WEB SERVICES

OVERVIEW OF WEB SERVICE CONCEPTS AND ARCHITECTURES

Peter R. Egli
peteregli.net

Contents

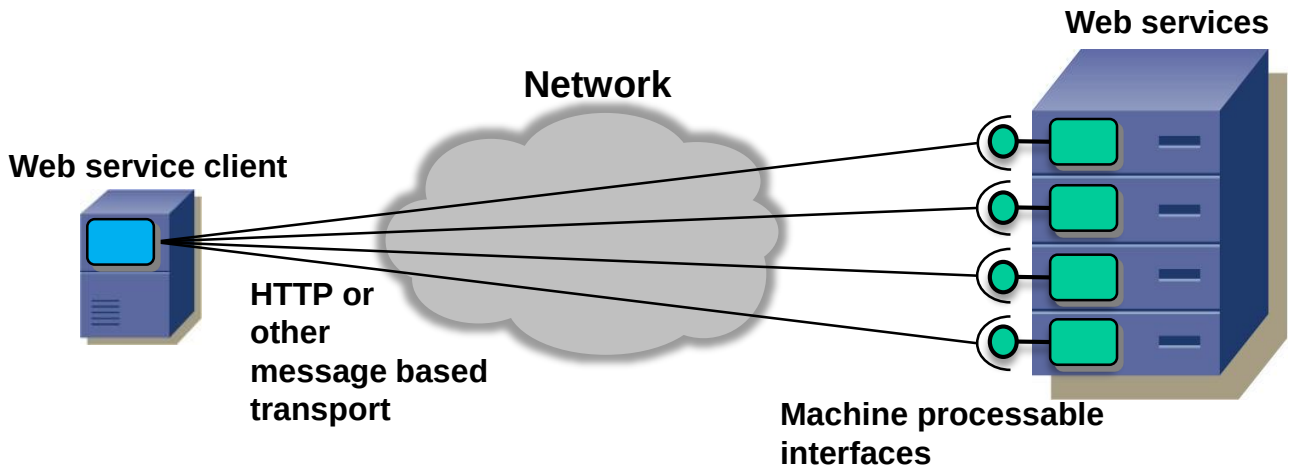
1. What is a web service?

Definition by W3C (see http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/):

A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

Key elements:

- **Machine-to-machine (M2M) interaction over a network**
- **Machine-processable interfaces**
- **APIs that can be accessed over a network**
- **Communication over HTTP (even though SOAP-WS make little use of the HTTP protocol)**



2. What is Service Oriented Architecture (SOA)?

In contrast to „traditional“ enterprise IT architectures, SOA aims to consolidate common functionality of different applications into services.

These services are exposed on a network so all consumers can use them when required.

→ Services offer a defined interface to a component that implements the business functionality.

→ Services should be designed such that they are as independent of each other as possible.

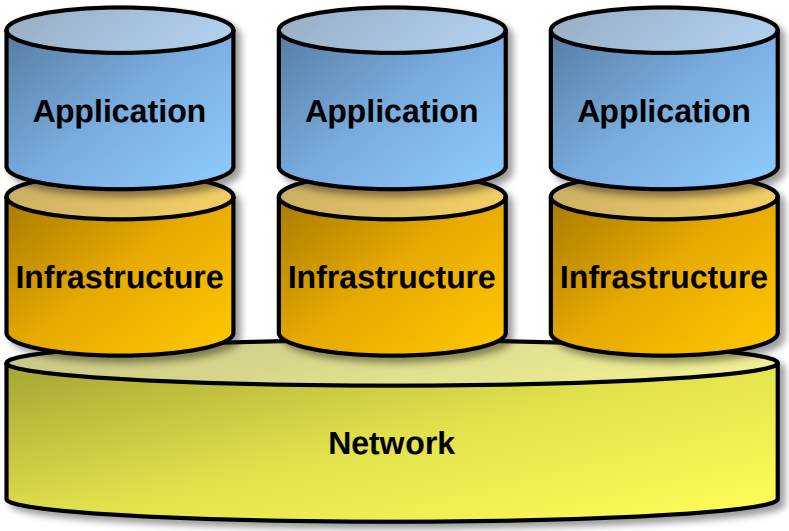
→ Services (functionality) are no longer bound to a specific application.

→ In addition to the partitioning of an IT-landscape into applications, there is an additional dimension «service».

Traditional «stovepipe» or «silo» IT architecture

→ Every application has its own infrastructure.

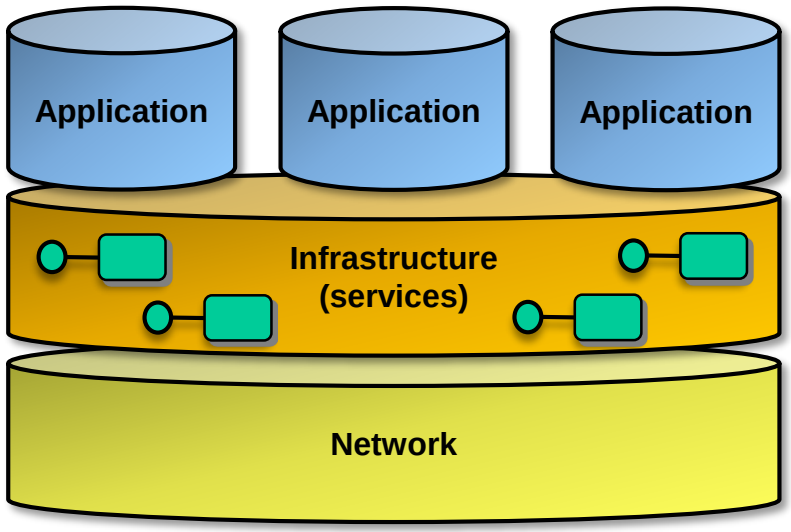
→ Only the network is shared among the applications.



Service oriented architecture

→ Common functionality is consolidated into shared services.

→ Applications run on a common service infrastructure.



3. What is a service?

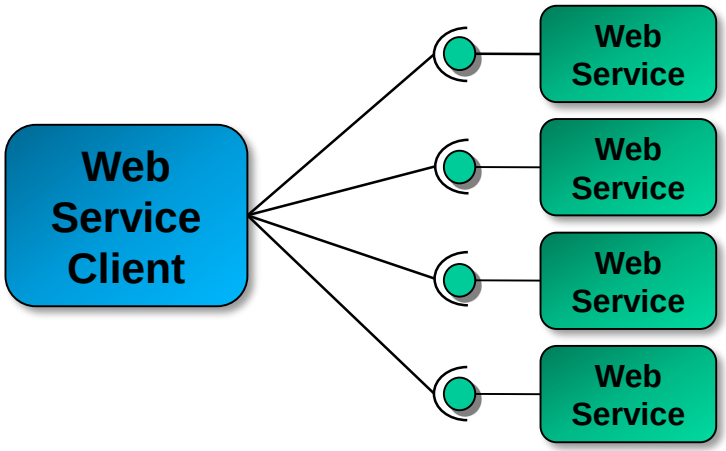
A service groups and encapsulates related functionality.
Example: Account service that allows tranferring / deducting money to / from a bank account.

How big should a service be?

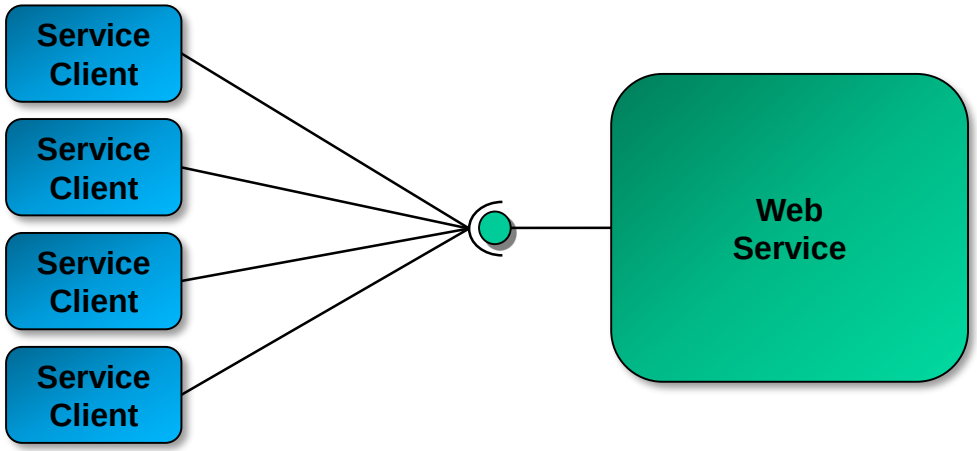
Not too big, not too small...

Too big	**→ Too many dependees, so overall coupling between clients and server is high; it becomes difficult to make changes to the service without affecting the clients.**
Too small	**→ Too many different services; service architecture may become brittle («spaghetti service architecture», «service fragmentation»).**

Tiny web services → service fragmentation



Big web service → tight coupling between service and clients



4. Overview of web service technologies (1/2)

Web service technologies can be classified according to the following 2 criteria:

1. Method information:

The method information expresses the action to be performed in the web service.
With regard to data, there are 3 possibilities:

a. Method = Action to be performed in web service, applied to data that is contained in request. Example: Store logging data on server.

b. Method = Action to be performed in web service, parametrized with data contained in request. Example: Get price quote for a specific product article.

c. Method = Action to be performed in web service, no data.
Example: Get current time and date.

2. Scope information:

The scope information defines the data context to which the request method is applied.

4. Overview of web service technologies (2/2)

1. RPC-style web services („big web services“, „classic web services“):

- HTTP is only used as envelope. The method information is contained in a SOAP- / XML-body.
- The scope information is contained in the SOAP- / XML-body only.

```
<w:getWeatherUpdate xmlns:m=http://examples.indigoo.com/weatherService
 env:encodingStyle=http://www.w3.org/2003/05/soap-encoding
  xmlns:w="http://weatherService.indigoo.com/">
  <w:location>
    <w:latitude>47.359169</w:latitude>
    <w:longitude>8.563843</w.longitude>
  </w:location>
</s:getWeatherUpdate>
```

2. REST-style web services:

- Method information in HTTP method only.
- Scope information in URL only.

```
GET HTTP://www.indigoo.com/weatherInfo?location=Zurich
```

3. Hybrid RPC-style / REST-style combined:

- Method information mapped to URL.
- Scope information contained in URL.

```
GET HTTP://www.indigoo.com/weatherInfo?operation=get&location=Zurich
```

5. Who makes web service standards?

1. W3C (www.w3.org)

W3C develops web technology standards for the Internet (like IETF for protocols).
List of W3C web service standards see http://www.w3.org/TR/.

2. OASIS (www.oasis-open.org)

Besides W3C, OASIS is the main WS standards body.
OASIS integrated the activities of WS-I (www.ws-i.org).
WS-I defines profiles for interoperability between web services (sets of OASIS standards to be supported).
WS-I does not itself define standards but promotes interoperability between WS by defining profiles (set of WS-* standards) to be supported by WS stacks / frameworks.
List of OASIS web service standards see https://www.oasis-open.org/standards.

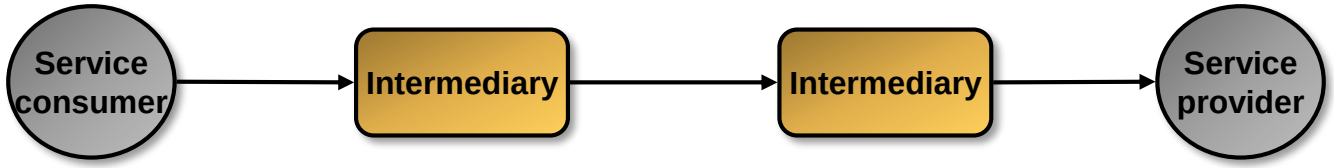
4. Companies

Companies like BEA, IBM and Microsoft develop standards that are sometimes later endorsed by OASIS.

6. Web service architectures (1/4)

Web intermediaries:

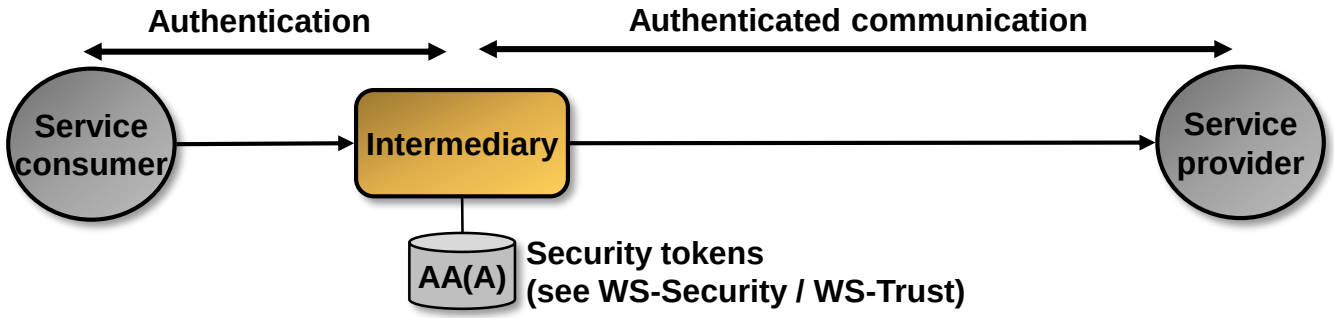
Web intermediaries are placed between web service client and web service provider.
They intercept messages (requests and responses), perform some function and forward the request to the destination.



Intermediaries perform typical functions that are shared in different web service architectures.

A. Authentication services:

The service client authenticates with the authentication service on an intermediary and receives a security token for use in the subsequent secured session with the service provider.

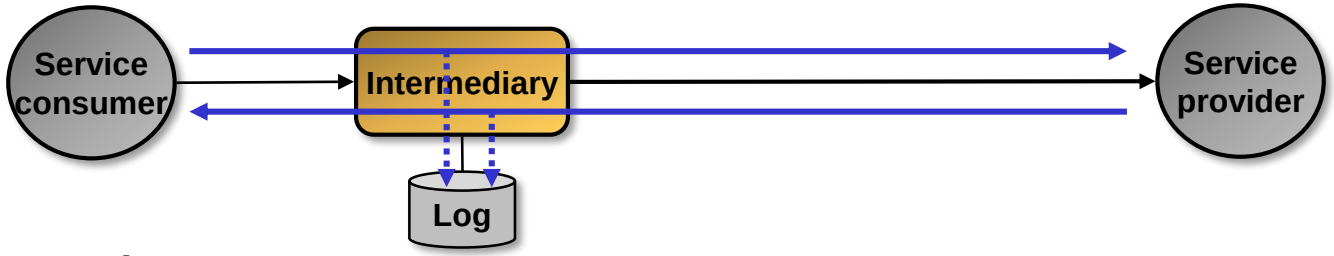


AAA Authentication, Authorization, Accounting

6. Web service architectures (2/4)

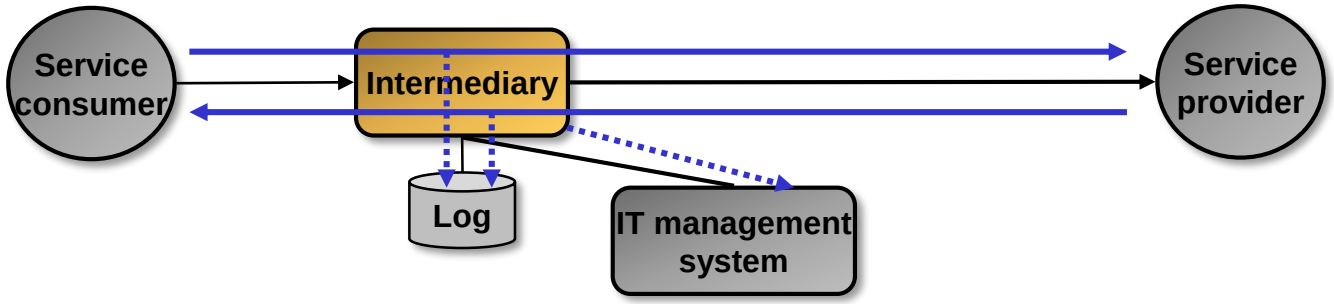
B. Auditing services:

An auditing intermediary logs the entire activity between a service client and provider (e.g. for debugging purposes or to satisfy regulatory requirements).



C. Management services:

A management intermediary is used to collect statistical information about the service usage and service health, maybe based on data from an auditing service.
The management intermediary may be connected to and integrated into an IT management network for service monitoring and management purposes (e.g. for guaranteeing the required web service quality of service).

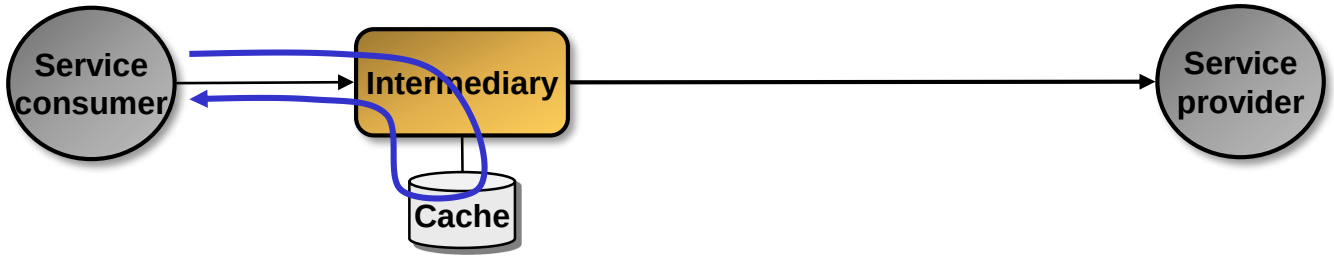


6. Web service architectures (3/4)

D. Performance improvement services (1/2):

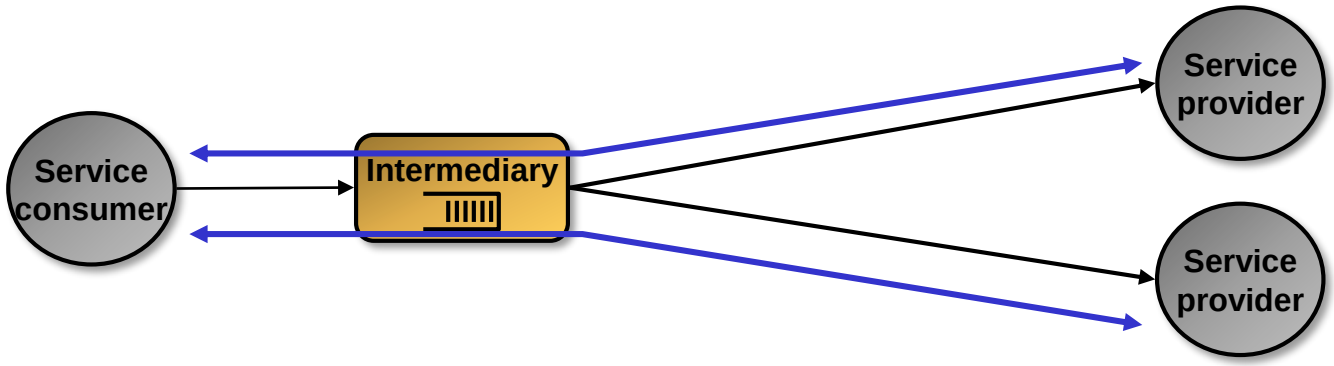
Web intermediaries are placed between service client and provider to improve the overall performance.

Caches: **Resources are retrieved from a cache storage to offload the service provider.**



Store-and-forward intermediaries:

Message queueing service for offloading the service provider, i.e. to control the load or also do load balancing. Such intermediaries may also be used to forward a request to a version-compatible service (see below, versioning of web services).

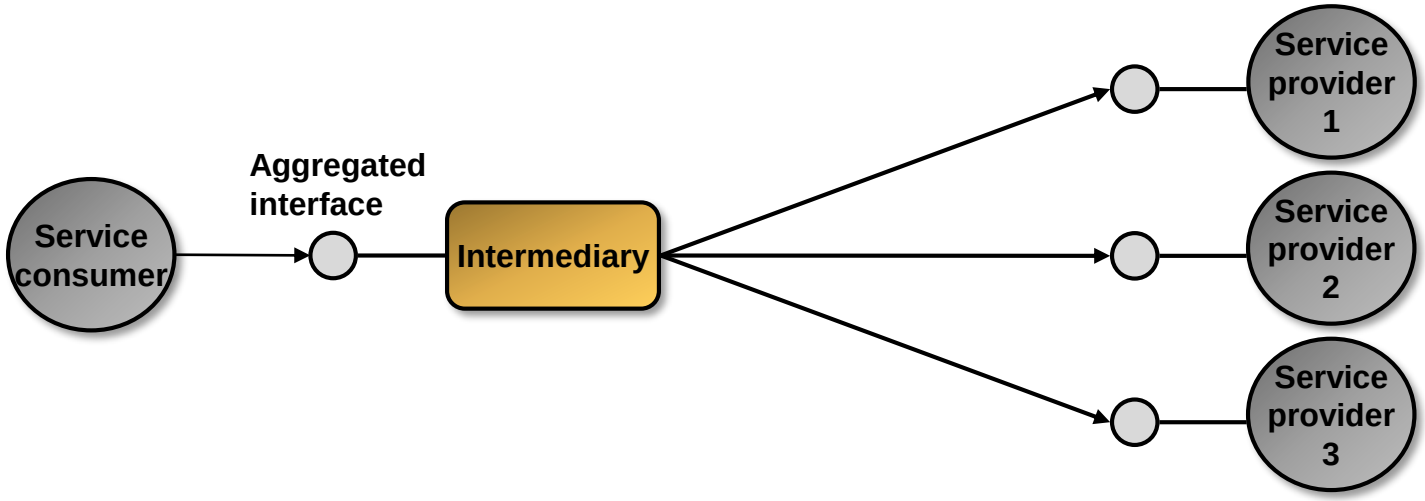


6. Web service architectures (4/4)

D. Performance improvement services (2/2):

Aggregation intermediaries:

Aggregate several web services provided by different web service providers into a composite web service as a facade or interface towards the client.



7. Web service versioning (1/2)

Problem:

SOA is an open architecture (services exposed to the „outside“).
→ Services cannot be easily modified or deleted because some other service or client depends on it.

Solution 1:

Update all clients and other web services each time the service changes («D-Day» approach).
→ Will probably not work well because it does not scale (a small change required by 1 service consumer leads to updates on all dependees).

Solution 2 (1/2):

Introduction of a versioning scheme with compatible and incompatible changes.
Version numbering scheme <major>.<minor>
Major (=incompatible) change → <major> is counted up.
Minor (=compatible) change → <minor> is counted up.
Compatible changes to a web service:
- Addition of an additional operation without changing the semantics of the existing operations
- Addition of an optional attribute to a data type
- Addition of new data type (probably along with a new operation)
Incompatible changes:
- Removal / renaming of a service, operation, data type or attribute
- Addition of a mandatory attribute
- Change of order of attributes (if ordering is applicable)

7. Web service versioning (2/2)

Solution 2 (2/2):

The version must be present in the SOAP message, either as:

a. XML-namespace in the SOAP-envelope:

Example: `<xsd:schema targetNamespace=„http://www.mycompany.com/MyService/2009-05-20>`

b. Service name or data element extension in SOAP body:

Example: `<xs:complexType name=„CustomerV1“>`

c. HTTP-request parameter for version:

Example:

```
HTTP GET HTTP/1.1
..
version: V1.2
```

In order to reduce or avoid any client changes when doing major changes to a web service, the following scheme may be employed (the web intermediary serves as a service router based on the version):