SOA

Interactive web services, Course, Fall 2005

Troels C. Damgaard

IT University of Copenhagen
Service-oriented Architecture (SOA)

Service-oriented Architecture (SOA) is:
- business approach;
- design approach;
- development approach.

Key (concrete) way to realize a SOA is:
- web services!
- Focus of this lecture: SOA as related to web services.
Normal application development

Most applications developed for *single user, single machine.*
Web applications: really server applications used by many clients.
Web services

Even more complex!

Service

Client

Registry
Services as the main structuring principle:

- a *service* is a “well-defined functional unit” of the application;
- *service contracts* defines the interface to a service;
- facilitates *loose coupling* of the application.
Orchestration

Services orchestrated/choreographed by application:

- *service orchestration* builds an app out of individual services;
- orchestration specifies the “workflow”.

![Diagram showing service orchestration for building an app](image-url)
Service discovery

Services can (optionally) be registered and discovered in a service registry/service catalog.
What is new?

◆ Can achieve much of the same with other principles:
  ◆ CORBA,
  ◆ DCOM,
  ◆ …
◆ No specifics about the *internal* design of services.
◆ Focus on:
  ◆ interoperability;
  ◆ platform and language independence;
  ◆ *loose coupling*;
  ◆ standards-based messaging;
  ◆ …
Stateless service design

Services should not depend on prior message exchanges or state of other services.

- services should be independent;
- services should correspond to self-contained requests;
- services should not depend on session keys;
- services should not depend on the state of other services.

Example:

- Bad interaction:
  
  new shopping cart
  find book
  add book
  proceed to checkout

- Better interaction:
  
  new shopping cart → c
  find book → b
  add book b to cart c
  proceed cart c to checkout

- Services often have (internal) state (ex: order history).
Three levels of abstraction in SOA:
- operations (ex: “deduct x dollars from account y”);
- services (ex: “debit credit card z”);

Distinguishing characteristics (towards architecture design):
- From short-lived, to long running.
- From fine-grained to coarse-grained.
Thinking about service granularity is important!

- **fine-grained services:**
  - many smaller services (“one per operation”);
  - large flexibility;
  - more invocation overhead;
  - harder to control.

- **coarse-grained services:**
  - smaller operations grouped into one service;
  - less flexibility;
  - less invocation overhead;
  - easier to control.

**Standard advice:**

- use fine-grained services internally;
- use coarse-grained services externally.
Loose coupling

A key benefit of services is *loose coupling* of components:

- service interface is the only contract between service consumer and service provider;
- interface does not mention *programming language*;
- interface specified in language and platform *independent* way;
- messaging framework for further language independence;
- through *discovery*, services can move;
  - consumer *needs* no technical details of the service.

Consequences:

- Service implementation can change and be improved.
- Implementation language is irrelevant and may change.
- Deployment platform is irrelevant and may change.
- Service location is irrelevant and may change.
Preview – XML based technologies for SOA
Well-defined interfaces through WSDL

WSDL provides key aspects of loose coupling:

◆ specifies contract:
  ◆ service description;
  ◆ message descriptions;
  ◆ interaction descriptions;
  ◆ how to construct messages;
  ◆ where to send messages.

◆ is programming language independent;
◆ is platform independent;
◆ allows the service to move (if WSDL can be found).
Messaging using SOAP

SOAP provides the last bit of language and platform independence:

- XML-based
  - rather than proprietary serialization format;
- (can be) HTTP-based
  - does not require application-level protocol;
- extensible.
UDDI combines the description of interfaces and messaging framework:

- services can register in UDDI;
- applications/services can query service information (white, yellow, and green pages);
- even less coupling.
Lots of material out there.

- **Brief introduction (with Java and WS point of view):**

- **Many IBM developerWorks papers:**
  For example: “New to SOA and Web Services”

- **Henrik Hvid Jensen:** “Service Orienteret Arkitektur: Integration som konkurrenceparameter”
Application-to-application web services

*Interactive web services, Fall 2005*

Troels C. Damgaard

IT University of Copenhagen
A classification of web services:

- *interactive web services*
  - client is a human being (using a browser);
  - JSP / Servlets / JWIG / . . .

- *application-to-application web services*
  - client is a *program*.

Examples of application-to-application web services:

- Web of Jokes;
- Amazon’s web service (access Amazon’s product information).
- Google’s web service (access the Google search engine);
Web services technologies

Basically: XML + HTTP.

◆ XML over HTTP;
◆ XML-RPC (which you might see more on next time);
◆ SOAP, WSDL, UDDI.

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.

[“W3C Web Services Glossary”]
The “web services platform” uses SOAP, WSDL, and UDDI to connect the components.

Here is the “SOA triangle” revisited:
Example: Web of Jokes

XML over HTTP, but with the “web services architecture”.

(If you haven’t already — I suggest you read the rest of chapter 12 in the Kompendium (in particular sect. 12.8))
SOAP

SOAP is a framework for exchanging XML-based information in a network.

◆ SOAP used to be an acronym: Simple Object Access Protocol.
◆ This is no longer the case…
  ◆ it is neither simple,
  ◆ nor has it got anything to do with objects.
◆ The currently most hyped XML/Web service technology.

◆ Current version 1.2 (status: “Recommendation”).
◆ Look at the SOAP 1.2 Part 0 “Primer” for an introduction.
WSDL (Web Services Definition Language)

WSDL is an XML based language for describing network services.
◆ WSDL describes capabilities and locations of services;
◆ like an interface description language for Web services;
◆ communication using SOAP or direct HTTP.

◆ Current version 2.0 (status: “Last call working draft”)
◆ Look at the WSDL 2.0 “Primer” for an introduction.
UDDI (Universal Description, Discovery, and Integration) provides a mechanism for clients and servers to find each other.

- like a service catalog;
- uses SOAP for communication.

- Current version 3.0 (status: “Committee draft”)
WSDL
WSDL (Web Services Description Language) allows a formal description of the interfaces of a web service.

- which operations does the service provide?
- which arguments and results are involved in the operations?
- which network addresses are used to locate the service?
- which communication protocol should be used?
- which data formats are the messages represented in?

Note: WSDL does not specify semantics (only “mechanics”).
Benefits:

- Description acts as a *contract* between server developers and client developers.
- A *formal* description allows tool support;
  - (stub) code generators.
The WSDL language can be described as having two layers:

- the service definition layer describes *abstract* properties:
  - data types
  - message types
  - operations
  - services

- the binding layer describes *concrete* properties:
  - protocols
  - data formats
  - addresses
  (using SOAP, HTTP, MIME)
An actual WSDL document consists of a `<description>` of the following kind:

- `<types>`: containing XML Schema element and type definitions, that is used to describe the kinds of messages that the service provides in...

- The `<interface>`: describing a set of operations basically by
  - selecting one of eight interaction patterns;
  - and describing the `input` and `output` types
  - and possible specifying a `fault` type.

- `<binding>`: selects communication protocol and data formats;

- `<service>`: describes named endpoints (bindings and network addresses) (`location` info, basically).

An import mechanism allows modularization of definitions.
WSDL Example (I)

(from WSDL primer)

```xml
<?xml version="1.0" encoding="utf-8" ?>
<description
 xmlns="http://www.w3.org/2005/08/wsdl"
targetNamespace= "http://greath.example.com/2004/wsdl/resSvc"
xmlns:wsoap= "http://www.w3.org/2005/08/wsdl/soap"
xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsdlx="http://www.w3.org/2005/08/wsdl-extensions">

  <types> ... </types>
  <message> ... </message>
  <binding> ... </binding>
  <service> ... </service>
</description>
```
WSDL Example (II)

Specifying the types used by the service:

```xml
<types>
    <xs:schema
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://greath.example.com/2004/schemas/resSvc"
        xmlns="http://greath.example.com/2004/schemas/resSvc">
        <xs:element name="checkAvailability" type="tCheckAvailability"/>
        <xs:complexType name="tCheckAvailability">
            <xs:sequence>
                <xs:element name="checkInDate" type="xs:date"/>
                <xs:element name="checkOutDate" type="xs:date"/>
                <xs:element name="roomType" type="xs:string"/>
            </xs:sequence>
        </xs:complexType>
        <xs:element name="checkAvailabilityResponse" type="xs:double"/>
        <xs:element name="invalidDataError" type="xs:string"/>
    </xs:schema>
</types>
```

◆ Data types can be defined using XML Schema.
WSDL Example (III)

Specifying the interface for messages sent to and from the service:

```xml
<interface name = "reservationInterface" >
  <fault name = "invalidDataFault"
    element = "ghns:invalidDataError"/>
  <operation name="opCheckAvailability"
    pattern="http://www.w3.org/2005/08/wsd1/in-out" >
    <input messageLabel="In"
      element="ghns:checkAvailability" />
    <output messageLabel="Out"
      element="ghns:checkAvailabilityResponse" />
    <outfault ref="tns:invalidDataFault" messageLabel="Out"/>
  </operation>
</interface>
```

◆ Interfaces are defined in terms of the defined data types.
WSDL Example (IV)

Specifying the concrete messaging framework:

```xml
<binding name="reservationSOAPBinding"
    interface="tns:reservationInterface"
    type="http://www.w3.org/2005/08/wsd1/soap"
    wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP">
    <fault ref="tns:invalidDataFault"
        wsoap:code="soap:Sender"/>
    <operation ref="tns:opCheckAvailability"
        wsoap:mep="http://www.w3.org/2003/05/soap/mep/soap-response"/>
</binding>
```

◆ Most commonly used: SOAP.
WSDL Example (V)

Specifying the address:

```xml
<service name="reservationService"
    interface="tns:reservationInterface">
    <endpoint name="reservationEndpoint"
        binding="tns:reservationSOAPBinding"
        address ="http://greath.example.com/2004/reservation"/>
</service>

◆ Tie the binding to an address.
```
WSDL Bindings

- **SOAP**
  - next slide …

- **HTTP**
  - selects GET or POST method;
  - absolute URI for each port;
  - relative URI for each operation;
  - optionally, encoding of request message parts (URL encoding, URL replacement).

- **MIME**
  - specifies MIME types for message parts (text/xml, multipart/related,…);
  - only describes data formats, needs SOAP/HTTP binding to specify communication protocol.
SOAP
SOAP (used to be *Simple Object Access Protocol*) is

“*fundamentally a stateless, one-way message exchange paradigm [for XML-based information].*”  
[“W3C SOAP primer”]

- **SOAP is big** (500 page spec and still growing):
  - Part 1: Messaging Framework;
  - Part 2: Adjuncts;
  - Email Binding;
  - Attachment Feature;
  - …

- **Typical message exchange styles:**
  - *document-style* (one-way XML messages);
  - *RPC* (Remote Procedure Call, request-response).

- **Typical protocol bindings:**
  - HTTP (not the same as the HTTP binding for WSDL!);
  - SMTP (Simple Mail Transport Protocol).
Using SOAP

Using SOAP typically means:

◆ when sending messages, put them into a SOAP envelope;
◆ when receiving messages, take them out of the SOAP envelope.

An envelope is a wrapper containing:

◆ a header
  ◆ meta-info to intermediaries and end-user (network nodes on the message path);
  ◆ data-extensions (helps provide backward compatibility).
◆ a body
  ◆ the actual contents (depending on the application).
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2002/06/soap-envelope">
  <env:Header>
    <m:reservation xmlns:m="http://travelcompany.example.org/reservation" env:role="http://www.w3.org/2002/06/soap-envelope/role/next" env:mustUnderstand="true">
      <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
      <m:dateAndTime>2001-11-29T13:20:00.000-05:00</m:dateAndTime>
    </m:reservation>
      <n:name>John Q. Public</n:name>
    </n:passenger>
  </env:Header>
  ...
</env:Envelope>
<env:Body>
  <p:itinerary xmlns:p="http://travelcompany.example.org/reservation/travel">
    <p:departure>
      <p:departureDate>2001-12-14</p:departureDate><p:departureTime>late afternoon</p:departureTime>
      <p:seatPreference>aisle</p:seatPreference>
    </p:departure>
    <p:return>
      <p:departureDate>2001-12-20</p:departureDate><p:departureTime>mid-morning</p:departureTime>
      <p:seatPreference/>
    </p:return>
  </p:itinerary>
  <q:lodging xmlns:q="http://travelcompany.example.org/reservation/hotels">
    <q:preference>none</q:preference>
  </q:lodging>
</env:Body>
</env:Envelope>
Using SOAP in WSDL

The SOAP binding in WSDL:

- selects document or rpc style;
- selects HTTP/SMTP/... protocol;
- selects encoding (typically, the “SOAP encoding”);
UDDI
UDDI stands for *(Universal Description, Discovery, and Integration)* and is a Web service *registry* mechanism.

◆ “a meta service for locating Web services by enabling robust queries against rich metadata”; ([http://www.uddi.org](http://www.uddi.org))

◆ 400+ page spec.

Provide access to:

◆ a SOAP+WSDL-based registry API for
  ◆ registering, and
  ◆ discovering Web services.

◆ web interfaces
  ◆ [http://www.soapclient.com/uddisearch.html](http://www.soapclient.com/uddisearch.html);
XML files used to describe business entities and their Web services. They collect and provide information consisting of three components:

- "white pages": business address, contact info, etc.;
- "yellow pages": industrial categorizations based on standard taxonomies;
- "green pages": more technical information, for instance WSDL descriptions.
A UDDI businessEntity describes a business and its services:

```
<businessEntity businessKey="ba744ed0-3aaf-11d5-80dc-002035229c64"
    operator="www.ibm.com/services/uddi" authorizedName="0100001QS1">
    <discoveryURLs>
        <discoveryURL useType="businessEntity">
        </discoveryURL>
    </discoveryURLs>
    <name>XMethods</name>
    <description xml:lang="en">Web services resource site</description>
    <contacts>
        <contact useType="Founder">
            <personName>Tony Hong</personName>
            <phone useType="Founder" />
            <email useType="Founder">thong@xmethods.net</email>
        </contact>
    </contacts>
    ...
```
<businessServices>
  <businessService serviceKey="d5921160-3e16-11d5-98bf-002035229c64"
businessKey="ba744ed0-3aaf-11d5-80dc-002035229c64">
    <name>XMethods Delayed Stock Quotes</name>
    <description xml:lang="en">20-minute delayed stock quotes</description>
    <bindingTemplates>
      <bindingTemplate bindingKey="d594a970-3e16-11d5-98bf-002035229c64"
serviceKey="d5921160-3e16-11d5-98bf-002035229c64">
        <description xml:lang="en">
          SOAP binding for delayed stock quotes service
        </description>
        <accessPoint URLType="http">
          http://services.xmethods.net:80/soap
        </accessPoint>
      </bindingTemplate>
    </bindingTemplates>
  </businessService>
</businessServices>
This `tModel` for the single service described above refers to bindings in a WSDL description:

```xml
<tModel tModelKey="uuid:0e727db0-3e14-11d5-98bf-002035229c64"
    operator="www.ibm.com/services/uddi" authorizedName="0100001QS1">
    <name>XMethods Simple Stock Quote</name>
    <description xml:lang="en">Simple stock quote interface</description>
    <overviewDoc>
        <description xml:lang="en">wsdl link</description>
        <overviewURL>
            http://www.xmethods.net/tmodels/SimpleStockQuote.wsdl
        </overviewURL>
    </overviewDoc>
    <categoryBag>
        <keyedReference tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4"
            keyName="uddi-org:types" keyValue="wsdlSpec"/>
    </categoryBag>
</tModel>
```

- the WSDL document has no service part - the service address is specified in the bindingTemplate.
- UDDI is much more general (and hence complicated) than this example might suggest.
The following SOAP message could be sent to a UDDI registry to inquire about services named “delayed stock quotes”:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
  <Body>
    <find_service businessKey="*" generic="1.0" xmlns="urn:uddi-org:ap" name="delayed stock quotes"/>
  </Body>
</Envelope>
```

- alternatively, we could search by category codes of various kinds
- the find_service meta-service operation is specified in the WSDL for the UDDI API
- of course, the UDDI registries are also registered services...
Summary

◆ Use WSDL for describing your web service.
◆ Use SOAP for sending messages to and from your web service.
◆ Use UDDI to keep track of web services.
Some selected links:

- webservices.xml.com
  XML.com’s Web service section

- webservices.xml.com/pub/a/ws/2002/02/12/webservicefaqs.html
  "Top Ten FAQs for Web Services"

- www.w3.org/2002/ws
  W3C’s Web Services activity

- ws.apache.org
  Apache’s Web Service project

- www.onjava.com/pub/a/onjava/excerpt/java_xml_2_ch2
  chapter on SOAP from the book "Java and XML"

- wsindex.org
  Web service links and resources
Exercises