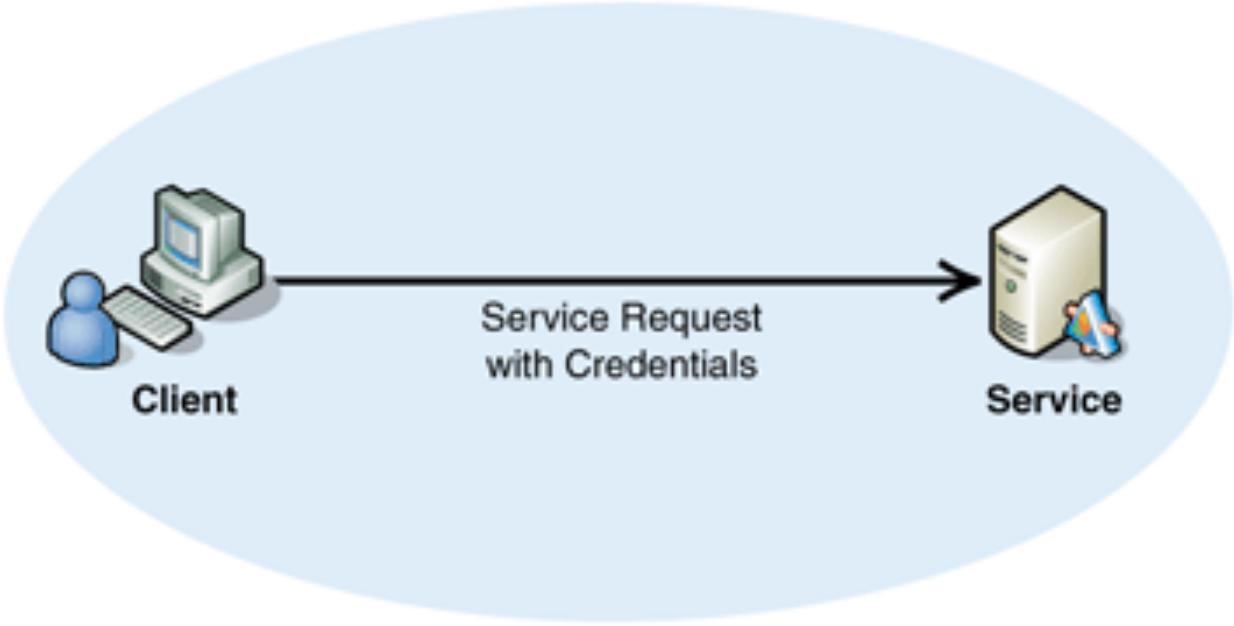
There are a variety of models for authentication

E.g., clients accessing a Web application may directly provide credentials for authentication. However, a third-party broker, such as a Kerberos domain controller, may be used to provide a security token for authentication.

DIRECT AUTHENTICATION

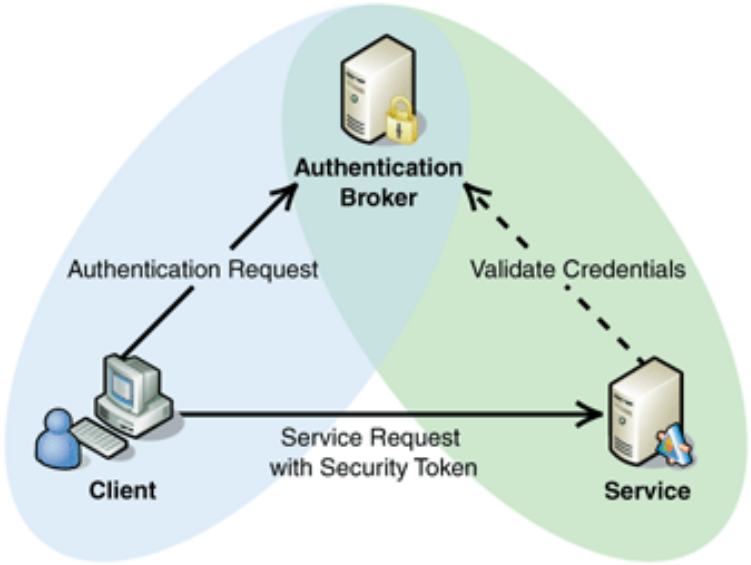
Both the client and service participate in a trust relationship.

It allows them to exchange and validate credentials including passwords.



BROKERED AUTHENTICATION

- The broker authenticates the client and then issues a security token that the service can use to authenticate the client.
- The security token is always verified, but the service does not need to interact with the broker to perform the verification.
- This is because the token itself can contain proof of a relationship with the broker, which can be used by the service to verify the token



AUTHORIZATION PATTERNS

Authorization is the process of determining whether an authenticated client is allowed to access a resource or perform a task within a security domain.

Authorization uses information about a client's identity and/or roles to determine the resources or tasks that a client can perform.

TYPES

Role-based Authorization	controlling which users have access to resources based on the role of the user
Resource-based Authorization	performed declaratively on a resource, depending on the type of the resource and the mechanism used to perform authorization
Activity-based Authorization	Who can do what, e.g., CRUD
Time-based Authorization	Time controlled access to a system or subsystem

Data Architecture Patterns

Master Data Patterns

Business Intelligence

- Data Mart
- Data Warehouse

Data Integration

- Extract Transform Load (ETL)
- Service Oriented Architecture

Data Storage

- Transactional Data Stores
- Operational Data Store

MASTER DATA PATTERNS

- The collective application of governance, business processes, policies, standards and tools facilitate consistency in master data
- reference / basic business data used in a single application, system or process
- a single source of reference data used across multiple systems, applications, and/or processes
- is the single source of reference data used across all systems, applications, and processes
- is the single source of basic business data for an entire marketplace

BUSINESS INTELLIGENCE

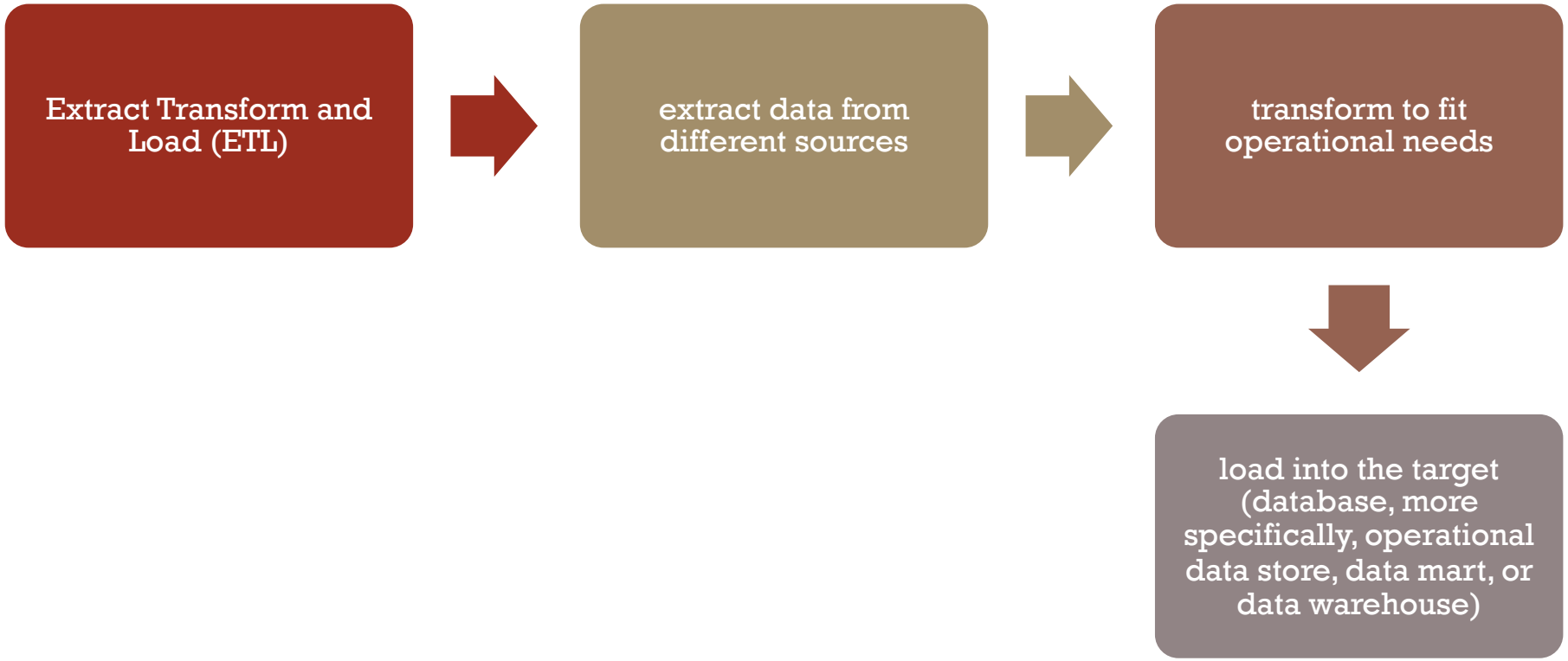
Data Warehouse

- is a database used for reporting and data analysis
- central repository of data which is created by integrating data from one or more disparate sources
- store current as well as historical data and are used for creating trending reports

Data Mart

- Access layer of the data warehouse environment that is used to get data out to the users.
- Easy access to frequently needed data
- Lower cost than implementing a full data warehouse
- Contains only business essential data and is less cluttered

DATA INTEGRATION



DATA STORAGE

Transactional Data Store

Transaction data are data describing an event (the change as a result of a transaction).

Transaction data always has a time dimension, a numerical value and refers to one or more objects (i.e. the reference data).

OPERATIONAL DATA STORE (ODS)

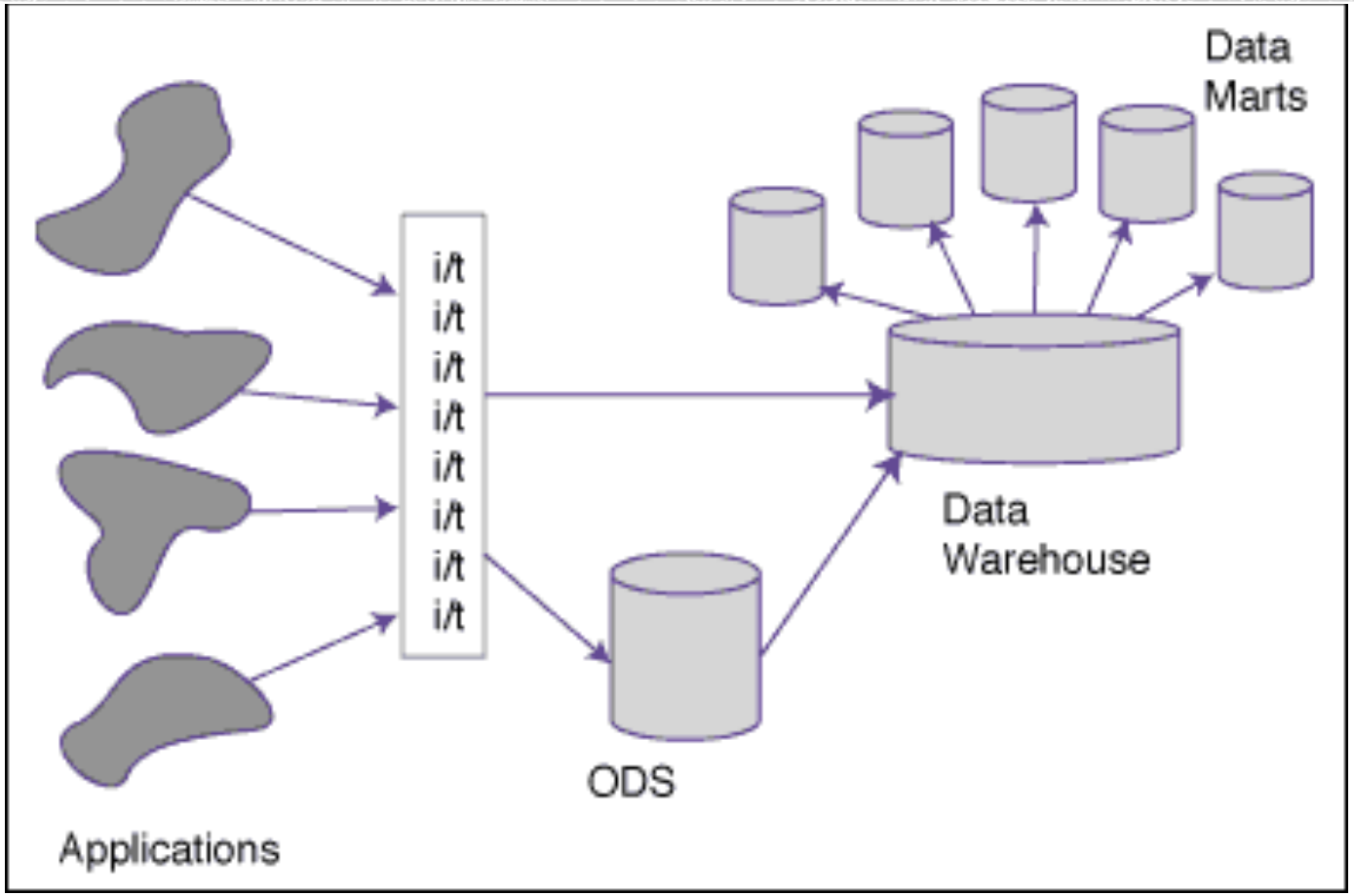
Database designed to integrate data from multiple sources for additional operations on the data.

The integration often involves cleaning, resolving redundancy and checking against business rules for integrity.

An ODS involves:

- Extracting data from operational systems;
- Moving it into ODS structures; and
- Reorganizing and structuring the data for analysis purposes

ODS


Data
Marts
i/t
i/t
i/t
i/t
i/t
i/t
i/t
i/t
Data
Warehouse
ODS
Applications

DW VS. ODS

Data Warehouse

- Is intended to support strategic planning and business intelligence decision support and should contain:
  - Integrated subject oriented data, e.g. sales data
  - Static data and historical data
  - Aggregated or summarized data

Operational Data Store :

- Is intended to support operational management and monitoring and should contain:
  - Integrated subject oriented data (similar to data warehouses) e.g. sales data
  - Volatile data, will probably change frequently
  - Current detailed data

SERVICE-ORIENTED ARCHITECTURE (SOA)

Discrete pieces of software providing application functionality as services to other applications.

Independent of any vendor, product or technology.

Services are unassociated, loosely coupled units of functionality that are self-contained.

Each service implements one action, such as

- submitting an online order
- retrieving an online bank statement
- modifying an online order

Within a SOA, services use defined protocols that describe how services pass and parse messages using description metadata.

SOA

Extensive use of XML in SOA to structure data

Web Services Description Language (WSDL) typically describes the services themselves

SOAP protocol describes the communication protocol

SOA depends on data and services that are described by metadata that should meet the following two criteria:

- software systems can use metadata to configure dynamically by discovery and incorporation of defined services.
- system designers can understand and manage with a reasonable expenditure of cost and effort.

SOA - BENEFITS

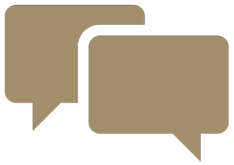
- Allows simultaneous use and easy mutual data exchange between programs of different vendors without additional programming
- These services are also reusable, resulting in lower development and maintenance costs
- Providing more value once the service is developed and tested
- Having reusable services readily available also results in quicker time to market

HOMEWORK

Review class notes and previous recordings

Additional reading:

Examples of Design Patterns



Start a discussion on Google Groups to clarify your doubts