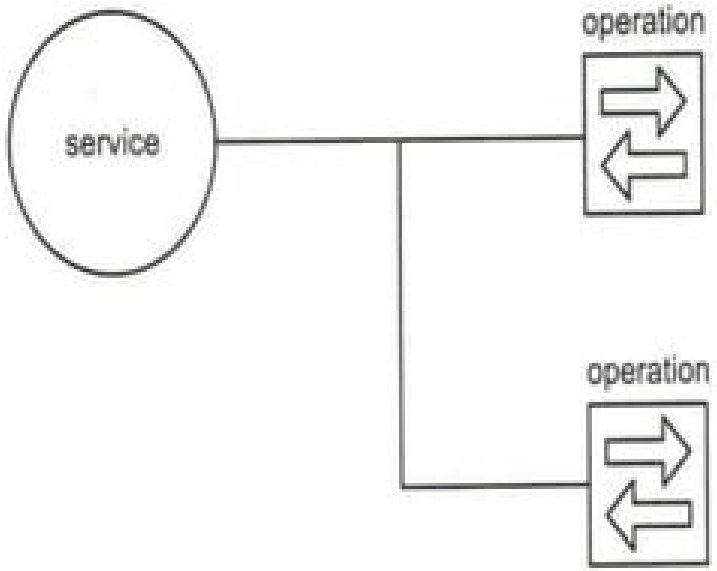
ANATOMY OF SOA

HOW COMPONENTS IN AN SOA INTERRELATE

PRINCIPLES OF SERVICE ORIENTATION

Logic components of the Web services framework

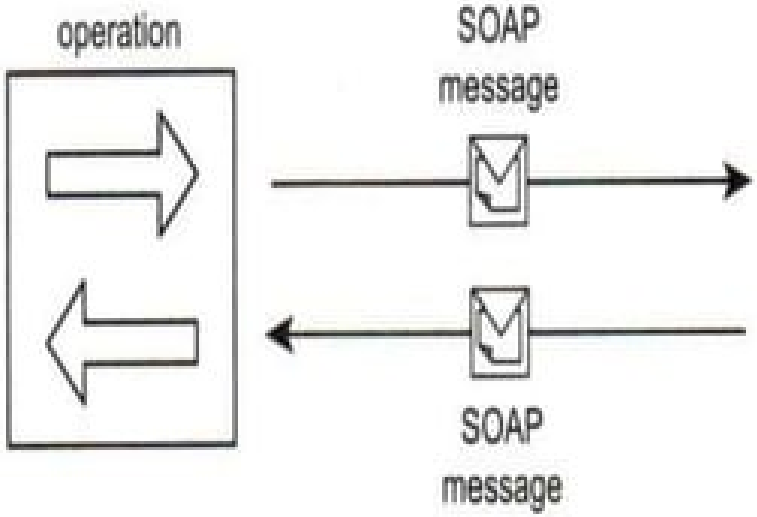
- Web services contain one or more operations.
- Figure shows an example



A Web service sporting two operations.

Logic components of the Web services framework

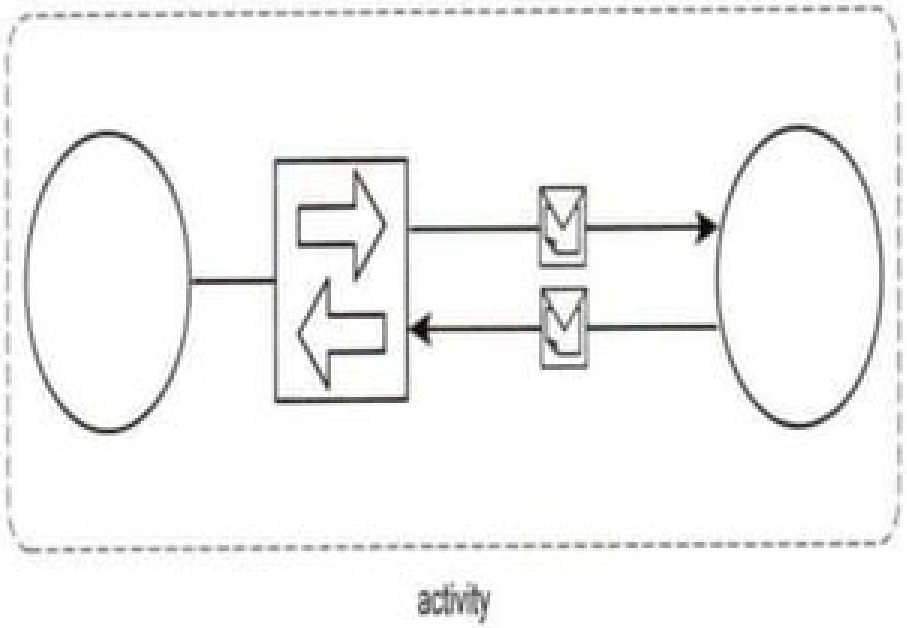
- Each operation governs the process of a **specific function** the web service is capable of performing.
- Figure gives an example of an operation sending and receiving SOAP messages



An operation processing outgoing and incoming SOAP messages.

Logic components of the Web services framework

- Web services form an activity through which they can collectively automate a task.
- Figure shows an example



A basic communications scenario between Web services.

Logic components of automation logic

- Fundamental parts of the framework
  - SOAP messages
  - Web service operations
  - Web services
  - Activities
- Renamed terms
  - Messages
  - Operations
  - Services
  - Processes
- Activity has been changed because it uses a different context when modeling service-oriented business processes.

Logic components of automation logic

- Messages = units of communication
- Operations = units of work
- Services = units of processing logic
- Processes = units of automation logic

Logic components of automation logic

- The purpose of these views is to express the process, services and operations.
- It also provides a flexible means of partitioning and modularizing the logic.
- These are the most basic concepts that underlies service-orientation.

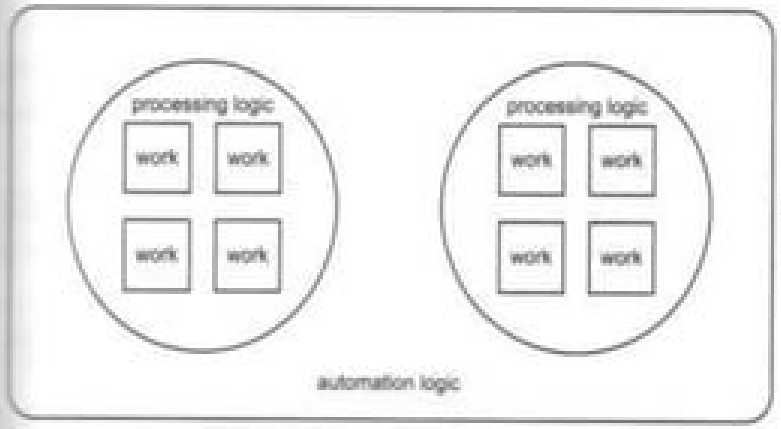


Figure 8.7
A primitive view of how SOA modularizes automation logic into units.

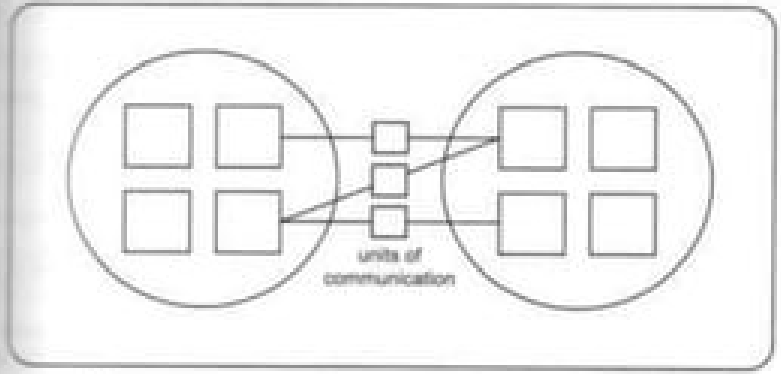


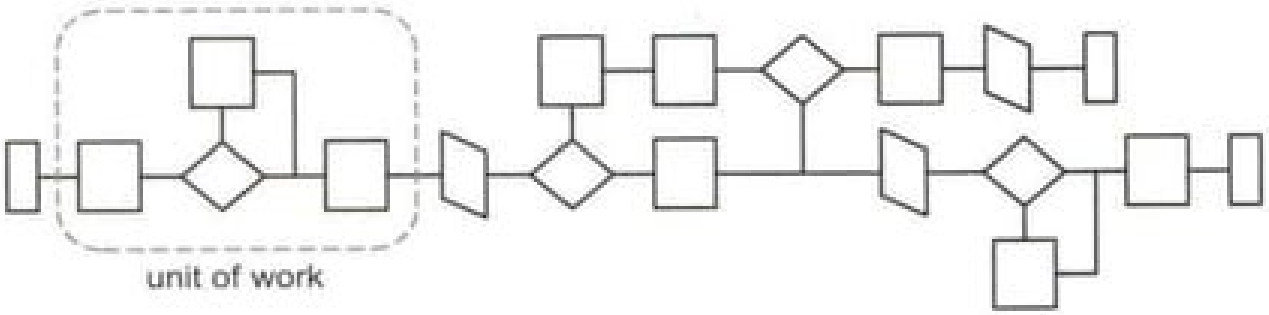
Figure 8.8
A primitive view of how units of communication enable interaction between units of logic.

Components of an SOA

- Message
  - A message represents the data required to complete some or all parts of a unit of work.

Components of an SOA

- Operation
  - An operation represents the logic required to process messages in order to complete a unit of work.

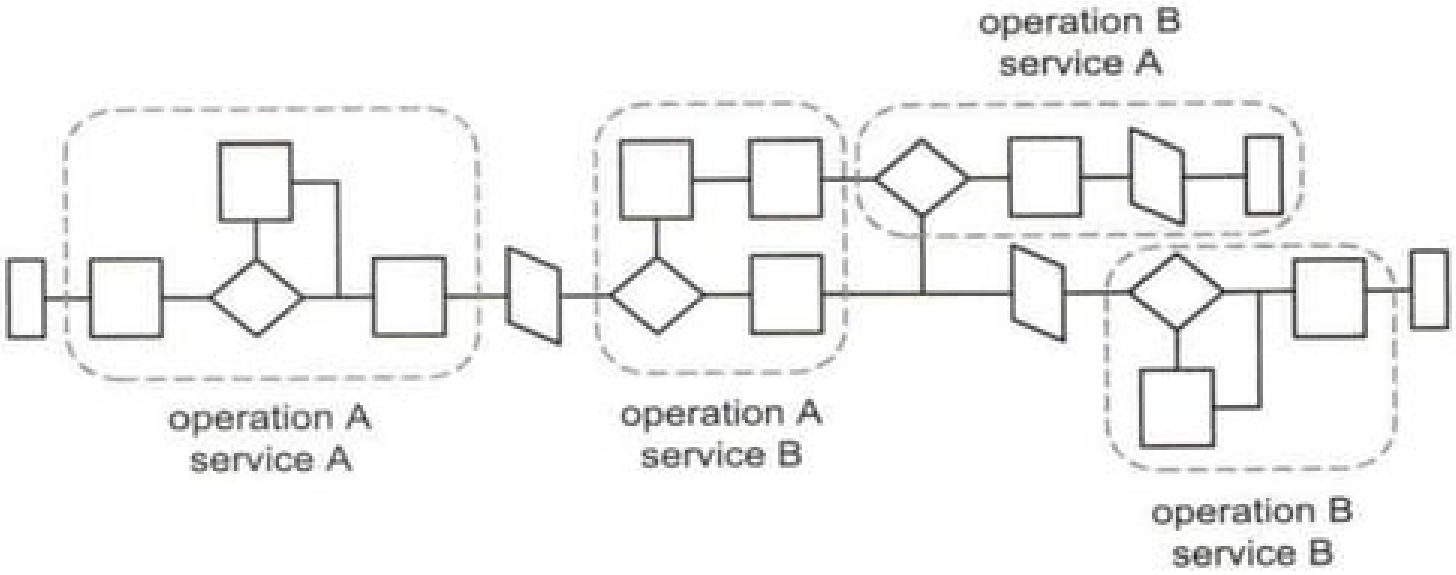


The scope of an operation within a process.

Components of an SOA

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

Components of an SOA



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

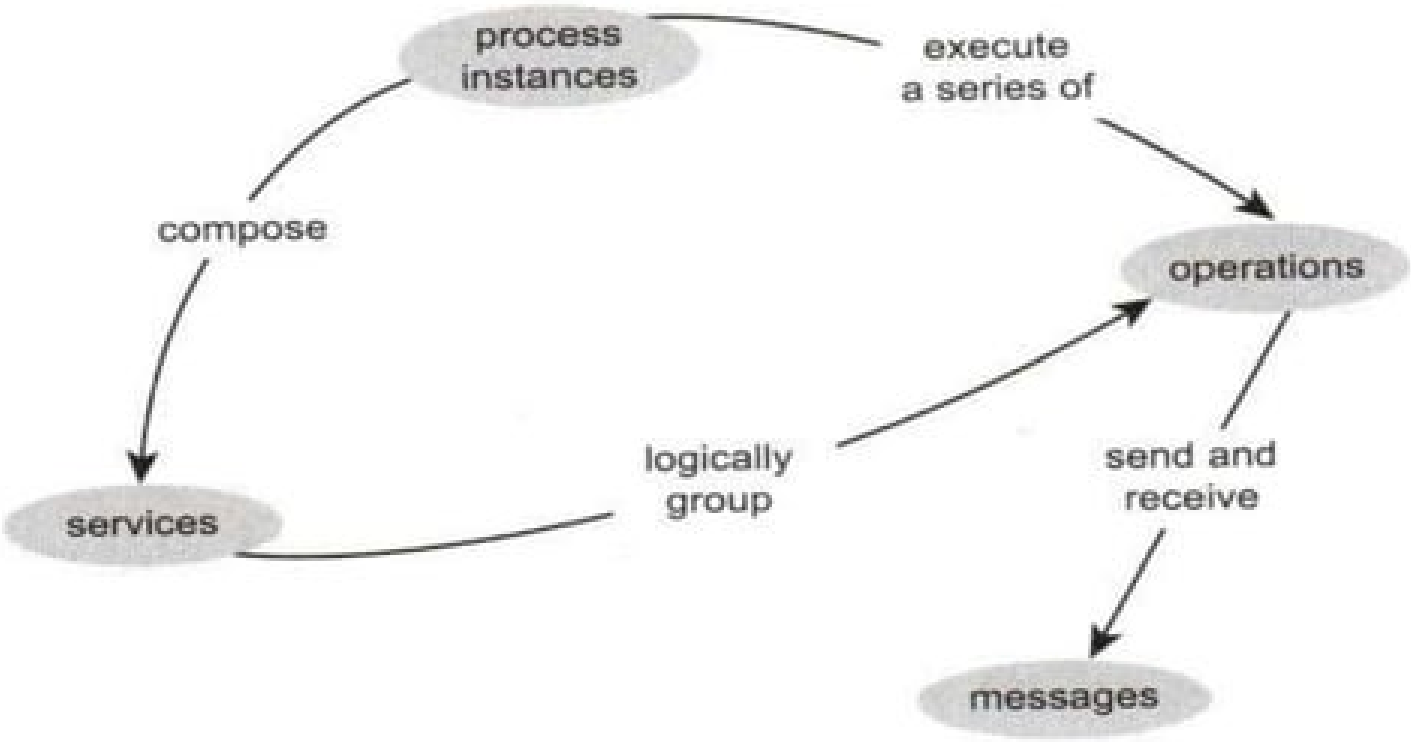
How components in an SOA inter-relate

- An operation sends and receives messages to perform work.
- An operation is therefore mostly defined by the message it processes.
- A service group is a collection of related operations.
- A service is therefore mostly defined by the operations that comprise it.

How components in an SOA inter-relate

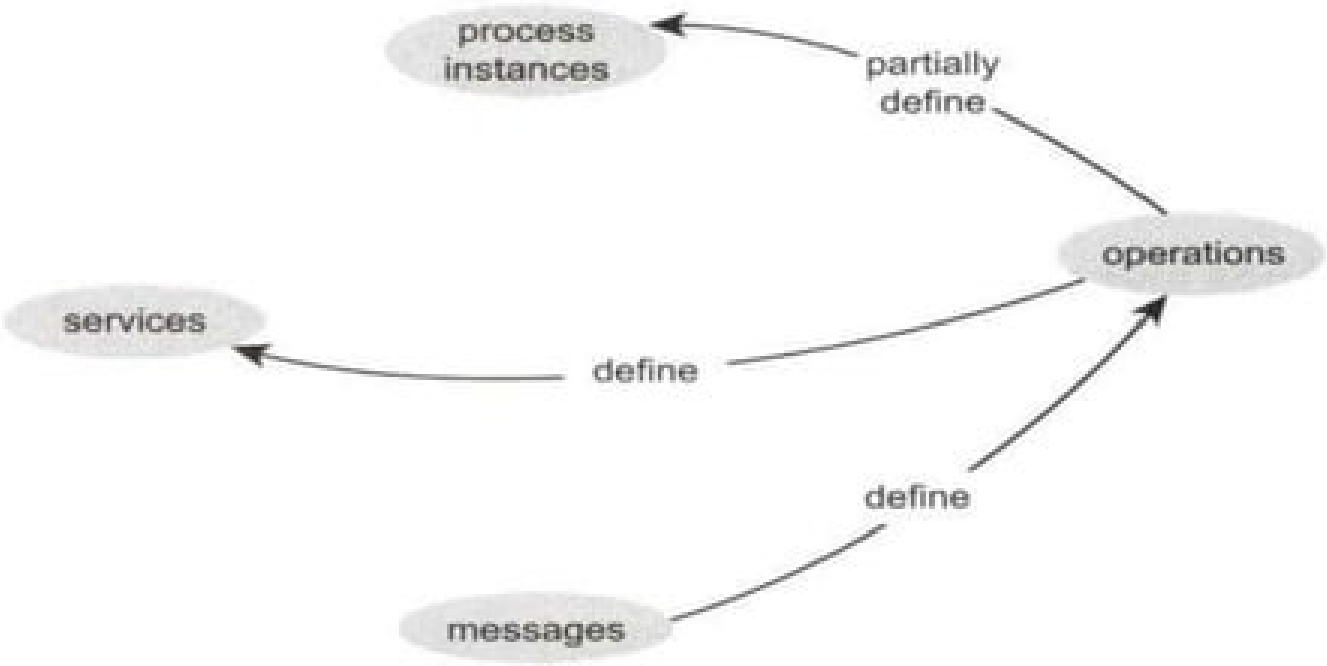
- A process instance can compose service.
- A process instance is not necessarily defined by its service because it may only require a subset of the functionality offered by the services.
- A process instance invokes a unique series of operations to complete its automation.
- Every process instance is therefore partially defined by the service operation it uses.

How components in an SOA inter-relate



How the components of a service-oriented architecture relate.

How components in an SOA inter-relate



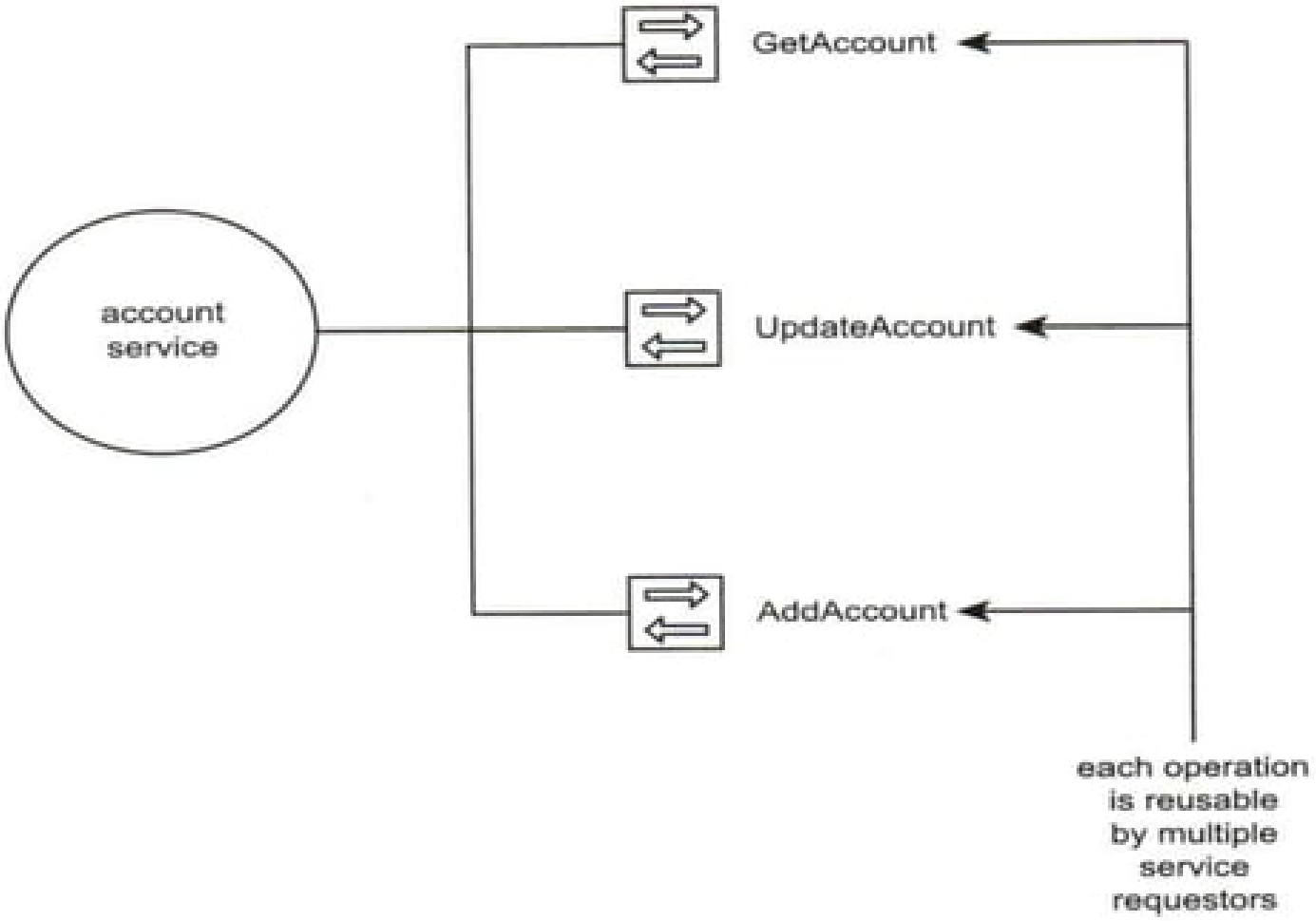
How the components of a service-oriented architecture can define each other.

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 autonomous
- Services are stateless
- Services are discoverable

Services are reusable

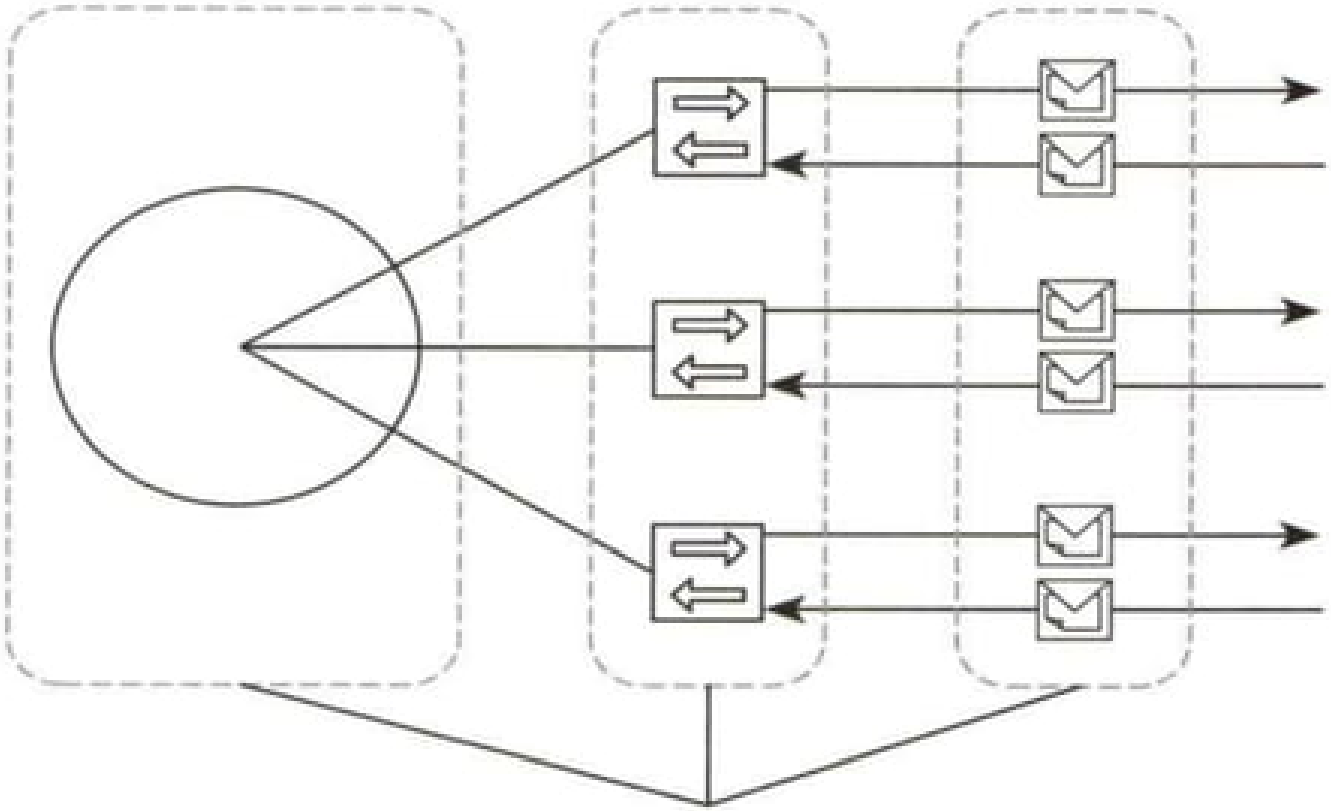
- Regardless of whether immediate reuse opportunities exist, services are designed to support potential reuse.
- Service-oriented encourages reuse in all services.
- By applying design standards that require reuse accommodate future requirements with less development effort



A reusable service exposes reusable operations.

Services share a formal contract

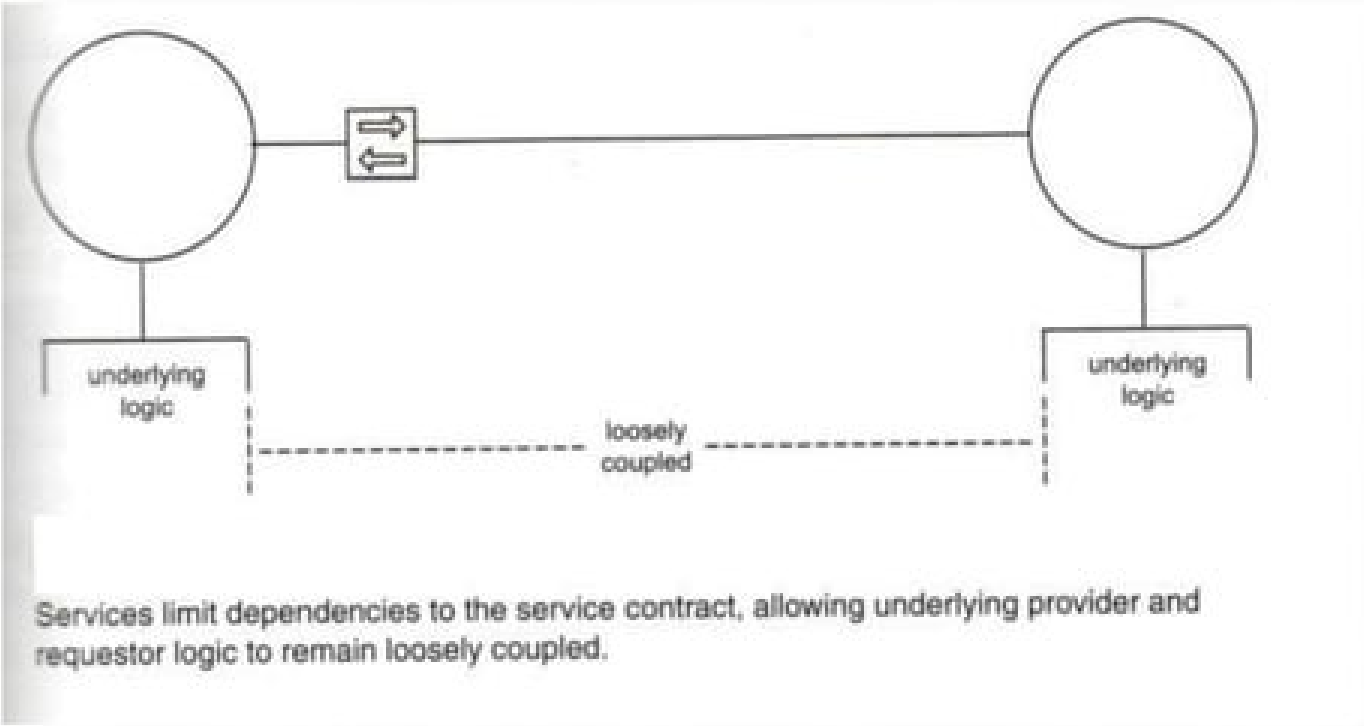
- For services to interact, they need to share formal contract that describe each service and define the terms of information exchange.
- Service contracts provide a formal definition of:
  - The service endpoint
  - Each service operation
  - Every input and output message supported by each operation
  - 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.

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 and requestor logic to remain loosely coupled.

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

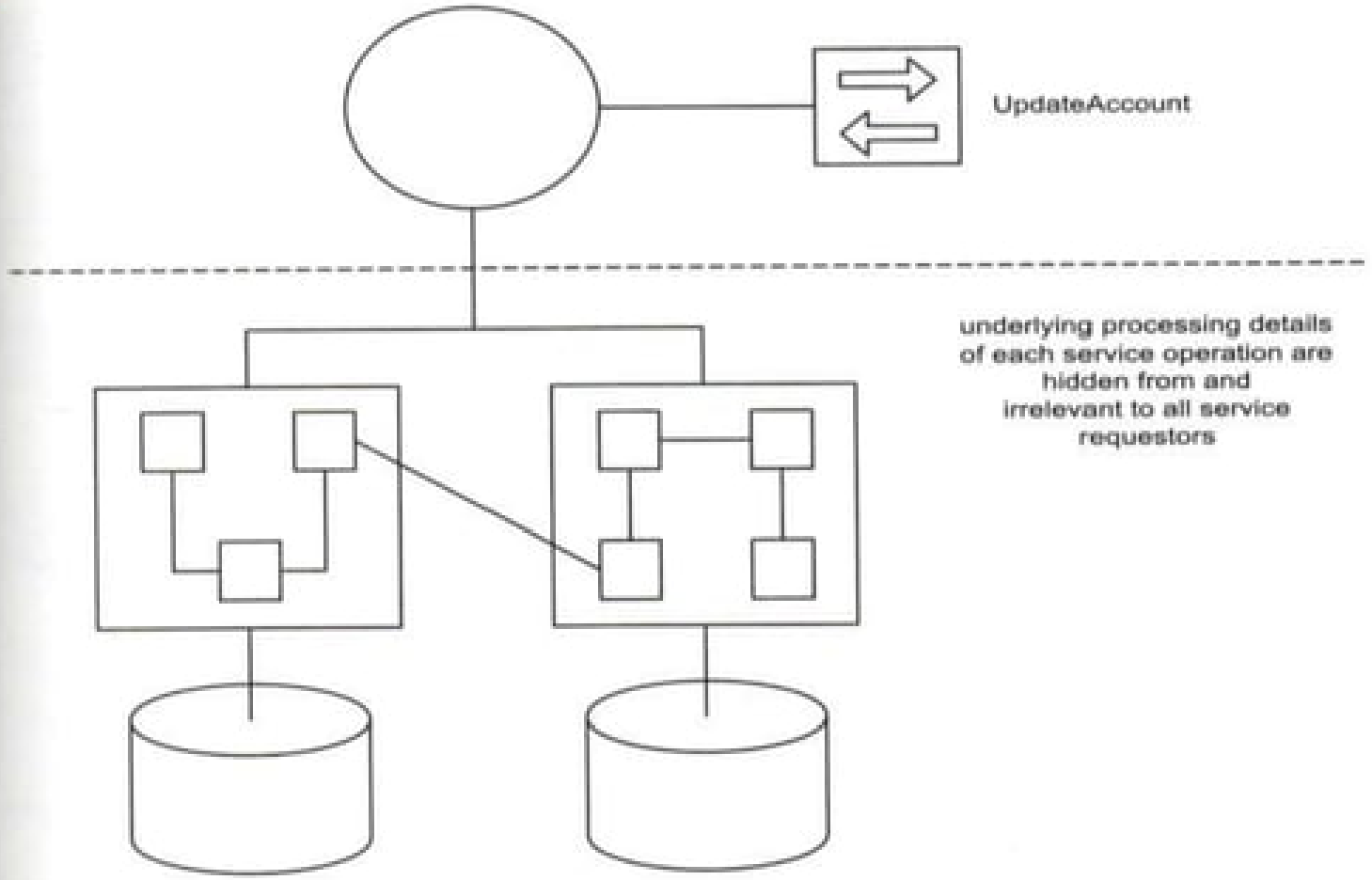


Figure 8.17
Service operations abstract the underlying details of the functionality they expose.

Services are composable

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

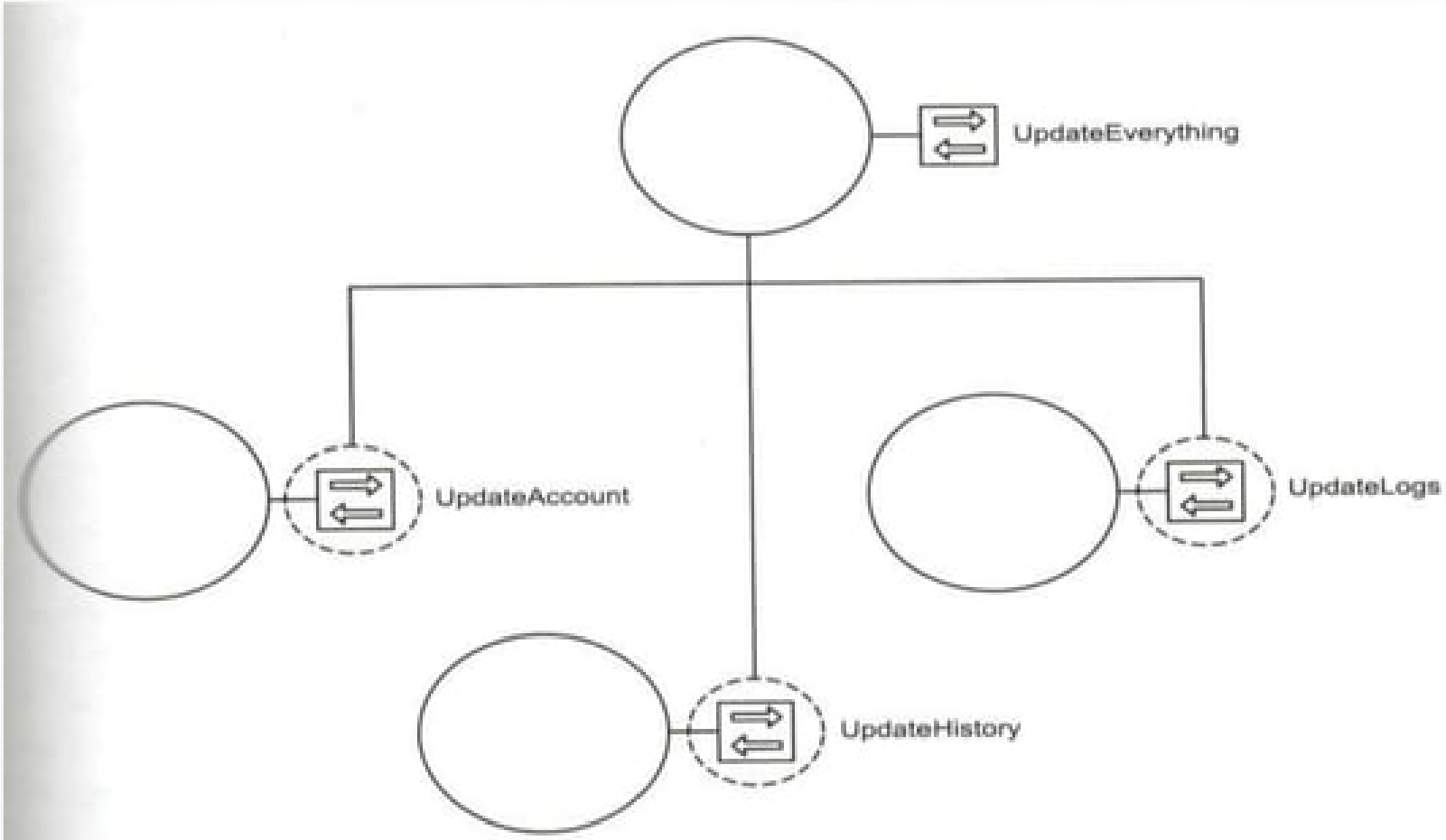
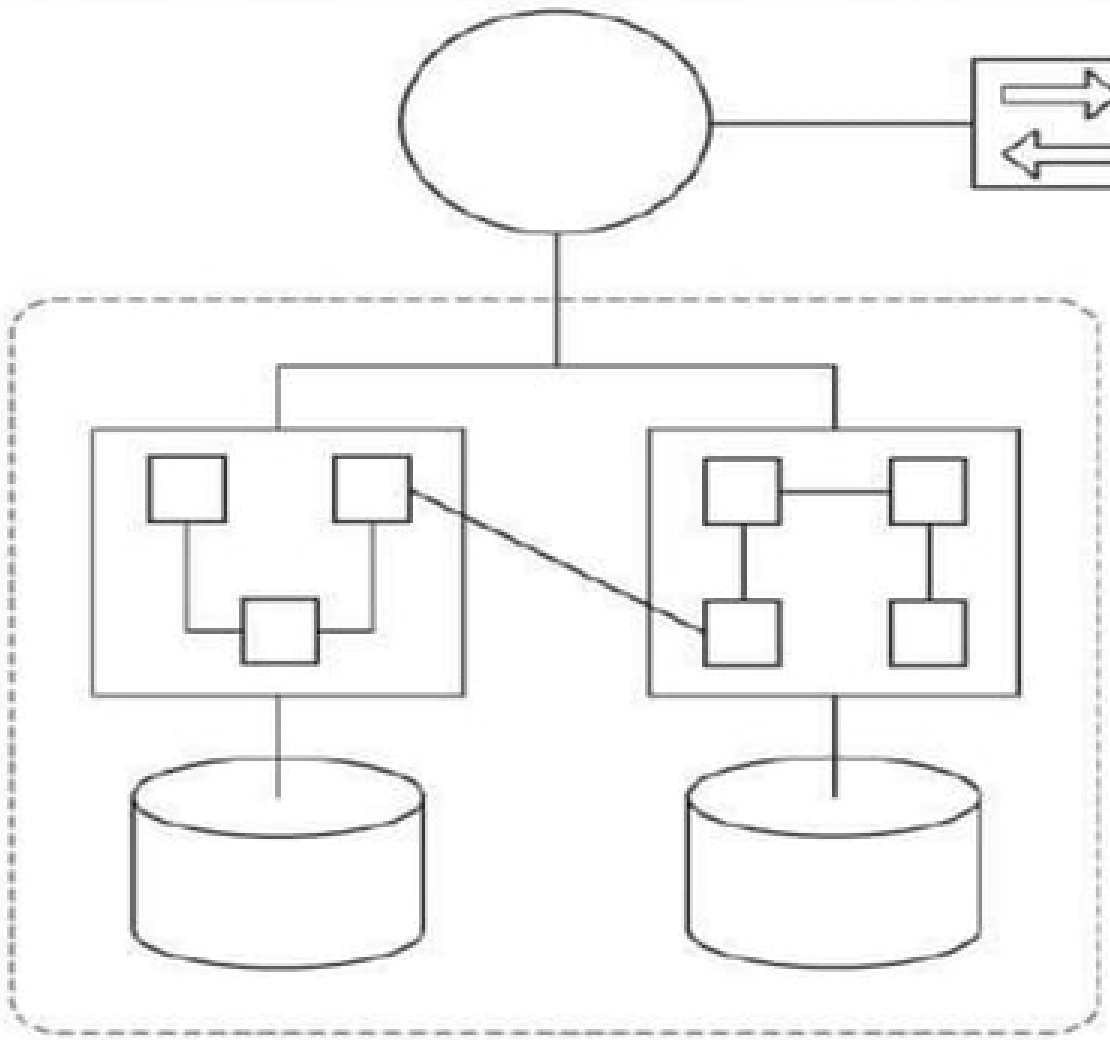


Figure 8.19
The UpdateEverything operation encapsulating a service composition.

Services are autonomous

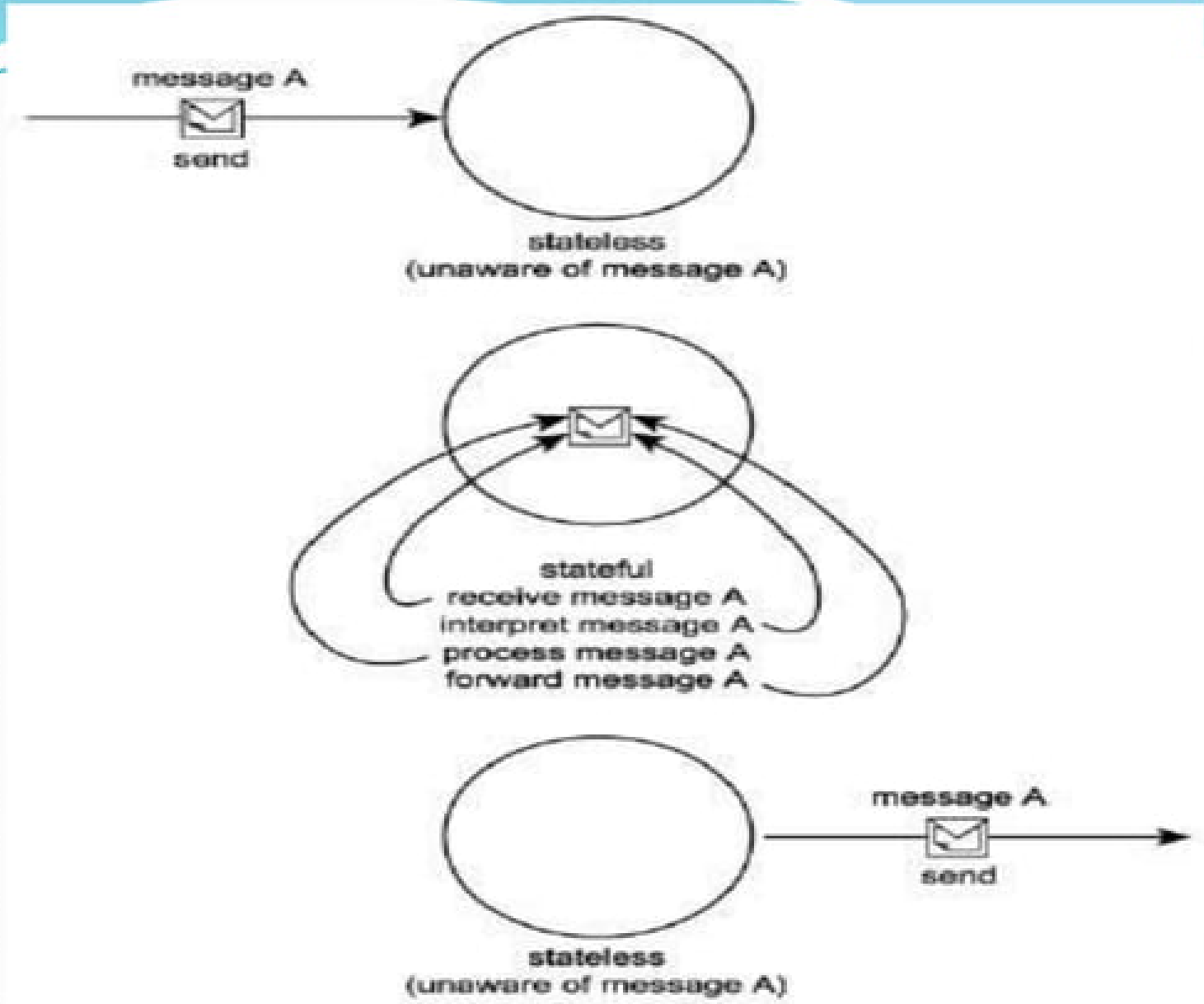
- The services reside inside a well defined boundary and for the successful execution of a service it does not depend on other service for it to execute its governance.



upon execution, the service has governance over its underlying application logic

Services are stateless

- Service are not allowed to store the state information as it will not allow the service to be loosely coupled.
- Services should be designed to maximize statelessness even if that means deferring state management elsewhere.
- Statelessness is a preferred condition for services and one that promotes reusability and scalability.


message A
send
stateless
(unaware of message A)
stateful
receive message A
interpret message A
process message A
forward message A
message A
send
stateless
(unaware of message A)

Services are discoverable

- The service should allow their description to be searched and understood by humans.

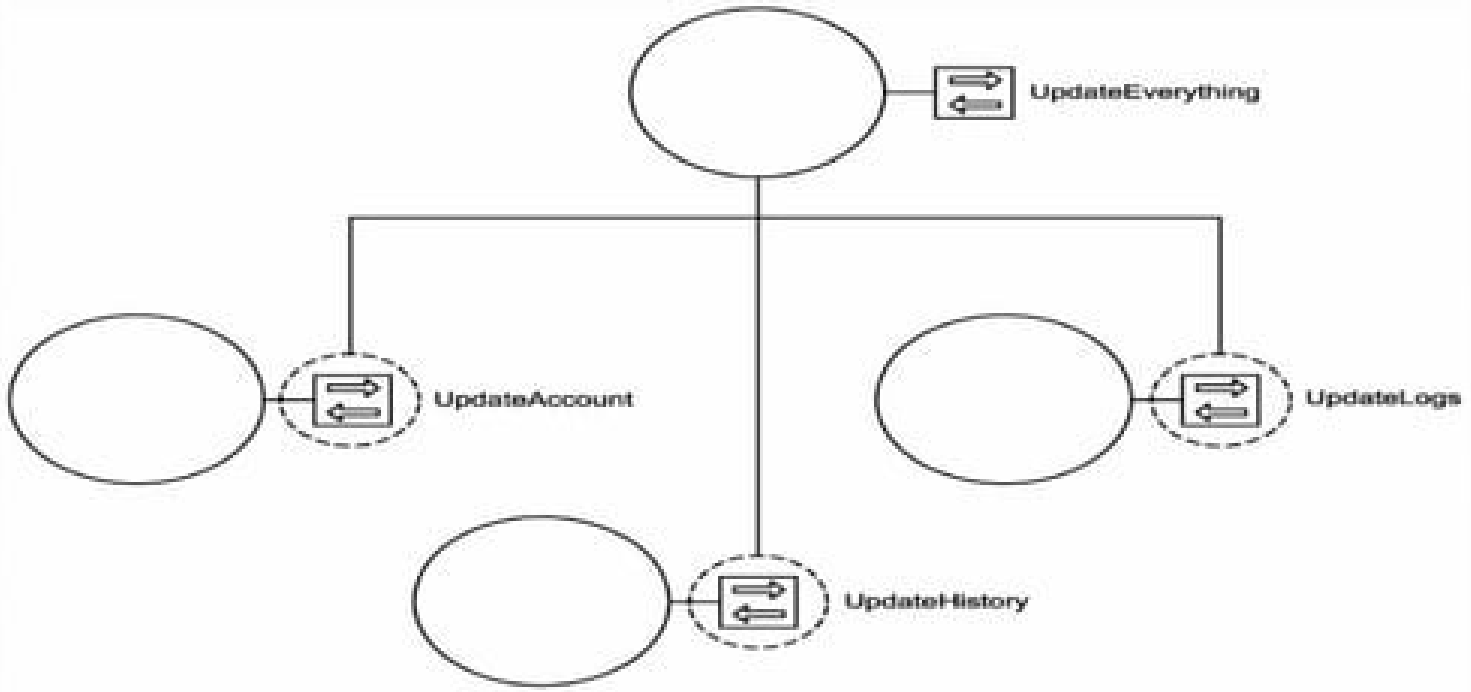


Figure :The UpdateEverything operation encapsulating a service composition.