Service Oriented Architcture

CS 752 Software Architecture and Design Practices

Prof. Chandrashekar R

Ref: Service-Oriented Architecture: Concepts, Technology, and Design By Thomas Erl

Outline

1. Introduction
2. Architecture Principles
3. Sub-Architectures (“under the hood”)
4. Implementation Approaches
5. Example

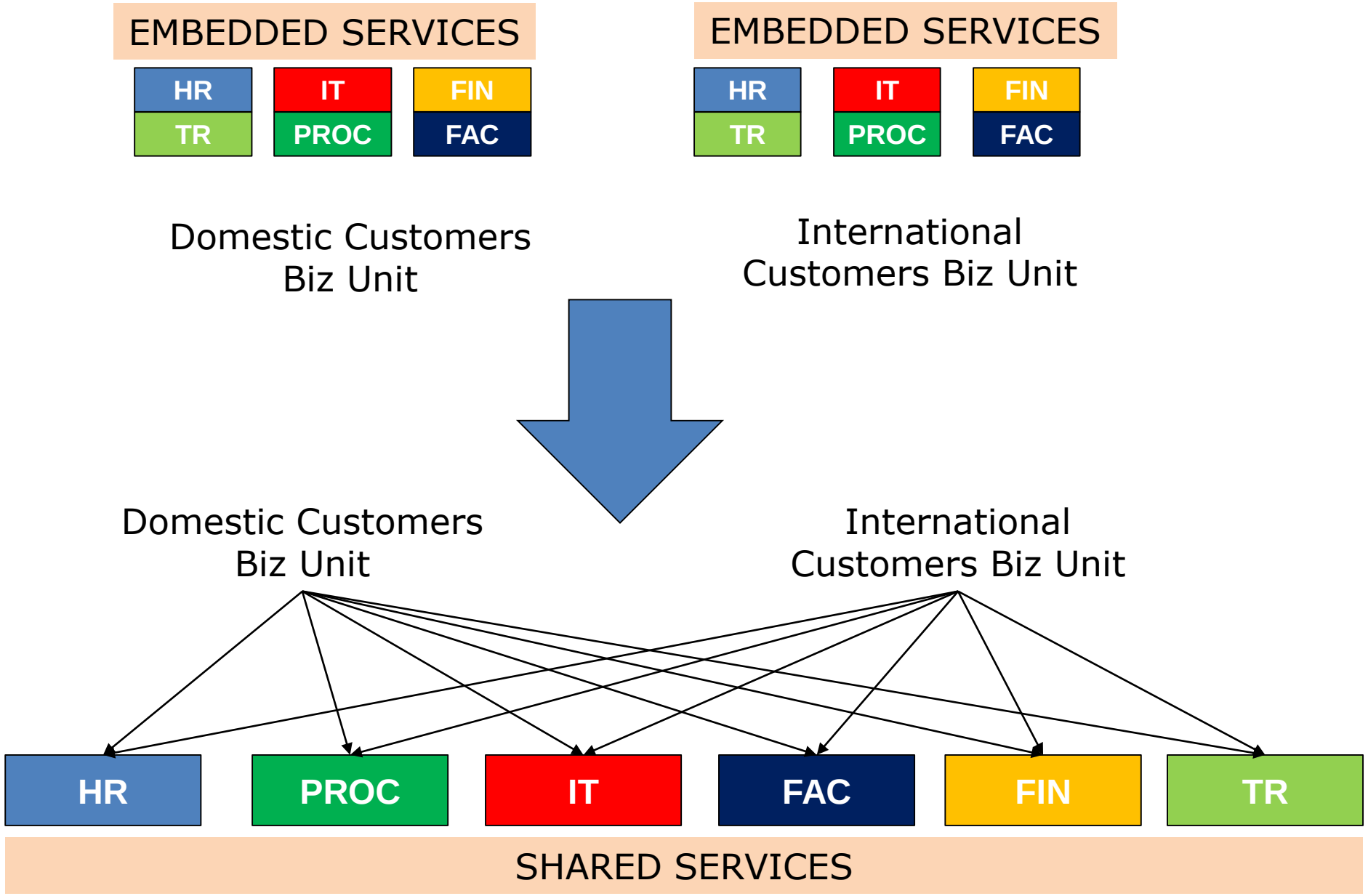
1. Introduction
2. Architecture Principles
3. Sub-Architectures ("under the hood")
4. Example

1. INTRODUCTION

Business Process Concepts

Business Process	•A formally defined method for achieving a specific goal
Task	•Steps involved in a specific business process
Process Input	•Information provided when process is triggered
Process Output	•Information generated after a process is completed
Manual Process	•Process in which all tasks are exclusively done by humans
Automated Process	•Process that is made up of tasks that are fully automated
Hybrid Process	•Process that where some tasks are manual and some are automated

SOA Motivation – Shared Services Model



Comparison

Embedded Services

- Services are provisioned within the divisions
- Services are optimized for the specific needs of the division
- Possibility of under-utilization and over-provisioning of resources
- High cost, high efficiency

Analogy: How will it be if each **PROCESS** in an operating system gets its own physical CPU, physical RAM, physical DISK?

Shared Services

- Multiple divisions share the same services
- Services are optimized for least common denominator
- Resources provisioned and utilized optimally across all the divisions
- Reduced cost, reduced efficiency (if not managed properly)

Analogy: Multiple processes share the services provided by a small group of CPU, RAM and DISK resources

Service-oriented Architecture (SOA)

- SOA is an architecture style for enterprise IT systems
- Inspired by the “shared-services” model
- At a conceptual level, **WHAT** needs to be done in a business process remains the same
- At the logical and physical level, **HOW** it needs to be done will vary in SOA

SOA Characteristics

- Functionality divided into **composable services** (and not single use “local” functions)
- Loose coupling between service provider and service consumer
- Serves as building blocks for more complex services
- Make business processes more dynamic

Task vs Service?

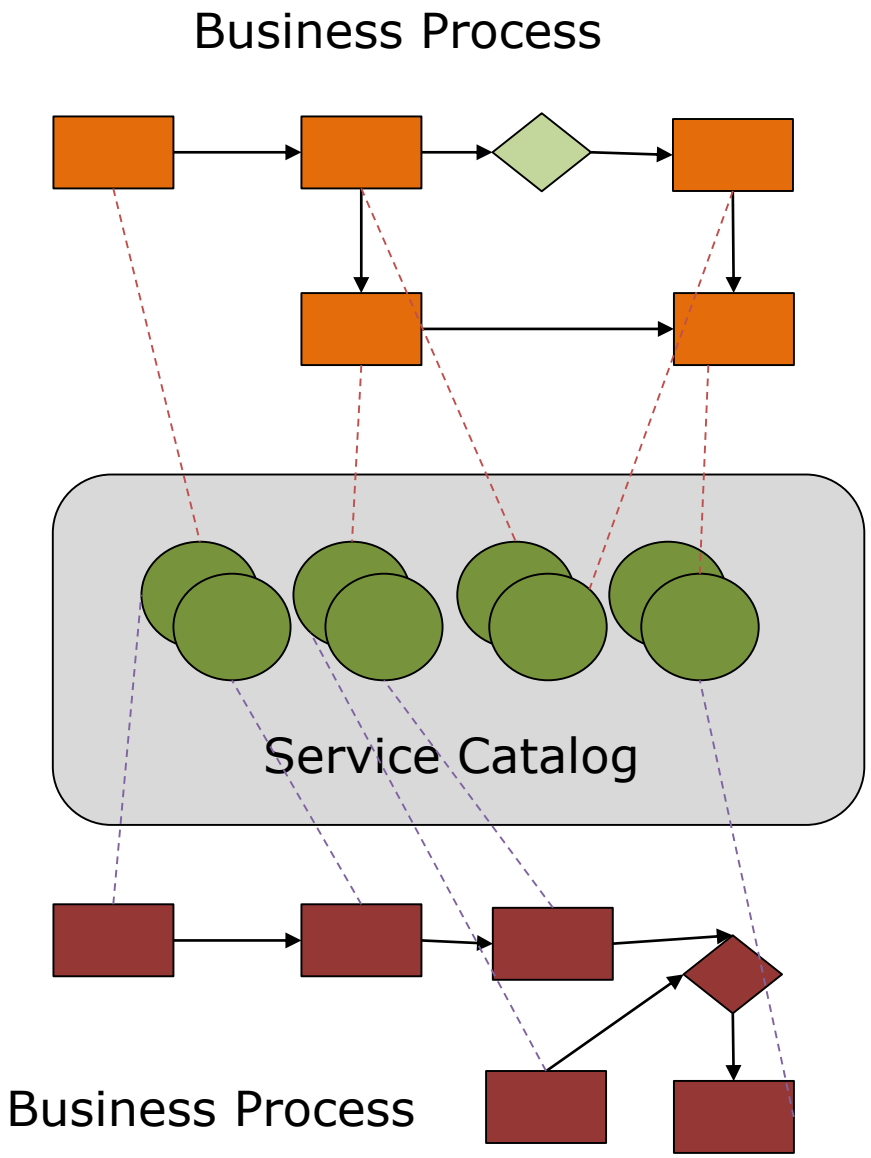
Service Concepts

Service Capability
- Every service is capable of carrying out certain **FUNCTION**

Service Contract
- Capability is formally published in the form of a **CONTRACT**

Service Discovery
- Service consumers should be able find out about both services available from a **CATALOG** and their capabilities via contracts

Service Invocation
- Compose a set of capabilities to carry out **TASKS**



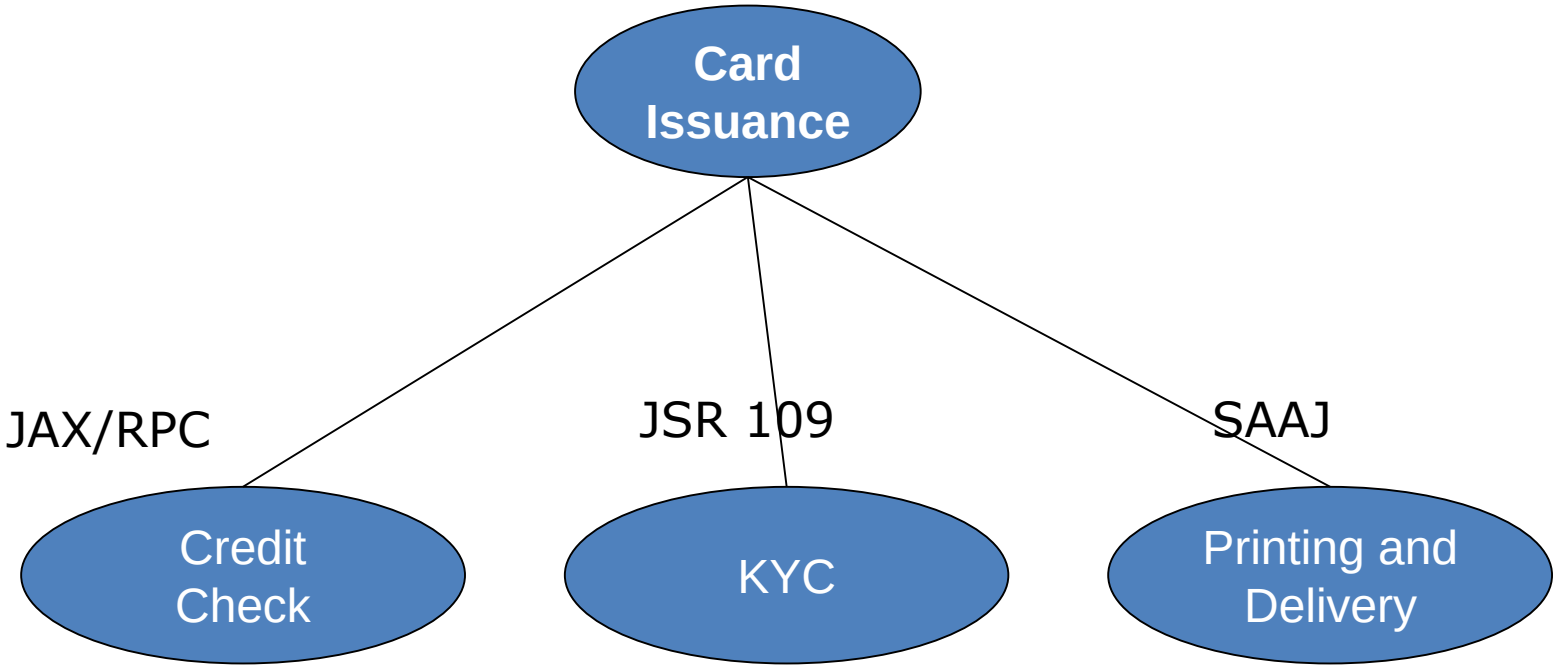
Key Elements of SOA

- Open Standards
- Integration points
- Virtualization (location, platform, service provider)
- Automation (business process vs workflow)
- Orchestration
- Choreography

Open Standards

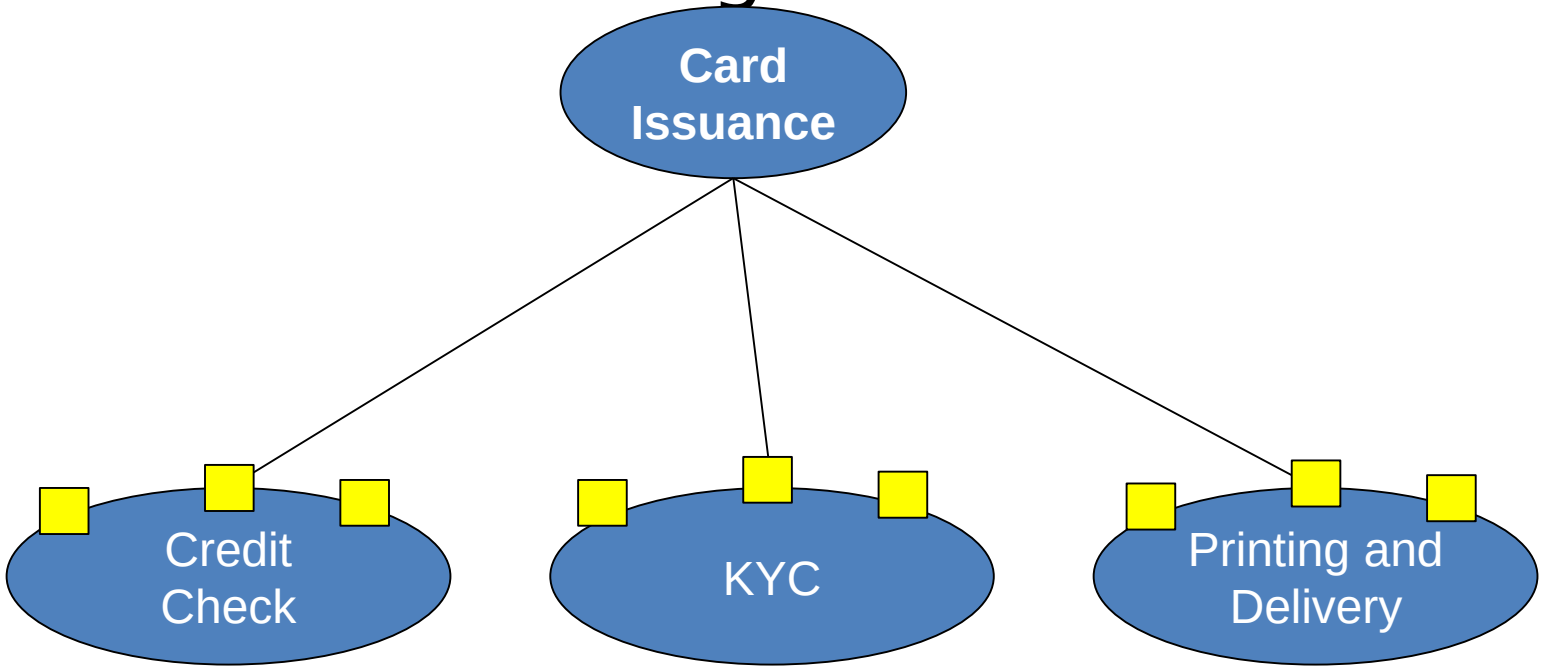
- Standards are needed
- Standards need to be Open
- Standards need to be used!

What does “closed” mean?
Duh, it is opposite of “open”!



Integration Points

- Entry points needed for integration
- Possibility of multiple entry points for the same service
- Platform-neutral integration



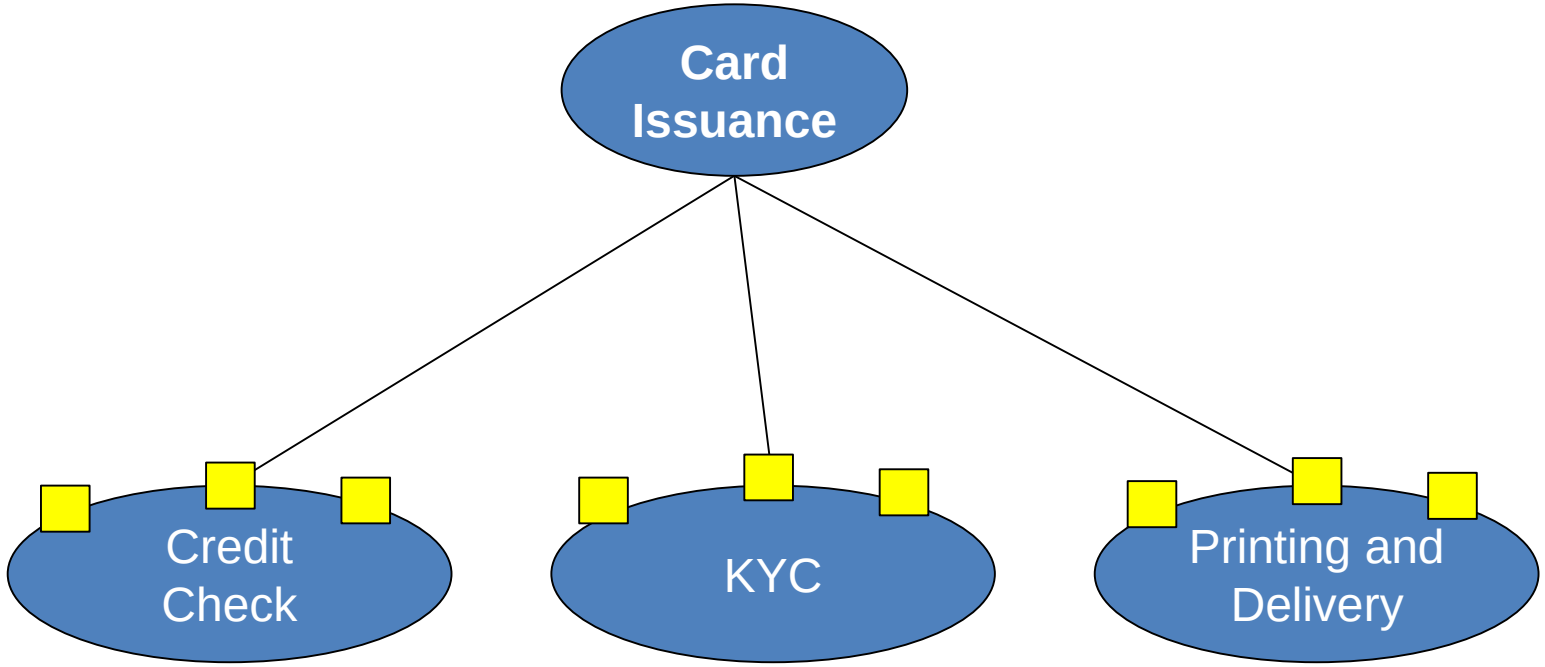
Virtualization

- Location
  - Service consumer not worried about location from where service is being delivered

Just like call center!

- Platform
  - Technology platform of consumer and provider need not be same
- Service Provider
  - Service consumer not worried which specific “worker” is providing the service

Virtualized services



Running in cloud

Implemented in .Net

Service provided multiple vendors

Automation

Key Elements of SOA

- Open Standards
- Integration points
- Virtualization
- Automation
- Orchestration
- Choreography

- From an architecture perspective, services can be delivered manually or through automation
- SOA's abstractions allow seamless conversion of manual services to automated services

E.g., Document Verification

Automation

Key Elements of SOA

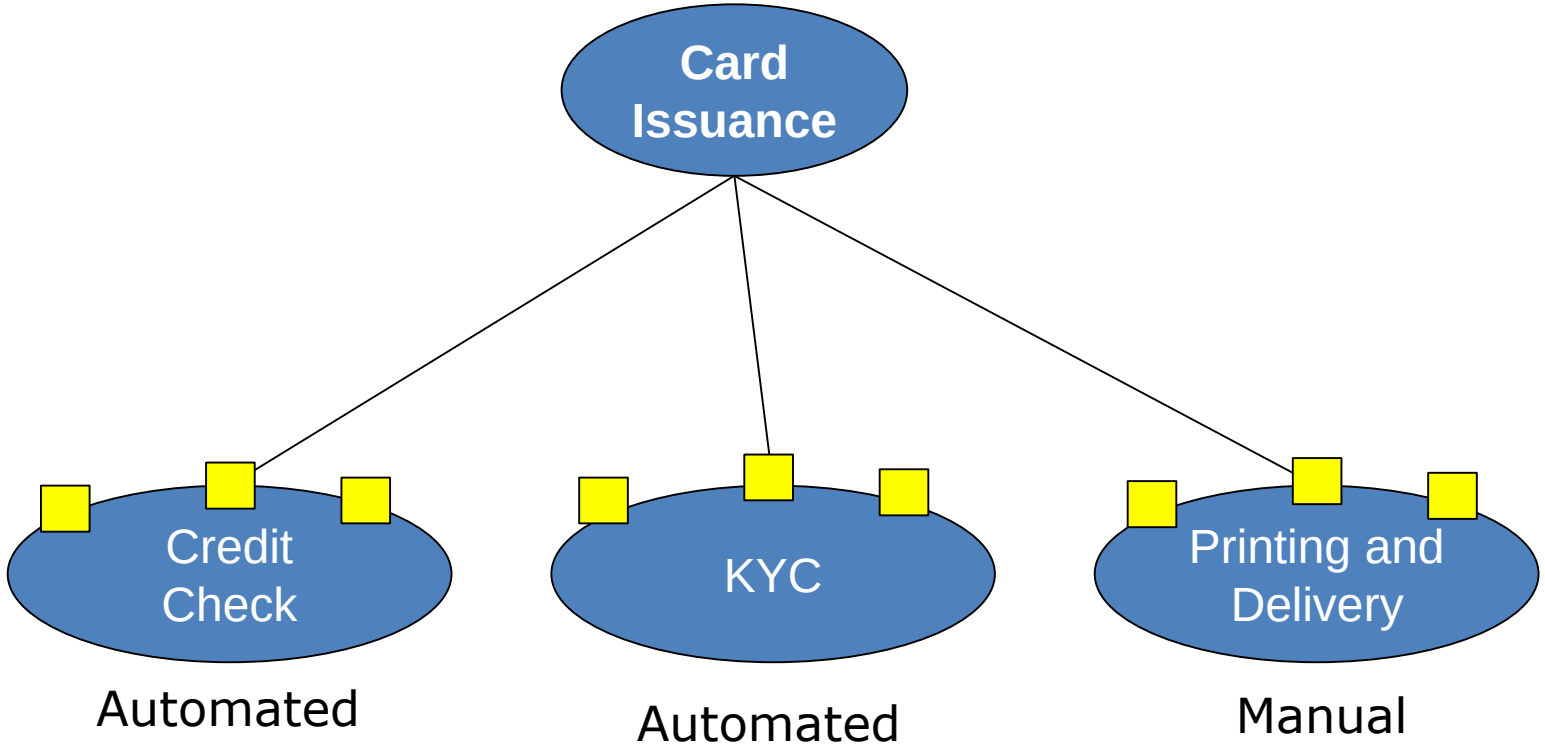
- Open Standards
- Integration points
- Virtualization
- Automation
- Orchestration
- Choreography

- From an architecture perspective, services can be delivered manually or through automation
- SOA's abstractions allow seamless conversion of manual services to automated services

E.g., Document Verification

- The processes can span within the organization ("workflow automation") or across organizations ("business process automation")

Hybrid Service



E.g., Hybrid process vs Hybrid service

Service Execution Models

Key Elements of SOA

- Open Standards
- Integration points
- Virtualization
- Automation
- Orchestration
- Choreography

Orchestration

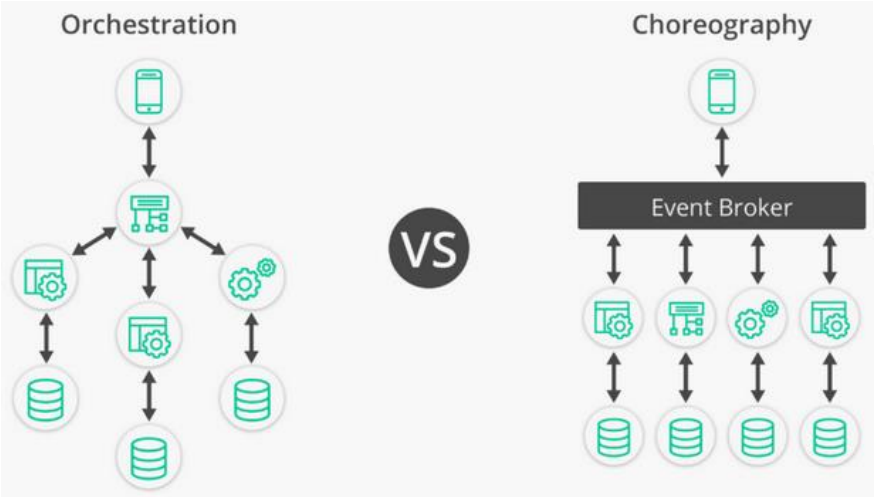
- The "controller" for each business process **invokes** services as part of the implementing process

E.g., Analogous to a conductor conducting an orchestra… no musician plays unless asked to be by the conductor!

Choreography

- The "controller" for each business process **announces** tasks that need to be done as part of the implementing process. Services pick up the tasks and do what is needed

E.g., Analogous to dancers doing their own moves once music starts playing!



Ref: Microservices Choreography vs Orchestration

Flavors of SOA

SOA for Enterprise Architecture

- Focuses on Business
- Business Processes Management
- Business Process Execution

SOA for IT Systems Architecture

- Focuses on IT needs of the enterprise
- IT Services are modeled as shared services

SOA for Technical Architecture

- Software architecture for implementing services
- Hardware infrastructure

Definition of a “Service”

- Services encapsulate a reusable business function
- Services are defined by explicit, implementation−independent interfaces
- Services are invoked through communication protocols that stress location transparency and interoperability

Service Granularity

Technical Function Services	• auditEvent, checkUserPassword, and checkUserAuthorization
Business Function Services	• calculateDollarValueFromYen and getStockPrice
Business Transaction Services	• checkOrderAvailability and createBillingRecord
Business Process Services	• openAccount, createStockOrder, reconcileAccount, and renewPolicy

1. Introduction
2. Architecture Principles
3. Sub-Architectures (“under the hood”)
4. Example

ARCHITECTURE PRINCIPLES

Ref: Appendix A. Service Orientation Principles Reference

Architecture Principles

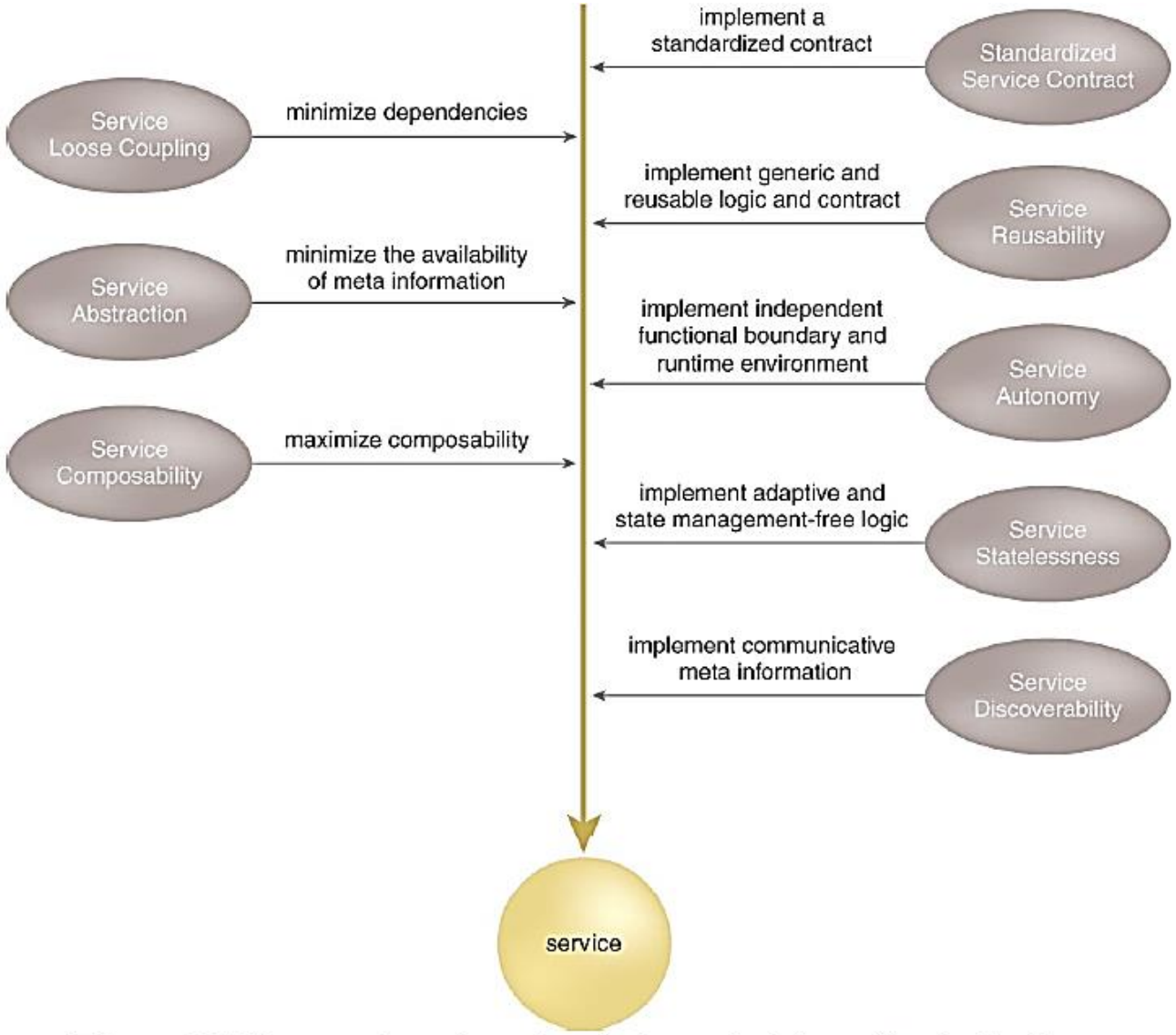
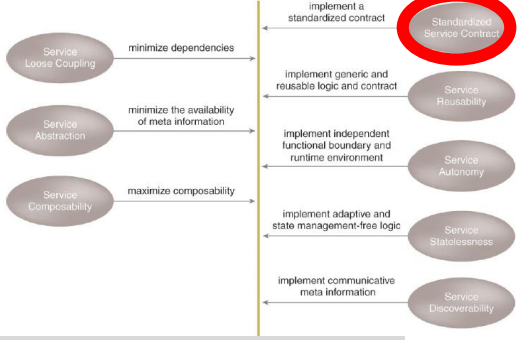


Figure 3.9 How service-orientation design principles collectively shape

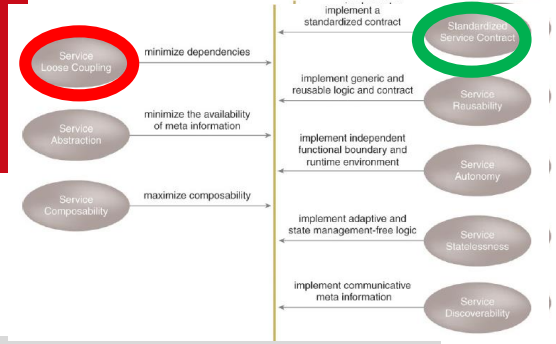
Standardized Service Contract



Services within the same service inventory are in compliance with the same contract design standards.

- Services publish and perform their capabilities in a standardized way

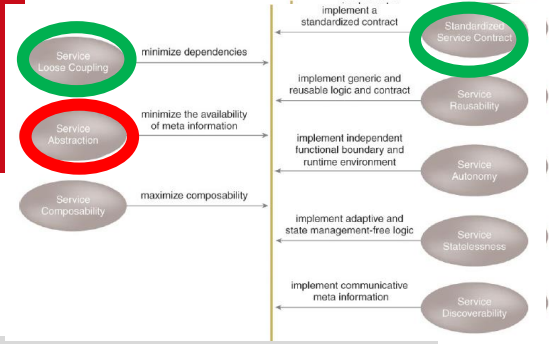
Loose Coupling



Service contracts impose low consumer coupling requirements and are themselves decoupled from their surrounding environment.

- Loose coupling between contract, implementation and consumption
- Enables independent evolution of design, implementation innovations without large scale changes

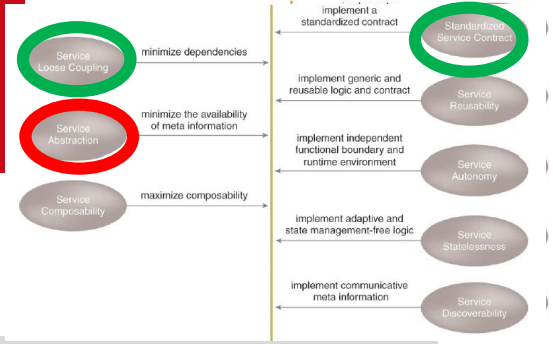
Service Abstraction



Service contracts only contain essential information and information about services is limited to what is published in service contracts

- Hide underlying details
- Necessary to enable loose-coupling

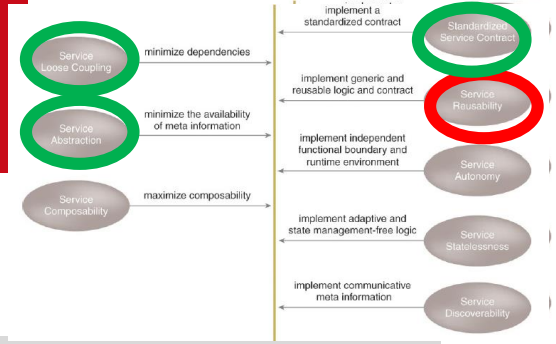
Service Abstraction



Service contracts only contain essential information and information about services is limited to what is published in service contracts

- Hide underlying details
- Necessary to enable loose-coupling

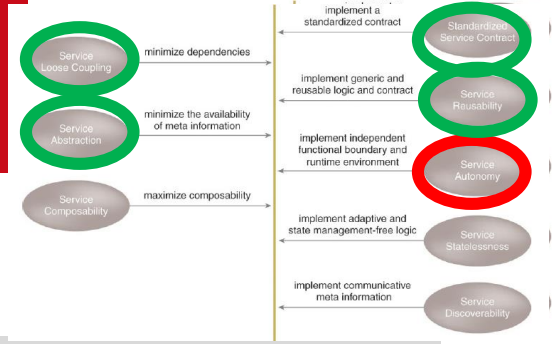
Service Reusability



*Services contain and express **agnostic** logic and can be positioned as reusable enterprise resources*

- Reusability necessary to take advantage of "shared services" benefits
- Not hardcoding for single-purpose makes it **agnostic**

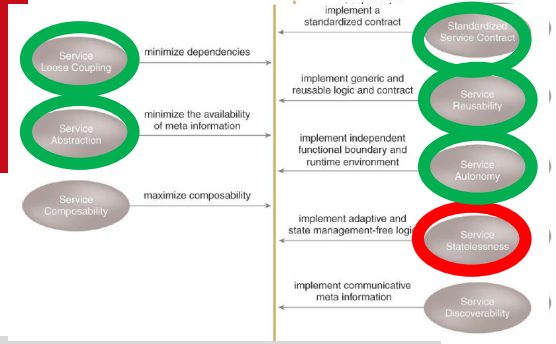
Service Autonomy



Services exercise a high level of control over their underlying runtime execution environment

- Significant degree of control over its environment and resources
- Enables evolution and improvements without impacting others

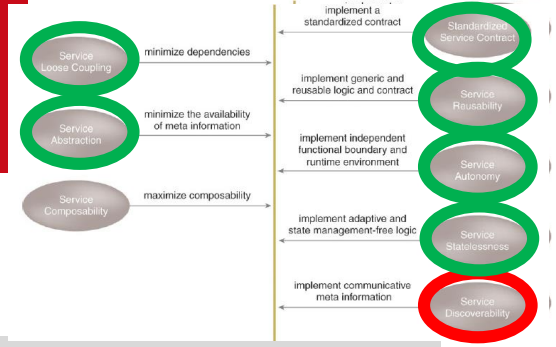
Service Statelessness



Services minimize resource consumption by deferring the management of state information when necessary

- Eliminates need of services to “remember” multiple execution threads
- Helps keep the service implementation “light-weight” – reduces memory requirements

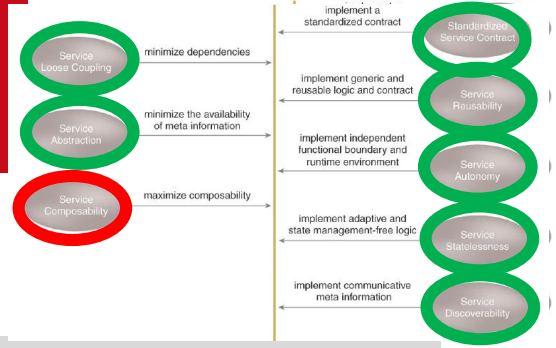
Service Discoverability



Services are supplemented with communicative metadata by which they can be effectively discovered and interpreted

- If services are not discoverable, there can be wasted effort in implementing multiple embedded services for same the same service
- Like a repair shop without customers - each one gets down to repairing their own equipment

Service Composability



*Services are effective composition **participants**, regardless of the size and complexity of the composition*

- Same service can be used in multiple varying contexts
- Many other principles like loose coupling, statelessness, etc. contribute towards this principle

GOING FORWARD

Looking ahead

1. Introduction
2. Architecture Principles
3. Sub-Architectures ("under the hood")
4. Example

1. Introduction
2. Architecture Principles
3. Sub-Architectures ("under the hood")
4. Example

SOA SUB-ARCHITECTURES

Sub-Architecture Types

- Service Architecture
- Service Composition Architecture
- Service Inventory Architecture
- Enterprise SOA

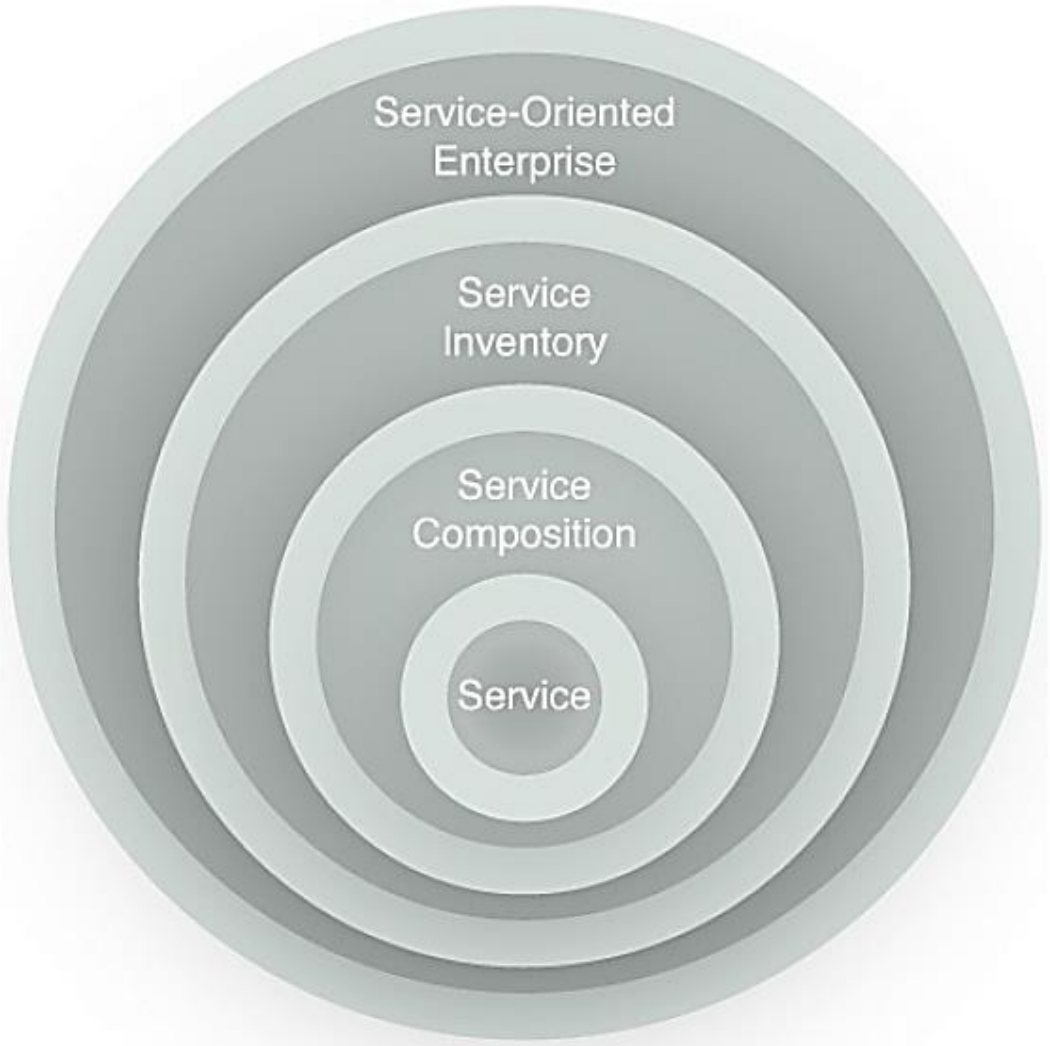


Figure 4.8 The layered SOA model establishes the four common SOA types:

Service Architecture

- Technology architecture of the actual service
- [Recall] Characteristics of a services (loosely coupled, composable, abstraction, etc.)
- Services can be grouped into inventories that follow similar architecture style like layered, event driven, etc.

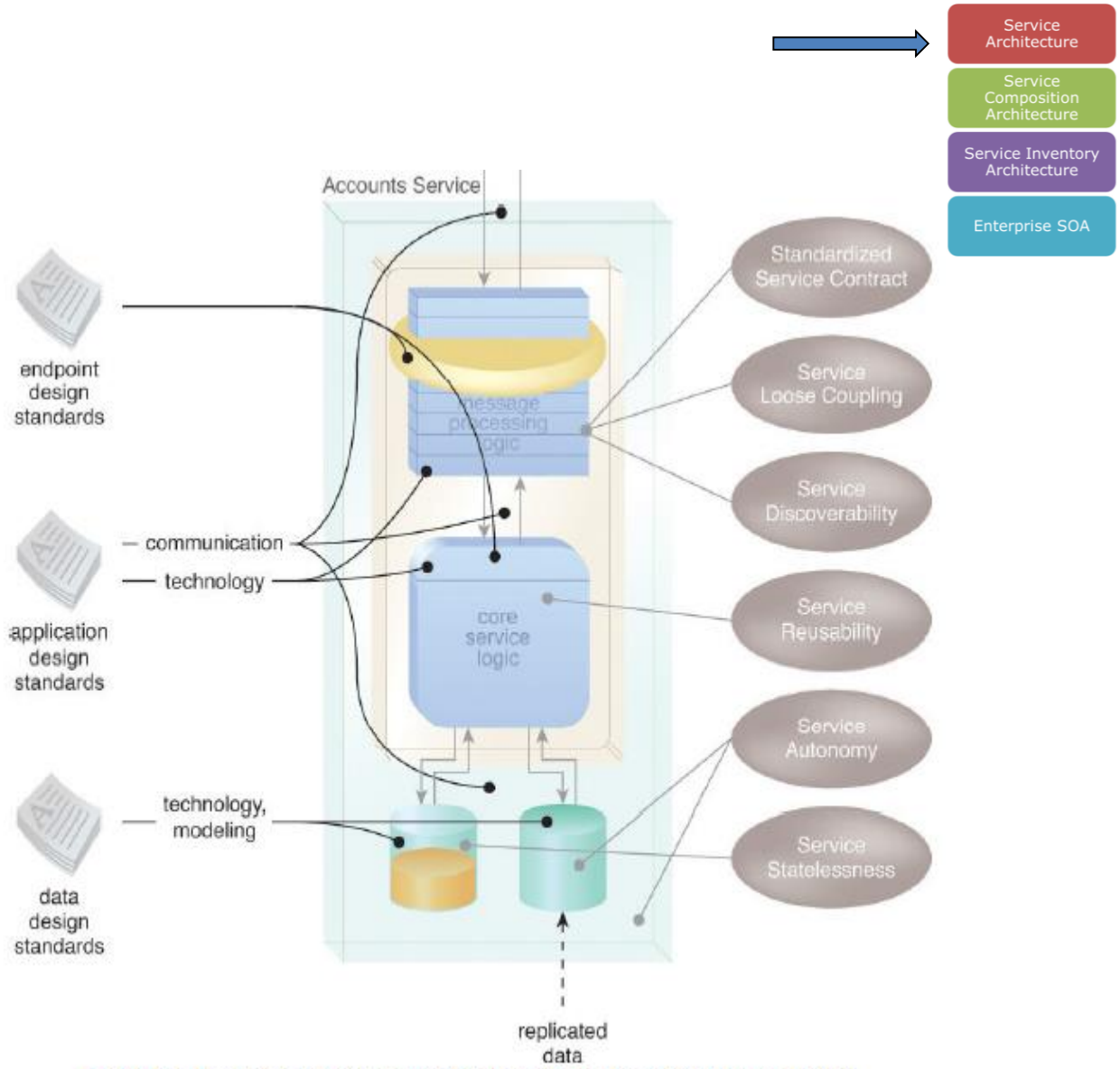
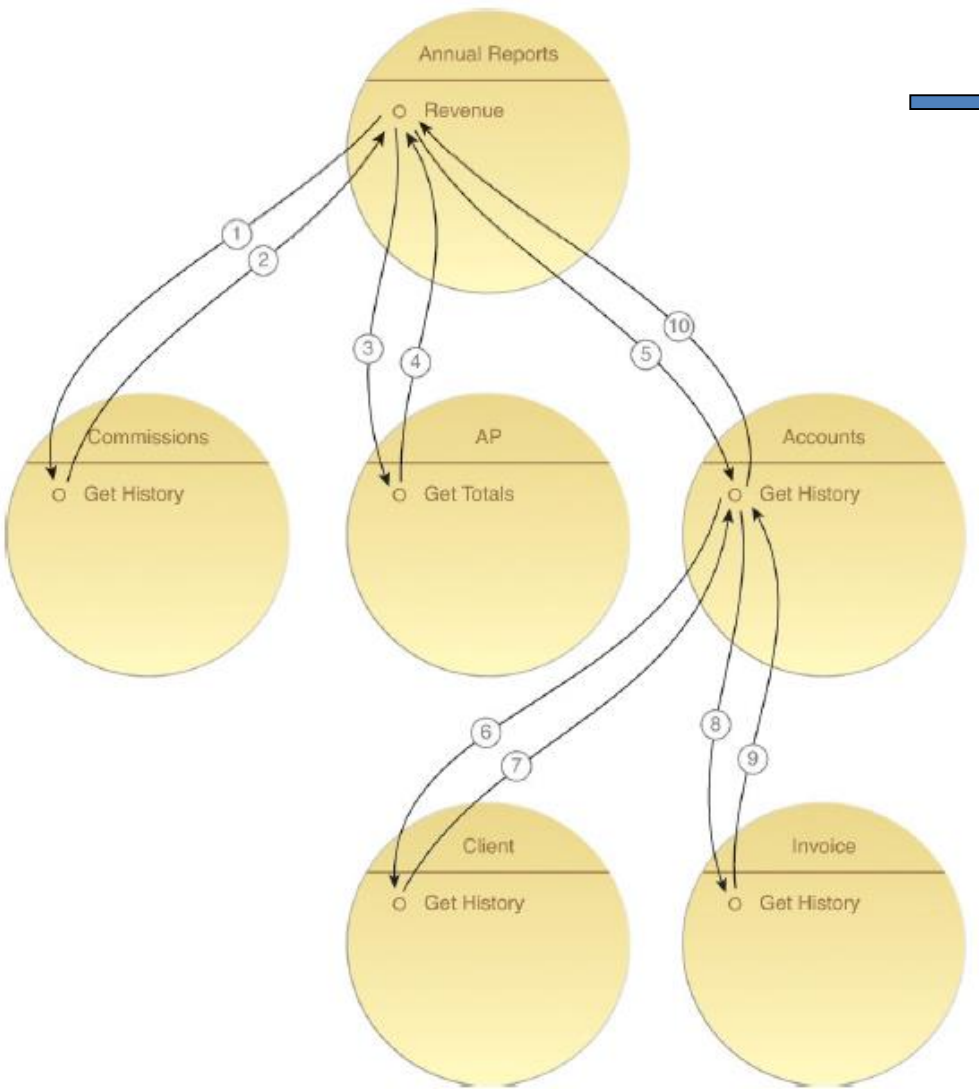
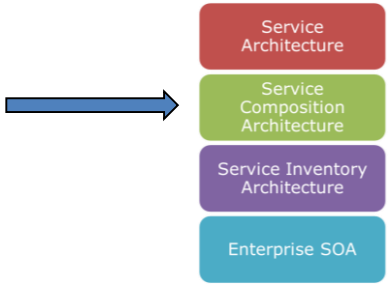


Figure 4.11 Custom design standards and service-orientation design

Service Composition Architecture

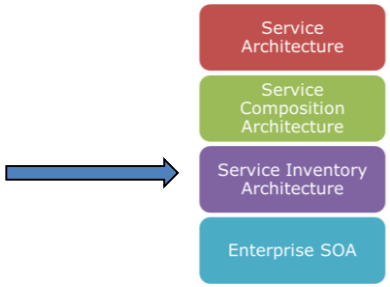
- Specifies architecture of **composite services** that use existing services
- Analogous to modular design that shows sub-modules of a module





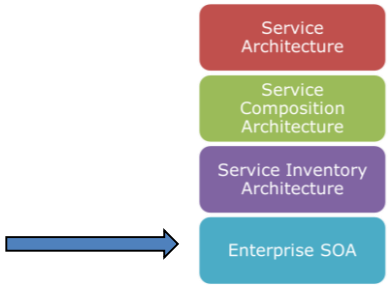
Service Inventory Architecture

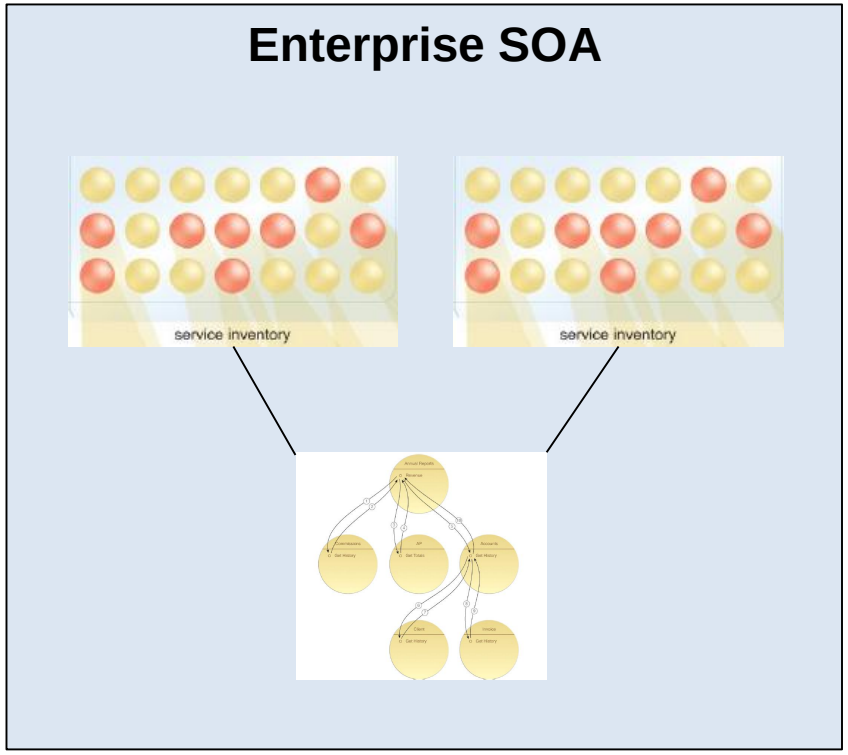
- How are the services organized made available for use?
- What are the rules governing addition, removal and retiring of services?
- What are the principles governing composition using services from the inventory?



Enterprise SOA

- A collection of service inventories on which entire enterprise relies on
- Rules governing design standards for inventories and services within inventories





IMPLEMENTATION APPROACHES

SOA using Web Services

- Web Service is an open-standards based protocol for implementing SOA over the web
- Web services provide an open–standard and machine–readable model for creating explicit, implementation–independent descriptions of service interfaces.
- Web services provide communication mechanisms that are location–transparent and interoperable.
- Key Standards
  - TCP/IP, XML, HTTP
  - WSDL, SOAP, UDDI

SOA using REST

- REST stands **RE**presentational **S**tate **T**ransfer
- Light weight alternative to SOAP-based Web Services
- Key behavior characteristics
  - Stateless, Idempotent or Non-Idempotent, RO/RW
- Key protocols
  - HTTP based methods such as GET, POST, PUT, DELETE
  - JAX-RS for Java implementations

SUMMARY OF SOA

Summary of SOA

- SOA is one of the most widely used technology architecture styles
- SOA is amenable to be implemented at multiple levels
- SOA is widely implemented using SOAP Web Services or REST micro-services