中国科学技术大学
University of Science and Technology of China

Software Architecture

SSE USTC Qing Ding
dingqing@ustc.edu.cn
http://staff.ustc.edu.cn/~dingqing

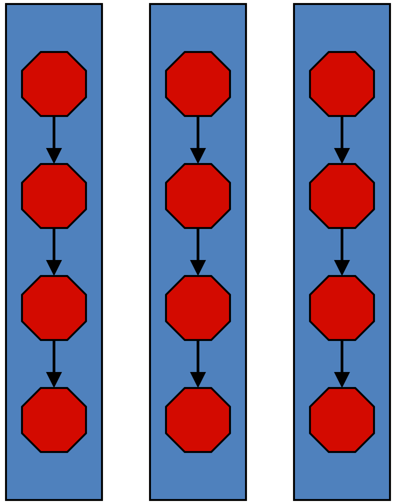
Service Oriented Architecture

outline

- Directions of System Architecture
- Overview of SOA
- History of SOA
- What is a Service Oriented Architecture?
- SOA Concepts
- Service-Oriented Architecture
- SOSE

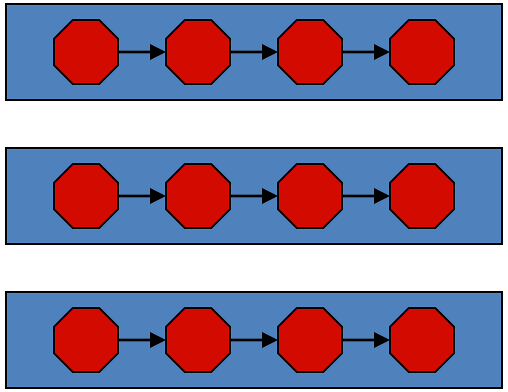
Directions of System Architecture

1960 - 1980



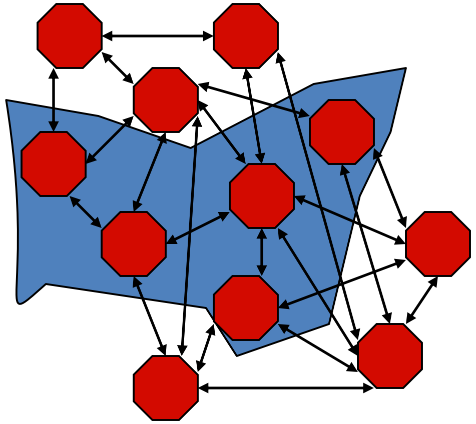
- Organization Focus
- Mainframe Centric
- Internal Use
- Unique Data

1990 - 2000



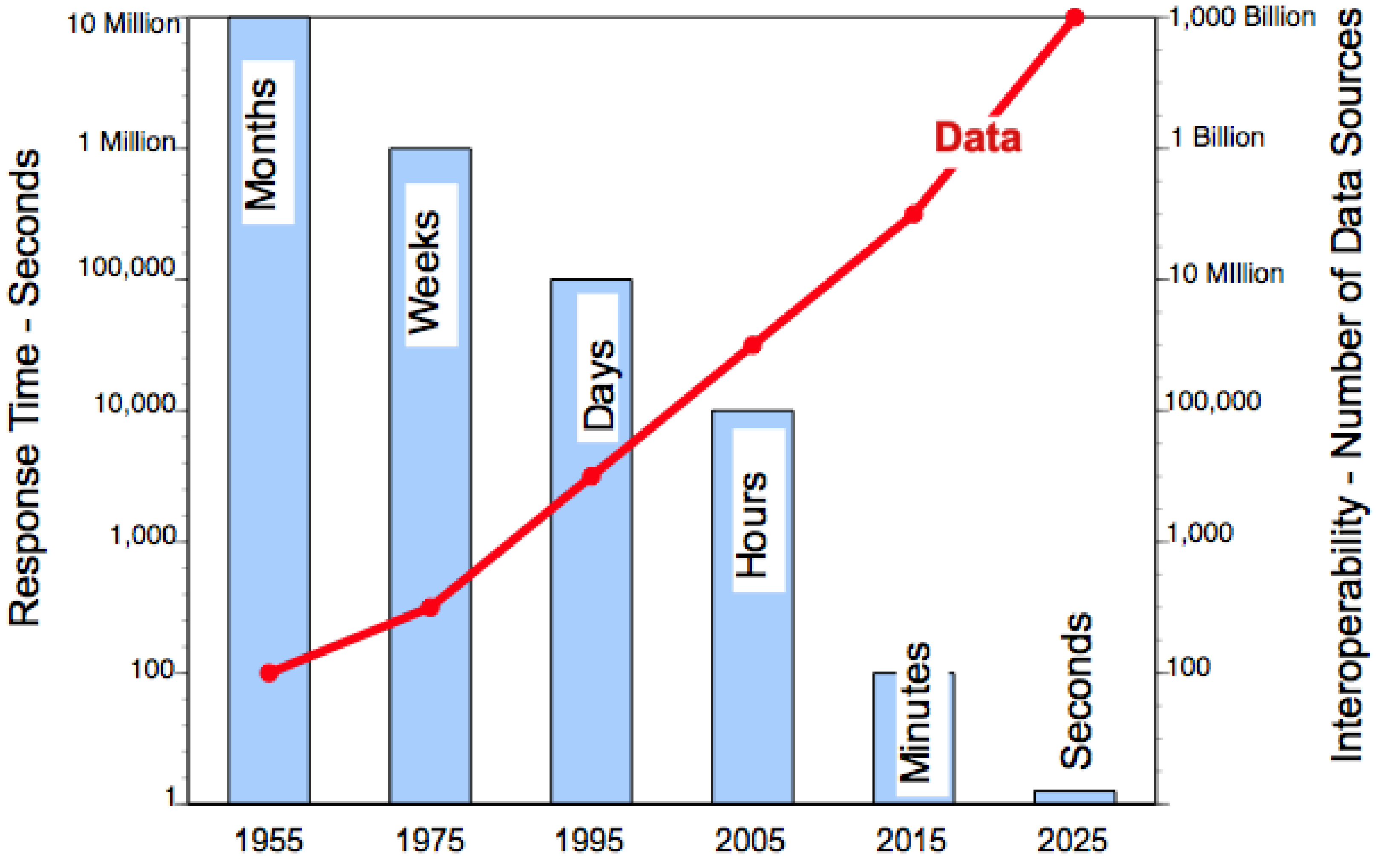
- Process Focus
- Client Server
- Partial Connectivity
- EDI File Transfer

2010 - 2050

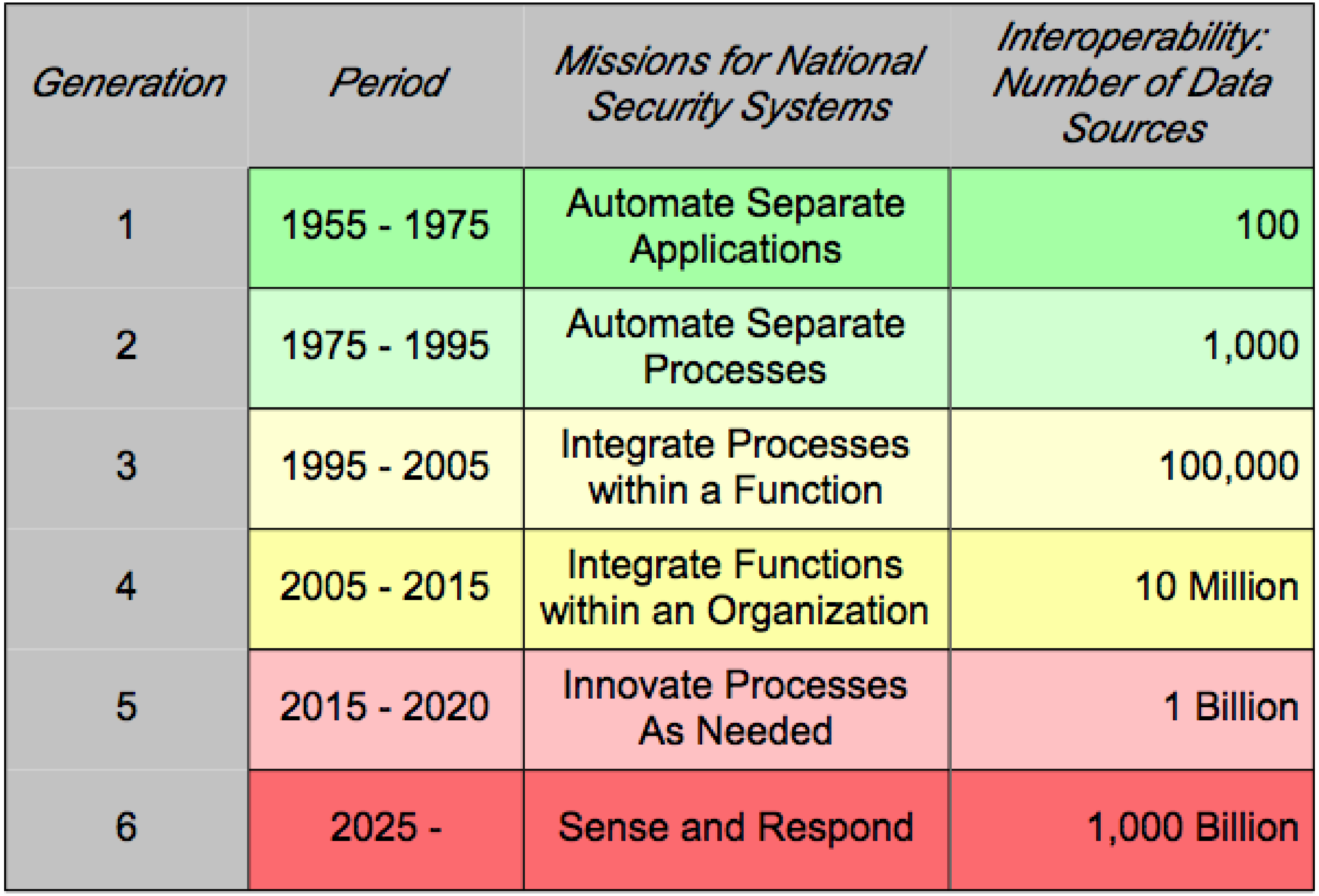


- Distributed Functions
- Data Centric
- Universal Interoperability
- Real-time Connectivity

Data Interoperability Expands as Response Time Shrink



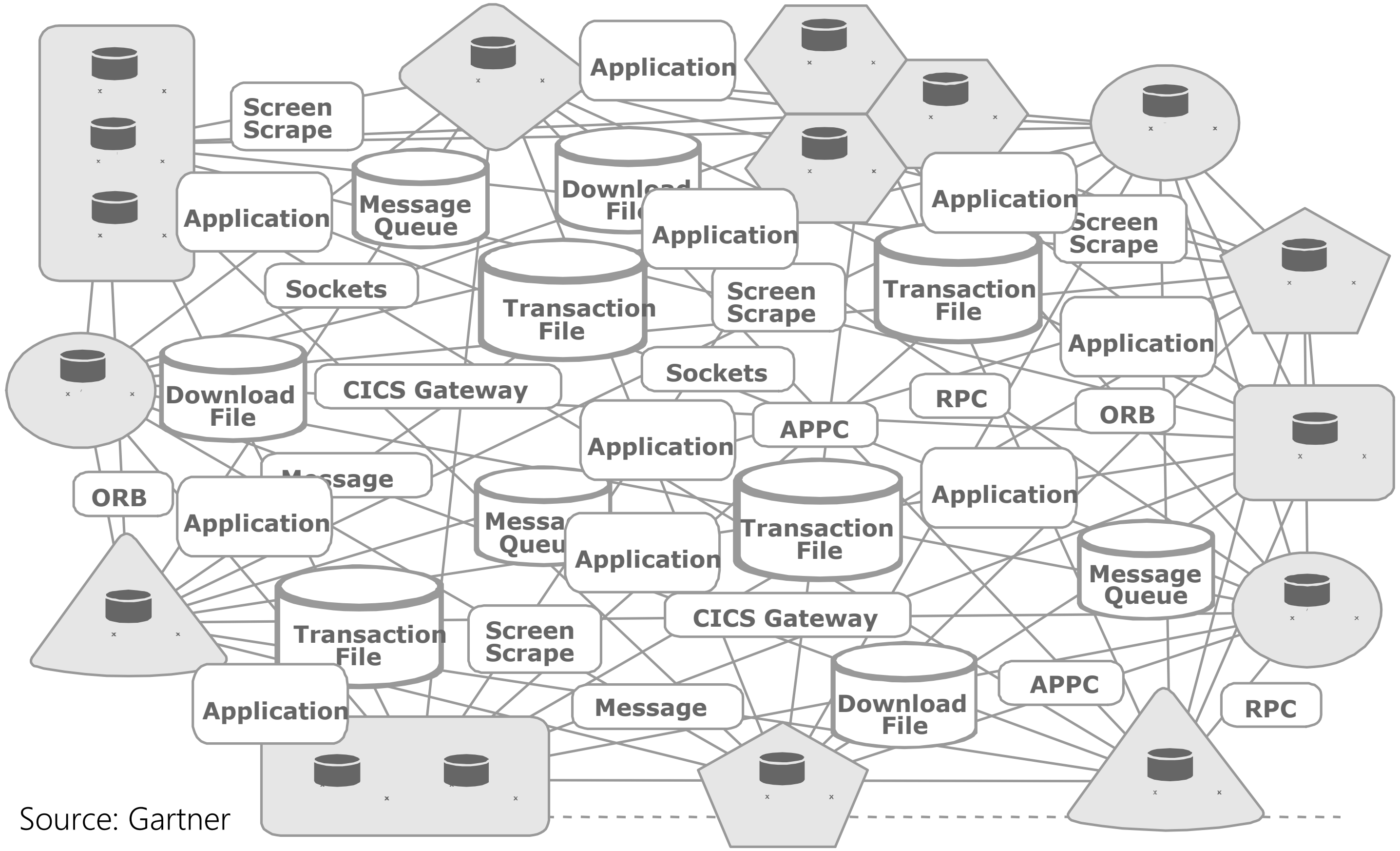
Interoperability Does not Scale



Generation	*Period*	*Missions for National Security Systems*	*Interoperability: Number of Data Sources*
1	1955 - 1975	Automate Separate Applications	100
2	1975 - 1995	Automate Separate Processes	1,000
3	1995 - 2005	Integrate Processes within a Function	100,000
4	2005 - 2015	Integrate Functions within an Organization	10 Million
5	2015 - 2020	Innovate Processes As Needed	1 Billion
6	2025 -	Sense and Respond	1,000 Billion

Why SOA? – The Cruel Reality

中国科学技术大学
University of Science and Technology of China



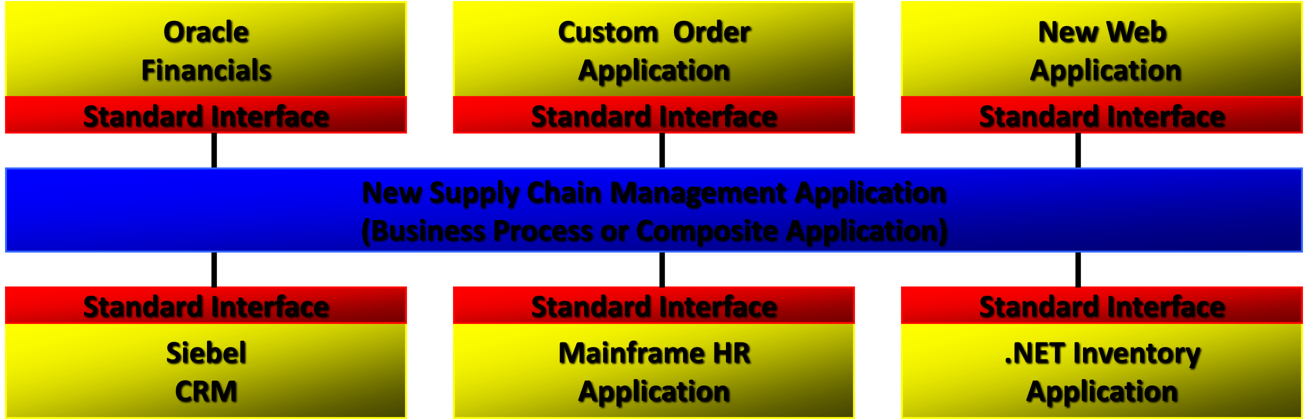
Source: Gartner

Why SOA?

相应新的需求

- Respond to business changes
- Address new needs with existing applications
- Unlock existing application investments
- Support new channels & complex interactions
- Support organic business

响应业务变化
•利用现有应用程序满足新的需求
•解锁现有的应用投资
•支持新的渠道和复杂的交互
支持有机业务



Overview of SOA

SOA主要实现灵活的组织，响应需求
ROA将数据当做资源，提供给其他程序远程调用

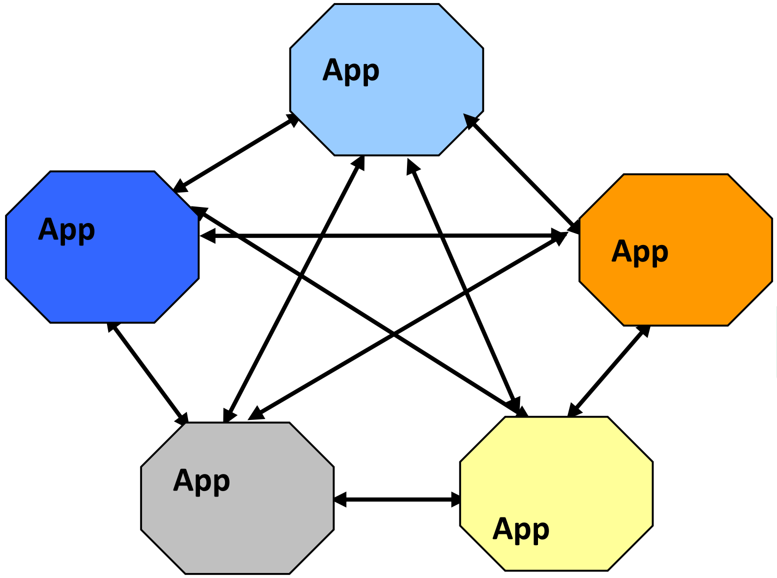
Service-Oriented Architecture (SOA) is an architectural style that represents business functionality as implementation-neutral, standards-based shared services

- SOA is a natural progression in the evolution that accelerated with the advent of XML and Web Services
- SOA enables enterprises to be more agile and to respond more quickly to changing business needs
- Some characteristics of SOA are:
  - Use of shared services — do not need to “reinvent the wheel”
  - Loose coupling — can update applications with minimal effect on services that invoke them
  - Location transparency — can re-host applications with minimal effect on services that invoke them
  - Based on open standards — decreased dependence on vendor-specific solutions

This means enhanced interoperability for government agencies

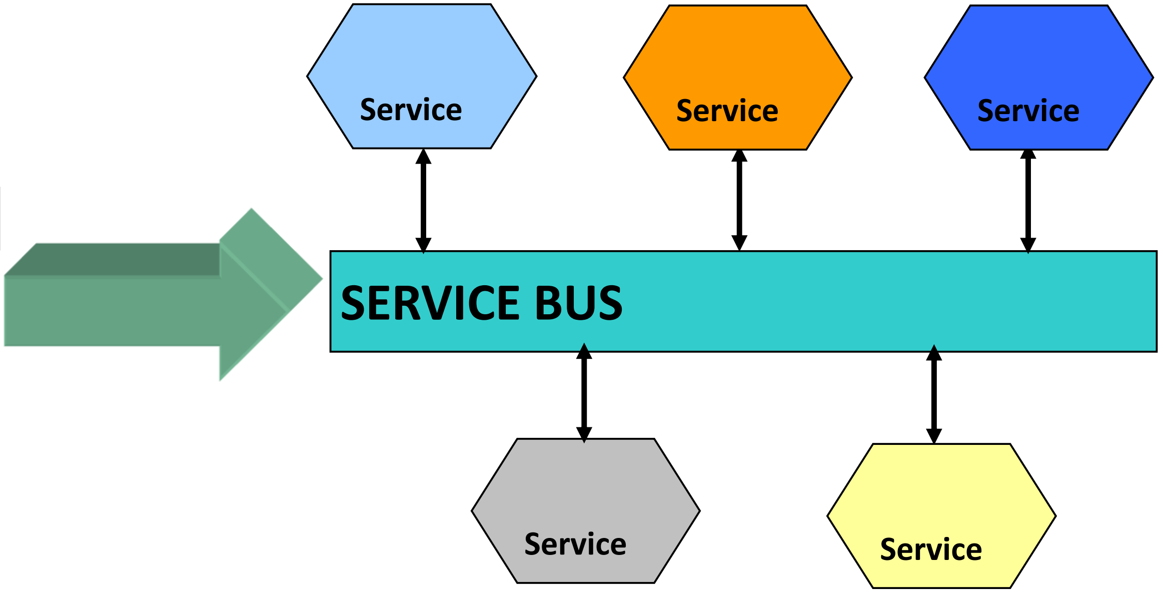
SOA enables an enterprise to avoid costly integration scenarios that utilize point-to-point connections between applications

Traditional “point-to-point” approaches to building IT environments have lead to a “spaghetti” approach to integration..



With this approach, when business processes or requirements change, agencies must undertake costly upgrade projects and introduce new connections

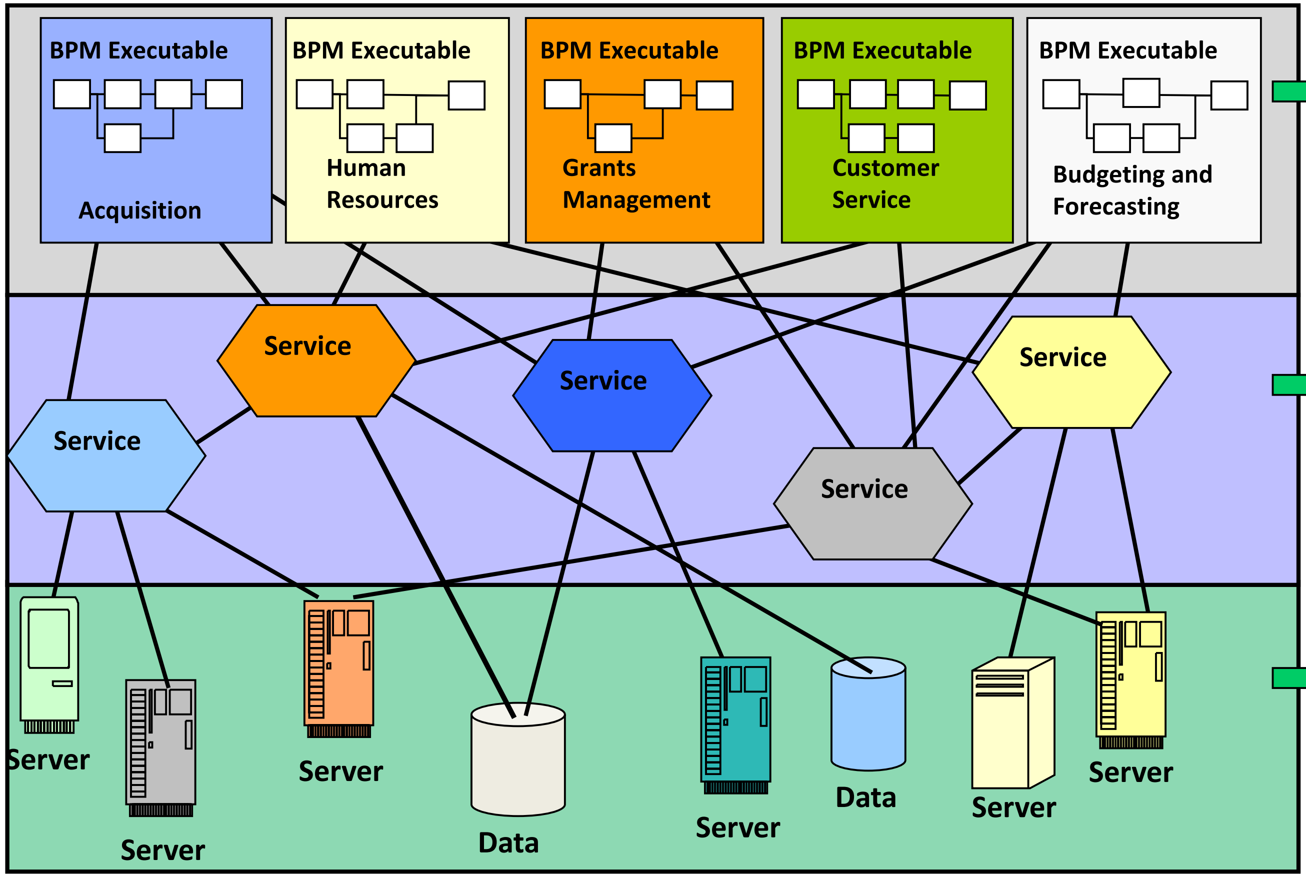
With SOA, applications are exposed as services that can be integrated through a unified service bus



This approach enables services to be “swapped in and out” or updated with minimal effect on existing services

SOA services are not necessarily Web Services, though in many cases they will be

SOA supports Business Process Management (BPM) by exposing shared services that can be shared across multiple processes



BPM executables implementing business processes access various information services to perform activities and manage workflow

实现业务流程的BPM可执行程序访问各种信息服务来执行活动和管理工作流

A robust SOA provides the framework to define reusable services to support a wide range of business processes

健壮的SOA提供了定义可重用服务的框架，以支持广泛的业务流程

Open standards enable leveraging of information assets from many agencies by “hiding” complexities of underlying agency infrastructures

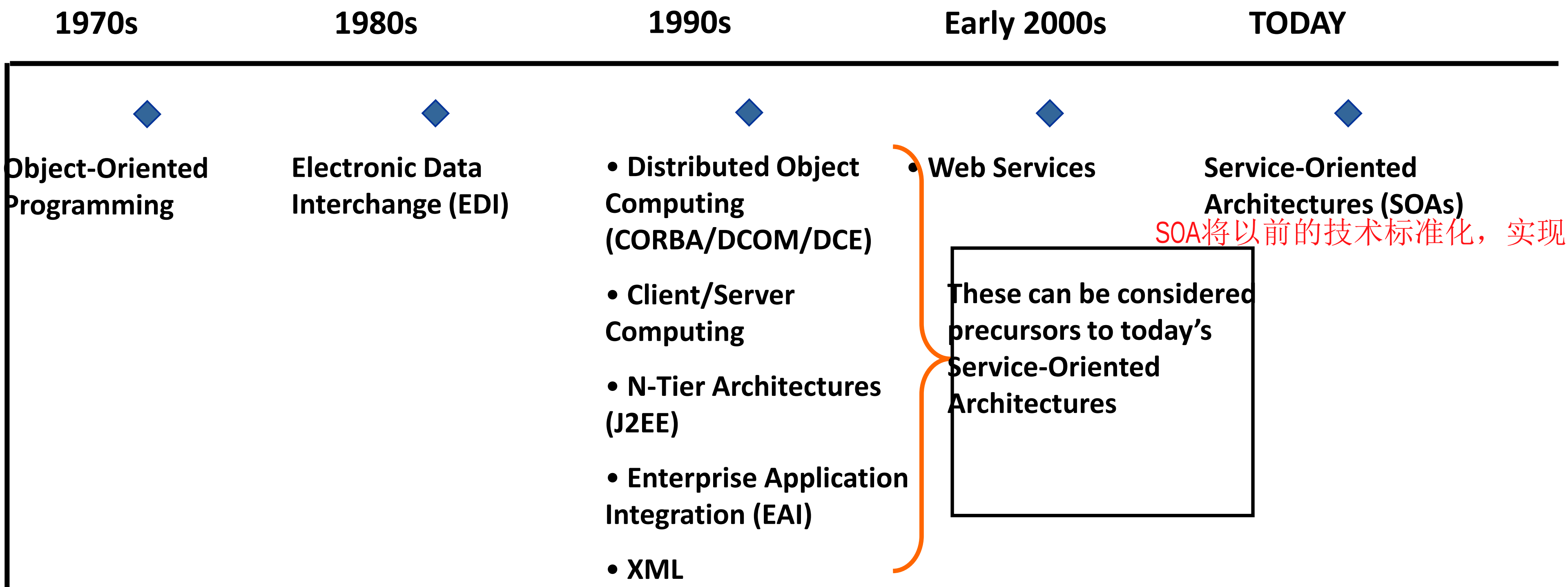
开放标准允许利用来自许多机构的信息资产通过“隐藏 ” 底层机构基础结构的复杂性

The result: A high degree of flexibility and agility for government operations 其结果是:管理运作具有高度的灵活性和敏捷性

第2，3层视为SOA的底层，已经实现好，SOA在第1层

History of SOA

SOA has a rich history



1970s

Object-Oriented Programming

1980s

Electronic Data Interchange (EDI)

1990s

- Distributed Object Computing (CORBA/DCOM/DCE)
- Client/Server Computing
- N-Tier Architectures (J2EE)
- Enterprise Application Integration (EAI)
- XML

Early 2000s

- Web Services

These can be considered precursors to today's Service-Oriented Architectures

TODAY

Service-Oriented Architectures (SOAs)

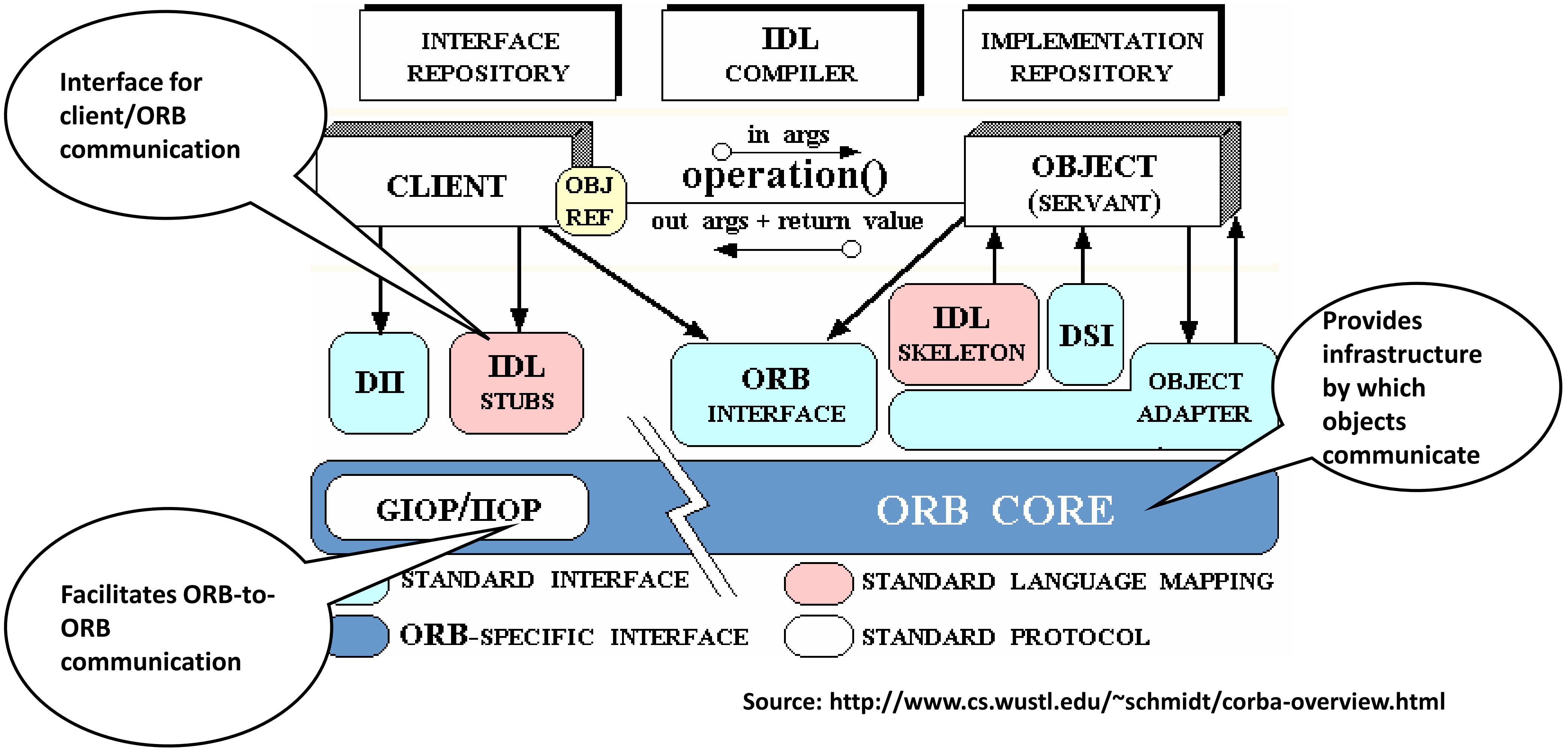
SOA将以前的技术标准化，实现各种接口，以解决新问题

SOA as a concept has been done for quite some time – but now, we are leveraging the power of the World Wide Web

Then and Now: A Brief Comparison Between CORBA and SOA

Common Object Request Broker Architecture (CORBA) is an open, vendor-independent architecture and infrastructure that

- It was created by the Object Management Group (OMG)



Interface for client/ORB communication

Provides infrastructure by which objects communicate

Facilitates ORB-to-ORB communication

Source: http://www.cs.wustl.edu/~schmidt/corba-overview.html

CORBA has many complexities that are mitigated by SOA

追求效率

追求开放、可读性

Factor	*CORBA*	*SOA*
"Weight" of implementation	Heavyweight	More lightweight
Degree of coupling	Tight coupling (to components)	Loose coupling (between services and their underlying applications)
Communication Mode	Synchronous only	Synchronous or asynchronous
Initial investment	Large	Small-medium (depending on requirements)
Protocol type	Binary	Text
Processing "grain"	Fine-grained processing	Coarse-grained or fine-grained processing (depending on requirements)
Proprietary Level	Proprietary implementations	Non-proprietary implementations

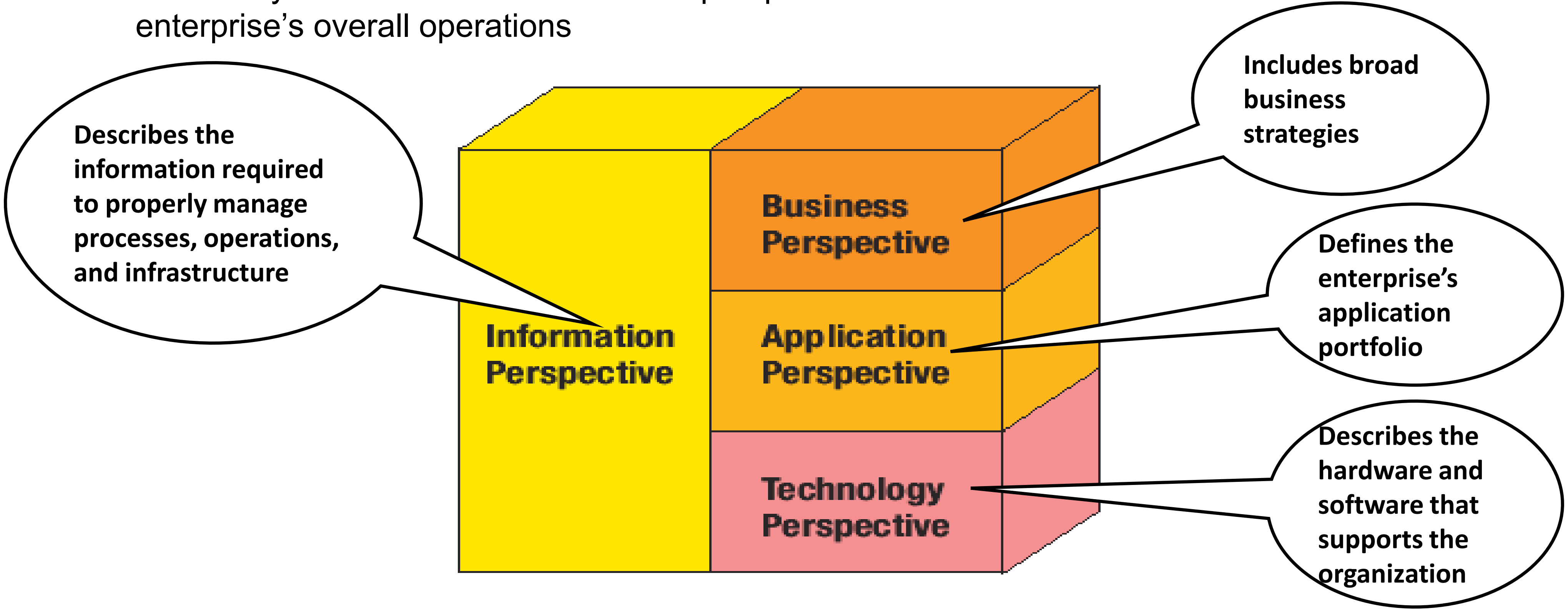
It is important to note that services within an SOA can have CORBA components "behind" them

Relation between Enterprise Architecture (EA) and SOA

An Enterprise Architecture is a critical ingredient in an organization’s technology planning and overall operation

- An enterprise architecture normally takes the form of a comprehensive set of cohesive models that describe an enterprise’s structure and functions

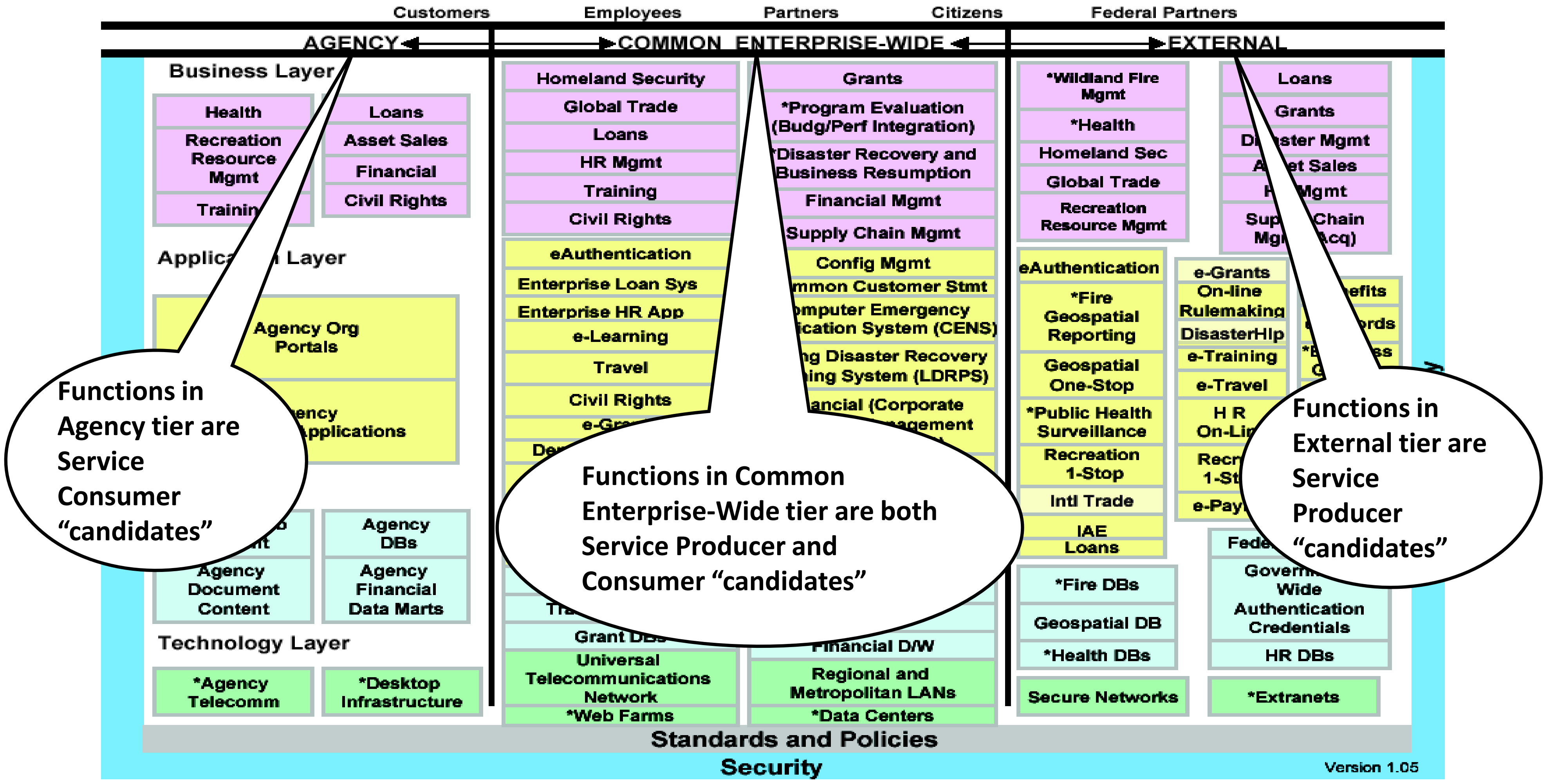
- It normally reflects different “views” or “perspectives” that relate to various areas of an enterprise’s overall operations


Describes the information required to properly manage processes, operations, and infrastructure
Information Perspective
Business Perspective
Application Perspective
Technology Perspective
Includes broad business strategies
Defines the enterprise’s application portfolio
Describes the hardware and software that supports the organization

Source: “Dissecting Service-Oriented Architectures”, Lublinksy and Tyomkin, Business Integration Journal, October 2003

An Enterprise Architecture provides a foundation by which a Service-Oriented Architecture can clearly emerge

- The following is a real-world example from a US federal agency's Enterprise Architecture



Functions in Agency tier are Service Consumer "candidates"

Functions in Common Enterprise-Wide tier are both Service Producer and Consumer "candidates"

Functions in External tier are Service Producer "candidates"

What is a Service Oriented Architecture (SOA)?

- A method of design, deployment, and management of both applications and the software infrastructure where:
  - All software is organized into business services that are network accessible and executable.
  - Service interfaces are based on public standards for interoperability.

一种设计、部署和管理应用程序和软件基础设施的方法，其中:
-所有软件被组织成网络可访问和可执行的业务服务。
–服务接口采用公共标准，实现互操作性。

Key Characteristics of SOA

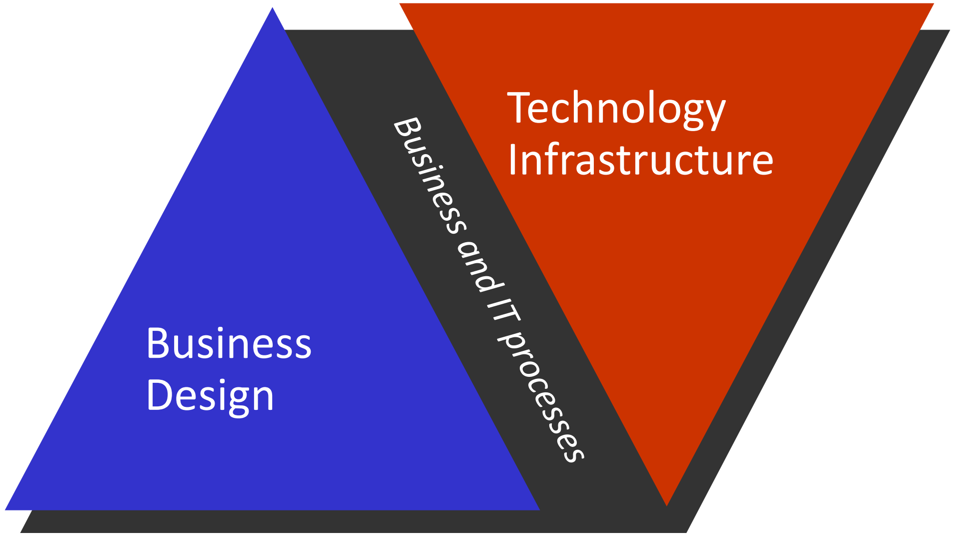
- Quality of service, security and performance are specified.
- Software infrastructure is responsible for managing.
- Services are cataloged and discoverable.
- Data are cataloged and discoverable.
- Protocols use only industry standards.

SOA Concepts

Becoming an On Demand Business

An On Demand Business is an enterprise whose business processes — integrated end-to-end across the company and with key partners, suppliers and customers — can respond with speed to any customer demand, market opportunity or external threat.

Align
business models and strategic objectives



Integrate
people, processes, and information

Optimize
application infrastructure

Extend
your reach

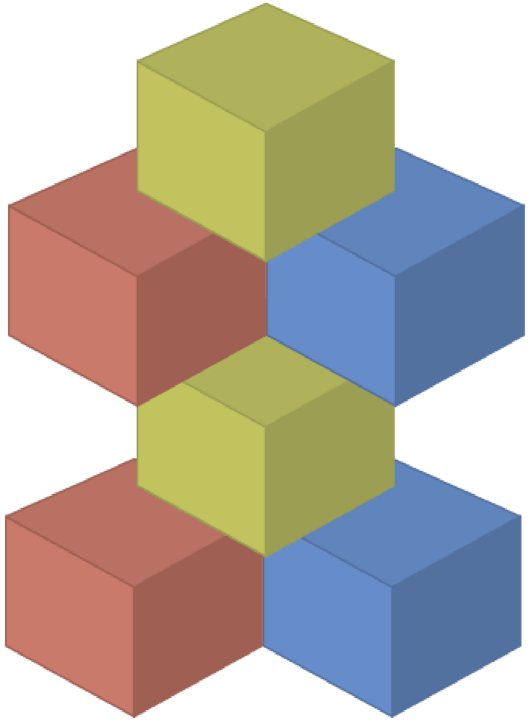
Four Characteristics of On Demand

- Integration
  - Providing the linkage between people, processes, and data
- Open
  - Supporting a strong commitment to standards for OS, Language and Web Services/SOA
- Virtualized
  - Providing a flexible Build-time and Runtime environment for developing and running applications across a highly distributed IT architecture
- Autonomic
  - Self regulating … self healing … self maintaining

SOA: Service Oriented Architecture

- An approach for building distributed systems that allows tight correlation between the business model and the IT implementation.

- Characteristics:
  - Represents business function as a service
  - Shifts focus to application assembly rather than implementation details
  - Allows individual software assets to become building blocks that can be reused in developing composite applications representing business processes
  - Leverages open standards to represent software assets



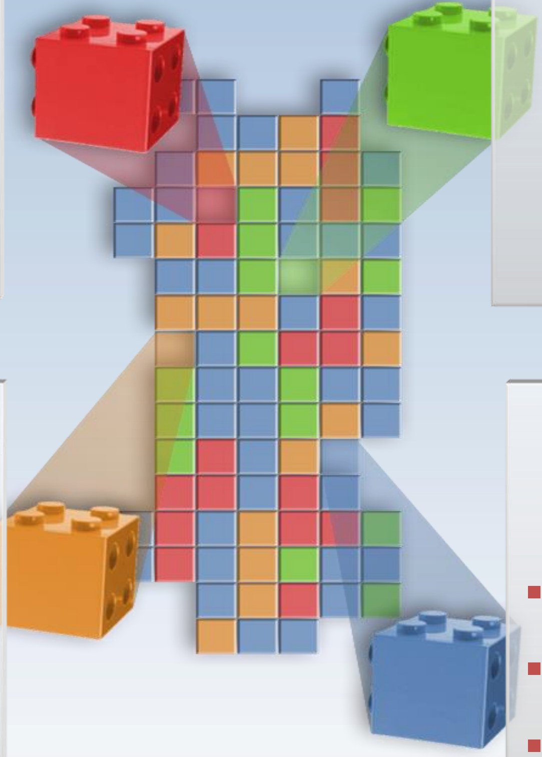
SOA Definitions

What is a service?

A repeatable business task – e.g., check customer credit; open new account

What is service orientation?

A way of integrating your business as linked services and the outcomes that they bring



What is service oriented architecture (SOA)?

The IT architectural style that supports service orientation

What does SOA mean to business?

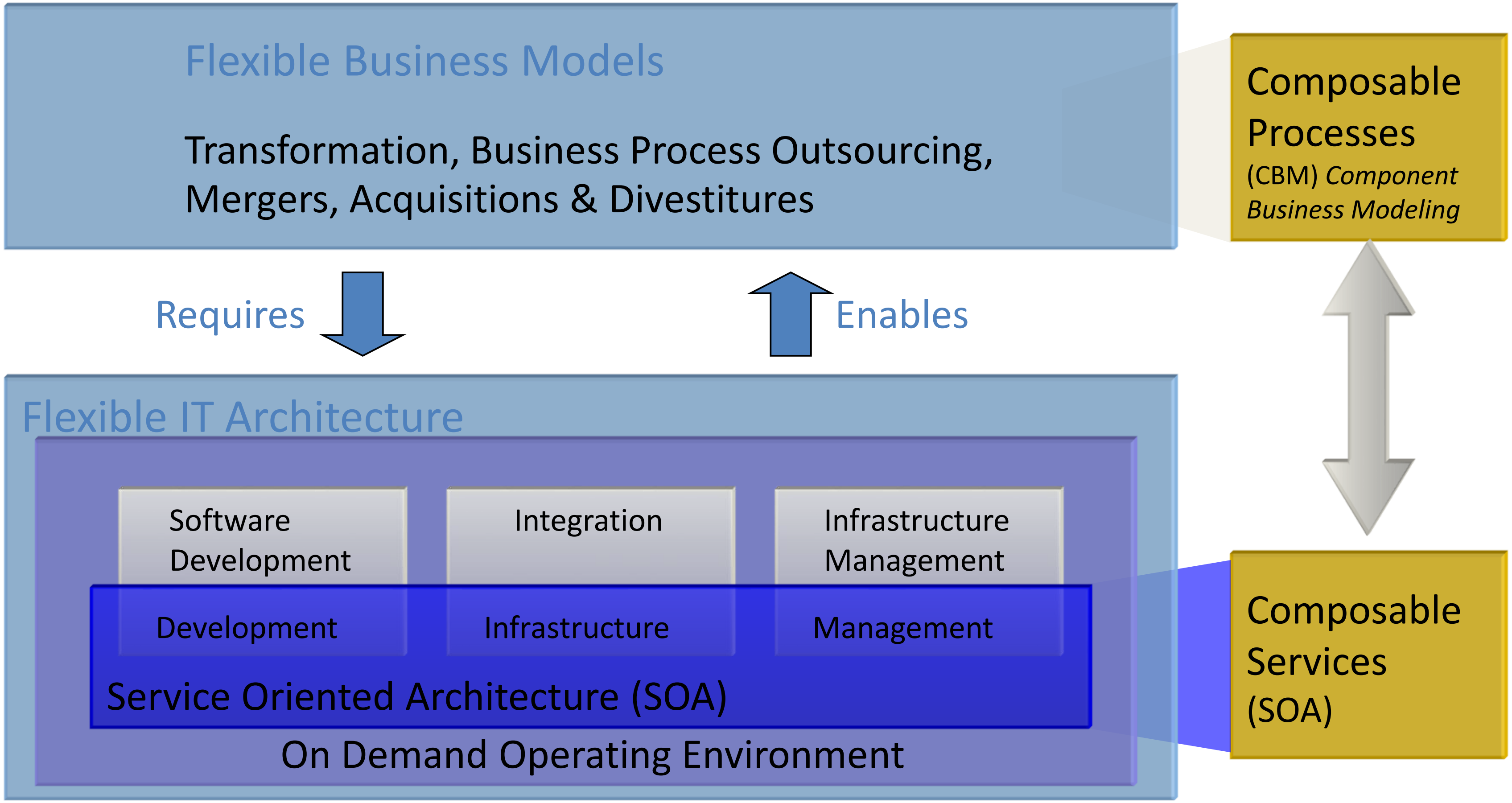
- Business flexibility
- Improved customer service
- Lower costs and greater revenue

SOA Concepts

- What is a service?
  - A coarse grained, self-contained entity that performs a distinct business function
- What is a service description?
  - A standards based interface definition that is independent of the underlying implementation
- What is service discovery?
  - Use of a service registry to access service interface descriptions at buildtime or runtime
- How do services interact?
  - Through loosely-coupled, intermediated connections
- What is service choreography?
  - Control of the execution sequence of services in ways that implement business processes
- How are SOA solutions created and enhanced?
  - Using tools and middleware according to SOA principles

Flexible & Adaptable business models & supporting IT architectures

…are required today for business survival



Flexible Business Models

Transformation, Business Process Outsourcing, Mergers, Acquisitions & Divestitures

Requires

Enables

Flexible IT Architecture

Software Development

Integration

Infrastructure Management

Development

Infrastructure

Management

Service Oriented Architecture (SOA)

On Demand Operating Environment

Composable Processes
(CBM) *Component Business Modeling*

Composable Services (SOA)

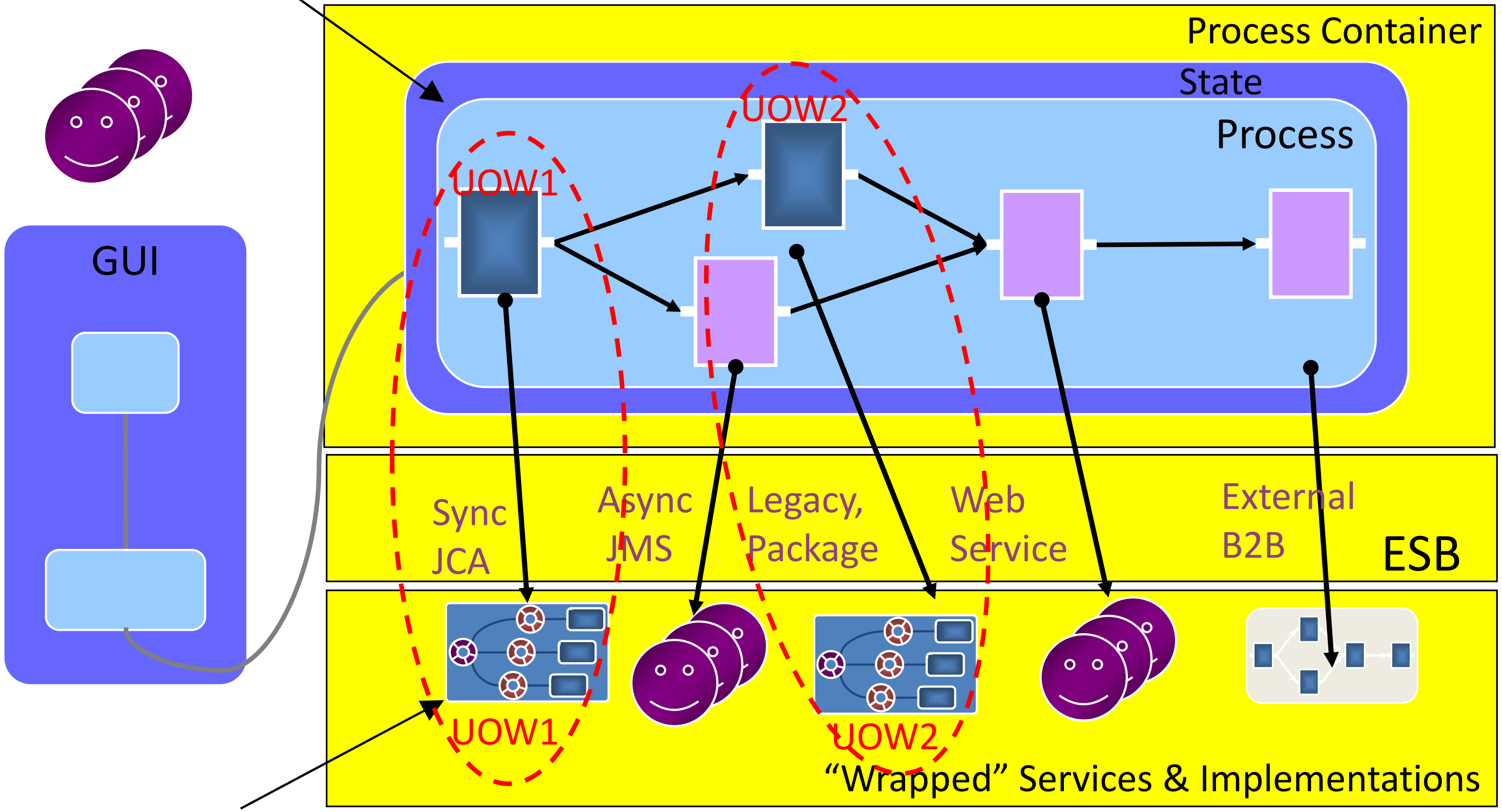
Three Key Concepts for the Foundation for On Demand

- Build –Model Driven Architecture
  - A style of enterprise application development and integration based on using automated tools to build system independent models and transform them into efficient implementations1
- Run –Service Oriented Architecture
  - An approach for designing and implementing distributed systems that allows a tight correlation between the business model and the IT implementation
- Manage –Business Performance Management
  - An approach to systems management that tightly links IT concerns with business process concerns

[1] Source: Booch, et al, “An MDA Manifesto”, published in the MDA Journal, May 2004

SOA & Business Process Choreography Services Animation

‘Coarse-Grained’ – Long Running, Interruptible, Compensation Transaction network



‘Fine-Grained’ – Short-Running, non-Interruptible, ‘ACID’ XA Transaction

Messaging Fundamentals

A single solution, with multi-platform APIs (JMS and MQI)

- Easy to use message centric interface
- Network independent
- Faster application development

Assured message delivery

- Exactly Once, Transactional

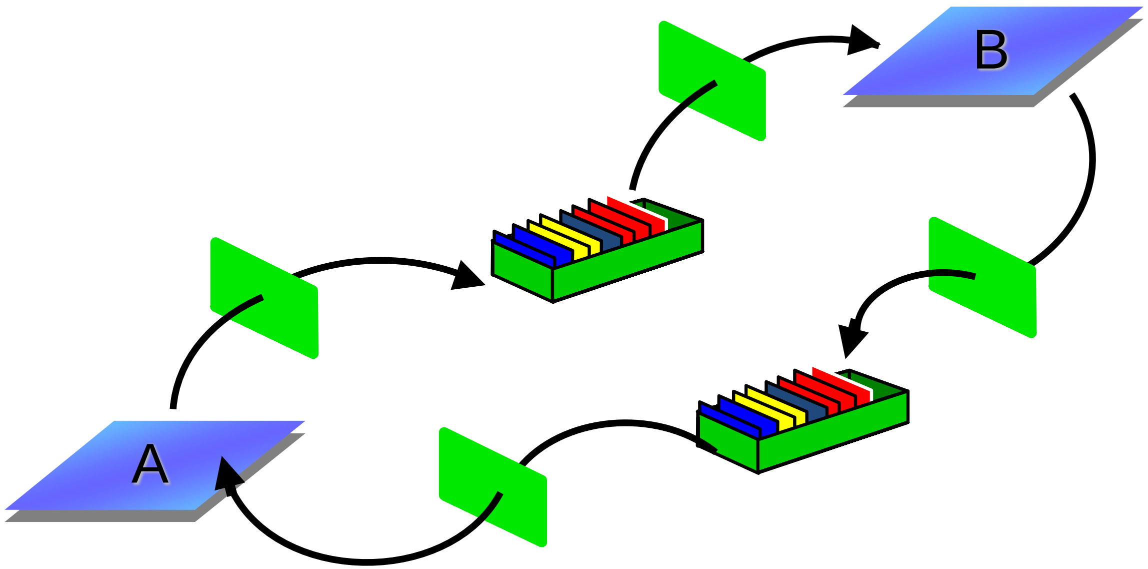
Loosely-coupled applications

- Asynchronous messaging
- Parallelism, Triggering

Scalable & Robust

- Publish\Subscribe or Point to Point
- Clustering, Large Messages

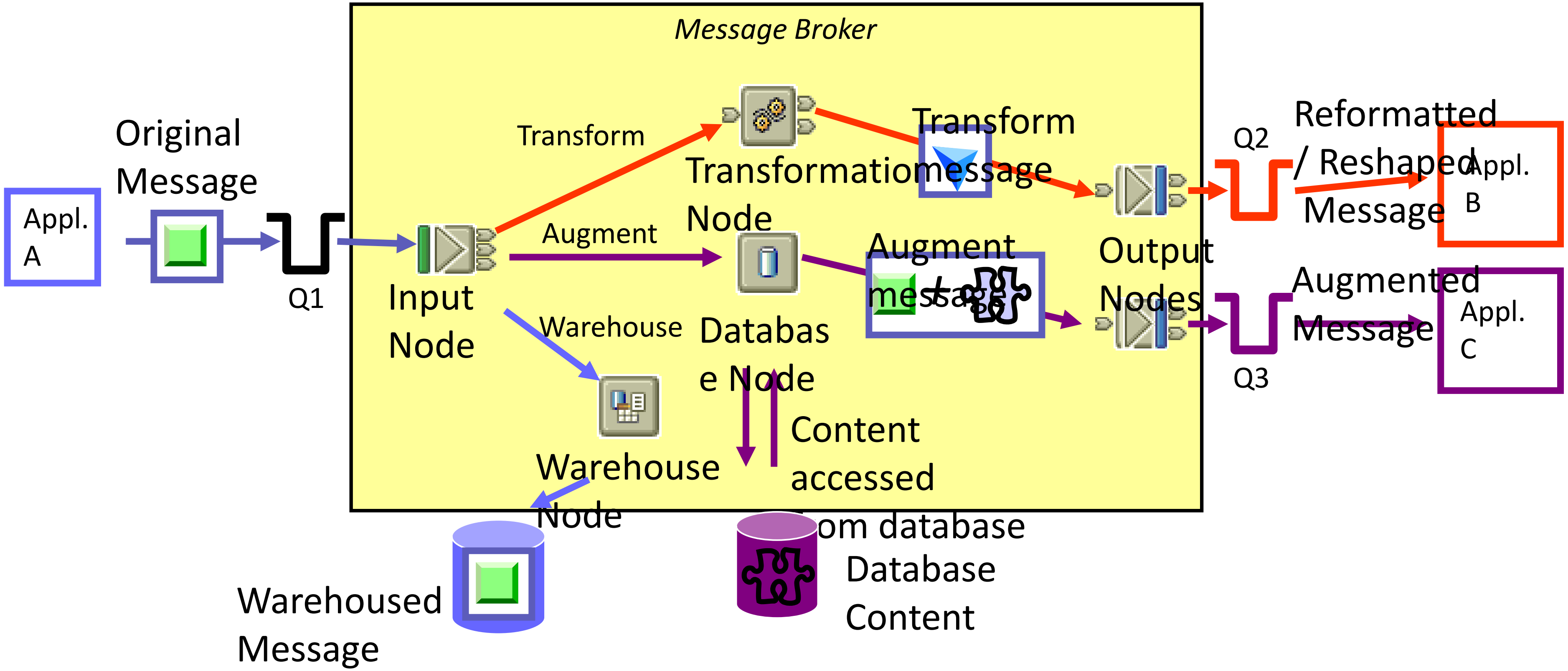
Pervasive



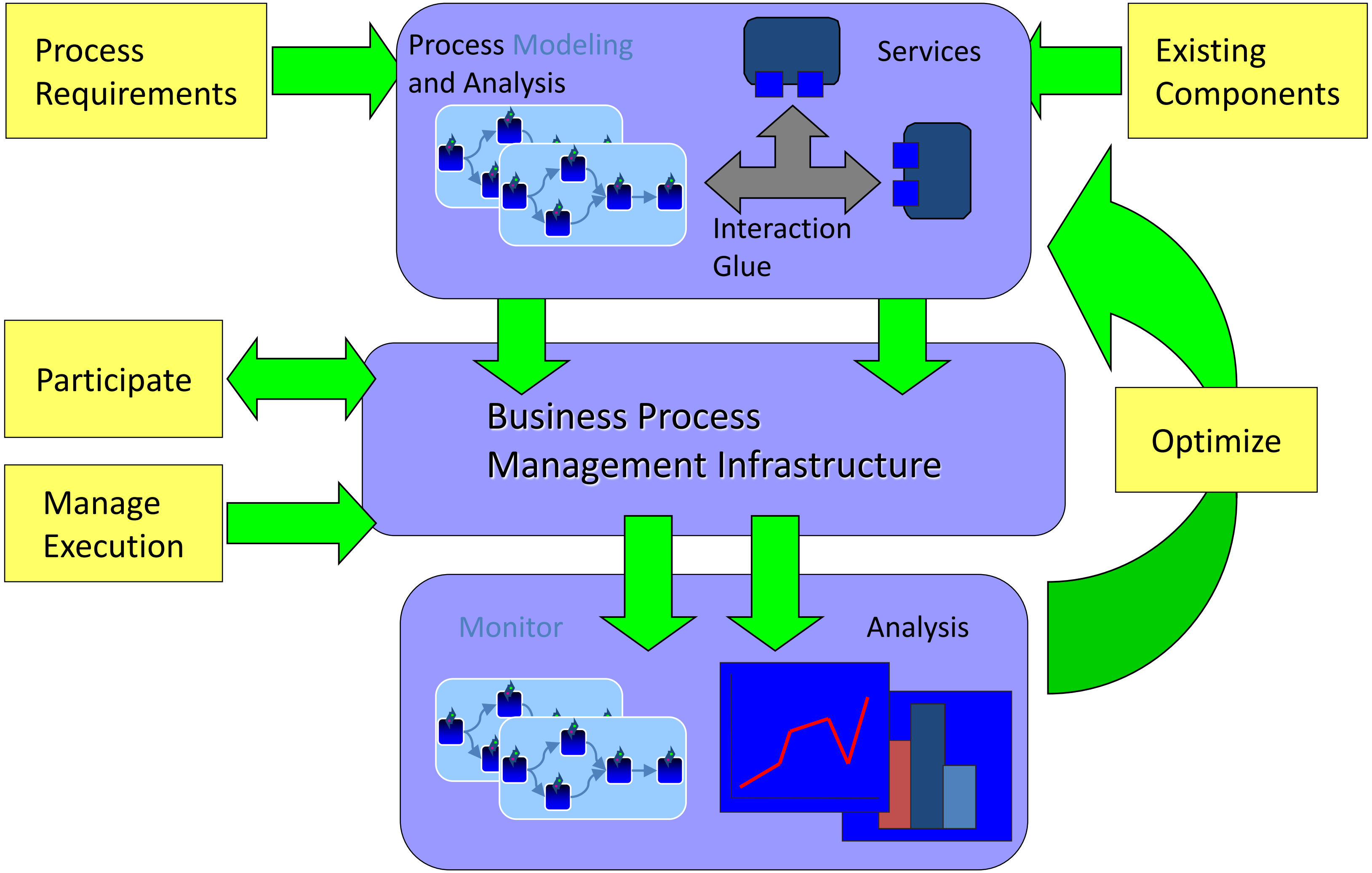
Message Broker -Transforms messages 'in flight'

Delivers messages to the right place and in the right format.

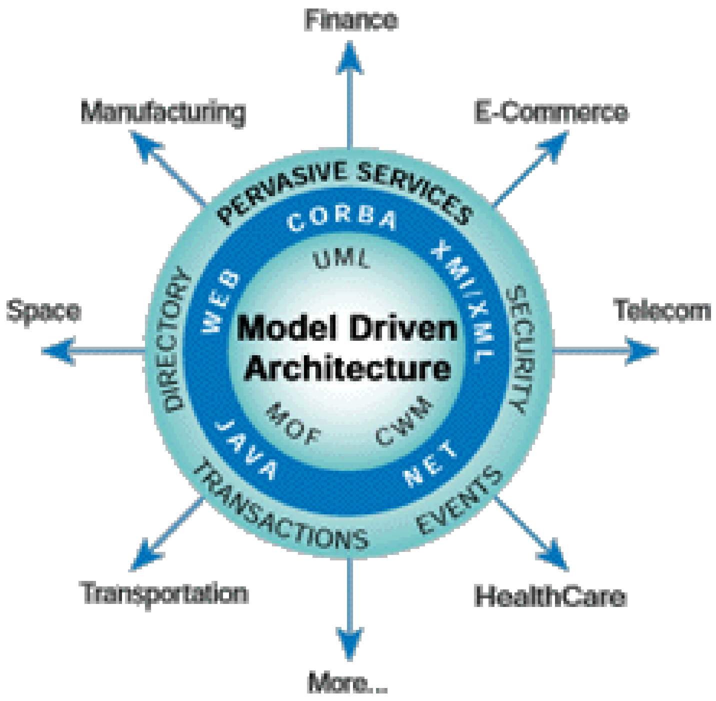
- Augment the message
- Warehouses the message
- …and assure Transactional delivery!.



Business Modeling and Monitoring Solution



MDA: Model Driven Architecture



www.omg.org/mda

Key Concept:

- An integration of best practices in Modeling, Middleware, Metadata and Software Architecture
- Based on standard Models, Metadata Models, and Model Transformations

Model Driven:

- (UML, MOF, CWM…)
- Platform Independent Business Models (PIM)
- Platform Specific Models (PSM)
- Mappings : PIM <==> PSM, PSM<==> PSM (Relative term!)

Metadata Driven:

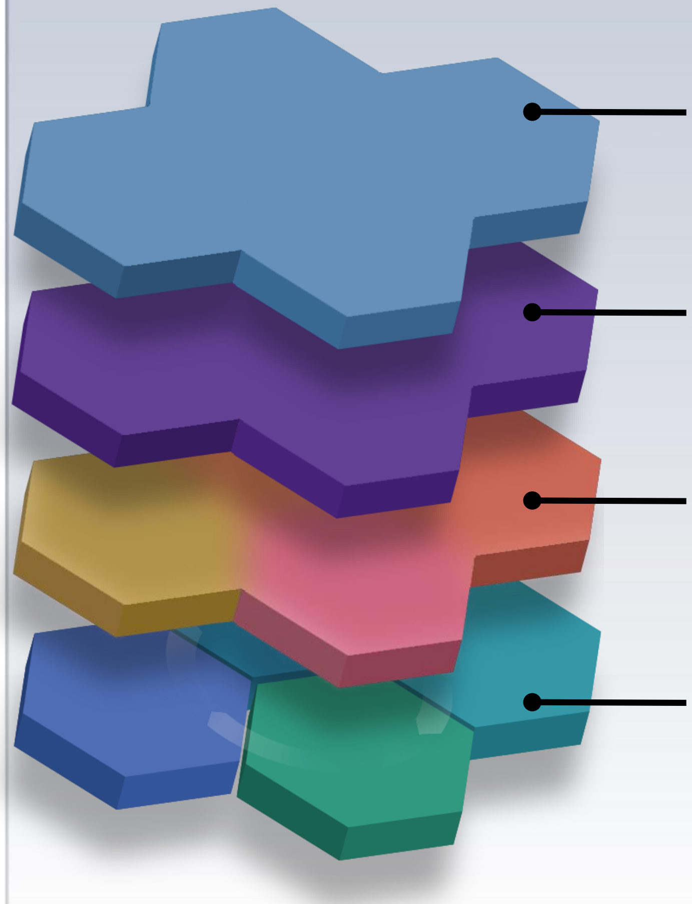
- (MOF, XSD, XMI)

Key Benefits:

- Improved Productivity for Architects, Designers, Developers and Administrators
- Lower cost of Application Development and Management
- Enhanced Portability and Interoperability
- Business Models and Technologies evolve at own pace on platform(s) of choice

What are the core elements that SOA brings together

Coming together under
Service Oriented Architecture



Skills - assistance, and best practices

Applications

Industry know-how and best practices linked to business

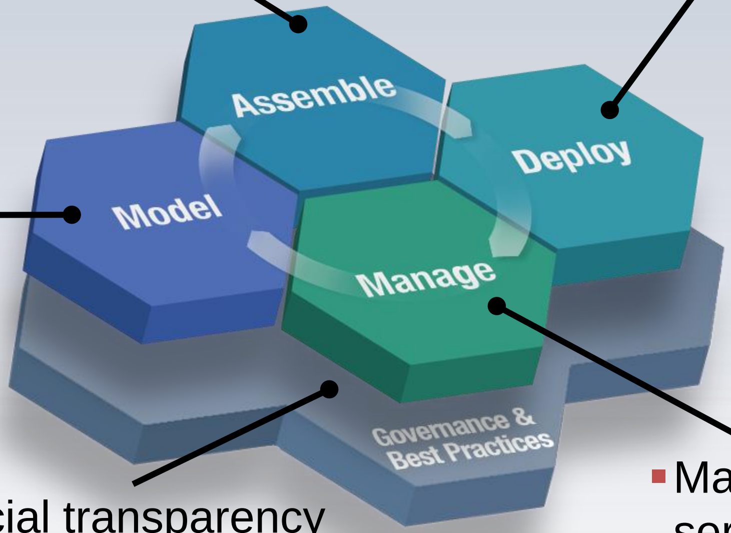
Flexible, robust infrastructure that reuses existing IT assets

The SOA Lifecycle .. For Flexible Business & IT

- Discover
- Construct & Test
- Compose

- Integrate people
- Integrate processes
- Manage and integrate information

- Gather requirements
- Model & Simulate
- Design



- Financial transparency
- Business/IT alignment
- Process control

- Manage applications & services
- Manage identity & compliance
- Monitor business metrics

Service-oriented architecture (SOA) definition

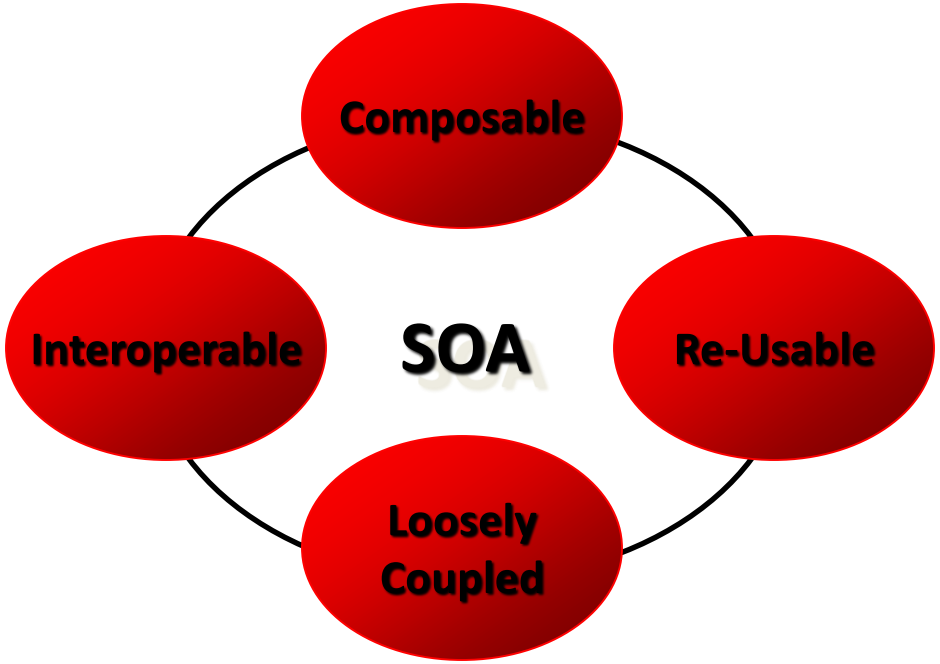
“A service-oriented architecture is essentially a collection of services. These services communicate with each other. The communication can involve either simple data passing or it could involve two or more services coordinating some activity. Some means of connecting services to each other is needed.”

(http://www.service-architecture.com/web-services/articles/service-oriented_architecture_soa_definition.html)

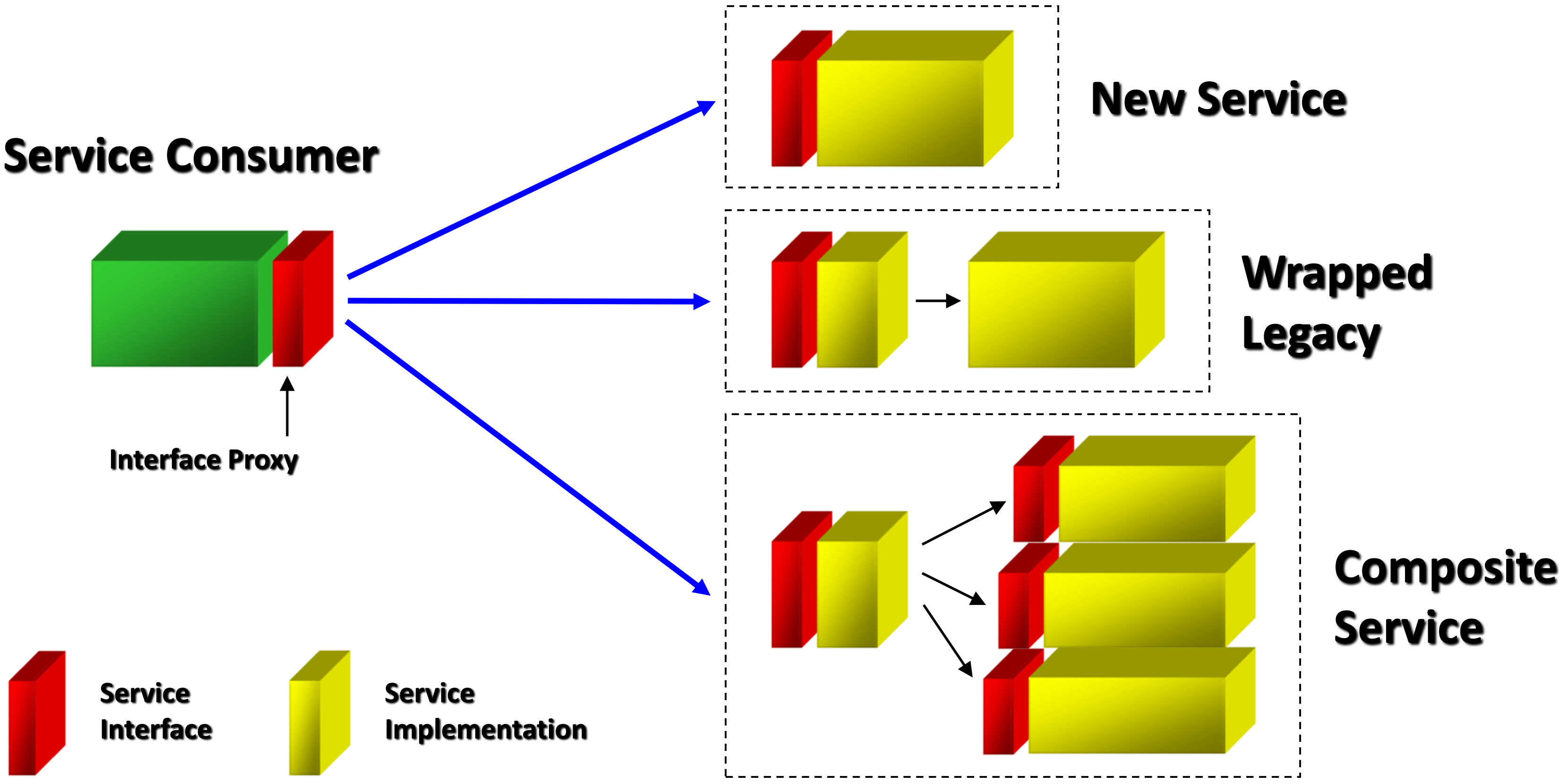
- Many other definitions can be found

Characteristics of SOA

- Services have platform independent, self describing interfaces (XML)
- Messages are formally defined
- Services can be discovered
- Services have quality of service characteristics defined in policies
- Services can be provided on any platform

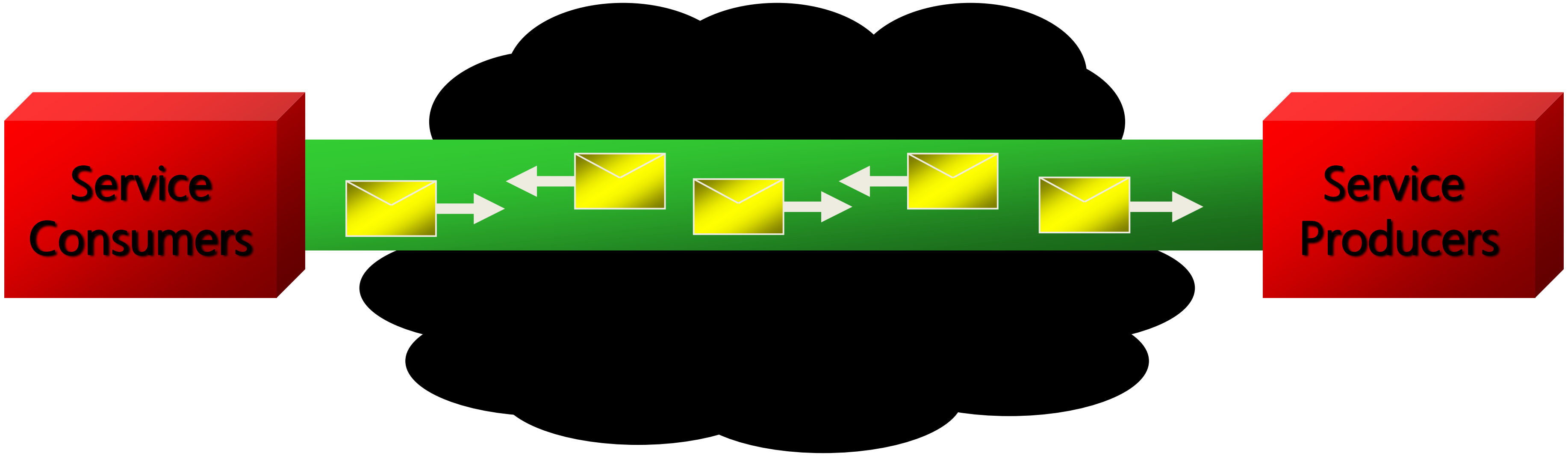


Anatomy of a Service

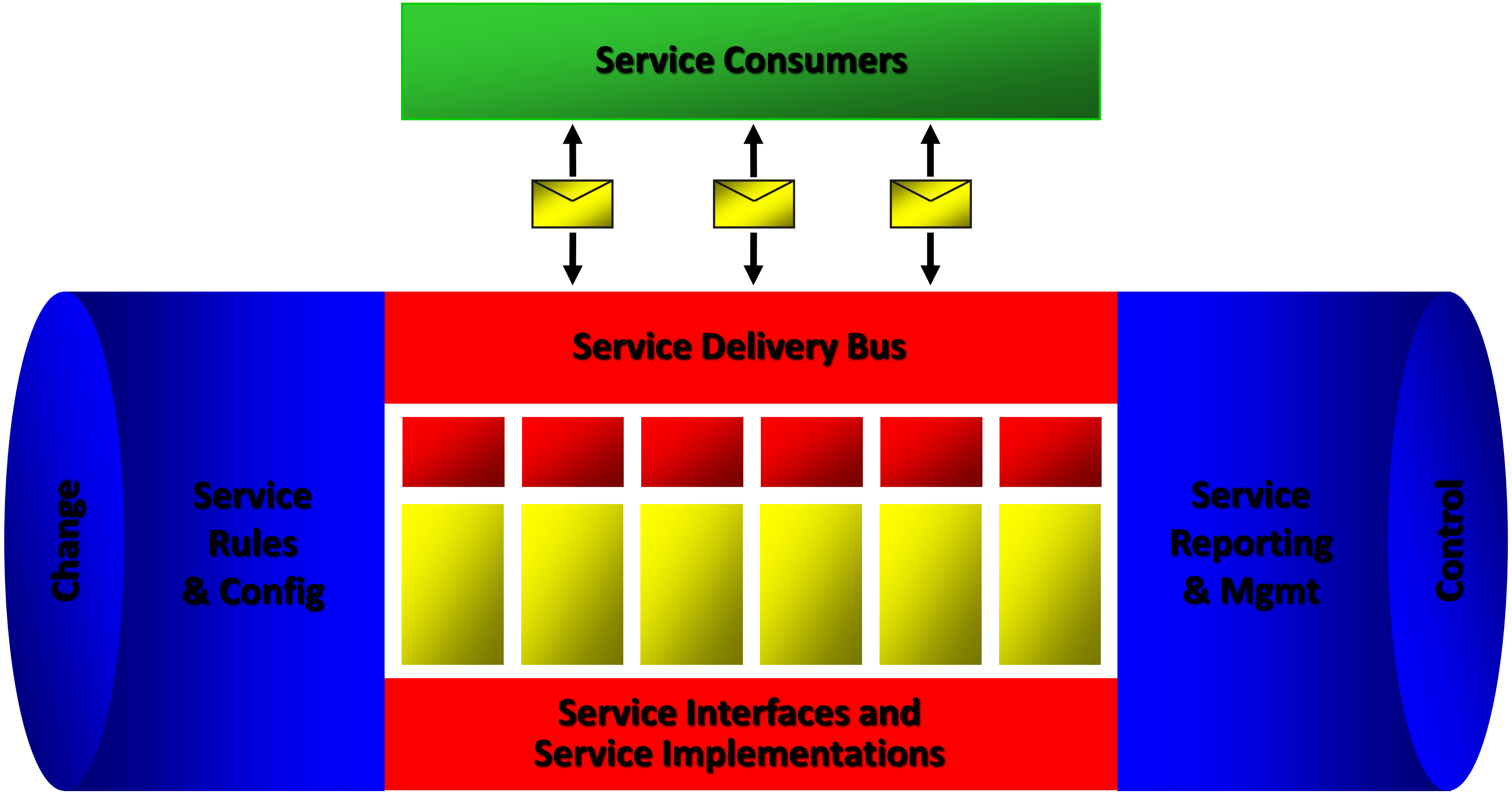


Service Communication

- Communicate with messages
- No knowledge about partner
- Likely heterogeneous



Service Platform

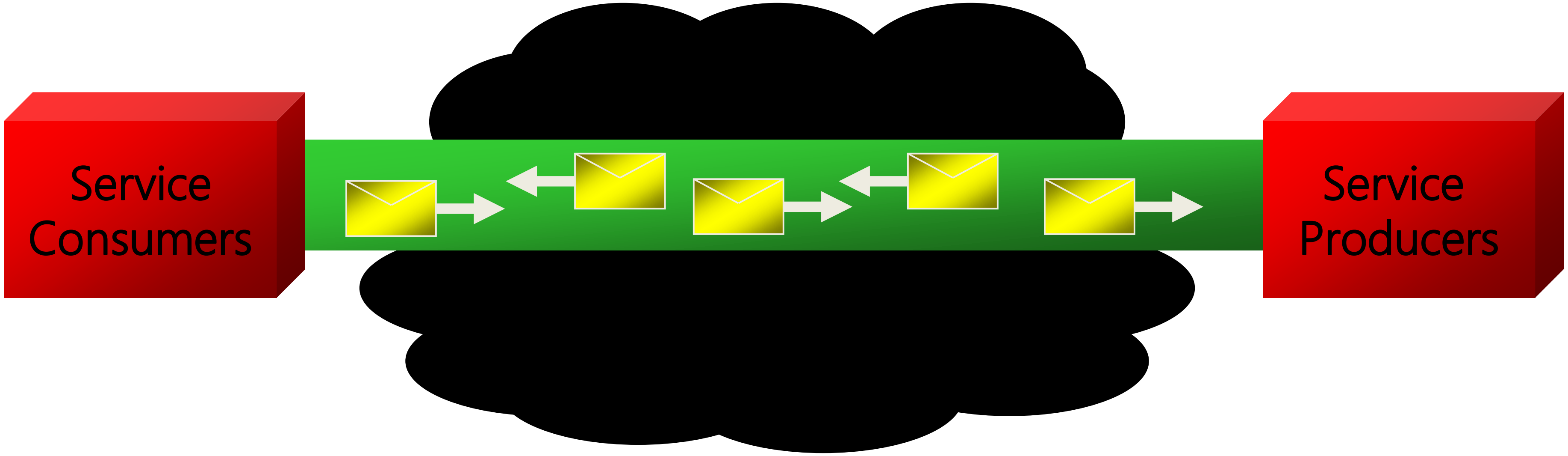


Benefits of SOA

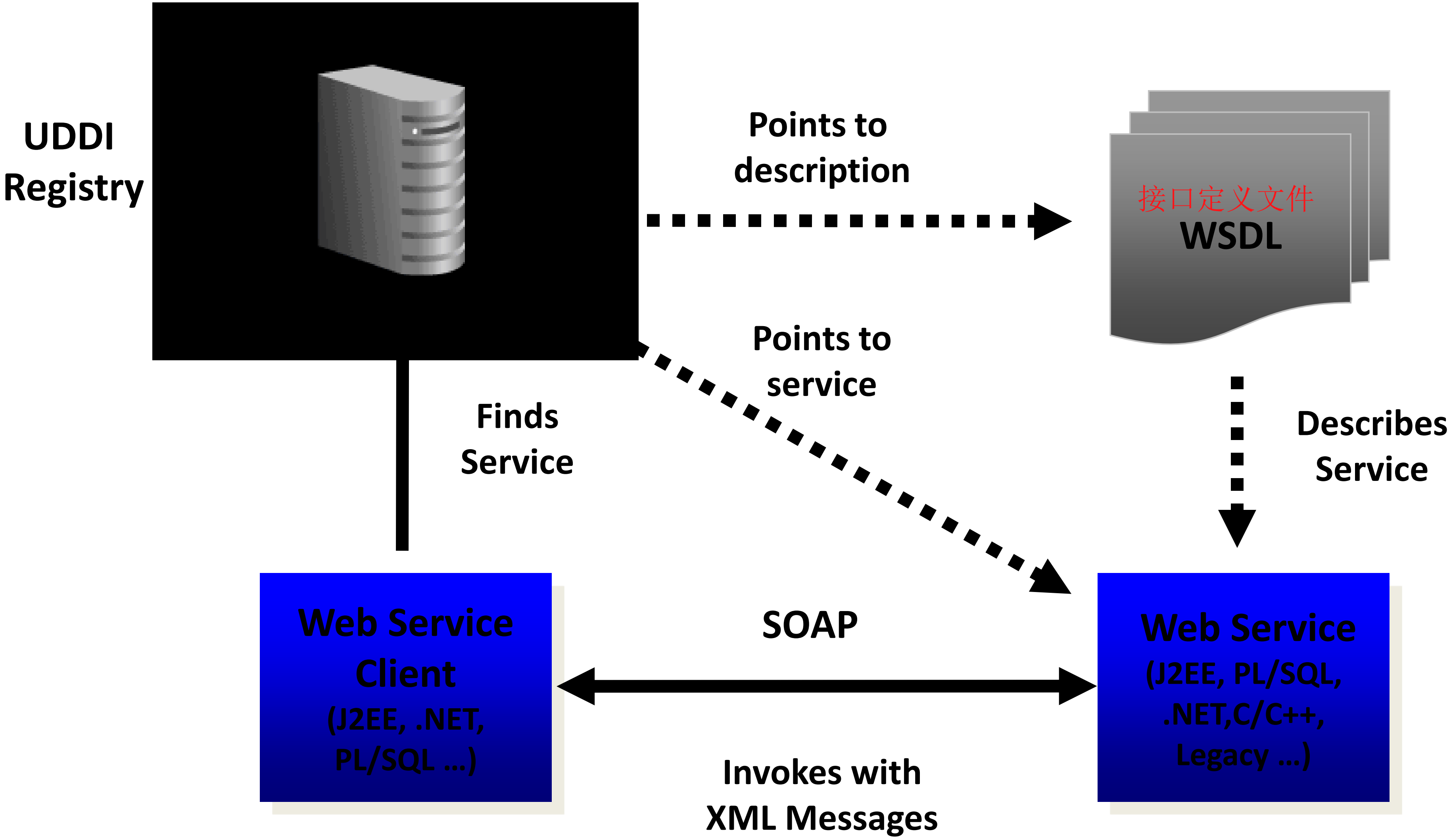
- Better reuse
  - Build new client functionality on top of existing Business Services
- Well defined interfaces
  - Make changes without affecting clients
- Easier to maintain
  - Changes/Versions are not all-or-nothing
- Better flexibility

Services Communicate With Messages

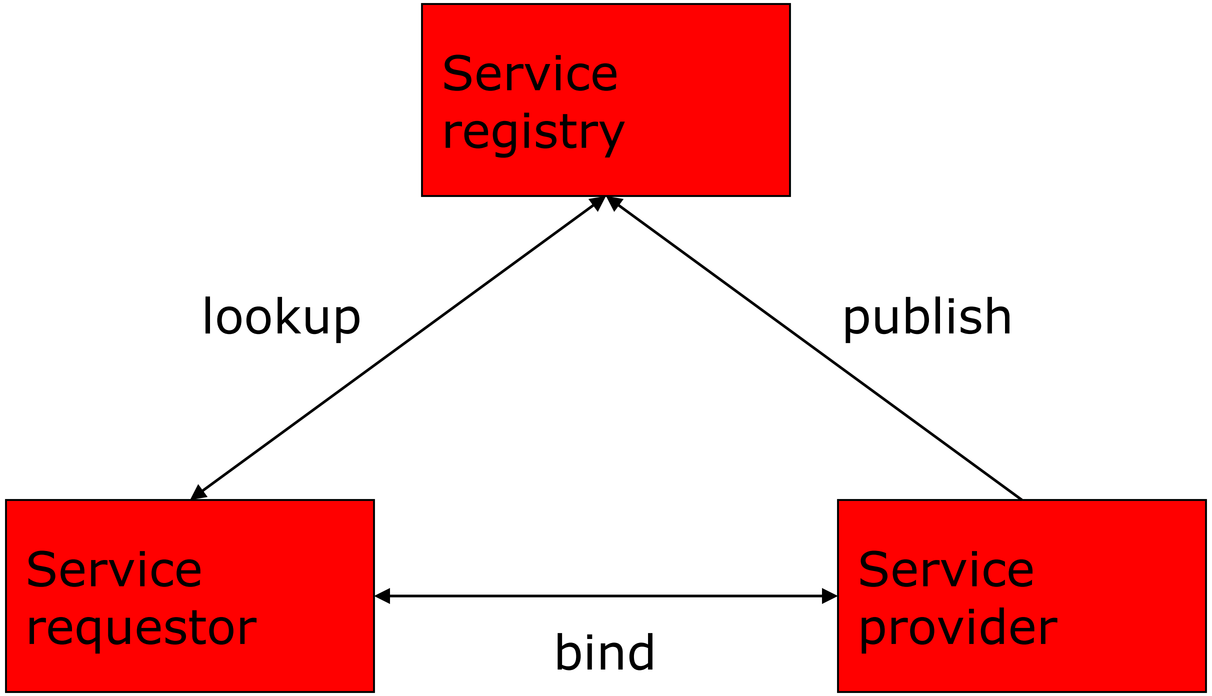
- Providing reliability and security to messages
- Sending messages across consumers and producers
- Service Orchestration



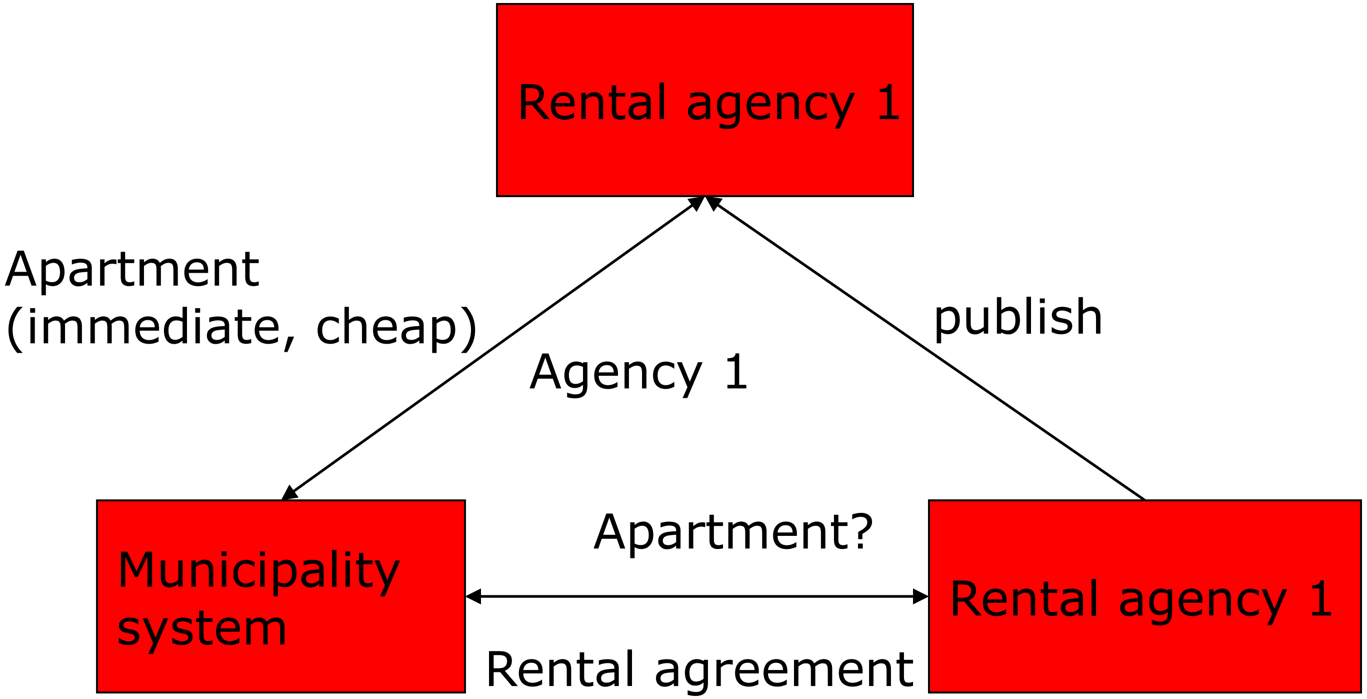
Basic Web Services



Service discovery



Service discovery



Service discovery

- Discovery is dynamic, each invocation may select a different one
- Primary criterion in selection: contract
- Selection may be based on workload, complexity of the question, etc ⇒ optimize compute resources
- If answer fails, or takes too long ⇒ select another service ⇒ more fault-tolerance

Is discovery really new?

- Many design patterns loosen coupling between classes
- Factory pattern: creates object without specifying the exact class of the object.

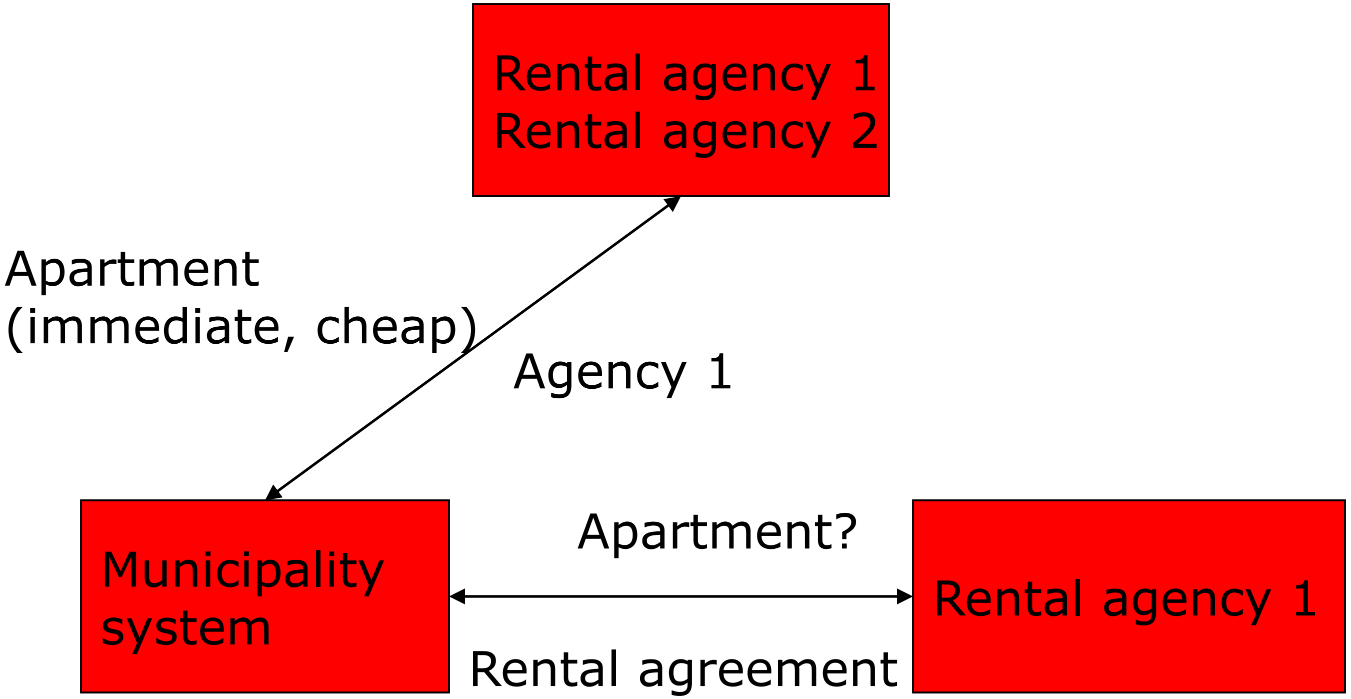
Services can be composed

- Service can be a building block for larger services
- Not different from CBSE and other approaches

Services adhere to a contract

- Request to registry should contain everything needed, not just functionality
- For “normal” components, much is implicit:
  - Platform characteristics
  - Quality information
  - Tacit design decisions
- Trust promises?
- Quality of Services (QoC), levels thereof
- Service Level Agreement (SLA)

Service discovery



Rental agency 1
Rental agency 2

Apartment
(immediate, cheap)

Agency 1

Municipality system

Apartment?

Rental agreement

Rental agency 1

Services are loosely coupled

- Rental agencies come and go
- No assumptions possible
- Stronger than CBSE loose coupling

Services are stateless

- Rental agency cannot retain information: it doesn’t know if and when it will be invoked again, and by whom

Services are autonomous, hide their logic

- Rental agency has its own rules on how to structure its process
- Its logic does not depend on the municipality service it is invoked by
- This works two ways: outside doesn’t know the inside, and vice versa

Services are reusable

- Service models a business process:
  - Not very fine grained
  - Collecting debt status from one credit company is not a service, checking credit status is
- Deciding on proper granularity raises lots of debate

Service use open standards

- Proprietary standards ⇒ vendor lockin

- There are lots of open standards:
  - How services are described
  - How services communicate
  - How services exchange data
  - etc

Services facilitate interoperability

- Because of open standards, explicit contracts and loose coupling
- Classical CBSE solutions pose problems:
  - Proprietary formats
  - Platform differences
  - Etc
- Interoperability within an organization (EAI) and between (B2B)

Service-Oriented Architecture

- Architecture:
  - the fundamental organization of a system in its components, their relationships to each other and to the environment and the principles guiding its design and evolution
- SOA: Any system made out of services?

Service bus

- Event-based messaging engine

- Origin: EAI, solve integration problems

- Often takes care of:
  - Mediation: protocol translation, data transformation, etc
  - Quality of Service issues: security, reliable delivery of messages, etc
  - Management issues: logging, audit info, etc.
  - Service discovery

- Can be central (broker, hub), or decentral (smart endpoints)

Service coordination

- Orchestration: central control
- Choreography: decentral control

Standards for Web Services

- Standards are managed by
- The **W3C** consortium
- **WS-I**, an organisation to promote the interoperability of web services (platform independent, vendor independent)
- **OASIS** (The Organization for the Advancement of Structured Information Standards)
- **WSDL** (Web Service Description/Definition Language) - XML format to specify the operations of a service
- **SOAP** (Simple Object Access Protocol) - one-way, stateless
- protocol to transfer XML data to a single receiver (since SOAP 1.2 there can be more receivers)

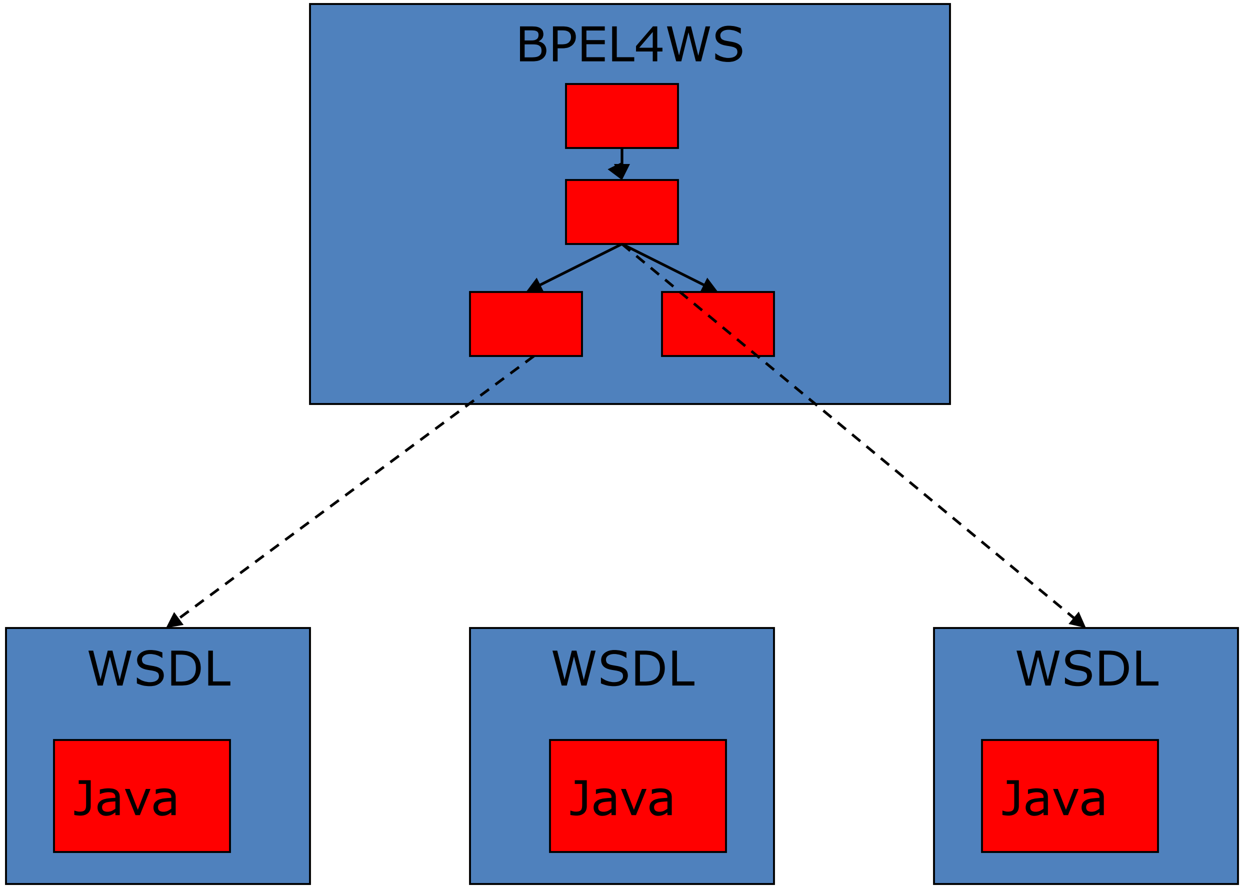
Standards for Web Services

- Additional standards (of lesser importance)
- **UDDI** (Universal Description, Discovery and Integration) - registry service for services
- **SAML** (Security Assertion Markup Language) - XML based
- framework for user authentication, description of authorization data
- **XKMS** (XML Key Management Specification) - management and registry of public keys

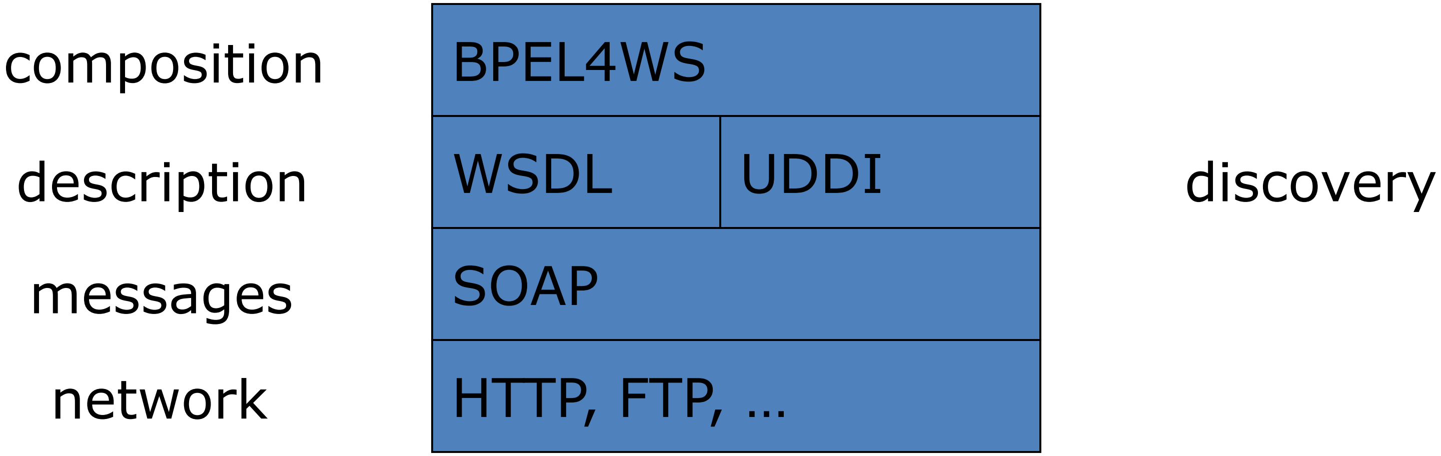
Web services

- Implementation means to realize services
- Based on open standards:
  - XML
  - SOAP: Simple Object Access Protocol
  - WSDL: Web Services Description Language
  - UDDI: Universal Description, Discovery and Integration
  - BPEL4WS: Business Process Execution Language for Web Services
- Main standardization bodies: OASIS, W3C

Coordination of Web services



Web services stack



composition — BPEL4WS

description — WSDL | UDDI — discovery

messages — SOAP

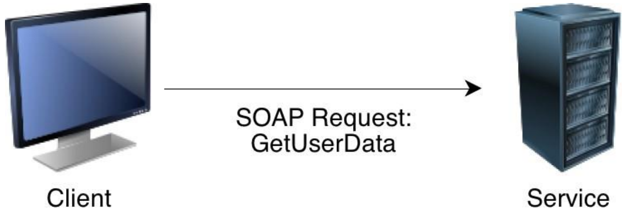
network — HTTP, FTP, …

XML

- Looks like HTML
- Language/vocabulary defined in schema: collection of trees
- Only syntax
- Semantic Web, Web 2.0: semantics as well: OWL and descendants

Example of WSDL & SOAP

- Definition of an operation: `GetUserData`
- The operation has an input and an output The input is defined via `GetUserDataRequest`
- The output is defined via `GetUserDataResponse`



SOAP

- Message inside an envelope
- Envelop has optional header (~address), and mandatory body: actual container of data
- SOAP message is unidirectional: it’s NOT a conversation

Example of SOAP

```
< soap:Envelope >
   <soap:Body  xmlns:m =" http: // www. example . org/ userdata ">
       < m:GetUserData >
               <m:email > user@example . org </ m:email >
       </ m:GetUserData >
    </ soap:Body >
</ soap:Envelope >
```

WSDL

- Four parts:
  - – Web service interfaces
  - – Message definitions
  - – Bindings: transport, format details
  - – Services: endpoints for accessing service. Endpoint = (binding, network address)

Example of WSDL

```
<wsdl >
< wsdl:operation name =" GetUserData ">
< wsdl:input message =" es:GetUserDataRequest "/>
< wsdl:output message =" es:GetUserDataResponse "/>
</ wsdl:operation >
< xsd:element name =" GetUserDataRequest ">
< xsd:complexType >
< xsd:sequence >
< xsd:element name =" username " type=" string "/>
< xsd:element name =" role " type=" string "/>
</ xsd:sequence >
</ xsd:complexType >
</ xsd:element >
< xsd:element name =" GetUserDataResponse ">
< xsd:complexType >
                           <xsd:all >
< xsd:element name =" email " type =" string "/>
                          </ xsd:all >
</ xsd:complexType >
</ xsd:element >
<wsdl >
```

数据表示需要完全一样，编程语言可以不一样，xsd定义了所有原始数据类型，统一标准

Orchestration of Services

- *How are services interconnected with each other?*
- Applications are realized by a combination of services
- Orchestration takes care which services interact and when Control flow can be automated via workflow engines Should help reuse by loose coupling and flexibility

Binding of Services

- Development-time binding vs. runtime binding
- **Development-time binding** is far simpler
- The services, their API and address are fixed during development time
- **Runtime binding** is more complex
- The exact services and addresses are found during runtime Lookup by service name often the best approach

Discovery of Services

- *How do I find the service for my needs?*
- UDDI aimed at a lookup service for businesses, organisation and services
- UDDI as yellow pages for web services
- No widespread use
- Should help addressability

UDDI

- Three (main) parts:
  - Info about organization that publishes the services
  - Descriptive info about each service
  - Technical info to link services to implementation

UDDI (cnt’d)

- Original dream: one global registry
- Reality: many registries, with different levels of visibility
  - Mapping problems

Service Types

- Typical service (component) types in SOA systems:
- **Application frontend** - typically not a service, initiate operation and receive results
- **Basic services** - they build the foundation
- **Intermediary services** - adapters and facades to add functionality, typically stateless
- **Process centric services** - implement the business logic, typically manage the process state
- **Public services** - for integration, higher level functionality

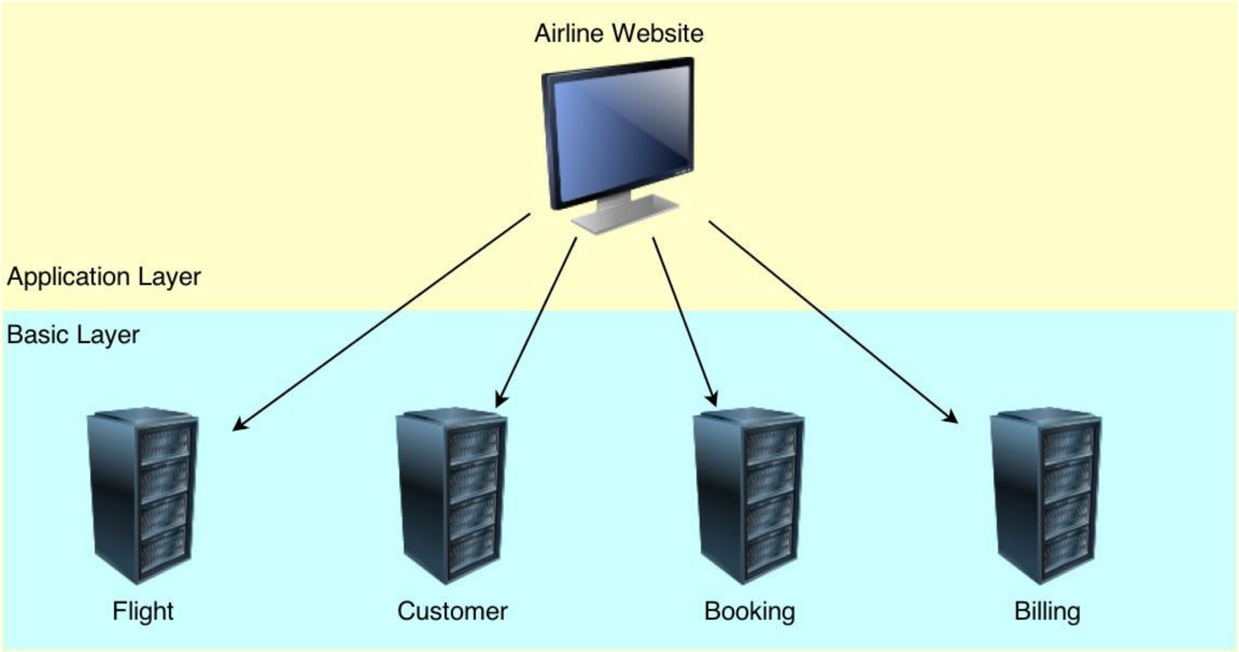
SOA and Software Architecture

- SOA aims to decouple the system from the software architecture No alignment between the system layers and the service layers
- system layers: browser, application server, web server, operation system
- service layers: application, process, intermediary, basic

Example: Simple SOA Architecture

- In the most simple case the SOA architecture consists of two layers
    - The application layer, which uses the basic services
    - The layer for the basic services
    - For example: A web site of an airline

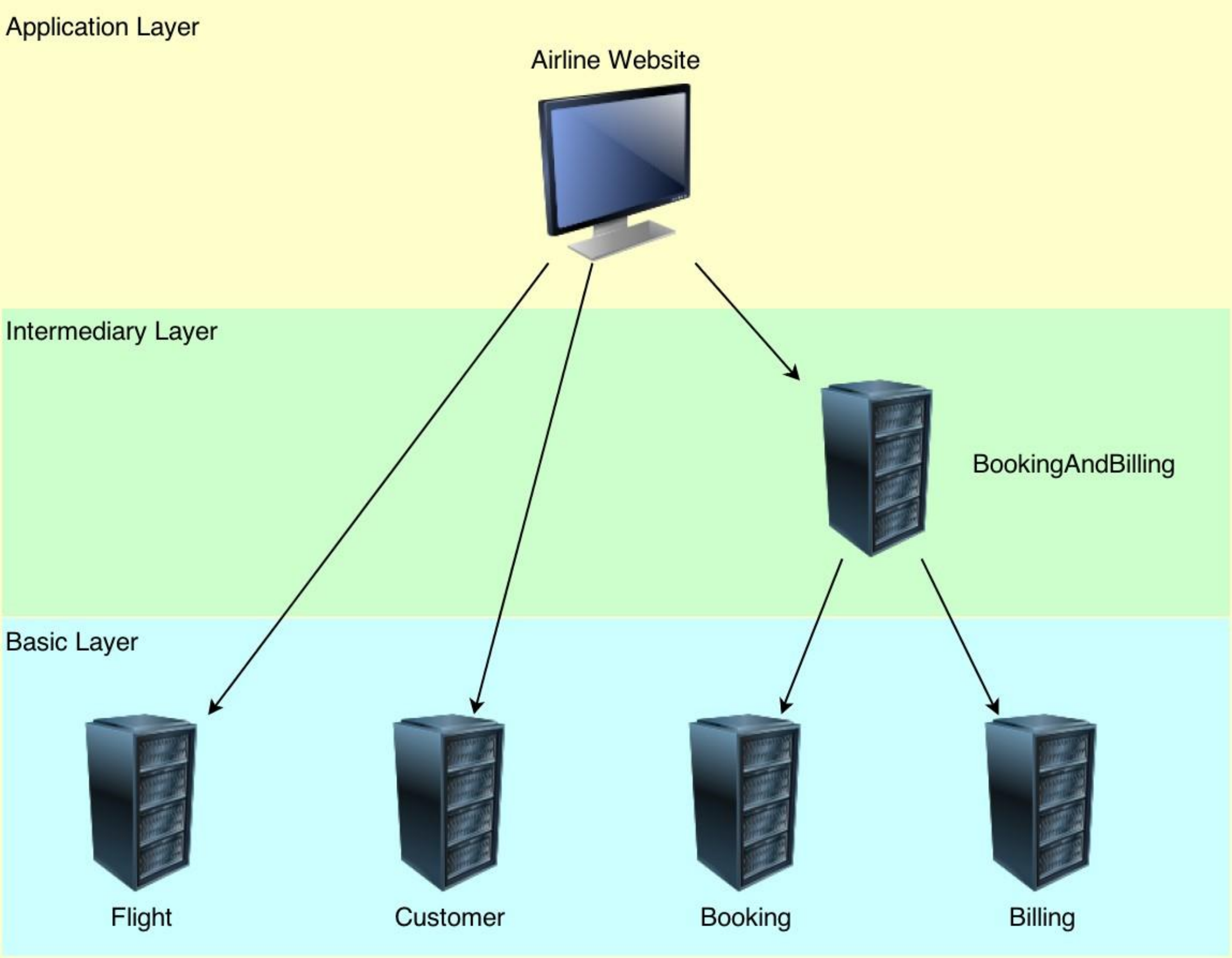
Example: Simple SOA Architecture



Facade Pattern

- Intermediary services often serve as facade for basic services
- They aggregate the functionality
- Example for a n-tier architecture

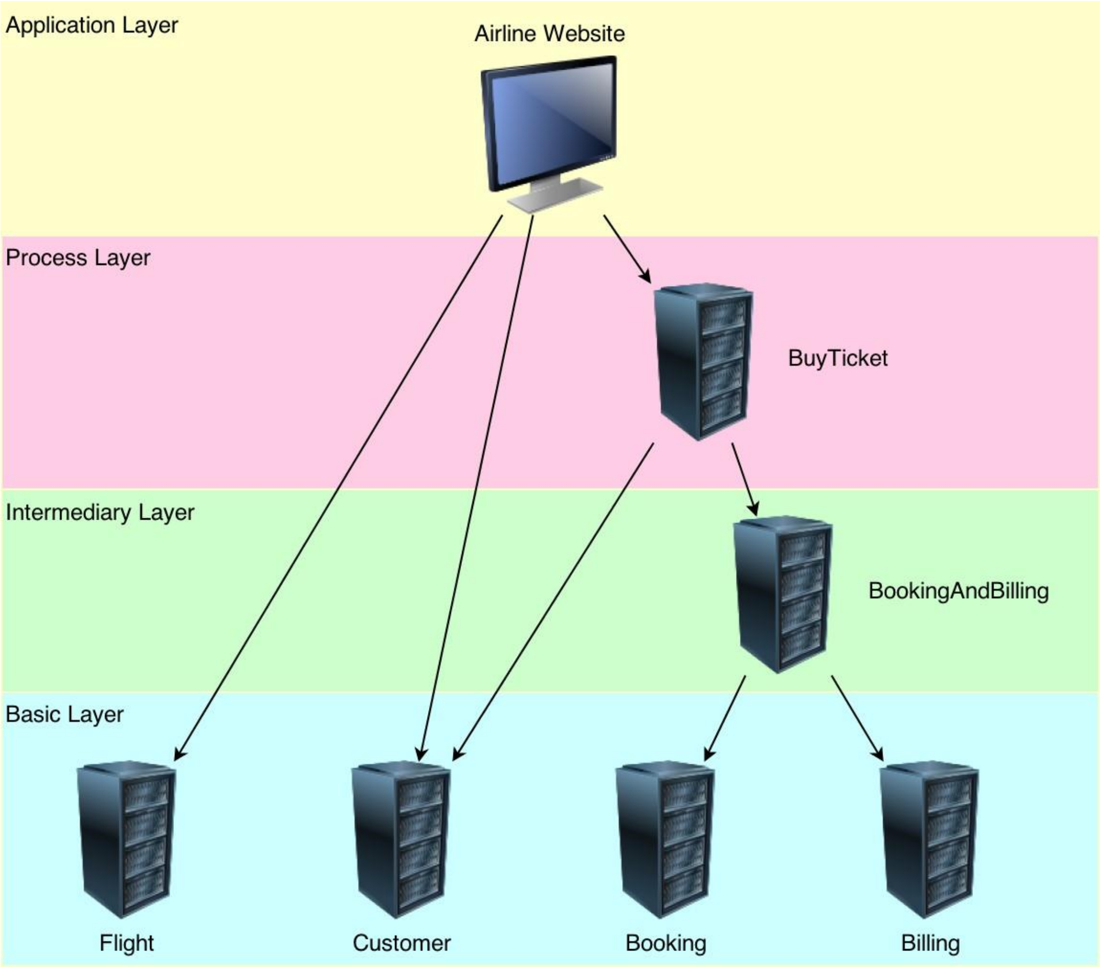
Example: Facade Pattern



Process-centric Services

- Optional component in SOA systems
- Process-centric services encapsulate process logic and application state
- An application frontend may delegate the process control to such a component
- Advantage: reuse when the process-centric services is shared by multiple clients
- Disadvantage: more complex system, process control might be split into multiple components

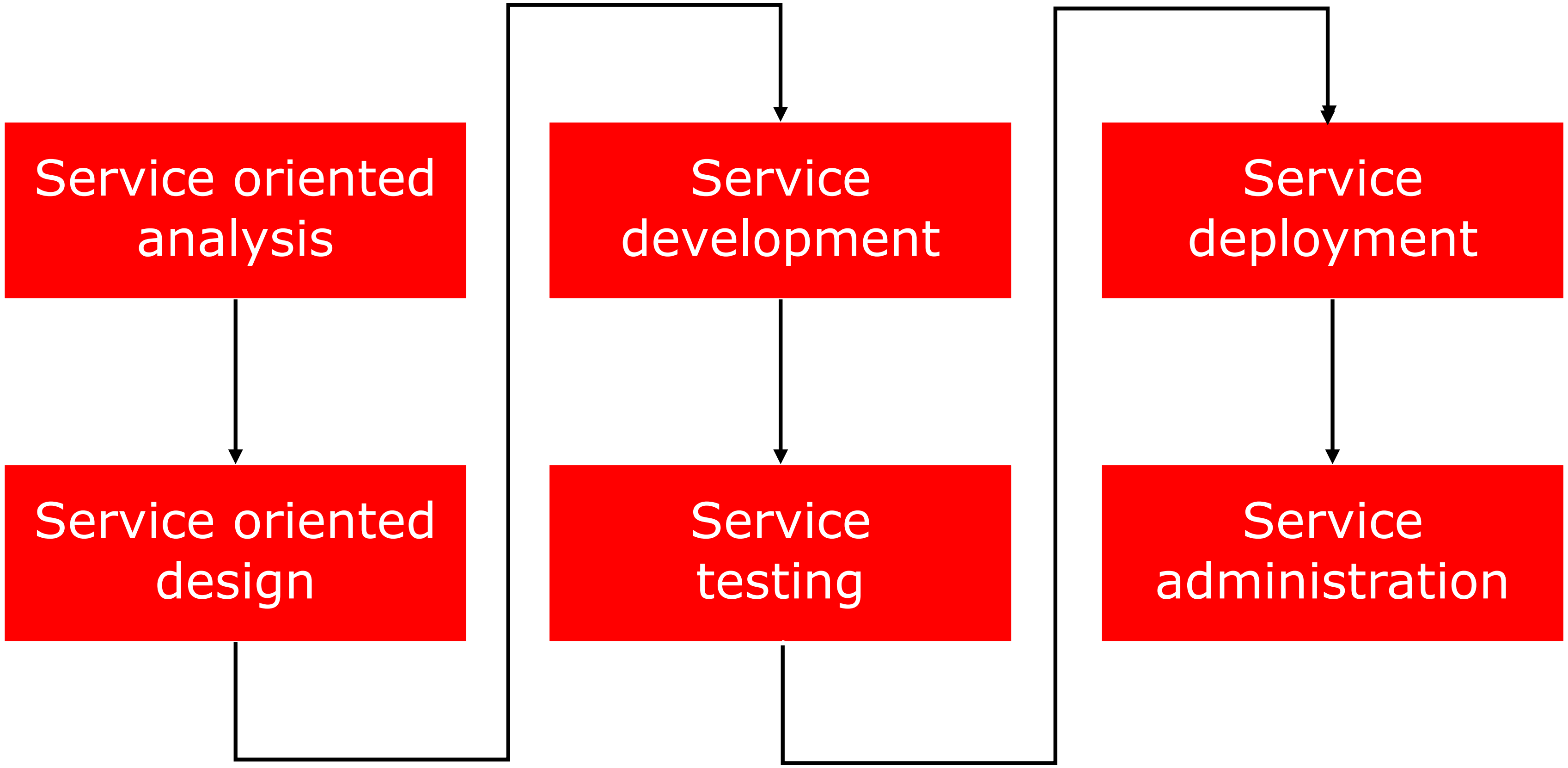
Example: Process-centric Services



BPEL4WS

- Three main parts:
  - Partnerlinks: dependencies between services: who sends what to whom
  - Global variables
  - Workflow model: “program”
- BPEL4WS is an orchestration language; executable
- WS-CDL (Web Services Choreography Description Language) is a choreography language; not executable

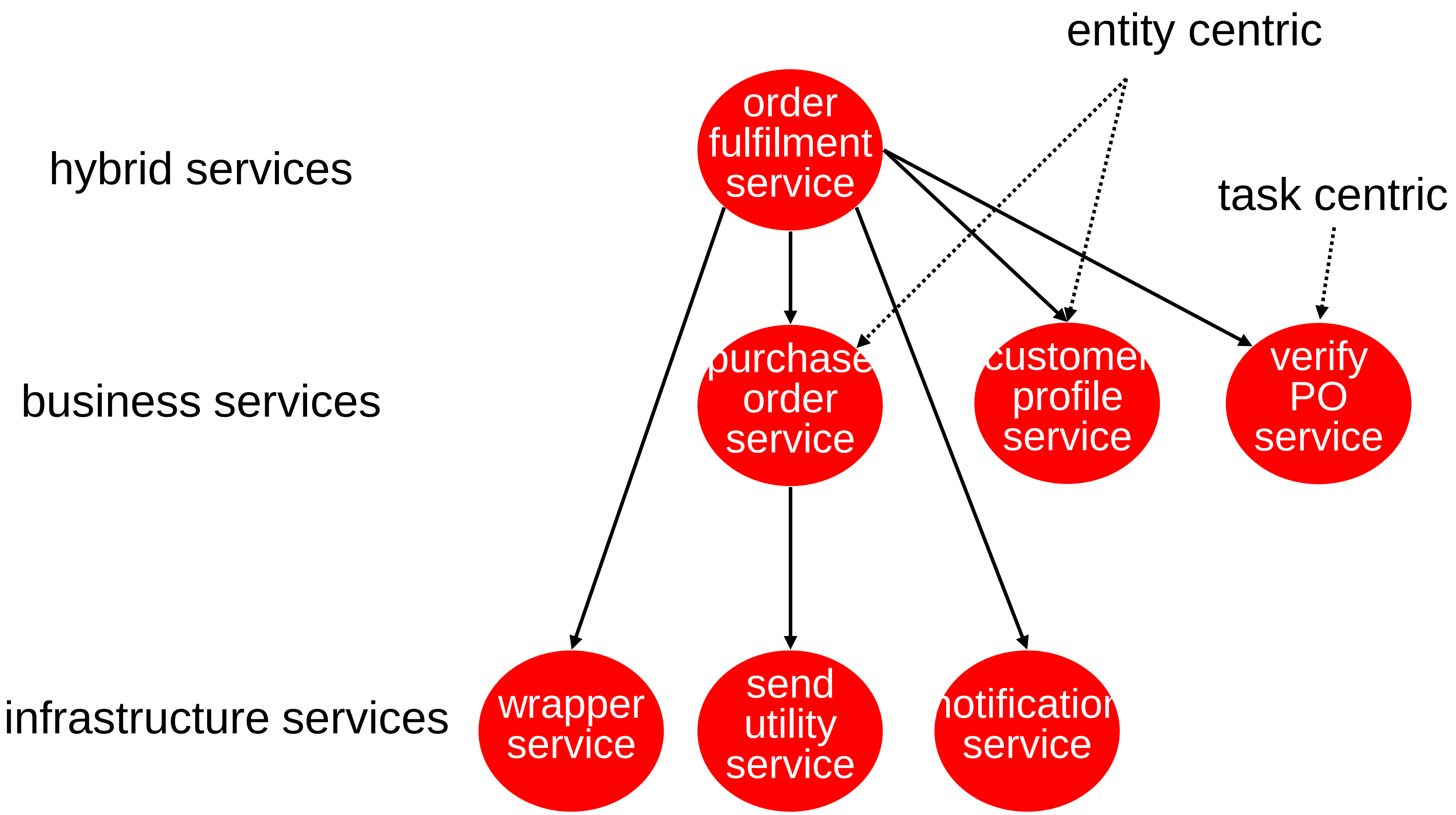
Service-Oriented Software Engineering (SOSE) life cycle



Terminology

- service oriented environment (or service oriented *ecosystem)*
- *business process + supporting services*
  - *application (infrastructure) service*
  - *business service*
    - *Task-centric business service*
    - *Entity-centric business service*
  - *hybrid service*

Terminology



entity centric

task centric

hybrid services

business services

infrastructure services

order fulfilment service

purchase order service

customer profile service

verify PO service

wrapper service

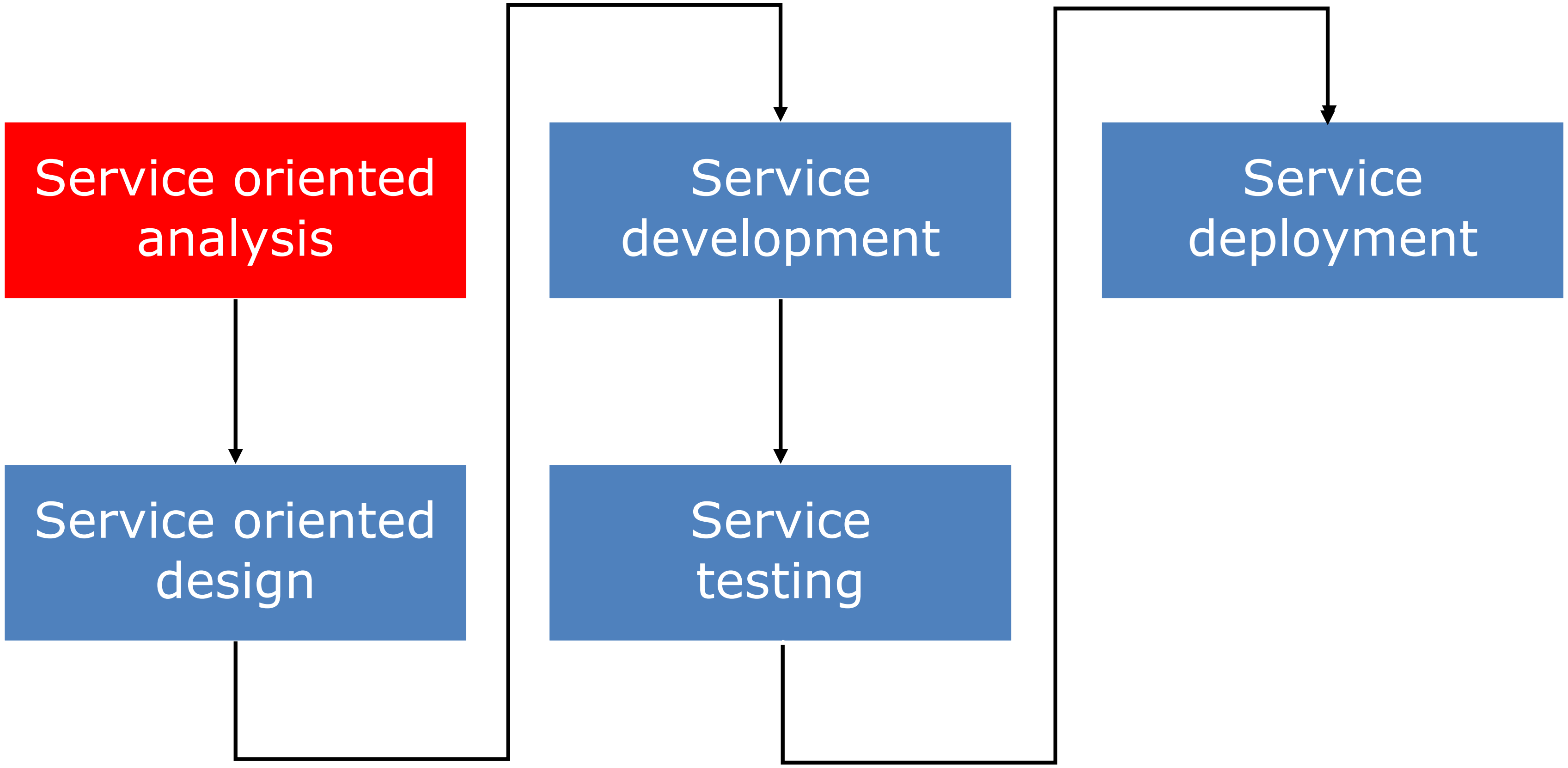
send utility service

notification service

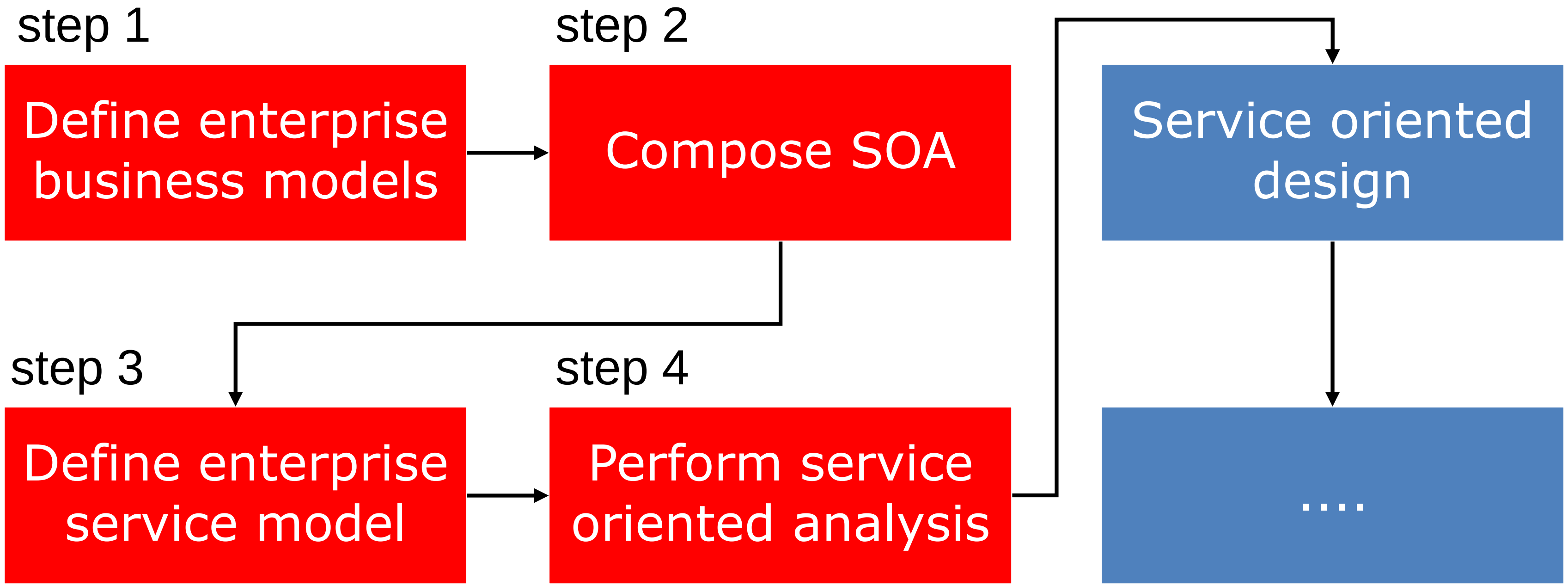
Strategies for life cycle organization

- Top-down strategy
- Bottom-up strategy
- Agile strategy

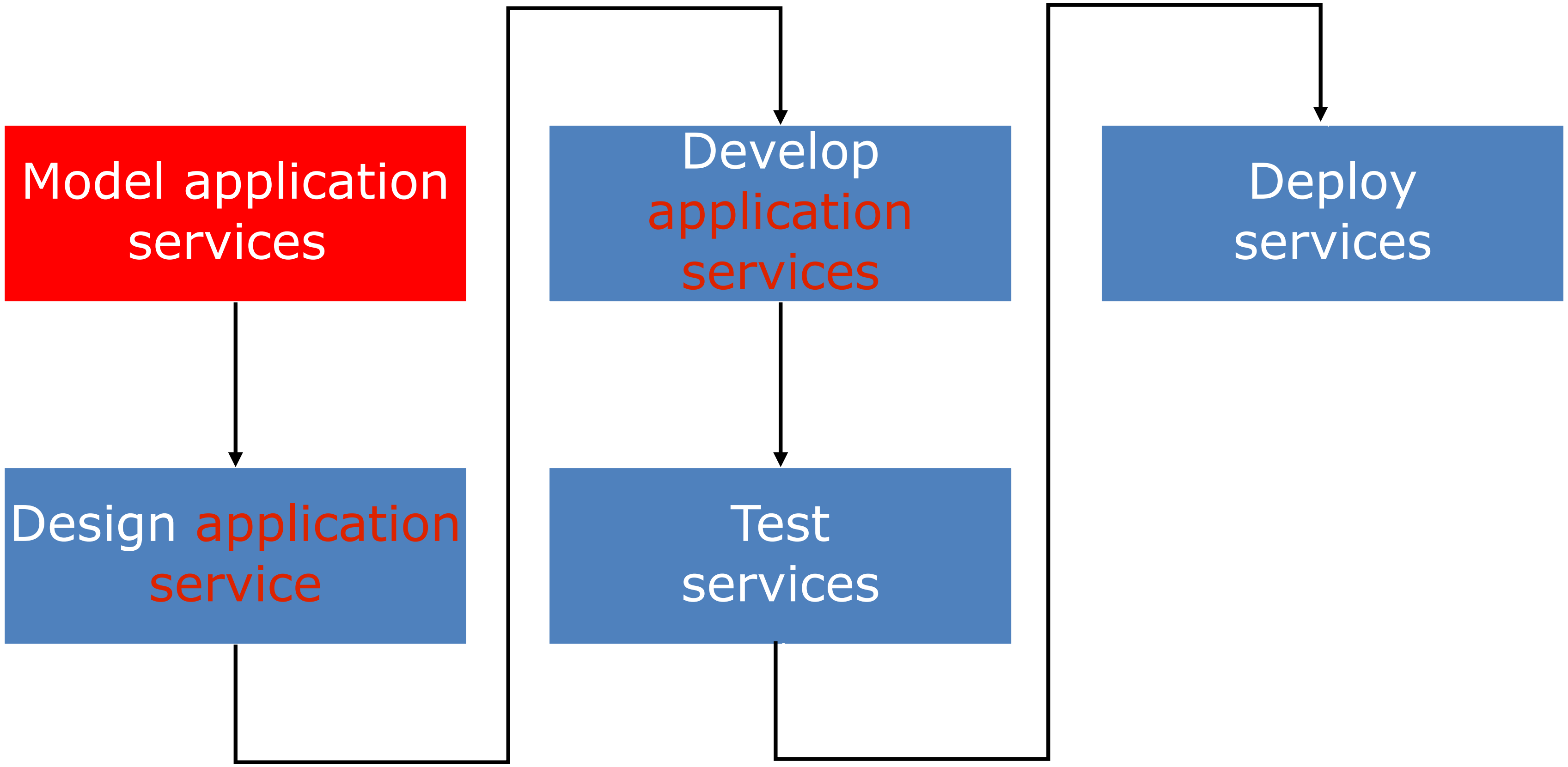
Top-down strategy



Top-down SO analysis


step 1
Define enterprise business models
step 2
Compose SOA
Service oriented design
step 3
Define enterprise service model
step 4
Perform service oriented analysis
....

Bottom-up strategy



Model application services

Design application service

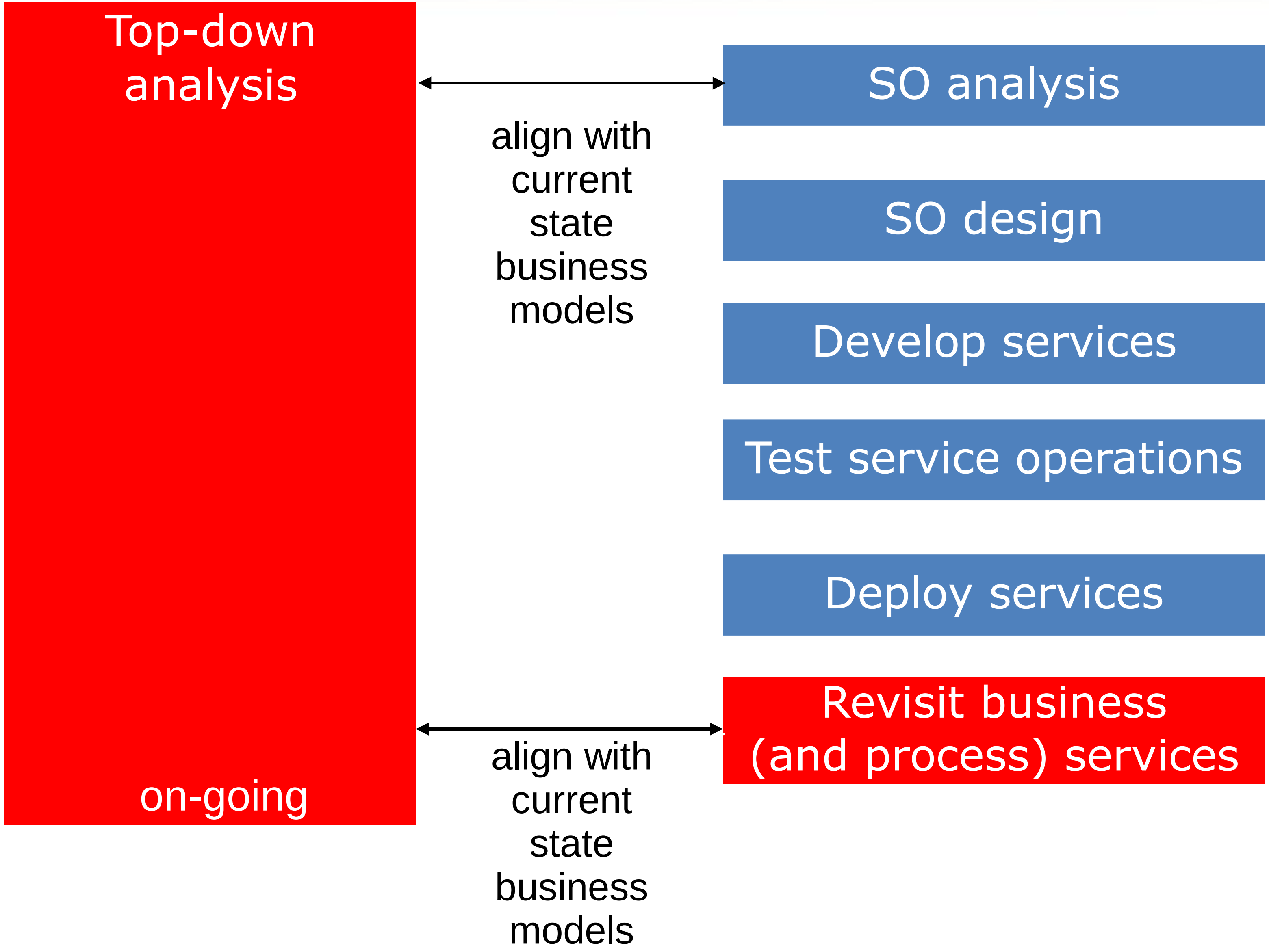
Develop application services

Test services

Deploy services

application service = infrastructure service

Agile strategy



Top-down analysis

on-going

align with current state business models

SO analysis

SO design

Develop services

Test service operations

Deploy services

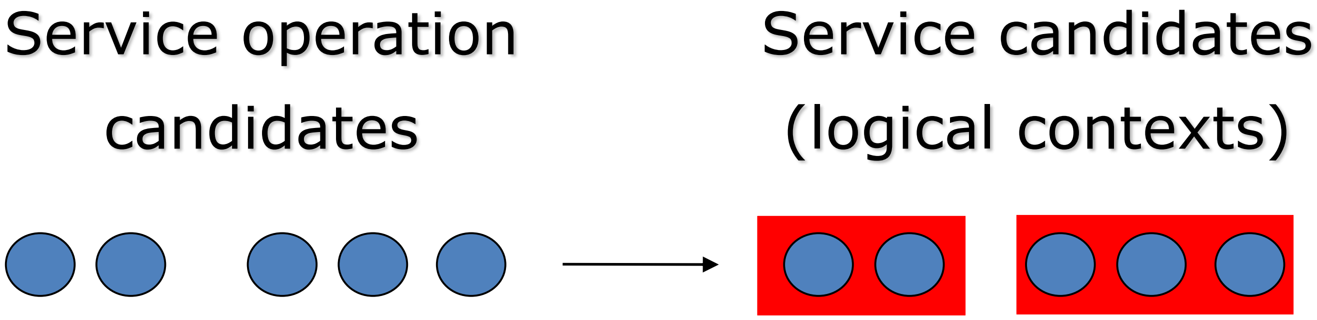
Revisit business (and process) services

align with current state business models

Service oriented analysis

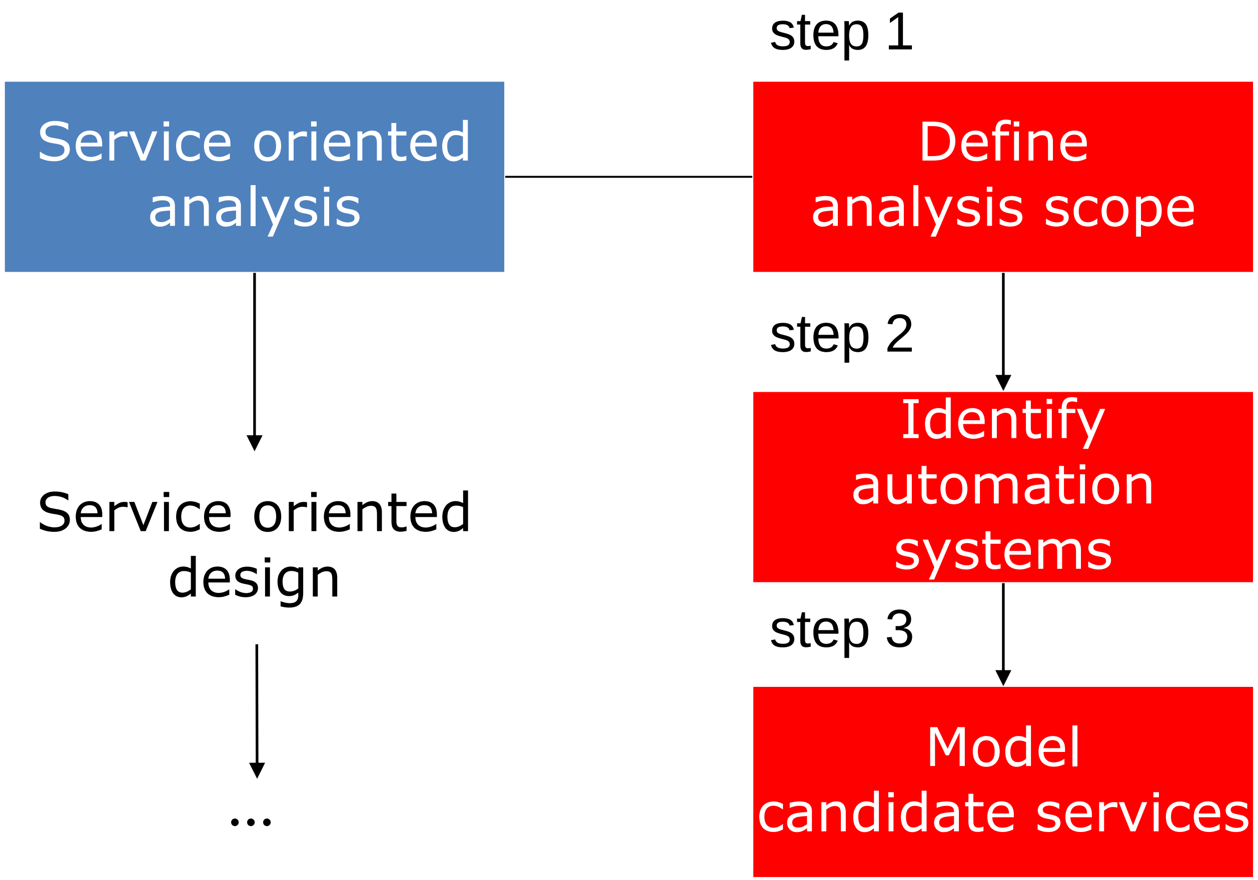
- The process of determining how business automation requirements can be represented through service orientation

Goals of SO analysis


Service operation candidates
Service candidates (logical contexts)

- Appropriateness for intended use
- Identify preliminary issues that may challenge required service autonomy
- Define known preliminary composition models

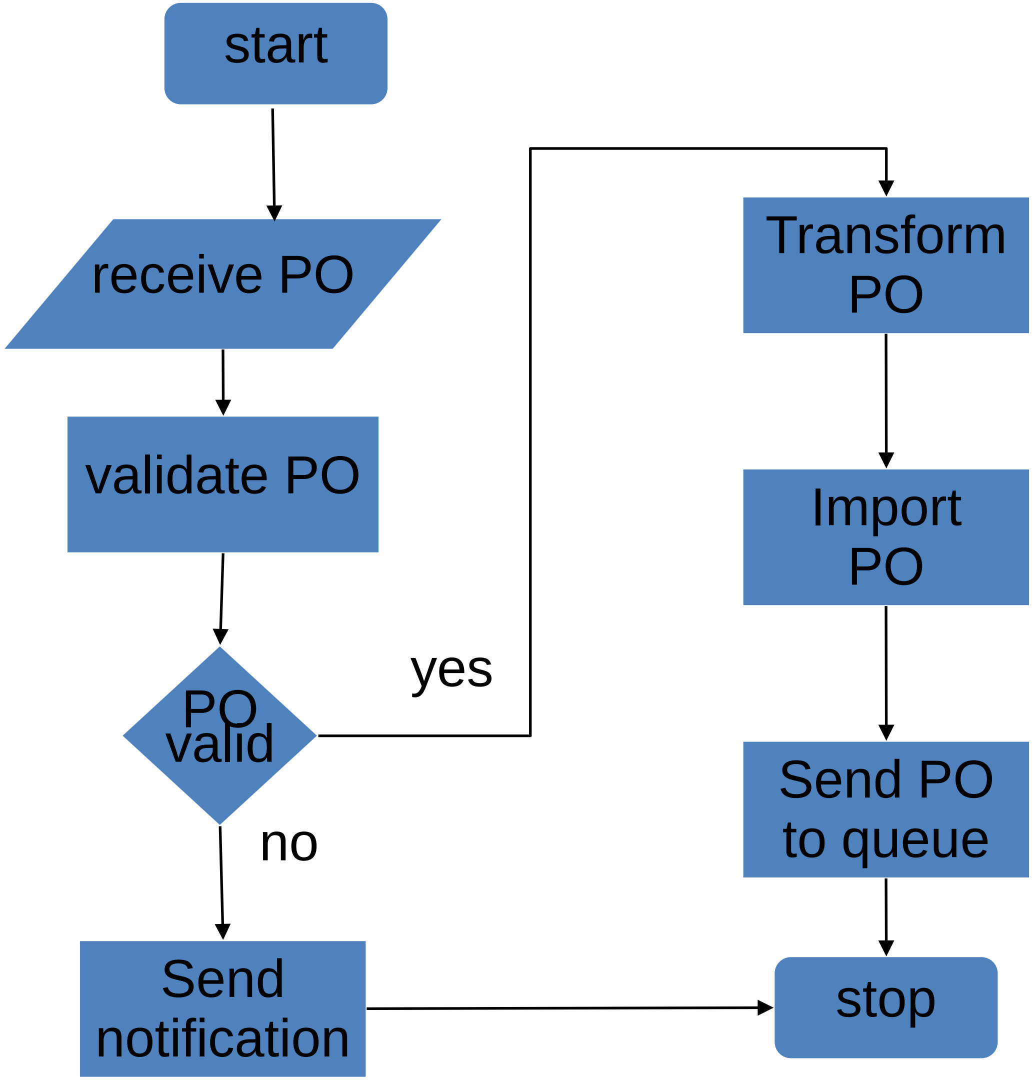
3 Analysis sub-steps



Step 1: Define analysis scope

- – Mature and understood business requirements
  - S = ∑i Si, where smaller services may still be quite complex
- – Can lead to
  - process-agnostic services/service operations (generic service portfolio)
  - services delivering business-specific tasks
- – Models: UML use case or activity diagrams

Order Fulfillment Process

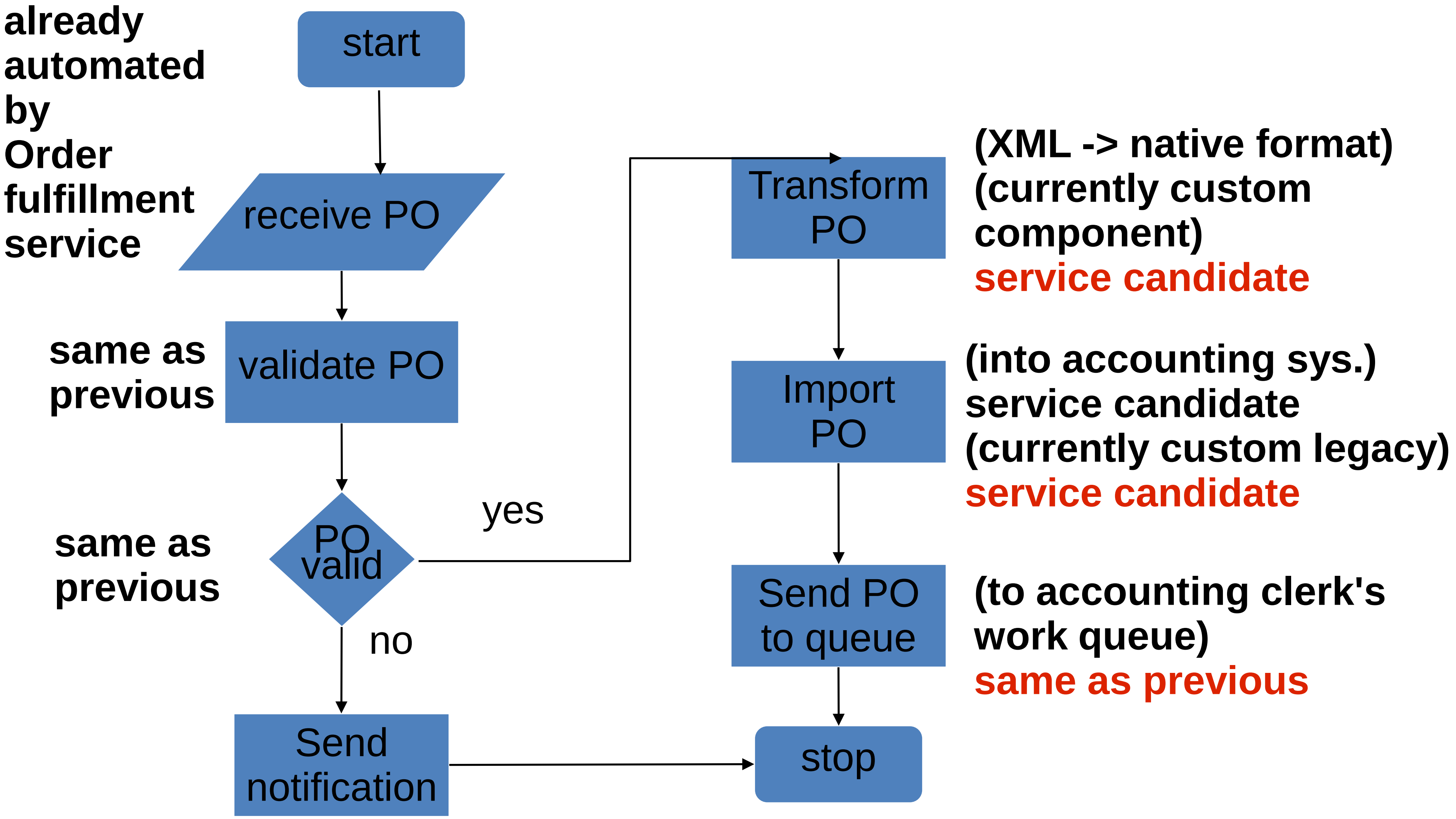


Step 2: Identify automation systems

- What is already implemented?
  - encapsulate
  - replace

- Models: UML deployment diagram, mapping tables

Order Fulfillment Process



already automated by Order fulfillment service

same as previous

same as previous

(XML -> native format)
(currently custom component)
service candidate

(into accounting sys.)
service candidate
(currently custom legacy)
service candidate

(to accounting clerk's work queue)
same as previous

Step 3: Model candidate services

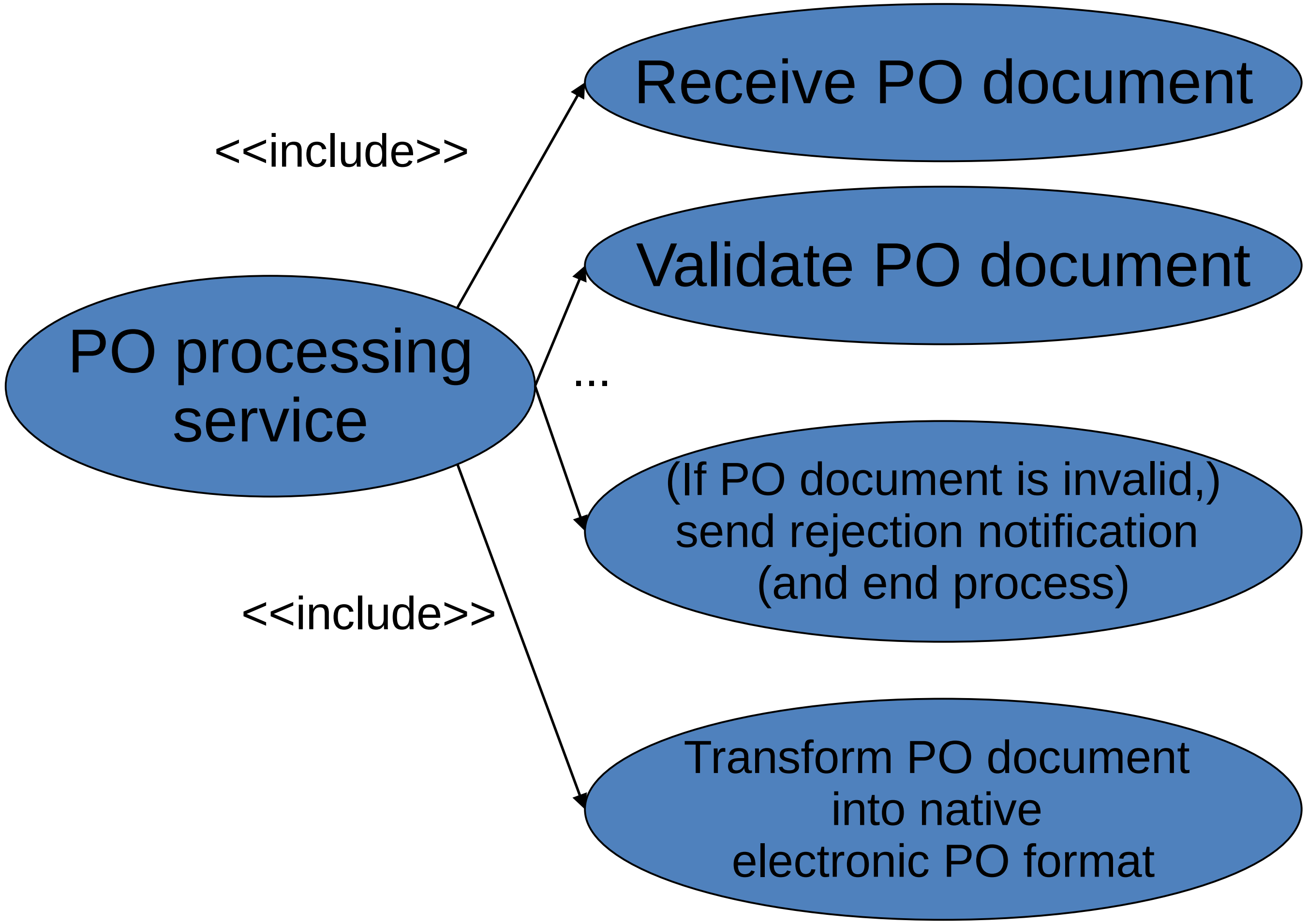
- How to compose services?

- Service (candidates) conceptual model
  - operations + service contexts
  - SO principles

- Focus on task- and entity-centred services

- Models: BPM, UML use case or class diag.

Example service operation candidates


Receive PO document
<<include>>
Validate PO document
PO processing service
...
(If PO document is invalid,) send rejection notification (and end process)
<<include>>
Transform PO document into native electronic PO format

Example business process logic

- Not service operation candidates
  - – if PO document is valid, proceed with the transform PO document step
  - – if the PO document is invalid, end process

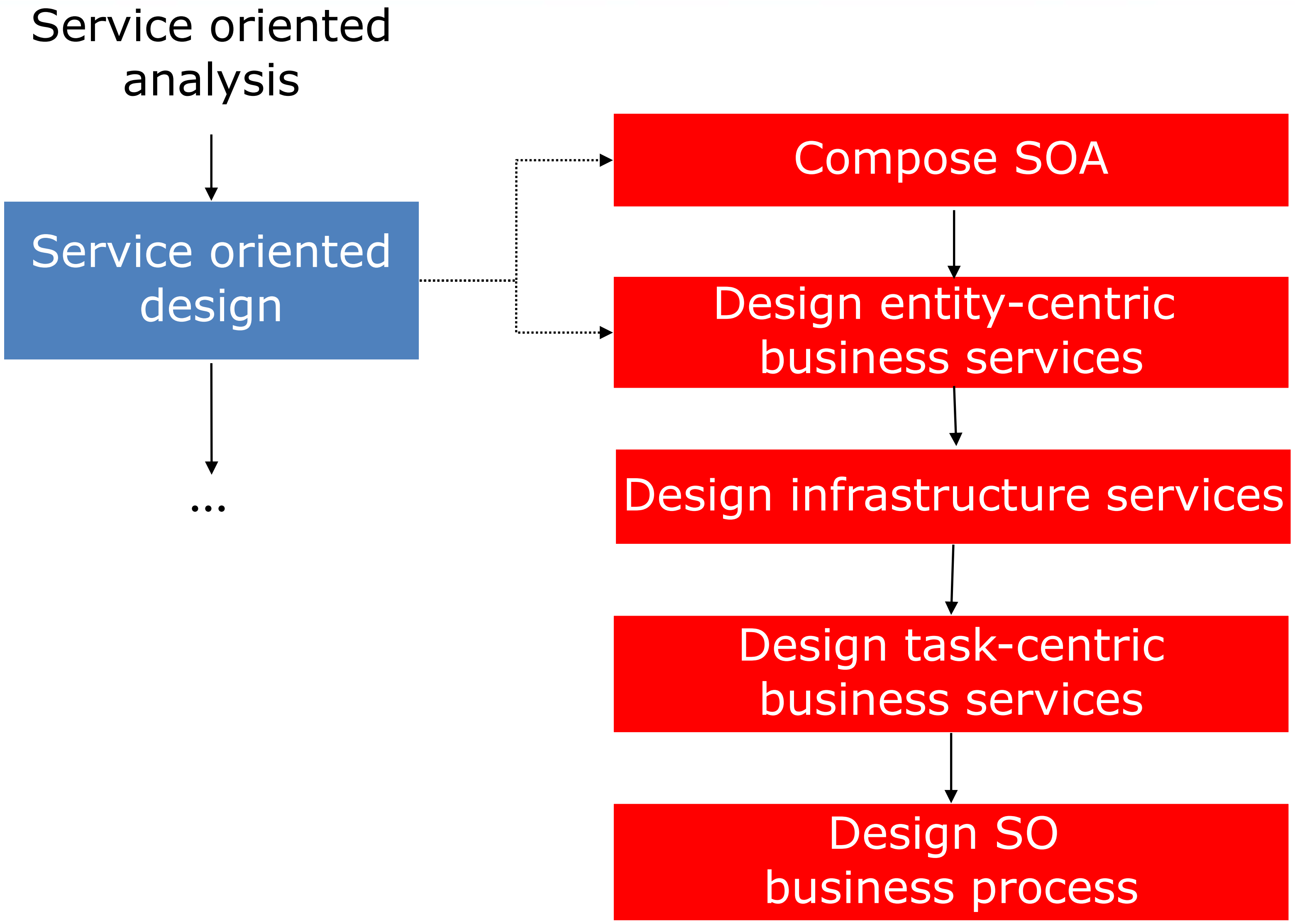
Task- versus entity-centred services

- Task-centred
  - (+) direct mapping of business requirements
  - (-) dependent on specific process
- Entity-centred
  - (+) agility
  - (-) upfront analysis
  - (-) dependent on controllers

Benefits of business-centric SOA

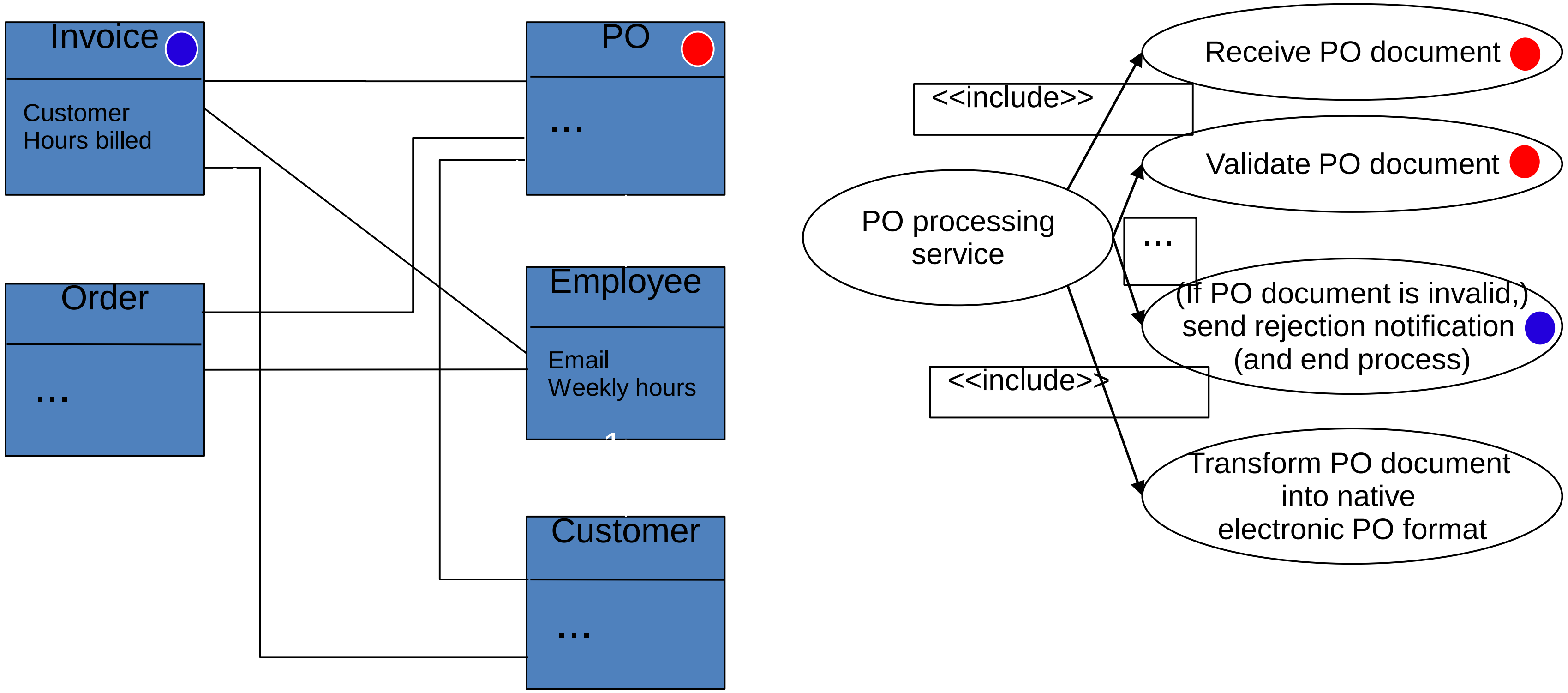
- introduce agility
- prepare for orchestration
- enable reuse

Service-oriented design: design sub-steps

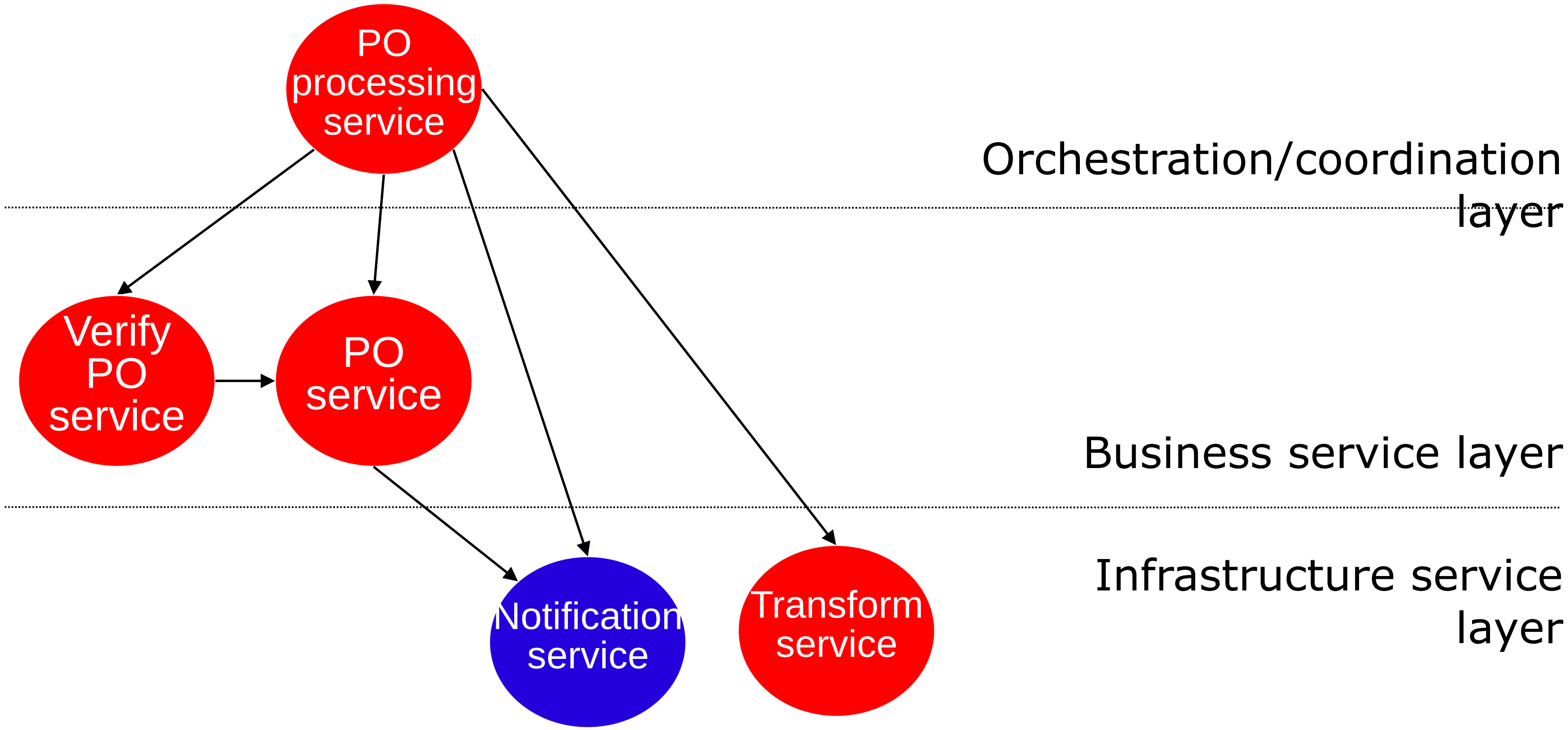

Service oriented analysis
Service oriented design
…
Compose SOA
Design entity-centric business services
Design infrastructure services
Design task-centric business services
Design SO business process

Entity-centric business services

- Goal: entity-centric business service layer + parent orchestration layer

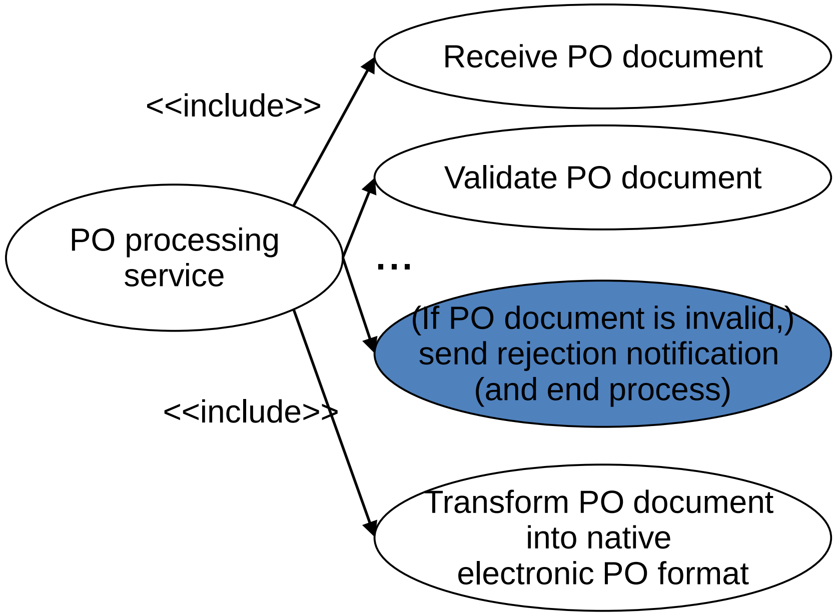


Infrastructure services



Task-centric business services

- UML sequence diagram
  - express and refine order of invocations implicit in the UML use case diagram



PO processing service

<<include>>

Receive PO document

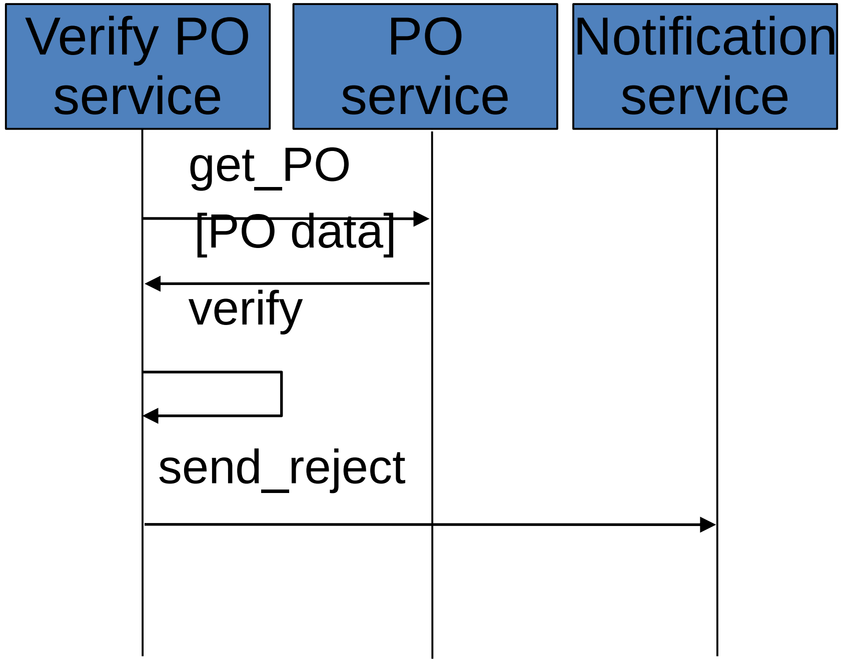
Validate PO document

...

(If PO document is invalid,) send rejection notification (and end process)

<<include>>

Transform PO document into native electronic PO format



Verify PO service

PO service

Notification service

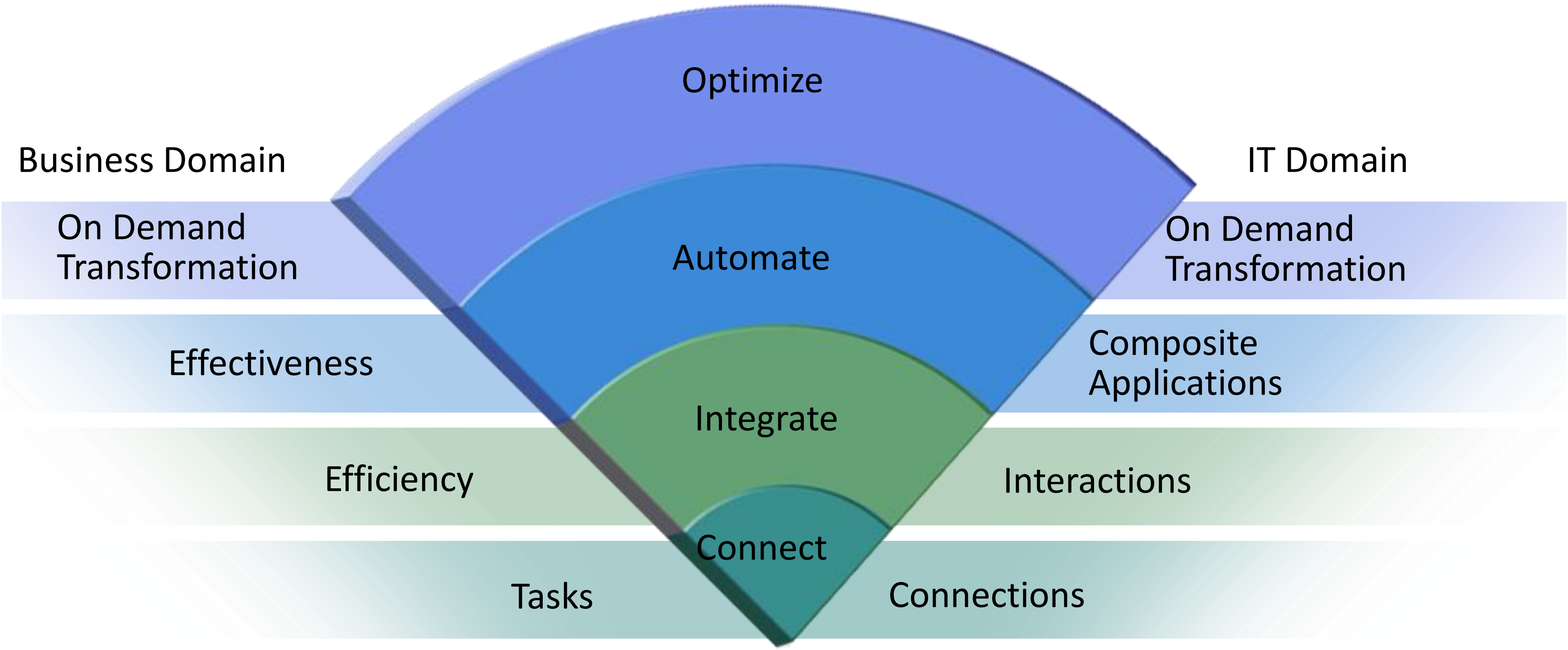
get_PO

[PO data]

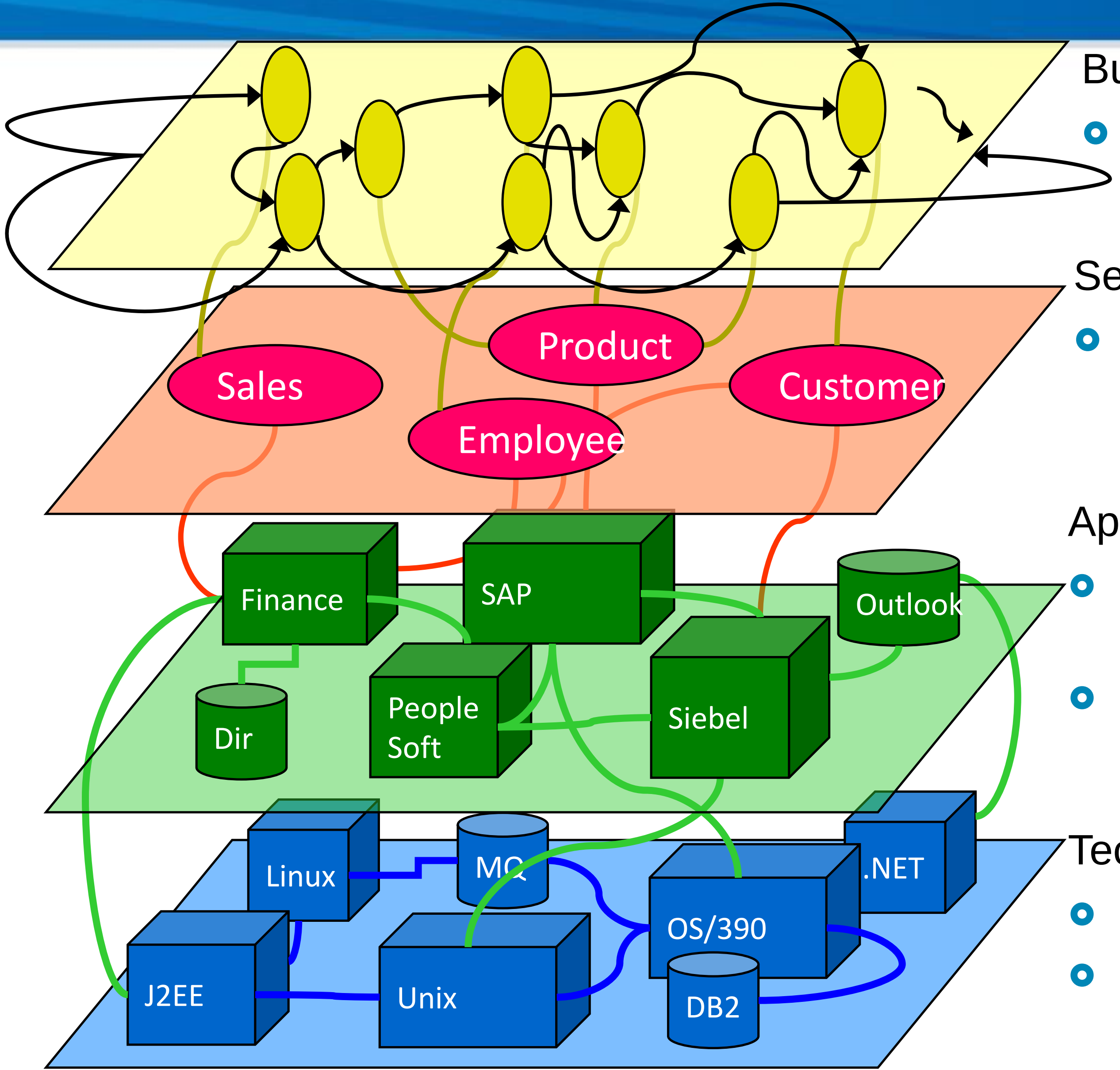
verify

send_reject

Getting To SOA



Moving to Services-Oriented Solutions



Business Process Layer

- *Cross Functional End-to-end Sales Order Process*

Service Layer

- *How do you connect sales to customers?*

Application Layer

- Applications, Components, Software
- *How do you connect SAP to Siebel?*

Technology Layer

- Hardware, Network
- *How do you connect J2EE to .NET?*

Source: CBDi Forum, http://www.cbdiforum.com

SOA in Practice

Business Process

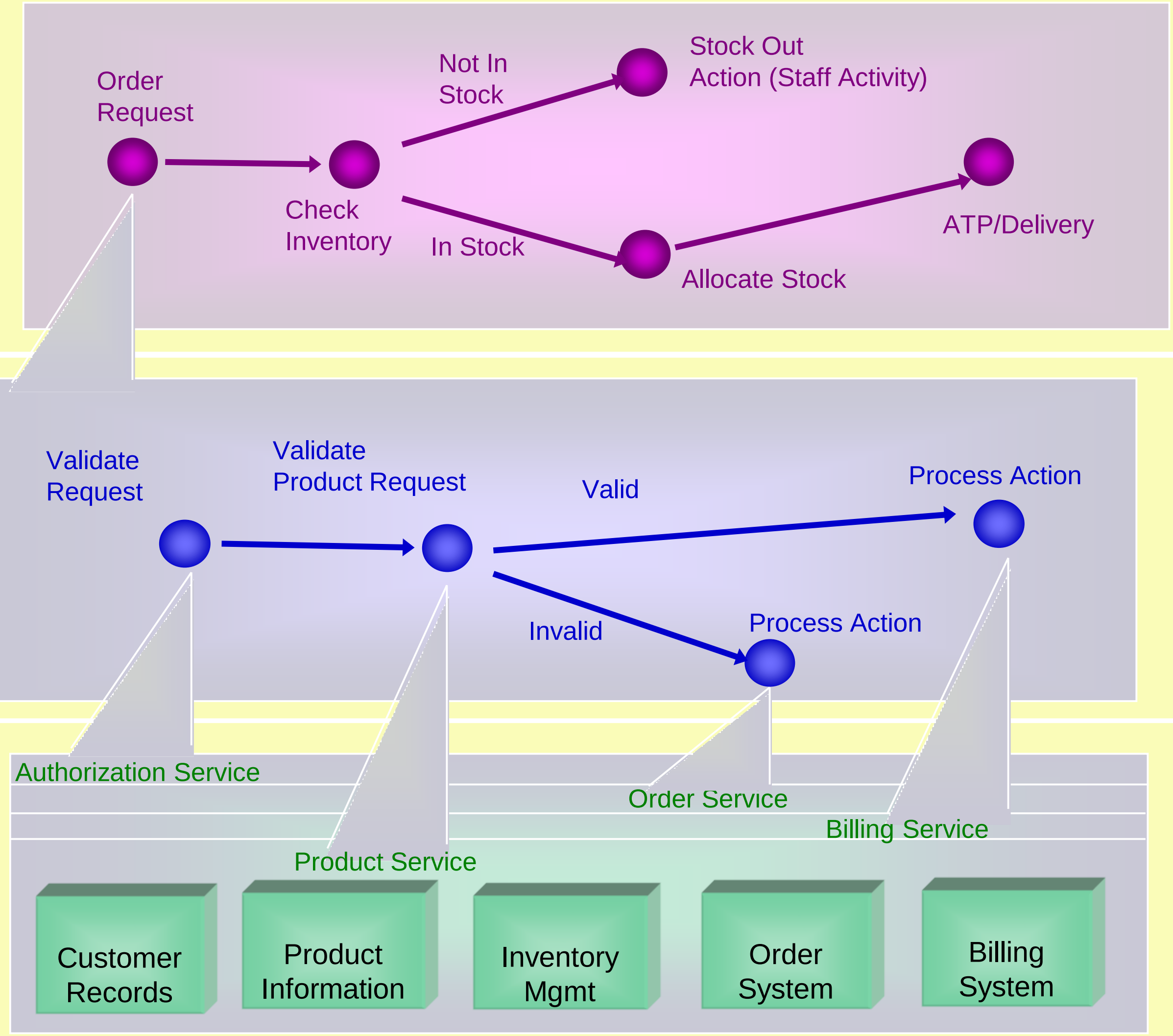
- may be long running
- multiple valid process states
- alternative workflows for non-normal conds and/or compensation for exception management

Business Transaction

- short term, non-interactive
- one change of business state or STP
- consumes one or more function service
- targeted level of service reuse
- loose coupling very important
- may require compensating transactions

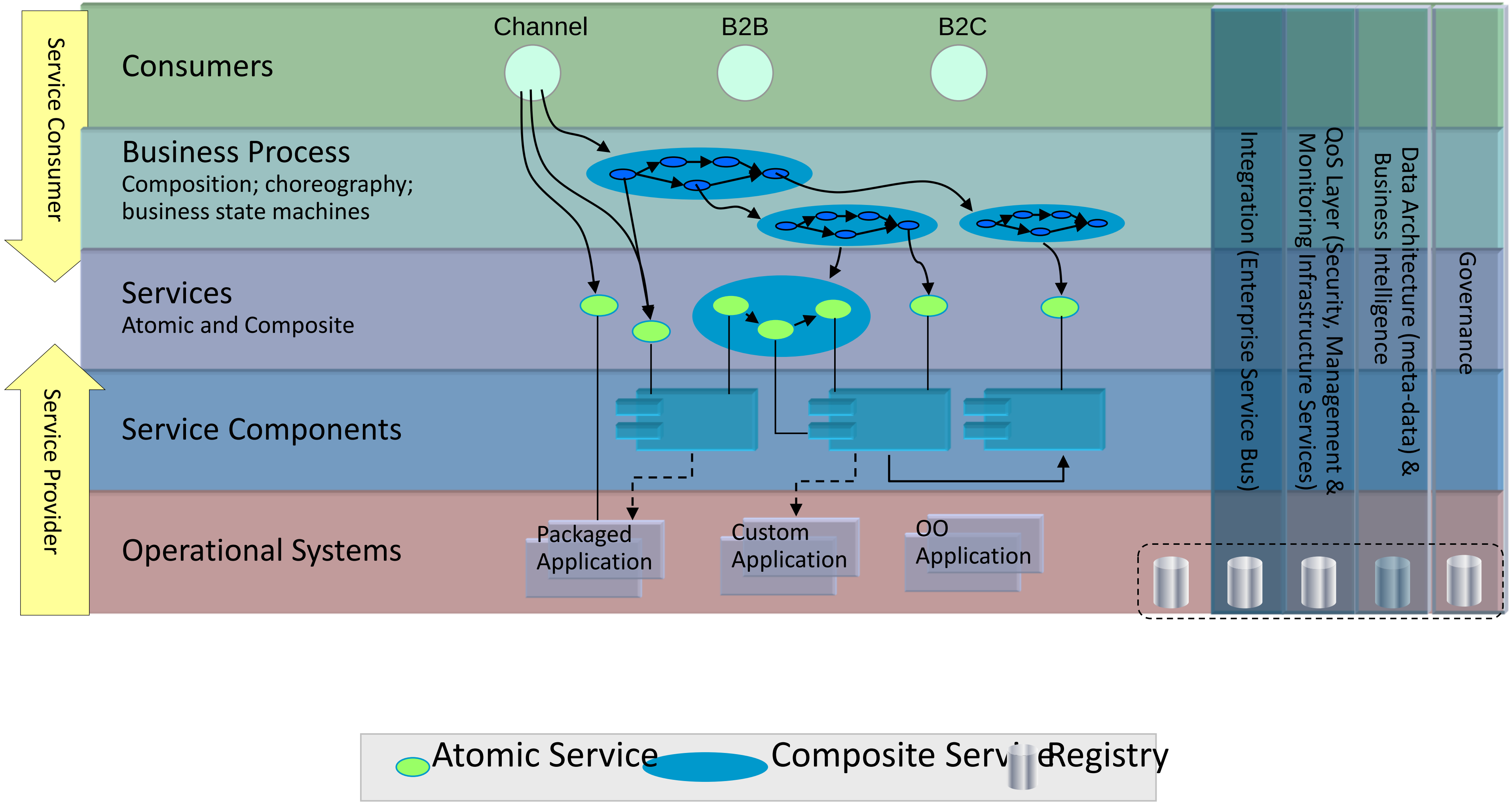
Function Service

- collaborations to implement a single FS
- collaborating apps encapsulated via FS(s)

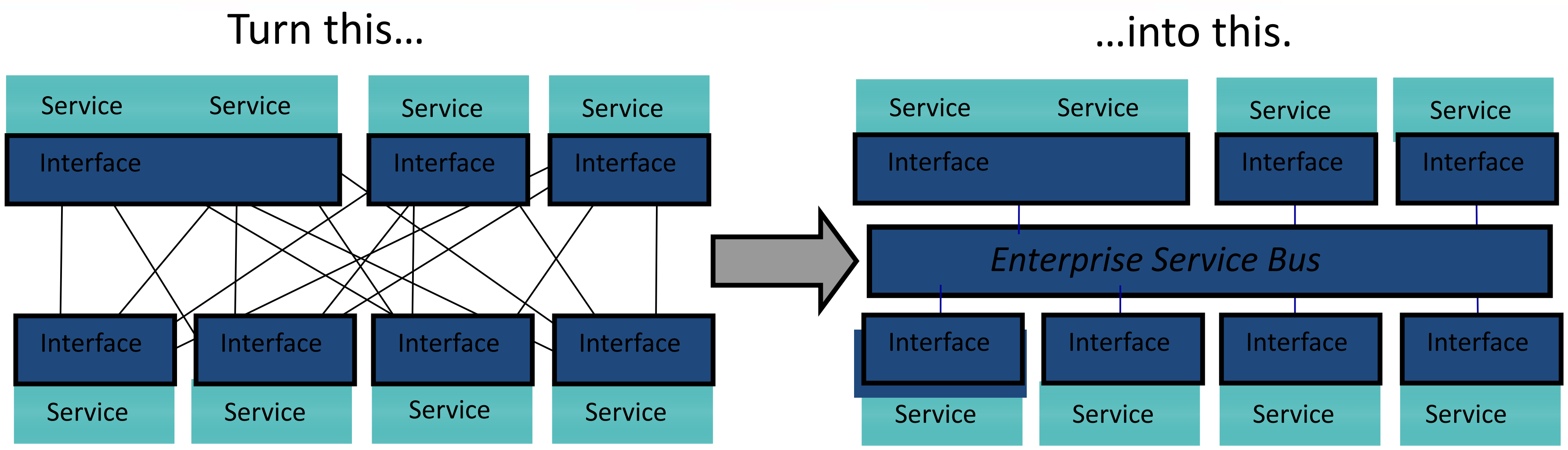


SOA Solution Abstraction Layering

. . . Leveraging the SOA Reference Architecture



Loose Coupling is enabled by an "ESB"



- ✓ Decouples the point-to-point connections from the interfaces
- ✓ Allows for dynamic selection, substitution, and matching
- ✓ Enables more flexible coupling and decoupling of the applications
- ✓ Enables you to find both the applications and the interfaces for re-use

RESULT → Greater Business Responsiveness

Section

中国科学技术大学
University of Science and Technology of China

SaaS

Software as a Service

SaaS

SaaS Idea

An application is made available via a service interface. The application is hosted by a service provider via the internet (commonly referred to as the Cloud).

If the service is hosted within an organisation (instead of the cloud), it is called “on premises software” .

SaaS

- **Properties of SaaS**
- Advantages: One has not worry about application deployment, maintainance, security
- Disadvantages: Loose control over the application, the service provider may conduct unwanted changes
- The billing is often done on per request basis and low entry prices

Examples: Salesforce.com, Sensium.io