Middleware Architectures 1

Lecture 2: Service Oriented Architecture

doc. Ing. Tomáš Vitvar, Ph.D.
tomas@vitvar.com • @TomasVitvar • http://vitvar.com

Czech Technical University in Prague
Faculty of Information Technologies • Software and Web Engineering • http://vitvar.com/courses/mdw

Modified: Sun Oct 11 2020, 19:46:07
Humla v0.3

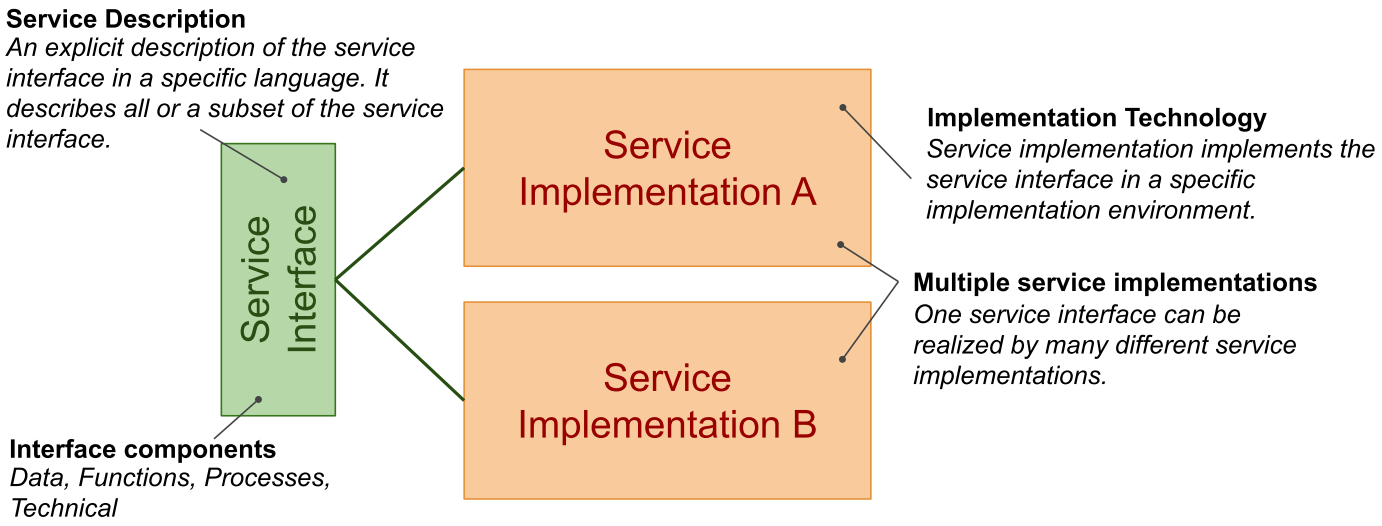
Overview

- Service Definition
- Integrating Applications
- Enterprise Service Bus
- Microservices Architecture

Service

- Difficult to agree on one definition
- Business definition
  - *A service realizes an effect that brings a business value to a service consumer*
    → *for example, to pay for and deliver a book*
- Conceptual definition
  - *service characteristics*
    → *encapsulation, reusability, loose coupling, contracting, abstraction, discoverability, composability*
- Logical definition
  - *service interface, description and implementation*
  - *message-oriented and resource-oriented*
- Architectural definition
  - *business service (also application service)*
    → *external, exposed functionality of an application*
  - *infrastructure service*
    → *internal/technical, supports processing of requests*

Interface, Description and Implementation



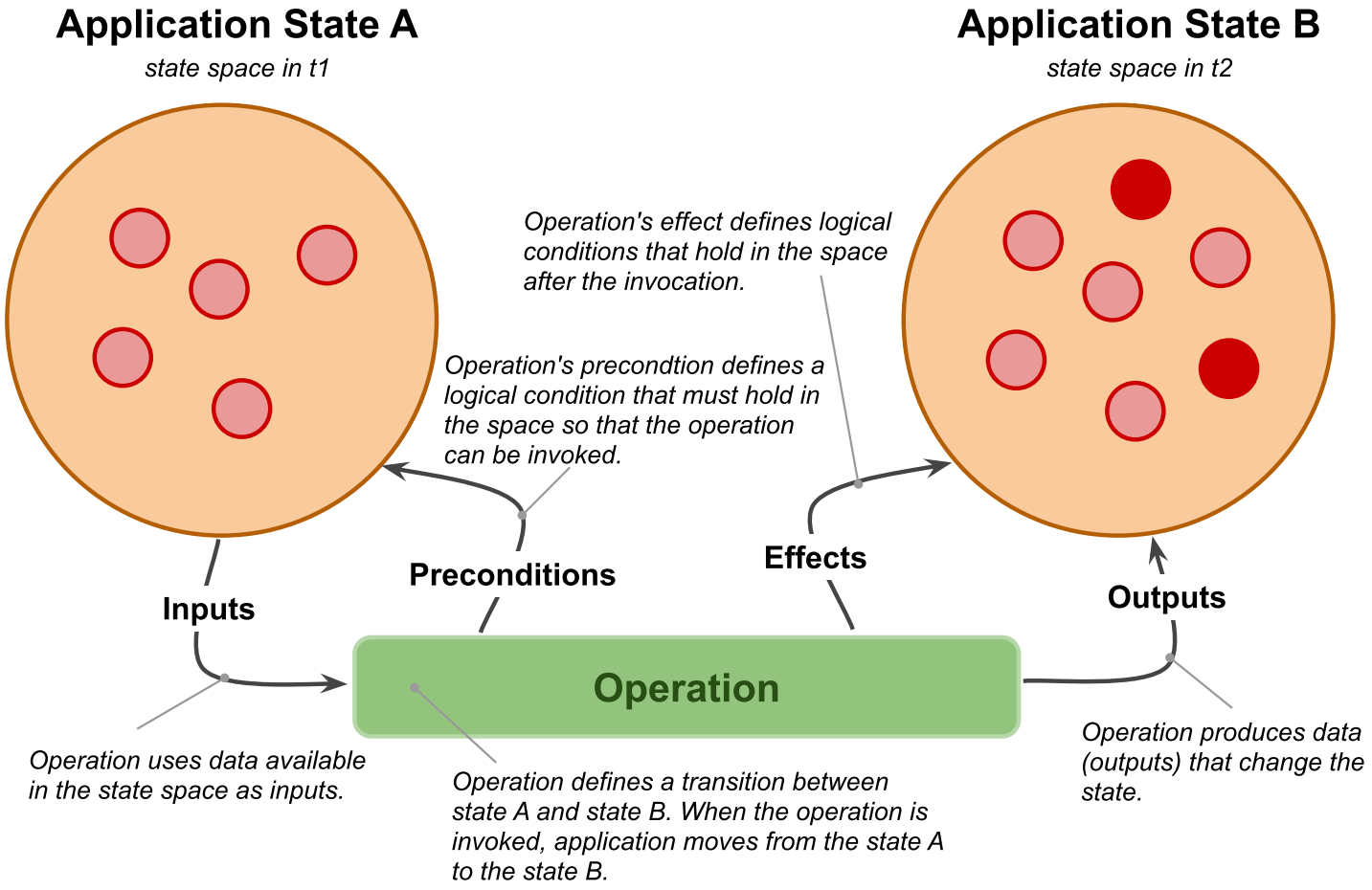
- Terminology clarification
  - *service ~ service interface + service implementation*
  - *WSDL service ~ service description in WSDL language*
  - *SOAP service ~ a service interface is possible to access through SOAP protocol; there is a WSDL description usually available too.*
  - *REST/RESTful service ~ service interface that conforms to REST architectural style and HTTP protocol*

Service Interface

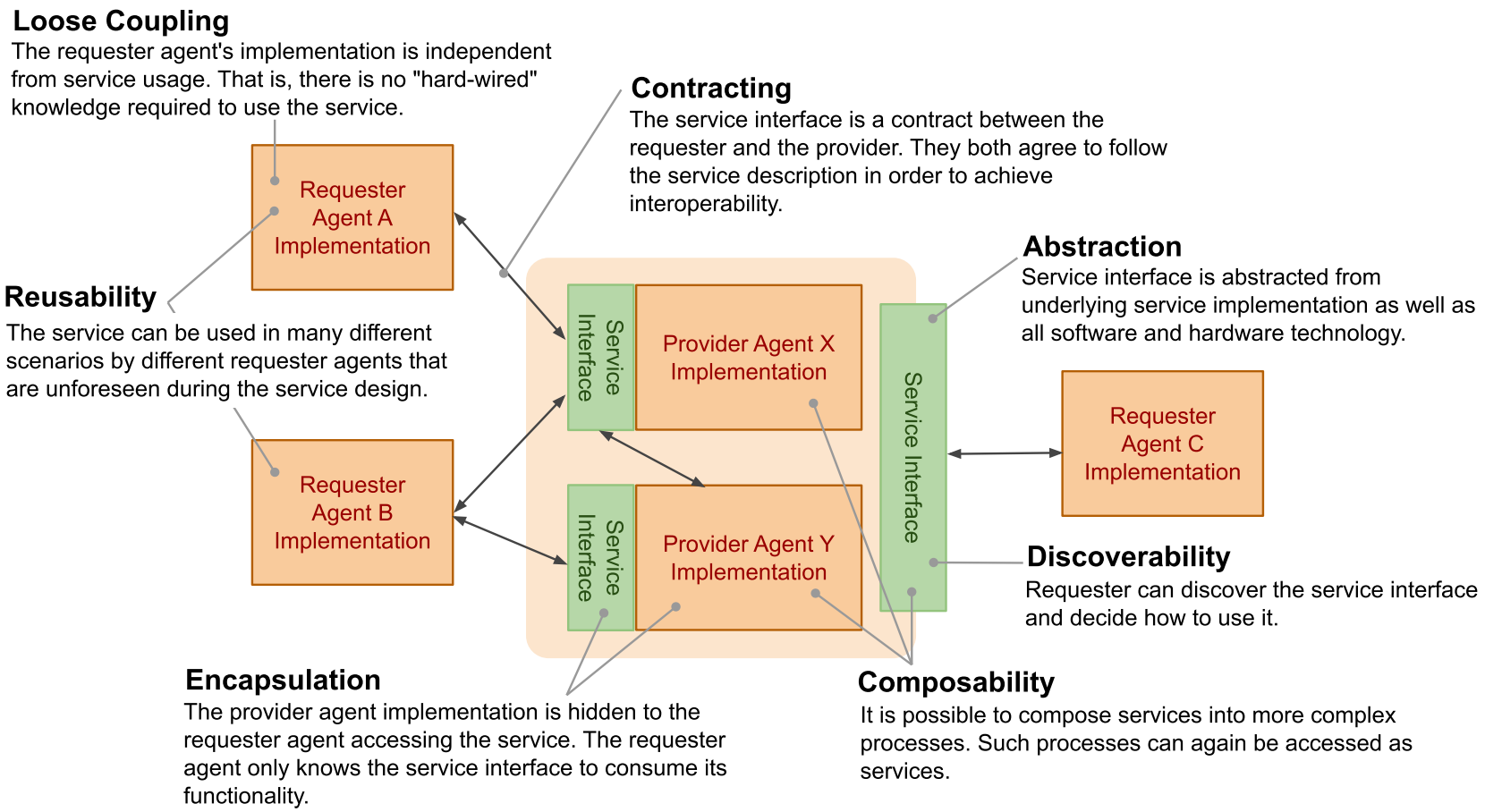
- Service interface components
  - *Data*
    - → *Data model definition used by the service*
    - → *for example, input and output messages, representation of resources*
  - *Functions*
    - → *operations and input and output data used by operations*
  - *Process*
    - → *public process: how to consume the service's functionality*
    - → *orchestration: realization of the service's functionality by its implementation*
  - *Technical*
    - → *security, usage aspects (SLA-Service Level Agreement)*
    - → *other technical details such as IP addresses, ports, protocols, etc.*

Public Process

- A state diagram
  - *operation of a service defines a **state transition** between two states.*



Service Characteristics



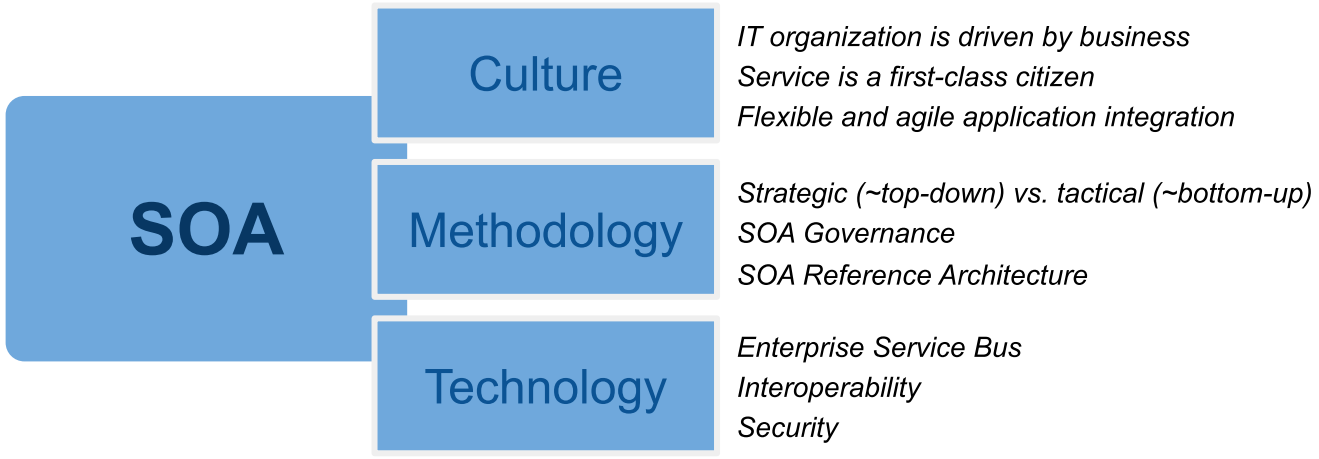
Overview

- Service Definition
- Integrating Applications
- Enterprise Service Bus
- Microservices Architecture

Integration and Interoperability

- Integration
  - *A process of connecting applications so that they can exchange and share capabilities, that is — information and functionalities.*
  - *Includes methodological approaches as well as technologies*
- Interoperability
  - *Ability of two or more applications to understand each other*
  - *Interoperability levels*
    - → *Data – syntax/structure and semantics*
    - → *Functions/Processes – syntax and semantics*
    - → *Technical aspects – protocols, network addresses, etc.*

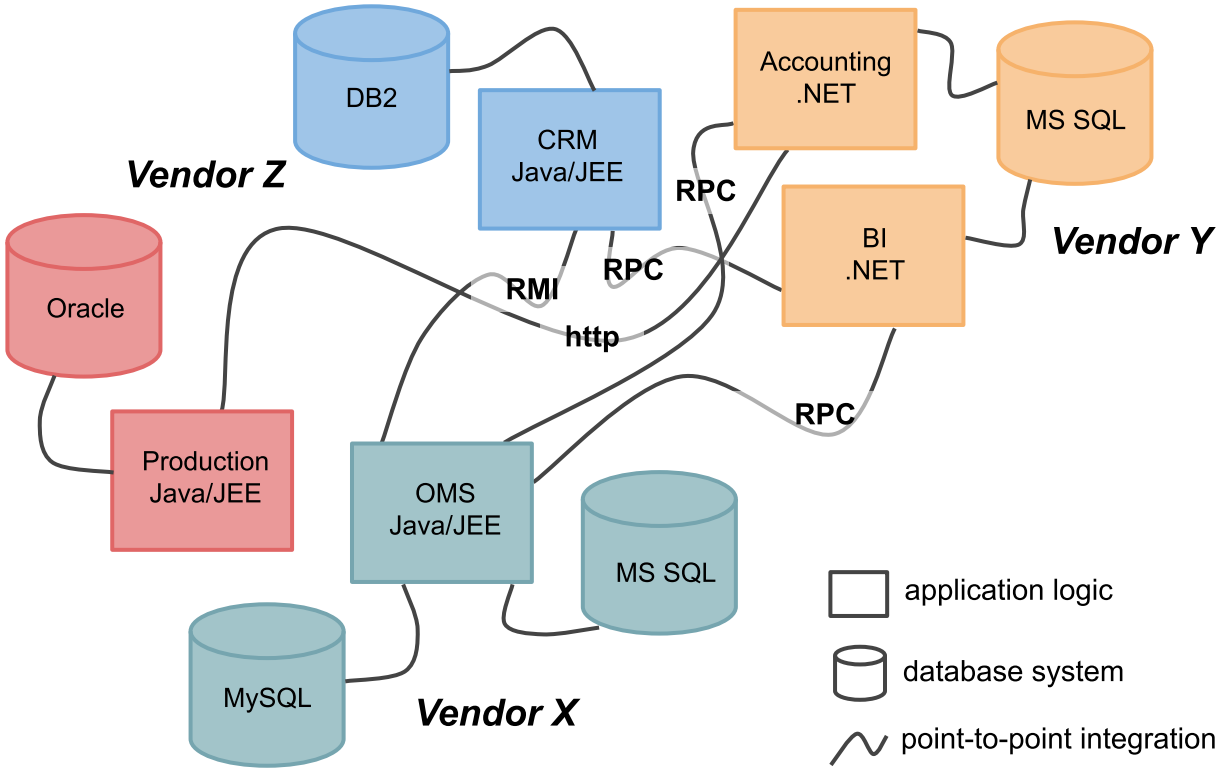
Service Oriented Architecture



- SOA supports two core business strategies
  - *Growing top-line revenue*
    - *→ Enterprise reacts quickly to requirements from the market*
    - *→ Business processes can be reconfigured rather than reimplemented*
  - *Improving bottom-line profit*
    - *→ Saving development costs by resuing existing services*
- Pre-integrated solutions
  - *Out-of-the-box applications and integration solutions among them*

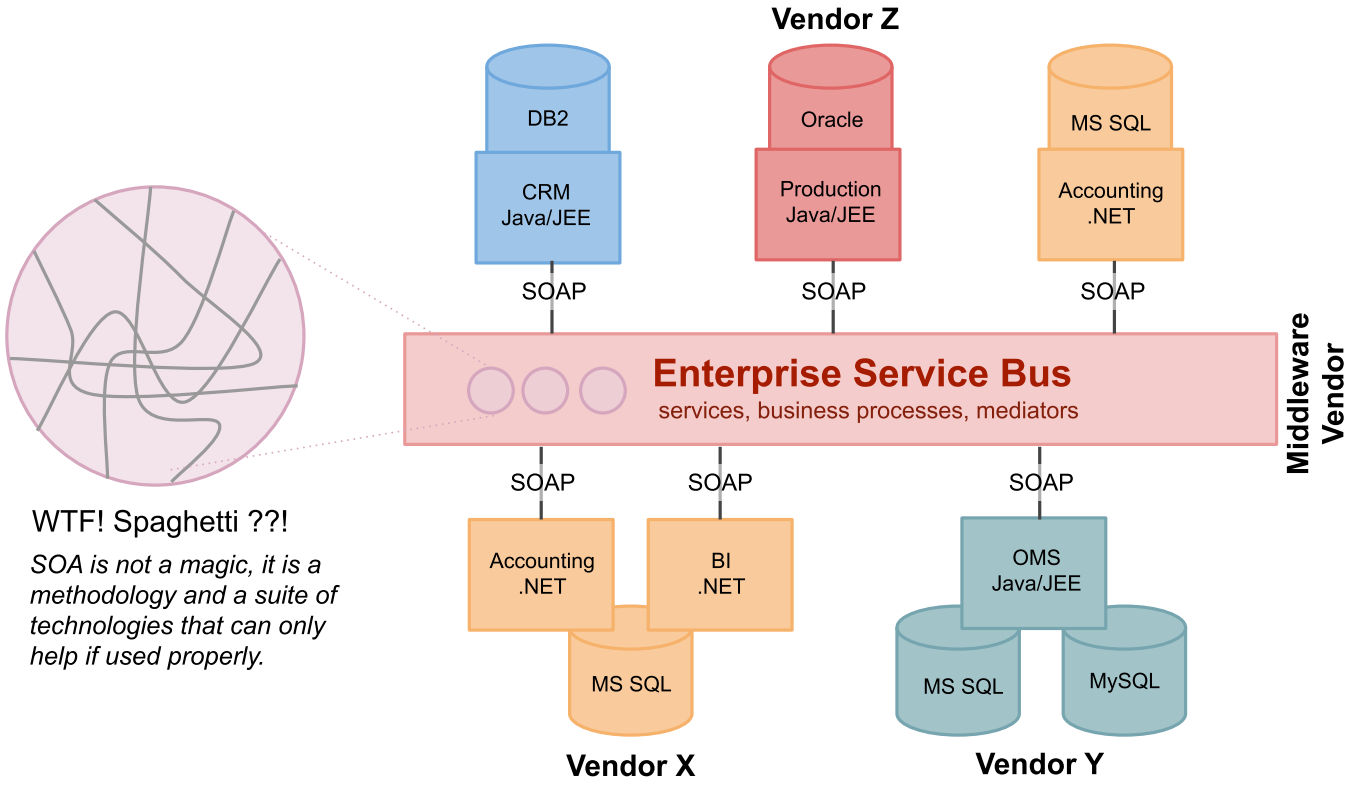
One-to-One Service Integration

- Direct integration of applications
  - *Multiple protocols problem, multiple vendor problem*
  - *Replication of integration functionalities such as interoperability solutions*



Many-to-Many Service Integration

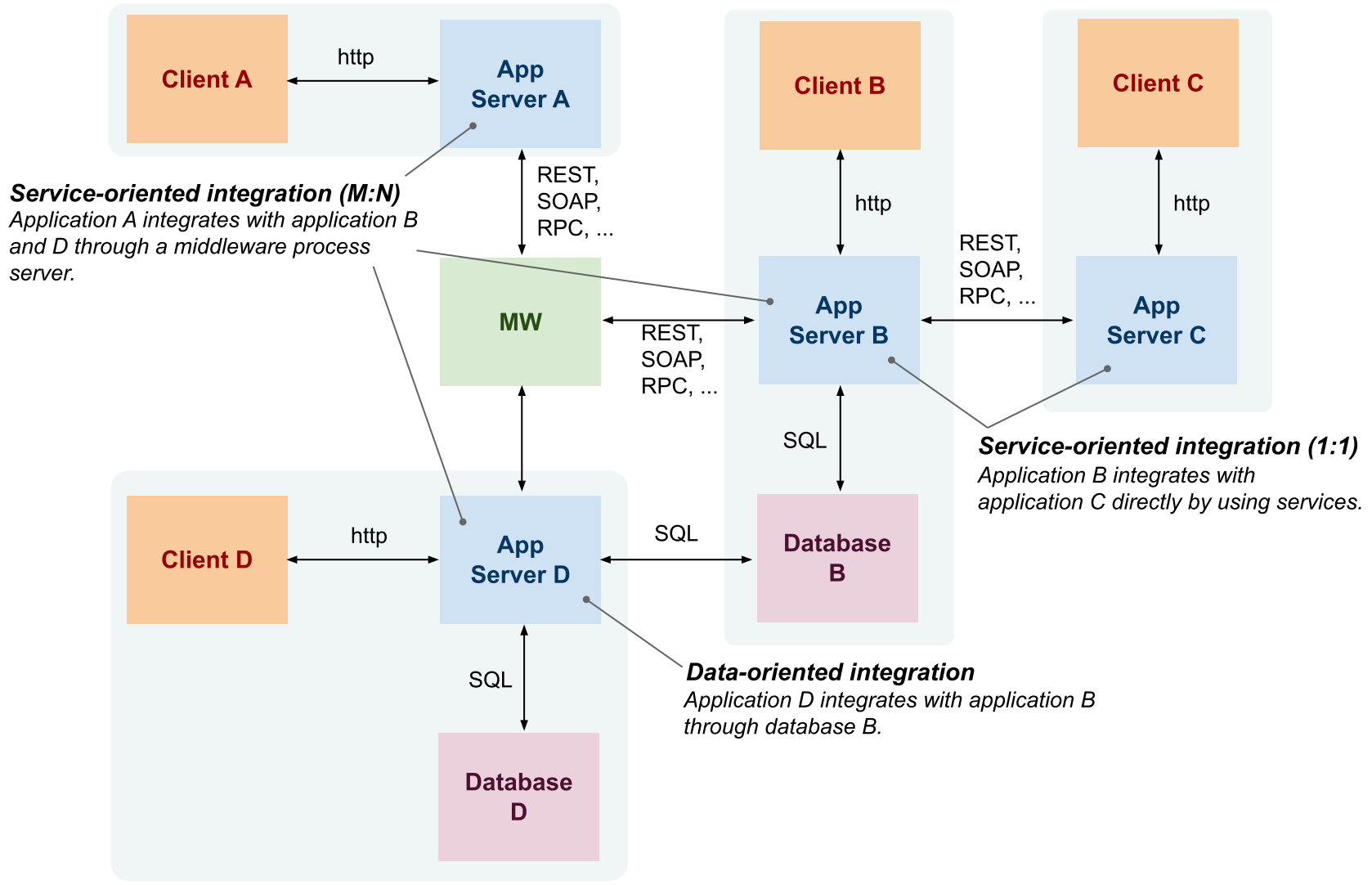
- Enterprise Service Bus – central integration technology
  - *Realizes so called Service Oriented Architecture (SOA)*
  - *Contains various integration components such as process server, mediators, messaging middleware, identity management, etc.*



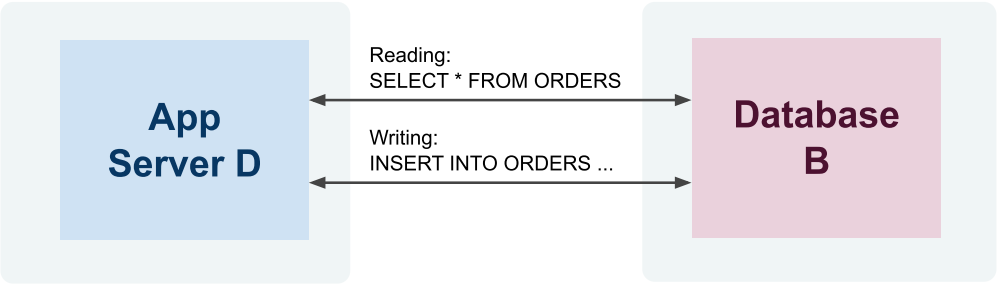
WTF! Spaghetti ??!

SOA is not a magic, it is a methodology and a suite of technologies that can only help if used properly.

Integration Approaches Overview

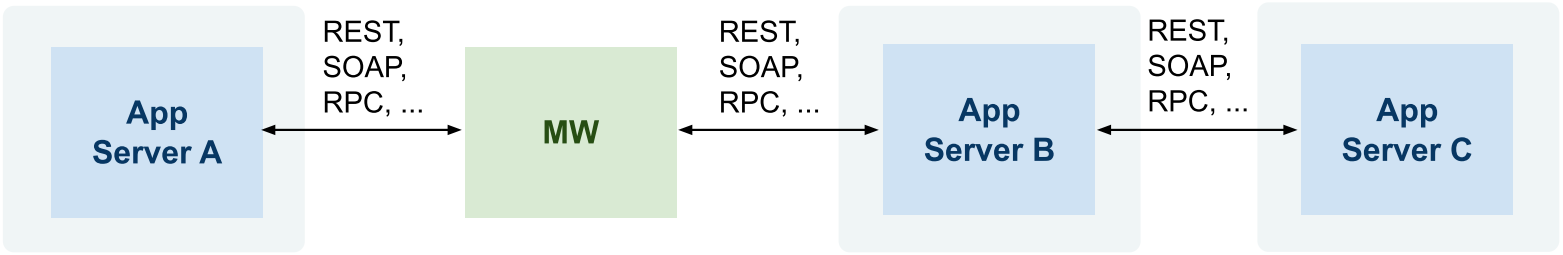


Data-oriented Integration



- Third-party database access
  - *Application D accesses a database of application B directly by using SQL and a knowledge of database B structure and constraints*
  - *In the past: monolithic and two-tier client/server architectures*
  - *Today: ETL (Extract, Transform, Load) technologies*
- Problems
  - *App D must understand complex structures and constraints*
    - → *Data – very complex, includes structure and integrity constraints*
    - → *Functions/processes – hidden in integrity constraints*
    - → *Technical – access mechanisms can vary*

Service-oriented Integration



- Integration at the application layer
  - *Application exposes services that other applications consume*
  - *Services hide implementation details but only define interfaces for integration*
- Problems
  - *Can become unmanageable if not properly designed*
  - *Interoperability*
    - → *Data – limited to input and output messages only*
    - → *Functions/processes – limited to semantics of services*
    - → *Technical – access mechanisms can vary*

Integration and Types of Data

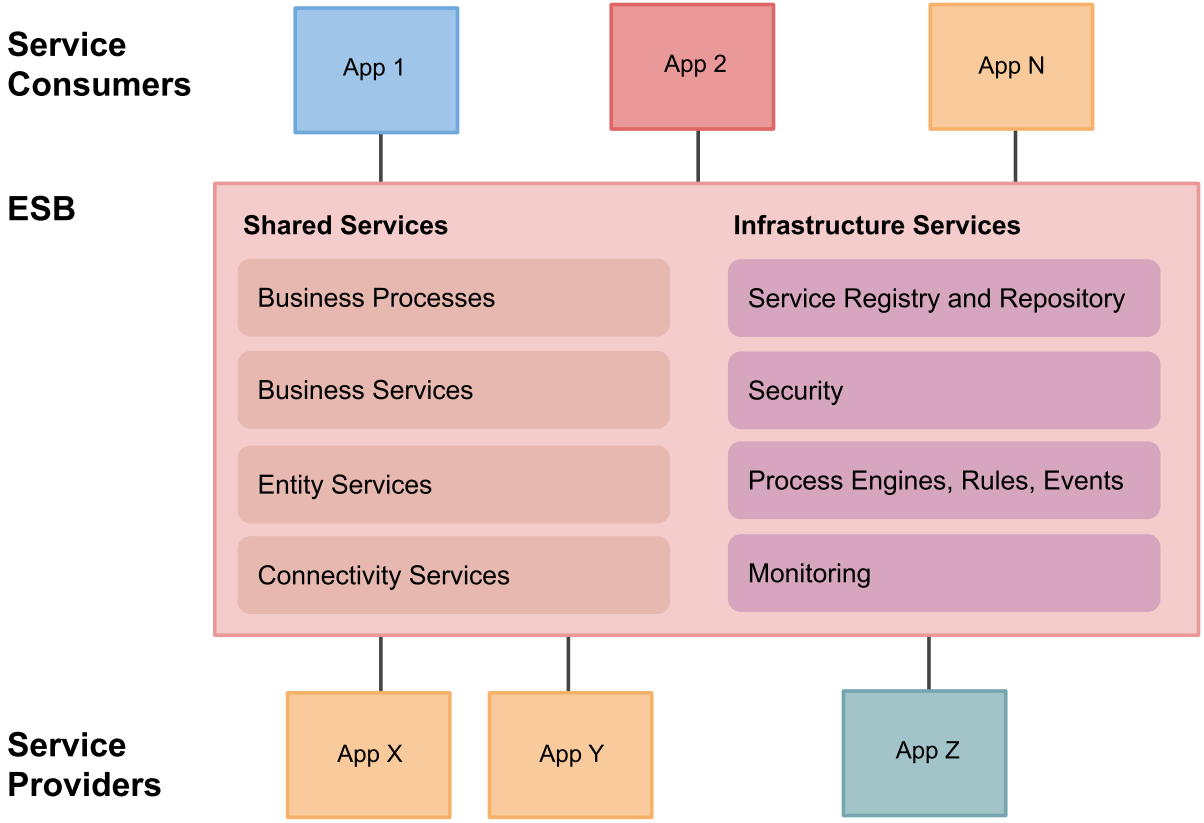
- Real-time data – Web services
  - *Service-oriented integration*
  - *online, realtime communication between a client and a service*
  - *Usually small data and small amount of service invocation in a process*
- Bulk data – ETL
  - *Data-oriented integration*
  - *processing of large amount of data in batches*
  - *Sometimes required for reconciliation across apps*
    - *→ when real-time integration fails and there is poor error handling*
- **SOA provides both Web service and ETL capabilities**

Enterprise Service Bus

- ESB is a central intermediary in SOA
  - *Types of services: shared and infrastructure*
  - *Types of processes: Technical and Business*
- ESB Application
  - *Application running on an application server*
  - *Exposes functionality via Web service interface*
  - *Allows to communicate with various messaging protocols*
- Integration Patterns
  - *Technical-level interoperability – message broker*
  - *Location transparency*
  - *Dynamic routing*
  - *Data transformations – mediator*
  - *Resequencing of messages*
  - *Session pooling*
  - *Service orchestrations – BPMN, BPEL*
  - *Message enrichment*

Service Types

- ESB services
  - *shared services – created for particular domain*
  - *infrastructure services – support integration and interoperability*



Service and non-service enabled providers

Connectivity Services

- Purpose
  - *Adapters for various back-end technologies*
  - *Connectivity to legacy applications*
  - *No business logic, Usually stateless, ESB internal*
- Example
  - *Database adapters*

    → *SQL statement:*

```
1  SELECT ID, NAME FROM CUSTOMERS C
2  WHERE  C.REVENUE > :revenue
```

    `Revenue` – *input parameter*

    `ID, NAME` – *structure of output message*

    → *Expose the SQL statement as a connectivity service*
  - *Example implementation: OSB Proxy service, JCA adapters*

Entity Services

- Purpose
  - *Expose services on top of one or more entities in a database*
  - *Do not add any specific logic to entities' operations*
    - → *Provide CRUD operations only*
  - *May be used to facilitate a Common Data Model*
    - → *Business entities – entities of CDM*
    - → *Business objects – instances of business entities*
    - → *Business Entity Service – manipulations for business entities*
  - *No business logic, usually stateless, ESB internal*
- Example
  - *Two entities in a database:* `CUSTOMERS`, `ADDRESS` *(1:N)*
  - *Business entity* `CUSTOMER`

```
<customer>
  <name>Company.cz</name>
  <invoice-address>
    …
  </invoice-address>
  <main-address>
    …
  </main-address>
</customer>
```

  - *Operarions: read, write*

Business Services

- Purpose
  - *Business/integration logic, can be stateful or stateless*
  - *Atomic business activities*
    - *→ direct mapping to back-end application services*
  - *Can be "imported" in ESB to be used in a business process*
  - *Can be exposed by ESB and add values in terms of business/integration logic or technical processing*
- Example
  - *Data transformation*
    - *→ Back-end application service exposed in CDM language*
  - *Message enrichment*
    - *→ Adds information to content from other sources*
  - *Monitoring*
    - *→ Every invocation of the service logged*
    - *→ Monitoring of business metrics*
      - *→ Number of orders, total revenue per customer*

Business Processes

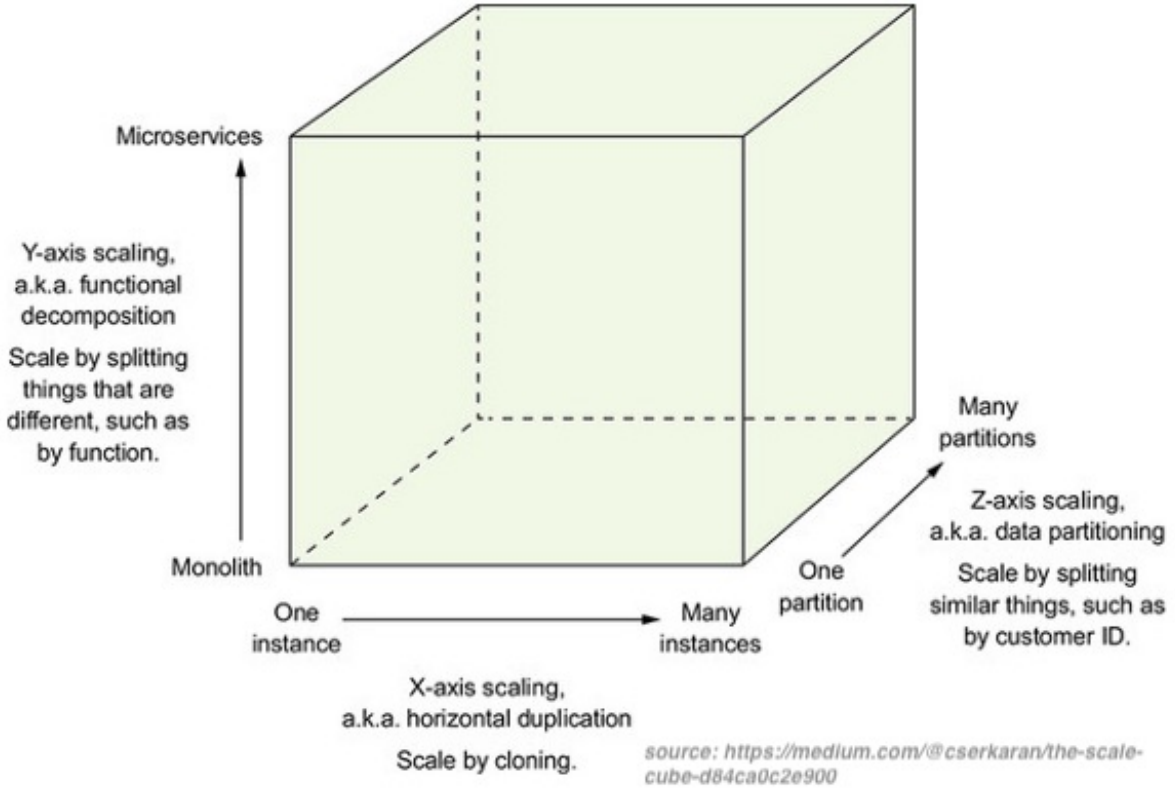
- Purpose
  - *Business/integration logic, usually stateful*
  - *Complex processes involving invocations of multiple business services at various back-end applications*
  - *Handles transformations from various data formats of back-end applications*
  - *Handles **key-mapping***
    - *→ Business entities exist in multiple systems*
    - *→ Each back-end application maintains its own ID for corresponding business objects*
  - *Usually implemented in a process language such as BPMN or BPEL*
  - *OSB uses its own orchestration language which translates to XQuery*
- Example
  - *Order processing*
    - *→ Get customer information from the CRM system*
    - *→ Add line items to OMS*

Overview

- Service Definition
- Integrating Applications
- Enterprise Service Bus
- Microservices Architecture

The Scale Cube

- Three-dimensional scalability model
  - *X-Axis scaling requests across multiple instances*
  - ***Y-Axis scaling decomposes an application into micro-services***
  - *Z-Axis scaling requests across "data partitioned" instances*

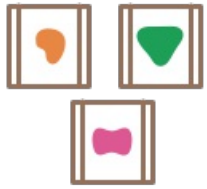


Overview

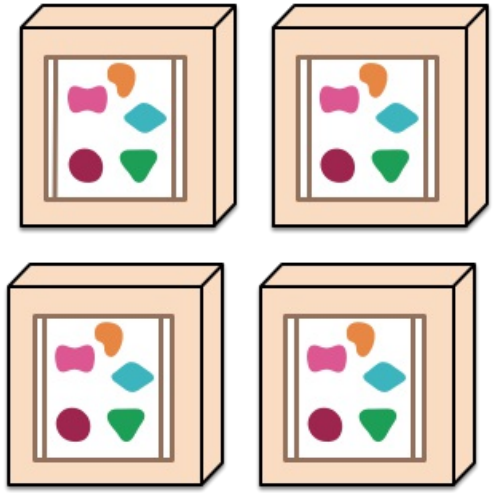
- Emerging software architecture
  - *monolithic vs. decoupled applications*
  - *applications as independenly deployable services*

A monolithic application puts all its functionality into a single process...

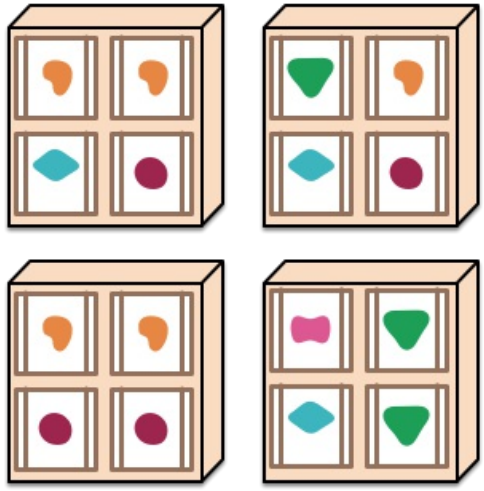
A microservices architecture puts each element of functionality into a separate service...



... and scales by replicating the monolith on multiple servers



... and scales by distributing these services across servers, replicating as needed.



Major Characteristics

- Loosely coupled
  – *Integrated using well-defined interfaces*
- Technology-agnostic protocols
  – *HTTP, they use REST architecture*
- Independently deployable and easy to replace
  – *A change in small part requires to redeploy only that part*
- Organized around capabilities
  – *such as accounting, billing, recommendation, etc.*
- Impplemented using different technologies
  – *polyglot – programming languages, databases*
- Owned by a small team