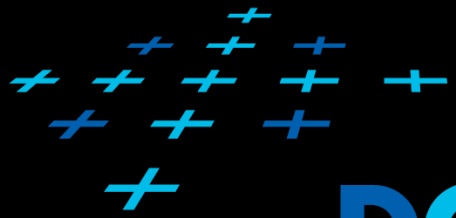


DCGI

KATEDRA POČÍTAČOVÉ GRAFIKY A INTERAKCE

Web applications 2
SoA and REST

Martin Klíma

Communication, remote procedure call

RMI = Remote Method Invocation

- Sun Microsystems
- Java platform only

RPC = Remote Procedure Calls

- Sun Microsystems
- Platform dependent

CORBA = Common Object Request Broker Architecture

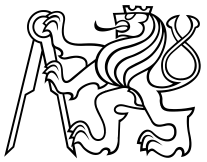
- OMG
- Platform independent, not easy to use

DCOM = Distributed Common Object Model

- Microsoft
- Platform dependent

Web services

- Platform independent
- REST, SOAP



What is SoA

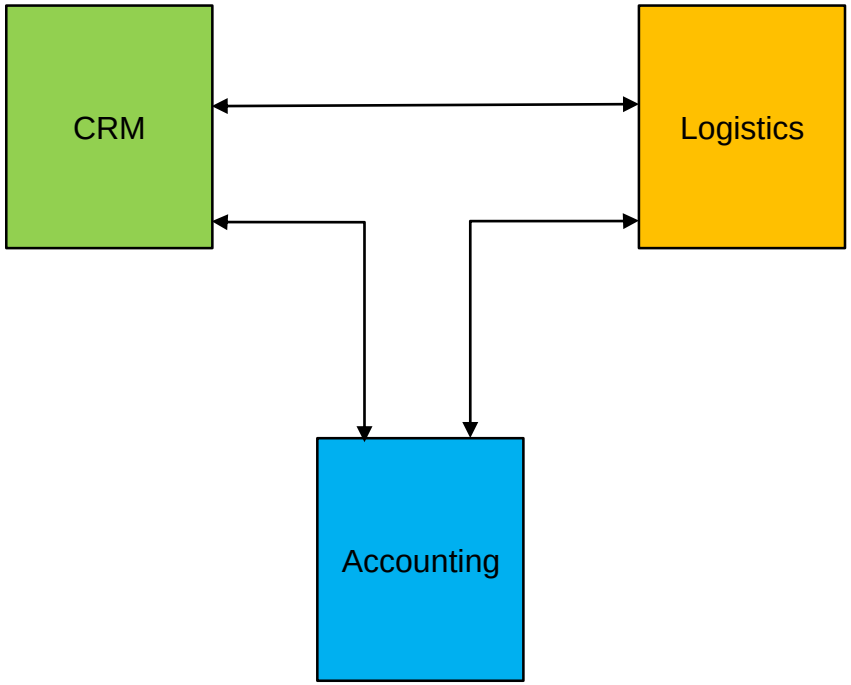
SOA is a software architecture model

- in which business functionality are logically grouped and encapsulated into
  - self contained,
  - distinct and reusable units

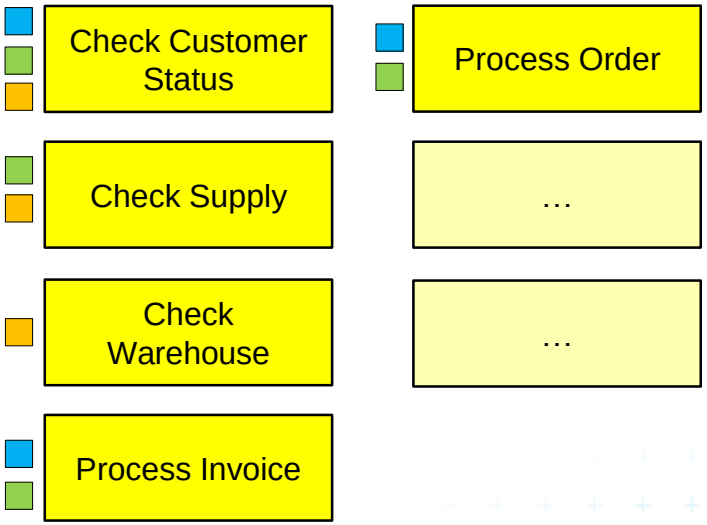
    called **services that**
  - represent a high level business concept
  - can be distributed over a network
  - can be reused to create new business applications
  - contain contract with specification of the purpose, functionality, interfaces (coarse grained), constraints, usage

Practical problems

Application centric view



SoA view – set of services



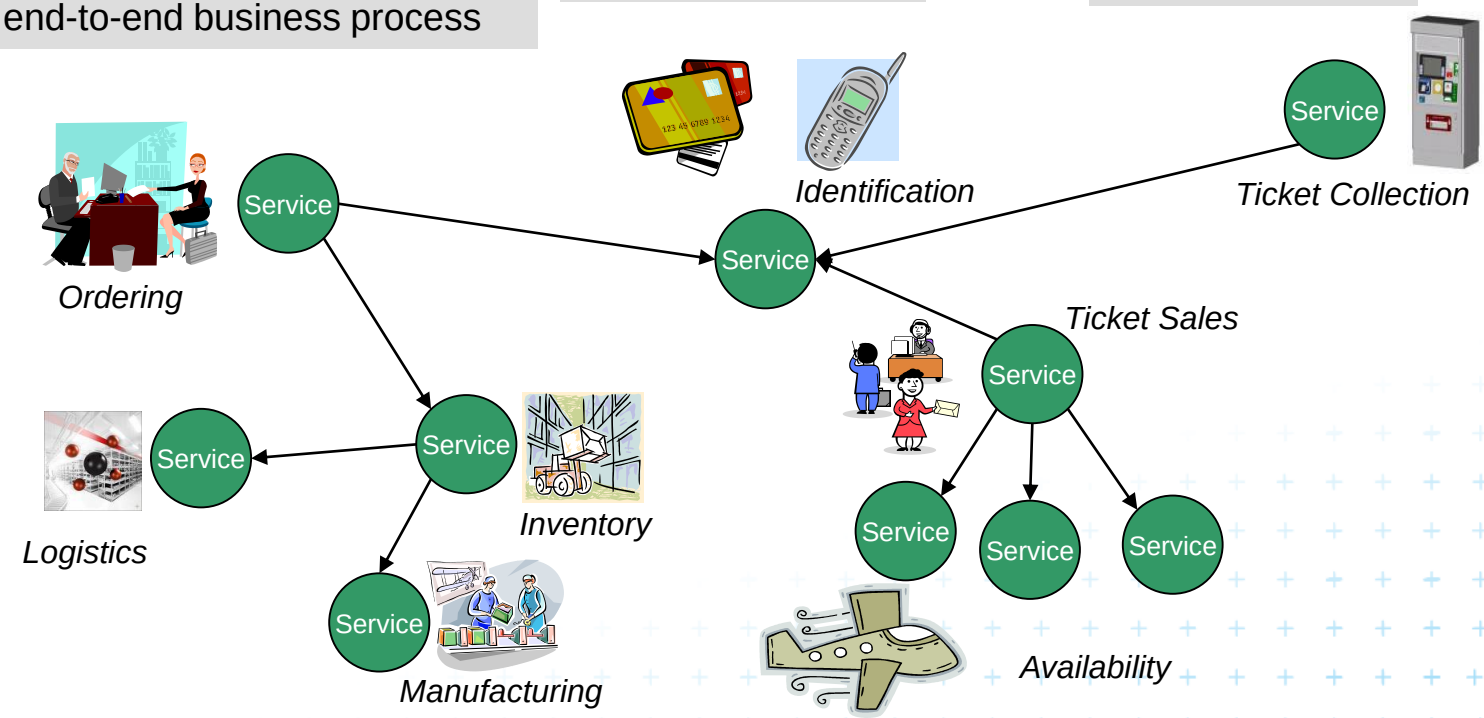
DCGI

Advantages of SoA?

Enabling a virtual federation of participants to collaborate in an end-to-end business process

Enabling alternative implementations

Enabling reuse of Services



Enabling virtualization of business resources

Enabling aggregation from multiple providers

source: TietoEnator AB, Kurts Bilder

DCGI

SoA Principles

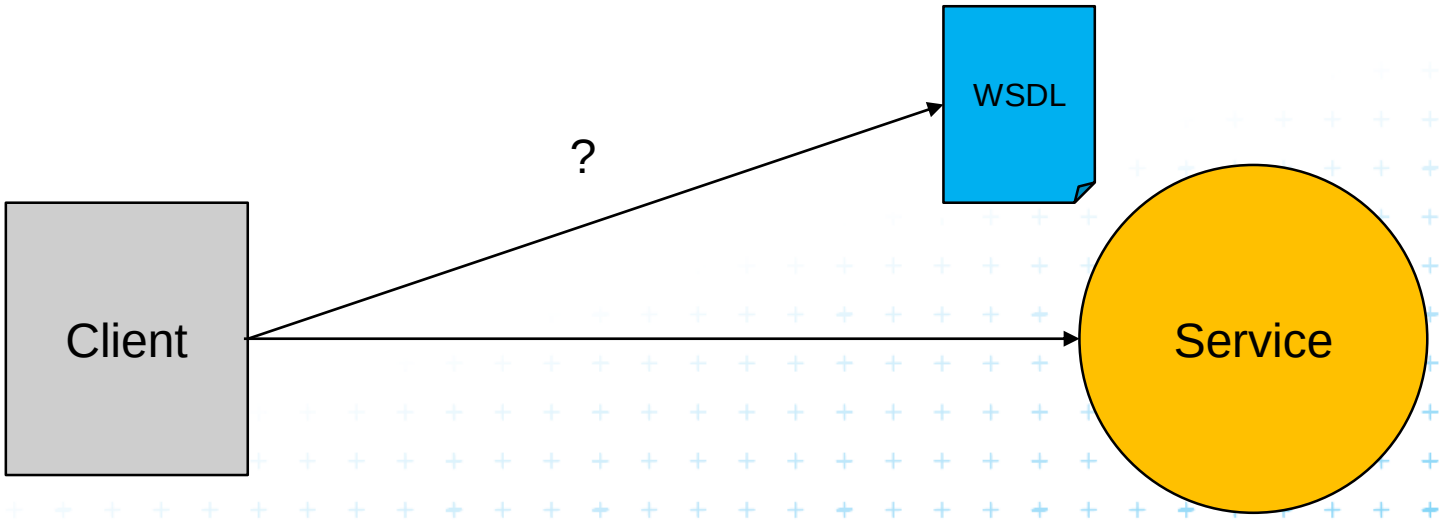
- Standardized Service Contracts
- Loose Coupling
- Abstraction
- Reusability
- Autonomy
- Statelessness
- Discoverability
- Composability

Other common properties

Asynchronous
Communication over internet
Web based

Standardized Service Contracts

- Services use service contract to
  - Express their purpose
  - Express their capabilities
- Use formal, standardized service contracts

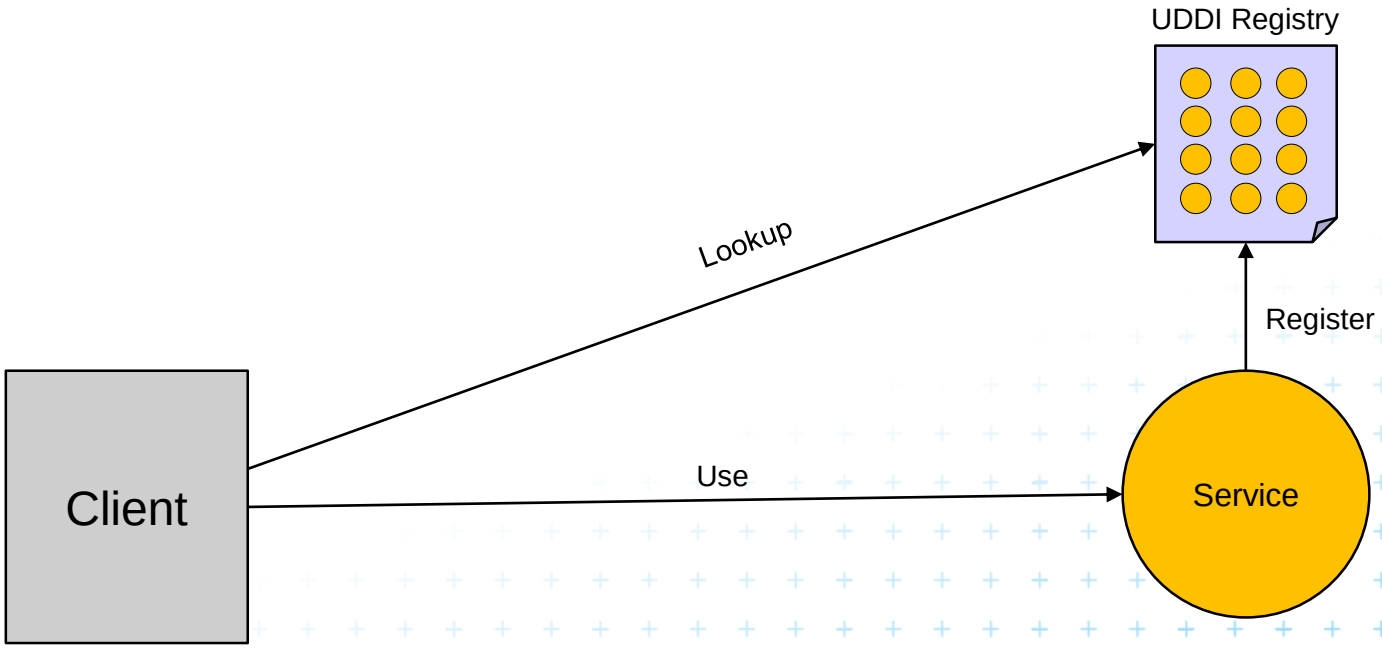


Statelessness

- Very strong principle!
- *"Services minimize resource consumption by deferring the management of state information when necessary."*
- Serve and forget
  - Does not prevent usage of external data sources including DBs.
- Scalability
  - Important for clouds.

Discoverability

- Build service registry
  - UDDI = Universal Description, Discovery, and Integration
- Use metadat for discovery



Web Services

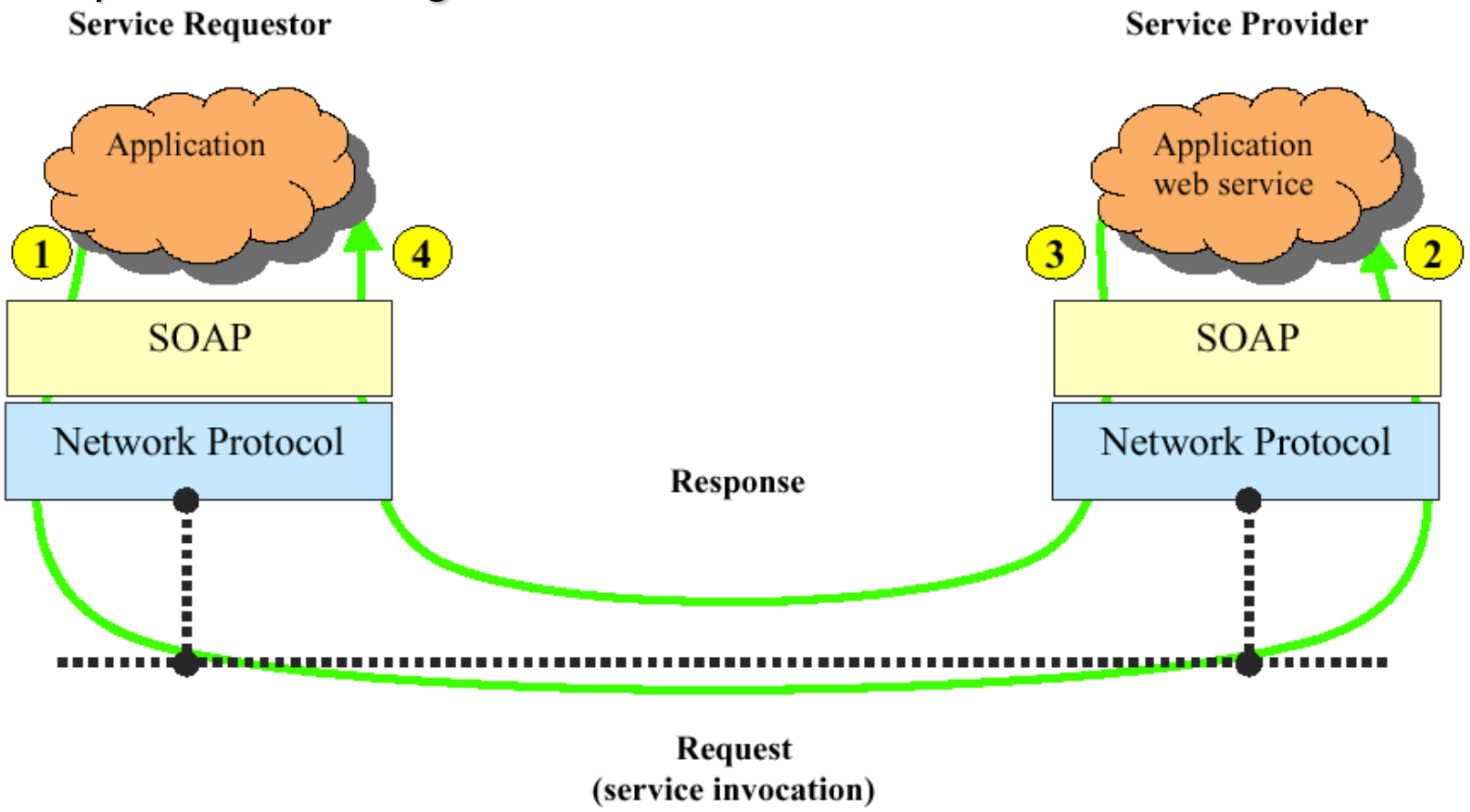
- Call a remote service over Internet
- Accessible via Internet protocols
  - HTTP
  - HTTPS
  - XML, JSON
  - SOAP
- Platform independent
- Messaging in SOAP = based on XML
- Firewall friendly, uses HTTP (ports 80, 443)

WS Protocols

Vrstva	Technologie
Service Directory	UDDI
Service Description	WSDL
Messaging	SOAP
Encoding	XML
Transport	HTTP,…

SOAP

- Simple Object Access Protocol
  - How to call a service
  - How to encode a mesage
- http://www.w3c.org/TR/SOAP/

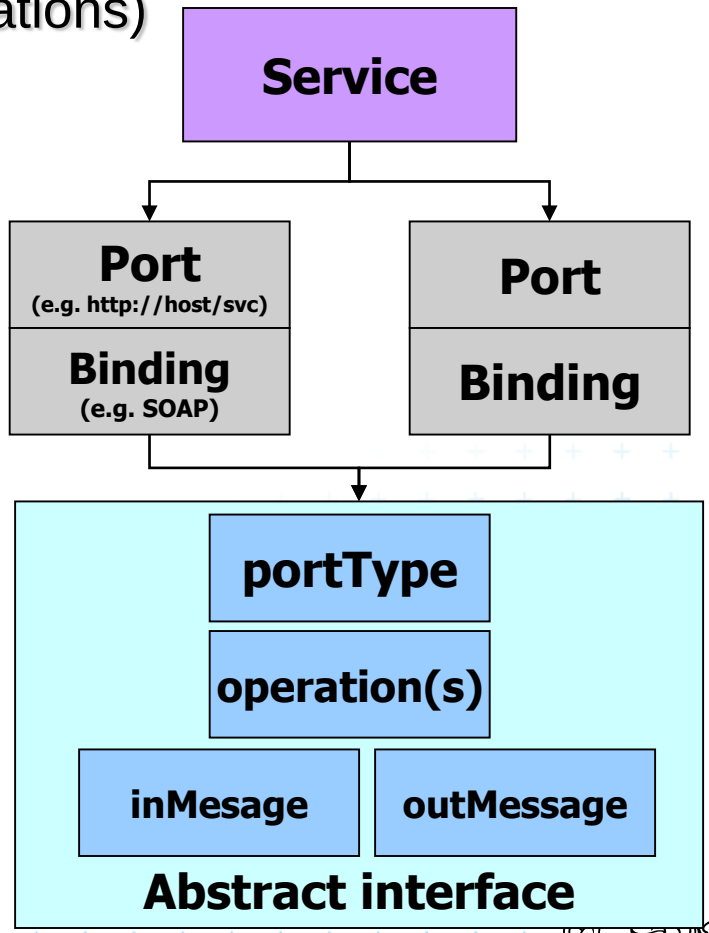


WSDL

- Web Services Definition Language
  - http://www.w3.org/TR/wsdl/
- Based on XML
- What it defines
  - What the web service does (description)
  - Hot to use it (method signatures)
  - Where to find it
- Independent on protocols of lower level
  - HTTP, IMAP

WSDL Structure

- portType
  - Abstract service definition (set of operations)
  - How to call the service
  - SOAP, JMS, direct call
- Ports
  - Where is it accessbile



Web services in Java

- Java Web Services
  - JAX-WS
    - Mapping to and from WSDL
  - JAXB 2.0
    - (un)marshar
    - XML, XML Schema
  - WS-Metadata
  - WSEE

Example WS

```
package cz.cvut.fel.wa2.ws;

import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;

/**
  * @author klima
 */
@WebService(serviceName = "WA2WebServiceHello")
public class WA2WebServiceHello {

    /**
     * This is a sample web service operation
     */
    @WebMethod(operationName = "hello")
    public String hello(@WebParam(name = "name") String txt) {
        return "Hello " + txt + " !";
    }
}
```

WSDL

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<definitions targetNamespace="http://ws.wa2.fel.cvut.cz/" name="WA2WebServiceHello"
xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:wsp="http://www.w3.org/ns/ws-policy"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns:tns="http://ws.wa2.fel.cvut.cz/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata">
  <types>
    <xsd:schema>
      <xsd:import namespace="http://ws.wa2.fel.cvut.cz/"
schemaLocation="WA2WebServiceHello_schema1.xsd"/>
    </xsd:schema>
  </types>
  <message name="hello">
    <part name="parameters" element="tns:hello"/>
  </message>
  <message name="helloResponse">
    <part name="parameters" element="tns:helloResponse"/>
  </message>
  <portType name="WA2WebServiceHello">
    <operation name="hello">
      <input wsam:Action="http://ws.wa2.fel.cvut.cz/WA2WebServiceHello/helloRequest"
message="tns:hello"/>
      <output wsam:Action="http://ws.wa2.fel.cvut.cz/WA2WebServiceHello/helloResponse"
message="tns:helloResponse"/>
    </operation>
  </portType>
  <binding name="WA2WebServiceHelloPortBinding" type="tns:WA2WebServiceHello">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
    <operation name="hello">
      <soap:operation soapAction=""/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
    </operation>
  </binding>
```

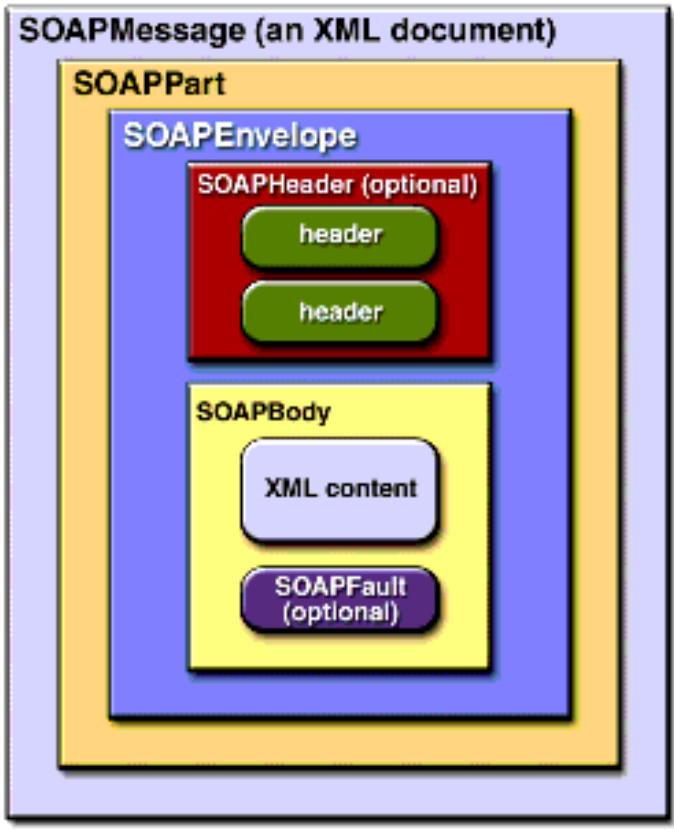
Test client


WA2WebServiceHello Web Service Tester
This form will allow you to test your web service implementation (WSDL File)
To invoke an operation, fill the method parameter(s) input boxes and click on the button labeled with the method name.
Methods :
public abstract java.lang.String cz.cvut.fel.wa2.ws.WA2WebServiceHello.hello(java.lang.String)
hello
abcd

SOAP Communication

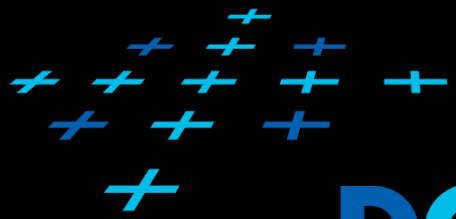
Request - client

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope
xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header/>
    <S:Body>
        <ns2:hello xmlns:ns2="http://ws.wa2.fel.cvut.cz/">
            <name>abcd</name>
        </ns2:hello>
    </S:Body>
</S:Envelope>
```



Response - server

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope
xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header/>
    <S:Body>
        <ns2:helloResponse
xmlns:ns2="http://ws.wa2.fel.cvut.cz/">
            <return>Hello abcd !</return>
        </ns2:helloResponse>
    </S:Body>
</S:Envelope>
```



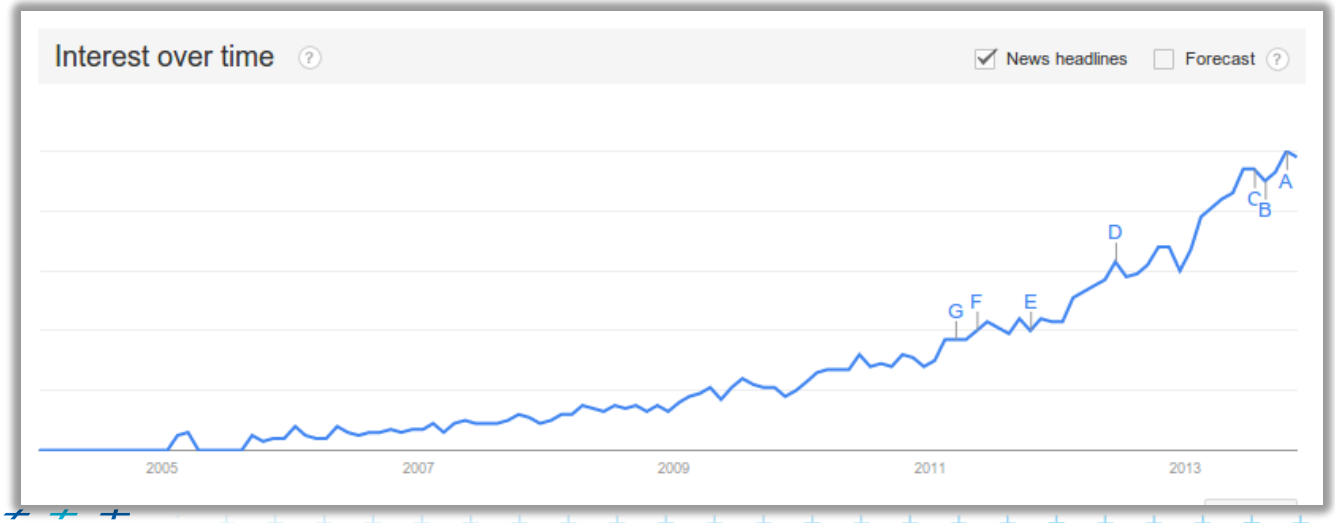
DCGI

KATEDRA POČÍTAČOVÉ GRAFIKY A INTERAKCE

REST web services

What is REST

- REST stands for Representational State Transfer
- REST is an architectural style for distributed hypermedia systems
- First described by Roy Thomas Fielding
- Google trends – REST API



DCGI

REST vs Web Services

REST

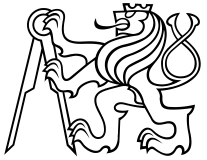
- In practice HTTP only
- No prescribed data format
- Lightweight
- Scalable: yes
- Defined API: yes
  WADL
- Describes resources

WS

- Protocol independent
- SOAP / XML
- Heavyweight
- Scalable: yes
- Defined API: yes
  WSDL
- Describes functions

Resources

- Resource is an abstraction of “peace of” information
- Resources is uniquely identified by URL
- A resource can be represented by an arbitrary format (XML, TXT, JSON, CSV, …)
- A representation of a resource is serialized data and metadata.



Clients

- Most of clients are thick clients

  …compare to thin web clients.
- Clients use content negotiation to get a resource representation.
- HTTP protocol is a perfect means to use.
  - HTTP methods denote operations on resource

CRUD

CREATE -> HTTP POST

creates a new resource and its identifier

READ -> HTTP GET

gets a representation of a resource

UPDATE -> HTTP PUT

updates or creates resource with given identifier

DELETE -> HTTP DELETE

permanent resource removal

All methods except POST are idempotent!

Other HTTP methods

READ METADATA -> HTTP HEAD

returns just HTTP headers (aka metadata), content/body is empty

METHOD DISCOVERY -> HTTP OPTION

returns list of allowed methods

PARTIAL UPDATE -> HTTP PATCH (RFC proposed)

in contrast to PUT this method allows update just some data