Web Service

Aditya Wikan Mahastama

Kondisi Web Sekarang Ini

- Web bisa digunakan untuk berbagai kebutuhan:

  - Berbagi informasi (distribusi konten)
  - Perdagangan B2C

- Tetapi:

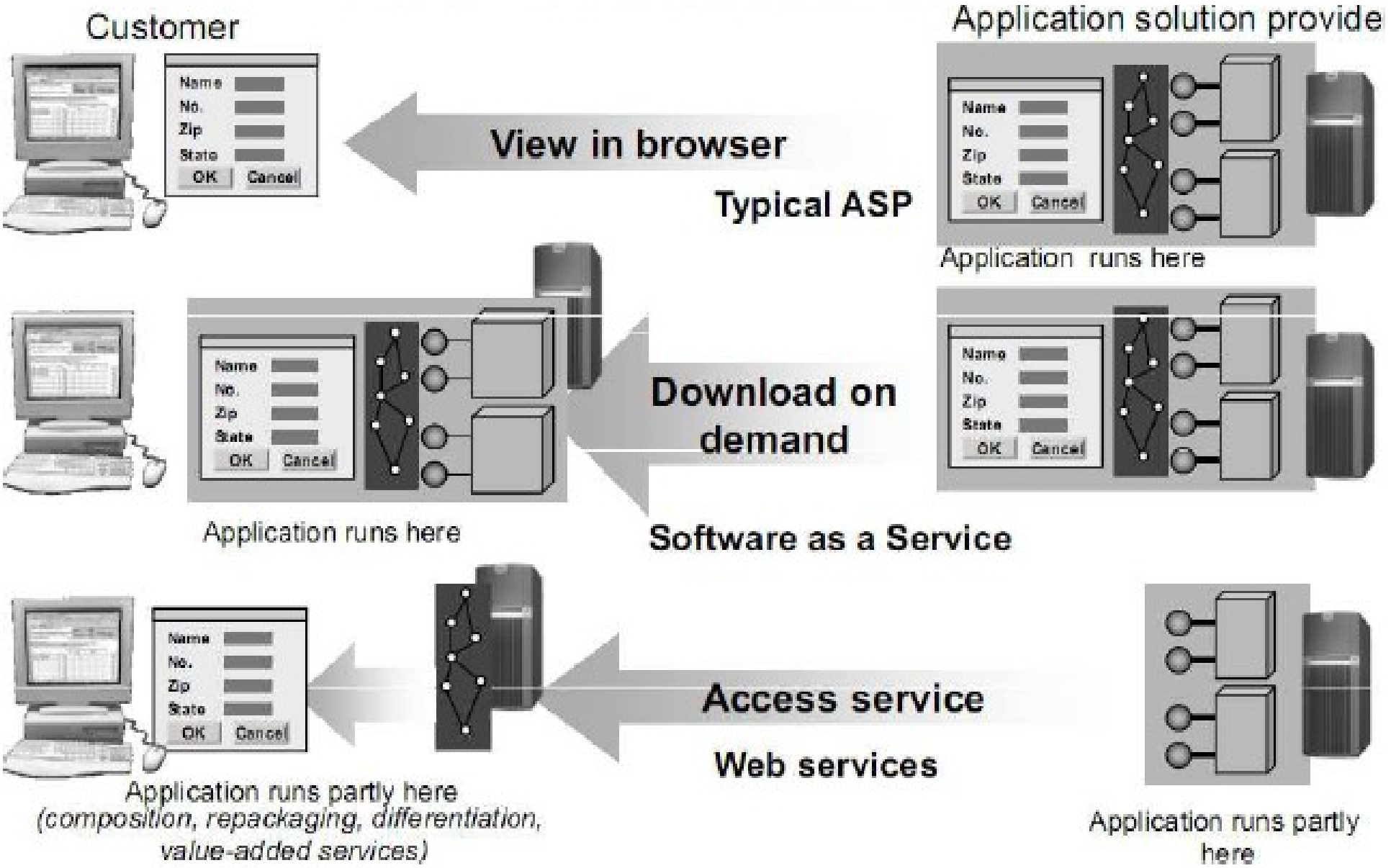
  - Dibangun pada standar HTTP & HTML saja
  - Interaksi antar aplikasi dengan menggunakan form HTML (data dikirimkan ke aplikasi yang akan mengolahnya) → belum optimal

  - Belum ada interaksi sistematis antar-aplikasi di Web

Harapan Manfaat Web Masa Kini

- Sebuah aplikasi bisa mempublikasikan fungsi dan pesan yang dimiliki ke seluruh dunia
- Fungsi dan pesan tersebut (service) dapat juga digunakan oleh aplikasi lainnya, dan harus multiplatform (misal antara aplikasi pada server UNIX dan Windows)
- Kenapa web? Karena melalui web mayoritas komputer di seluruh dunia biasa terhubung (melalui HTTP) → digagaslah WEB SERVICE

Tujuan Utama: Efisiensi Resource



Definisi Web Service

Umum/Generik:

- Sebuah aplikasi yang dapat diakses oleh alikasi lain melalui Web (berarti di sini URL pun masuk sebagai web service → terlalu umum)

Menurut Konsorsium UDDI:

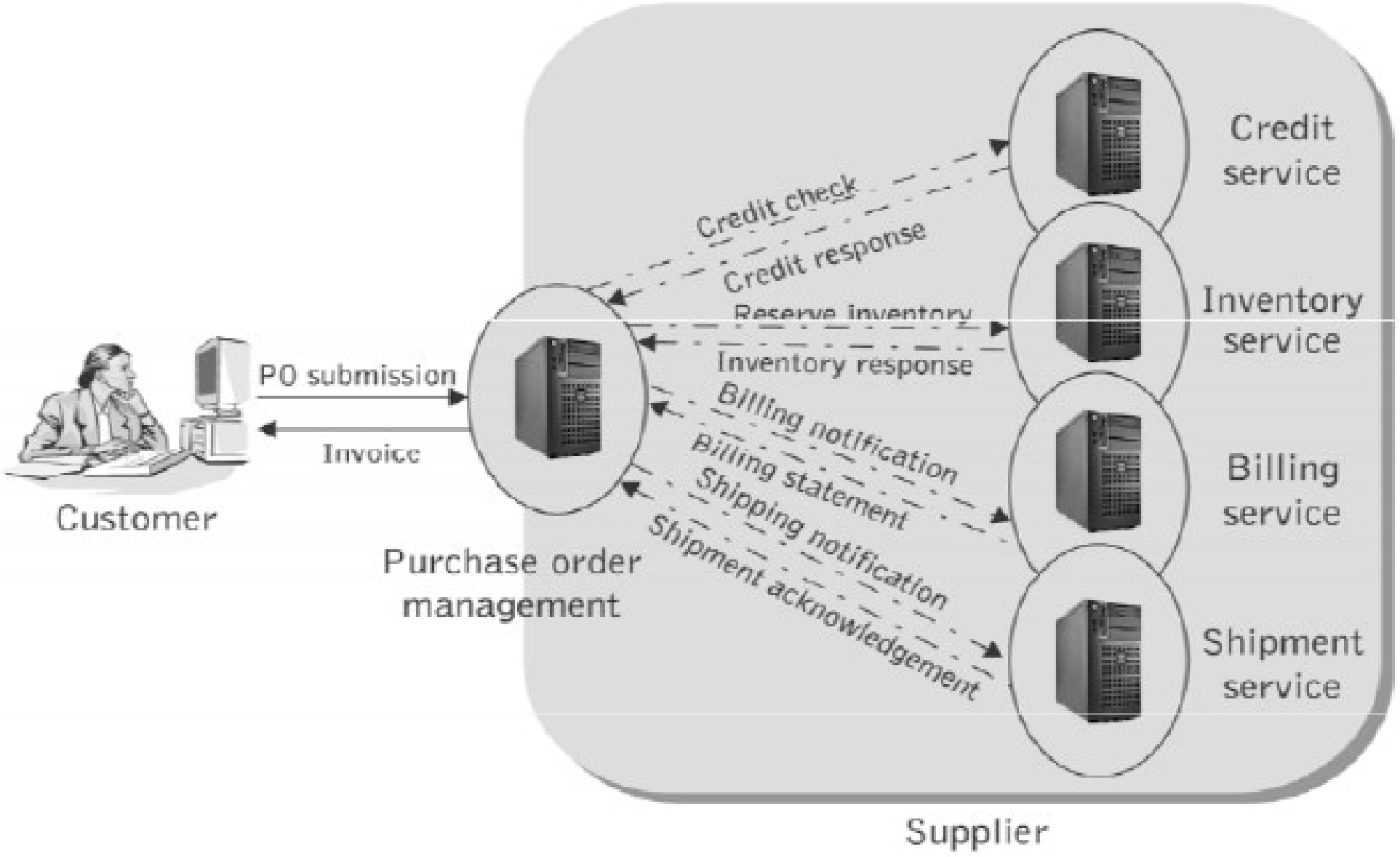
- Web services are self-contained, modular business applications that have open, Internet-oriented, standards-based interfaces (lebih baik, tetapi masih kurang spesifik)

Definisi Web Service

Menurut 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

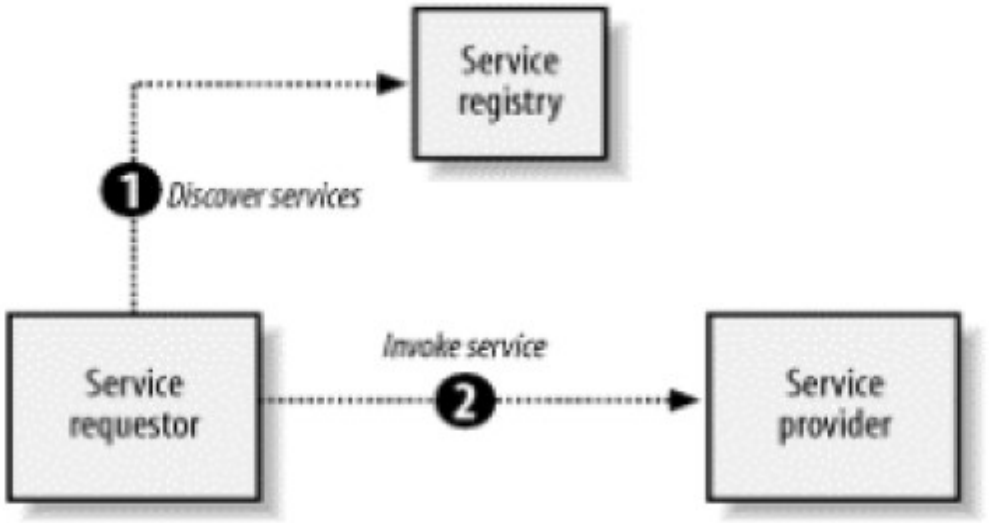
Sehingga, hal seperti ini dimungkinkan:

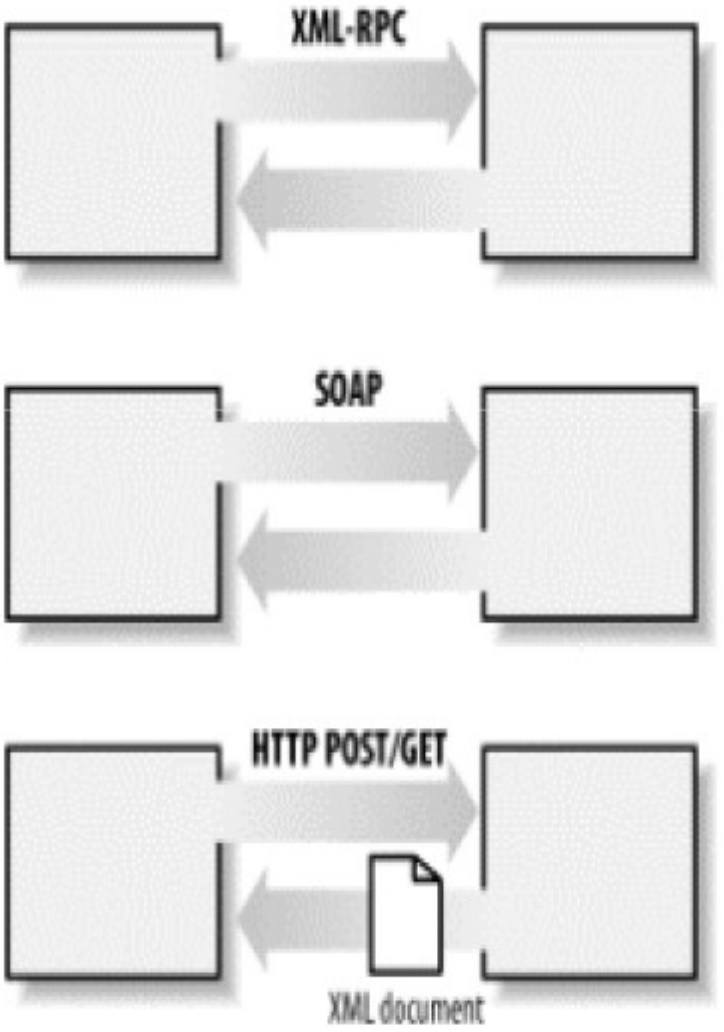


Jadi, Apa itu Web Service?

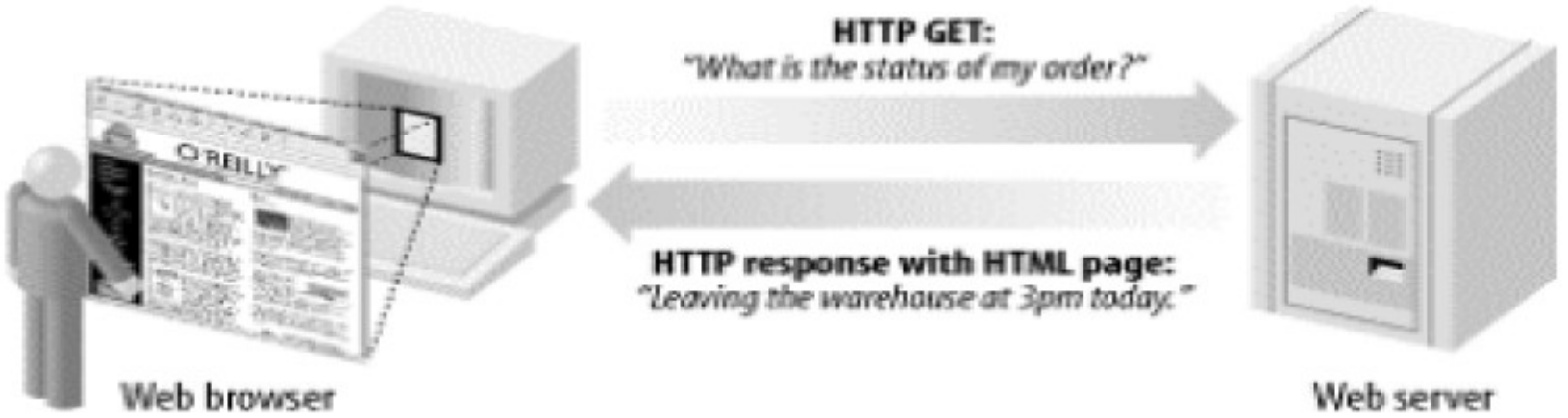
- Service yang mempertukarkan data dalam format **XML message** yang **non-binary** melalui jaringan, menggunakan HTTP
- Bersifat **OS, platform dan prog.language-independent** (bisa diakses oleh aplikasi web, desktop ataupun mobile)
- Penyedia berupa aplikasi yang **tidak memiliki web interface**
- Menerapkan salah satu teknologi XML-RPC, SOAP atau REST
- Memiliki sifat-sifat service pada umumnya: interoperability, self-describing, discoverable, reusable

Jadi, Apa itu Web Service?

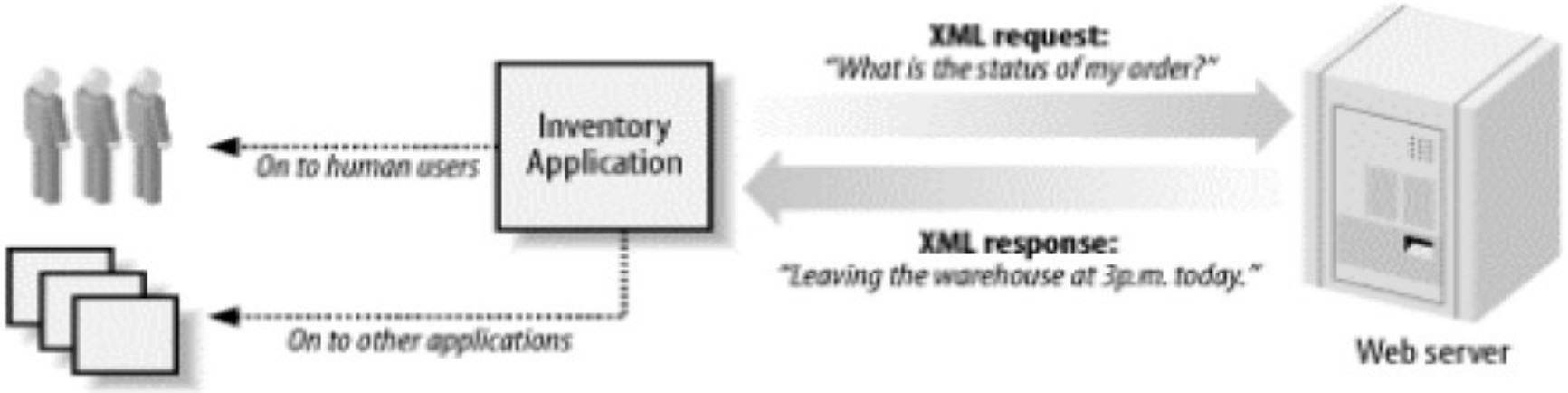




Website dan Web Service

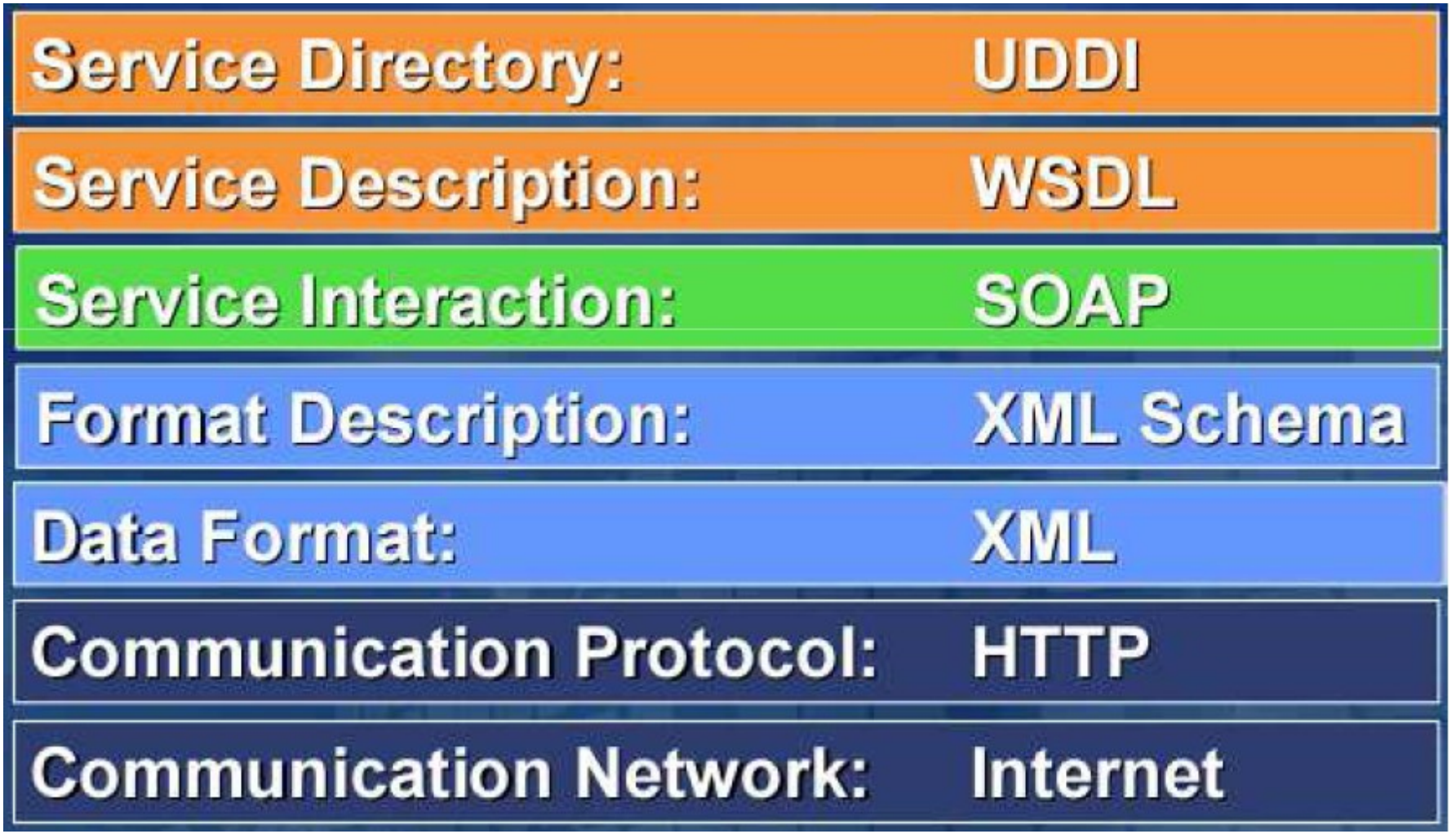


Front-end dan human centric (komunikasi dengan manusia)



Back-end dan application-centric (komunikasi antar aplikasi)

Lapisan Web Service



Lapisan Web Service

Biru tua: Service Transport

- Bertanggung jawab mengirim message antar aplikasi

Biru muda: Format XML message

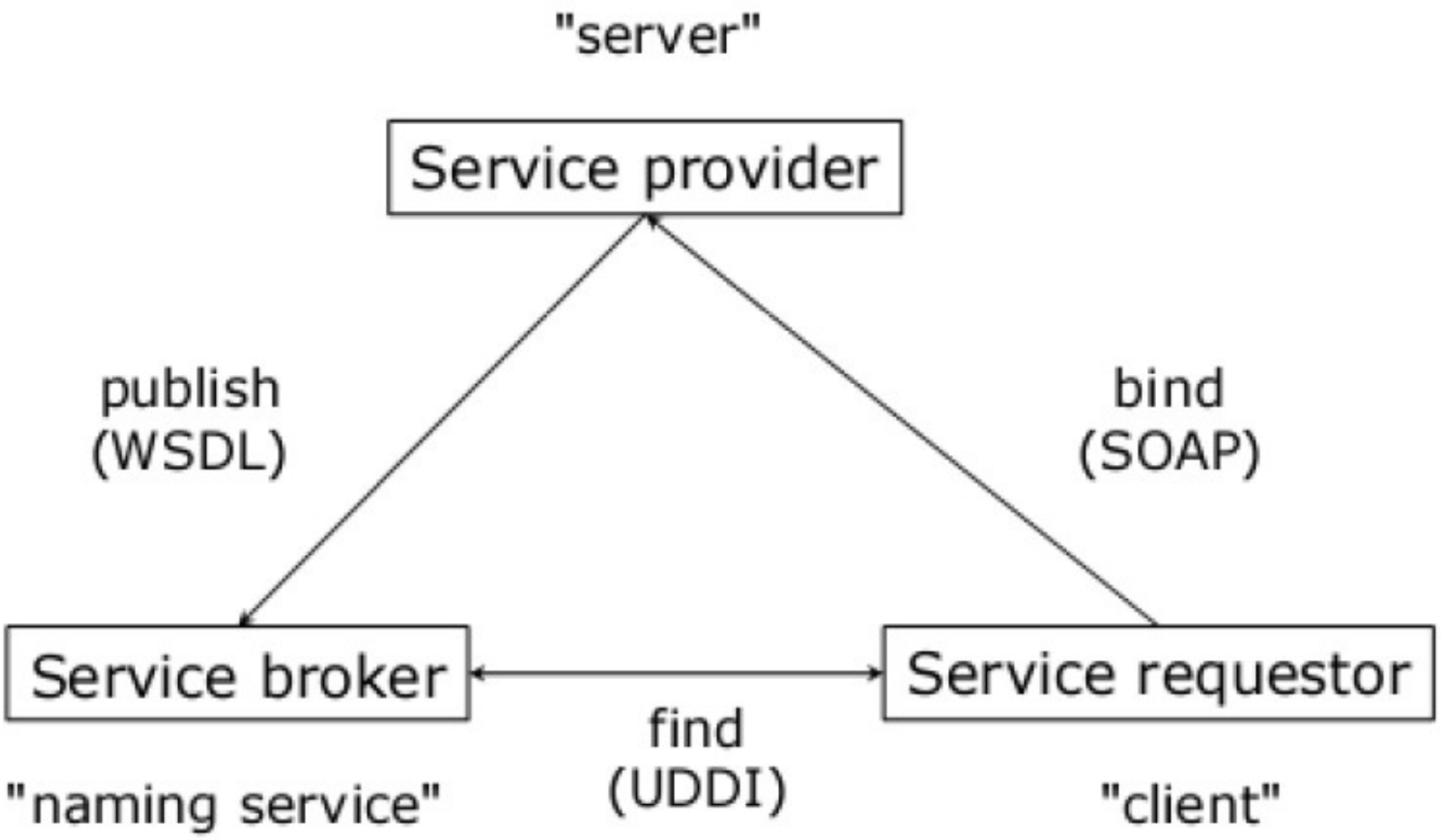
Hijau: Service Interaction

- Bertanggung jawab melakukan *encoding/decoding* message sesuai format XML yang ditetapkan → supaya dapat dimengerti dan dipertukarkan

Oranye:

- Service description: mendeskripsikan web service tersebut dalam bentuk *public interface* menggunakan WSDL (Web Service Description Language) untuk dipublish ke service broker
- Service discovery: mempublikasikan (mendaftarkan, menyimpan dan mengkategorikan) service ke dalam service broker/registry serta menyediakan fasilitas untuk pencarian service dan providernya, ditangani oleh UDDI (Universal Description, Discovery and Integration)

Arsitektur Web Service



Bagaimana Web Service Beroperasi?

Sisi Server:

- Membuat fungsi utama/*core function*
- Membuat *service wrapper* berupa XML-RPC atau SOAP
- Membuat deskripsi service berupa WSDL atau instruksi integrasi XML-RPC (memuat semua method *public,* argumen dan return valuenya); plus dokumentasi yang *human readable*
- *Deploy* (rilis) service
- Daftarkan service tersebut melalui UDDI agar *discoverable*

Sisi Client:

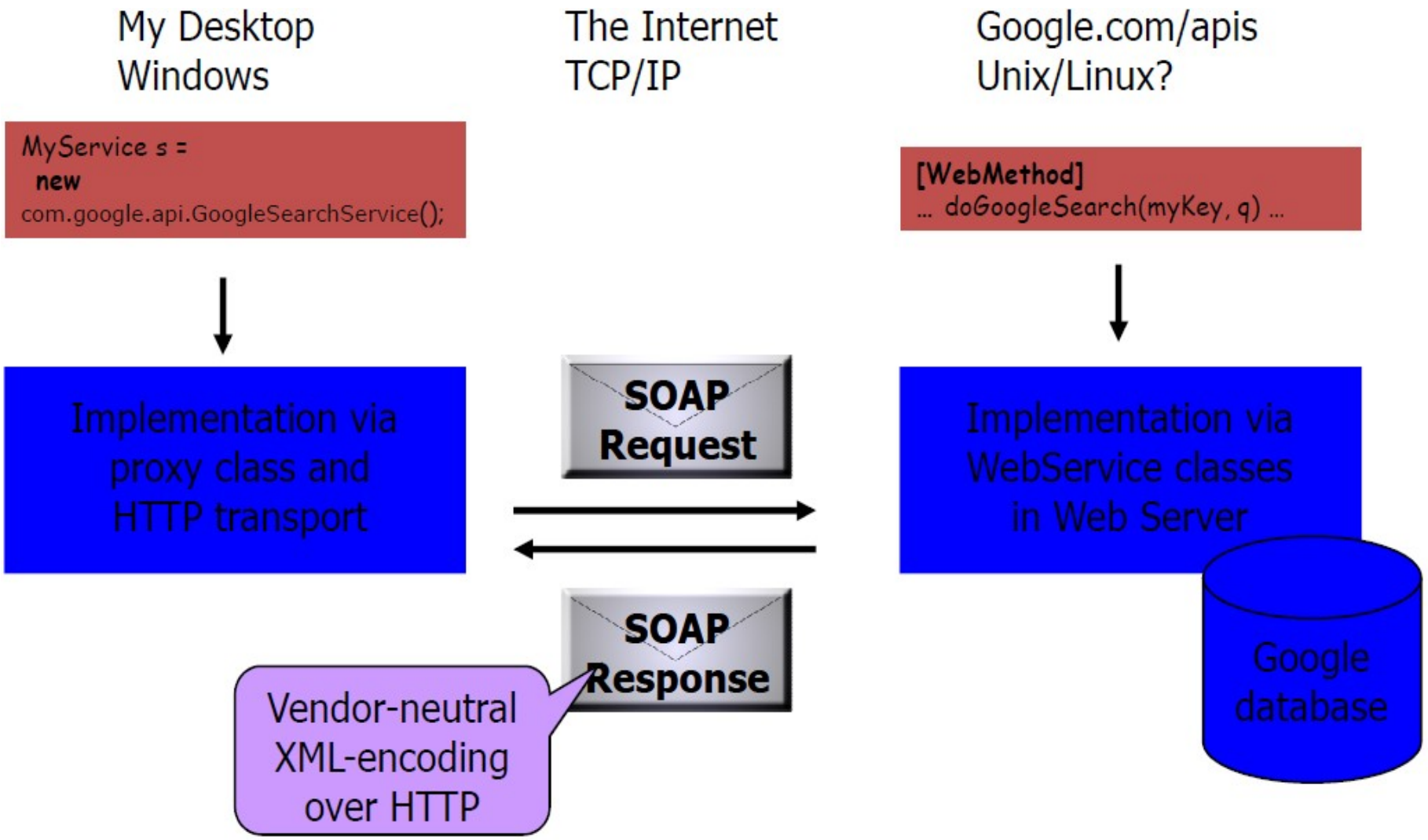
- Mencari service melalui UDDI
- Mengambil *service description file* berupa WSDL atau instruksi XML-RPC
- Membuat klien XML-RPC atau SOAP (dapat berupa fungsi lokal atau pesan XML untuk dikirim → berdasarkan WSDLnya)
- Memanggil *remote service* tersebut

Contoh Implementasi: Klien Google di VB

- Membuat sebuah class *proxy* untuk berkirim message, *instansiasi* (wujudkan), dan *invoke* (panggil)
- Class *proxy* dibuat dari hasil pembacaan file WSDL yang berupa file XML berisi deskripsi service

```
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)
```

Contoh Implementasi: Klien Google di VB



Contoh Deskripsi Web Service

- 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

Yang perlu diperhatikan saat mendesain Web Service

Ingat Tujuannya:

- Menyediakan *universal interoperability*
- Bisa digunakan oleh banyak pihak, dan cepat untuk digunakan oleh aplikasi manapun
- Memungkinkan *dynamic binding* dalam skala Internet
- Mendukung SOA
- Mendukung lingkungan Web dan lainnya

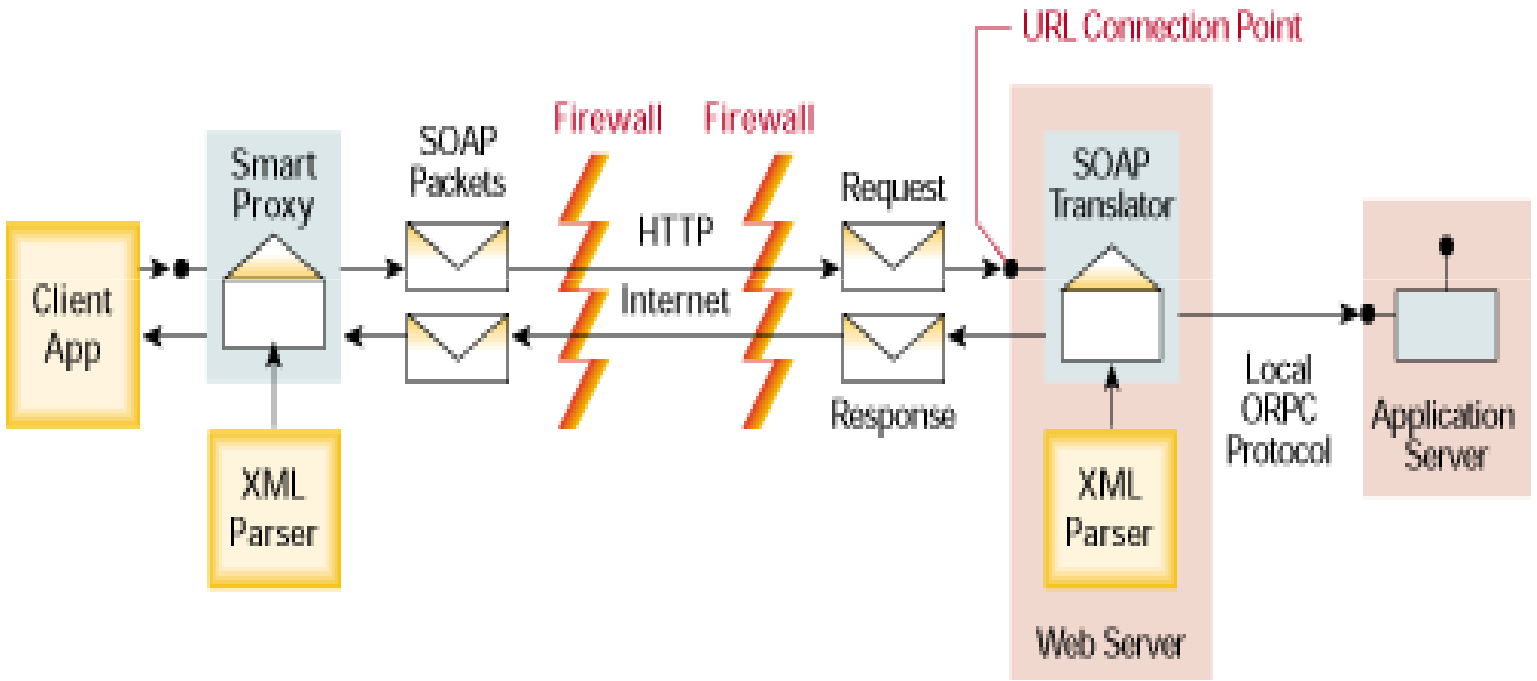
Sehingga, perlu untuk:

- Menaati standar-standar yang ada
- Memanfaatkan infrastruktur seminimal mungkin (menggunakan standar yang paling minimal)
- Integrasi aplikasi yang *low-level*
- Fokus pada *message* dan dokumen, bukan pada API-nya

SOAP (Simple Object Access Protocol)

- Protokol komunikasi berbasis XML yang memungkinkan aplikasi saling bertukar informasi melalui HTTP
- *Platform* dan *language-independent*
- Berupa format untuk mengirimkan message melalui Internet dan merupakan standar W3C
- Membungkus request dan 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

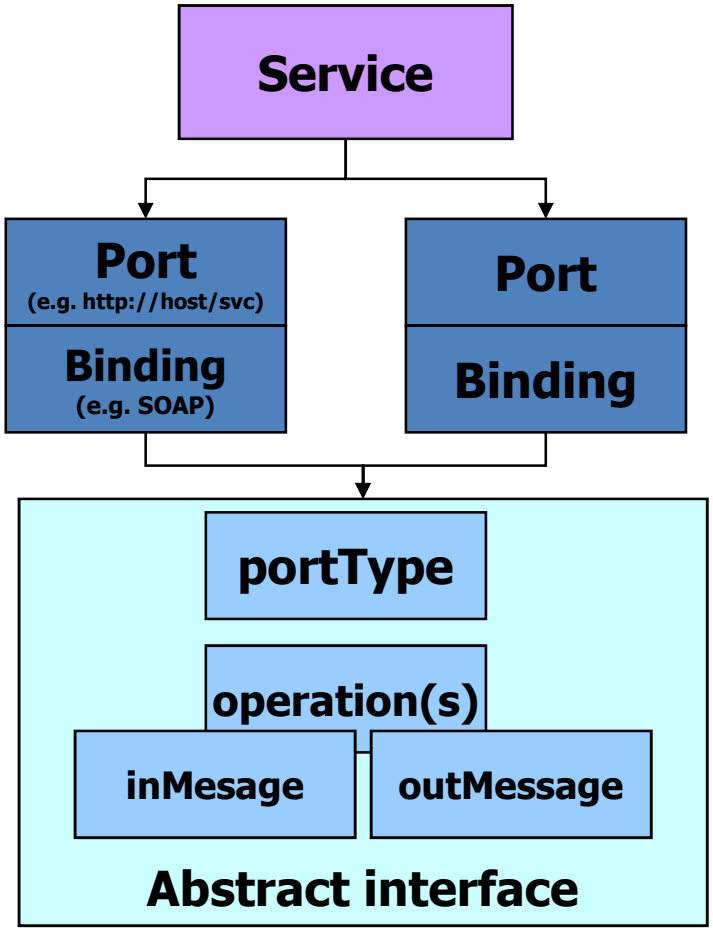
```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:ns1="urn:testns"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"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

Web Services Description Language

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

WSDL Structure

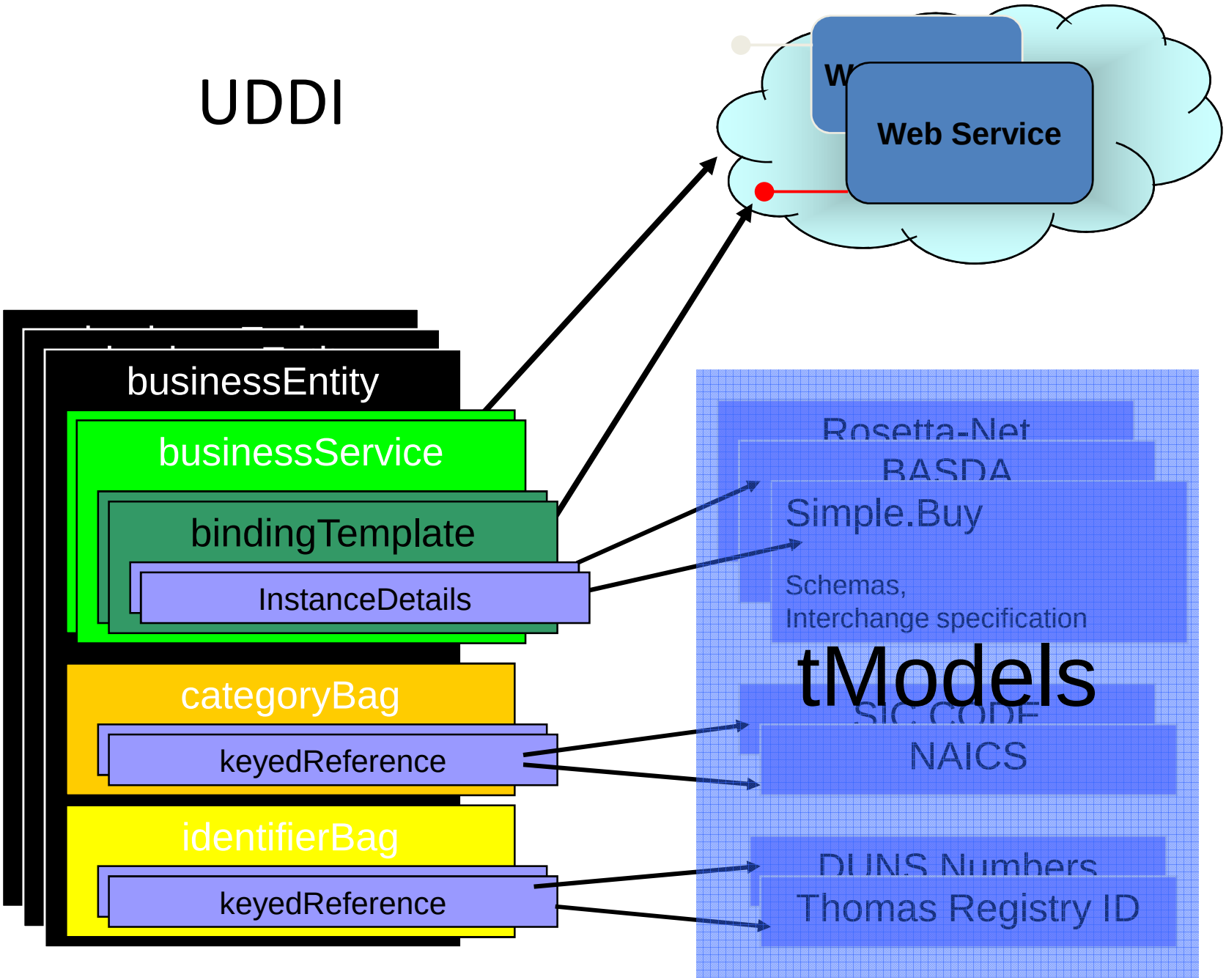
- portType
  - Abstract definition of a service (set of operations)
- Multiple bindings per portType:
  - How to access it
  - SOAP, JMS, direct call
- Ports
  - Where to access it



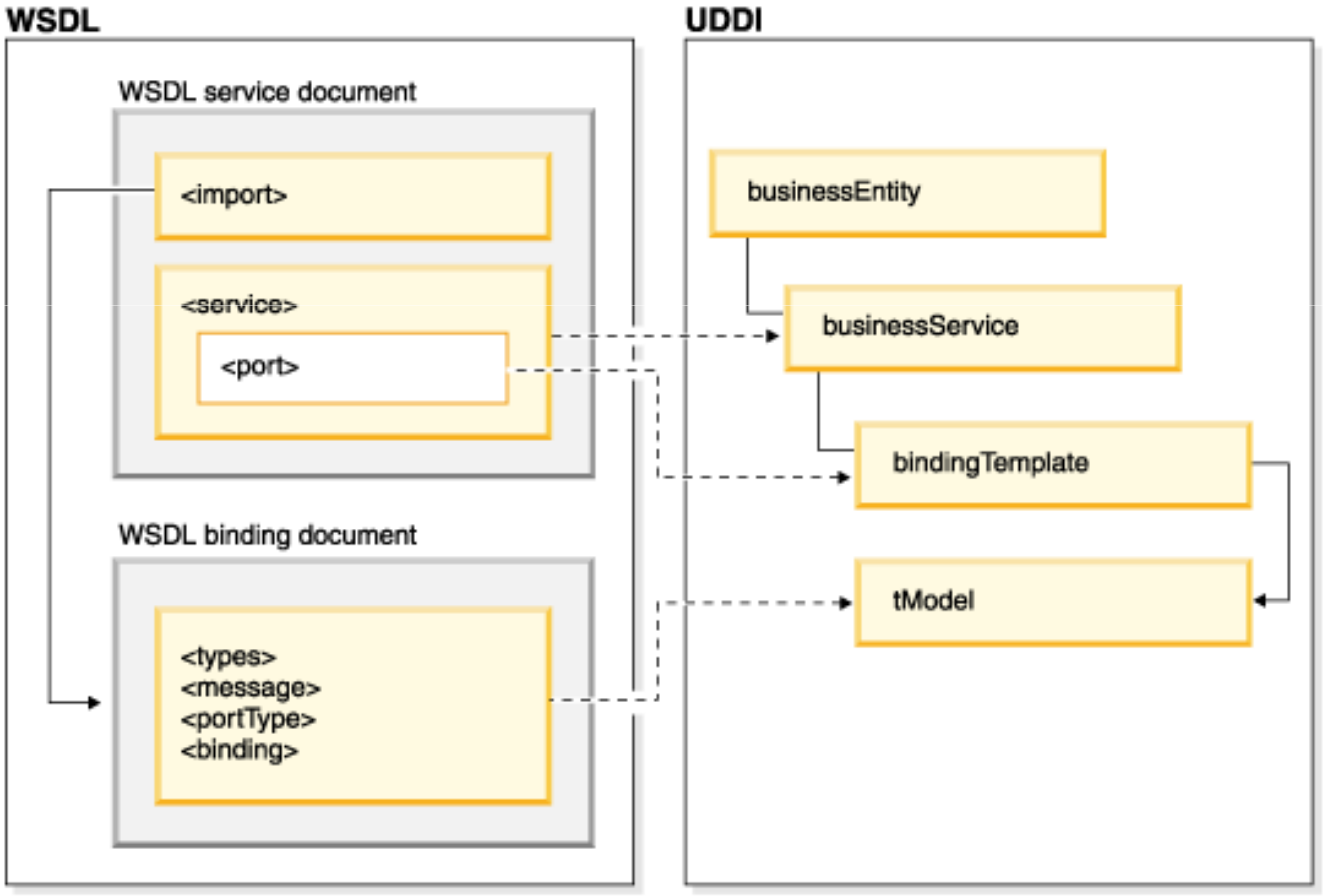
UDDI Overview

- UDDI defines the **operation** of a service **registry**:
  - **Data structures** for registering
    - Businesses
    - Technical specifications: tModel is a keyed reference to a technical specification.
    - Service and service endpoints: referencing the supported tModels
  - SOAP Access **API**
  - **Rules** for the operation of a global registry

UDDI


Web Service
businessEntity
businessService
bindingTemplate
InstanceDetails
categoryBag
keyedReference
identifierBag
keyedReference
Rosetta-Net
RASDA
Simple.Buy
Schemas,
Interchange specification
tModels
NAICS
DUNS Numbers
Thomas Registry ID

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

Next

- Cloud Computing