Universidad de Oviedo

ES
Español

Escuela de Ingeniería Informática

Integración

Curso 2018/2019

Jose Emilio Labra Gayo

Integración

Integración de aplicaciones

Gran reto de la informática

Integración

- Estilos de integración
  - Transferencia de ficheros
  - Base de datos compartida
  - Invocación Procedimiento Remotos
  - Mensajería
  - Event log
- Topologías
  - Hub & Spoke, Bus
- Arquitecturas orientadas a servicios
  - WS-*, REST
- Microservicios
- Serverless

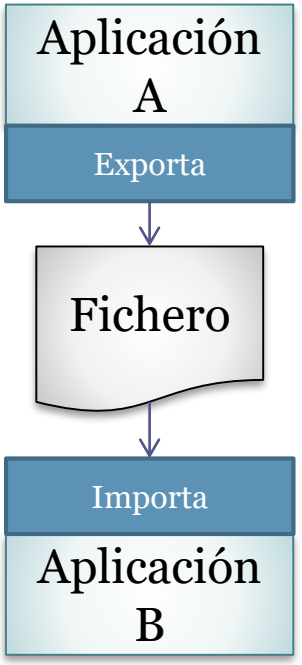
Estilos de integración

Transferencia de ficheros
Base de datos compartida
Invocación procedimiento remoto
Mensajería
Event log

Transferencia de ficheros

Una aplicación genera un fichero de datos que es consumido por otra

Una de las soluciones más comunes



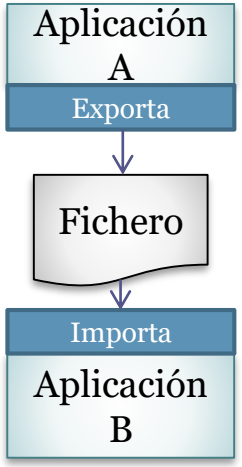
Transferencia de ficheros

Ventajas

- Bajo acoplamiento
  - Independencia entre aplicación A y B
- Facilita depuración
  - Se pueden analizar datos del fichero

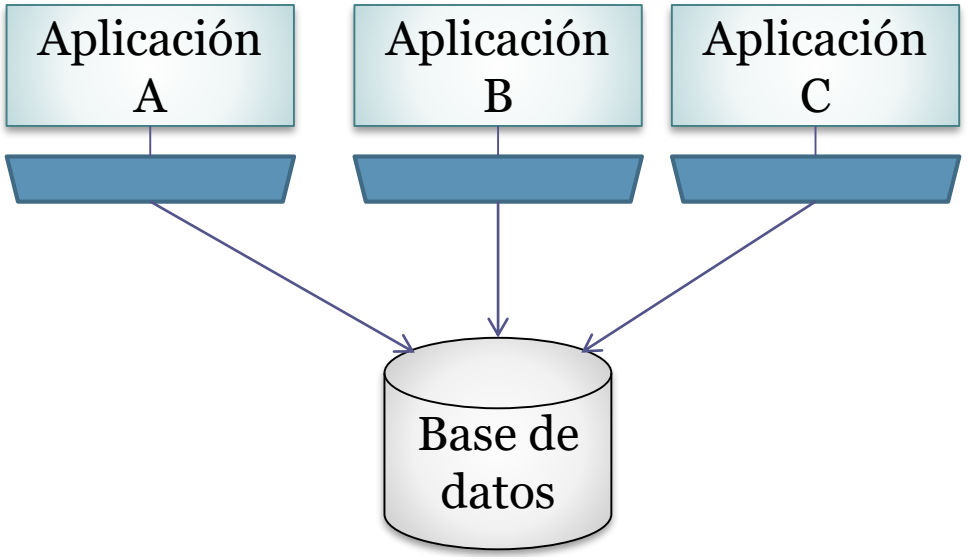
Problemas

- Acordar formato de fichero común
  - Puede aumentar acoplamiento
- Coordinación
  - Una vez enviado el fichero, la aplicación B puede modificarlo ⇒ ¡2 ficheros!
- Suele requerir intervención manual



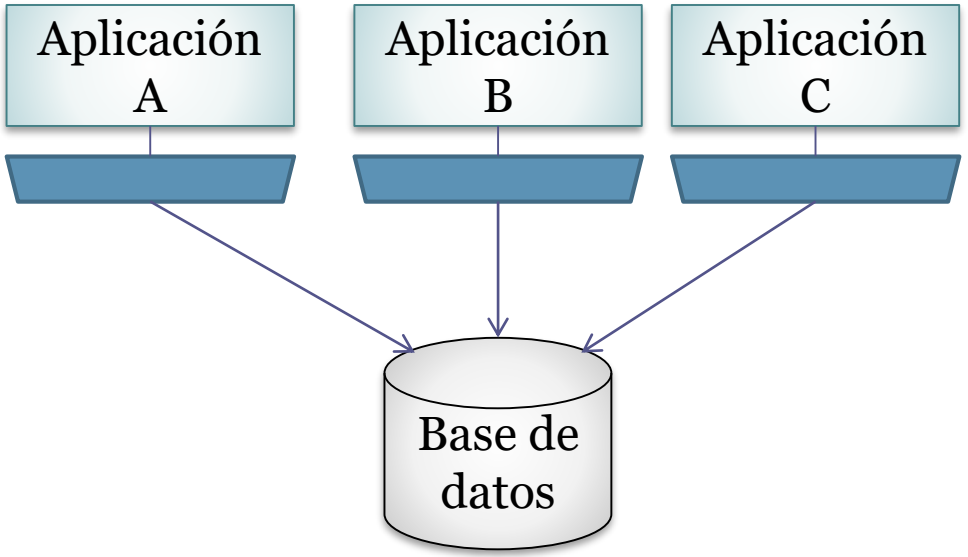
Base de datos compartida

Las aplicaciones almacenan sus datos en una base de datos común



Base de datos compartida

- Ventajas
  - Datos siempre disponibles
  - Todo el mundo accede a la misma información
  - Consistencia
  - Formato familiar
    - SQL para todo?



Base de datos compartida

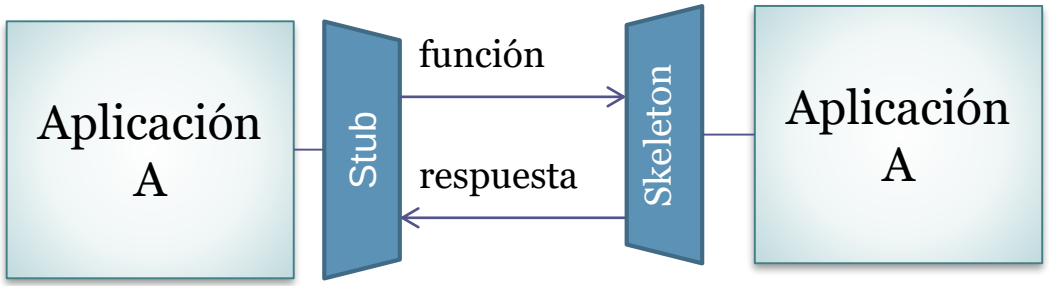
- Problemas
  - El esquema de la base de datos puede variar
    - Requiere esquema común para todas las aplicaciones
      - Fuente de problemas y conflictos
    - Necesidad de paquetes externos (acceso BD común)
  - Rendimiento y escalabilidad
    - Base de datos como cuello de botella
  - Sincronización
    - Problema con bases de datos distribuidas
    - Escalabilidad
    - NoSQL ?

Base de datos compartida

- Variaciones
  - *Data warehousing*: Base de datos utilizada para análisis de datos e informes
  - ETL: proceso basado en tres fases
    - Extracción: Obtención de fuentes heterogéneas
    - Transformación: Procesado de los datos
    - Carga (Load): Almacenamiento en base de datos compartida

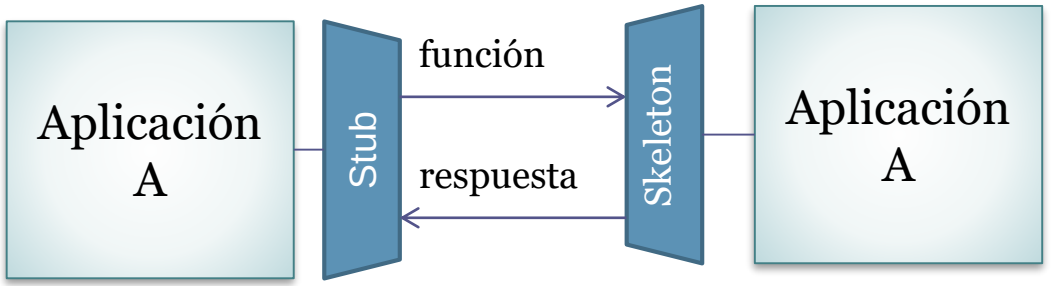
Invocación procedimiento remoto

- Una aplicación invoca una función de otra aplicación que puede estar en otra máquina
  - En la invocación puede pasar parámetros
  - Obtiene una respuesta
- Gran variedad de aplicaciones
  - RPC, RMI, CORBA, .Net Remoting, ...
  - Servicios web, ...



Invocación procedimiento remoto

- Ventajas
  - Encapsulación de implementación
  - Múltiples interfaces para la misma información
    - Se pueden ofrecer distintas representaciones
  - Modelo familiar para desarrolladores
    - Similar a llamar a un método



Invocación procedimiento remoto

Problemas

Falsa sensación de sencillez

Procedimiento remoto ≠ Procedimiento

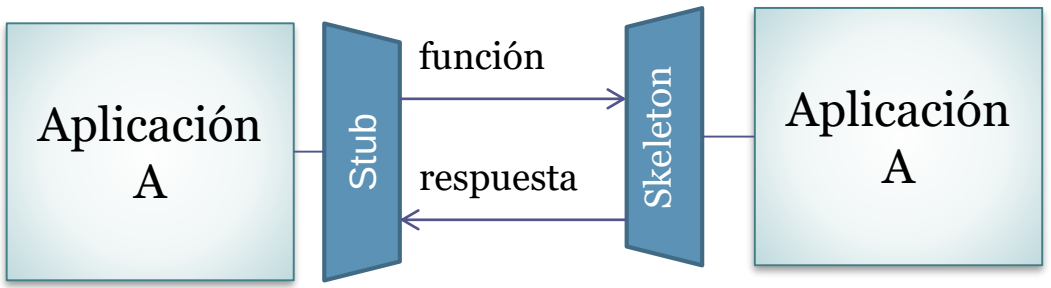
8 falacias de computación distribuida

Invocaciones mediante sincronización

Aumenta acoplamiento entre aplicaciones

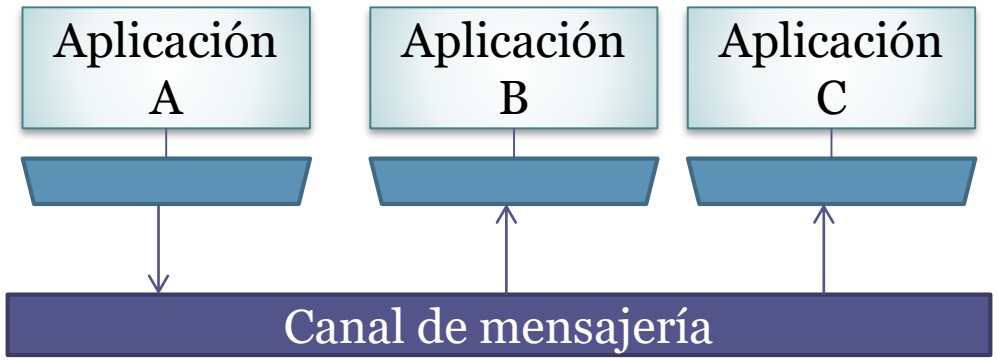
La red es fiable
La latencia es cero
El ancho de banda es infinito
La red es segura
La topología no cambia
Hay un administrador
El coste de transporte es cero
La red es homogénea

8 falacias computación distribuida



Mensajería

- Múltiples aplicaciones independientes se comunican enviando mensajes a un canal
- Comunicación asíncrona
  - Las aplicaciones envían mensajes y continúan ejecutándose



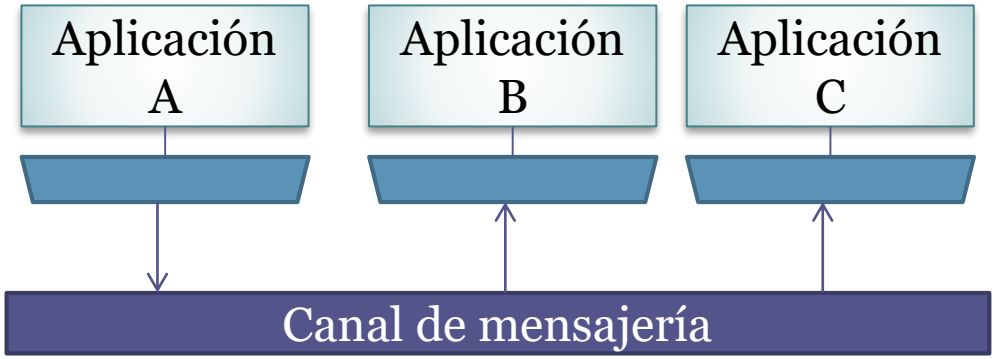
Mensajería

Ventajas

- Bajo acoplamiento
  - Aplicaciones independientes entre sí
- Comunicación asíncrona
  - Las aplicaciones continúan la ejecución
- Encapsulación
  - Sólo se expone el tipo de mensajes

Problemas

- Complejidad de implementación
  - Comunicación asíncrona
- Transformación de datos
  - Adaptación formato de mensajes
- Diferentes topologías



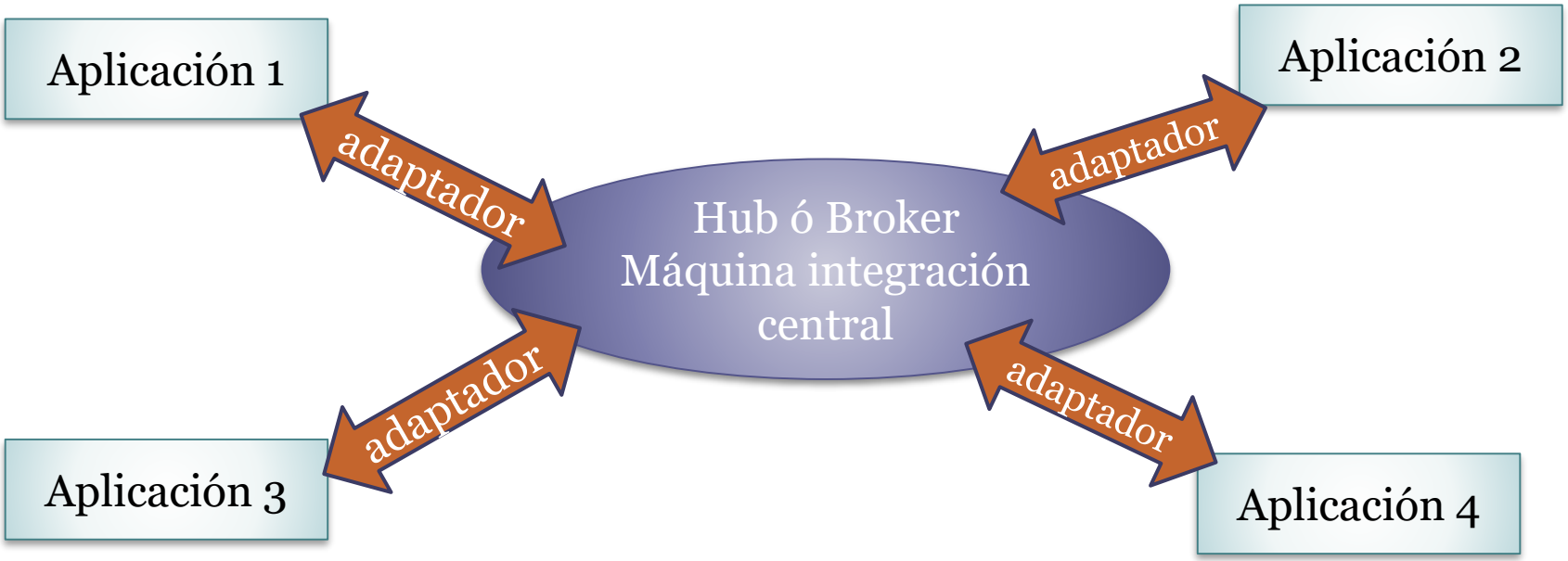
Topologías de integración

Hub & Spoke

Bus

Hub & Spoke (central/radial)

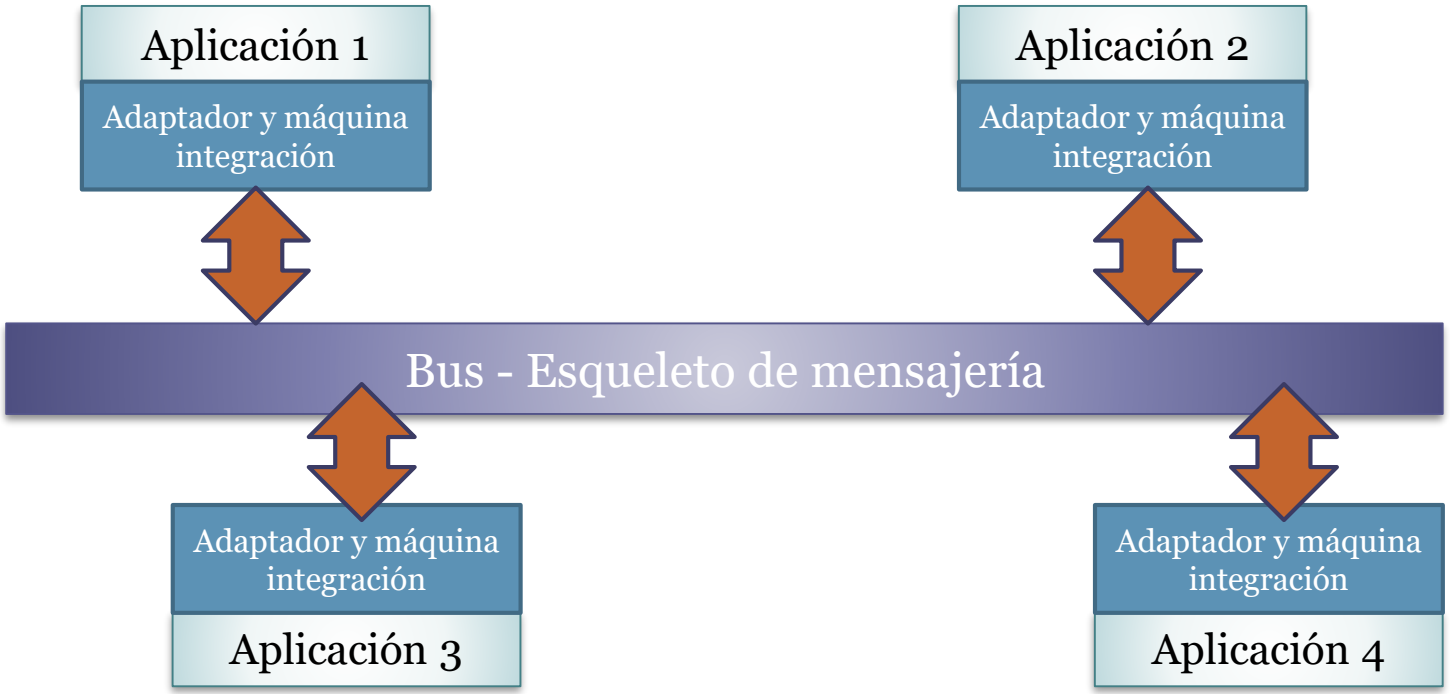
Relacionado con patrón Bróker
Hub = Bróker centralizado de mensajes
- Se encarga de la integración



Bus

Cada aplicación contiene su máquina de integración

Estilo Publish/Subscribe



Bus

- ESB - Enterprise Service Bus
- Define un esqueleto (*backbone*) de mensajería
  - Conversión de protocolos
  - Transformación de formatos
  - Enrutamiento
- Proporciona un API para desarrollar servicios
  - MOM (Message Oriented Middleware)

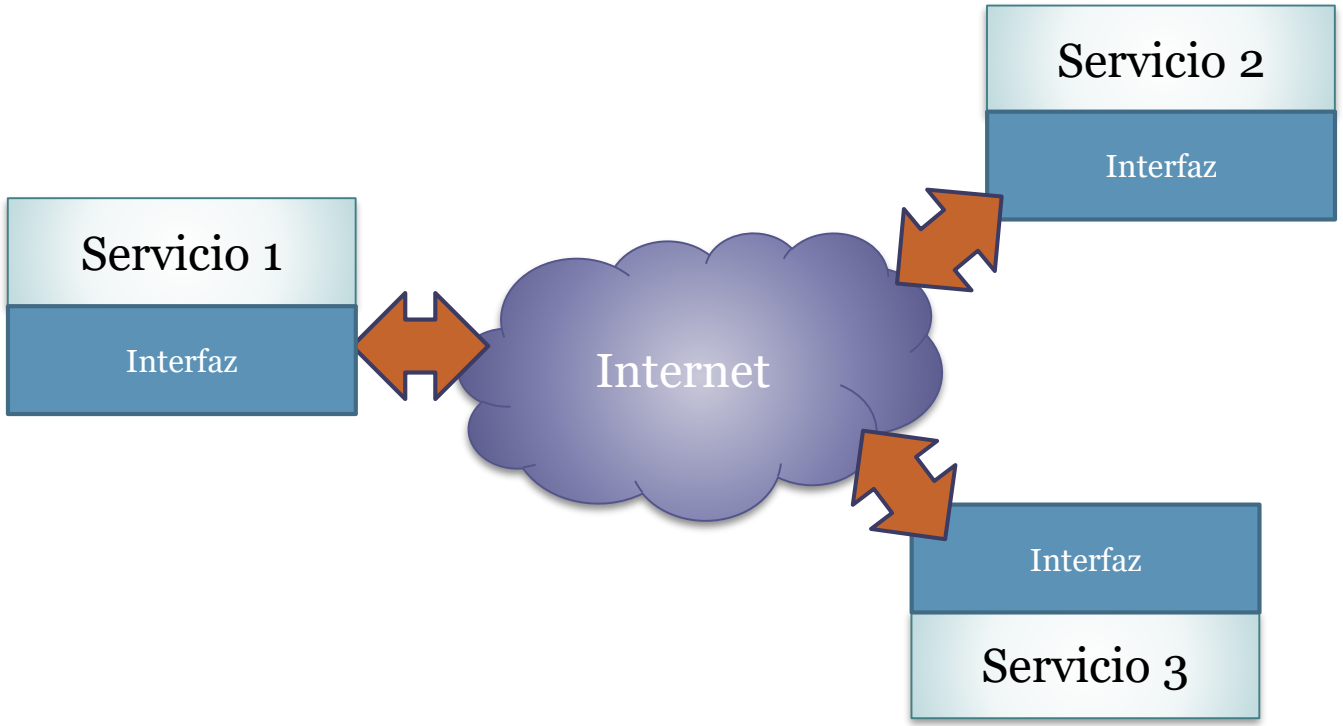
Servicios

- SOA - Service Oriented Architectures
  - WS-*
  - REST
- Microservicios
- Serverless

SOA

SOA = Service Oriented Architecture

Los servicios están definidos mediante un interfaz



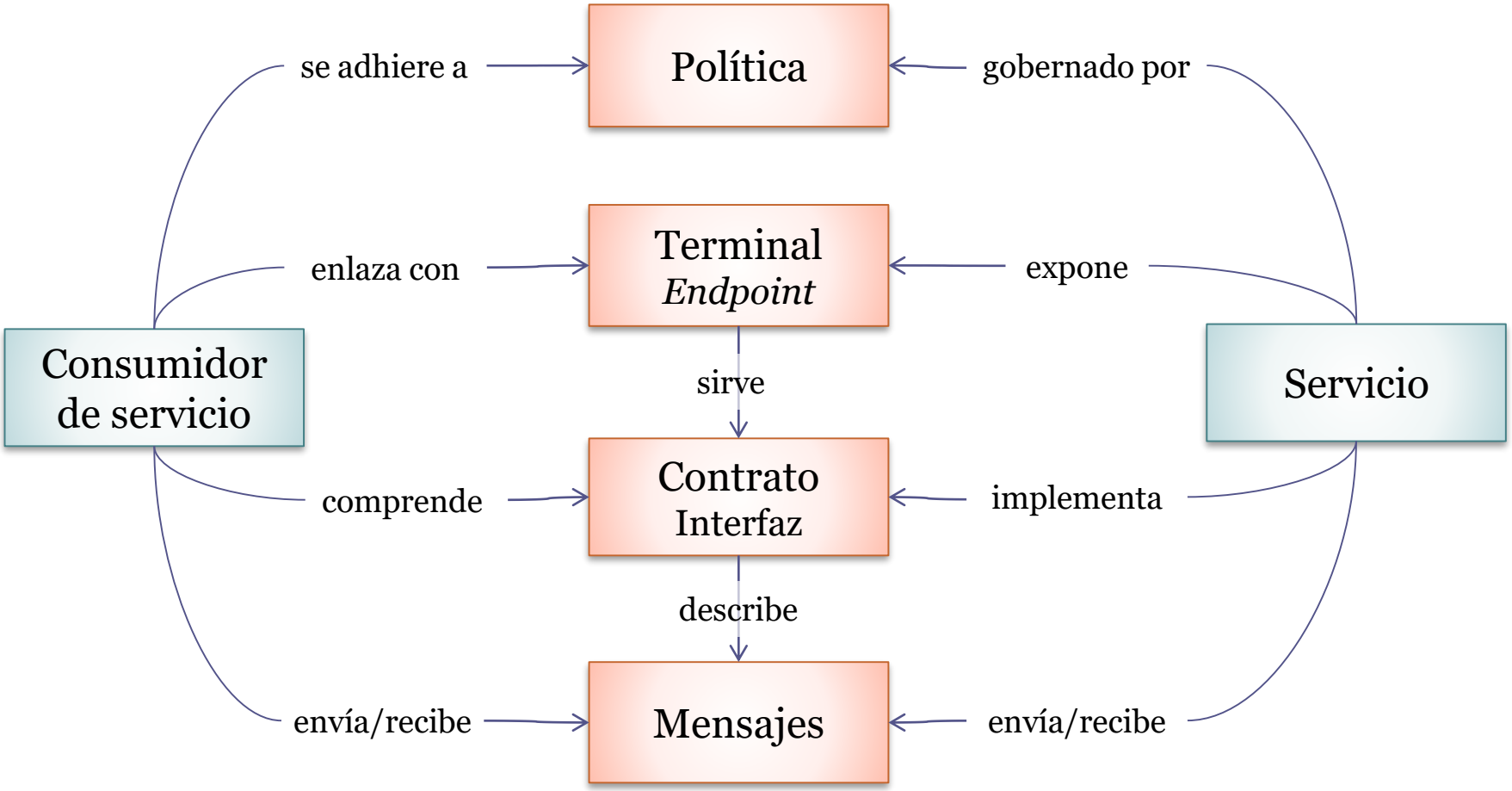
SOA

Elementos

- Proveedor : Proporciona el servicio
- Consumidor: Realiza peticiones al servicio
- Mensajes: Información intercambiada
- Contrato o interfaz: Descripción de la funcioanlidad ofrecida por el servicio
- Terminal: Ubicación del servicio
- Política: Acuerdos de gobierno del servicio
  - Seguridad, rendimiento, etc.

SOA

Restriccciones



Fuente: SOA Patterns, A. Rottem Gal Oz

SOA

Ventajas

- Independencia de lenguaje y plataforma
- Interoperabilidad
  - Utilización de estándares
- Acoplamiento débil
- Descentralizado
- Reusabilidad
- Escalabilidad
  - Comunicación uno-a-uno frente uno-a-muchos
- Mantenimiento sistemas *legacy*
  - Añadir una capa de servicios web

Problemas

- Eficiencia.
  - Puede no ser necesario en:
    - Entornos muy homogéneos, tiempo real, ...
- Exposición abierta
  - Riesgo de exponer API al exterior
- Seguridad
- Composición de servicios

SOA

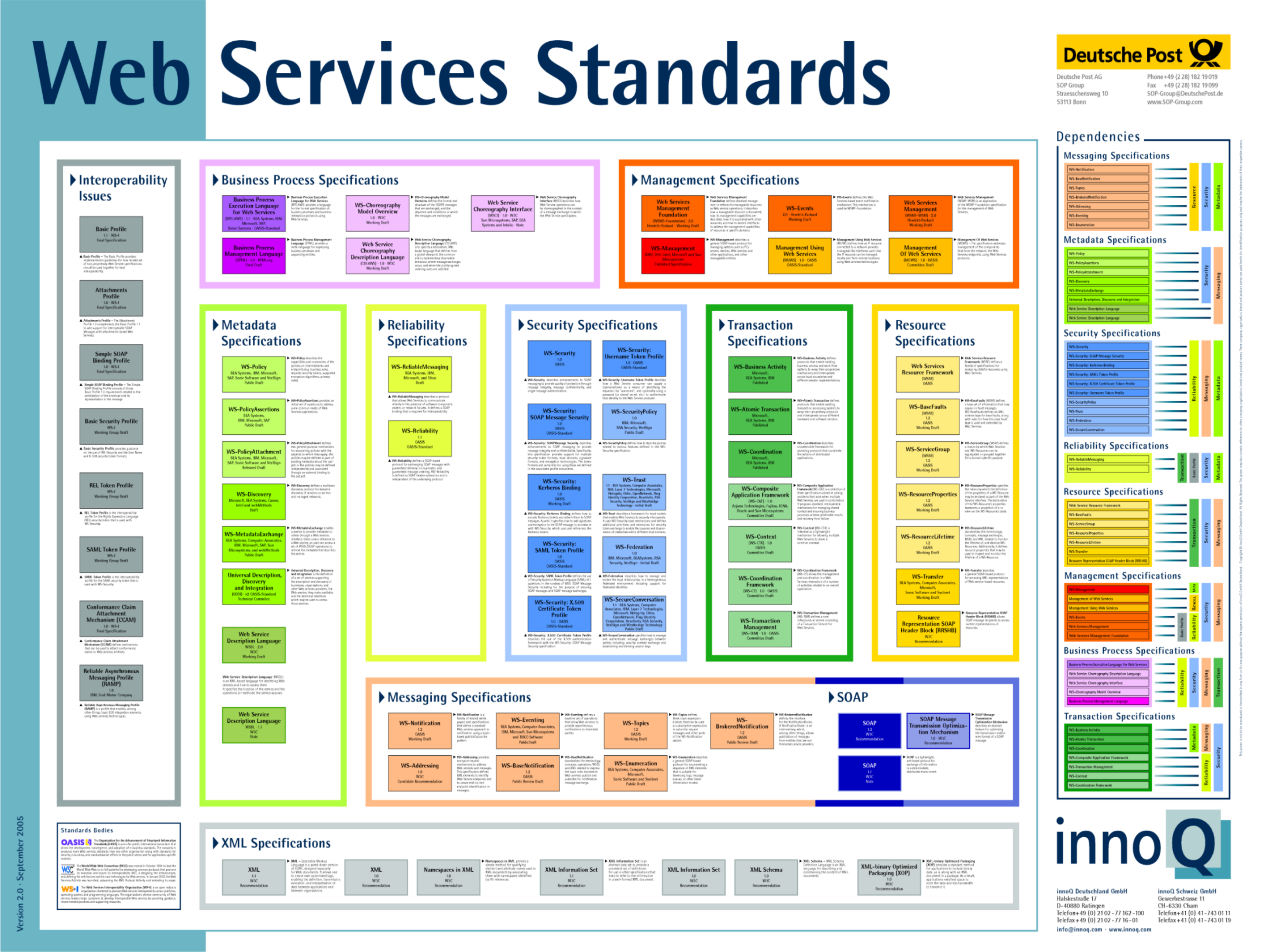
Variantes:

- WS-*
- REST

WS-*

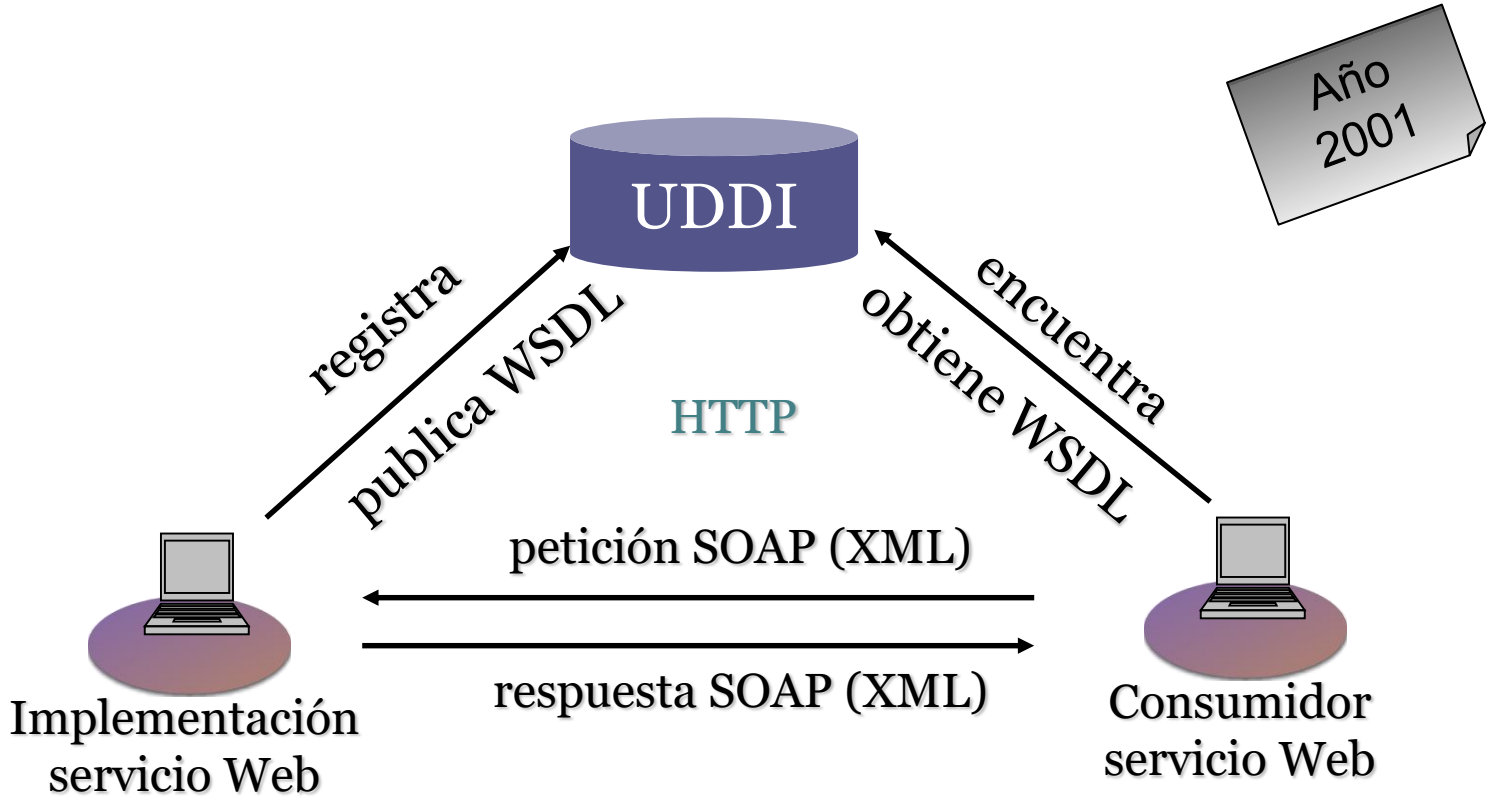
Modelo WS-* = Conjunto de especificaciones

- SOAP, WSDL, UDDI, etc....
- Propuesto por W3c, OASIS, WS-I, etc.
- Objetivo: Implementación SOA de referencia



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
Attachments Profile 1.0 · WS-I Final Specification
Simple SOAP Binding Profile 1.0 · WS-I Final Specification
Basic Security Profile WS-I Working Group Draft
REL Token Profile WS-I Working Group Draft
SAML Token Profile WS-I Working Group Draft
Conformance Claim Attachment Mechanism (CCAM) 1.0 · WS-I Final Specification
Reliable Asynchronous Messaging Profile (RAMP) 1.0 IBM, Ford Motor Company
Business Process Specifications
Business Process Execution Language for Web Services (BPEL4WS) · 1.1 · BEA Systems, IBM, Microsoft, SAP, Siebel Systems · OASIS-Standard
Business Process Management Language (BPML) · 1.0 · BPMI.org Final Draft
WS-Choreography Model Overview 1.0 · W3C Working Draft
Web Service Choreography Description Language (CDL4WS) · 1.0 · W3C Working Draft
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-Management AMD, Dell, Intel, Microsoft and Sun Microsystems Published Specification
WS-Events 2.0 · Hewlett-Packard Working Draft
Management Using Web Services (MUWS) · 1.0 · OASIS OASIS-Standard
Web Services Management (WSMF-WSM) · 2.0 Hewlett-Packard Working Draft
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-PolicyAssertions BEA Systems, IBM, Microsoft, SAP Public Draft
WS-PolicyAttachment BEA Systems, IBM, Microsoft, SAP, Sonic Software and VeriSign Released Draft
WS-Discovery Microsoft, BEA Systems, Canon, Intel and webMethods Draft
WS-MetadataExchange BEA Systems, Computer Associates, IBM, Microsoft, SAP, Sun Microsystems, and webMethods Public Draft
Universal Description, Discovery and Integration (UDDI) · v2 OASIS-Standard Technical Committee
Web Service Description Language WSDL · 2.0 W3C Working Draft
Web Service Description Language WSDL · 1.1 W3C Note
Reliability Specifications
WS-ReliableMessaging BEA Systems, IBM, Microsoft, and Tibco Draft
WS-Reliability 1.1 OASIS OASIS-Standard
Security Specifications
WS-Security 1.0 OASIS
WS-Security: Username Token Profile 1.0 · OASIS OASIS-Standard
WS-Security: SOAP Message Security 1.0 OASIS OASIS-Standard
WS-SecurityPolicy 1.0 IBM, Microsoft, RSA Security, VeriSign Public Draft
WS-Security: Kerberos Binding 1.0 OASIS Working Draft
WS-Trust
WS-Security: SAML Token Profile 1.0 OASIS OASIS-Standard
WS-Federation 1.0 IBM, Microsoft, BEA Systems, RSA Security, VeriSign · Initial Draft
WS-Security: X.509 Certificate Token Profile 1.0 · OASIS OASIS-Standard
WS-SecureConversation
Transaction Specifications
WS-Business Activity Microsoft, BEA Systems, IBM Published
WS-Atomic Transaction Microsoft, BEA Systems, IBM Published
WS-Coordination Microsoft, BEA Systems, IBM Published
WS-Composite Application Framework (WS-CAF) · 1.0 Arjuna Technologies, Fujitsu, IONA, Oracle and Sun Microsystems Committee Draft
WS-Context (WS-CTX) · 1.0 OASIS Committee Draft
WS-Coordination Framework (WS-CF) · 1.0 · OASIS Committee Draft
WS-Transaction Management (WS-TXM) · 1.0 · OASIS Committee Draft
Resource Specifications
Web Services Resource Framework (WSRF) OASIS
WS-BaseFaults (WSRF) 1.2 OASIS Working Draft
WS-ServiceGroup (WSRF) 1.2 OASIS Working Draft
WS-ResourceProperties 1.2 OASIS Working Draft
WS-ResourceLifetime 1.2 OASIS Working Draft
WS-Transfer BEA Systems, Computer Associates, Microsoft, Sonic Software and Systinet Working Draft
Resource Representation SOAP Header Block (RRSHB) W3C Recommendation
Messaging Specifications
WS-Notification 1.2 OASIS Working Draft
WS-Eventing BEA Systems, Computer Associates, IBM, Microsoft, Sun Microsystems and TIBCO Software Public Draft
WS-Topics 1.2 OASIS Working Draft
WS-BrokeredNotification 1.2 OASIS Public Review Draft
WS-Addressing 1.0 W3C Candidate Recommendation
WS-BaseNotification 1.2 OASIS Public Review Draft
WS-Enumeration BEA Systems, Computer Associates, Microsoft, Sonic Software and Systinet Public Draft
SOAP
SOAP 1.2 W3C Recommendation
SOAP Message Transmission Optimization Mechanism 1.0 · W3C Recommendation
SOAP 1.1 W3C Note
XML Specifications
XML 1.1 W3C Recommendation
XML 1.0 W3C Recommendation
Namespaces in XML 1.0 W3C Recommendation
XML Information Set 1.1 W3C Recommendation
XML Information Set 1.0 W3C Recommendation
XML Schema 1.0 W3C Recommendation
XML-binary Optimized Packaging (XOP) 1.0 W3C Recommendation
Standards Bodies
OASIS
W3C
WS-I
Dependencies
Messaging Specifications
Metadata Specifications
Security Specifications
Reliability Specifications
Resource Specifications
Management Specifications
Business Process Specifications
Transaction Specifications
innoQ
innoQ Deutschland GmbH
Halskestraße 17
D-40880 Ratingen
Telefon +49 (0) 21 02 - 77 162 - 100
Telefax +49 (0) 21 02 - 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
Version 2.0 · September 2005

WS-*


Año 2001
UDDI
registra
publica WSDL
encuentra
obtiene WSDL
HTTP
petición SOAP (XML)
respuesta SOAP (XML)
Implementación servicio Web
Consumidor servicio Web

WS-*

Ecosistema de servicios Web

WS-*

SOAP

Define el formato de los mensajes y varios enlaces con protocolos

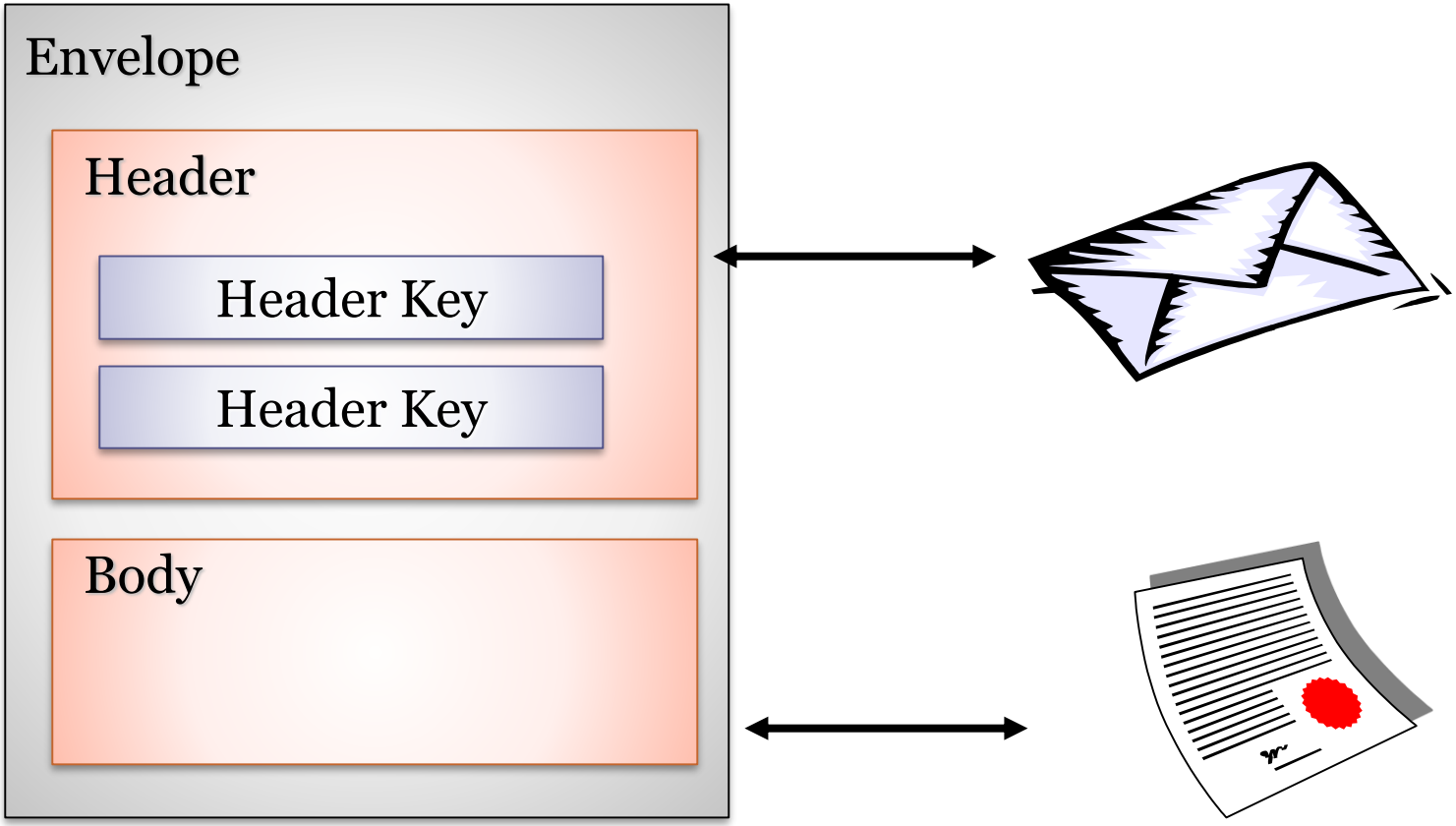
- Originalmente *Simple Object Access Protocol*

Evolución

- Desarrollado a partir de XML-RPC
- SOAP 1.0 (1999), 1.1 (2000), 1.2 (2007)
- Participación inicial de Microsoft
- Adopción posterior de IBM, Sun, etc.
- *Bastante* aceptación industrial

WS-*

Esquema de SOAP



WS-*

Ejemplo de SOAP sobre HTTP

Año 2001

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>
  <suma xmlns="http://tempuri.org/">
     <a>3</a>
     <b>2</b>
  </suma>
 </soap:Body>
</soap:Envelope>
```

WS-*

Ventajas

- Especificaciones realizadas por comunidad
  - W3c, OASIS, etc.
- Adopción industrial
  - Implementaciones
- Visión integral de servicios web
- Numerosas extensiones:
  - Seguridad, orquestación, coreografía, etc.

Problemas

- No todas las especificaciones están maduras
  - Sobre-especificación
  - Falta de implementaciones
- Abuso del estilo RPC
  - Interfaz no uniforme
  - No se sigue arquitectura HTTP
    - Métodos GET/POST sobrecargados

WS-*

- SOAP en la práctica
  - Numerosas aplicaciones utilizan SOAP
    - Ejemplo: eBay (50mill. transacciones SOAP al día)
  - Pero…algunos servicios web populares dejaron de ofrecer soporte SOAP
    - Ejemplos: Amazon, Google, etc.

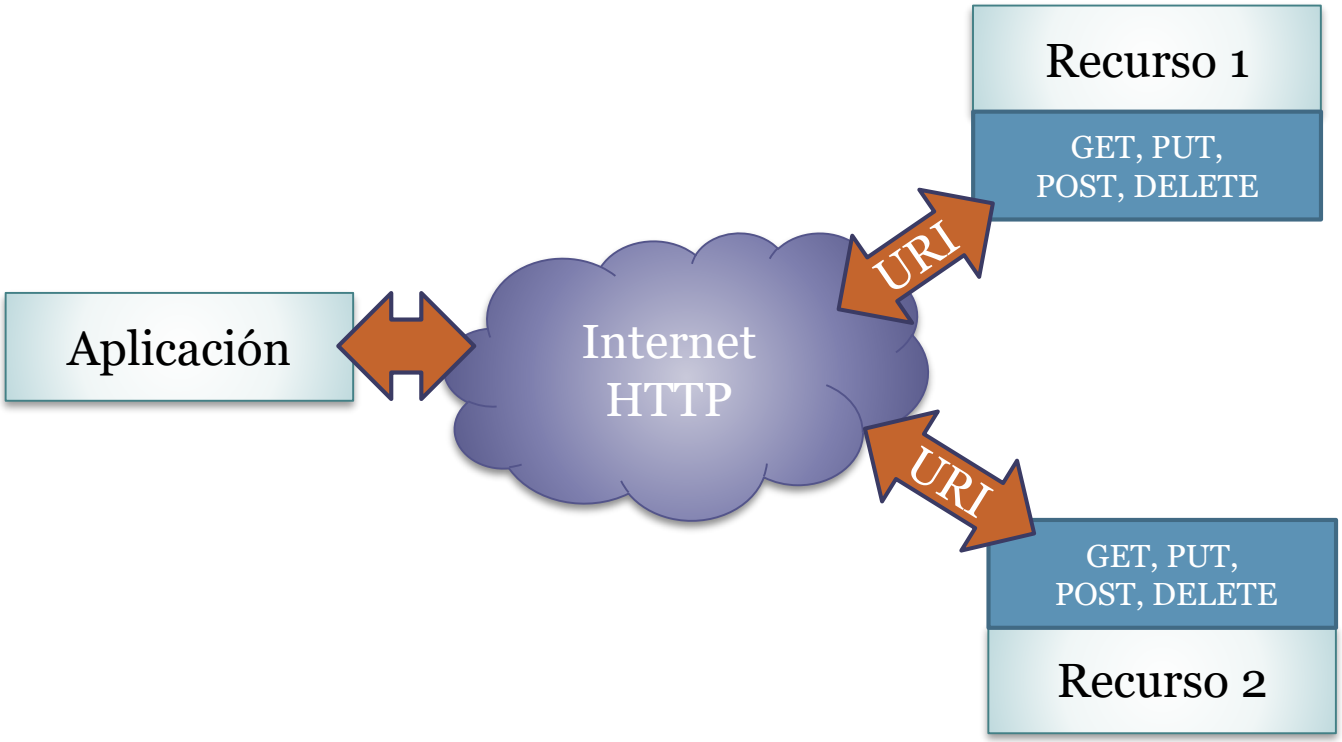
REST

REST = REpresentational State Transfer

- Estilo de arquitectura
- Origen: Tesis doctoral de Roy T Fielding (2000)
- Inspirado en la arquitectura de la Web (HTTP/1.1)

REST

REST - Representational State Transfer
Transferencia de representación de estado



REST

- Conjunto de restricciones
  - Recursos con interfaz uniforme
    - Identificables mediante URIs
  - Se devuelven representaciones de los recursos
  - Sin estado
  - Interfaces genéricos
    - Conjunto acciones: GET, PUT, POST, DELETE
- REST = estilo de arquitectura
  - Varios niveles de adopción:
    - RESTful
    - Híbrido REST-RPC

REST

- En capas
- Cliente-servidor
  - Sin estado
  - Caché
  - Servidor replicado
- Interfaz uniforme
  - Identificación de recursos (URIs)
  - Manipulación de representaciones de recursos
  - Mensajes auto-descriptivos (tipos MIME)
  - Enlaces a otros recursos (HATEOAS)
- Código bajo demanda (opcional)

REST

Interfaz uniforme:

- Conjunto de operaciones limitado
  - GET, PUT, POST, DELETE
- Conjunto limitado de tipos de contenidos
  - En HTTP se identifican mediante tipos MIME: XML , HTML…

Método	En Bases de datos	Función	Segura?	Idempotente?
PUT	≈Create/Update	Crear/actualizar	No	SI
POST	≈Update	Crea/actualiza subordinado	No	No
GET	Retrieve	Consultar recurso	Si	Si
DELETE	Delete	Eliminar recurso	No	Si

Seguro = No modifica los datos del servidor
Idempotente = El efecto de ejecutarlo n-veces es el mismo que el de ejecutarlo 1 vez

REST

Protocolo cliente/servidor sin estado

Estado gestionado por el cliente

HATEOAS *(Hypermedia As The Engine of Application State)*

Respuestas devuelven URIs a opciones disponibles

Peticiones sucesivas de recursos

Ejemplo: Gestión de alumnos

1.- Obtener lista de alumnos

```
GET http://ejemplo.com/alumnos
```

Devuelve lista de URIs de alumnos

2.- Obtener información de ese alumno

```
GET http://ejemplo.com/alumnos/id2324
```

3.- Actualizar información de ese alumno

```
PUT http://ejemplo.com/alumnos/id2324
```

REST

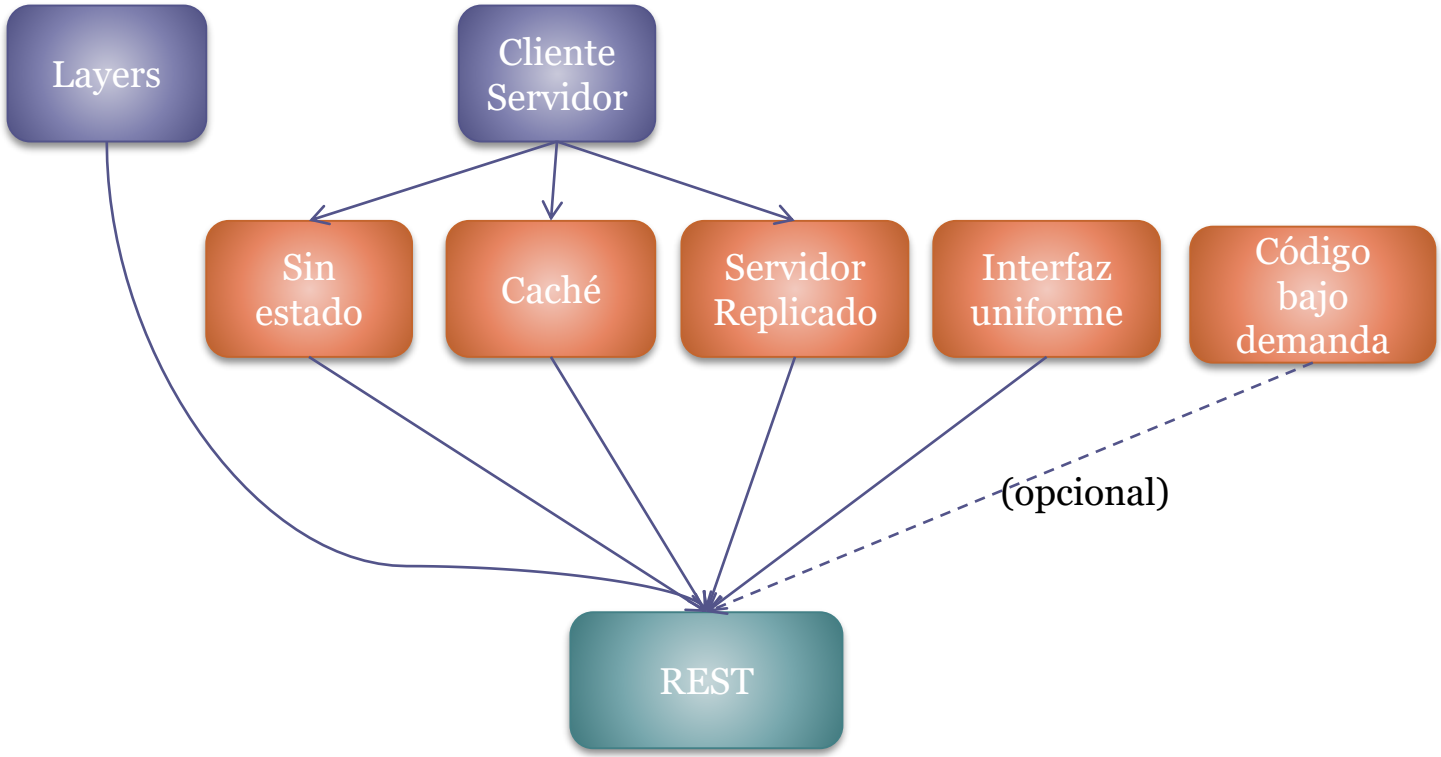
Ventajas

- Cliente/Servidor
  - Separación de responsabilidades
  - Acoplamiento débil
- Interfaz uniforme
  - Facilita comprensión
  - Desarrollo independiente
- Escalabilidad
- Mejora tiempos de respuesta
- Menor carga en red (caché)
  - Ahorro de ancho de banda

Problemas

- REST mal entendido
  - Uso de JSON o XML sin más
    - Servicios Web sin contrato ni descripción
  - REST con estilo RPC
- Dificultades para incorporar otros requisitos
  - Seguridad, transacciones, composición, etc.

REST como estilo compuesto



Microservicios

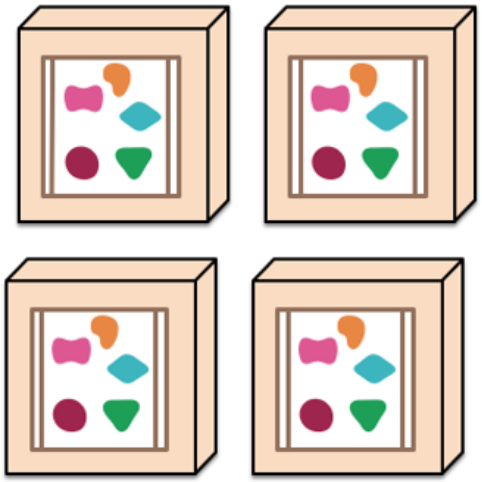
- Aplicaciones complejas se dividen en componentes, llamados servicios
- Cada servicio = bloque de construcción pequeño
  - Altamente desacoplados
  - Enfocados a hacer una pequeña tarea
- Diferencia respecto a SOA
  - En SOA servicios de diferentes aplicaciones
  - Microservicios pertenecen a una misma aplicación

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

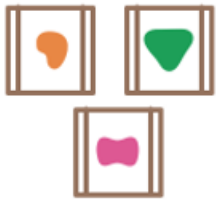
Microservicios y escalabilidad

Aplicación monolítica
Toda funcionalidad en un solo proceso

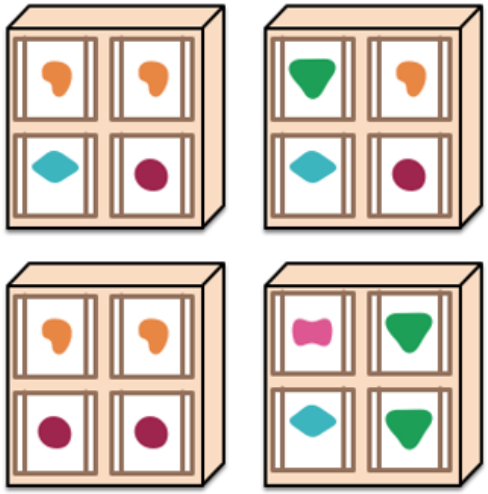
...escalabilidad mediante replicación del monolito en diferentes servidores



Microservicios: Cada funcionalidad distribuida en un microservicio

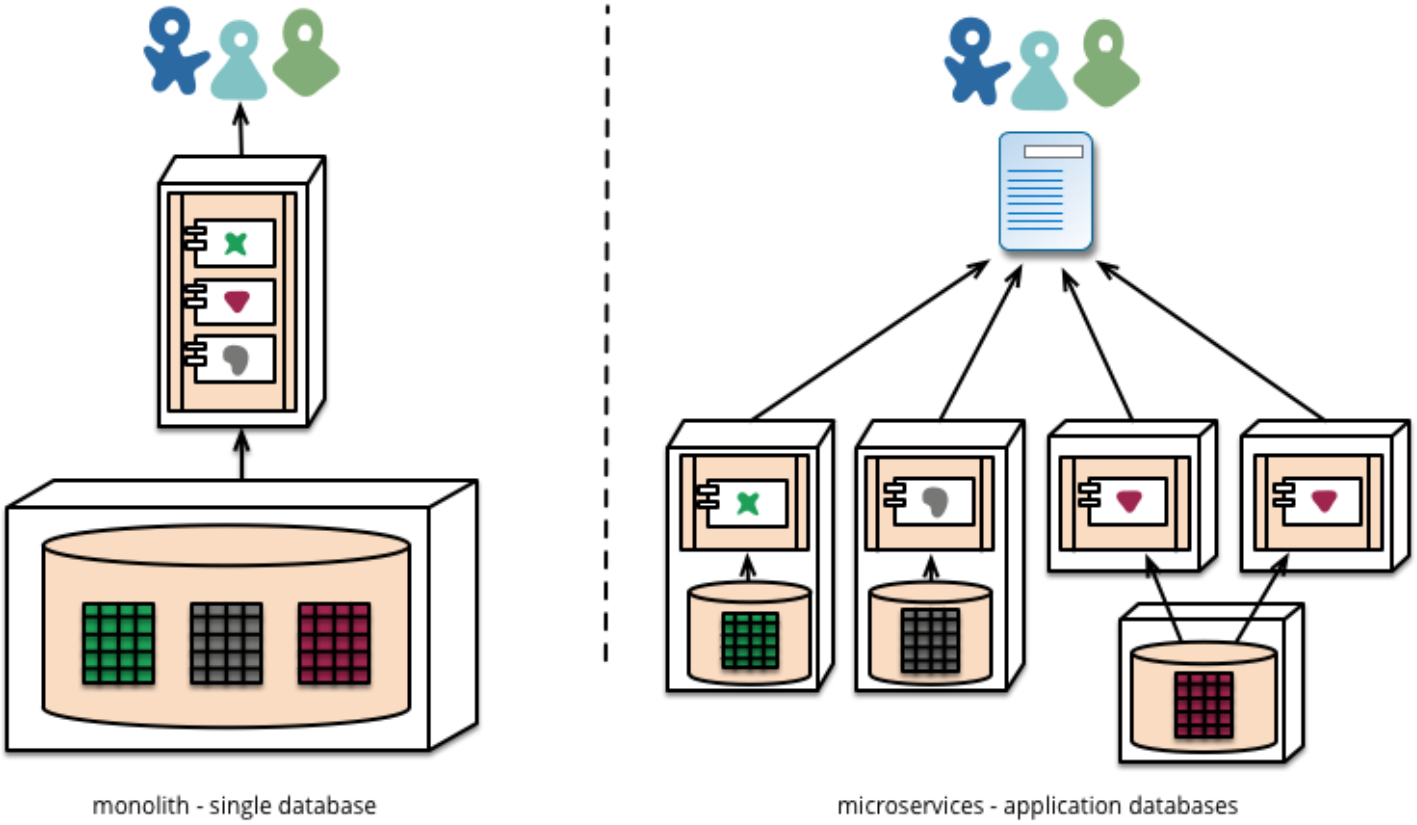


...escalabilidad mediante distribución de servicios en servidores y replicación



Microservicios

Gestión de datos descentralizada



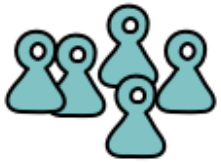
Microservicios

Ley de Conway en aplicación tradicional

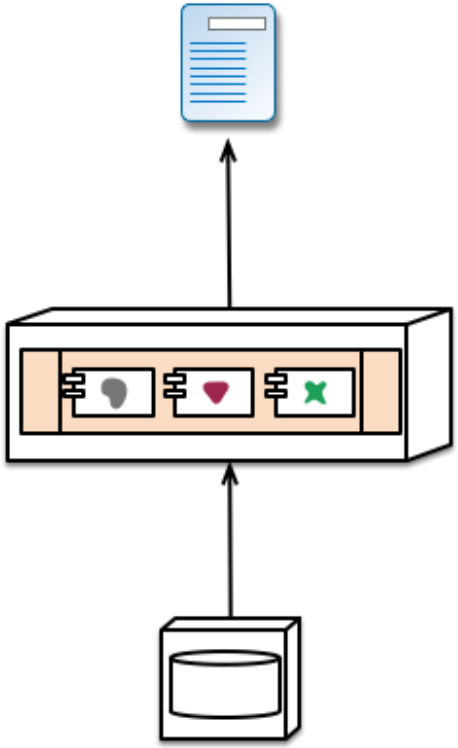
Especialistas Interfaz Usuario

Especialistas Middleware

DBAs



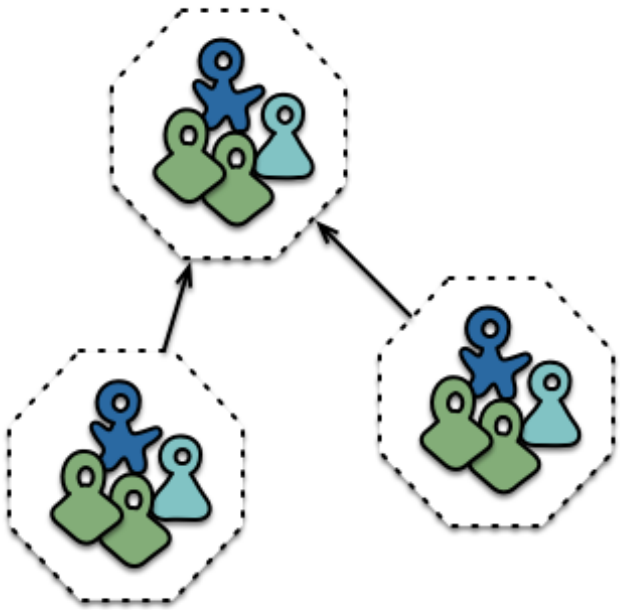
Equipos especializados (en silos)



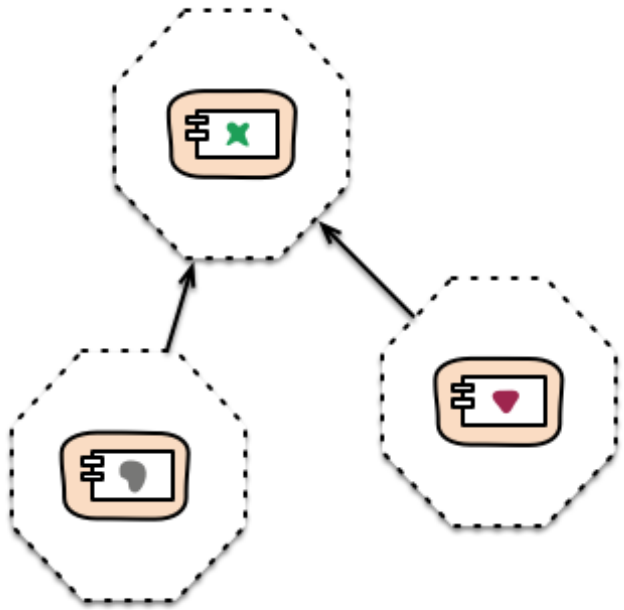
...lleva a arquitecturas basadas en silos Debido a la Ley de Conway

Microservicios

Ley de Conway con microservicios: Equipos basados en funcionalidad



Equipos multidisciplinares
Funcionalidad cruzada



Organizados alrededor de las capacidades
Debido a la Ley de Conway

Microservicios

Ventajas

- Modularización del desarrollo
  - Reusabilidad del microservicio
- Desarrollo y despliegue independiente
  - Escalabilidad
  - Descentralización
- Independencia de tecnologías concretas
  - Cada servicio puede desarrollarse con un lenguaje y una tecnología diferentes
  - Diversidad tecnológica

Problemas/retos

- Gestión de muchos microservicios
  - Demasiados microservicios = antipatrón (nanoservicios)
  - Garantizar la consistencia de la aplicación
- Complejidad de desarrollo
  - Sistemas distribuidos son difíciles de gestionar
  - Aparecen nuevos problemas: latencia, formato de mensajes, balance de carga, tolerancia a fallos, etc.
- Pruebas y despliegue
  - Complejidad operacional

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

Arquitectura *Serverless*

También conocido como:
- Function as a service (FaaS)
- Backend as a service (BaaS)

Aplicaciones dependen de servicios de terceras partes

Los desarrolladores no tienen que preocuparse de los servidores
- Escalabilidad automática

Clientes ricos
- *Aplicaciones Single Page, Aplicaciones móviles*

Ejemplos:
- AWS Lambda, Google Cloud Functions, Ms Azure Functions

Arquitectura *Serverless*

Ventajas

- Escalabilidad
- Disponibilidad
- Rendimiento
- Costes reducidos
  - Costes operacionales
  - Sólo se paga por los recursos computacionales requeridos
- *Time to market* reducido

Retos

- Dependencia de un vendedor
  - Vendor lock-in
  - Incompatibilidad entre soluciones de diferentes vendedores
- Seguridad
- Latencia de arranque
- Pruebas de integración
- Monitorización/depuración

Fin