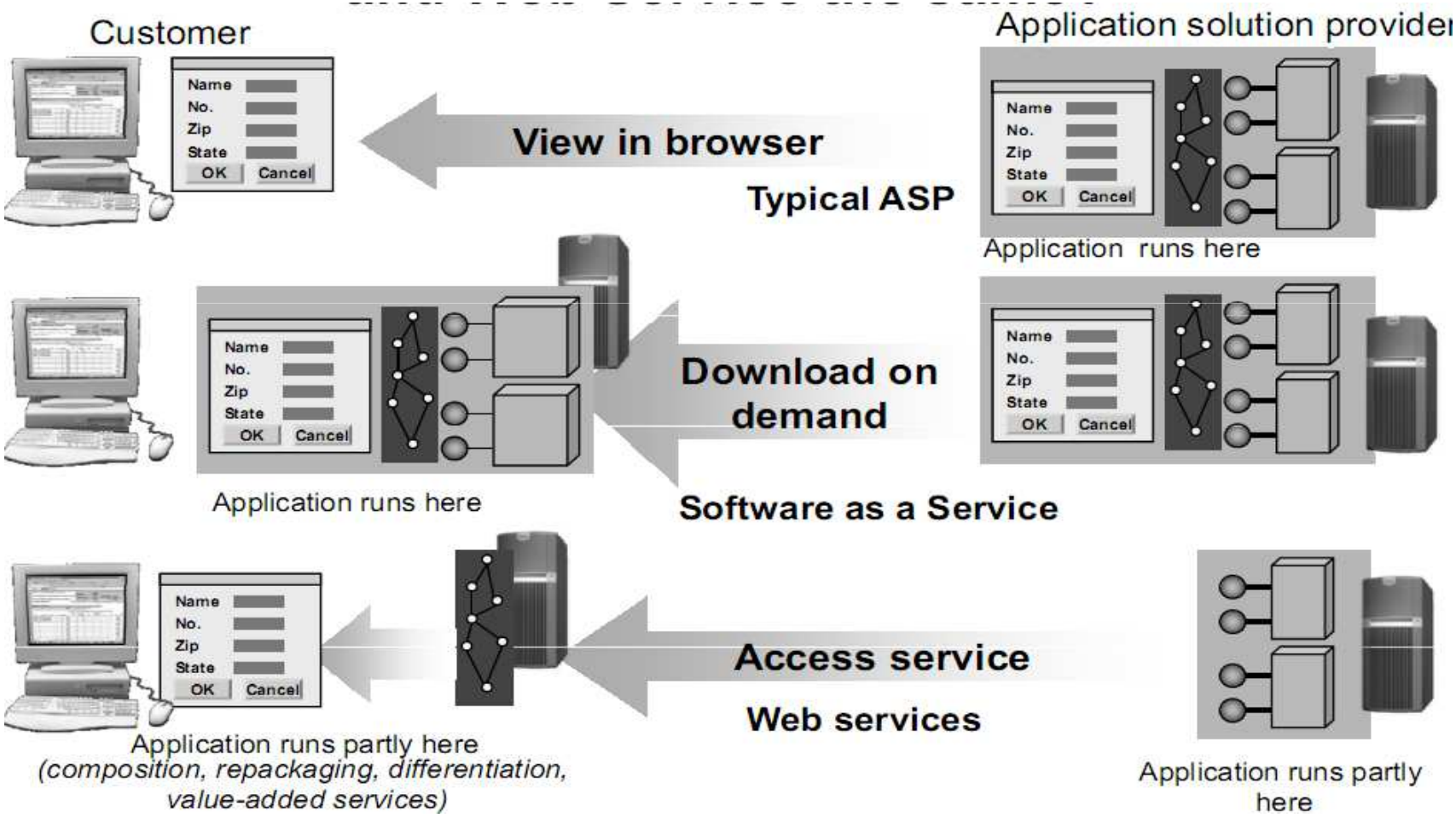
ATI 5 – Web Services (**SOAP**)

Antonius Rachmat C, S.Kom, M.Cs

Today's Web

- Web designed for application to **human interactions**
- It's purpose:
  - Information sharing: a distributed content library.
  - **Enabled B2C** e-commerce.
  - **Non-automated** interactions between applicaiton
- Standard
  - Built on very few standards: **http + html**
- The Web is everywhere.
  - E-marketplaces.
  - Business process **integration** on the Web.
- Current approach **is *ad-hoc* on top of existing standards.**
  - e.g., application-to-application interactions with HTML forms.
- How to: **enabling systematic application-to-application interaction on the Web.**

Application Solution Provider, Software as a Service, and Web Service



Web Service can…

- **Web Services can convert your applications into Web-application service.**
  - By using Web services, your application can **publish** its **function** or message to the rest of the world.
- **Web Services can be used by other applications.**
  - Ex: with Web services your accounting department's Win 2k servers can connect with your IT supplier's UNIX server.
  - Can be used as **Web API**
  - Interacting *among* application

Pengertian-pengertian WS

- **Generic**: any application accessible to other applications over the Web
  - Very open definition, almost any URL becomes a “Web service” under this definition
- **UDDI consortium**: web services are self-contained, modular business applications that have open, Internet-oriented, standards-based interfaces
  - Better, but still not specific enough
- **W3C**: A Web service is a software application identified by a URI, whose interfaces and bindings are capable of being defined, described, and discovered as XML.
  - A Web service supports direct interactions with other software agents using XML-based messages exchanged via Internet-based protocols
  - Slightly better

Pengertian-pengertian WS

- **Wikipedia**: according to the W3C, a Web service is a software system designed to support *interoperable machine-to-machine* interaction over a network.
- It has an interface that is described in a machine-process-able format such as **WSDL**.
- Software applications written in *various programming languages* and running *on various platforms* can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer.
  - Much more specific

Designing Web Services I

- **Goals**
  - Enable universal **interoperability**.
  - Widespread adoption, ubiquity, fast!
  - Enable (Internet scale) **dynamic binding**.
    - Support a service oriented architecture (**SOA**).
  - Efficiently support both **open** (Web) and more constrained environments.

Designing Web Services II

- **Requirements**
  - Based on **standards**
  - **Minimal** amount of required infrastructure is assumed.
    - Only a minimal set of standards must be implemented.
  - **Very low** level of application integration is expected.
  - Focuses on **messages and documents**

WS Example

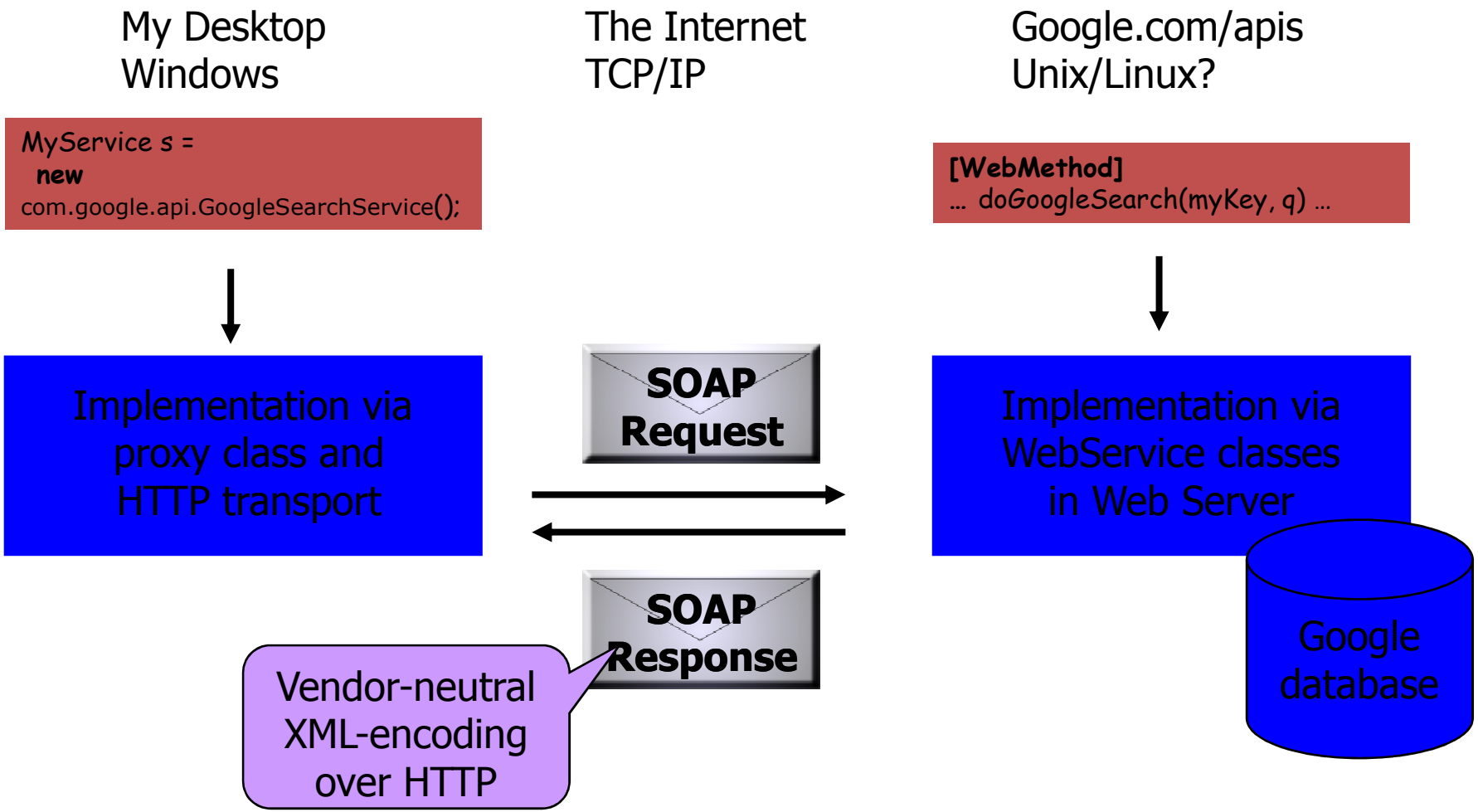
- http://www.google.com/apis/
- http://terraserver.microsoft.net/TerraService.asmx
- http://www.xmethods.net
- http://soap.amazon.com/schemas2/AmazonWebServices.wsdl
- http://api.google.com/GoogleSearch.wsdl

Example: A Google Client

- Create a local proxy class, instantiate, and invoke
- The proxy class MyService generated from a WSDL file, an XML-encoded service description

```
Dim MyLicenseKey As String ' Variable to Store the License Key
' Declare variable for the Google search service
Dim MyService As com.google.api.GoogleSearchService = New _
                    com.google.api.GoogleSearchService
' Declare variable for the Google Search Result
Dim MyResult As com.google.api.GoogleSearchResult
' Please Type your license key here
MyLicenseKey = "tGCTJkYos3YItLYzI9Hg5quBRY8bGqiM"
' Execute Google search on the text enter and license key
MyResult = MyService.doGoogleSearch(MyLicenseKey, _
                TextBox1.Text, 0, 1, False, "", False, "", "")
' output the total Results found
Label2.Text = "Total Found : " & _
                    CStr(MyResult.estimatedTotalResultsCount)
```

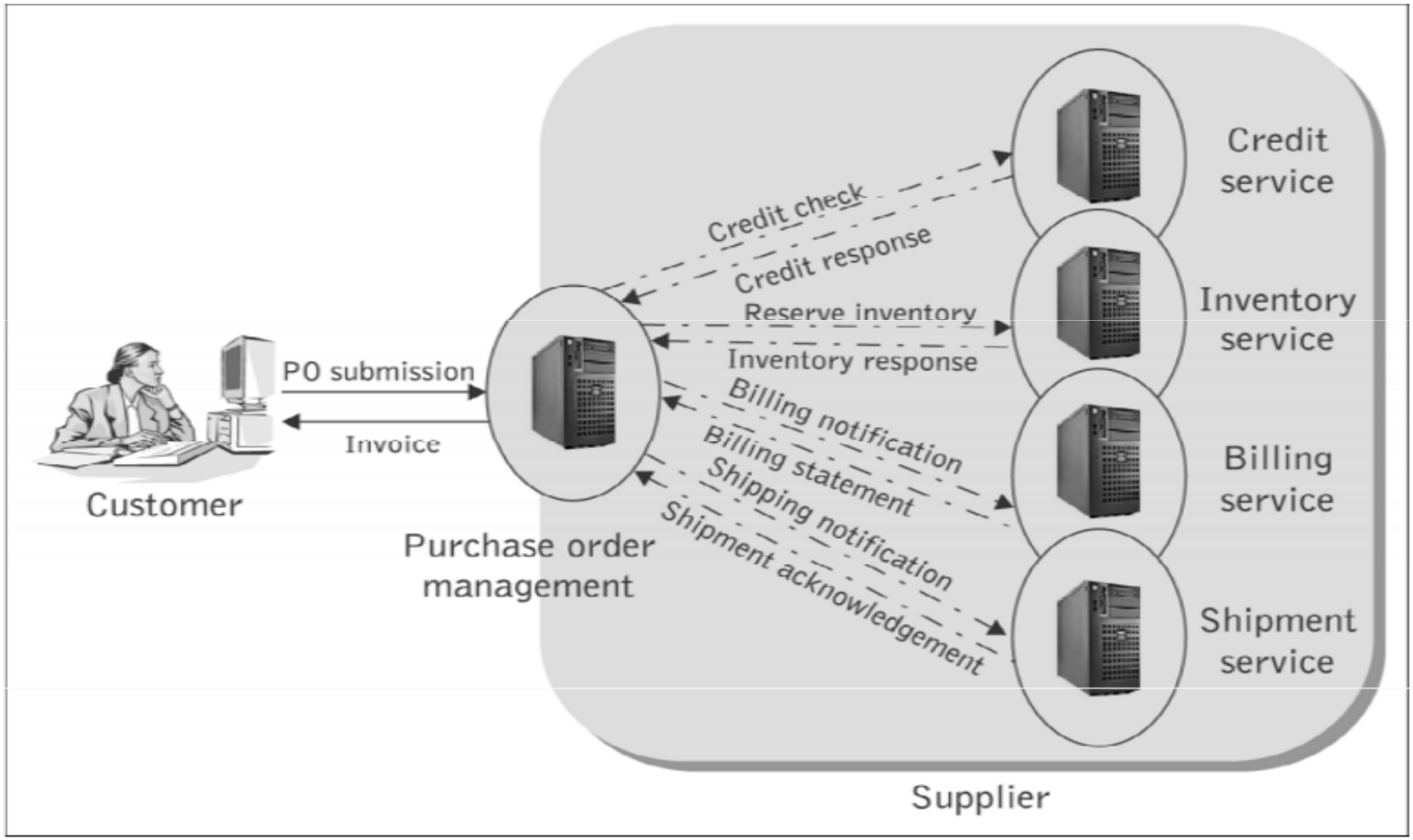
Outline Architecture



Google WSDL

```
<message name="doGetCachedPage">
  <part name="key"            type="xsd:string"/>
  <part name="url"            type="xsd:string"/>
</message>

<message name="doGetCachedPageResponse">
  <part name="return"         type="xsd:base64Binary"/>
</message>

<message name="doSpellingSuggestion">
  <part name="key"            type="xsd:string"/>
  <part name="phrase"         type="xsd:string"/>
</message>

<message name="doSpellingSuggestionResponse">
  <part name="return"         type="xsd:string"/>
</message>

<!-- note, ie and oe are ignored by server; all traffic is UTF-8. -->

<message name="doGoogleSearch">
  <part name="key"            type="xsd:string"/>
  <part name="q"              type="xsd:string"/>
  <part name="start"          type="xsd:int"/>
  <part name="maxResults"     type="xsd:int"/>
  <part name="filter"         type="xsd:boolean"/>
  <part name="restrict"       type="xsd:string"/>
  <part name="safeSearch"     type="xsd:boolean"/>
  <part name="lr"             type="xsd:string"/>
  <part name="ie"             type="xsd:string"/>
  <part name="oe"             type="xsd:string"/>
</message>
```

Contoh penggunaan web service



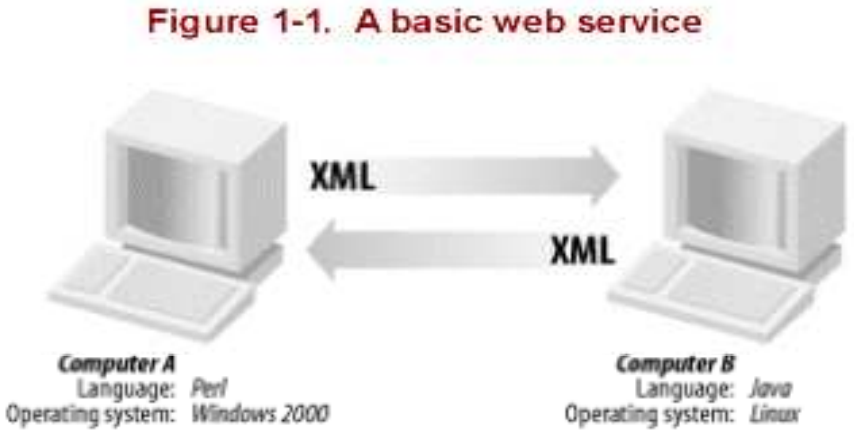
Web Service

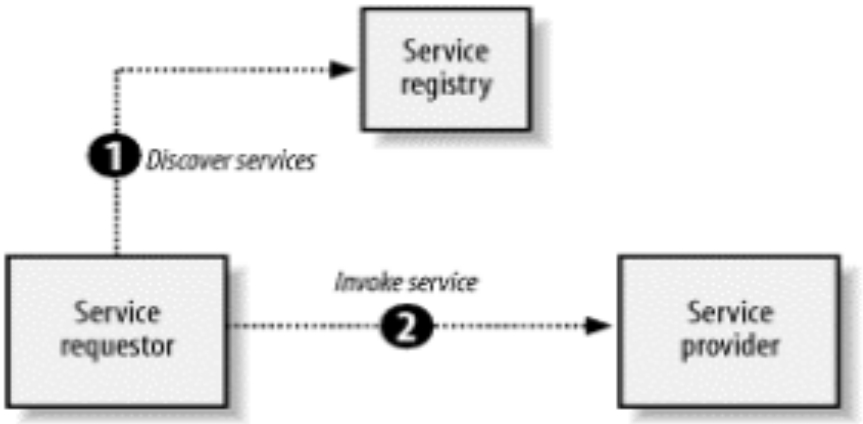
- Mempertukarkan data dalam format **XML**.
- Tersedia dan dikomunikasikan melalui **Internet** atau intranet.
- Bersifat operating system/programming language **independent**.
- Berupa web application yang **tidak memiliki** web interface.
- Web service mempertukarkan data antara **service requestor**) dan **service provider** menggunakan **service registry**, dengan salah satu teknologi:
  - XML-RPC
  - SOAP
  - REST

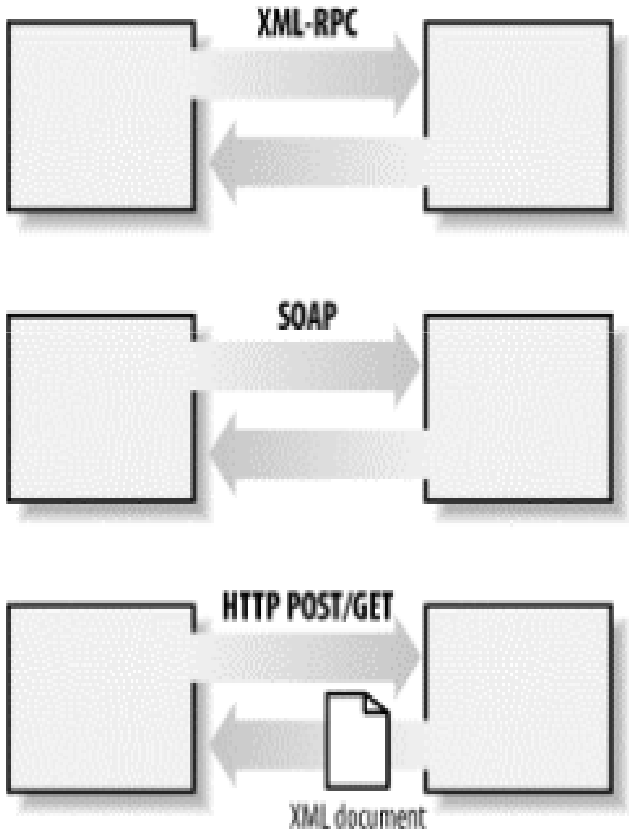
Web Service

- Web service mempertukarkan data dalam format **message** yang berarti tidak bersifat **binary**
- Web service dapat dipanggil/digunakan melalui *web, aplikasi desktop, ataupun aplikasi mobile*
- Kelebihan:
  - **Interoperability** (platform dan aplikasi)
  - Dapat **mempublikasikan** service dan method sehingga mudah digunakan
  - Mendukung **reusable-components**

Bentuk Web Services

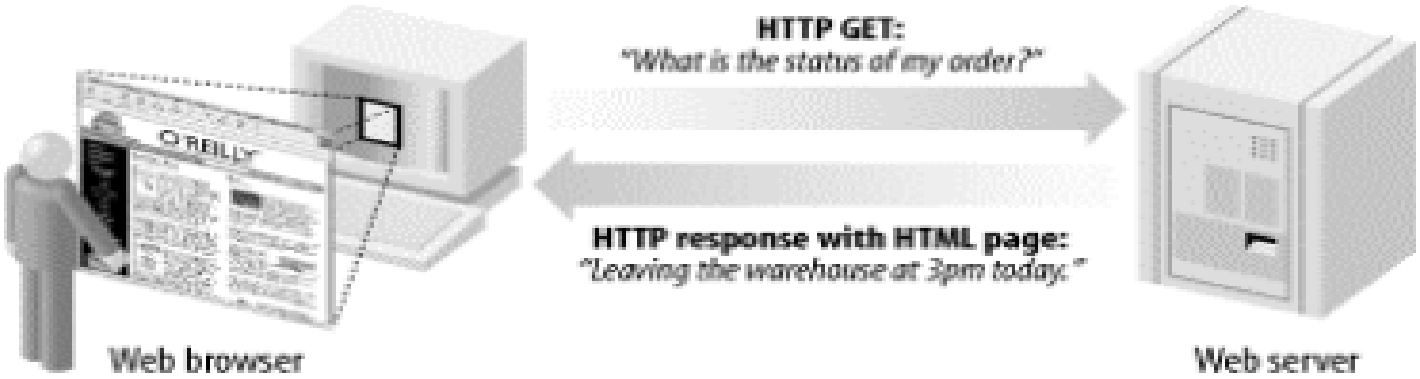






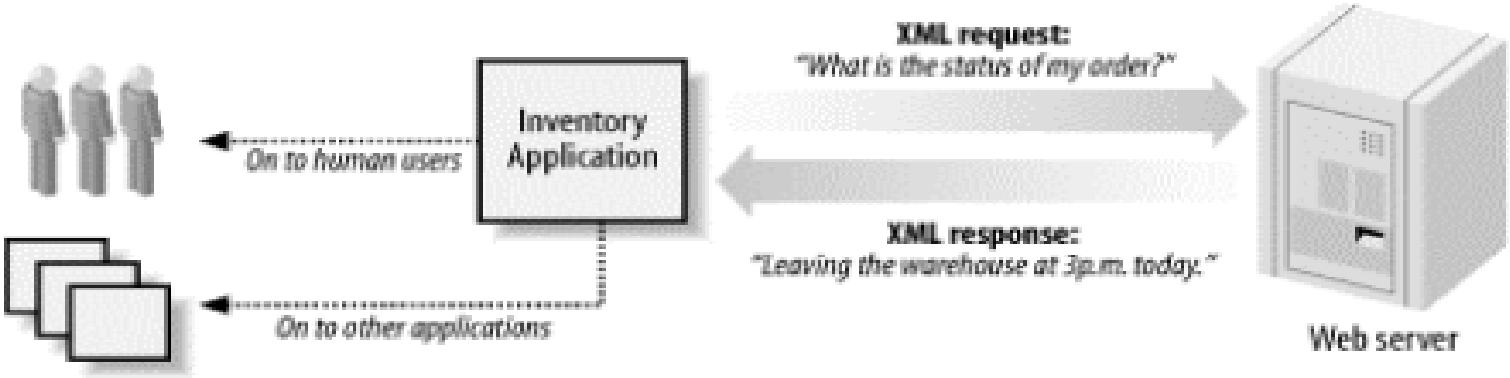
Website vs Web service

- Memiliki web interface
- Dibuat untuk berinteraksi langsung dengan user
- Dibuat untuk bekerja pada web browser
- Bersifat **front-end**
- Bersifat **human-centric**: manusia menjadi aktor utama



Website vs Web Service

- Tidak memiliki interface yang baik
- Dibuat untuk berinteraksi langsung dengan applikasi yang lain, baik berbeda OS sekalipun, bukan dengan user.
- Dibuat untuk bekerja pada semua tipe client applikasi / perangkat device
- Bersifat **behind the scene**.
- Bersifat **application-centric**: komunikasi terjadi antar aplikasi



Web Service properties

- Web service harus ”**self-describing**”: jika kita membuat web service, kita harus juga mempublikasikan **public interface** yang dapat dimengerti dan dipakai.
  - Minimalnya, service harus memiliki **human-readable documentation** sehingga developer lain dapat mengintegrasikan aplikasinya dengan web service yang kita buat.
- Web service harus ”**discoverable”**: jika kita membuat web service, harus ada mekanisme sederhana agar service dan public method yang kita buat dapat **dikenal** dan **ditemukan** oleh aplikasi lain.

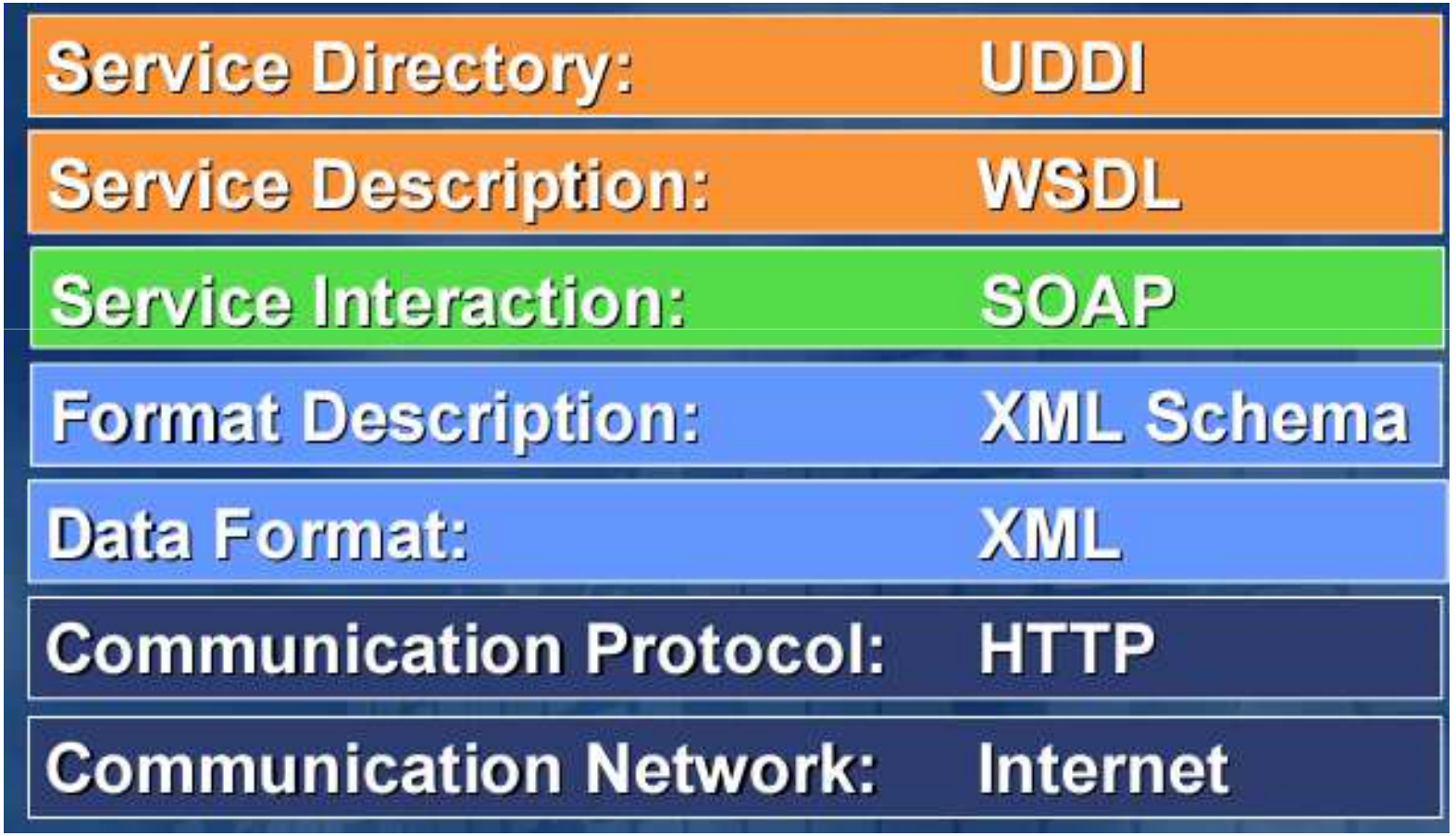
WS Layer

- **Service transport**
  - Bagian ini bertanggung jawab untuk **mengirim message** antar aplikasi.
  - Internet Protokol yang ada di bagian ini: Hyper Text Transfer Protocol (**HTTP**)
- **XML messaging dan encoding/decoding**
  - Bagian ini bertanggung jawab untuk **mengencode/mendecode message** dalam format XML sehingga message dapat dimengerti dan dipertukarkan.
  - Protokol/Service Interaction yang ada di bagian ini: **XML-RPC dan SOAP.**

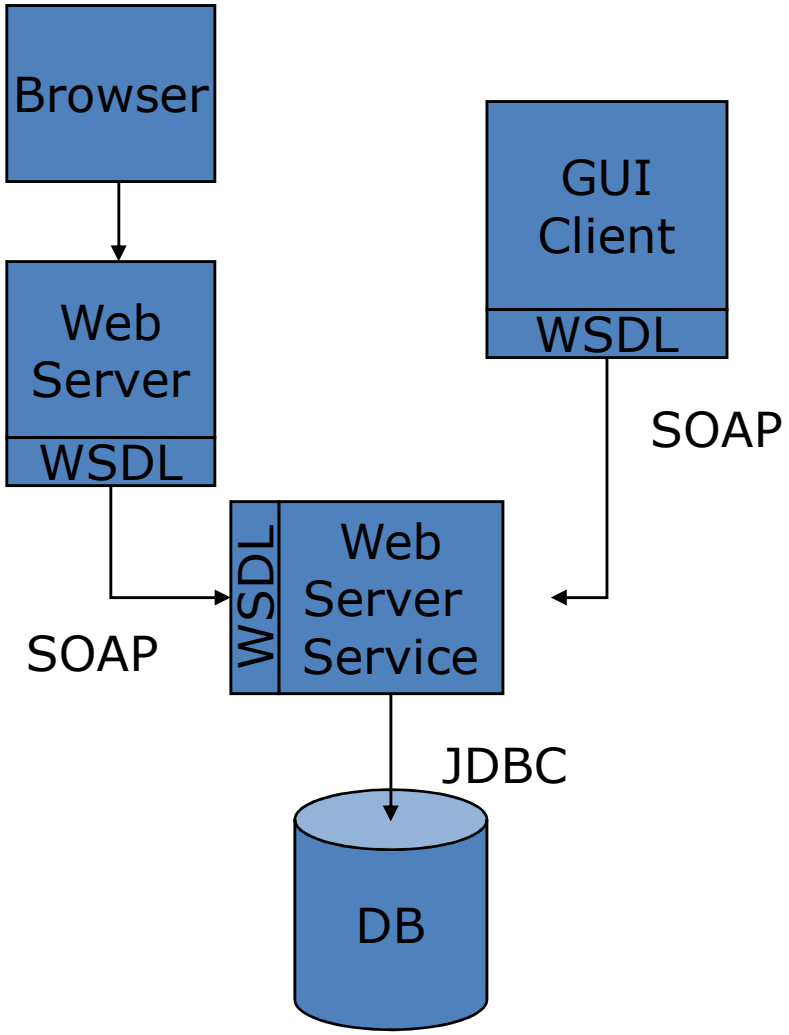
WS Layer

- **Service description**
  - Bagian ini bertanggung jawab untuk **mendeskripsikan public interface** sesuai dengan web service yang spesifik.
  - Bagian ini dihandle melalui Service Description: Web Service Description Language (**WSDL**) dan **XML Schema**
- **Service discovery**
  - Bagian ini bertanggung jawab untuk **mengumpulkan services ke dalam common registry dan menyediakan kemudahan untuk mempublikasikan interface dan kemudahan dalam pencarian method**.
  - Bagian ini dihandle oleh Service Directory: Universal Description, Discovery, and Integration (**UDDI**).

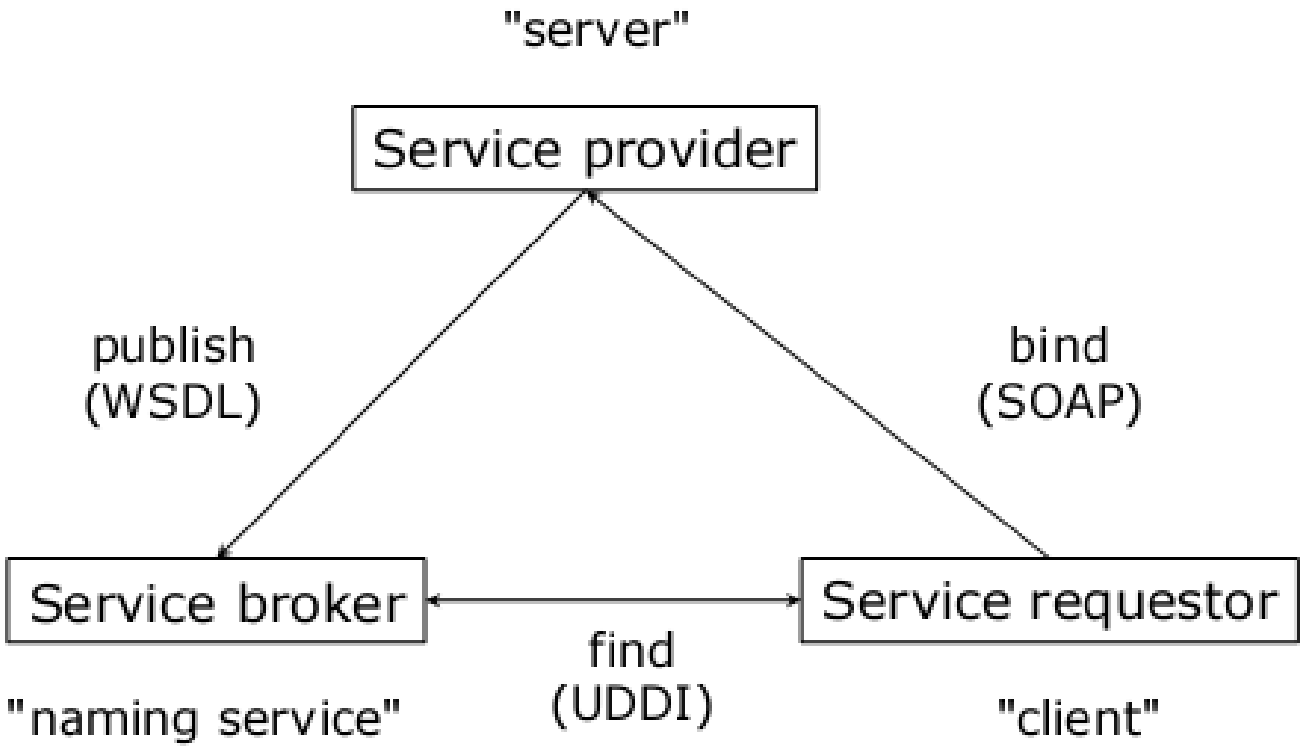
Pondasi Teknologi Web Service



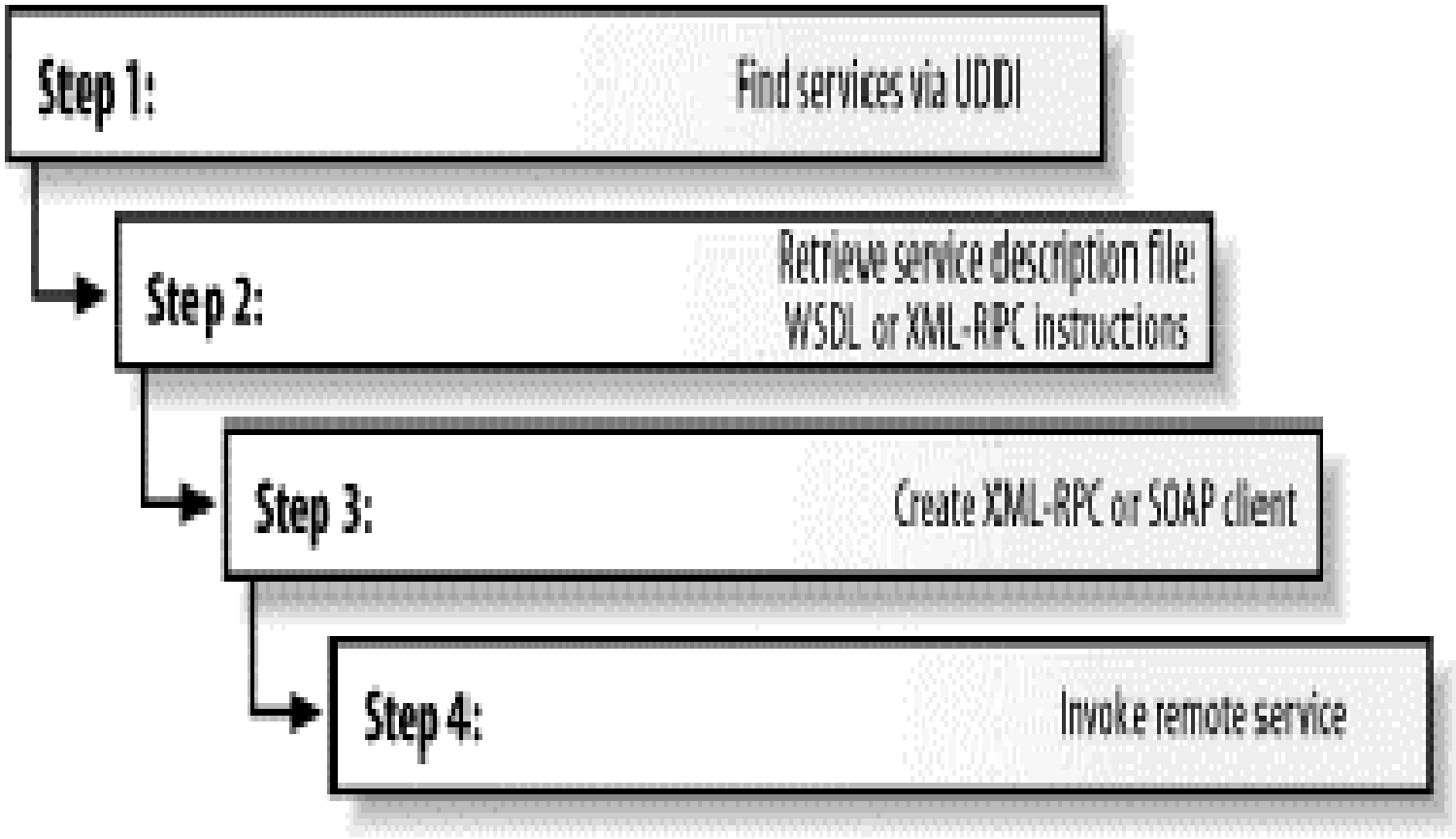
Web service architecture (1)



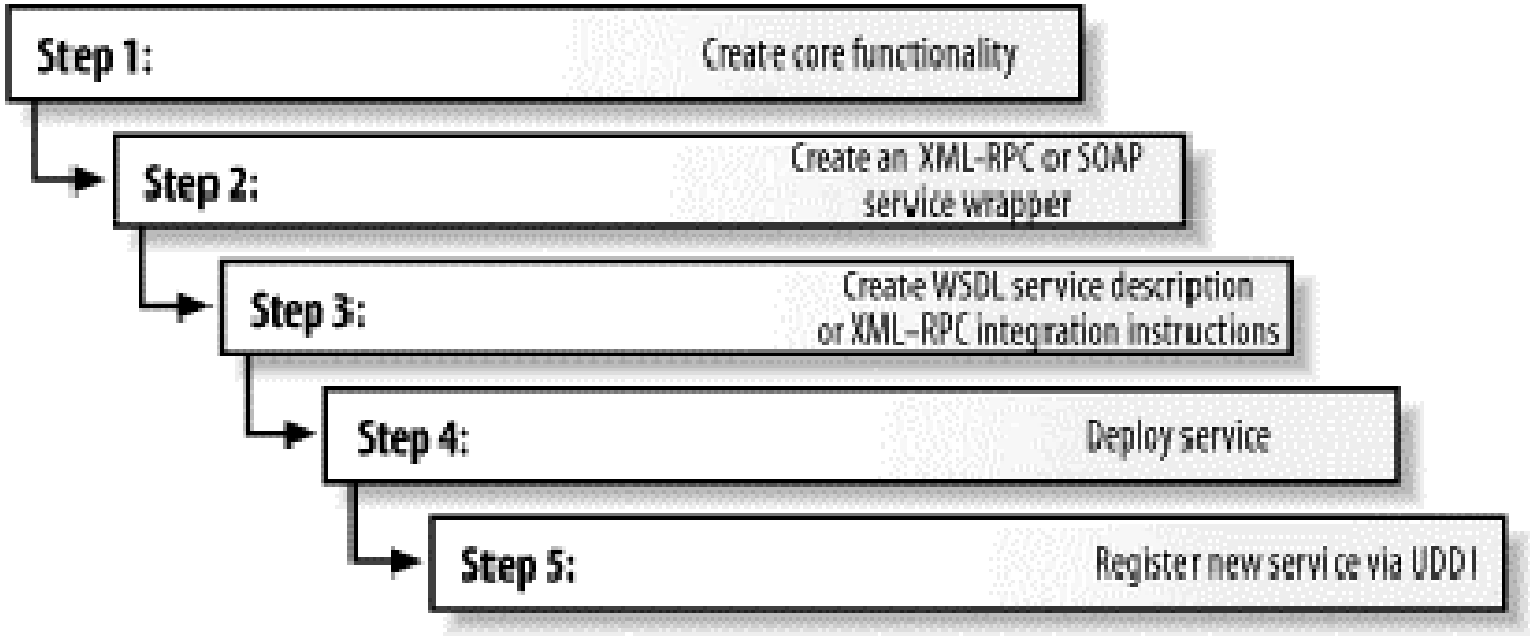
WS Architecture (2)



Pengembangan sisi Client



Pengembangan sisi Server



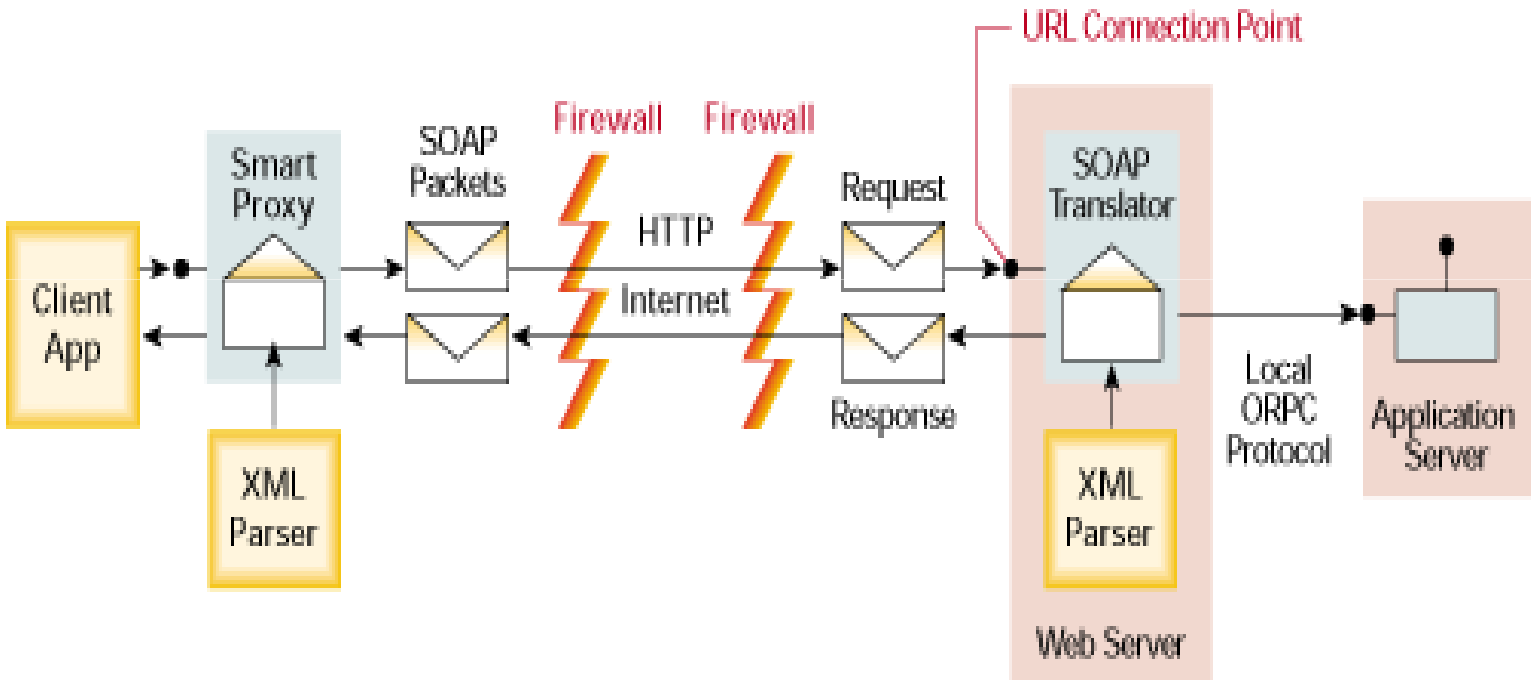
Service Broker / Service Registry

- The service brokers allow service providers to publish their services (**register, save, and categorize**).
- They provide also mechanisms to **locate** services and their providers
- Diimplementasikan pada **UDDI (Universal Description Discovery and Integration)**

SOAP (Simple Object Access Protocol)

- SOAP merupakan **protokol komunikasi berbasis** XML yang memperbolehkan aplikasi **saling bertukar informasi** melalui HTTP
- SOAP merupakan salah satu protokol yang menangani **web service**
- SOAP merupakan **format** untuk mengirimkan message melalui Internet, bersifat platform independent, language independent, dan merupakan **standar W3C**
- SOAP membungkus **request & response XML**

SOAP call anatomy



SOAP Skeleton

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Header>
...
</soap:Header>

<soap:Body>
...
  <soap:Fault>
  ...
  </soap:Fault>
</soap:Body>

</soap:Envelope>
```

Elemen SOAP

- Elemen **Envelope** yang mengidentifikasi XML dokumen sebagai SOAP message (wajib)
  - Top element of the XML document representing the **message**
- Elemen **Header** yang berisi informasi header (opsional)
  - Determines how a recipient of a SOAP message should process the message
  - Adds features to the SOAP message such as authentication, message routes, additional information, etc…
- Elemen **Body** yang berisi informasi call dan response (wajib)
- Elemen **Fault** yang berisi informasi error yang terjadi (opsional)

Plus-Minus SOAP

Plus

- Uses HTTP which is widely used and scalable
- Flexible for growth because of XML properties
- Data in String message

Minus

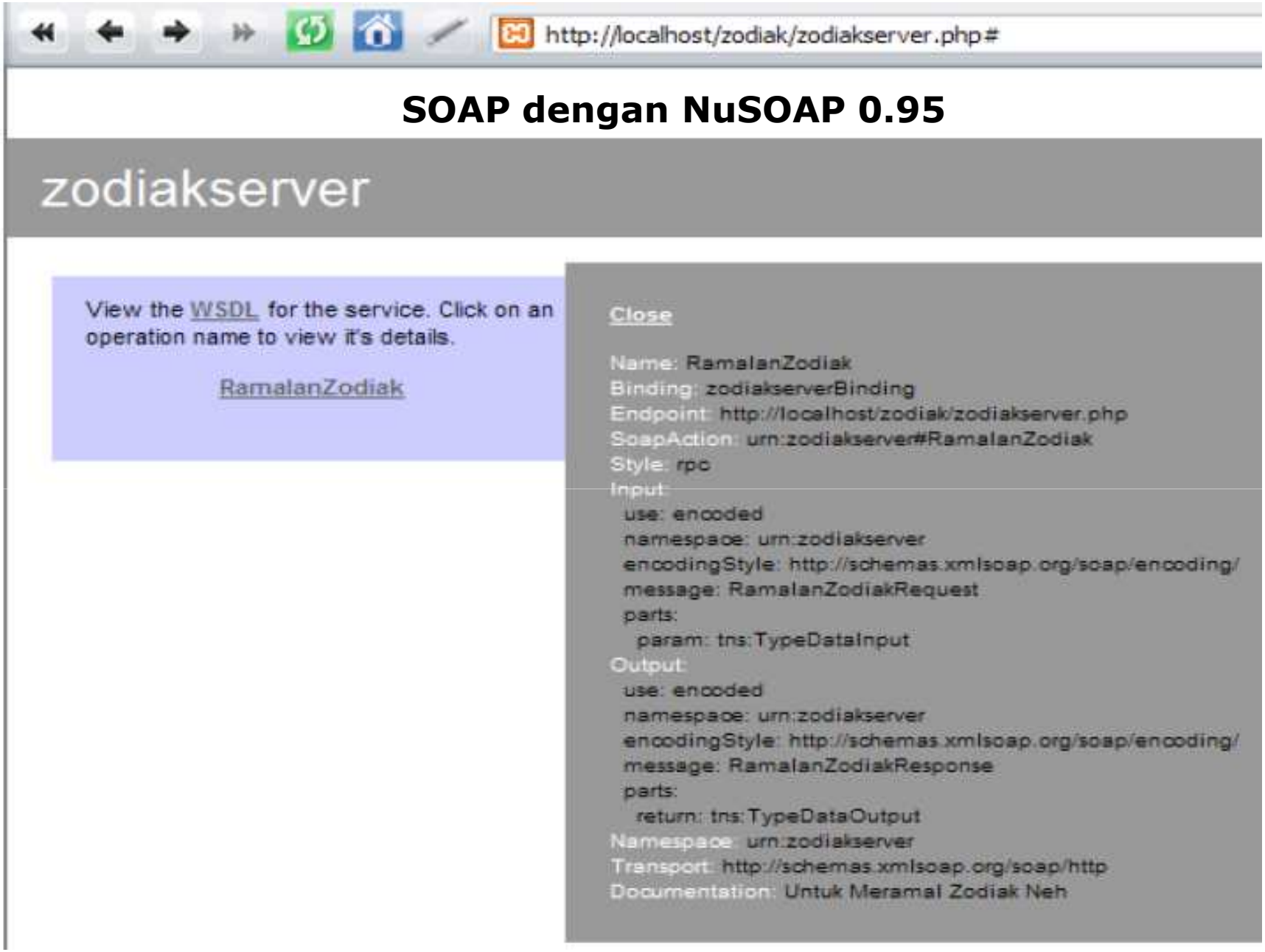
- Parsing of SOAP packet and mapping to objects **reduces performance**
- If XML data are **too long**
- **Doesn’t implement security** because it is a wire protocol—relies on HTTP

Request Message

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:ns1="urn:testns“
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

        <SOAP-ENV:Body>
            <ns1:getProfessor>
                <course>470</course>
            </ns1:getProf>
        </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

Response Message

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:ns1="urn:testns"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

        <SOAP-ENV:Body>
            <ns1:getProfessorResponse>
                <Result>Antonie</Result>
            </ns1:getProfResponse>
        </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```


http://localhost/zodiak/zodiakserver.php#
SOAP dengan NuSOAP 0.95
zodiakserver
View the WSDL for the service. Click on an operation name to view it's details.
RamalanZodiak
Close
Name: RamalanZodiak
Binding: zodiakserverBinding
Endpoint: http://localhost/zodiak/zodiakserver.php
SoapAction: urn:zodiakserver#RamalanZodiak
Style: rpc
Input:
use: encoded
namespace: urn:zodiakserver
encodingStyle: http://schemas.xmlsoap.org/soap/encoding/
message: RamalanZodiakRequest
parts:
param: tns:TypeDataInput
Output:
use: encoded
namespace: urn:zodiakserver
encodingStyle: http://schemas.xmlsoap.org/soap/encoding/
message: RamalanZodiakResponse
parts:
return: tns:TypeDataOutput
Namespace: urn:zodiakserver
Transport: http://schemas.xmlsoap.org/soap/http
Documentation: Untuk Meramal Zodiak Neh

Web Services Description Language

- Provides functional **description** of network services:
  - Method description
  - Protocol details
- A short history:
  - WSDL v1.0, 9/2000
  - WSDL v1.1 submitted to W3C 3/2001.
  - A *de facto* industry standard.

The Main Structure of WSDL

<definition **namespace** = “http/… “>

Bagian Abstract

<**type**> xschema types </type>

<**message**> … </message>

<**port**> a set of operations </port>

Bagian Konkret

<**binding**> communication protocols </binding>

<**service**> a list of binding and ports </service>

<definition>

UDDI Overview

- UDDI defines the **operation** and **data structures** of a service **registry**:
  - **Data structures** for registering
    - Businesses
    - Technical specifications
    - Service and service endpoints
  - SOAP Access **API**
  - **Rules** for the operation of a global registry

How UDDI Works

1. 

SW companies, standards bodies, and programmers populate the registry with descriptions of different types of services

2.

Businesses populate the registry with descriptions of the services they support

UDDI Business Registry

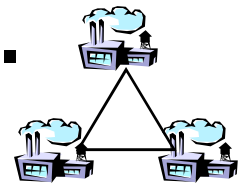
Business Registrations

Segrvice Type Reistrations

Programmer assigns a programmatically unique identifier to each service and business registration

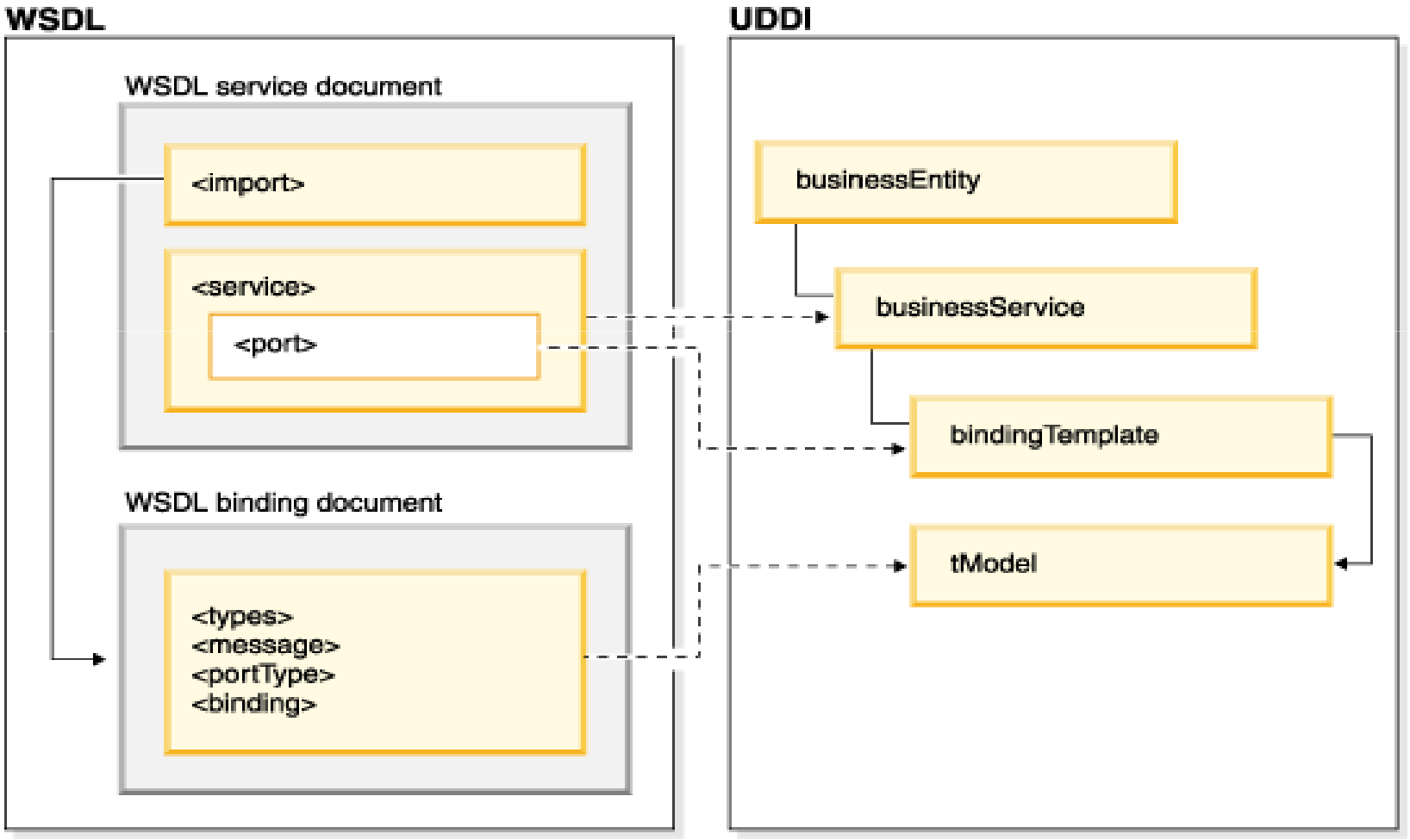
4. 

Marketplaces, search engines, and business apps query the registry to discover services at other companies

5. 

Business uses this data to facilitate easier integration with each other over the Web

WSDL and UDDI relationship



And now... Some Services

- Submitjob *(wsdl)*
  - test
  - execLocalCommand
  - execRemoteCommand
- ApplicationInstance3 *(wsdl)*
  - getHostName
  - setEmail
  - getInputDescription
  - getOutputDescription
  - getErrorDescription
  - getQueueType
  - getQsubPath
  - setApplicationName
  - setJobName
  - setNumberOfCPUs
  - setWalltime
  - getJobName
  - getNumberOfCPUs
  - getWalltime
  - getApplicationName
  - readApplIns
  - createQueueInstance
  - createHostInstance
  - createApplicationInstance
  - writeApplIns
  - setMemoryOption
  - getApplInsString
  - getInputLocation
  - getOutputLocation
  - getErrorLocation
  - getMemoryOption
- Remotefile *(wsdl)*
  - writeFile
  - readFile
- AdminService *(wsdl)*
  - AdminService
- Version *(wsdl)*
  - getVersion
- SOAPMonitorService *(wsdl)*
  - publishMessage
- ContextManager *(wsdl)*

Web Services Example

```
<?xml version="1.0" encoding="UTF-8" ?>
- <wsdl:definitions targetNamespace="http://grids.ucs.indiana.edu:8045/GCWS/services/Submitjob" xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:apachesoap="http://xml.apache.org/xml-soap" xmlns:impl="http://grids.ucs.indiana.edu:8045/GCWS/services/Submitjob"
    xmlns:intf="http://grids.ucs.indiana.edu:8045/GCWS/services/Submitjob" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  - <wsdl:types>
    - <schema targetNamespace="http://grids.ucs.indiana.edu:8045/GCWS/services/Submitjob" xmlns="http://www.w3.org/2001/XMLSchema">
        <import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
      - <complexType name="ArrayOf_xsd_string">
        - <complexContent>
          - <restriction base="soapenc:Array">
              <attribute ref="soapenc:arrayType" wsdl:arrayType="xsd:string[]" />
            </restriction>
          </complexContent>
        </complexType>
        <element name="ArrayOf_xsd_string" nillable="true" type="impl:ArrayOf_xsd_string" />
      </schema>
    </wsdl:types>
  - <wsdl:message name="execLocalCommandResponse">
      <wsdl:part name="execLocalCommandReturn" type="impl:ArrayOf_xsd_string" />
    </wsdl:message>
  - <wsdl:message name="testResponse">
      <wsdl:part name="testReturn" type="xsd:string" />
    </wsdl:message>
  - <wsdl:message name="execLocalCommandRequest">
      <wsdl:part name="in0" type="xsd:string" />
    </wsdl:message>
    <wsdl:message name="testRequest" />
  - <wsdl:message name="execRemoteCommandResponse">
      <wsdl:part name="execRemoteCommandReturn" type="impl:ArrayOf_xsd_string" />
    </wsdl:message>
  - <wsdl:message name="execRemoteCommandRequest">
      <wsdl:part name="in0" type="xsd:string" />
      <wsdl:part name="in1" type="xsd:string" />
      <wsdl:part name="in2" type="xsd:string" />
      <wsdl:part name="in3" type="xsd:string" />
    </wsdl:message>
  - <wsdl:portType name="SJwsImp">
    - <wsdl:operation name="test">
        <wsdl:input message="impl:testRequest" name="testRequest" />
        <wsdl:output message="impl:testResponse" name="testResponse" />
      </wsdl:operation>
    - <wsdl:operation name="execLocalCommand" parameterOrder="in0">
        <wsdl:input message="impl:execLocalCommandRequest" name="execLocalCommandRequest" />
        <wsdl:output message="impl:execLocalCommandResponse" name="execLocalCommandResponse" />
      </wsdl:operation>
    - <wsdl:operation name="execRemoteCommand" parameterOrder="in0 in1 in2 in3">
        <wsdl:input message="impl:execRemoteCommandRequest" name="execRemoteCommandRequest" />
```

WSDL generated by inspecting the Java Implementation of WS

Demo

- MathWebService SOAP in VB.NET

Next

- Cloud Computing