EN
English

University of Oviedo

Universidad de Oviedo

School of Computer Science

Integration

SOFTWARE
ARCHITECTURE

School of Computer Science

Course 2019/20

Jose E. Labra Gayo

Integration

Application Integration = Biggest challenge

Integration

- Integration styles
    - File transfer
    - Shared database
    - Remote procedure call
    - Messaging
        - Event log
- Topologies
    - Hub & Spoke, Bus
- Service Oriented Architectures
    - WS-*, REST
- Microservices
- Serverless

Integration styles

File transfer

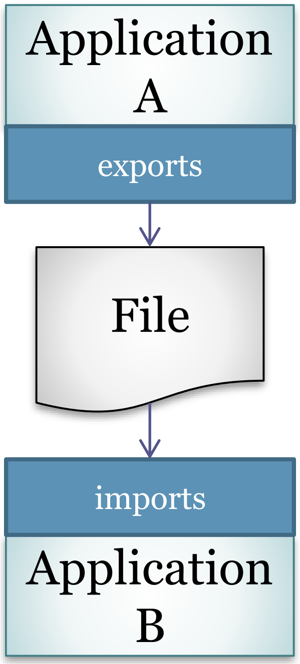
Shared database

Remote procedure call

Messaging

File transfer

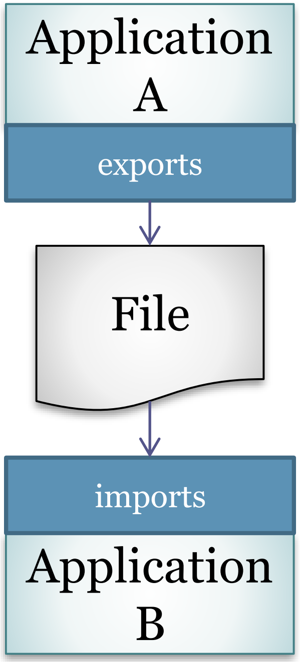
- An application generates a data file that is consumed by another
  - One of the most common solutions
- Advantages
  - Independence between A and B
  - Low coupling
  - Easier debugging
    - By checking intermediate files



File transfer

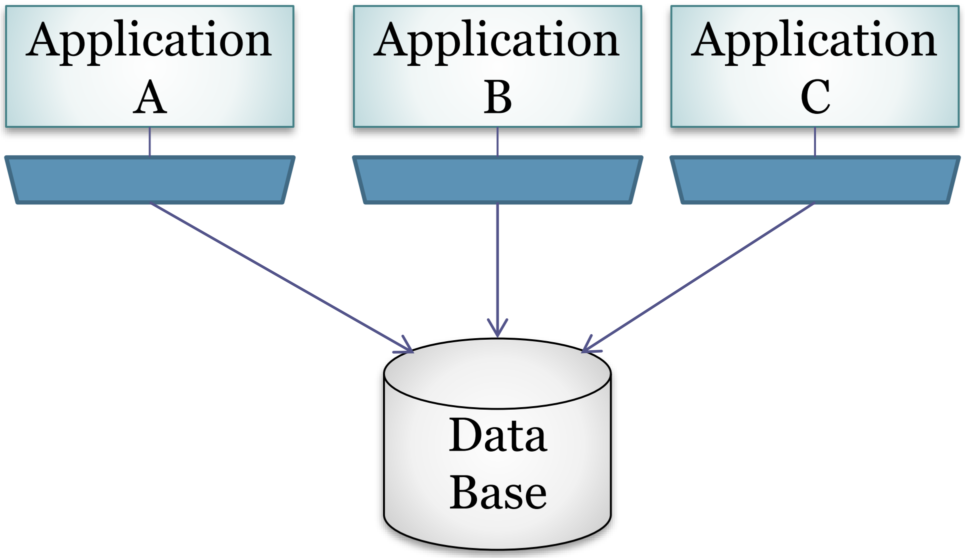
Challenges

- Both applications must agree a common file format
  - It can increase coupling
- Coordination
  - Once the file has been sent, the receiver could modify it ⇒ 2 files!
- It may require manual adjustments



Shared database

- Applications store their data in a shared database
- Advantage
  - Data are always available
  - Everyone has access to the same information
  - Consistency
  - Familiar format
    - SQL for everything



Shared database

Challenges

- Database schema can evolve
  - It requires a common schema for all applications
    - That can cause problems/conflicts
  - External packages are needed (common database)
- Performance and scalability
  - Database as a bottleneck
- Synchronization
  - Distributed databases can be problematic
  - Scalability
  - NoSQL ?

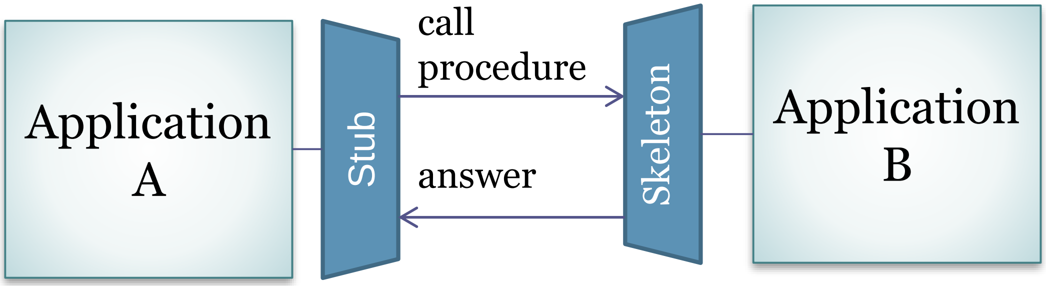
Shared database

Variants

- *Data warehousing*: Database used for data analysis and reports
- ETL: process based on 3 stages
  - Extraction: Get data from heterogeneous sources
  - Transform: Process data
  - Load: Store data in a shared database

Remote procedure call

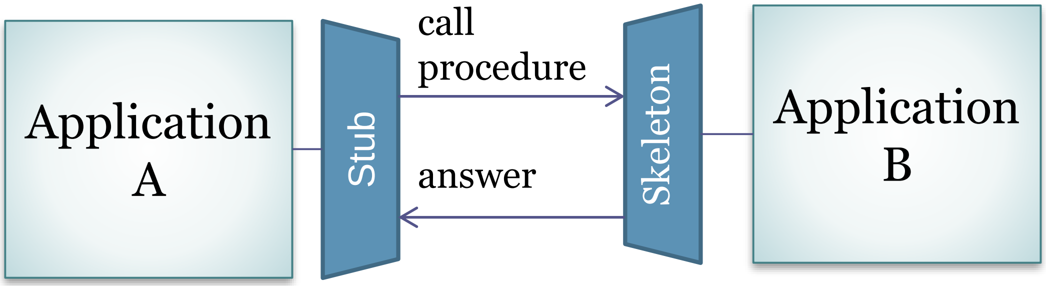
- An application calls a function from another application that could be in another machine
  - Invocation can pass parameters
  - Obtains an answer
- Lots of applications
  - RPC, RMI, CORBA, .Net Remoting, ...
  - Web services, ...



Remote procedure call

Advantages

- Encapsulation of implementation
- Multiple interfaces for the same information
  - Different representations can be offered
- Model familiar for developers
  - It is similar to invoke a method



Remote procedure call

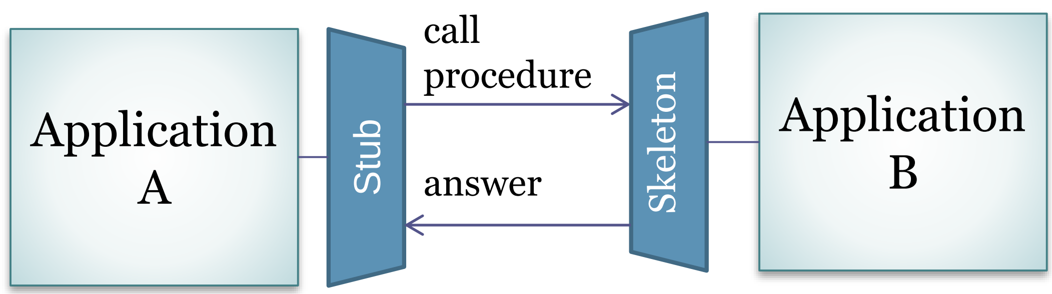
Challenges

- False sense of simplicity
  - Remote procedure ≠ procedure
    - 8 fallacies of distributed computing
- Synchronous procedure calls
  - Increase application coupling

The network is reliable
Latency is zero
Bandwidth is infinite
The network is secure
Topology doesn't change
There is one administrator
Transport cost is zero
The network is homogeneous

8 fallacies of distributed computing

http://en.wikipedia.org/wiki/Fallacies_of_distributed_computing



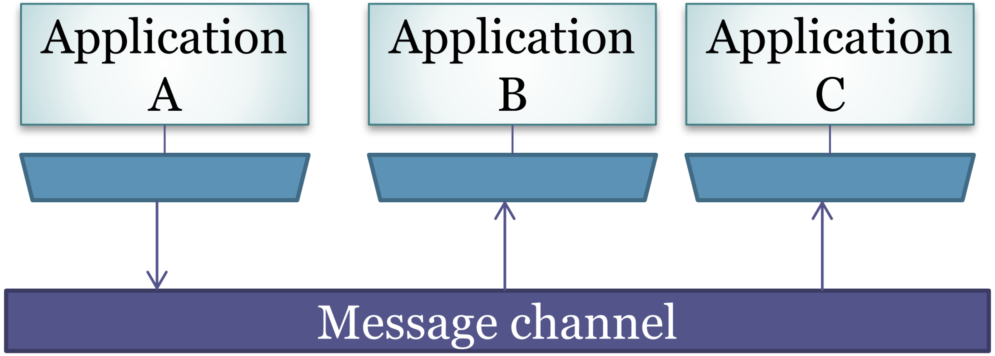
https://www.youtube.com/watch?v=UZxLYv5RFyI&t=54s

Remote procedure call

- New proposals: gRPC (https://grpc.io/)
- Google proposal
- High performance RPC framework
  - http/2 transport protocol

Messaging

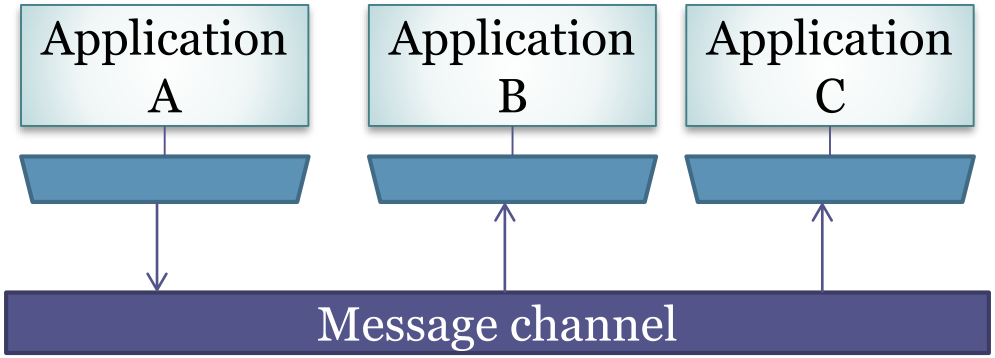
- Multiple independent applications communicate sending messages through a channel
- Asynchronous communication
  - Applications send messages a continue their execution



Messaging

Advantages

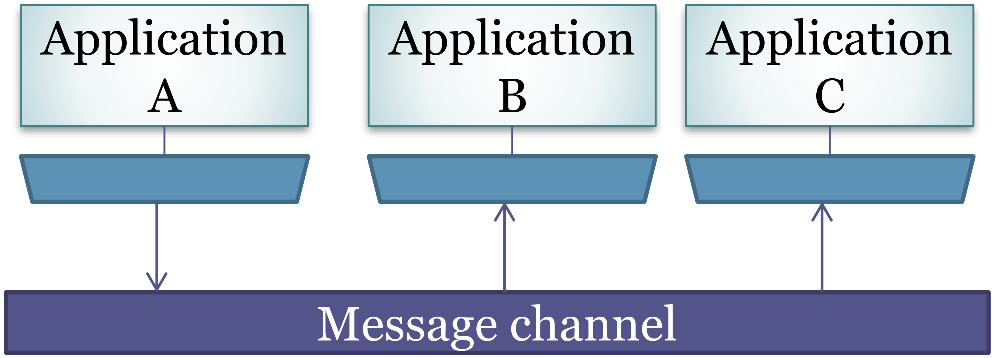
- Low coupling
  - Applications are independent between each other
- Asynchronous communication
  - Applications continue their execution
- Implementation encapsulation
  - The only thing exposed is the type of messages



Messaging

Challenges

- Implementation complexity
  - Asynchronous communication
- Data transfer
  - Adapt message formats
- Different topologies



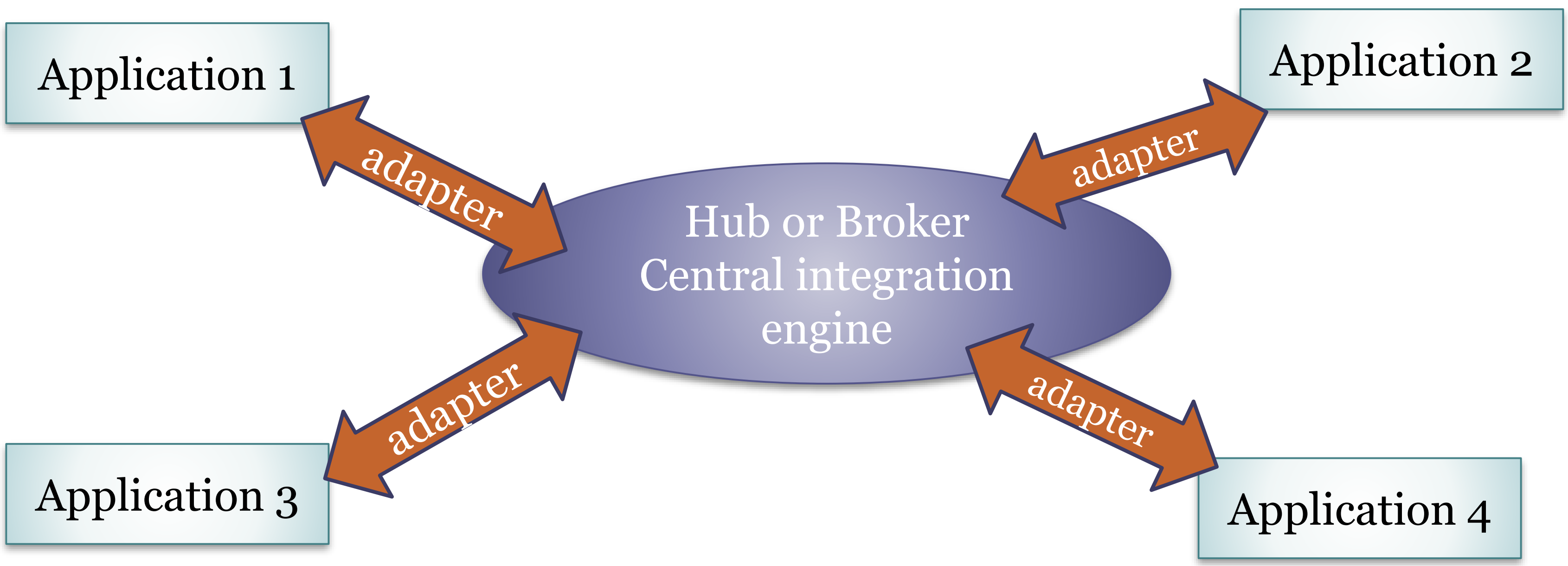
Integration topologies

Hub & Spoke

Bus

Hub & Spoke

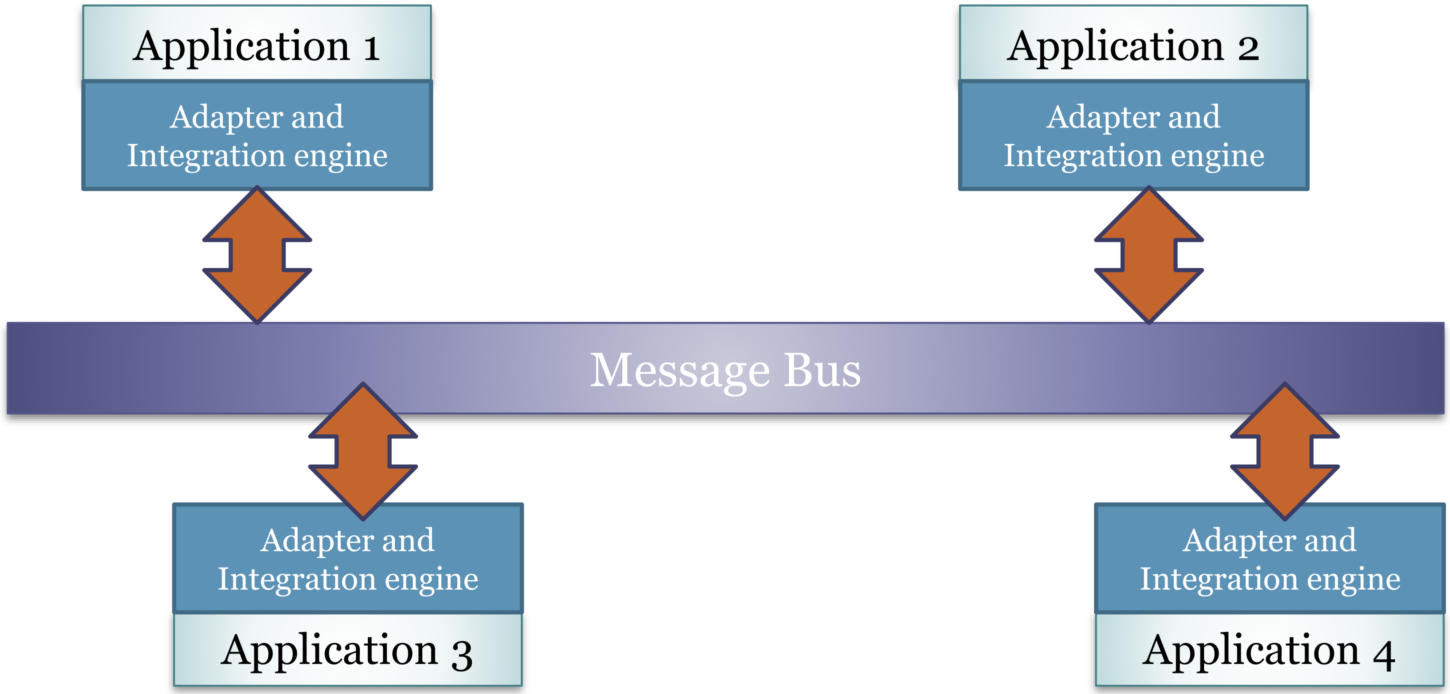
Related with Broker pattern
Hub = Centralized message Broker
- It is in charge of integration



Bus

Each application contains its own integration machine

Publish/Subscribe style



Bus

- ESB - Enterprise Service Bus
- Defines the messaging backbone
  - Some tasks
    - Protocol conversion
    - Data transformation
    - Routing
- Offers an API to develop services
  - MOM (Message Oriented Middleware)

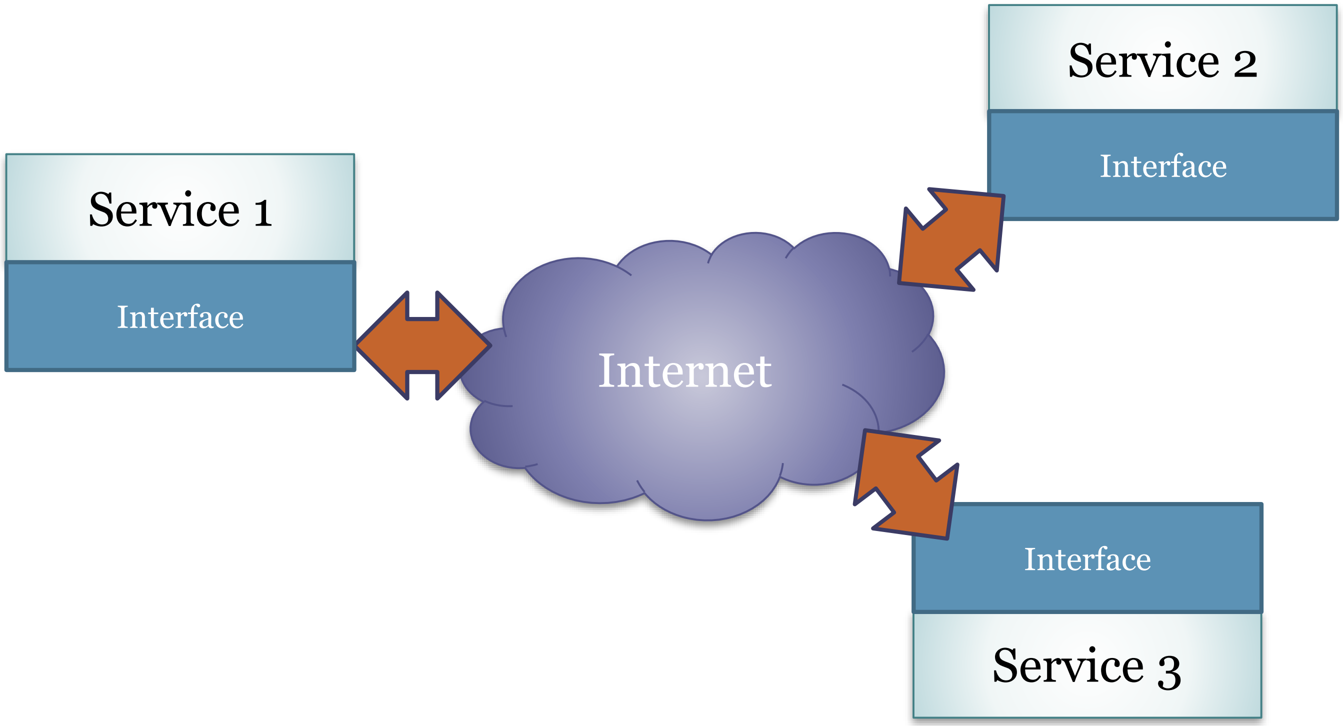
Service Oriented Architectures

SOA
WS-*
REST

SOA

SOA = Service Oriented Architecture

Services are defined by an interface



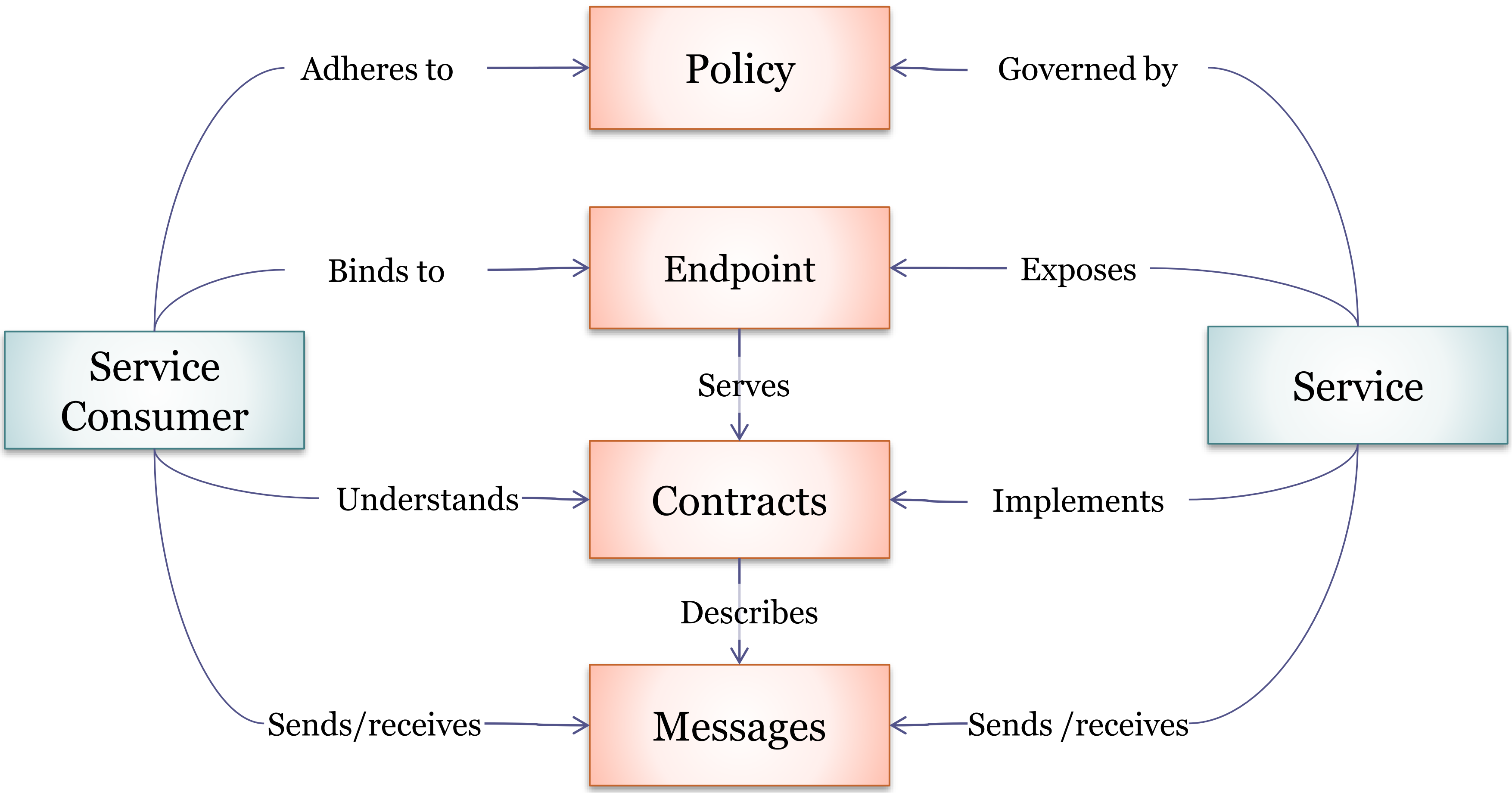
SOA

Elements

- Provider: Provides service
- Consumer: Does requests to the service
- Messages: Exchanged information
- Contract: Description of the functionality provided by the service
- Endpoint: Service location
- Policy: Service level agreements
  - Security, performance, etc.

SOA

Constraints



Fuente: SOA Patterns, A. Rottem Gal Oz

SOA

Advantages

- Independent of language and platform
- Interoperability
  - Use of standards
- Low coupling
- Decentralized
- Reusability
- Scalability
  - one-to-many vs one-to-one
- Partial solution for legacy systems
  - Adding a web services layer

Challenges

- Performance
  - E.g. real time systems
- Overkill in very homogeneous environments
- Security
  - Risk of public exhibition of API to external parties
  - DoS attacks
- Service composition and coordination

SOA

Variants:

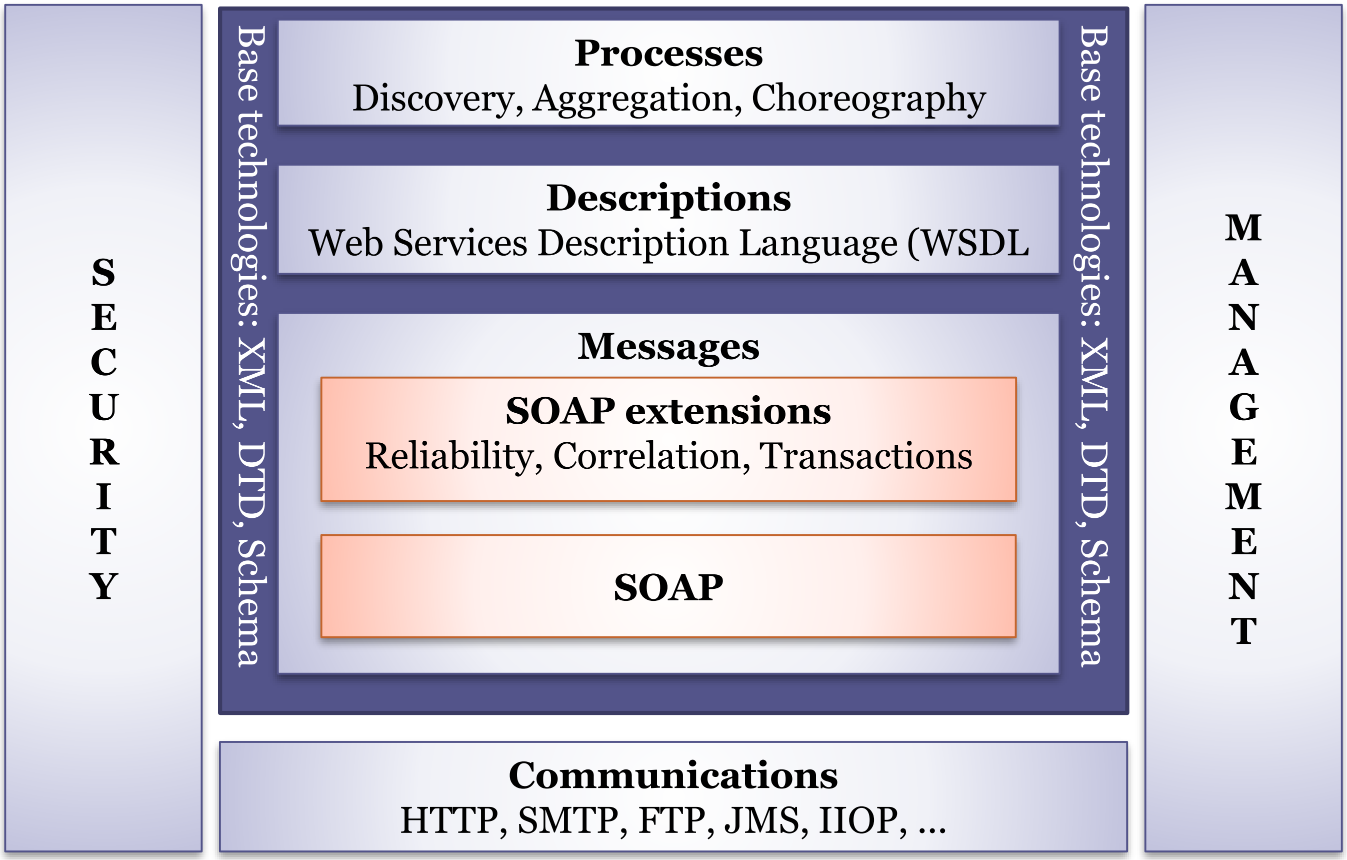
- WS-*
- REST

WS-*

- WS-* model = Set of specifications
  - SOAP, WSDL, UDDI, etc....
  - Proposed by W3c, OASIS, WS-I, etc.
  - Goal: Reference SOA implementation

WS-*

Web Services Architecture



Web Services Standards

Deutsche Post
Deutsche Post AG
SOP Group
Straesschensweg 10
53113 Bonn
Phone +49 (2 28) 182 19019
Fax +49 (2 28) 182 19099
SOP-Group@DeutschePost.de
www.SOP-Group.com

Interoperability Issues

Basic Profile 1.1 · WS-I Final Specification

▲ **Basic Profile** – The Basic Profile provides implementation guidelines for how related set of non-proprietary Web Service specifications should be used together for best interoperability.

Attachments Profile 1.0 · WS-I Final Specification

▲ **Attachments Profile** – The Attachment Profile 1.0 complements the Basic Profile 1.1 to add support for interoperable SOAP Messages with attachments-based Web Services.

Simple SOAP Binding Profile 1.0 · WS-I Final Specification

▲ **Simple SOAP Binding Profile** – The Simple SOAP Binding Profile consists of those Basic Profile 1.0 requirements related to the serialization of the envelope and its representation in the message.

Basic Security Profile WS-I Working Group Draft

▲ **Basic Security Profile** provides guidance on the use of WS-Security and the User Name and X.509 security token formats.

REL Token Profile WS-I Working Group Draft

▲ **REL Token Profile** is the interoperability profile for the Rights Expression Language (REL) security token that is used with WS-Security.

SAML Token Profile WS-I Working Group Draft

▲ **SAML Token Profile** is the interoperability profile for the SAML security token that is used with WS-Security.

Conformance Claim Attachment Mechanism (CCAM) 1.0 · WS-I Final Specification

▲ **Conformance Claim Attachment Mechanism (CCAM)** defines mechanisms that can be used to attach conformance claims to Web services artifacts.

Reliable Asynchronous Messaging Profile (RAMP) 1.0 IBM, Ford Motor Company

▲ **Reliable Asynchronous Messaging Profile (RAMP)** is a profile that enables, among other things, basic B2B integration scenarios using Web services technologies.

Business Process Specifications

Business Process Execution Language for Web Services (BPEL4WS) · 1.1 · BEA Systems, IBM, Microsoft, SAP, Siebel Systems · OASIS-Standard

▶ **Business Process Execution Language for Web Services** (BPEL4WS) provides a language for the formal specification of business processes and business interaction protocols using Web Services.

Business Process Management Language (BPML) · 1.0 · BPMI.org Final Draft

▶ **Business Process Management Language** (BPML) provides a meta-language for expressing business processes and supporting entities.

WS-Choreography Model Overview 1.0 · W3C Working Draft

▶ **WS-Choreography Model Overview** defines the format and structure of the (SOAP) messages that are exchanged, and the sequence and conditions in which the messages are exchanged.

Web Service Choreography Description Language (CDL4WS) · 1.0 · W3C Working Draft

▶ **Web Service Choreography Description Language** (CDL4WS) is to specify a declarative, XML based language that defines from a global viewpoint the common and complementary observable behaviour, where message exchanges occur, and when the jointly agreed ordering rules are satisfied.

Web Service Choreography Interface (WSCI) · 1.0 · W3C Sun Microsystems, SAP, BEA Systems and Intalio · Note

Management Specifications

Web Services Management Foundation (WSMF-Foundation) · 2.0 Hewlett-Packard · Working Draft

WS-Events 2.0 · Hewlett-Packard Working Draft

Web Services Management (WSMF-WSM) · 2.0 Hewlett-Packard Working Draft

WS-Management AMD, Dell, Intel, Microsoft and Sun Microsystems Published Specification

Management Using Web Services (MUWS) · 1.0 · OASIS OASIS-Standard

Management Of Web Services (MOWS) · 1.0 · OASIS Committee Draft

Metadata Specifications

WS-Policy BEA Systems, IBM, Microsoft, SAP, Sonic Software and VeriSign Public Draft

▶ **WS-Policy** describes the capabilities and constraints of the policies on intermediaries and endpoints (e.g. business rules, required security tokens, supported encryption algorithms, privacy rules).

WS-PolicyAssertions BEA Systems, IBM, Microsoft, SAP Public Draft

▶ **WS-PolicyAssertions** provides an initial set of assertions to address some common needs of Web Services applications.

WS-PolicyAttachment BEA Systems, IBM, Microsoft, SAP, Sonic Software and VeriSign Released Draft

▶ **WS-PolicyAttachment** defines two general-purpose mechanisms for associating policies with the subjects to which they apply; the policies may be defined as part of existing metadata about the subject or the policies may be defined independently and associated through an external binding to the subject.

WS-Discovery Microsoft, BEA Systems, Canon, Intel and webMethods Draft

▶ **WS-Discovery** defines a multicast discovery protocol for dynamic discovery of services on ad-hoc and managed networks.

WS-MetadataExchange BEA Systems, Computer Associates, IBM, Microsoft, SAP, Sun Microsystems, and webMethods Public Draft

▶ **WS-MetadataExchange** enables a service to provide metadata to others through a Web services interface. Given only a reference to a Web service, an user can access a set of WSDL/SOAP operations to retrieve the metadata that describes the service.

Universal Description, Discovery and Integration (UDDI) · v2 OASIS-Standard Technical Commitee

▶ **Universal Description, Discovery and Integration** is the definition of a set of services supporting the description and discovery of businesses, organizations, and other Web services providers, the Web services they make available, and the technical interfaces which may be used to access those services.

Web Service Description Language WSDL · 2.0 W3C Working Draft

Web Service Description Language (WSDL) is an XML-based language for describing Web services and how to access them. It specifies the location of the service and the operations (or methods) the service exposes.

Web Service Description Language WSDL · 1.1 W3C Note

Reliability Specifications

WS-ReliableMessaging BEA Systems, IBM, Microsoft, and Tibco Draft

▲ **WS-ReliableMessaging** describes a protocol that allows Web Services to communicate reliably in the presence of software component, system, or network failures. It defines a SOAP binding that is required for interoperability.

WS-Reliability 1.1 OASIS OASIS-Standard

▲ **WS-Reliability** defines a SOAP-based protocol for exchanging SOAP messages with guaranteed delivery, no duplicates, and guaranteed message ordering. WS-Reliability is defined as SOAP header extensions and is independent of the underlying protocol.

Security Specifications

WS-Security 1.0 OASIS

▲ **WS-Security** describes enhancements to SOAP messaging to provide quality of protection through message integrity, message confidentiality, and single message authentication.

WS-Security: Username Token Profile 1.0 · OASIS OASIS-Standard

▲ **WS-Security: Username Token Profile** describes how a Web Service consumer can supply a UsernameToken as a means of identifying the requestor by "username", and optionally using a password (or shared secret, etc.) to authenticate that identity to the Web Service producer.

WS-Security: SOAP Message Security 1.0 OASIS OASIS-Standard

▲ **WS-Security: SOAPMessage Security** describes enhancements to SOAP messaging to provide message integrity and confidentiality. Specifically, this specification provides support for multiple security token formats, trust domains, signature formats, and encryption technologies. The token formats and semantics for using these are defined in the associated profile documents.

WS-SecurityPolicy 1.0 IBM, Microsoft, RSA Security, VeriSign Public Draft

▲ **WS-SecurityPolicy** defines how to describe policies related to various features defined in the WS-Security specification.

WS-Security: Kerberos Binding 1.0 OASIS Working Draft

▲ **WS-Security: Kerberos Binding** defines how to encode Kerberos tickets and attach them to SOAP messages. As well, it specifies how to add signatures and encryption to the SOAP message, in accordance with WS-Security, which uses and references the Kerberos tokens.

WS-Trust 1.1 · BEA Systems, Computer Associates, IBM, Layer 7 Technologies, Microsoft, Netegrity, Oblix, OpenNetwork, Ping Identity Corporation, Reactivity, RSA Security, VeriSign and Westbridge Technology · Initial Draft

▲ **WS-Trust** describes a framework for trust models that enables Web Services to securely interoperate. It uses WS-Security base mechanisms and defines additional primitives and extensions for security token exchange to enable the issuance and dissemination of credentials within different trust domains.

WS-Security: SAML Token Profile 1.0 OASIS OASIS-Standard

▲ **WS-Security: SAML Token Profile** defines the use of Security Assertion Markup Language (SAML) v1.1 assertions in the context of WSS: SOAP Message Security including for the purpose of securing SOAP messages and SOAP message exchanges.

WS-Federation 1.0 IBM, Microsoft, BEASystems, RSA Security, VeriSign · Initial Draft

▲ **WS-Federation** describes how to manage and broker the trust relationships in a heterogeneous federated environment including support for federated identities.

WS-Security: X.509 Certificate Token Profile 1.0 · OASIS OASIS-Standard

▲ **WS-Security: X.509 Certificate Token Profile** describes the use of the X.509 authentication framework with the WS-Security: SOAP Message Security specification.

WS-SecureConversation 1.1 · BEA Systems, Computer Associates, IBM, Layer 7 Technologies, Microsoft, Netegrity, Oblix, OpenNetwork, Ping Identity Corporation, Reactivity, RSA Security, VeriSign and Westbridge Technology Public Draft

▲ **WS-SecureConversation** specifies how to manage and authenticate message exchanges between parties including security context exchange and establishing and deriving session keys.

Transaction Specifications

WS-Business Activity Microsoft, BEA Systems, IBM Published

▶ **WS-Business Activity** defines protocols that enable existing business process and work flow systems to wrap their proprietary mechanisms and interoperate across trust boundaries and different vendor implementations.

WS-Atomic Transaction Microsoft, BEA Systems, IBM Published

▶ **WS-Atomic Transaction** defines protocols that enable existing transaction processing systems to wrap their proprietary protocols and interoperate across different hardware and software vendors.

WS-Coordination Microsoft, BEA Systems, IBM Published

▶ **WS-Coordination** describes an extensible framework for providing protocols that coordinate the actions of distributed applications.

WS-Composite Application Framework (WS-CAF) · 1.0 Arjuna Technologies, Fujitsu, IONA, Oracle and Sun Microsystems Committee Draft

▶ **WS-Composite Application Framework** (WS-CAF) is a collection of three specifications aimed at solving problems that arise when multiple Web Services are used in combination. It proposes standard, interoperable mechanisms for managing shared context and ensuring business processes achieve predictable results and recovery from failure.

WS-Context (WS-CTX) · 1.0 OASIS Committee Draft

▶ **WS-Context** (WS-CTX) is intended as a lightweight mechanism for allowing multiple Web Services to share a common context.

WS-Coordination Framework (WS-CF) · 1.0 · OASIS Committee Draft

▶ **WS-Coordination Framework** (WS-CF) allows the management and coordination in a Web Services interaction of a number of activities related to an overall application.

WS-Transaction Management (WS-TXM) · 1.0 · OASIS Committee Draft

▶ **WS-Transaction Management** (WS-TXM) defines a core infrastructure service consisting of a Transaction Service for Web Services.

Resource Specifications

Web Services Resource Framework (WSRF) OASIS

▶ **Web Services Resource Framework** (WSRF) defines a family of specifications for accessing stateful resources using Web Services.

WS-BaseFaults (WSRF) 1.2 OASIS Working Draft

▶ **WS-BaseFaults** (WSRF) defines a base set of information that may appear in fault messages. WS-BaseFaults defines an XML schema type for base faults, along with rules for how this base fault type is used and extended by Web Services.

WS-ServiceGroup (WSRF) 1.2 OASIS Working Draft

▶ **WS-ServiceGroup** (WSRF) defines a means by which Web Services and WS-Resources can be aggregated or grouped together for a domain specific purpose.

WS-ResourceProperties 1.2 OASIS Working Draft

▶ **WS-ResourceProperties** specifies the means by which the definition of the properties of a WS-Resource may be declared as part of the Web Service interface. The declaration of the WS-Resource's properties represents a projection of or a view on the WS-Resource's state.

WS-ResourceLifetime 1.2 OASIS Working Draft

▶ **WS-ResourceLifetime** standardizes the terminology, concepts, message exchanges, WSDL and XML needed to monitor the lifetime of, and destroy WS-Resources. Additionally, it defines resource properties that may be used to inspect and monitor the lifetime of a WS-Resource.

WS-Transfer BEA Systems, Computer Associates, Microsoft, Sonic Software and Systinet Working Draft

▶ **WS-Transfer** describes a general SOAP-based protocol for accessing XML representations of Web service-based resources.

Resource Representation SOAP Header Block (RRSHB) W3C Recommendation

▶ **Resource Representation SOAP Header Block (RRSHB)** allows SOAP message recipients to access cached representations of resources.

Messaging Specifications

WS-Notification 1.2 OASIS Working Draft

▶ **WS-Notification** is a family of related white papers and specifications that define a standard Web services approach to notification using a topic-based publish/subscribe pattern.

WS-Eventing BEA Systems, Computer Associates, IBM, Microsoft, Sun Microsystems and TIBCO Software Public Draft

▶ **WS-Eventing** defines a baseline set of operations that allow Web services to provide asynchronous notifications to interested parties.

WS-Topics 1.2 OASIS Working Draft

▶ **WS-Topics** defines three topic expression dialects that can be used as subscription expressions in subscribe request messages and other parts of the WS-Notification system.

WS-BrokeredNotification 1.2 OASIS Public Review Draft

▶ **WS-BrokeredNotification** defines the interface for the NotificationBroker. A NotificationBroker is an intermediary, which, among other things, allows publication of messages from entities that are not themselves service providers.

WS-Addressing 1.0 W3C Candidate Recommendation

▶ **WS-Addressing** provides transport-neutral mechanisms to address Web services and messages. This specification defines XML elements to identify Web Service endpoints and to secure end-to-end endpoint identification in messages.

WS-BaseNotification 1.2 OASIS Public Review Draft

▶ **WS-BaseNotification** standardizes the terminology, concepts, operations, WSDL and XML needed to express the basic roles involved in Web services publish and subscribe for notification message exchange.

WS-Enumeration BEA Systems, Computer Associates, Microsoft, Sonic Software and Systinet Public Draft

▶ **WS-Enumeration** describes a general SOAP-based protocol for enumerating a sequence of XML elements that is suitable for traversing logs, message queues, or other linear information models.

SOAP

SOAP 1.2 W3C Recommendation

SOAP Message Transmission Optimization Mechanism 1.0 · W3C Recommendation

▶ **SOAP Message Transmission Optimization Mechanism** describes an abstract feature for optimizing the transmission and/or wire format of a SOAP message.

SOAP 1.1 W3C Note

▶ **SOAP** is a lightweight, xml-based protocol for exchange of information in a decentralized, distributed environment.

XML Specifications

XML 1.1 W3C Recommendation

▶ **XML** – Extensible Markup Language is a pared-down version of SGML, designed especially for Web documents. It allows one to create own customized tags, enabling the definition, transmission, validation, and interpretation of data between applications and between organizations.

XML 1.0 W3C Recommendation

Namespaces in XML 1.0 W3C Recommendation

▶ **Namespaces in XML** provide a simple method for qualifying element and attribute names used in XML documents by associating them with namespaces identified by IRI references.

XML Information Set 1.1 W3C Recommendation

▶ **XML Information Set** is an abstract data set to provide a consistent set of definitions for use in other specifications that need to refer to the information in a well-formed XML document.

XML Information Set 1.0 W3C Recommendation

XML Schema 1.0 W3C Recommendation

▶ **XML Schema** – XML Schema Definition Language is an XML language for describing and constraining the content of XML documents.

XML-binary Optimized Packaging (XOP) 1.0 W3C Recommendation

▶ **XML-binary Optimized Packaging (XOP)** provides a standard method for applications to include binary data, as is, along with an XML document in a package. As a result, applications need less space to store the data and less bandwidth to transmit it.

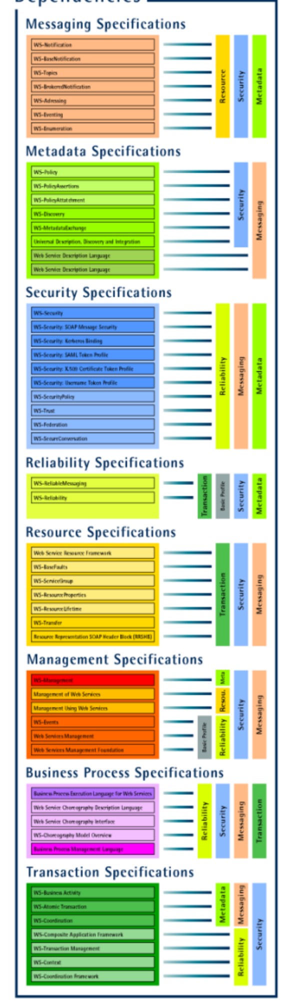
Standards Bodies

OASIS The **Organization for the Advancement of Structured Information Standards (OASIS)** is a not-for-profit, international consortium that drives the development, convergence, and adoption of e-business standards. The consortium produces more Web services standards than any other organization along with standards for security, e-business, and standardization efforts in the public sector and for application-specific markets.

W3C The **World Wide Web Consortium (W3C)** was created in October 1994 to lead the World Wide Web to its full potential by developing common protocols that promote its evolution and ensure its interoperability. W3C is designing the infrastructure, and defining the architecture and the core technologies for Web services. In January 2002, the Web Services Activity was launched, subsuming the XML Protocol Activity and extending its scope.

WS-I The **Web Services Interoperability Organization (WS-I)** is an open industry organization chartered to promote Web services interoperability across platforms, operating systems and programming languages. The organization's diverse community of Web services leaders helps customers to develop interoperable Web services by providing guidance, recommended practices and supporting resources.

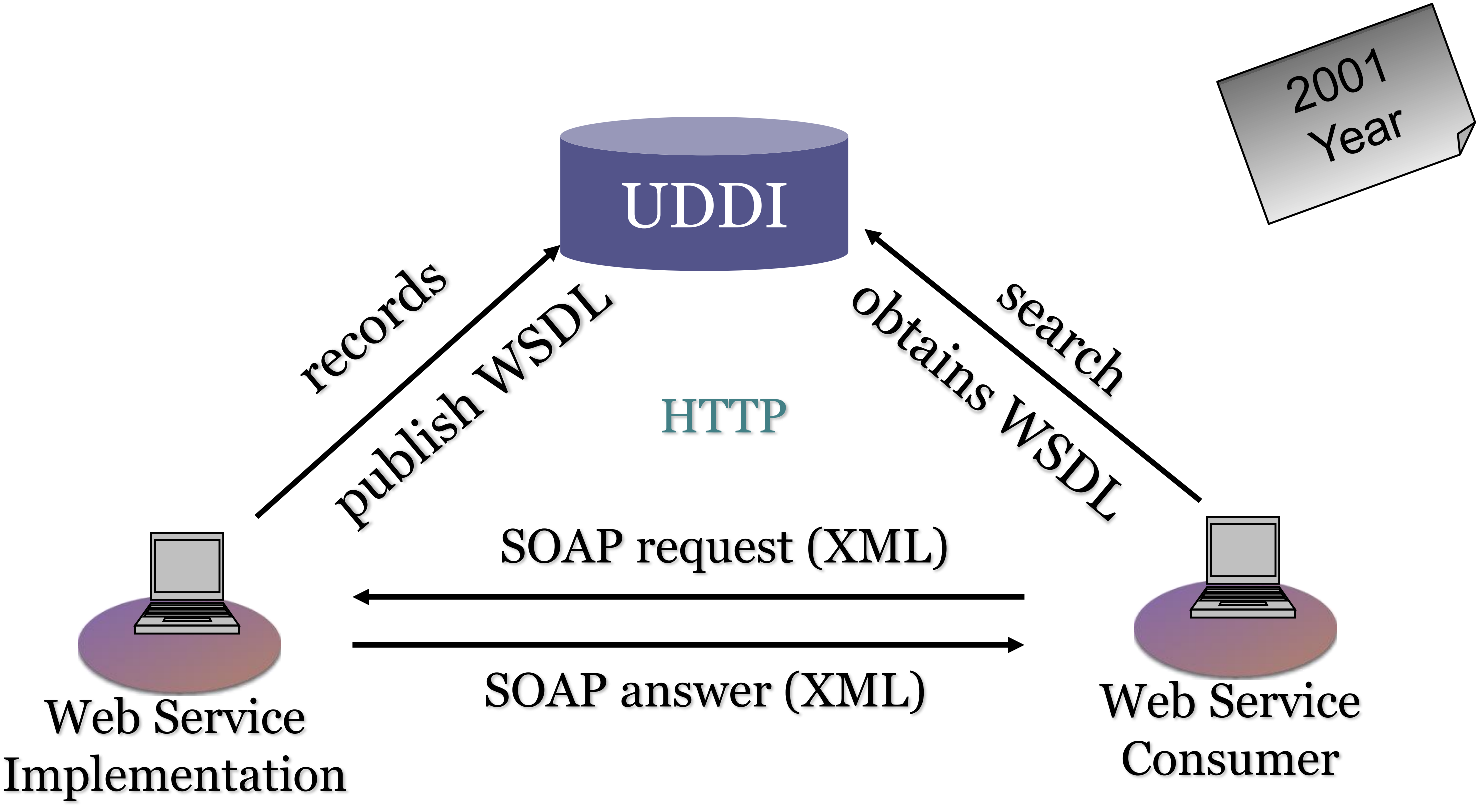
Dependencies



Version 2.0 · September 2005

innoQ
innoQ Deutschland GmbH
Halskestraße 17
D-40880 Ratingen
Telefon +49 (0) 2102 - 77 162 - 100
Telefax +49 (0) 2102 - 77 16 - 01
innoQ Schweiz GmbH
Gewerbestrasse 11
CH-6330 Cham
Telefon +41 (0) 41 - 743 01 11
Telefax +41 (0) 41 - 743 01 19
info@innoq.com · www.innoq.com

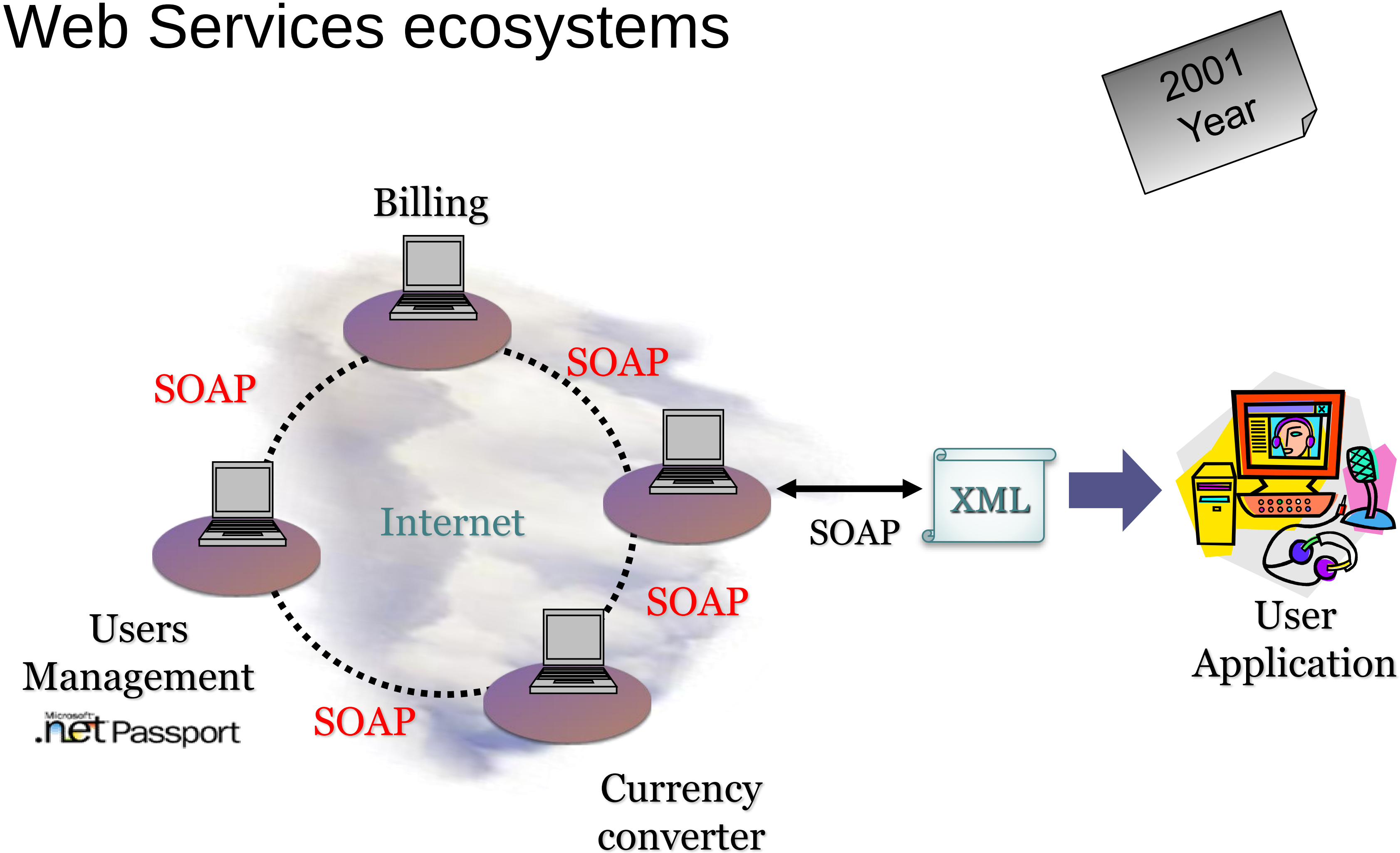
WS-*



2001
Year
UDDI
records
publish WSDL
search
obtains WSDL
HTTP
SOAP request (XML)
SOAP answer (XML)
Web Service
Implementation
Web Service
Consumer

WS-*

Web Services ecosystems



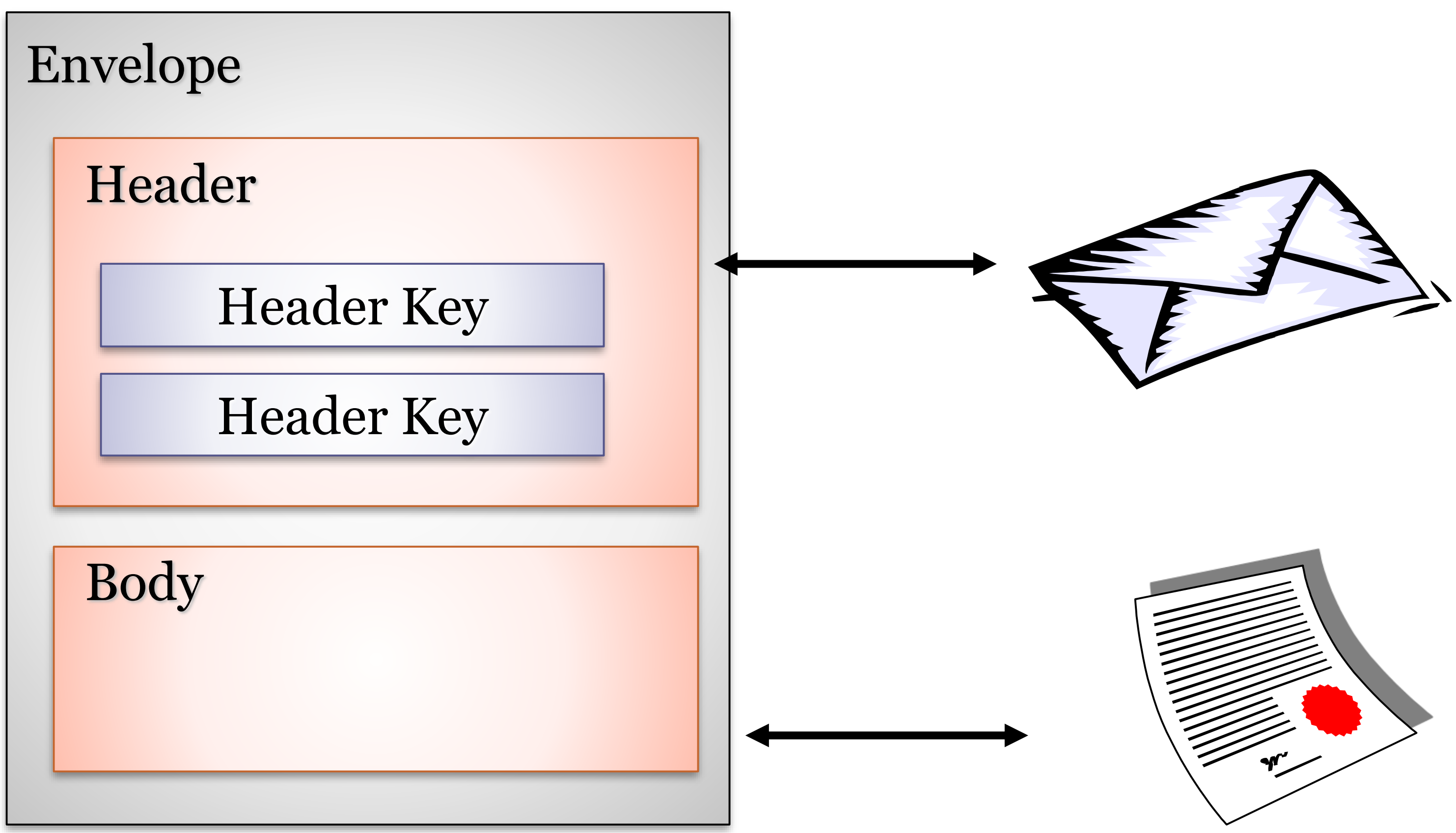
WS-*

SOAP

- Defines messages format and bindings with several protocols
  - Initially *Simple Object Access Protocol*
- Evolution
  - Developed from XML-RPC
  - SOAP 1.0 (1999), 1.1 (2000), 1.2 (2007)
  - Initial development by Microsoft
  - Posterior adoption by IBM, Sun, etc.
  - Good Industrial adoption

WS-*

Message format in SOAP



WS-*

Example of SOAP over HTTP

2001 Year

POST ?

```
POST /Suma/Service1.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: longitod del mensaje
SOAPAction: "http://tempuri.org/suma"
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
     xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
  <sum xmlns="http://tempuri.org/">
     <a>3</a>
     <b>2</b>
  </sum>
 </soap:Body>
</soap:Envelope>
```

WS-*

Advantages

- Specifications developed by community
  - W3c, OASIS, etc.
- Industrial adoption
  - Implementations
- Integral view of web services
- Numerous extensions
  - Security, orchestration, choreography, etc.

Challenges

- Not all specifications were mature
  - Over-specification
  - Lack of implementations
- RPC style abuse
  - Uniform interface
  - Sometimes, bad use of HTTP architecture
    - Overload of GET/POST methods

WS-*

Applications

- Lots of applications have been using SOAP
  - Example: eBay (50mill. SOAP transactions/day)
- But…some popular web services ceased to offer SOAP support
  - Examples: Amazon, Google, etc.

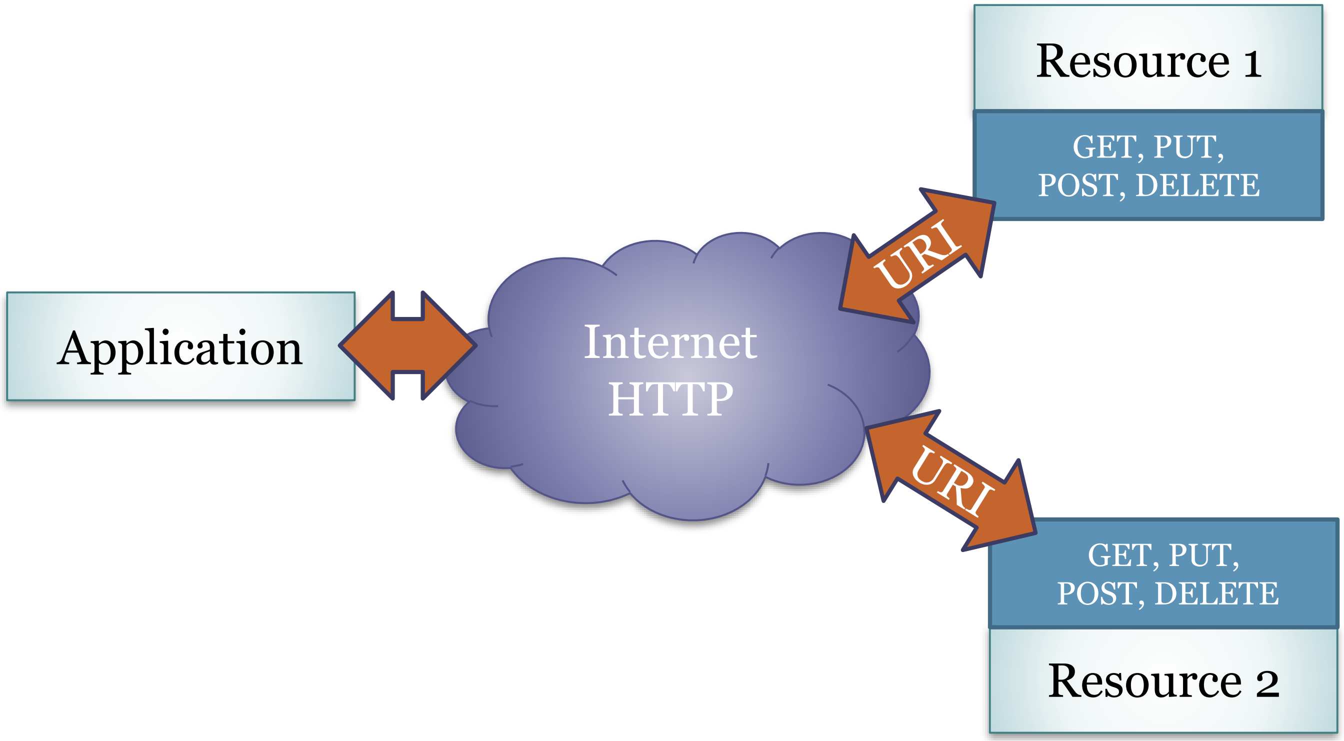
REST

- REST = REpresentational State Transfer
  - Architectural style
  - Source: Roy T Fielding PhD dissertation (2000)
  - Inspired by Web architecture (HTTP/1.1)

REST

REST - Representational State Transfer

Diagram



REST

- Set of constraints
  - Resources with uniform interface
    - Identified by URIs
    - Fixed set of actions: GET, PUT, POST, DELETE
  - Resource representations are returned
  - Stateless
- REST = Architectural style
  - Some levels of adoption:
    - RESTful
    - REST-RPC hybrid

REST as a composed style

- Layers
- Client-Server
  - Stateless
  - Cached
  - Replicated server
- Uniform interface
  - Resource identifiers (URIs)
  - Auto-descriptive messages (MIME types)
  - Links to other resources (HATEOAS)
- Code on demand (optional)

REST uniform interface

Fixed set of operations

GET, PUT, POST, DELETE

Method	In databases	Function	Safe?	Idempotent?
PUT	≈Create/Update	Create/update	No	Yes
POST	≈Update	Create/ Update children	No	No
GET	Retrieve	Query resource info	Yes	Yes
DELETE	Delete	Delete resource	No	Yes

Safe = Does not modify server data
Idempotent = The effect of executing N-times is the same as executing it once

REST

- Stateless client/server protocol
- State handled by client
  - HATEOAS *(Hypermedia As The Engine of Application State)*
    - Representations return URIs to available options
    - Chaining of resource requests

Example: Student management

1.- Get list of students

```
GET http://example.org/student
```

Returns list of students with each student URI

2.- Get information about an specific student

```
GET http://example.org/student/id2324
```

3.- Update information of an specific student

```
PUT http://example.org/student/id2324
```

REST

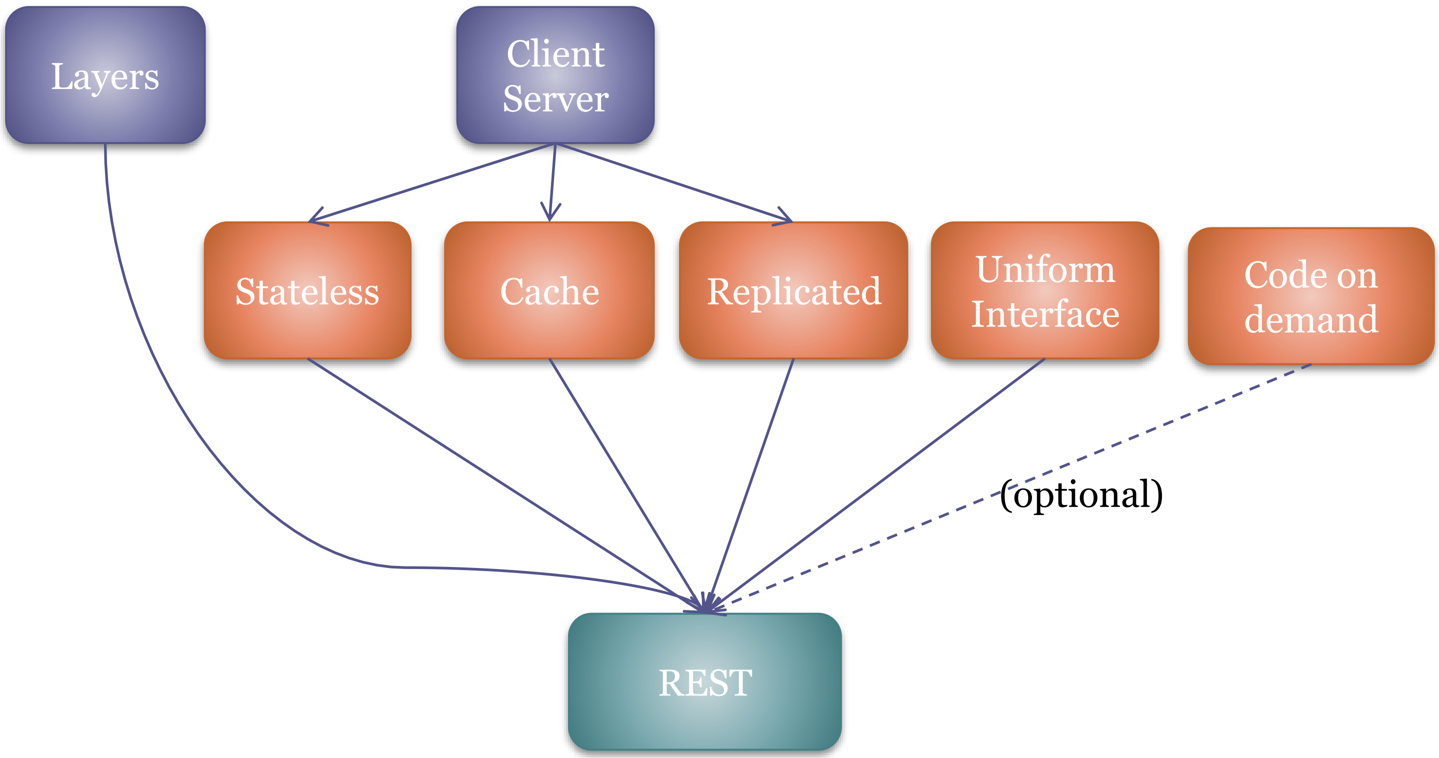
Advantages

- Client/Server
  - Separation of concerns
  - Low coupling
- Uniform interface
  - Facilitates comprehension
  - Independent development
- Scalability
- Improves answer times
- Less network load (cached)
  - Less bandwidth

Challenges

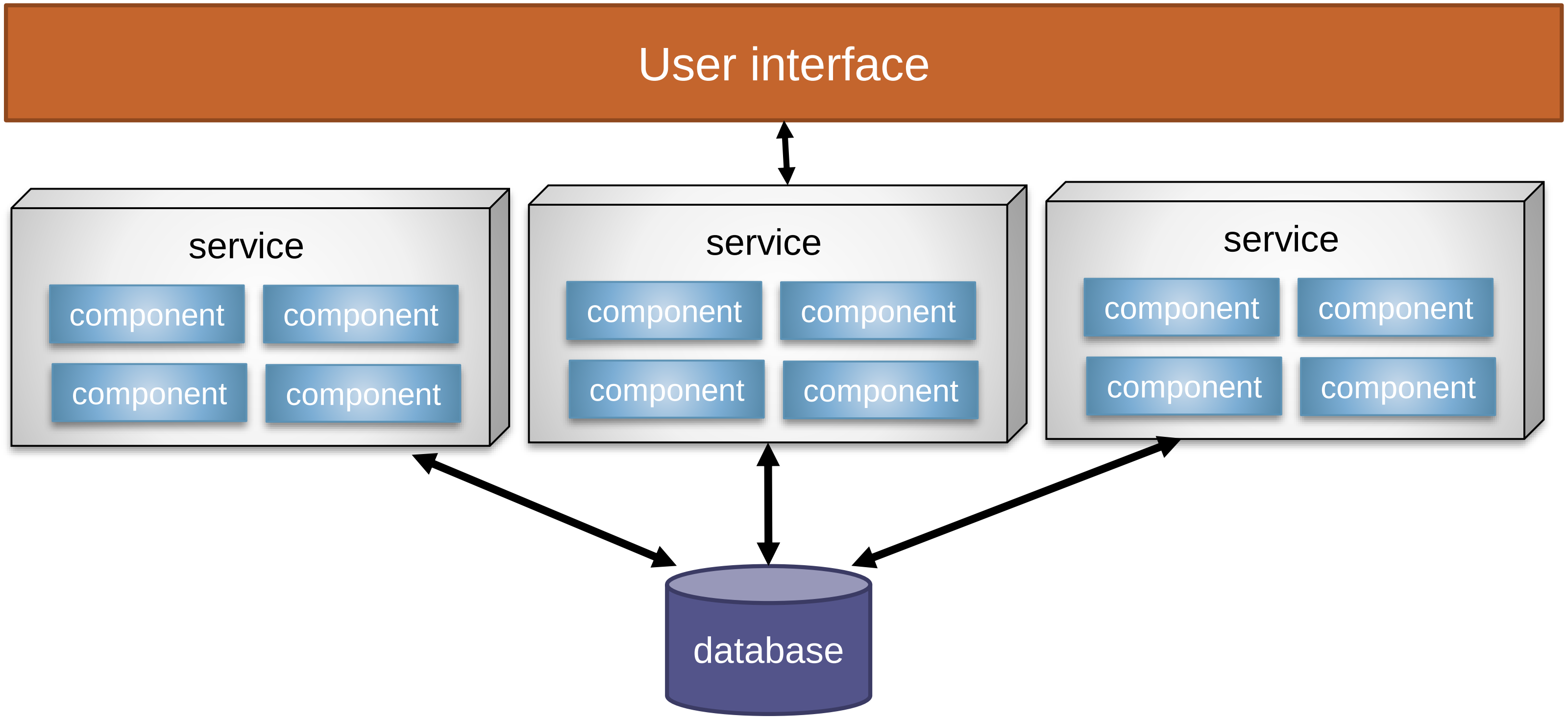
- REST partially adopted
  - Just using JSON or XML
    - Web services without contract or description
  - RPC style REST
- Difficult to incorporate other requirements
  - Security, transaction, composition, etc.

REST as a composed style



Service based architecture

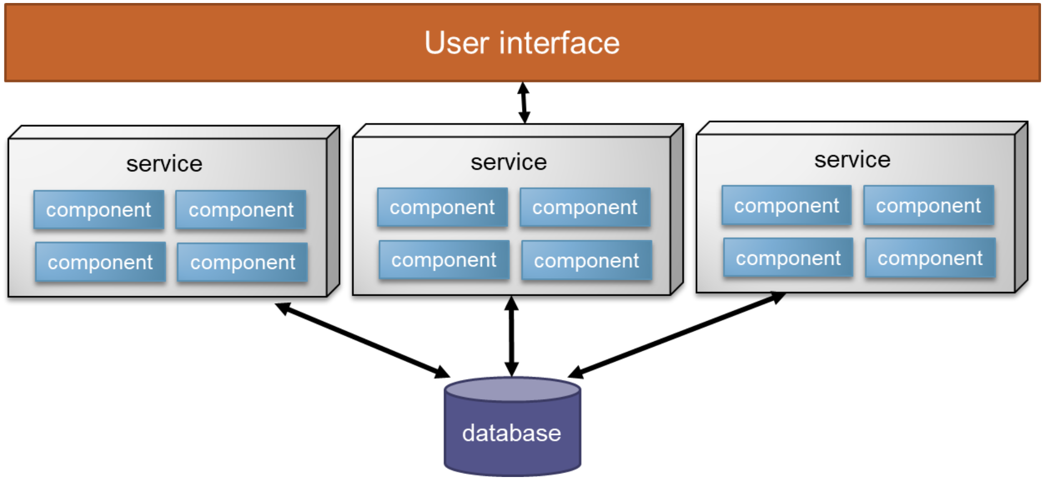
Pragmatic architectural style based on SOA



Service based architecture

Elements

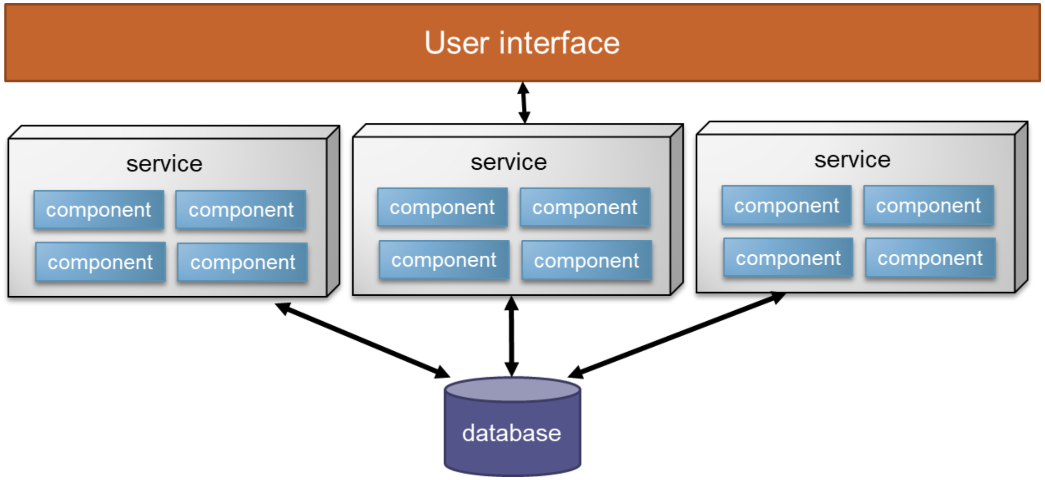
- Services = independently deployed units
  - Usually composed of different components
- User interface accesses services remotely (Internet)
- Database shared by those services



Service based architecture

Constraints

- Each service is independently deployed
- Services are usually coarse grained
- User interface can be divided (different topologies)
- Database is usually shared by each service



Service based architecture

Advantages

- Modularity of development
  - Services can be independently developed
- Technology diversity
  - Each service can be developed using a different programming language & technology
- Time to market
  - Several frameworks
- Availability
- Reliability

Challenges

- Scalability (database partitioning)
- Evolution of services
  - Adaption to change is usually difficult
  - Services can be monoliths
- Conway's law
  - Database team
  - User interface team
  - Programmers

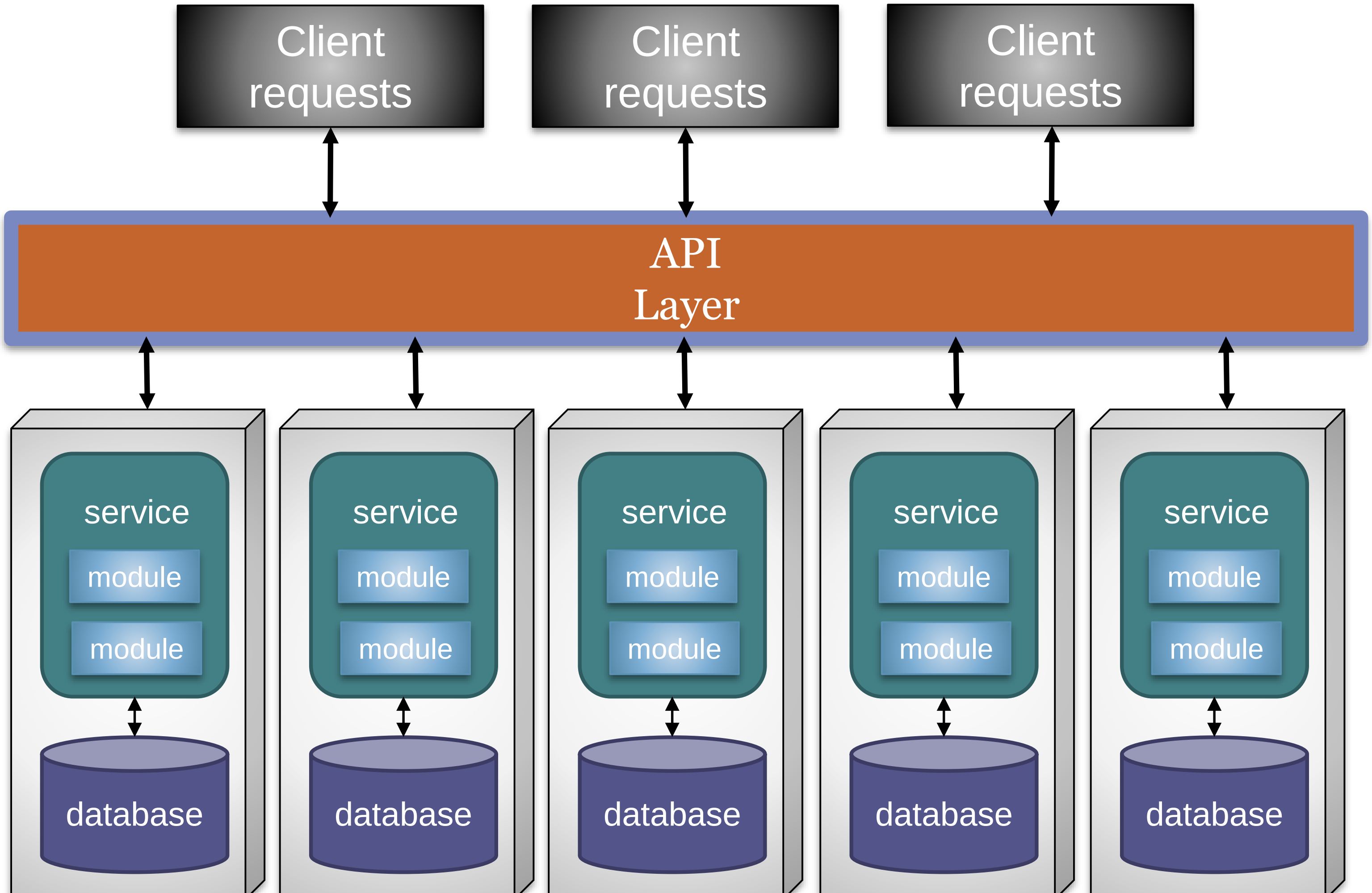
Microservices

- Applications divided in small components called microservices
- Each microservice = small building block
  - Highly uncoupled
  - Focus on a specific task
- Difference with SOA
  - In SOA, services are in different applications
  - Microservices belong to the same application

http://martinfowler.com/articles/microservices.html

Microservices

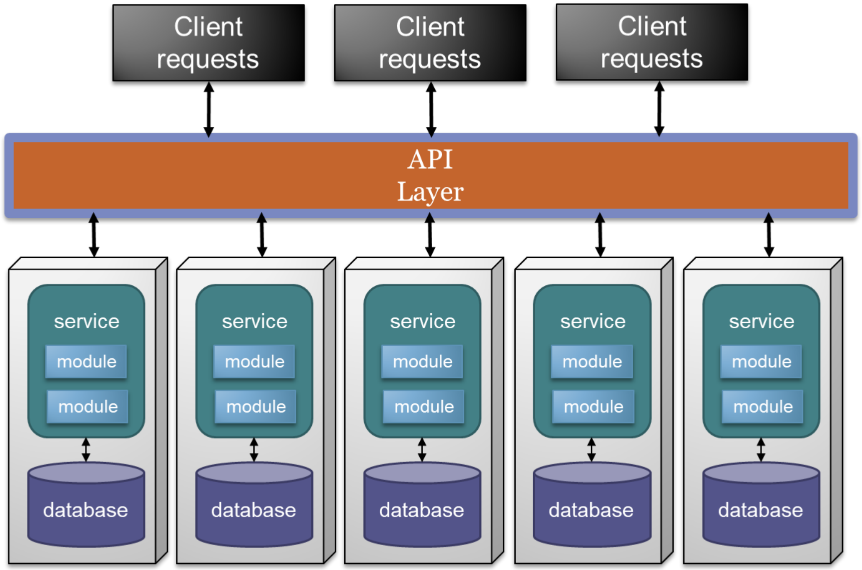
Diagram



Microservices

Elements

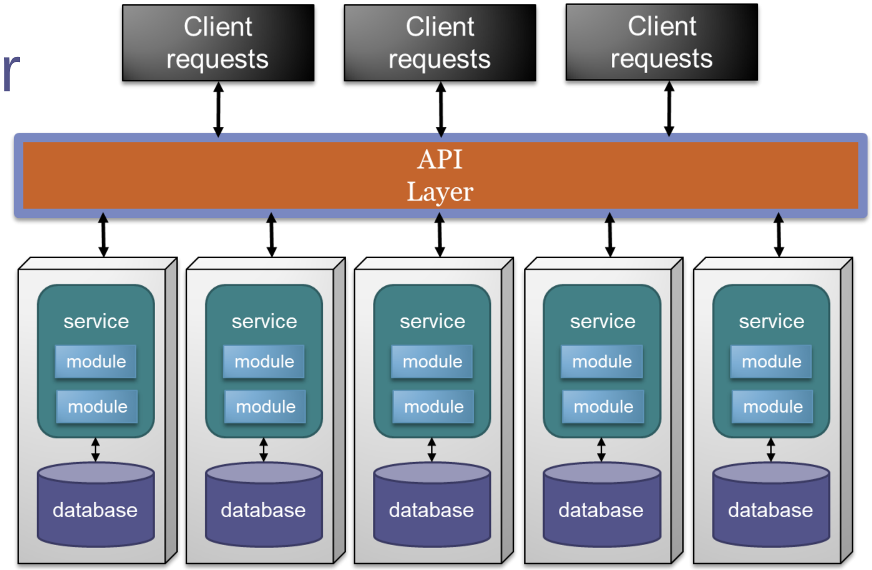
- A service + database form a deployed component
- A service contains several modules and its own database
- API layer (optional) offers a proxy or naming service



Microservices

Constraints

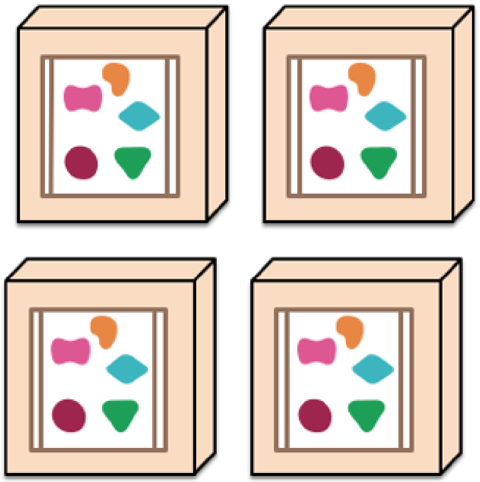
- Distributed
- Bounded context:
  - Each service models a domain or workflow
- Data isolation
- Independency:
  - No mediator or orchestrator



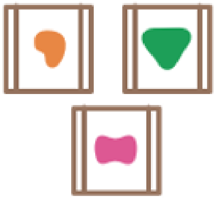
Microservices & scalability

Monolithic: all functionality in a single process

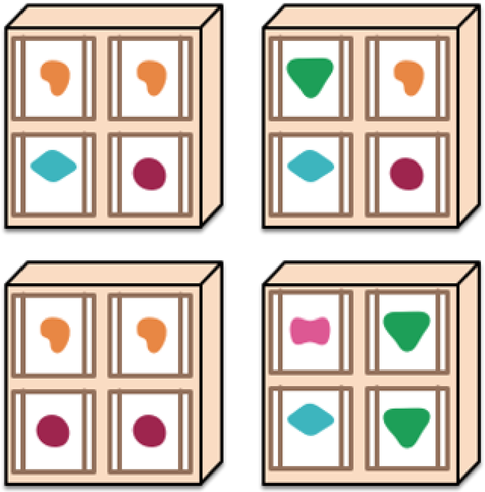
...scales replicating the monolith on multiple services



Microservices: each element of functionality into a separate service

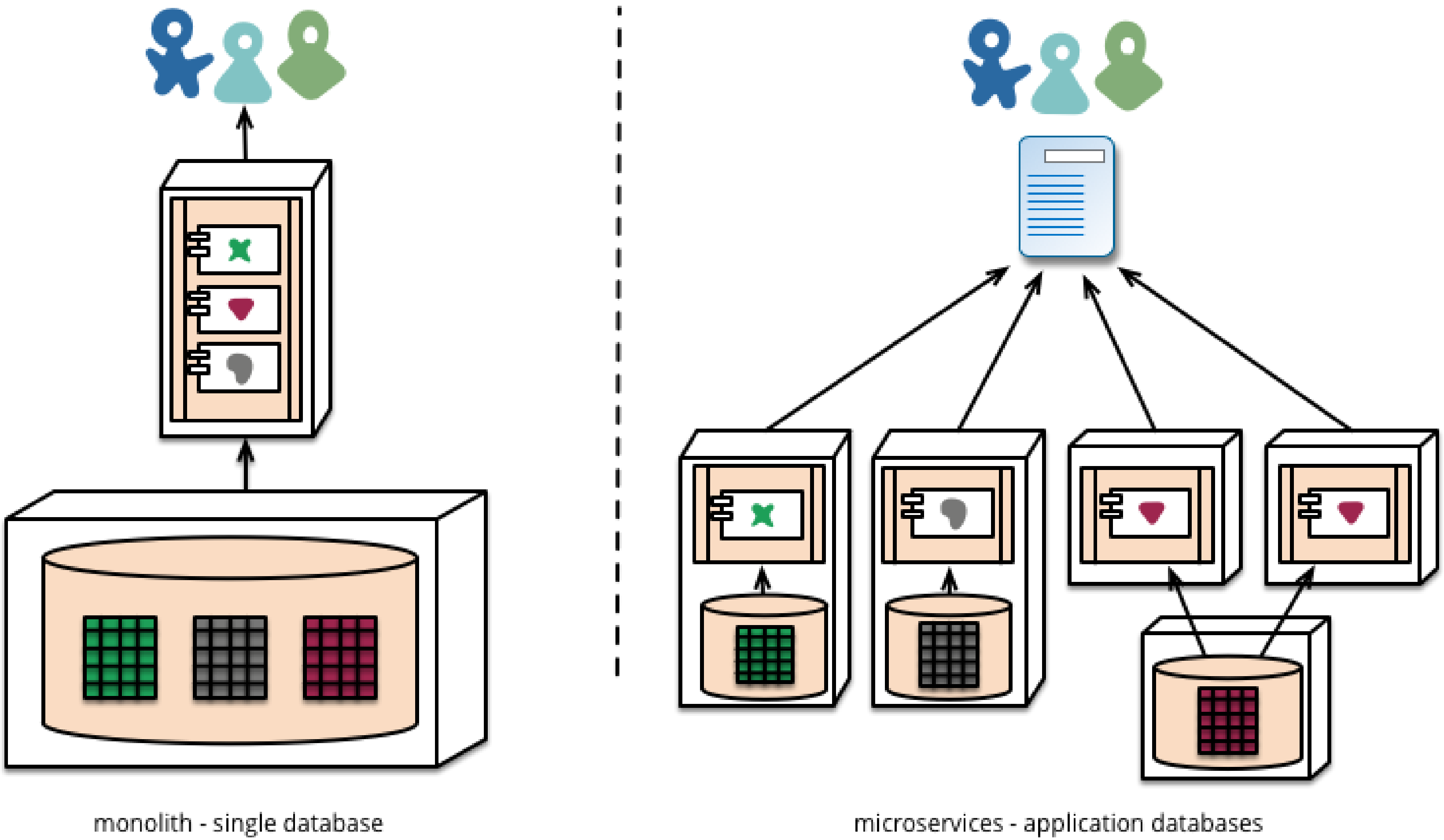


... scales distributing these services, replicating as needed



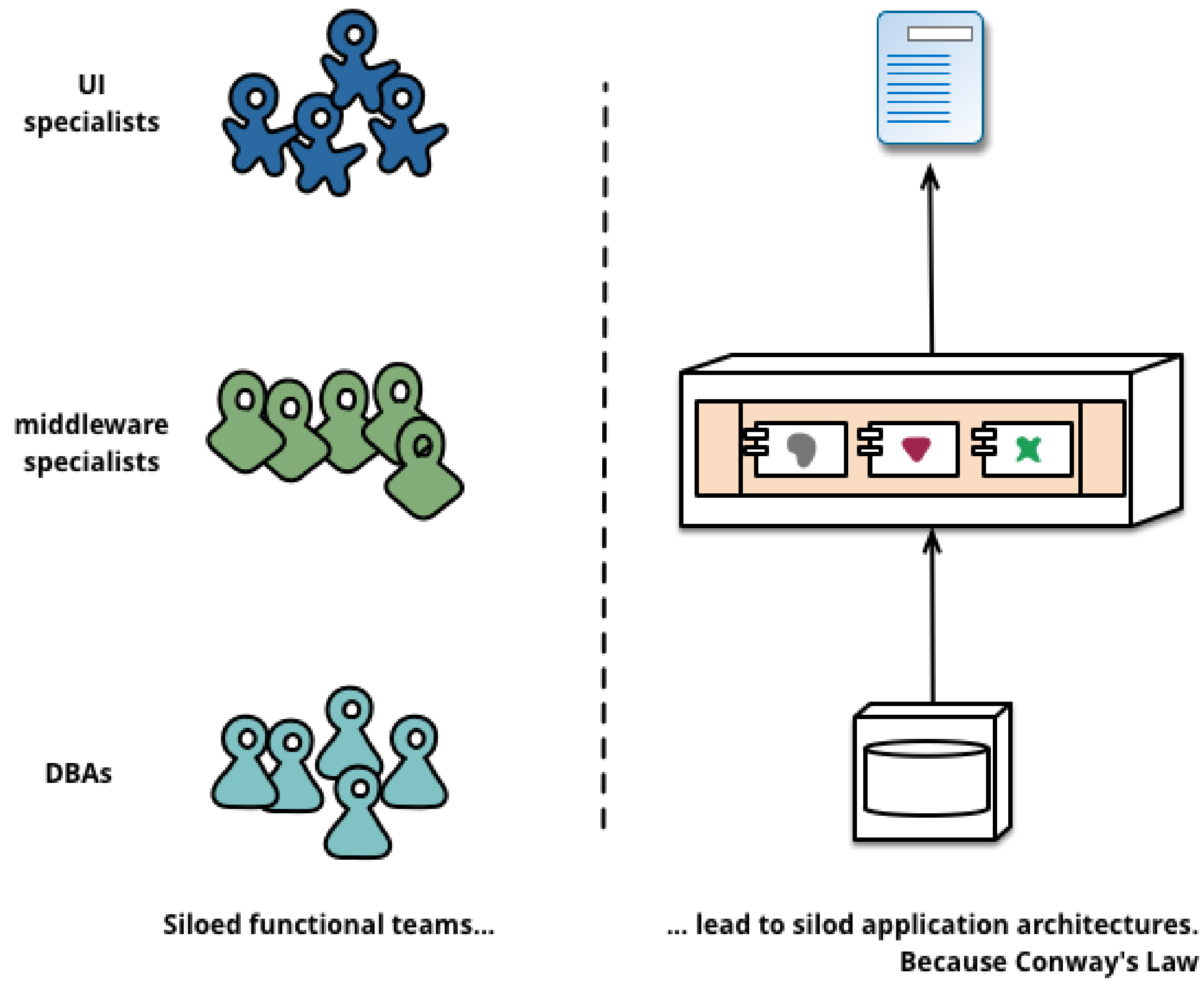
Microservices

Decentralized data management



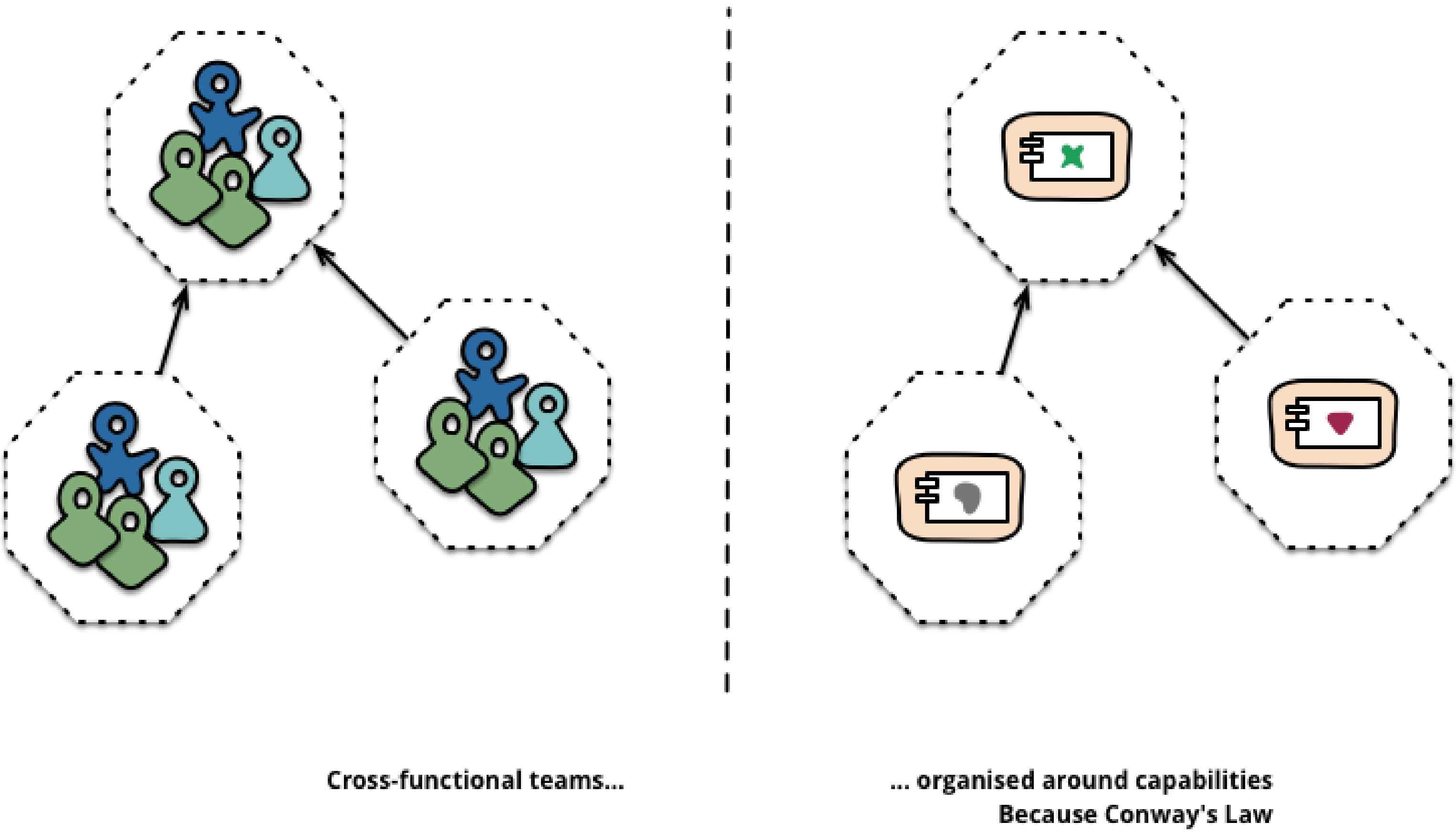
Microservices

Conway Law (traditional application)



Microservices

Conway Law (microservices): Teams are decomposed around capabilities



Microservices

Advantages

- Strong Modularity of development
  - Microservices reusability
- Independent development and deployment
  - Scalability
  - Decentralization
- Technology diversity
  - Each service can be developed using a different programming language & technology

Challenges

- Managing lots of microservices
  - Too much microservices = antipattern (nanoservices)
  - Ensure application consistency
- Complexity
  - Distributed system management
  - New challenges: latency, message format, load balance, fault tolerance, etc.
- Testing & deployment
  - Operational complexity
- Structural decay

http://martinfowler.com/articles/microservice-trade-offs.html

Microservices structural decay

- Code dependencies between services
  - Too much shared libraries
- Too much interservice communication
- Too many orchestration requests
- Database coupling

Analyzing architecture (microservices)
https://www.youtube.com/watch?v=U7s7Hb6GZCU

Microservices

Variants

Self contained Systems (SCS) Architecture

- Separation of functionality into many independent systems
- https://scs-architecture.org/
- Each SCS contains logic and data

Serverless

- Also known as:
  - Function as a service (FaaS)
  - Backend as a service (BaaS)
- Applications depend on third-party services
- Developers don't need to care about servers
  - Automatic scalability
- Rich clients
  - *Single Page Applications, Mobile apps*
- Examples:
  - AWS Lambda, Google Cloud Functions, Ms Azure Functions

https://en.wikipedia.org/wiki/Serverless_computing

Serverless

Advantages

- Scalability
- Availability
- Performance
- Reduce costs
  - Operational cost
  - Only pay for the compute you need
- Time to market

Challenges

- Vendor control
- Vendor lock-in
  - Incompatibility between vendors
- Security
- Startup latency
- Integration testing
- Monitoring/debugging

End of presentation