SERVICE ORIENTED ARCHITECTURE

UNIT - I

(K.S.C IV IT)

SOA : SOA represents an open, agile, extensible, federated, composable architecture of autonomous, QOS - capable, vendor diverse, interoperable, discoverable, reusable services implemented as web services.

Web Service:

Web Services are components of an application that communicate via open protocols. They are self contained and self-described. WSDL, SOAP, UDDI form the core of Web Services.

I ROOTS OF SOA:

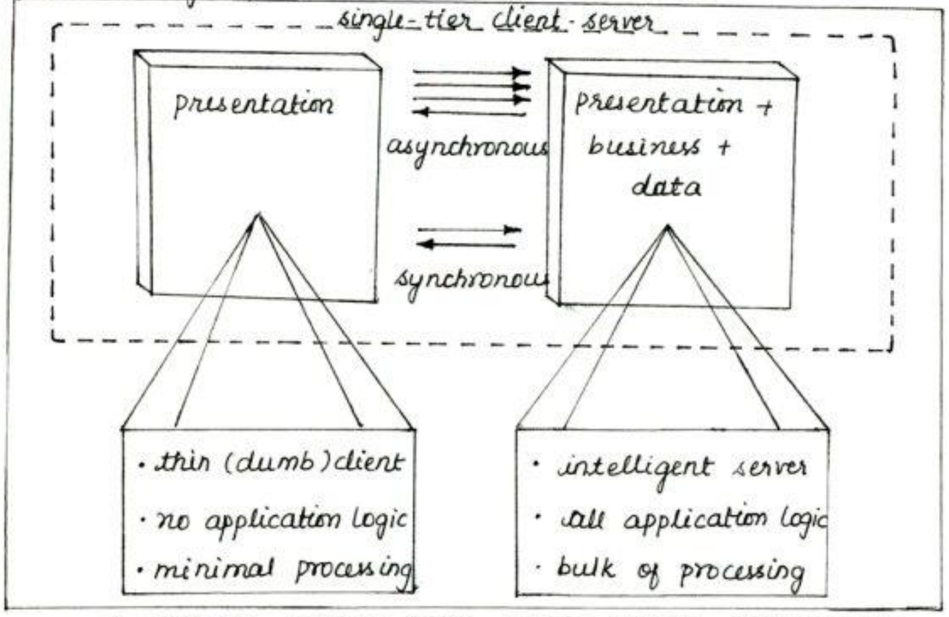
a) SOA vs. Client Server Architecture:

- The technology set for client - server applications included 4GLs like VB and Power Builder, RDBMSs.
- The SOA technology set has expanded to build Web technologies (HTML, CSS, HTTP, etc)
- SOA requires the use of XML data representation architecture along with a SOAP messaging framework.

b) Client - Server Application Server:

- Centralized at the Server level.
- Database manage user accounts and groups.
- Also controlled within the client executable.
- Security for SOA is much more complex.
- Security complexity is directly related to the degree of security measures required.

- Multiple technologies are required, many in WS WS-security framework.



A TYPICAL SINGLE-TIER CLIENT-SERVER ARCHITECTURE

c) Client-Server Application Administration:

- Significant maintenance costs associated with client-server.
- Each client housed application code.
- Each update required redistribution.
- Client stations were subject to environment-specific problems.
- Increased server-side demands on databases.
- SOA solutions are not immune to client-side maintenance challenges.
- Distributed back-end supports scalability, but new admin demands are introduced.
- Management of server resources and service interfaces may require new admin tools and even a private registry.

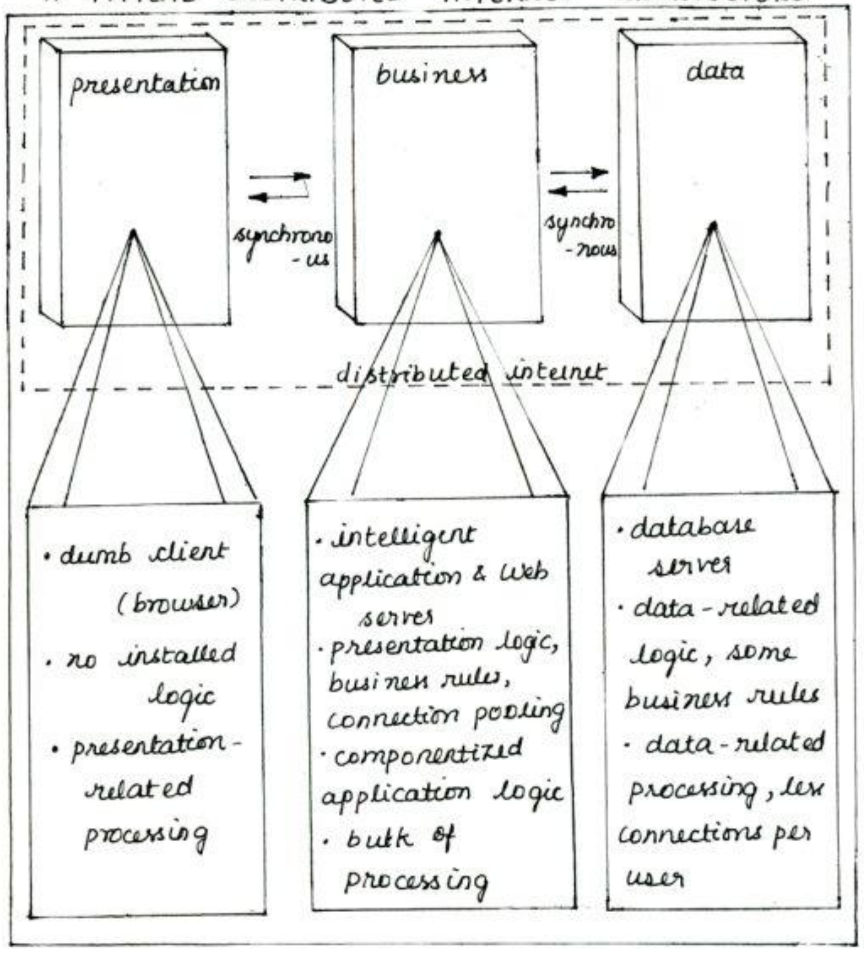
- Commitment to services and their maintenance may require cultural change in an organisation.

d) <u>SOA vs. Traditional Distributed Internet Architecture:</u>

- Multiple client-server architectures have appeared.
- Client-server DB connections have been replaced with Remote Procedure Call connections (RPC) using COBRA or DCOM.
- Middleware application servers and transaction monitors require significant attention.
- Multi-tired client-server environments began incorporating internet technology in 90s.
- Distributed Internet application put all the application logic on the server side.
- Even client-side scripts are downloaded from the server.
- Entire solution is centralized.
- Emphasis is on:
  → How application logic is partitioned.
  → Where partitioned units reside.
  → How units of logic should interact.
- Difference lies in the principles used to determine the three primary design considerations.
- Traditional systems create components that reside on one or more application servers.
- Components have varying degrees of fundamental granularity.
- Components on the same server communicate via proprietary APIs.

- RPC protocols are used across servers via proxy stubs.
- Actual references to other physical components can be embedded in programming code (tight coupling).
- SOAs also rely on components.
- Services encapsulate components.
- Services expose specific sets of functionality.
- Functionality can originate from legacy systems or other sources.

A TYPICAL DISTRIBUTED INTERNET ARCHITECTURE


presentation
business
data
synchronous
synchronous
distributed internet
• dumb client (browser)
• no installed logic
• presentation-related processing
• intelligent application & Web server
• presentation logic, business rules, connection pooling
• componentized application logic
• bulk of processing
• database server
• data-related logic, some business rules
• data-related processing, less connections per user

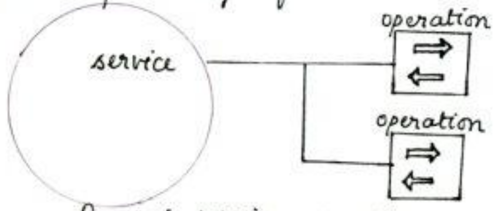
II CHARACTERISTICS OF SOA :

1. SOA is at the core of software oriented computing platform.
2. It improves quality of service.
3. SOA is autonomous.
4. SOA is based on open standards (no lisence)
5. SOA supports vendor diversity.
6. SOA promotes discovery.
7. SOA fosters interoperability.
8. SOA promotes federation.
9. SOA emphasis on reusability.
10. SOA emphasis on extensibility.
11. SOA implements layers of abstraction.
12. SOA supports software oriented business modelling.
13. SOA promotes loose coupling throughout the enterprise.
14. SOA is an acheivable ideal.
15. SOA promotes organisational ability.
16. SOA is an evolution.

III ANATOMY OF SOA :

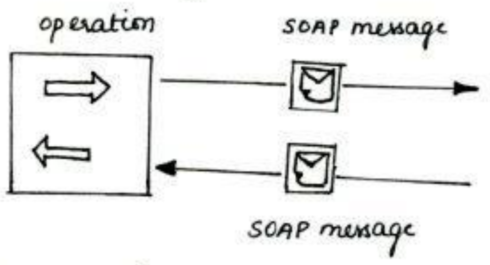
a) Logical components of the Web Services Framework:

As shown in Figure 1 each Web Service contains one or more operations. Note that this diagram introduces a new model to represent operations separately from the service.



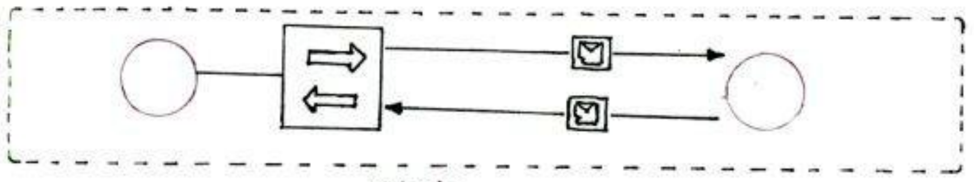
A web service sporting 2 operations

Each operation governs the processing of a specific function the Web service is capable of perfooming. The processing consists of sending and receiving SOAP messages, as shown in Figure 2.



An operation processing outgoing & incoming SOAP messages

By composing these parts, Web services form an activity through which they can collectively automate a task.



activity

A basic communications scenario between web Services

b) <u>Logical components of automation logic</u>:

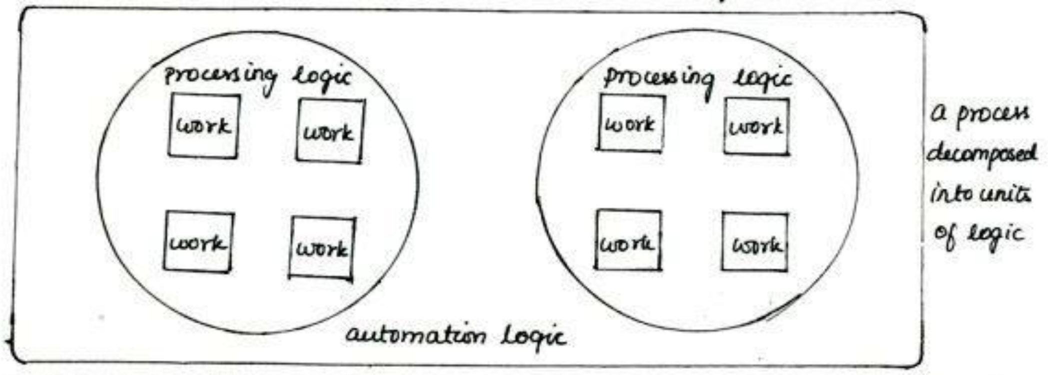
The fundamental parts of the framework are,

- SOAP messages
- Web service operations
- Web services
- Activities

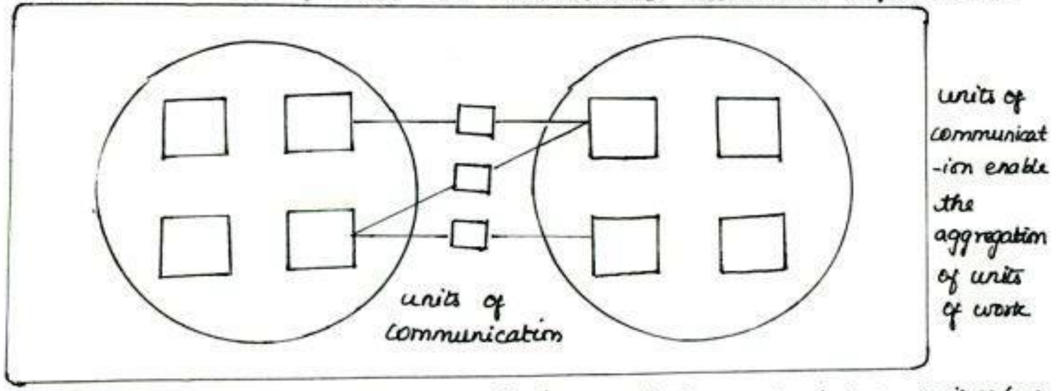
The latter three items represent units of logic that perform work and communication using SOAP messages. To better illustrate this in a service-oriented perspective, let's replace these terms with new ones, as follows:

messages
operations
services
process (and process instances)

messages = units of communication
operations = units of work
service = units of processing logic (collections of units of work)
processes = units of automation logic (coordinated aggregation of units of work)



A primitive view of how SOA modularizes automation logic into units

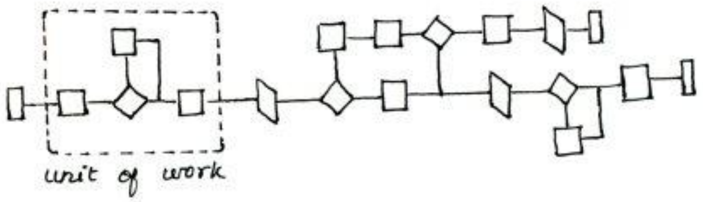

units of communication
units of communicat-ion enable the aggregation of units of work

A primitive view of how units of communication enable interaction between units of logic.

c) Components of SOA :

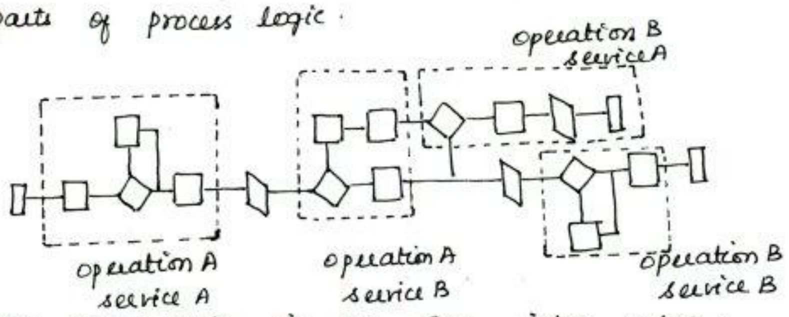
- A message represents the data required to complete some or all parts of a unit of work.
- An operation represents the logic required to process messages in order to complete a unit of work

The scope of an operation within a process.



- A service represents a logically grouped set of operations capable of performing related units of work.
- A process contains the business rules that determine which service operations are used to complete a unit of automation. In other words, a process represents a large piece of work that requires the completion of smaller units of work.

Operations belonging to different services representing various parts of process logic.



(d) <u>How components in an SOA inter-relate:</u>

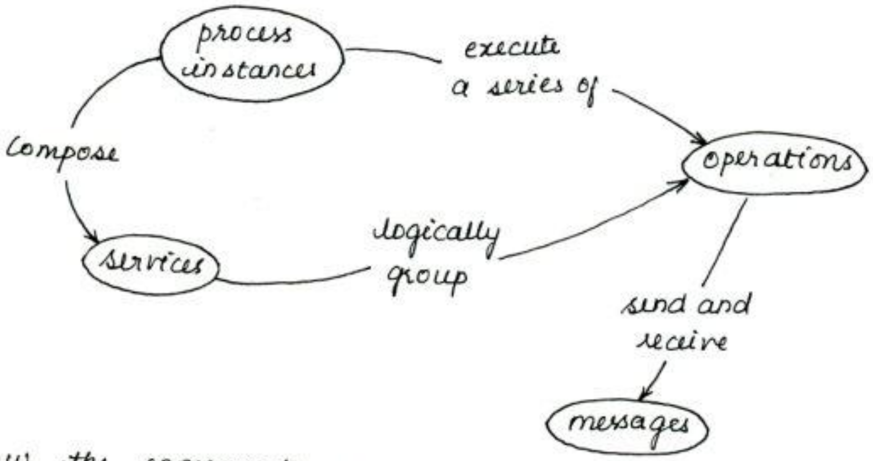
- An operation sends and receives messages to perform work.
- An operation is therefore mostly defined by the messages it processes.
- A service groups a collection of related operations.
- A service is therefore mostly defined by the operations that comprise it.
- A process instance can compose services.
- A process instance is not necessarily defined by its

services because it may only require a subset of the functionality offered by the services.

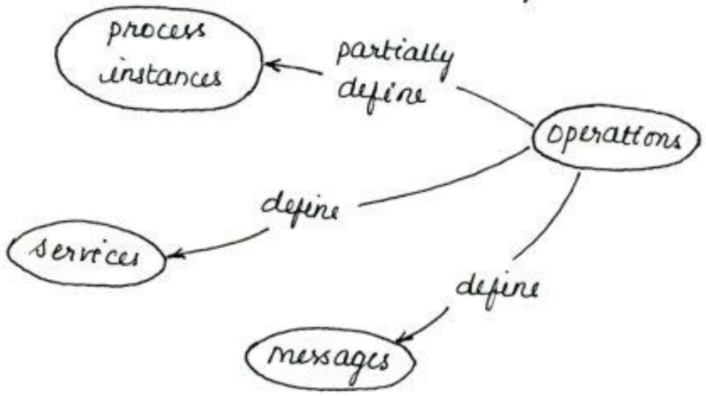
- A process instances invokes a unique series of operations to complete its automation.
- Every process instance is therefore partially defined by the service operations it uses.

Following figures further illustrate these relationships:

- How the components of an SOA relate:



- How the components of an SOA define each other:



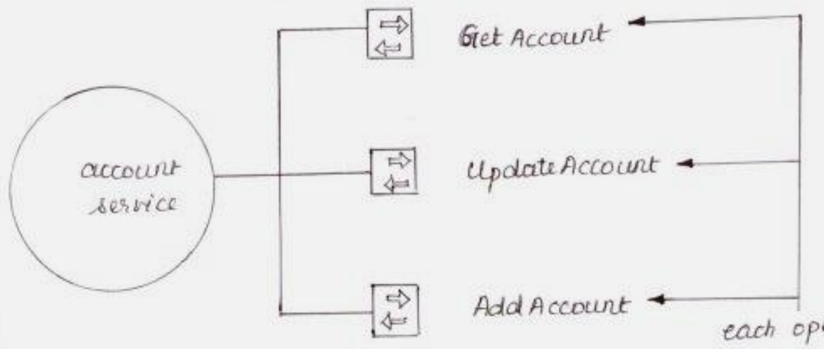
COMMON PRINCIPLES OF SERVICE - ORIENTATION :

- Services are reusable.
- Services share a formal contract.
- Services are loosely coupled.
- Services abstract underlying logic.
- Services are composable.
- Services are stateless.
- Services are discoverable.

i) Services are reusable :

- Regardless of whether immediate reuse opportunities exist, services are designed to support potential reuse.
- Service - oriented encourages reuse in all services.
- By applying design standards that require reuse accomodate future requirements with less development effort.

A reusable service exposes reusable operations.



each operation is reusable by multiple service requestors

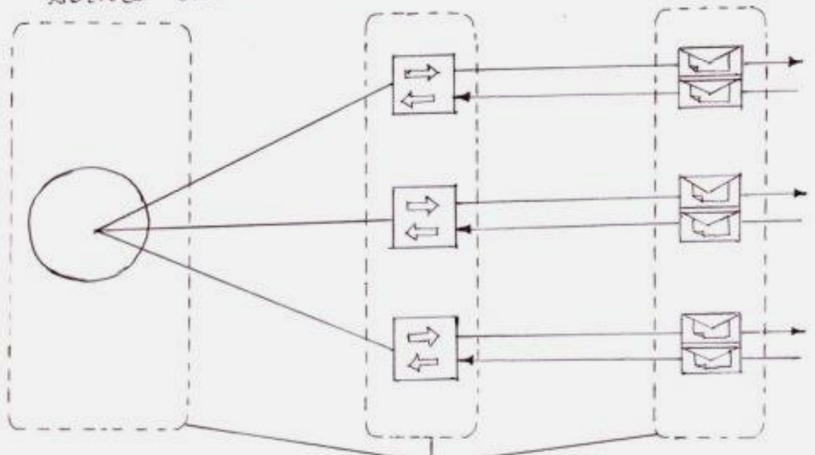
ii) Services share a formal contract :

- Service contracts provide a formal definition of :
  → The service endpoint
  → Each service operation
  → Every input and output message supported by each operation.

-> Rules and characteristics of the service and its operations.

Service contacts define almost all of the primary parts of an SOA.

Service contracts formally define the service, operation, and message components of a service-oriented architecture.

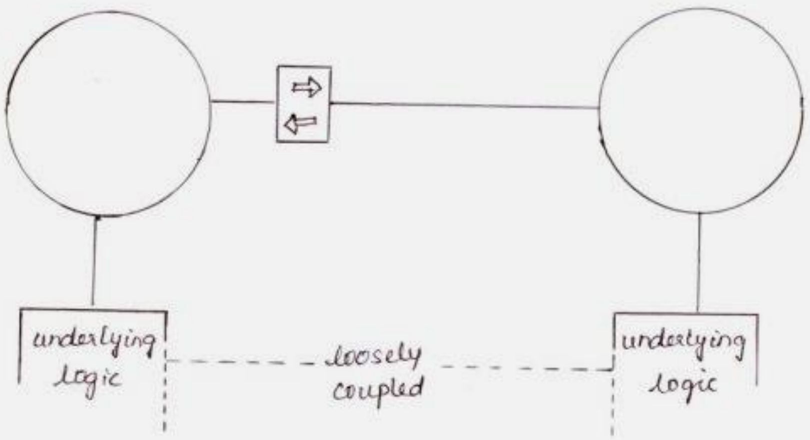


defined by service contract.

iii) Services are loosely coupled:

- Services must be designed to interact without the need for tight, cross-service dependencies.

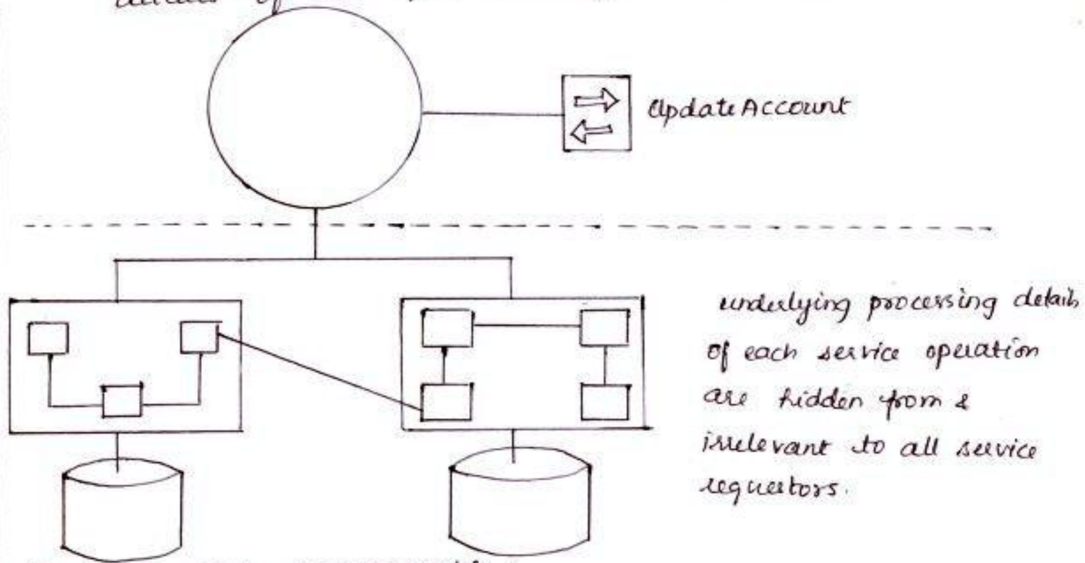
Services limit dependencies to the service contract, allowing underlying provider & requestor logic to remain loosely coupled.



iv) Services abstract underlying logic:

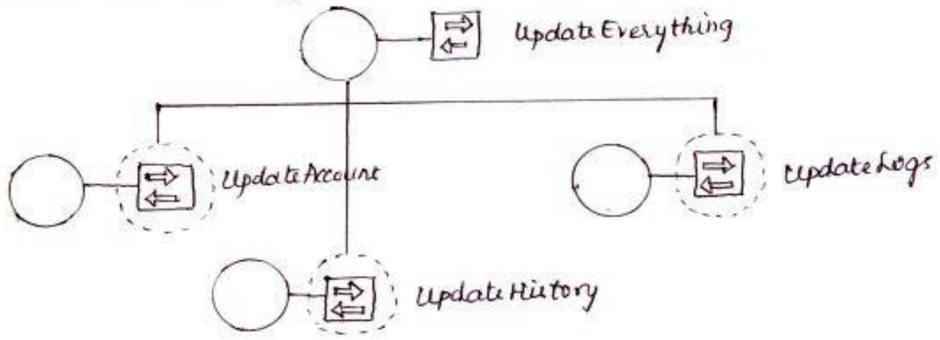
The only part of a service that is visible to the outside world is what is exposed via the service contract. Underlying logic, beyond what is expressed in the descriptions that comprise the contract, is invisible and irrelevant to service requestors.

Service operations abstract the underlying details of the functionality they expose.


UpdateAccount
underlying processing details of each service operation are hidden from & irrelevant to all service requestors.

v) Services are composable:

Services may compose other services. This allows logic to be represented at different levels of granularity and promotes reusability and the creation of abstraction layers.


UpdateEverything
UpdateAccount
UpdateLogs
UpdateHistory