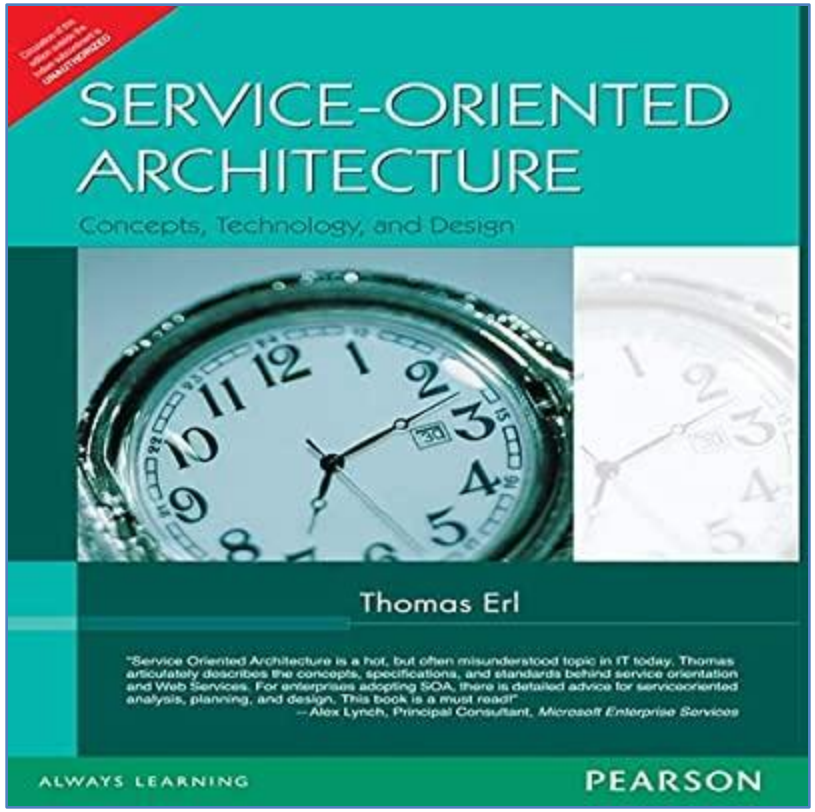
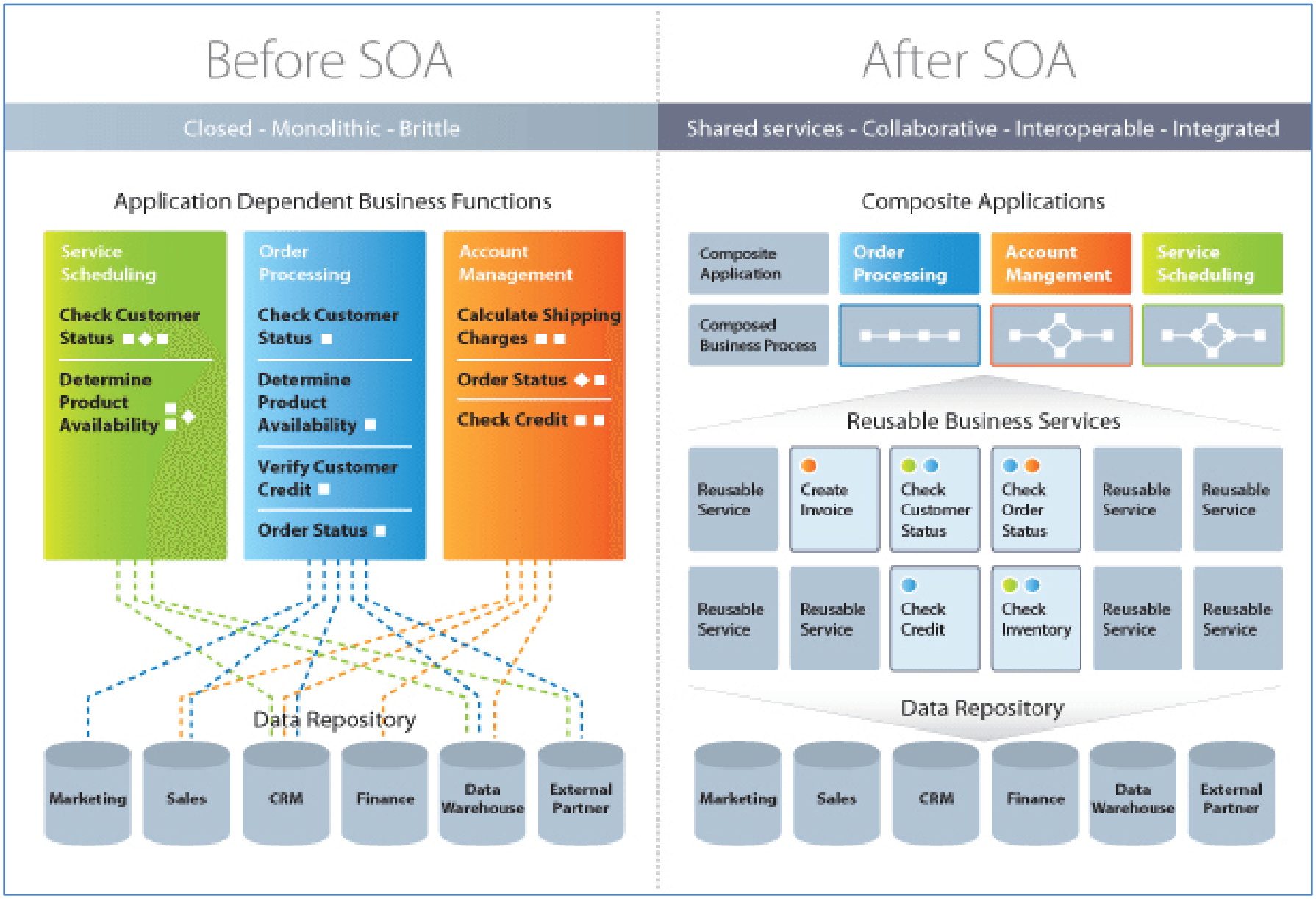
Service-Oriented Computing

Two sides of SOC

Our guide!


SERVICE-ORIENTED ARCHITECTURE
Concepts, Technology, and Design
Thomas Erl
"Service Oriented Architecture is a hot, but often misunderstood topic in IT today. Thomas articulately describes the concepts, specifications, and standards behind service orientation and Web Services. For enterprises adopting SOA, there is detailed advice for service-oriented analysis, planning, and design. This book is a must read!"
– Alex Lynch, Principal Consultant, Microsoft Enterprise Services
ALWAYS LEARNING
PEARSON


Before SOA
Closed - Monolithic - Brittle
Application Dependent Business Functions
Service Scheduling
Check Customer Status
Determine Product Availability
Order Processing
Check Customer Status
Determine Product Availability
Verify Customer Credit
Order Status
Account Management
Calculate Shipping Charges
Order Status
Check Credit
Data Repository
Marketing
Sales
CRM
Finance
Data Warehouse
External Partner
After SOA
Shared services - Collaborative - Interoperable - Integrated
Composite Applications
Composite Application
Order Processing
Account Mangement
Service Scheduling
Composed Business Process
Reusable Business Services
Reusable Service
Create Invoice
Check Customer Status
Check Order Status
Reusable Service
Reusable Service
Reusable Service
Reusable Service
Check Credit
Check Inventory
Reusable Service
Reusable Service
Data Repository
Marketing
Sales
CRM
Finance
Data Warehouse
External Partner

What is “Service Oriented”?

- **Separating concern**
  - Logic required to solve a large problem can be better constructed, carried out, and managed if it is decomposed into a collection of smaller, related pieces.
  - Each of these pieces addresses a concern or a specific part of the problem.
  - SOA differs from other approaches in the manner in which it achieves separation

A service-oriented analogy

Example of service orientation

- A city has many service oriented outlets
- Each outlet provides distinct service to many customers
- Collectively these outlets form a business community
- Why do we need different outlets?
- What should be the amount of dependency among the outlets?

Service Oriented Architecture

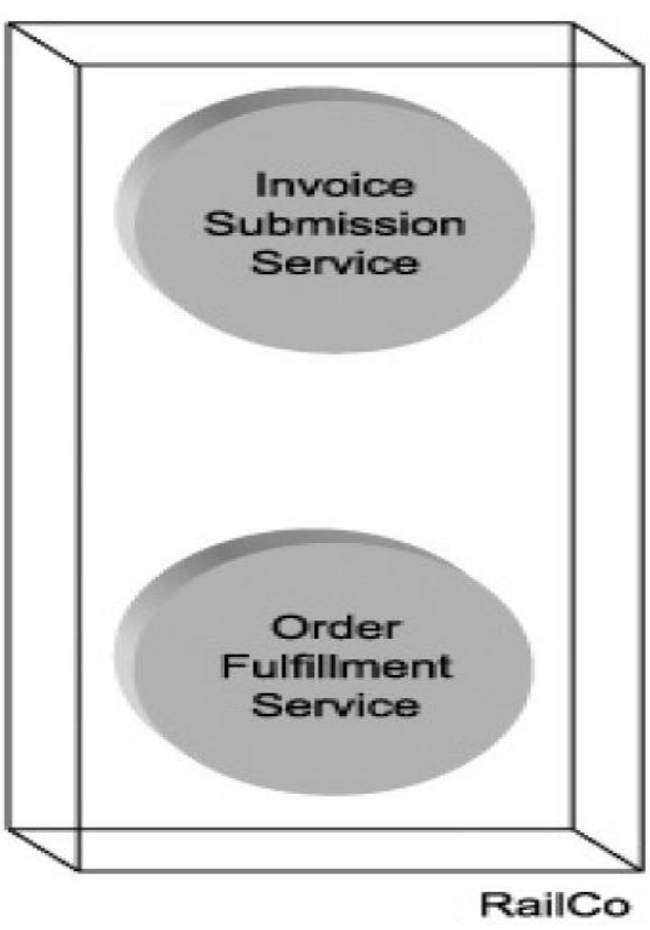
- "It is a term that represents a model in which automation logic is decomposed into smaller, distinct units of logic"
- Individually, these units can be distributed and collectively they form a large piece of automation logic
- *"Individual units exist autonomously, yet not isolated from each other"*
- These units of logic are called services

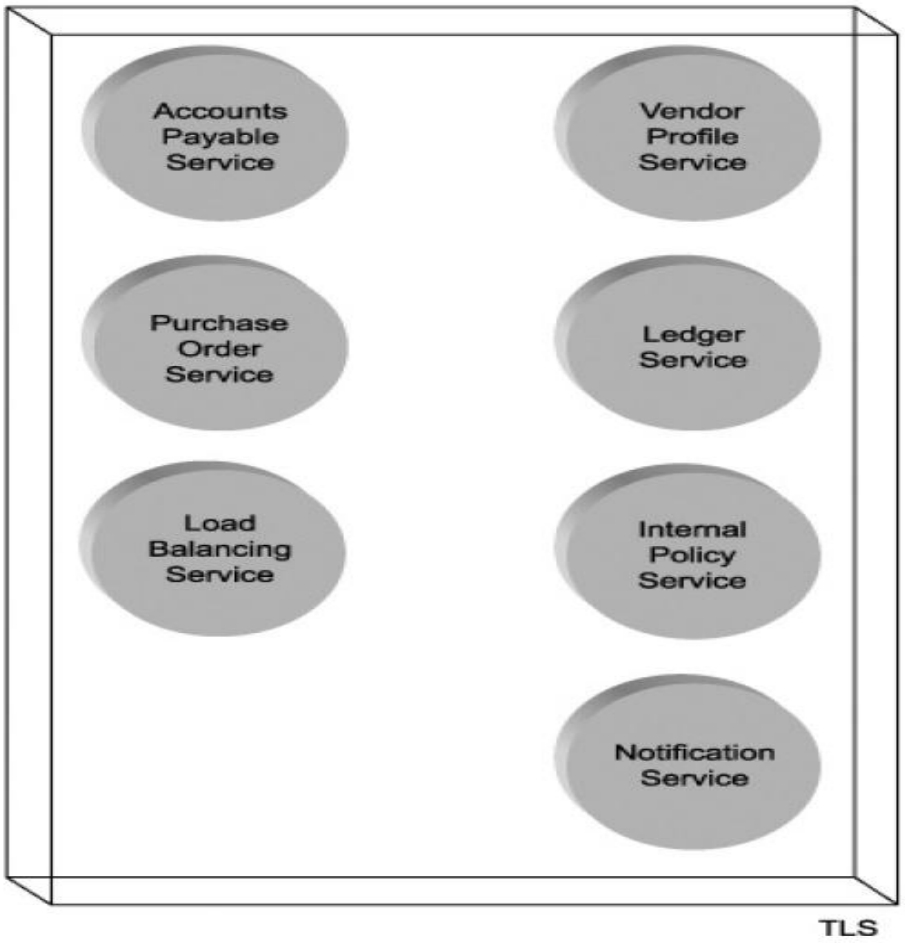
Service Oriented Architecture

- *Service-oriented separation*
  - Avoid tight connections that result in constrictive inter-dependencies.
  - Allow them to evolve and grow relatively independent from each other.
  - Must follow baseline conventions/principles.

What are the real world examples of units of logic (i.e. services)?

Case Study

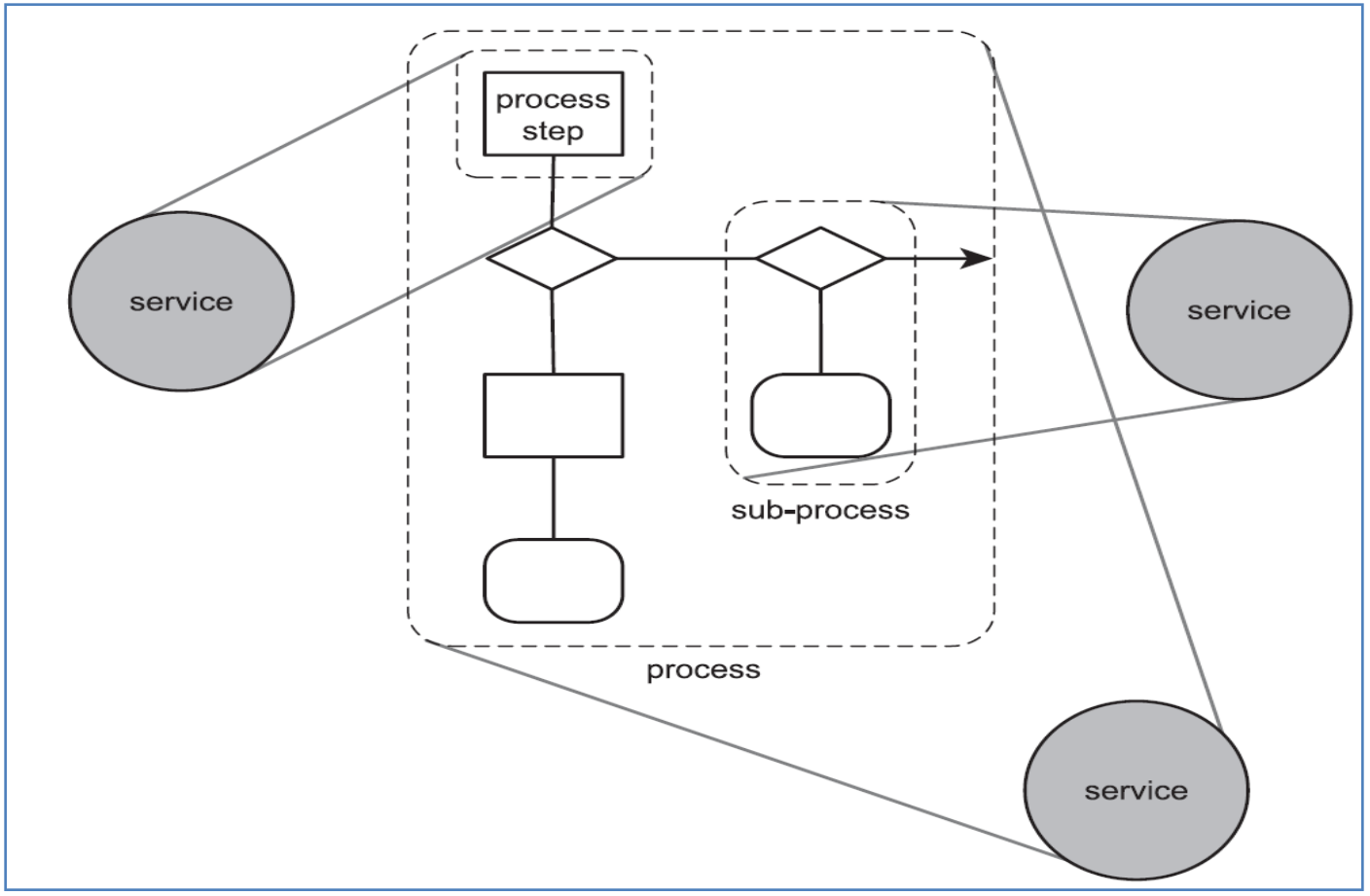

Invoice Submission Service
Order Fulfillment Service
RailCo


Accounts Payable Service
Vendor Profile Service
Purchase Order Service
Ledger Service
Load Balancing Service
Internal Policy Service
Notification Service
TLS

How services encapsulate logic

- Services can encapsulate varying amounts of logic.
  - A small process step
  - A sub process (collection of steps)
  - Entire process

How services encapsulate logic

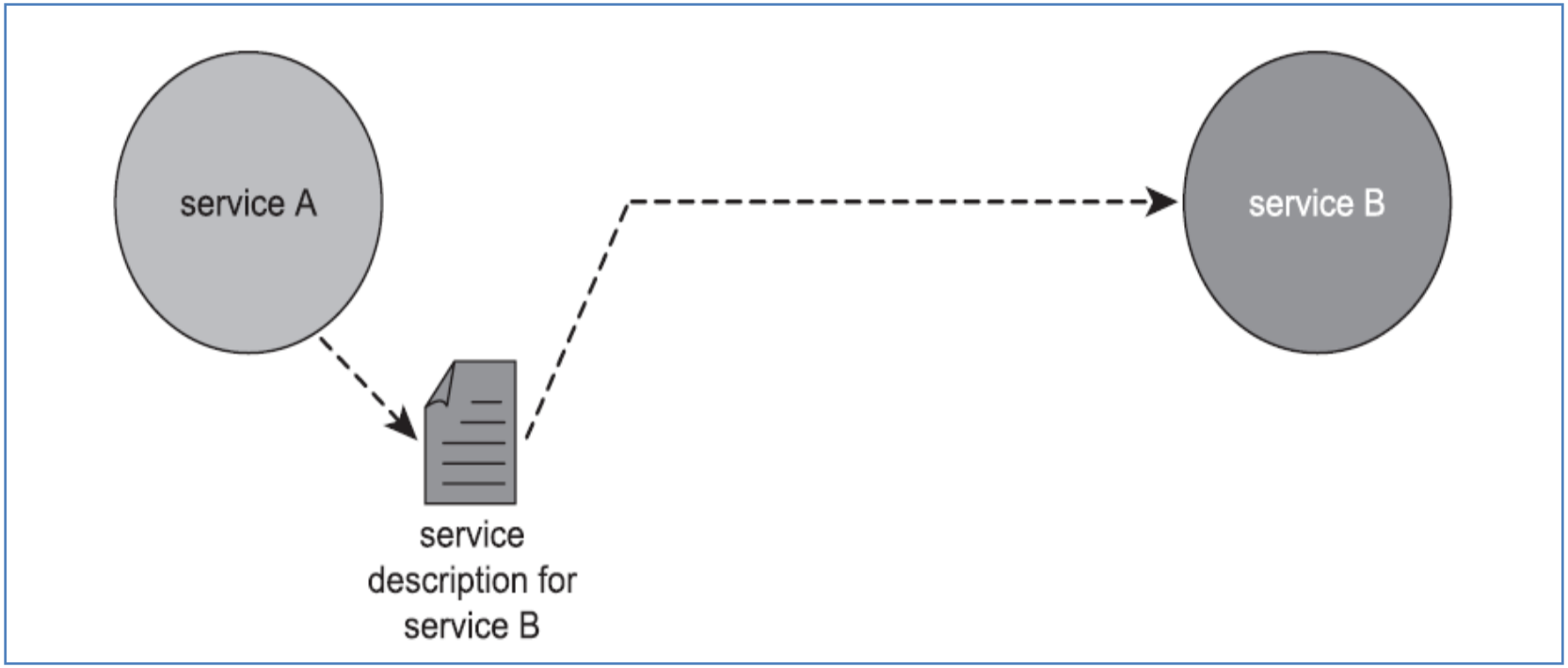


How services relate

- Within SOA, services can be used by other services or other programs.
- For services to interact, they must **be aware of each other**.
- This awareness is achieved through the use of ***service descriptions***.
  - *Includes service name, data expected and returned by service etc.*

How services relate

Because A has access to service B's service description, service A has all of the information it needs to communicate with service B.



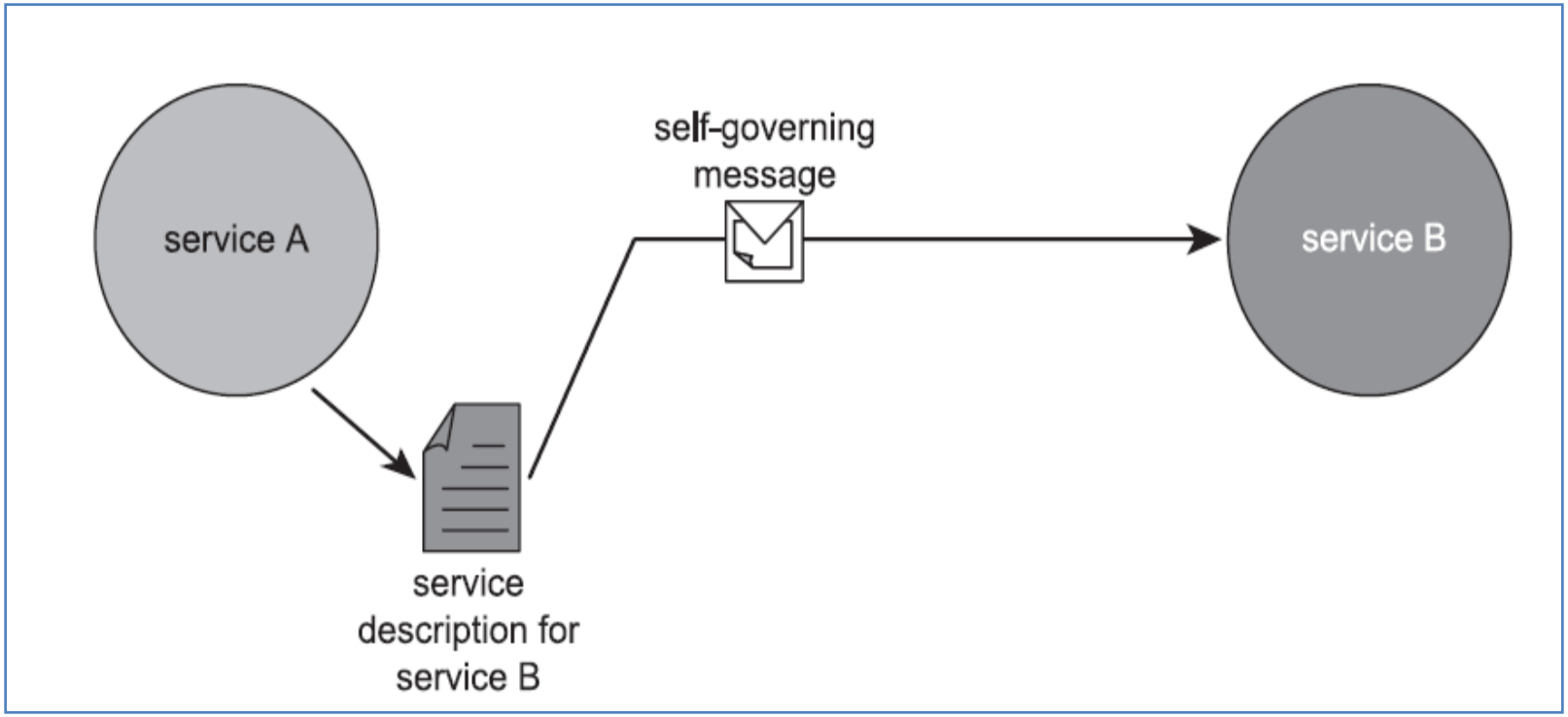
How services relate

- *"Service description helps to achieve loose coupling between services"*

- Once the services be aware of each other, next aspect required is a way to communicate

- This is provided by messaging framework

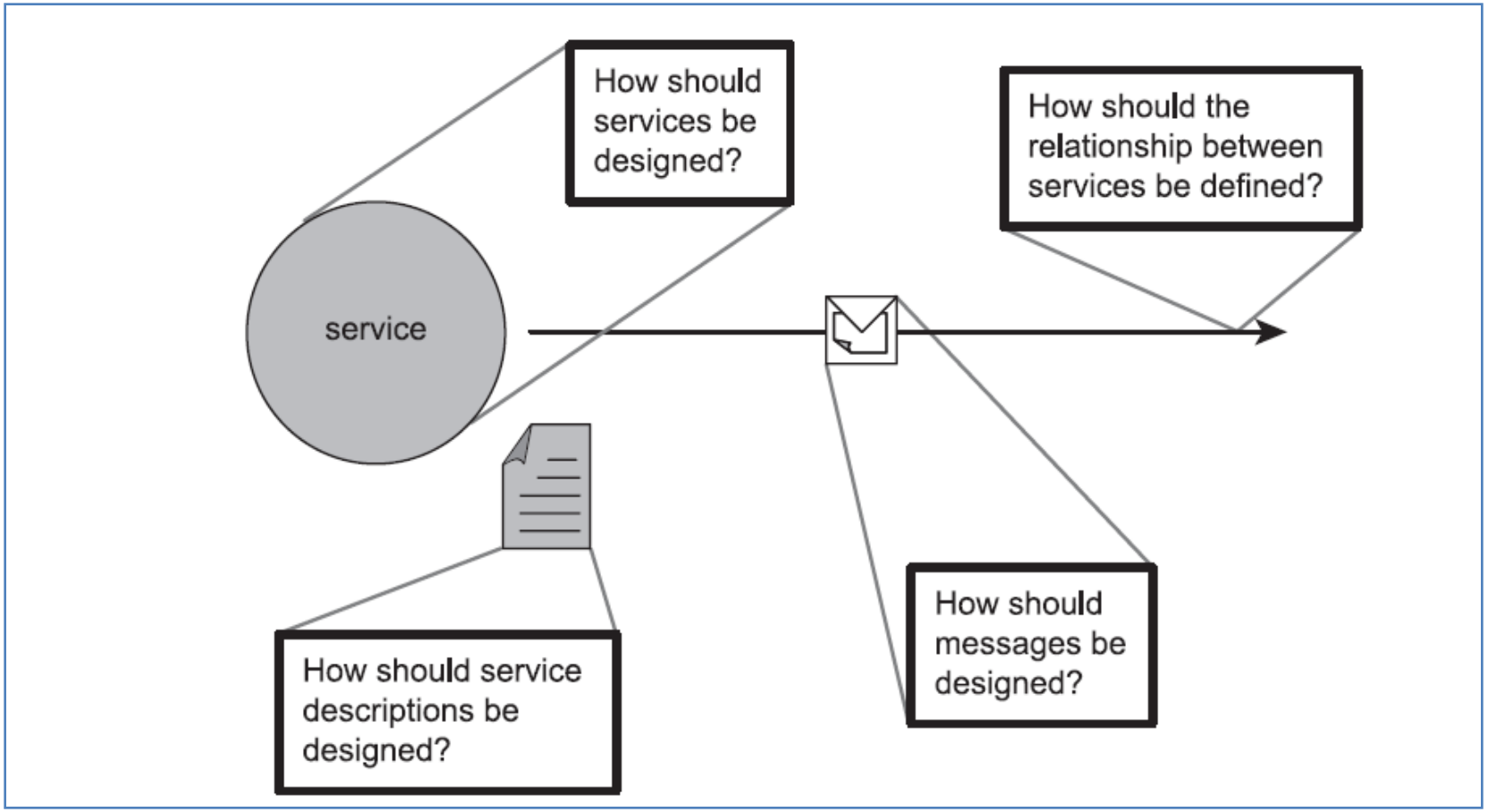
How services communicate



How services communicate

- Services are designed to be stateless
- Hence, once a service sends a message, it loses control of the message
- This implies that messages should also be autonomous, like services
- *"Messages should be fitted with self governing logic"*

Basic Components



How services are designed

- Services are designed by following service-orientation principles.
- *Loose coupling - Minimize dependency*
- *Service contract – Communication Agreement*
- *Autonomy - Control over logic*
- *Abstraction - Hide everything except contract*
- *Reusability – Decompose to Reuse*
- *Composability – Coordinate when needed*
- *Statelessness – Don't maintain activity specific info*
- *Discoverability – Easily accessible*

Primitive SOA

- **Primitive SOA** represents a baseline technology architecture that is supported by current major vendor platforms.
  - Includes following components:
    - Services
    - Descriptions
    - Messages
- Variations/Extensions of primitive SOA are known as contemporary SOA

Contemporary SOA characteristics

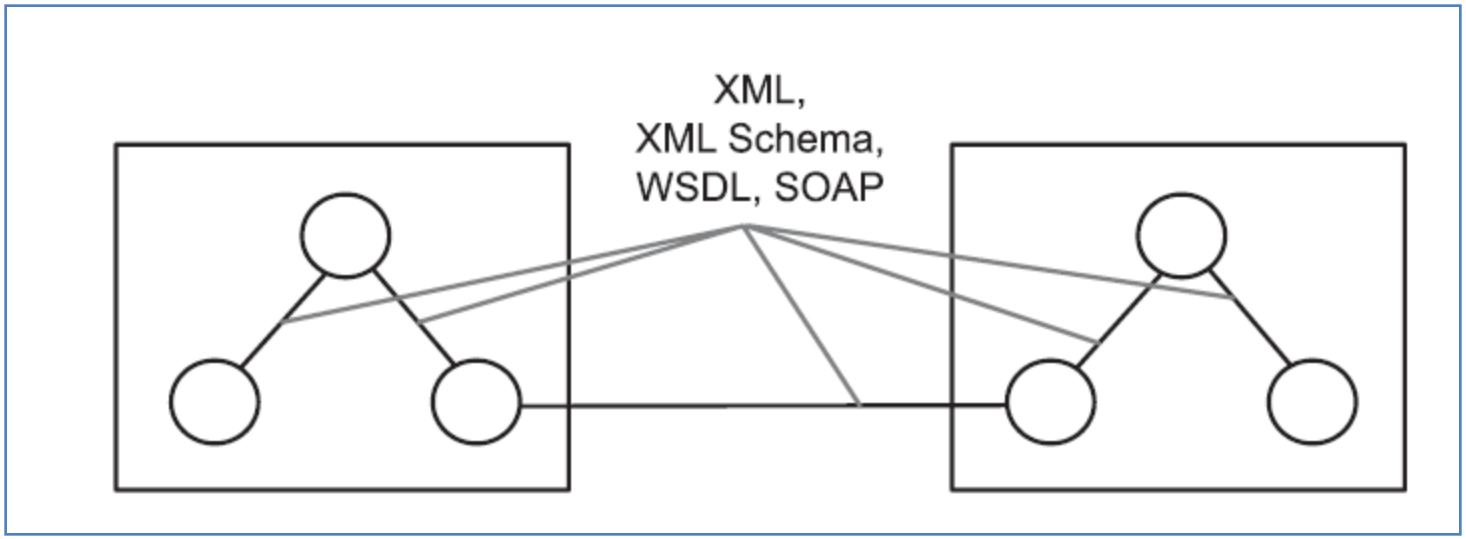
- Contemporary SOA increases quality of service
  - QoS
    - Protect content, Secure access to service
    - Reliability of message delivery
    - Integrity through transactions
    - Less overhead of processing
- ws-* extensions provide QoS

Contemporary SOA characteristics

- Contemporary SOA is fundamentally autonomous
  - Services should be as autonomous as possible
  - Service level autonomy is provided by Message level autonomy
    - Messages are self governing and intelligence heavy

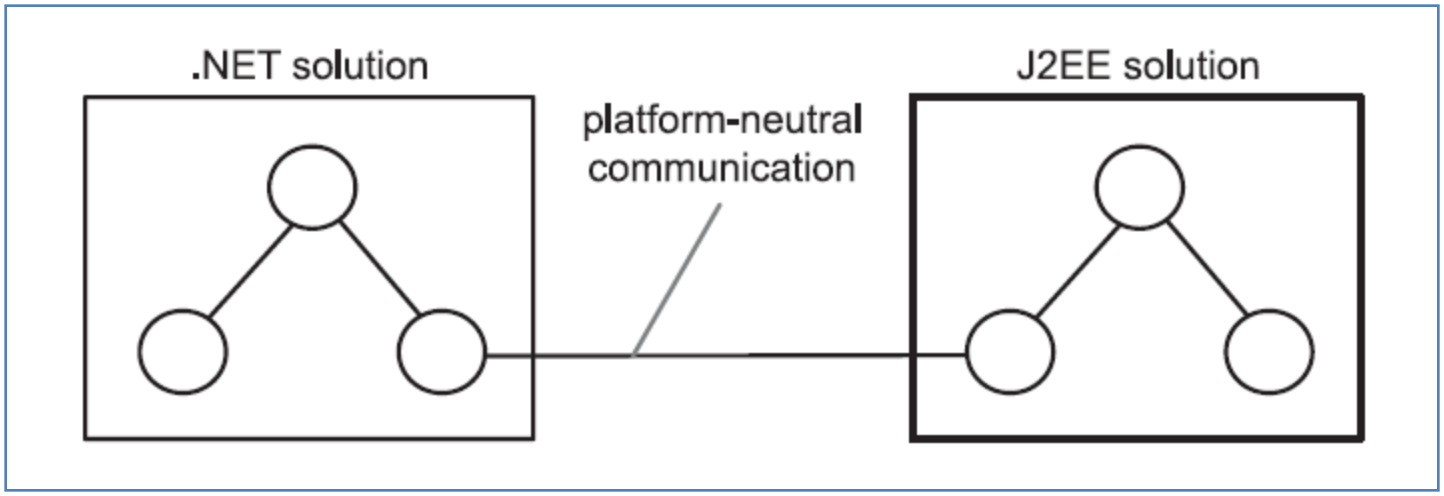
Contemporary SOA characteristics

- Contemporary SOA is based on open standards



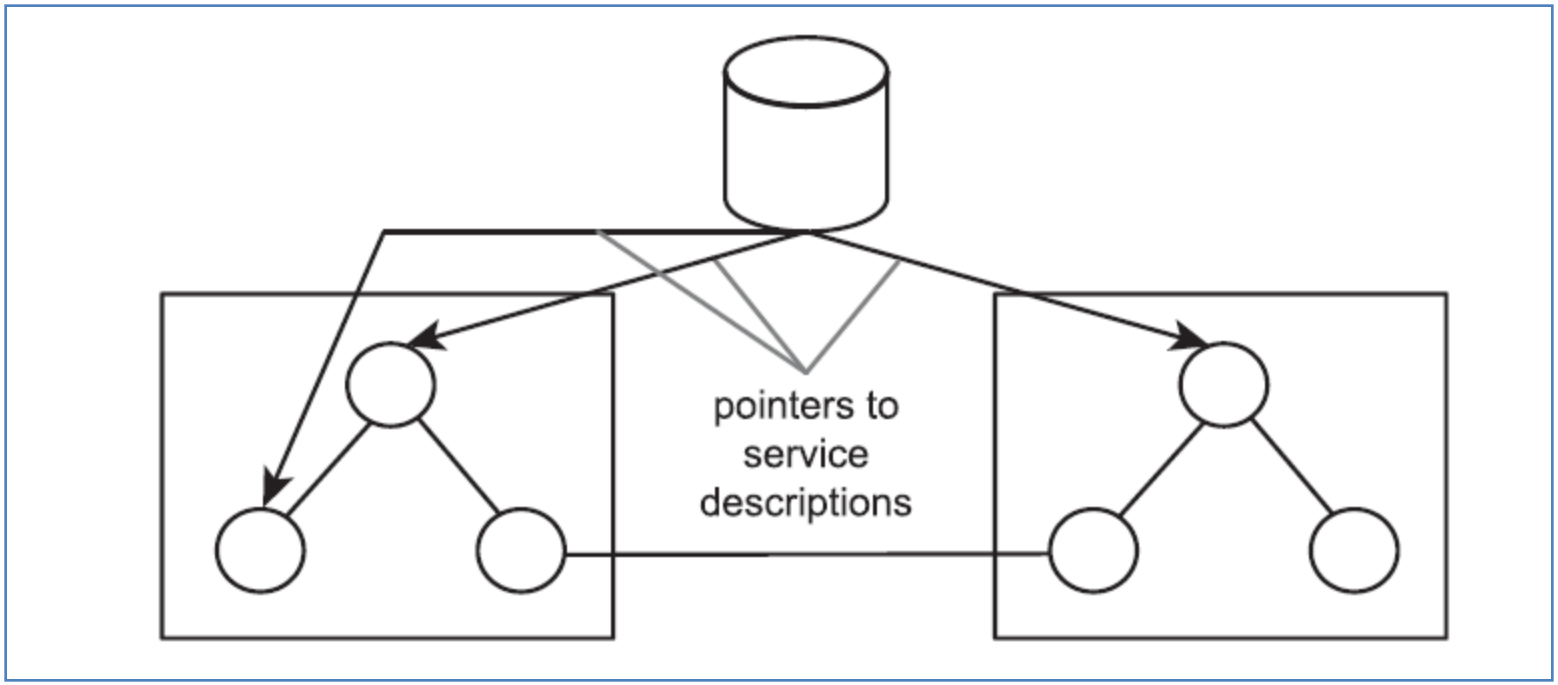
Contemporary SOA characteristics

- Contemporary SOA supports vendor diversity
  - If standard services are created, then it is possible to provide non proprietary service interface layer



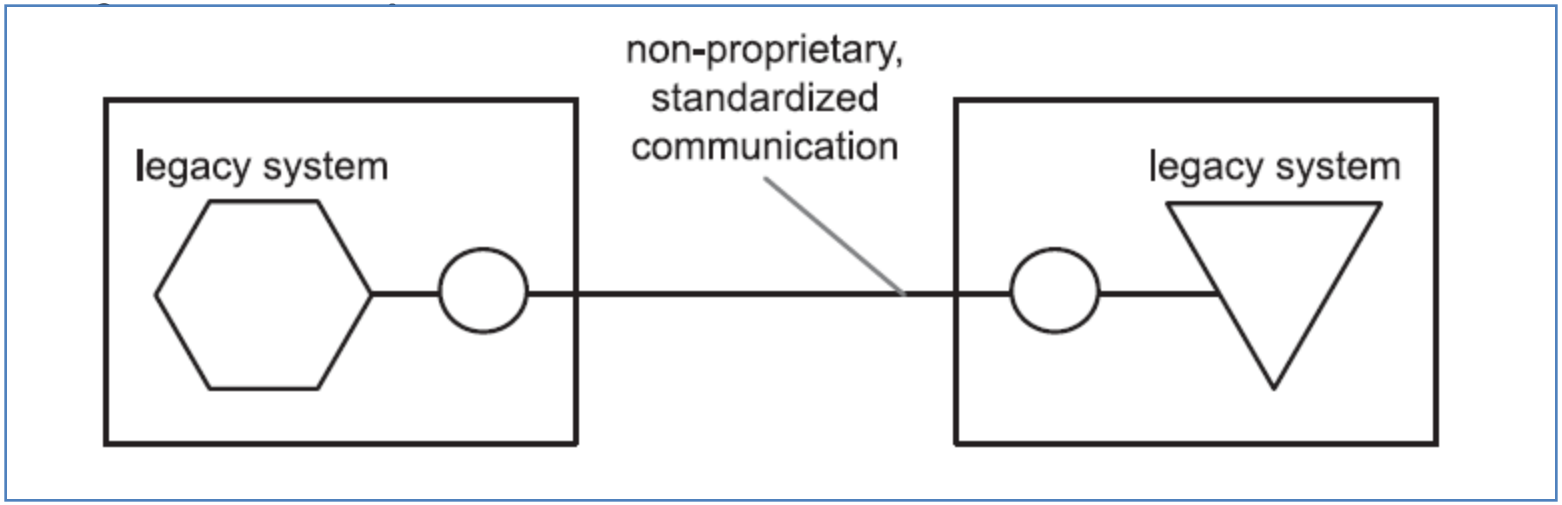
Contemporary SOA characteristics

- Contemporary SOA promotes discovery
  - SOA relies on some form of registry to manage service descriptions



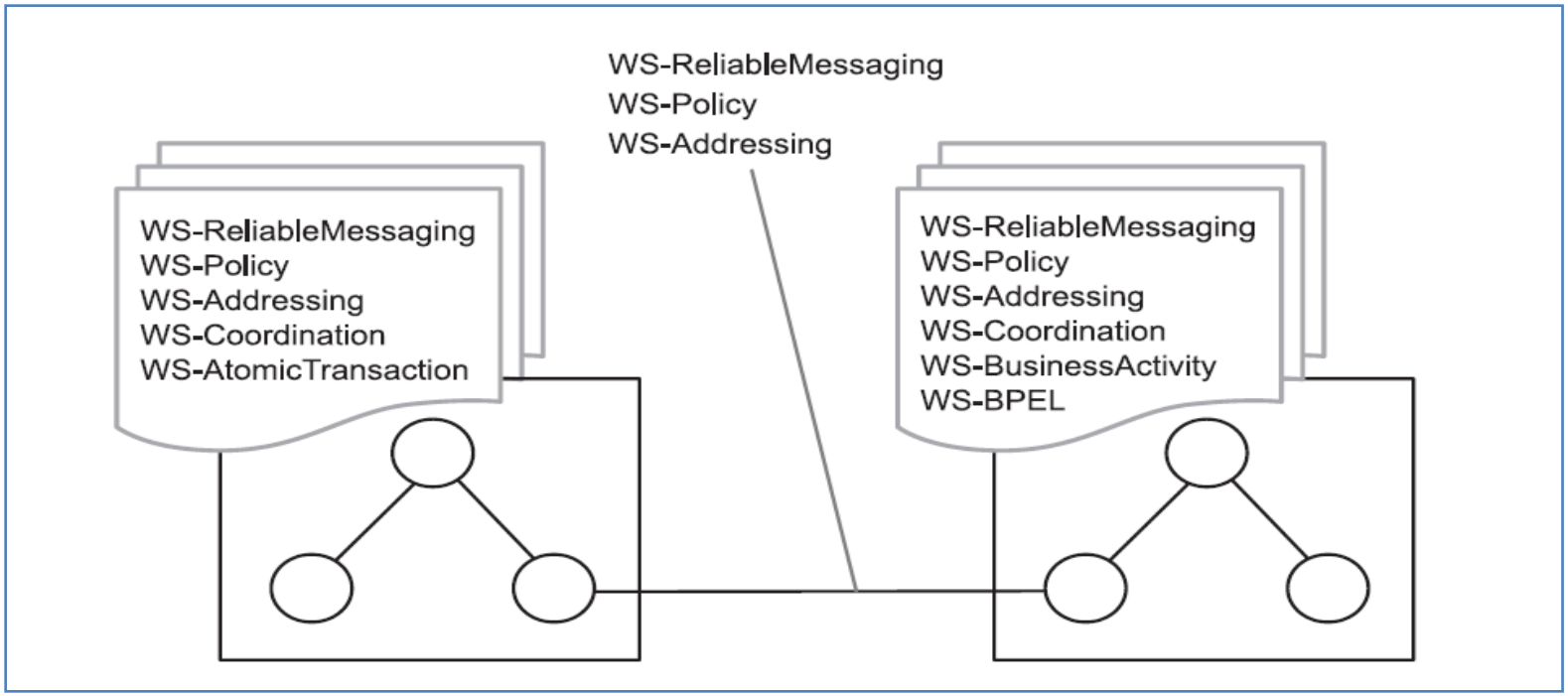
Contemporary SOA characteristics

- Contemporary SOA promotes federation
  - Establishing SOA within enterprise does not require replacement of existing applications
  - Legacy and non-legacy applications can be encapsulated via standard communication



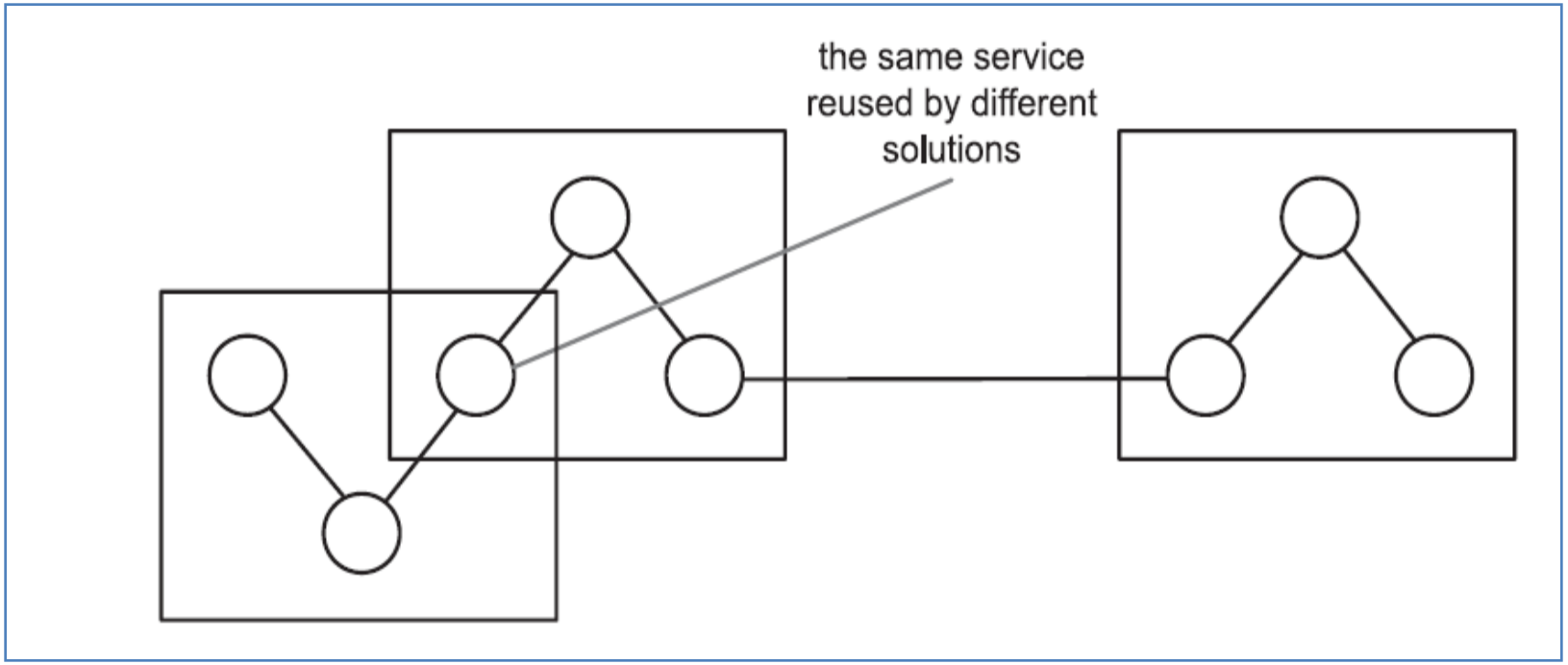
Contemporary SOA characteristics

- Contemporary SOA promotes architectural composability
  - Can be applied to service level or solution level



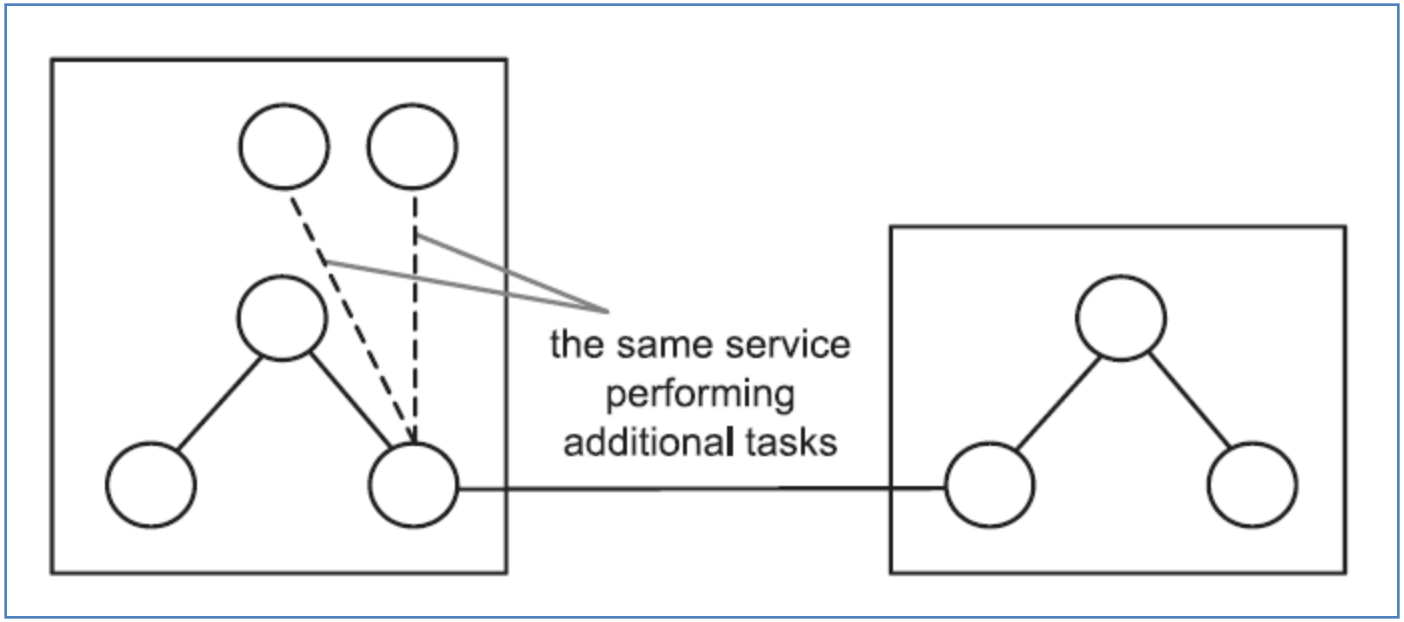
Contemporary SOA characteristics

- Contemporary SOA fosters inherent reusability



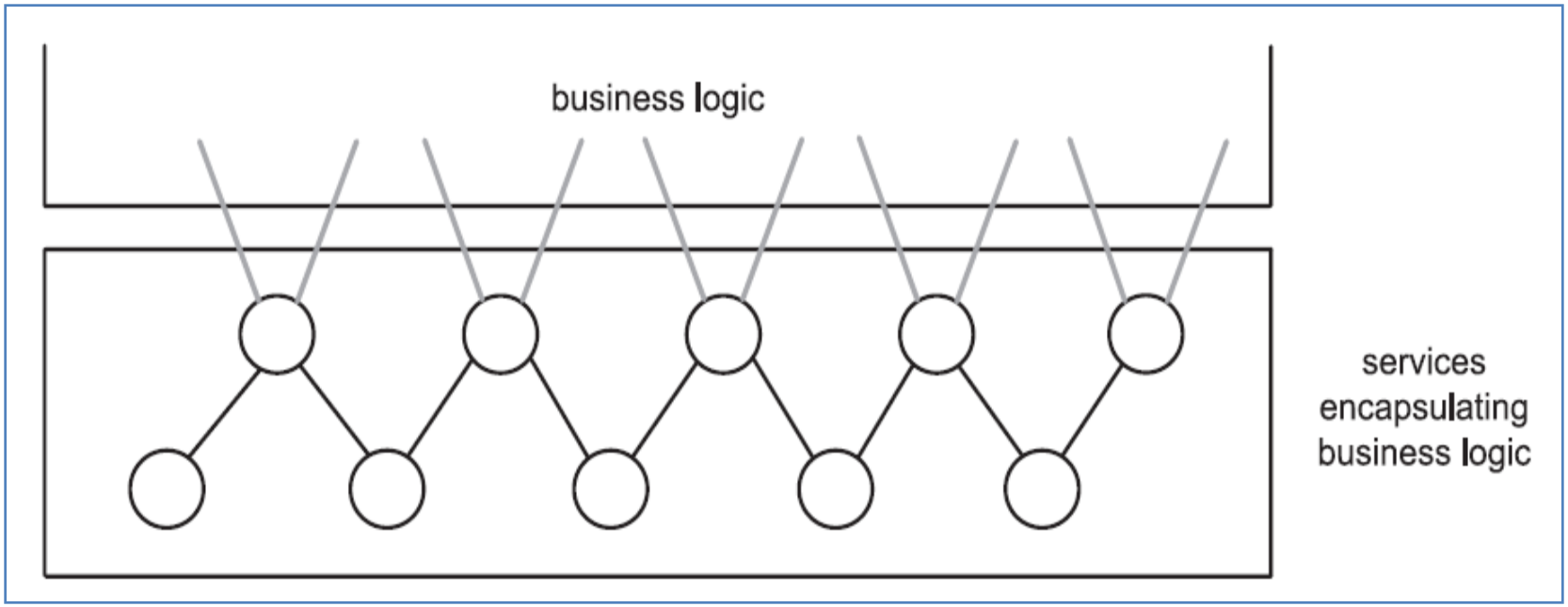
Contemporary SOA characteristics

- Contemporary SOA emphasizes extensibility
  - Services should be extended without breaking already established interface



Contemporary SOA characteristics

- Contemporary SOA supports a service-oriented business modeling paradigm



Contemporary SOA characteristics

- Contemporary SOA implements layers of abstraction
- Contemporary SOA promotes loose coupling throughout the enterprise
- Contemporary SOA promotes organizational agility

What is the Difference between Contemporary SOA and Primitive SOA?

Contemporary SOA is generally

- based on open standards
- architecturally composable
- capable of improving QoS

Contemporary SOA supports, fosters or promotes

- Vendor diversity
- Intrinsic interoperability
- Discoverability
- Federation
- Inherent reusability
- Extensibility
- Service-oriented business modeling
- Layers of abstraction
- Enterprise-wide loose coupling
- Organizational agility