



Diplomski studij
Informacijska i
komunikacijska tehnologija:
Telekomunikacije i informatika
Računarstvo:
Programsko inženjerstvo i
informacijski sustavi
Računarska znanost

Ak.god. 2009./2010.

Raspodijeljeni sustavi

5.

Web 2.0 i usluge weba

Zaštićeno licencom <http://creativecommons.org/licenses/by-nc-sa/2.5/hr/>





■ slobodno smijete:

- **dijeliti** — umnožavati, distribuirati i javnosti priopćavati djelo
- **remiksirati** — prerađivati djelo

■ pod sljedećim uvjetima:

- **imenovanje**. Morate priznati i označiti autorstvo djela na način kako je specificirao autor ili davatelj licence (ali ne način koji bi sugerirao da Vi ili Vaše korištenje njegova djela imate njegovu izravnu podršku).
- **nekomercijalno**. Ovo djelo ne smijete koristiti u komercijalne svrhe.
- **dijeli pod istim uvjetima**. Ako ovo djelo izmijenite, preoblikujete ili stvarate koristeći ga, prerađu možete distribuirati samo pod licencom koja je ista ili slična ovoj.

U slučaju daljnjeg korištenja ili distribuiranja morate drugima jasno dati do znanja licencne uvjete ovog djela. Najbolji način da to učinite je linkom na ovu internetsku stranicu.

Od svakog od gornjih uvjeta moguće je odstupiti, ako dobijete dopuštenje nositelja autorskog prava. Ništa u ovoj licenci ne narušava ili ograničava autorova moralna prava.

Tekst licencije preuzet je s <http://creativecommons.org/>.

- ◆ Uvod
- ◆ Ideja
- ◆ Usporedba s drugim modelima
- ◆ Vrste usluga weba
- ◆ Jezici i protokoli: SOAP, WSDL, UDDI
- ◆ Usluge weba temeljene na RPC-u
- ◆ Usluge weba temeljene na dokumentima
- ◆ Uslužno orijentirana arhitektura
- ◆ Usluge weba temeljene na stanju resursa

- ◆ pojam nastao u O'Reilly Media (2003)
- ◆ ideja: Internet kao platforma poput operacijskog sustava
- ◆ potiče inovacije i sastavljanje funkcionalnih dijelova iz neovisnih izvora
- ◆ korisnici postaju središte
 - interaktivnost, sudjelovanje, objavljivanje
- ◆ višemedijski sadržaji (glazba, video, lokacije, slike, karte, ...)

◆ RIA (*Rich Internet Applications*)

- Flash
- Ajax (*Asynchronous JavaScript and XML*)
 - JavaScript - programski okviri: JQuery, Yahoo UI Library, ExtJS, Dojo, ...
 - HTML
 - CSS (Cascading Style Sheets)
 - XML
- Eclipse Platform
- GWT (*Google Web Toolkit*)

◆ SOA (*Service Oriented Architecture*)

- Feeds (*RSS - Really Simple Syndication, Atom*)
- usluge weba (SOAP, WSDL, UDDI)
- kombiniranje sadržaja - *mashup* (podaci, ljudi)

- ◆ označavanje sadržaja (*tagging*) - svatko može označiti stranicu
- ◆ wiki uređivanje - svi mogu mijenjati stranice
- ◆ *blogovi* (*web log*) - dnevници na webu
- ◆ *podcast* - intervjui, predavanja i drugi sadržaji
- ◆ dijeljenje video sadržaja
- ◆ komentiranje svega
- ◆ nove aplikacije (npr. *poke*)

- ◆ Google
 - AdWords - diskretno oglašavanje u tražilici
 - AdSense - Googlovi oglasi na vašim stranicama
 - Google Maps (Earth) - karte - [Discovery Channel: Planet Earth](#)
 - Google Docs - kolaborativna izrada dokumenata
 - [YouTube](#) - dijeljenje video sadržaja
- ◆ Wikipedia
- ◆ razni blogovi
- ◆ [Digg](#) - dodavanje članaka i dijeljenje s drugima
- ◆ [Flickr](#) - dijeljenje slika
- ◆ Technorati - tražilica blogova
- ◆ [Facebook](#) i MySpace
- ◆ [del.isio.us](#) - označavanje članaka
- ◆ Skype, ICQ, ...
- ◆ [Netvibes](#) - agregator stranica (RSS)

- ◆ Katalozi:
 - www.hr, www.zagreb.hr, www.varazdin-online.com, ...
- ◆ Vijesti
 - Index.hr, www.net.hr, www.javno.hr, ...
- ◆ Novine
 - Jutarnji.hr, [večernji.hr](http://vecernji.hr), ...
- ◆ Video vijesti
 - dnevnik.hr, www.hrt.hr, www.rtl.hr, ...
- ◆ Blogovi
 - www.blog.hr, www.blogger.hr, www.mojblog.hr, ...
- ◆ Specijalizirane stranice:
 - www.poslovni.hr
 - www.bug.hr, www.pcchip.hr, www.vidi.hr
 - ...

Web 1.0

- ♦ čitanje informacija
- ♦ klijent-poslužitelj
- ♦ HTML
- ♦ portali
- ♦ parsiranje informacija
- ♦ posjedovanje
- ♦ modemska veza i HW
- ♦ direktoriji
- ♦ tipična tvrtka: Netscape

Web 2.0

- ♦ pisanje informacija
- ♦ P2P (osobe i aplikacije)
- ♦ XML
- ♦ usluge (*services*)
- ♦ API sučelja i RSS
- ♦ dijeljenje
- ♦ širokopojasna veza i SW
- ♦ oznake
- ♦ tipična tvrtka: Google

- ◆ 10. najpopularnija aplikacija na Facebooku
- ◆ oko 600.000 aktivnih korisnika
- ◆ 0,5 mil. jedinstvenih korisnika dnevno (i raste)
- ◆ 10 mil. otvaranja stranica na dan (oko 400.000/sat)
- ◆ 300% mjesečni rast
- ◆ 200 zahtjeva/sec
- ◆ 5TB mjesečnog prometa

- ◆ 4 DB poslužitelja, 6 apl. pos., 1 "front end" poslužitelj
 - 6, 4 jezgri 8 GB na aplikasijskom poslužitelju
 - 2 32GB 4 jezgrena poslužiteljs s 4x 15K SCSI RAID 10 diskova u modu "master-slave"

podaci preuzeti s: highscalability.com (2008.)



1. HTTP upit

2. HTTP odgovor
HTML

poslužitelj

poslužitelj
web-
aplikacija

SQL
upit

podaci

baza
podataka



1. HTTP
→
6. HTTP
HTML

Facebook
poslužitelj

2. HTTP
REST
→

poslužitelj

poslužitelj
web-
aplikacija

SQL
upit

podaci

baza
podataka

3. API/FQL
←

4. API odg.
→

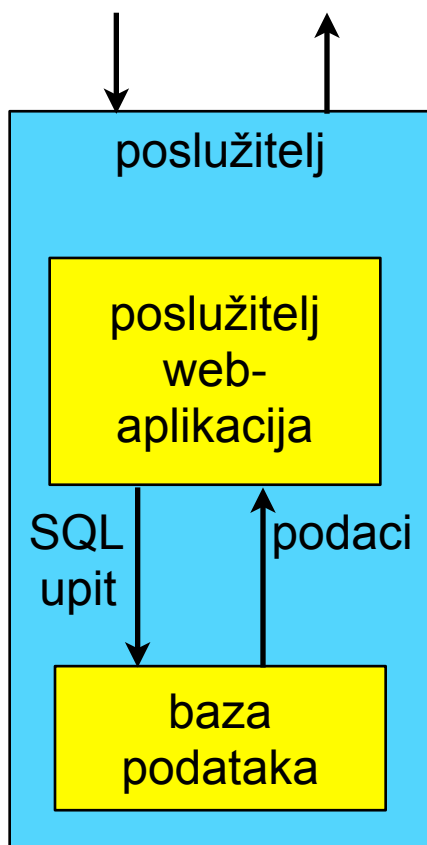
5. FBML
←

Skalabilnost (1)

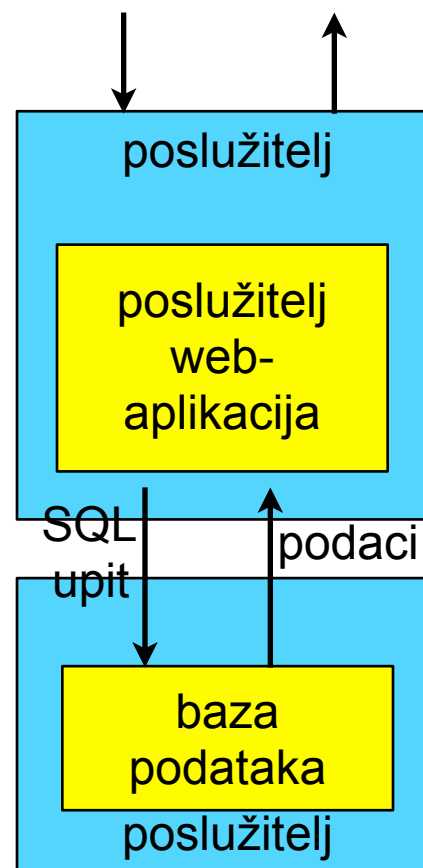


Zavod za komunikacije

jedno računalo



dva računala



- ◆ repliciranje baze (*master/slave*)
- ◆ uranoteženje opterećenja (*load balancing*)
- ◆ particioniranje sadržaja
 - jedna informacija na jednom poslužitelju
- ◆ privremene kopije (*cache*)

- ◆ “obični” Web tipično koriste ljudi za pribavljanje informacija
- ◆ tehnologija Web Services (WS) omogućuje programima lakše korištenje Weba
 - umjesto RPC-a/Java RMI-a, CORBA-e, DCOM-a, itd.
- ◆ svojstva:
 - komunikacija se temelji na XML-u
 - koristi već postojeću internetsku infrastrukturu i protokole
 - usluge dostupne putem Interneta
 - omogućuje integraciju između različitih aplikacija
 - ne ovisi o programskom jeziku ili zatvorenoj tehnologiji jedne tvrtke
 - omogućuje otkrivanje usluga koje nude te aplikacije
 - usluge su slabo povezane
 - temelji se na industrijskim standardima

«Usluga weba je aplikacija identificirana URI-jem, čija se sučelja i veze mogu definirati, opisati i otkriti XML-om i koja podržava direktnu interakciju s drugim aplikacijama putem internetskih protokola koristeći poruke temeljene na XML-u»

(www.w3.org)

«Tehnologija usluga weba predstavlja novu vrstu aplikacija weba. To su samostalne, samoopisujuće aplikacije građene od modula, a koje se mogu objaviti, otkriti i pobuditi putem Weba. Usluge weba obavljaju funkcije koje mogu biti bilo što, od jednostavnih zahtijeva do kompliciranih poslovnih transakcija...»

(IBM's tutorial, www.xml.com)

◆ Usluga weba je program koji:

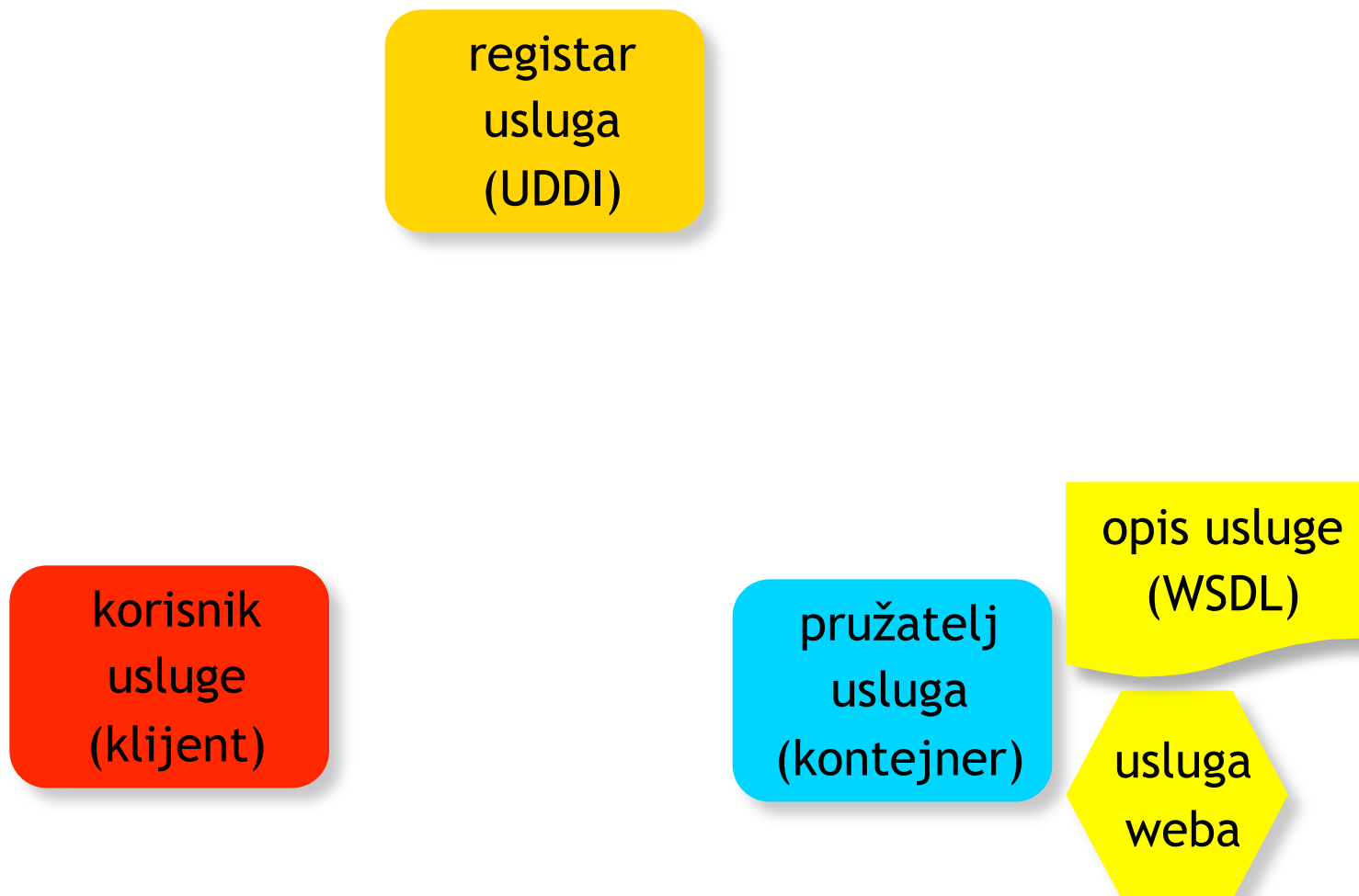
- je identificiran URI-jem
- komunicira s klijentskim programima putem Weba
- ima sučelje (API) opisano standardima usluge weba
- omogućuje korištenje neovisno o platformama i programskim jezicima

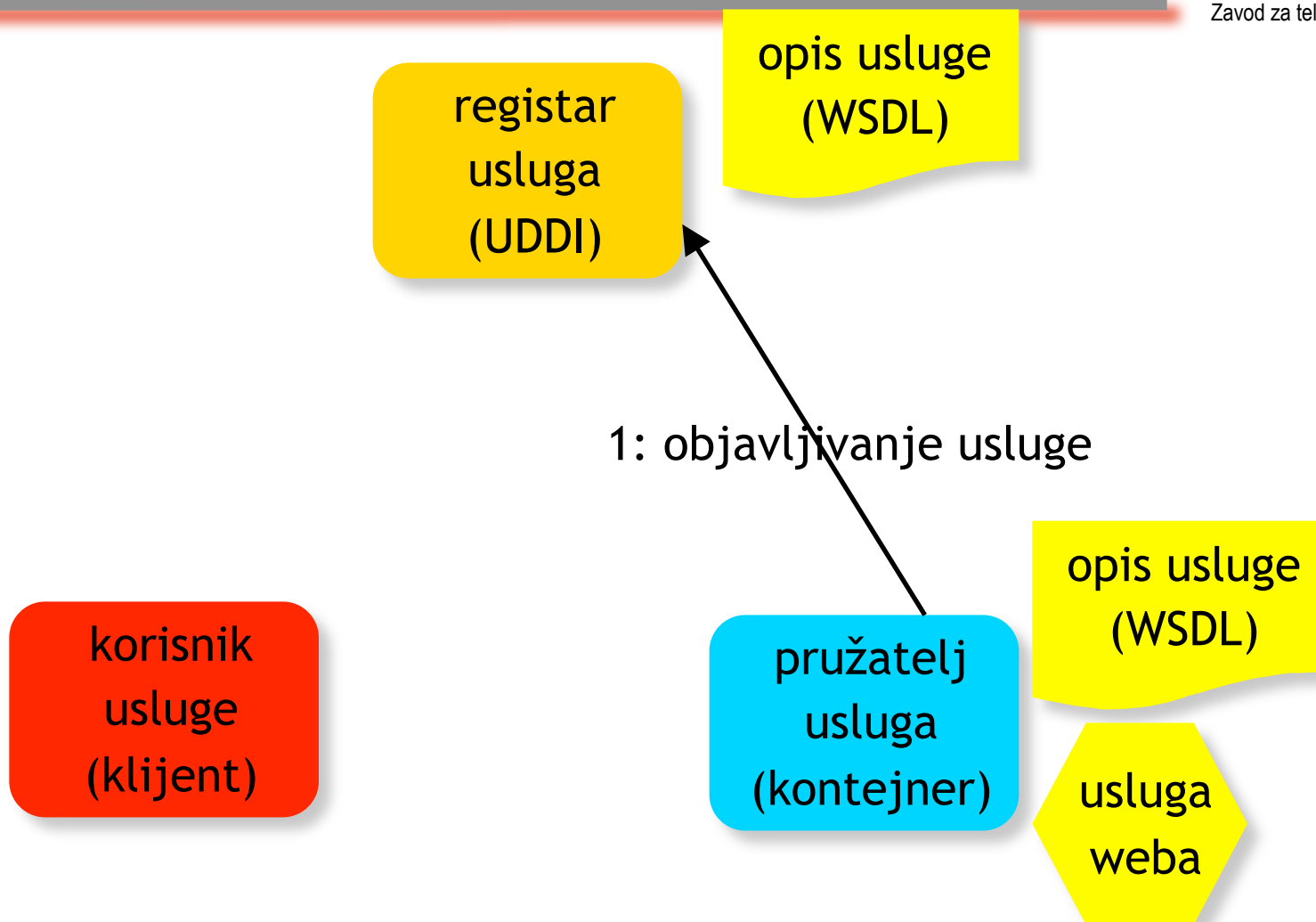
Tehnologije usluga weba

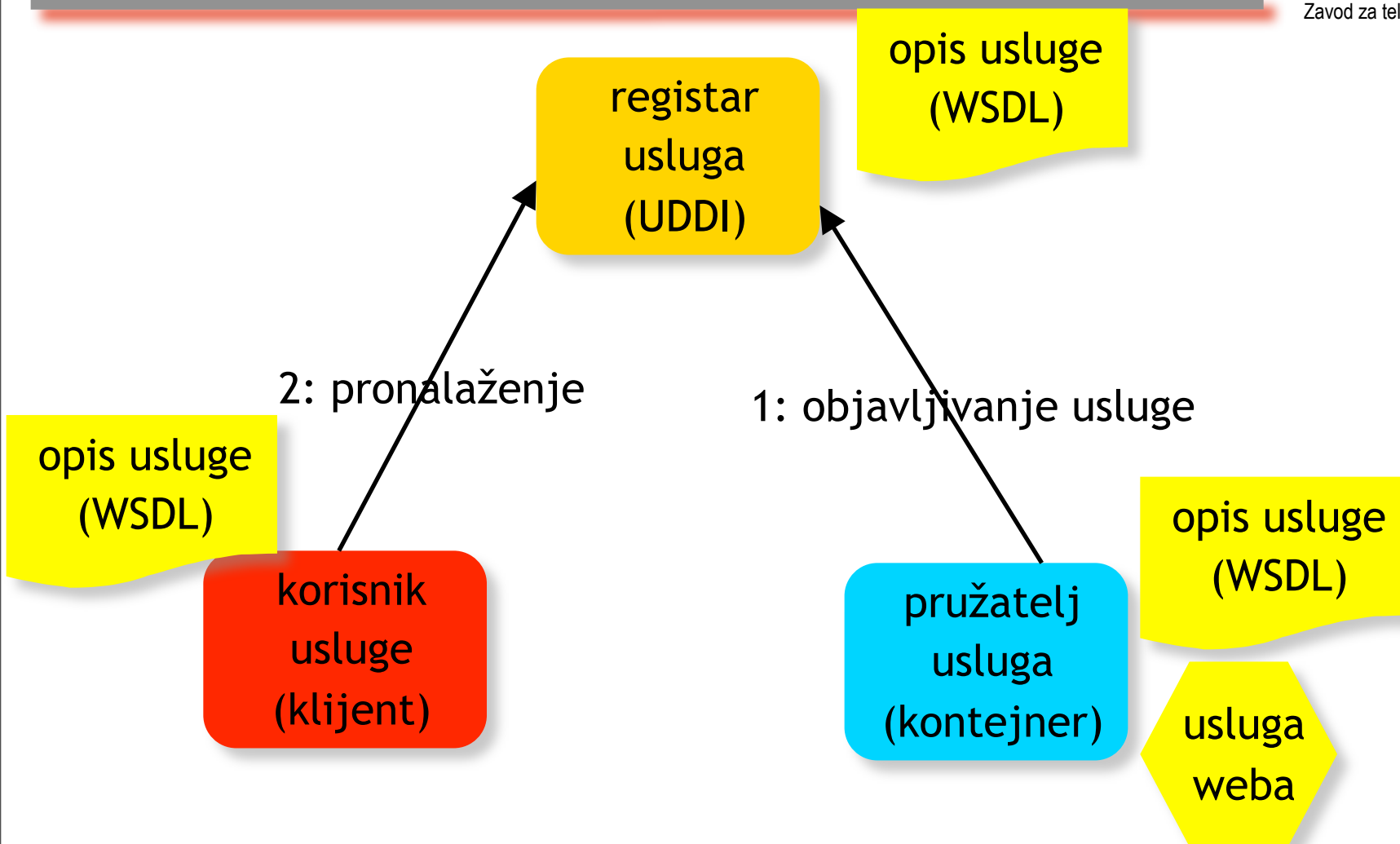
- ◆ WSDL (*Web Services Definition Language*)
 - opisuje uslugu
- ◆ SOAP (*Simple Object Access Protocol*)
 - format poruke
- ◆ UDDI (*Universal Description, Discovery and Integration*)
 - za otkrivanje usluga

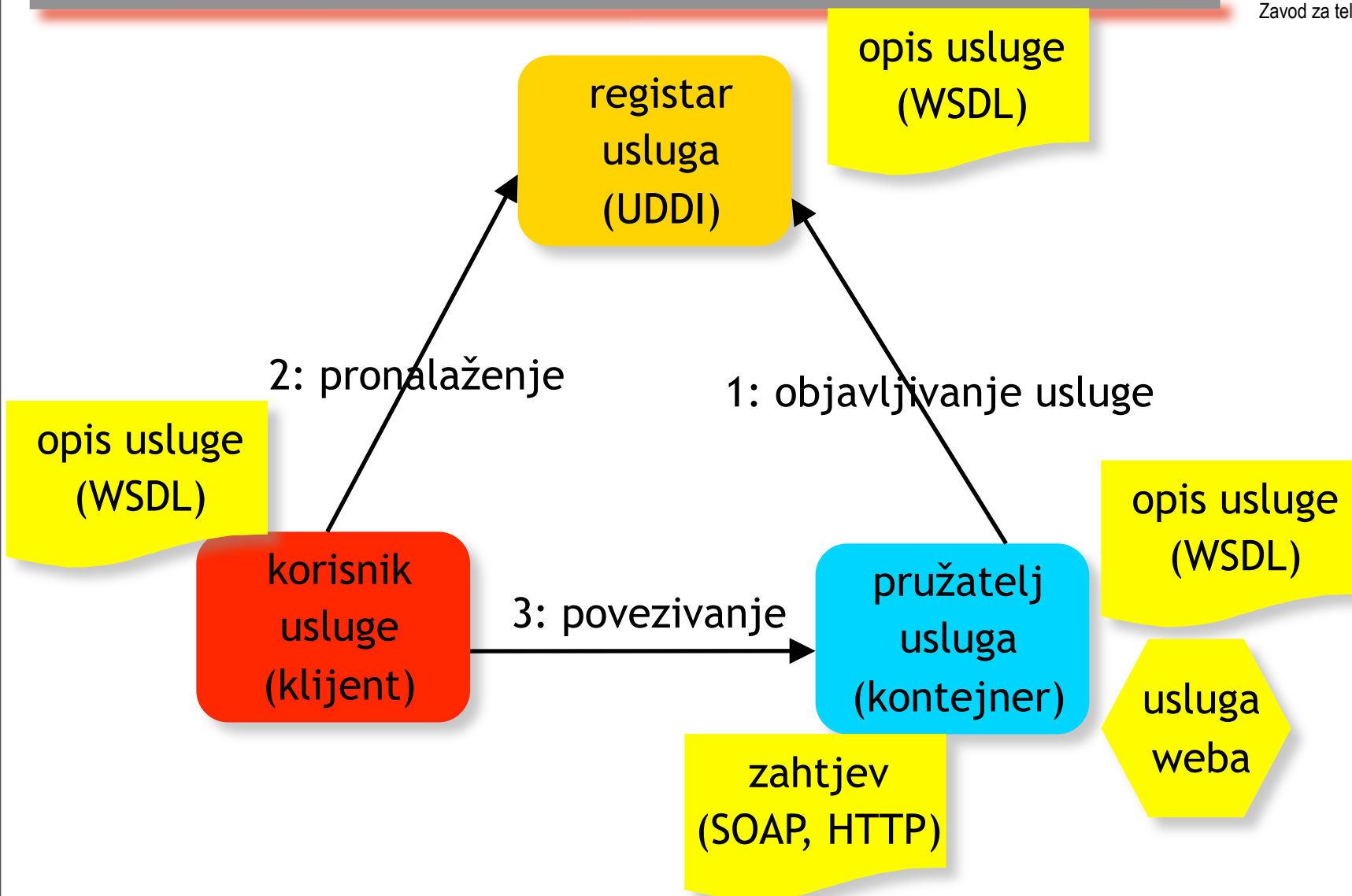
Druga generacija usluga weba (WS-*)

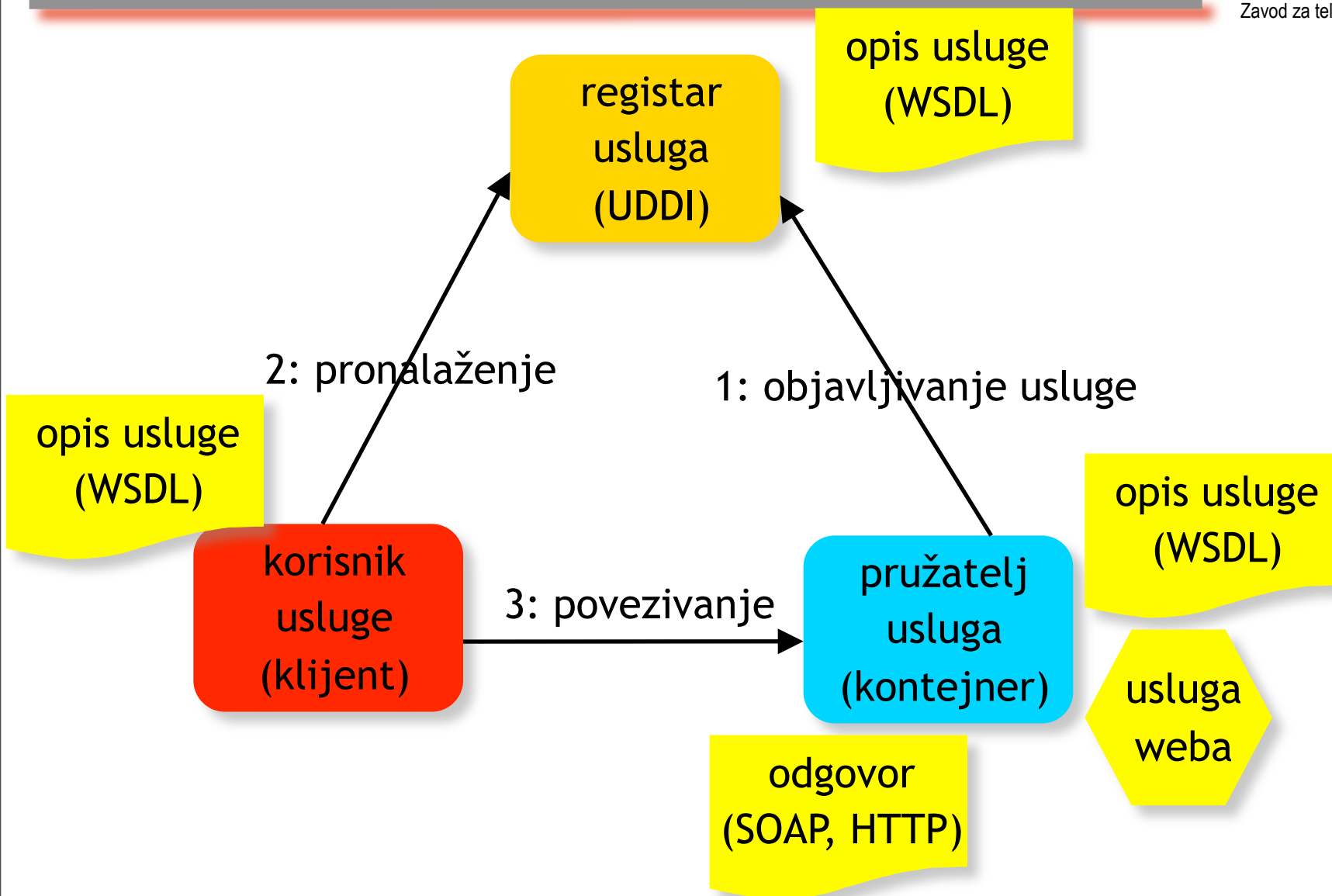
- ◆ WS-Coordination
 - protokol za koordinaciju distribuiranih aplikacija
- ◆ WS-Transaction
- ◆ BPEL4WS (*Business Process Execution Language*)
 - jezik za formalnu specifikaciju poslovnih procesa i interakcijskih protokola
- ◆ WS-Security
 - sigurnosni protokol (TLS, integritet, privatnost, ...)
- ◆ WS-ReliableMessaging
- ◆ WS-Policy
- ◆ WS-Attachments
- ◆ WS-Addressing
 - adresiranje usluga i poruka

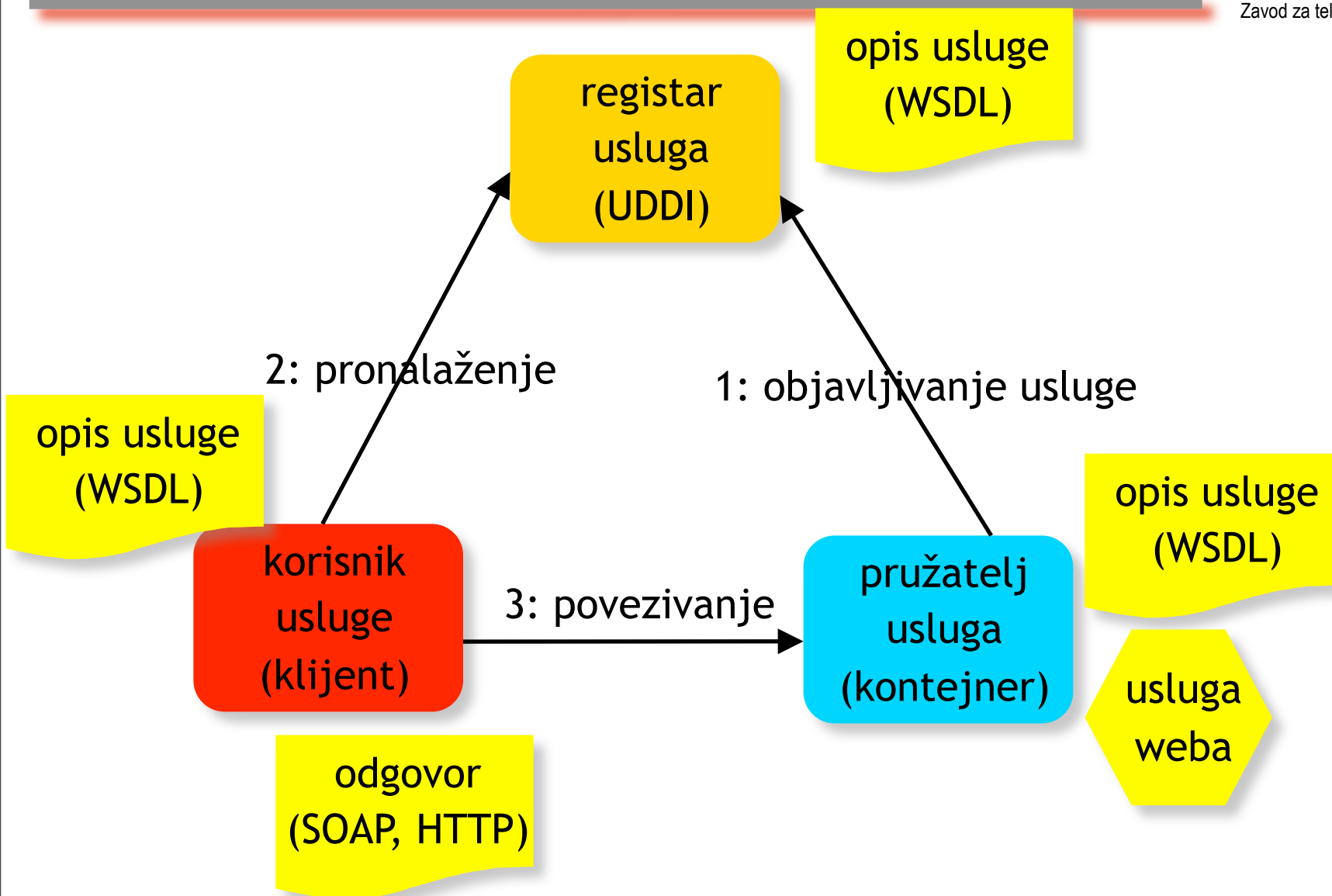








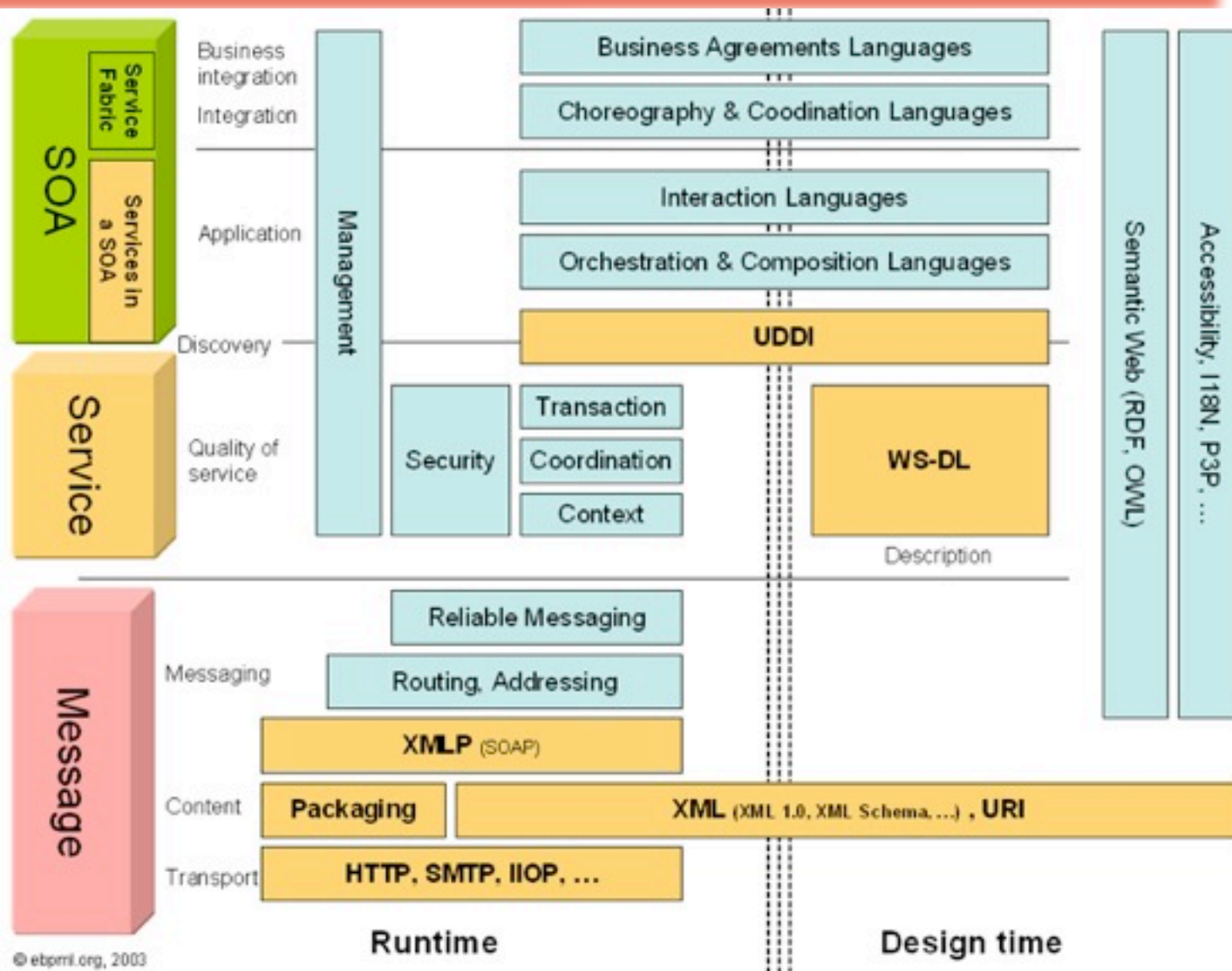




Složaj usluga weba



Zavod za telekomunikacije



<http://www.ebpmi.org/webservices.htm>

klijent – poslužitelj

- ◆ unutar tvrtke
- ◆ ograničeno na skup programskih jezika
- ◆ proceduralno
- ◆ obično ograničeno na određeni transportni sloj
- ◆ jako povezani dijelovi
- ◆ učinkovito procesiranje (prostor/vrijeme)

usluge weba

- ◆ između tvrtki
- ◆ neovisne o programskom jeziku
- ◆ temelji se na porukama
- ◆ jednostavno se koristi različitim transportnim mehanizmima
- ◆ slabo povezane usluge
- ◆ relativno neefikasno procesiranje

aplikacije weba

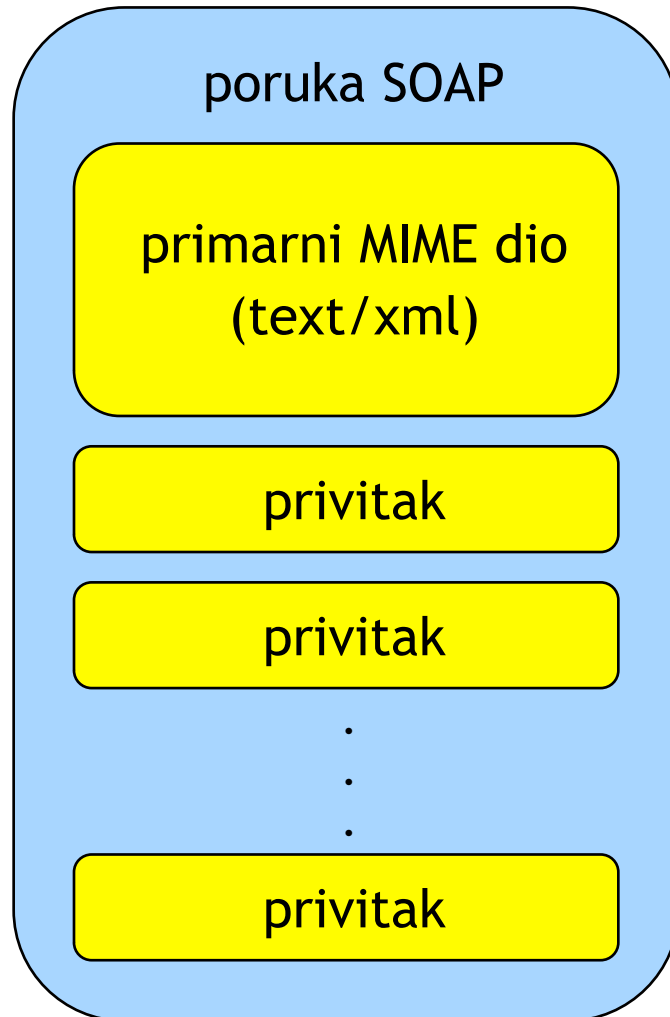
- ◆ interakcija korisnik-program
- ◆ statička integracija komponenti
- ◆ monolitna usluga

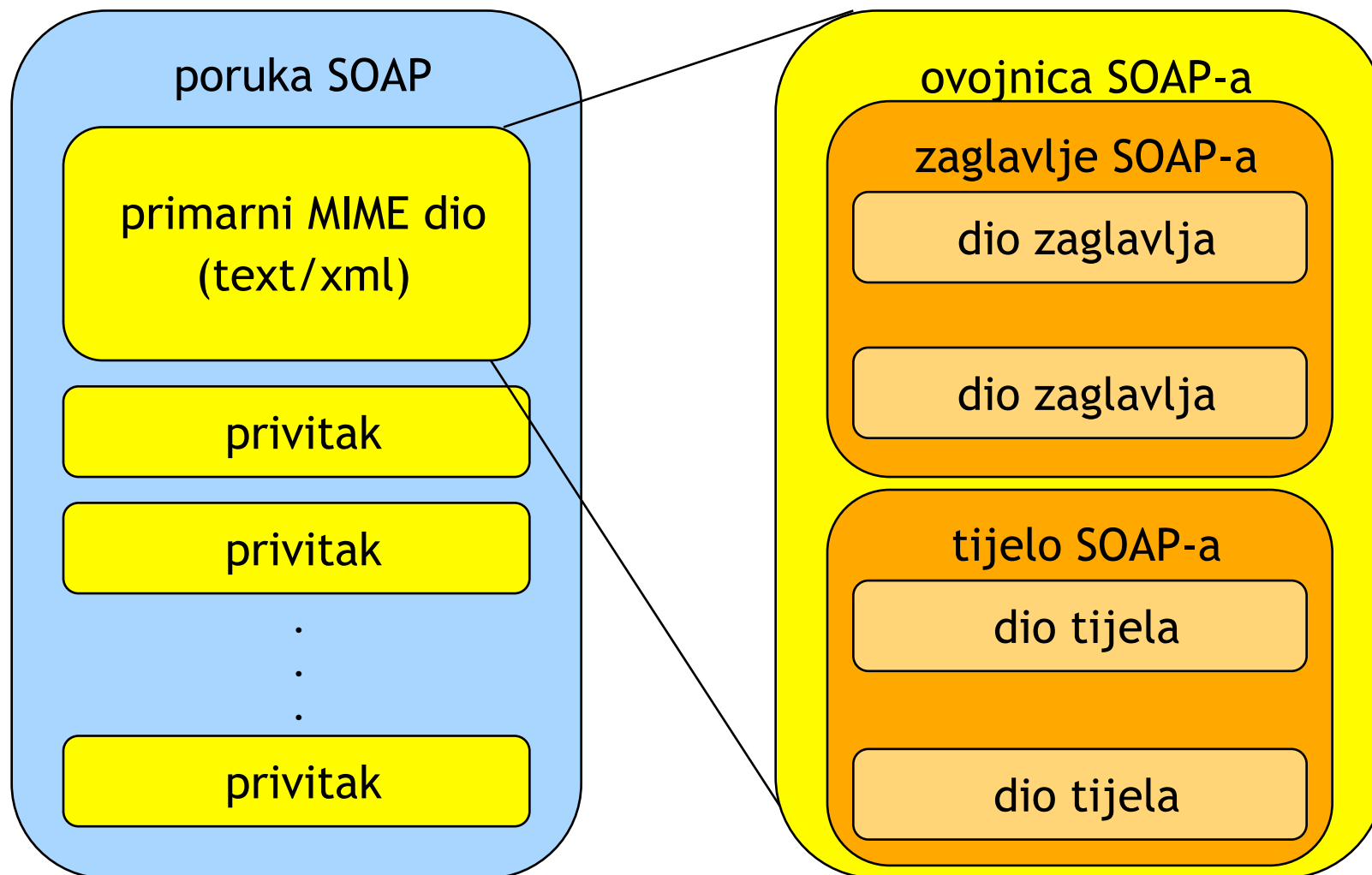
usluge weba

- ◆ interakcija program-program
- ◆ moguća dinamička integracija komponenti
- ◆ moguće je sastavljanje usluga od jednostavnijih

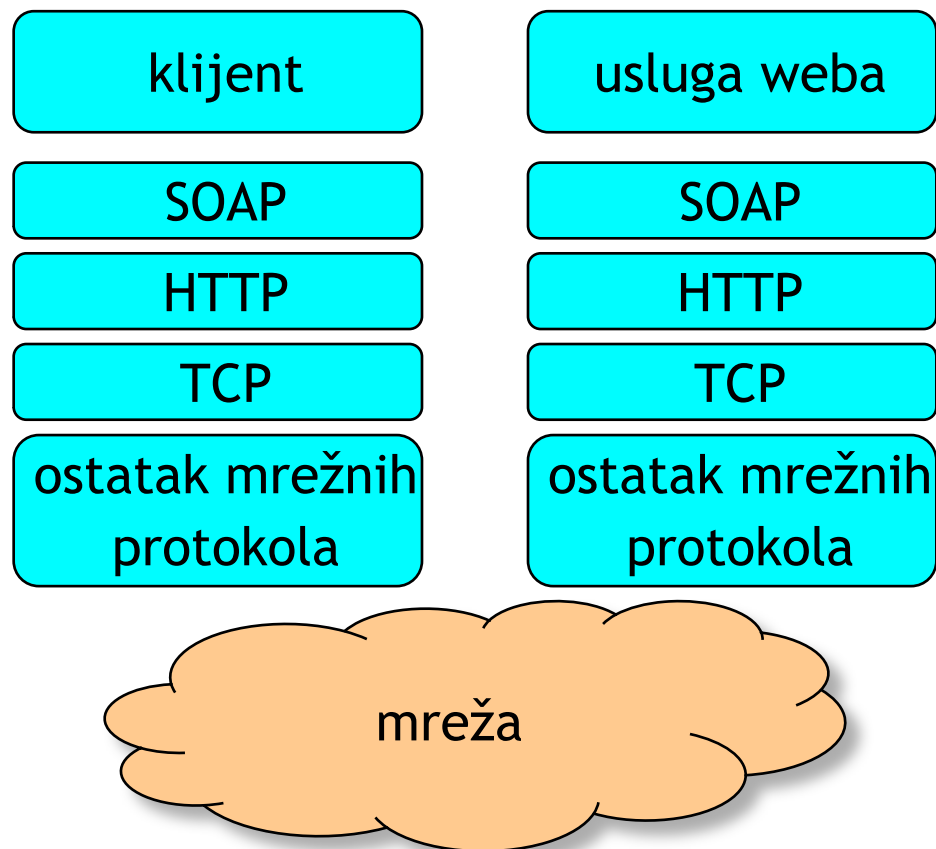
- Usluge temeljene na udaljenim procedurama (RPC)
 - rade kao poziv udaljenih procedura
 - koriste protokol SOAP i specifikaciju WSDL
 - na poslužitelju se poziva metoda u objektu
 - podaci su povezani s metodama koje se pozivaju
- Usluge temeljene na dokumentima/porukama
 - definiraju se poruke koje se razmjenjuju (XML Schema, WSDL)
 - nisu jako povezane
 - nije bitna implementacija, već samo podaci
- Usluge temeljene na stanju resursa
 - RESTful (*Representational state transfer*)
 - temelje se na protokolu HTTP
 - koriste metode protokola: GET, PUT, DELETE

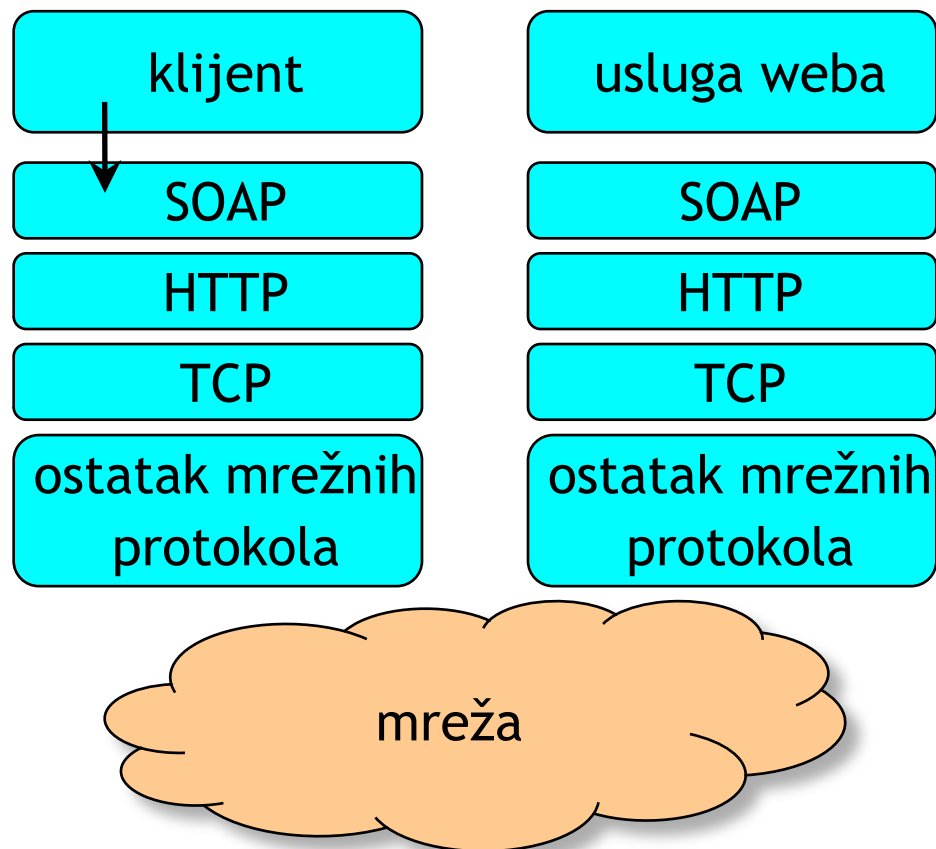
- ◆ omogućuje komunikaciju s uslugom weba
- ◆ dva osnovna načina rada:
 - poziv udaljenih procedura (RPC)
 - slično kao Corba, DCOM, Java RMI
 - služi za prijenos serijaliziranih parametara i rezultata
 - posljedica:
 - ▶ dobro definirana sučelja i tipovi podataka
 - ▶ prilagodni kod može biti generiran automatski
 - razmjena dokumenata/poruka
 - sadrži XML dokument
 - fleksibilnije u odnosu na RPC
 - XSLT i XQuery se koristi za prilagodbu dokumenata
 - lakše se koriste uzorci razmjene poruka
 - polako se uključuje semantički web (ontologije, procesiranje i sl.)
- ◆ specifikacija: <http://www.w3.org/TR/soap/>

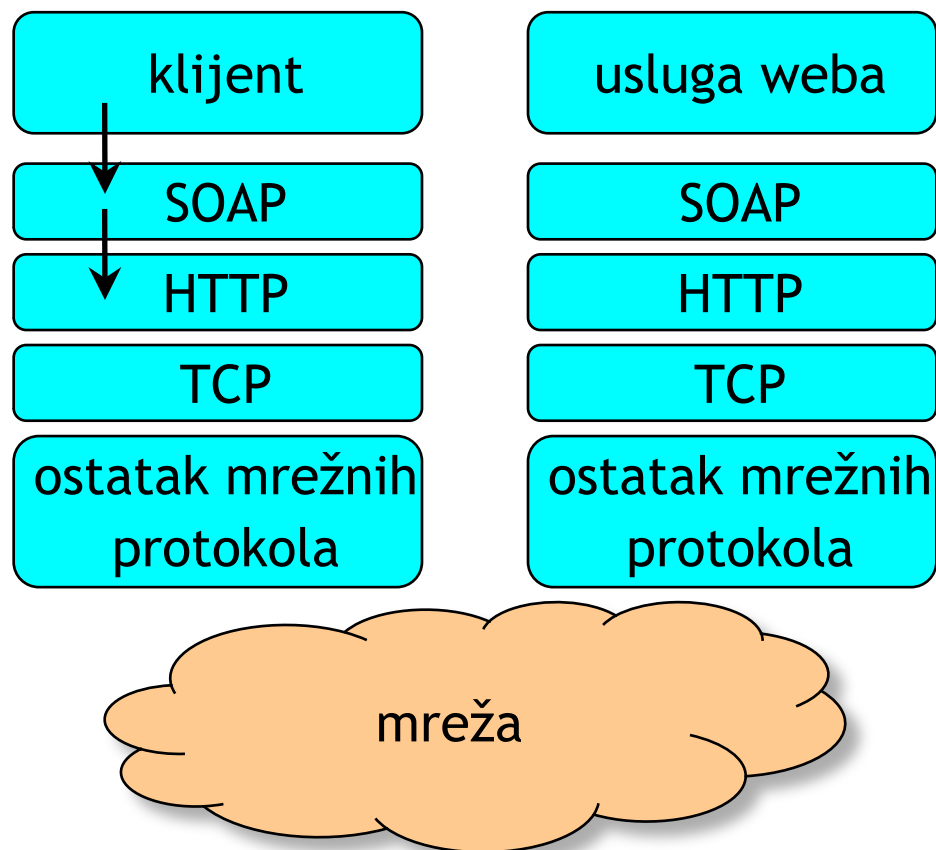




```
<?xml version = "1.0" encoding = "UTF-8"?>
<Envelope xmlns="http://schemas.xmlsoap.org/soap/
  envelope">
  <Header>
    <!-- sadržaj zaglavlja poruke SOAP -->
  </Header>
  <Body>
    <!-- sadržaj poruke SOAP -->
  </Body>
</Envelope>
```



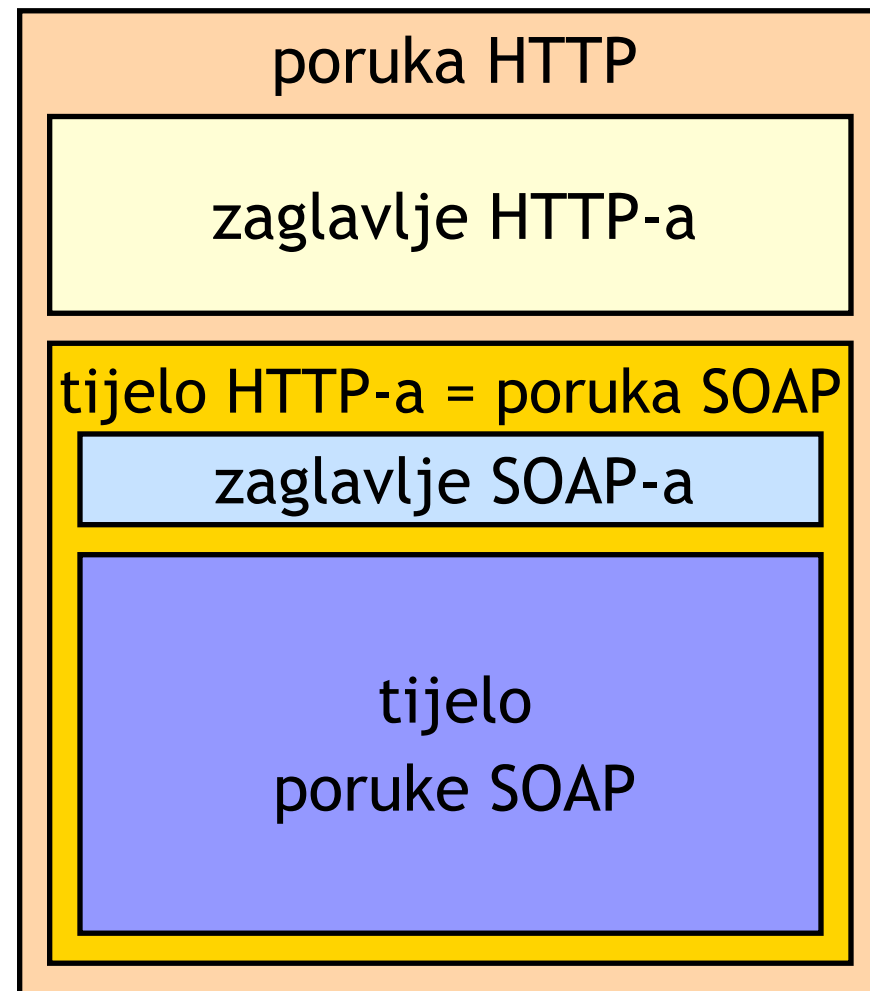
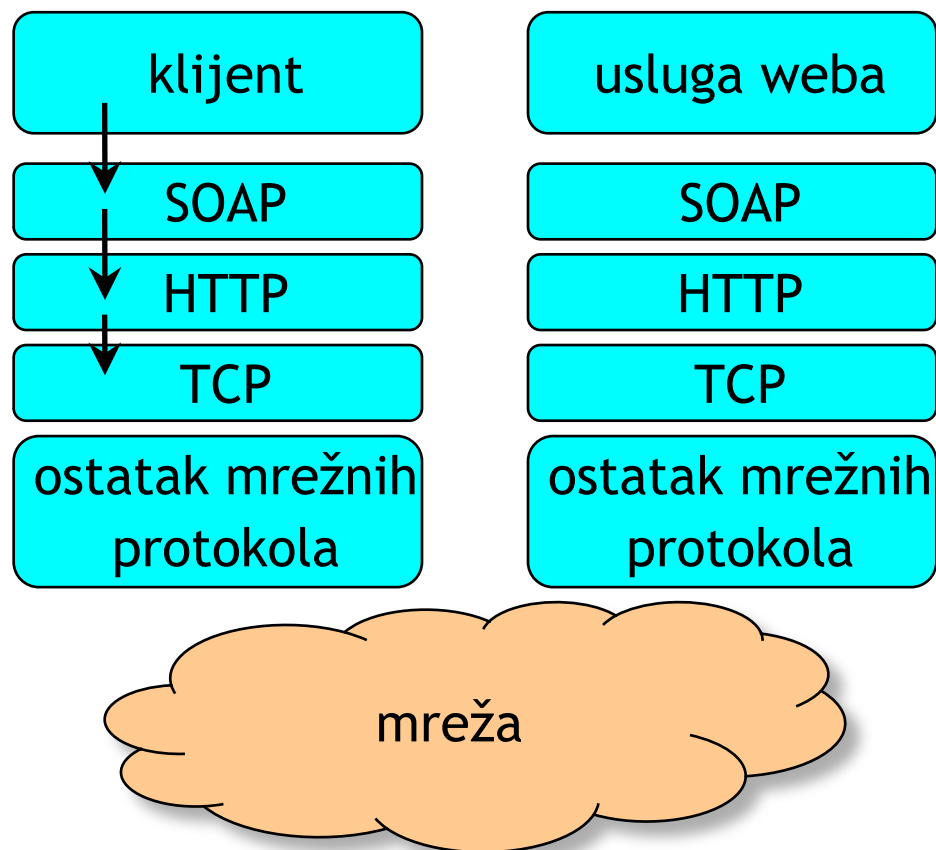


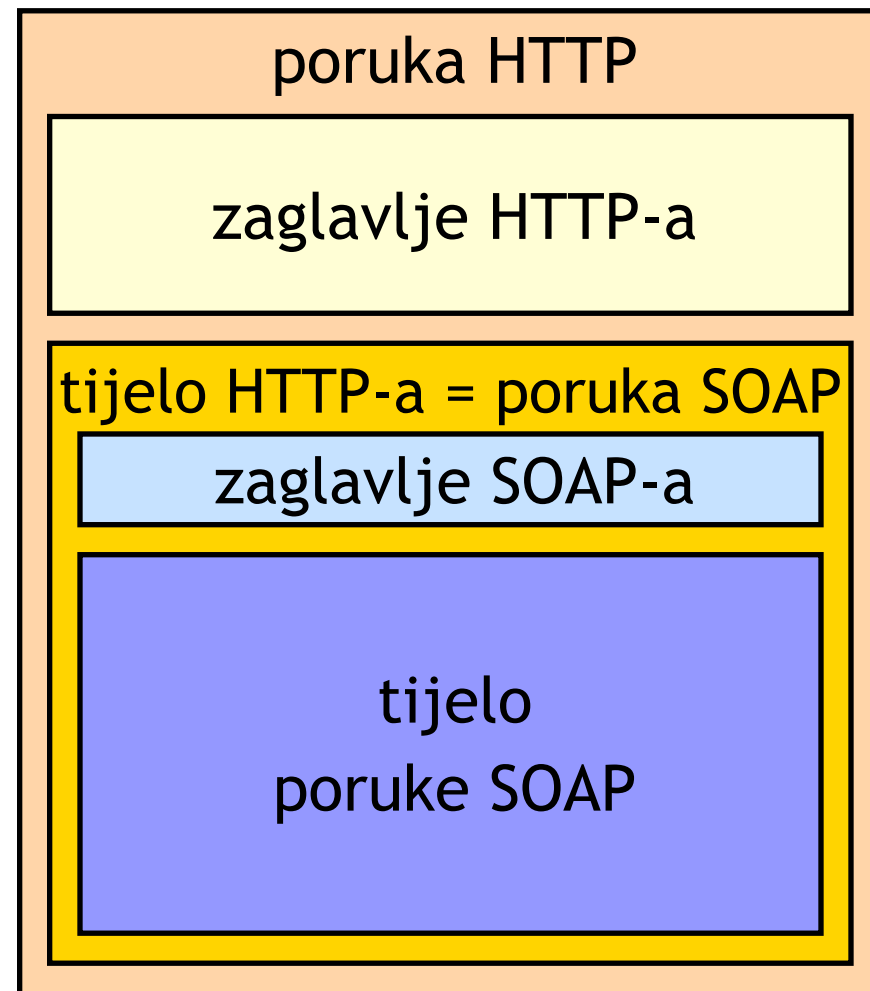
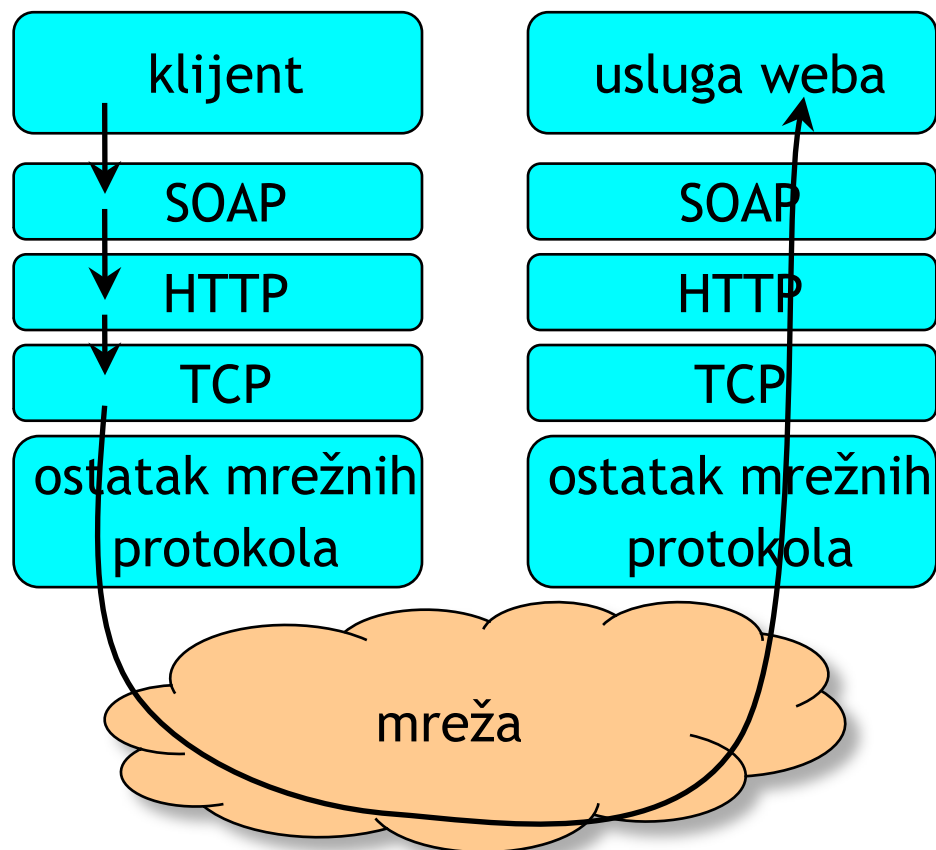


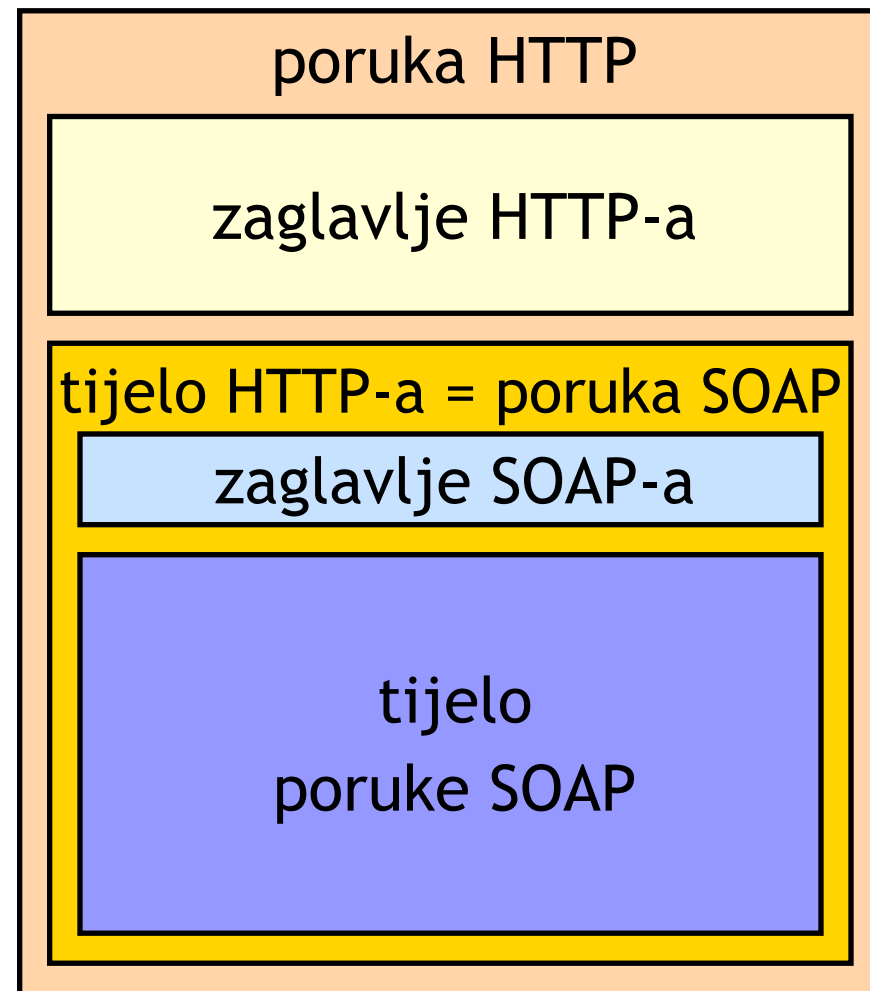
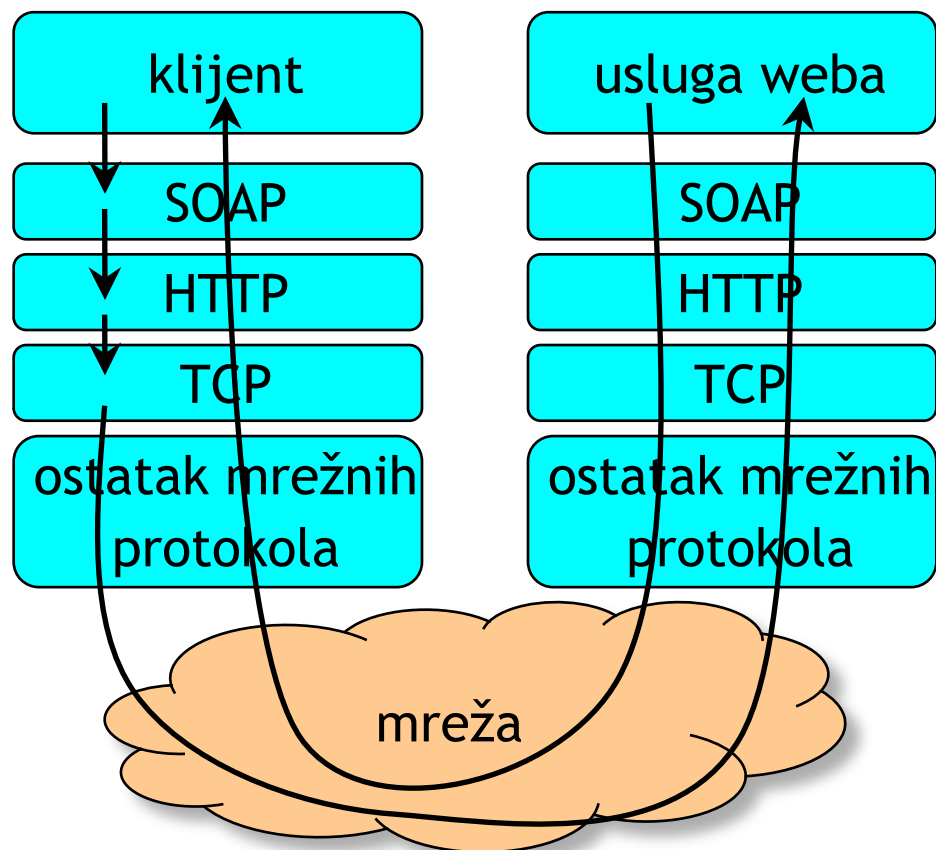
tijelo HTTP-a = poruka SOAP

zaglavlje SOAP-a

tijelo
poruke SOAP







Izgled stvarne poruke SOAP (1)



Zavod za telekomunikacije

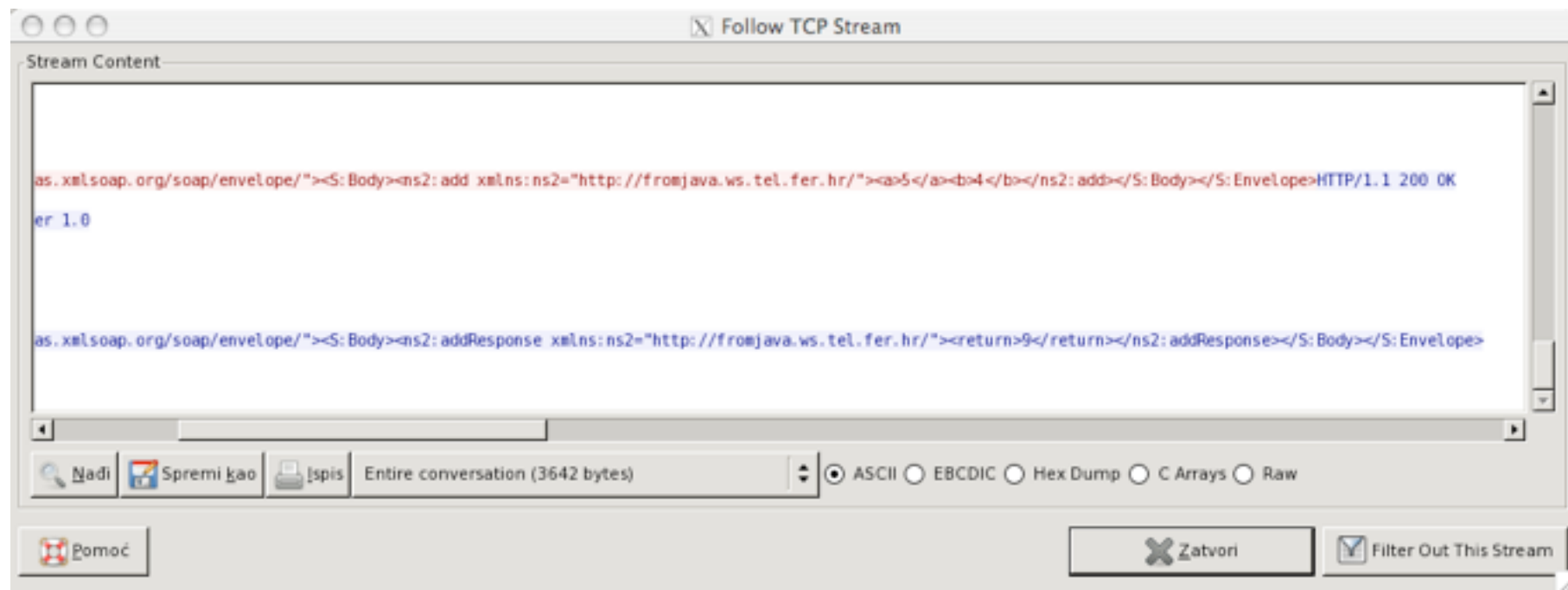
Wireshark interface showing a captured SOAP message. The packet list shows a GET request to /UMPosluzitelj/MyServiceService?wsdl. The packet details pane shows the message structure: Frame 37 (250 bytes on wire, 250 bytes captured) - Null/Loopback - Internet Protocol, Src: 127.0.0.1 (127.0.0.1), Dst: 127.0.0.1 (127.0.0.1) - Transmission Control Protocol, Src Port: 52527 (52527), Dst Port: http-alt (8080), Seq: 489, Ack: 2524, Len: 194 - Hypertext Transfer Protocol - Data (194 bytes) - Data: 3C3F786D6C207665727369666E3022312E3022203F3E3C53...

The packet bytes pane shows the raw data of the SOAP message, including the XML header and body. The XML header is: <?xml version="1.0" ?><Envelope xmlns:="http://schemas.xmlsoap.org/soap/envelope/"><Body><ns2:add x...

Izgled stvarne poruke SOAP (1)



Zavod za telekomunikacije



◆ zahtjev

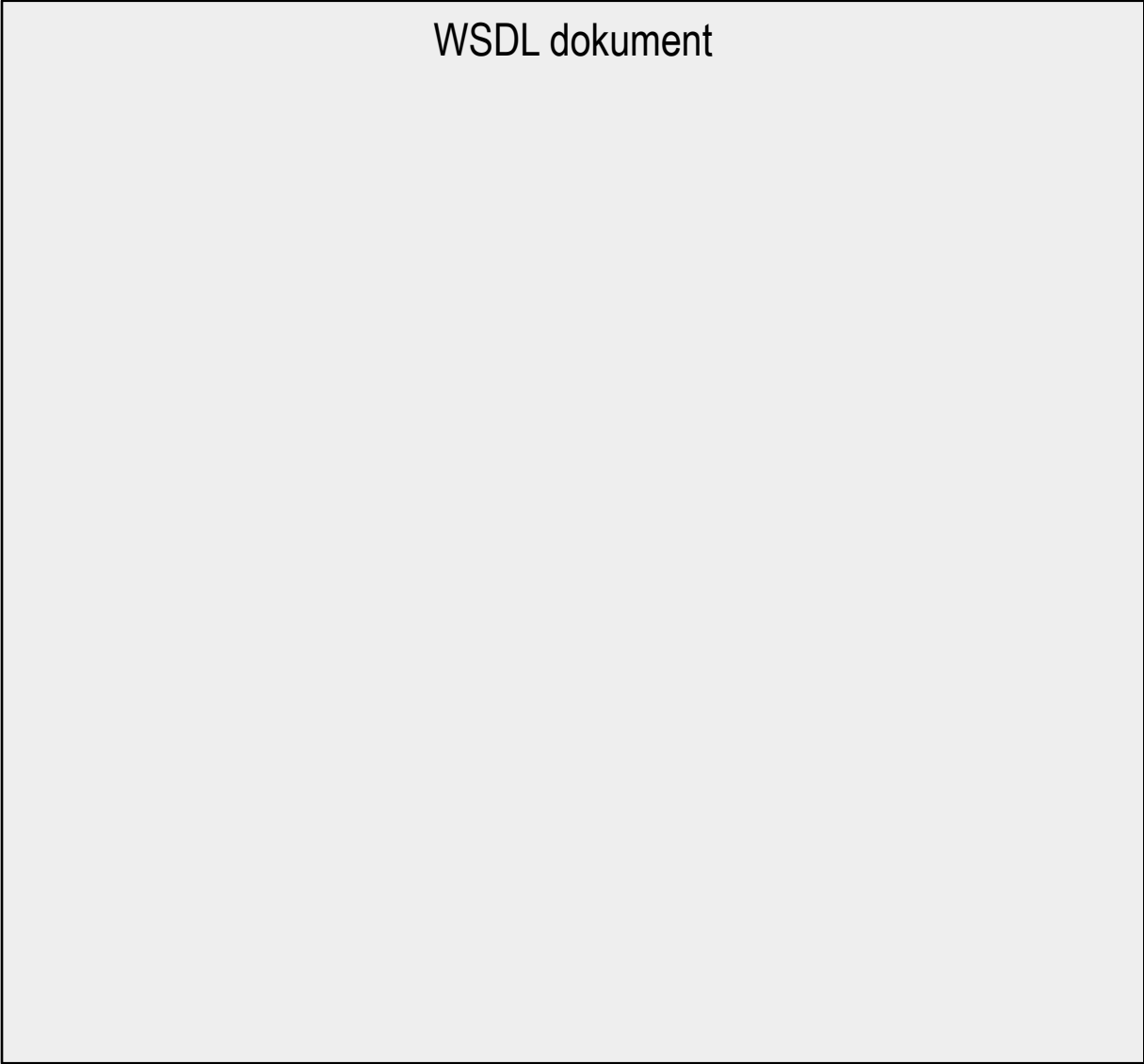
```
<?xml version="1.0" ?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns2:add xmlns:ns2="http://fromjava.ws.tel.fer.hr/">
      <a>5</a>
      <b>4</b>
    </ns2:add>
  </S:Body>
</S:Envelope>
```

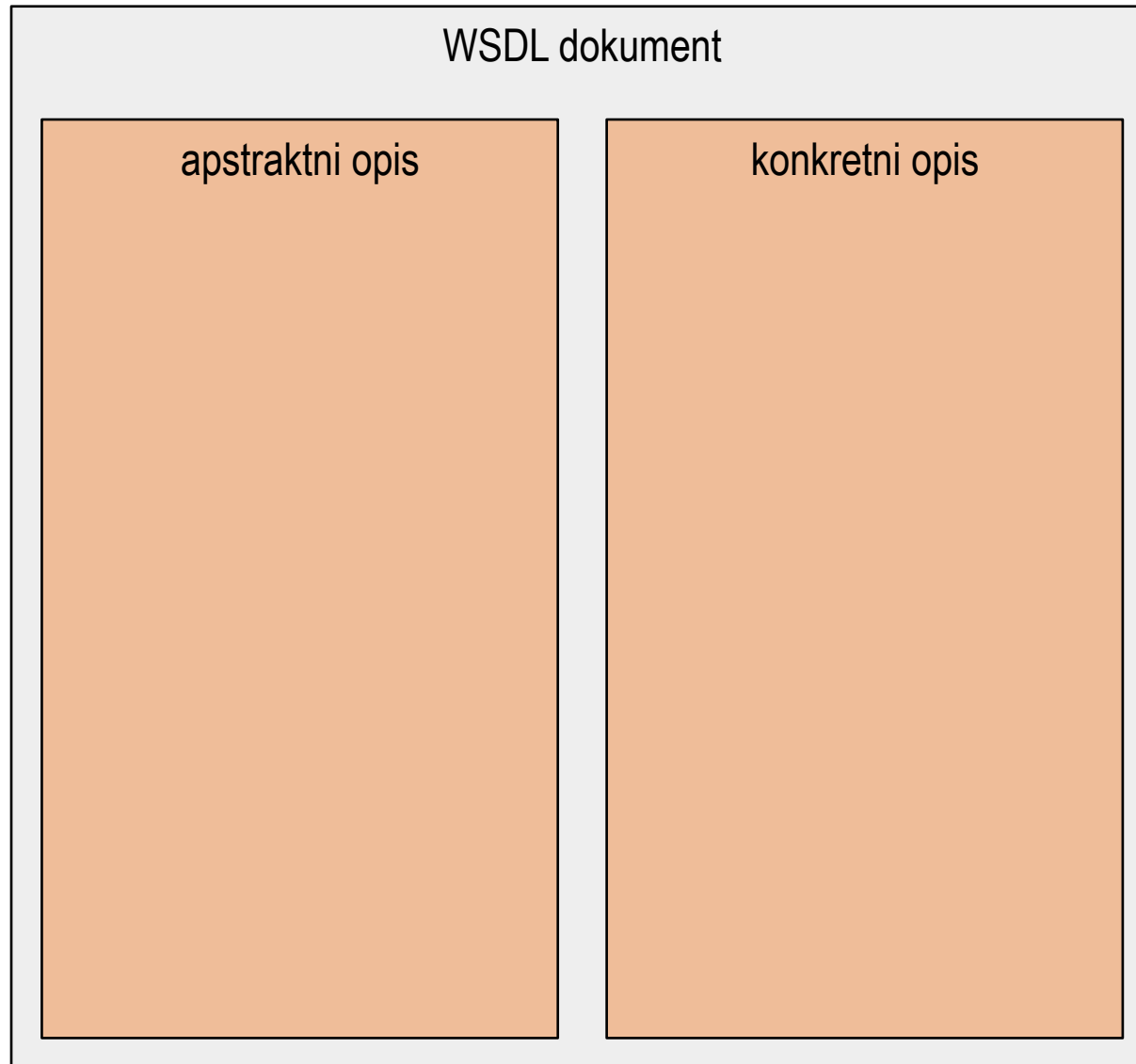
◆ odgovor

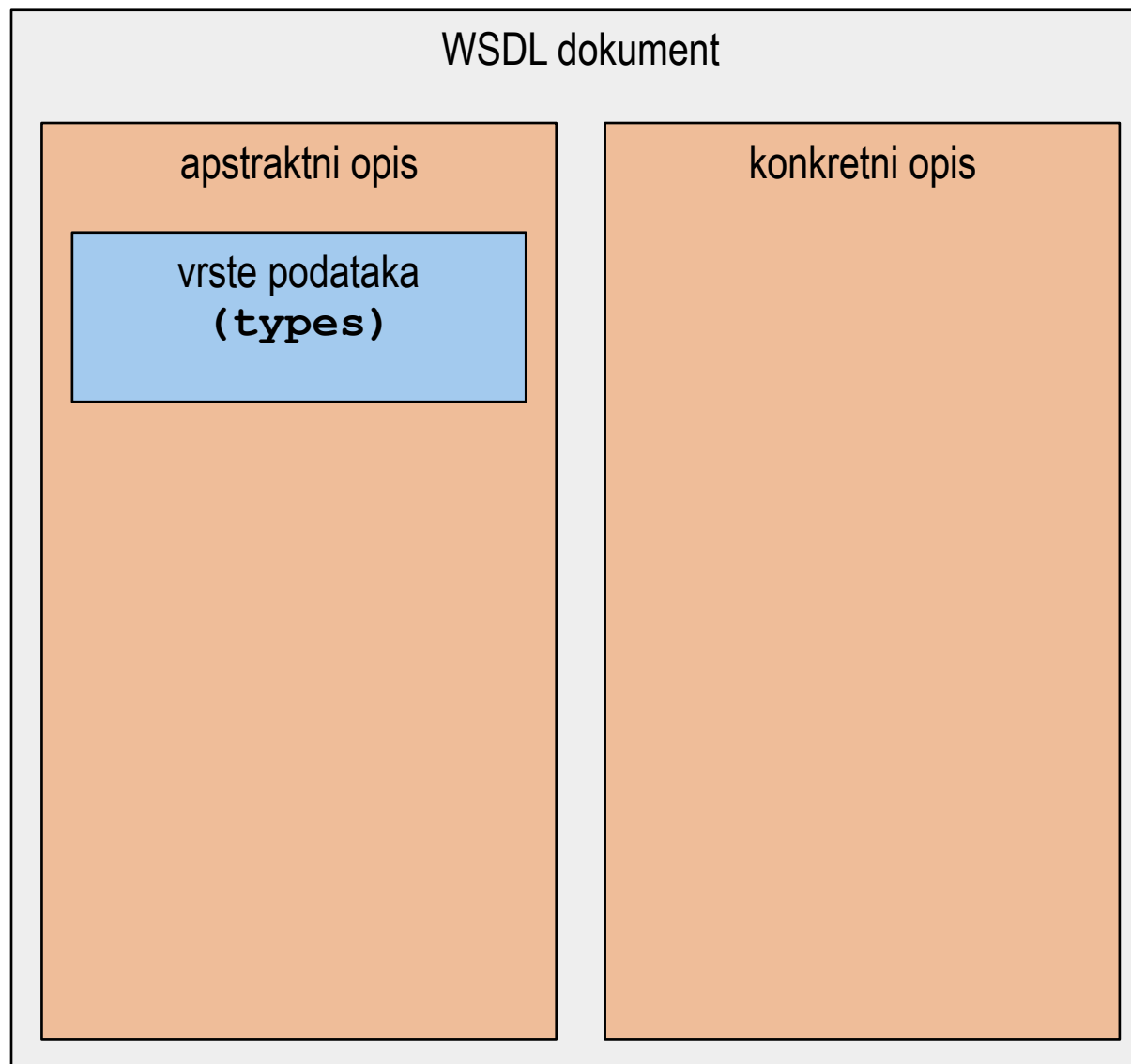
```
<?xml version="1.0" ?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns2:addResponse xmlns:ns2="http://fromjava.ws.tel.fer.hr/">
      <return>9</return>
    </ns2:addResponse>
  </S:Body>
</S:Envelope>
```

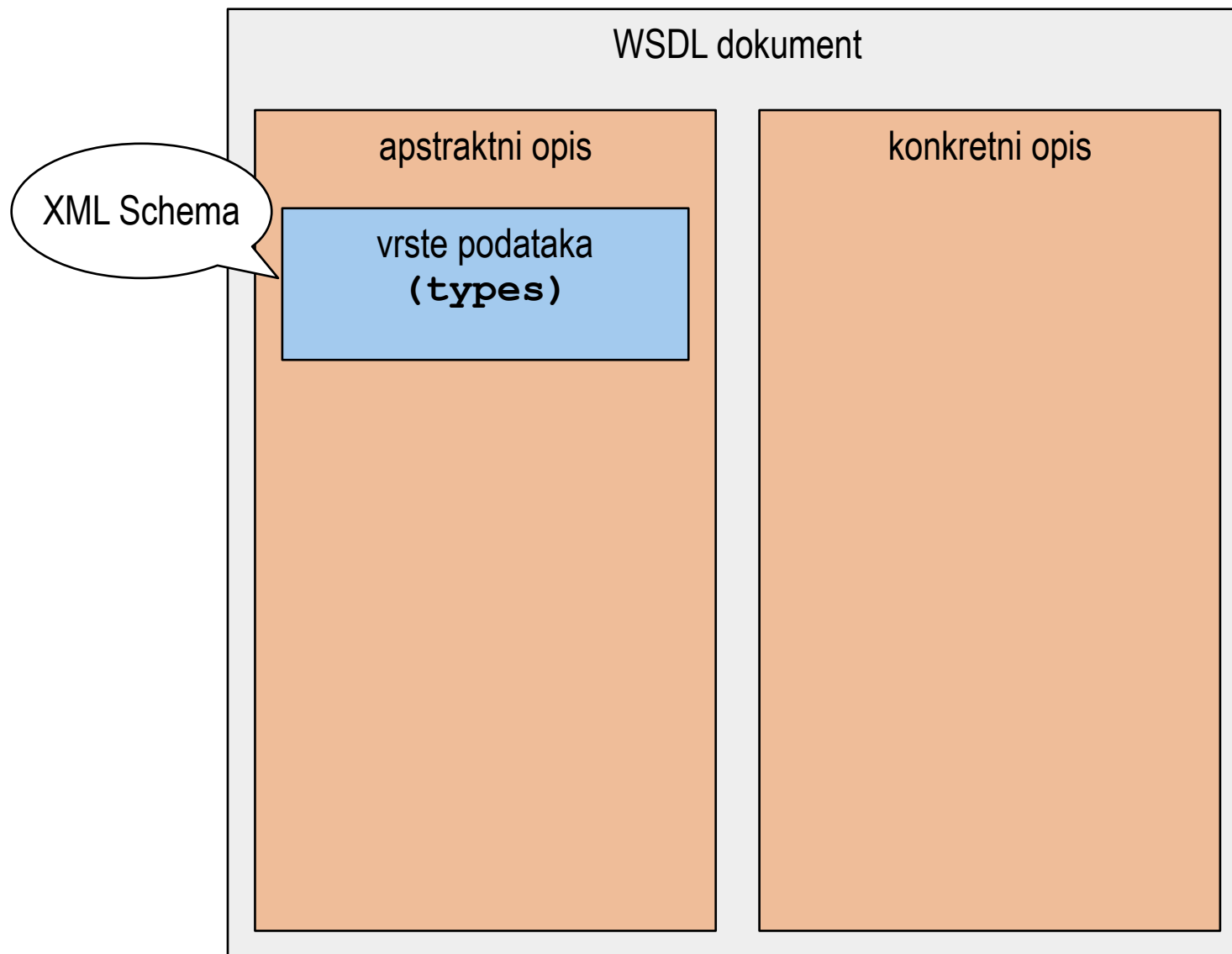

- ◆ jezik za opis usluge weba
- ◆ usluga weba opisana je skupom komunikacijskih krajnjih točaka (*ports*)
- ◆ krajnja točka se sastoji od dva dijela:
 - apstraktne definicije operacija i poruka
 - specifikacije mrežnog protokola i pojedine krajnje točke te formata poruke
- ◆ opisuje komunikacijske detalje između klijenta i usluge
 - strojevi (računala) mogu pročitati WSDL
 - mogu pozvati uslugu definiranu WSDL-om
- ◆ jedan je od mehanizama koji omogućuje da se usluga može otkriti pomoću registra
- ◆ specifikacija: <http://www.w3.org/TR/wsdl> ili <http://www.w3.org/TR/wsdl20>

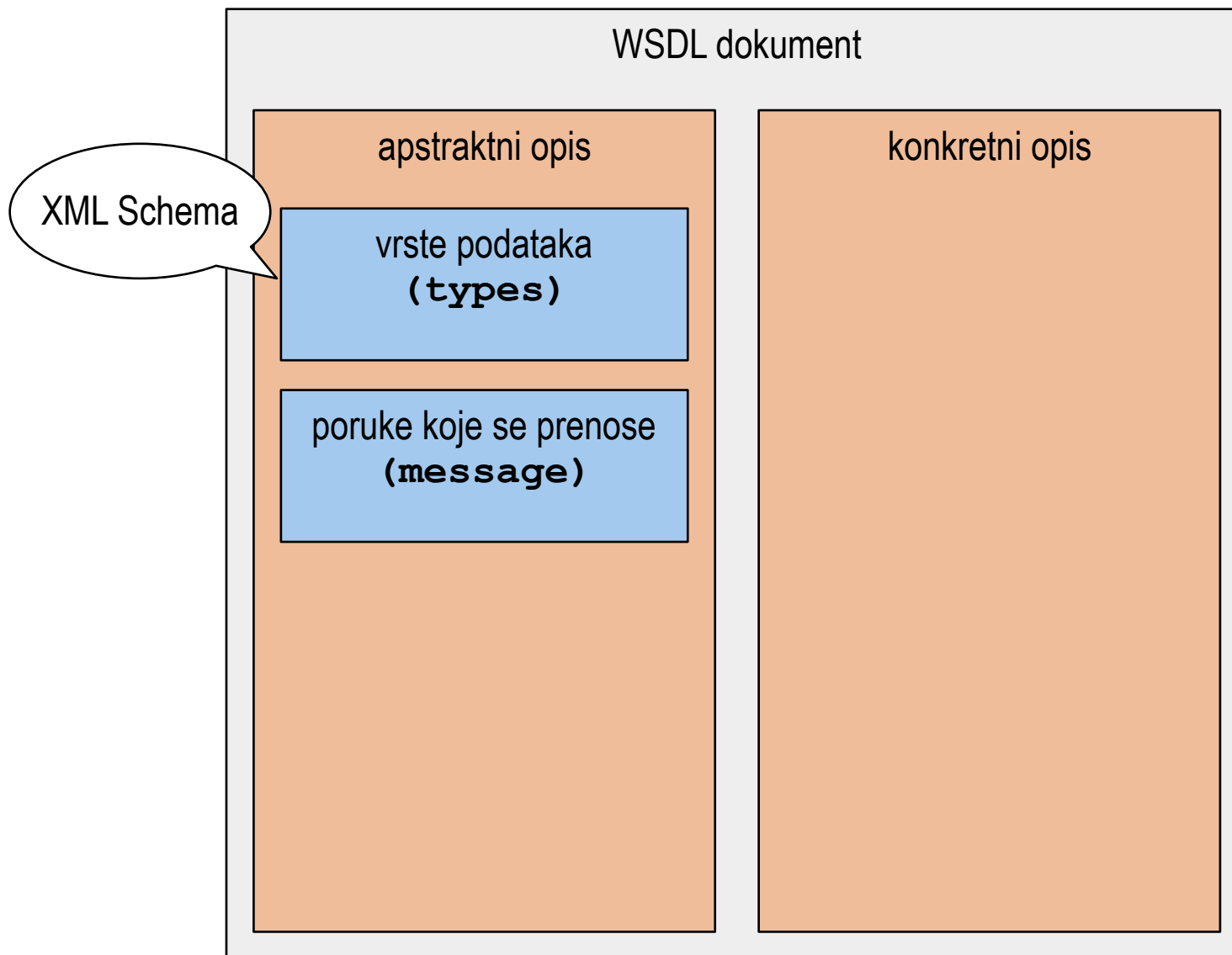
WSDL dokument

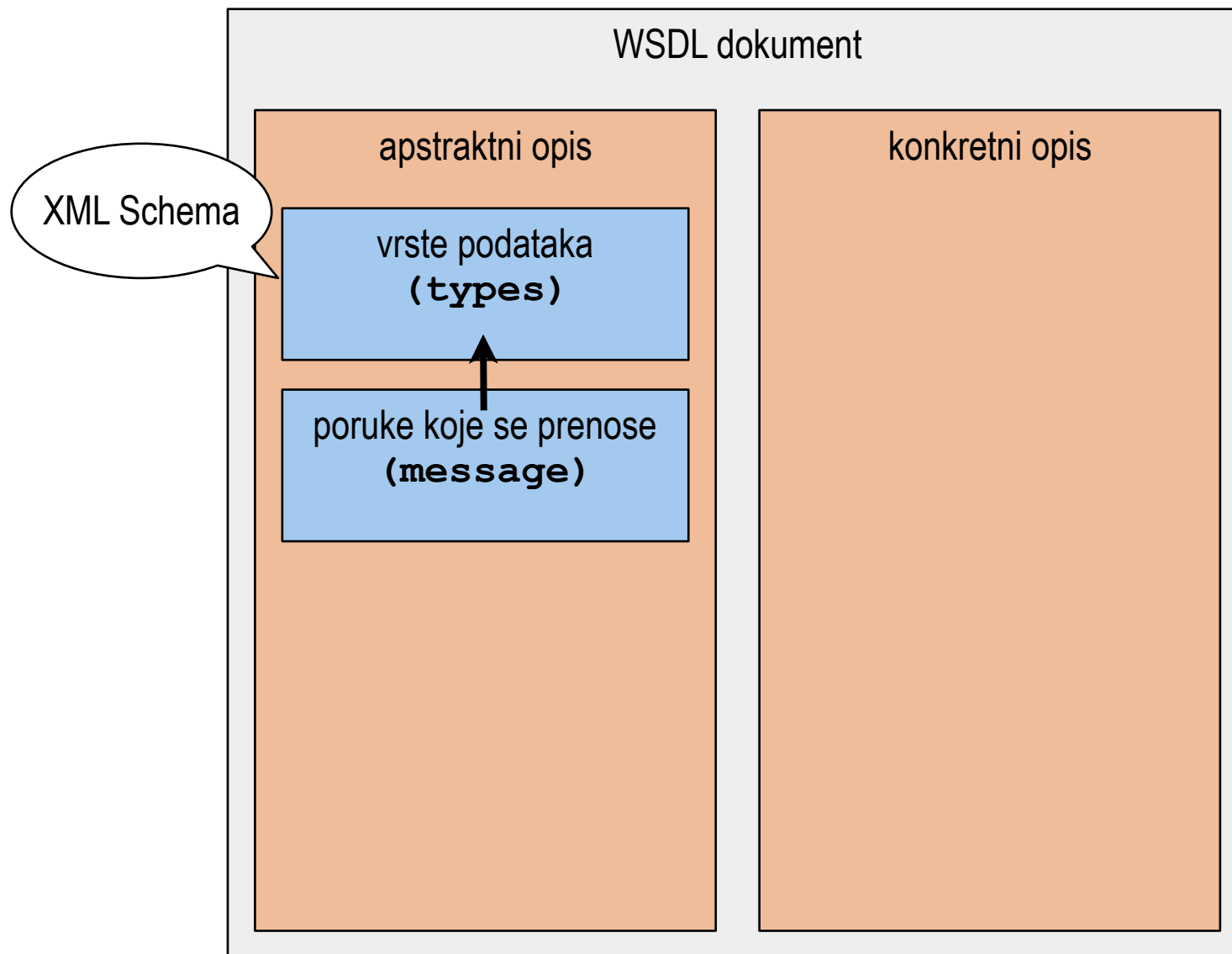
A large, light gray rectangular box with a black border occupies the center of the slide. It is intended for a diagram or content related to the WSDL document structure.

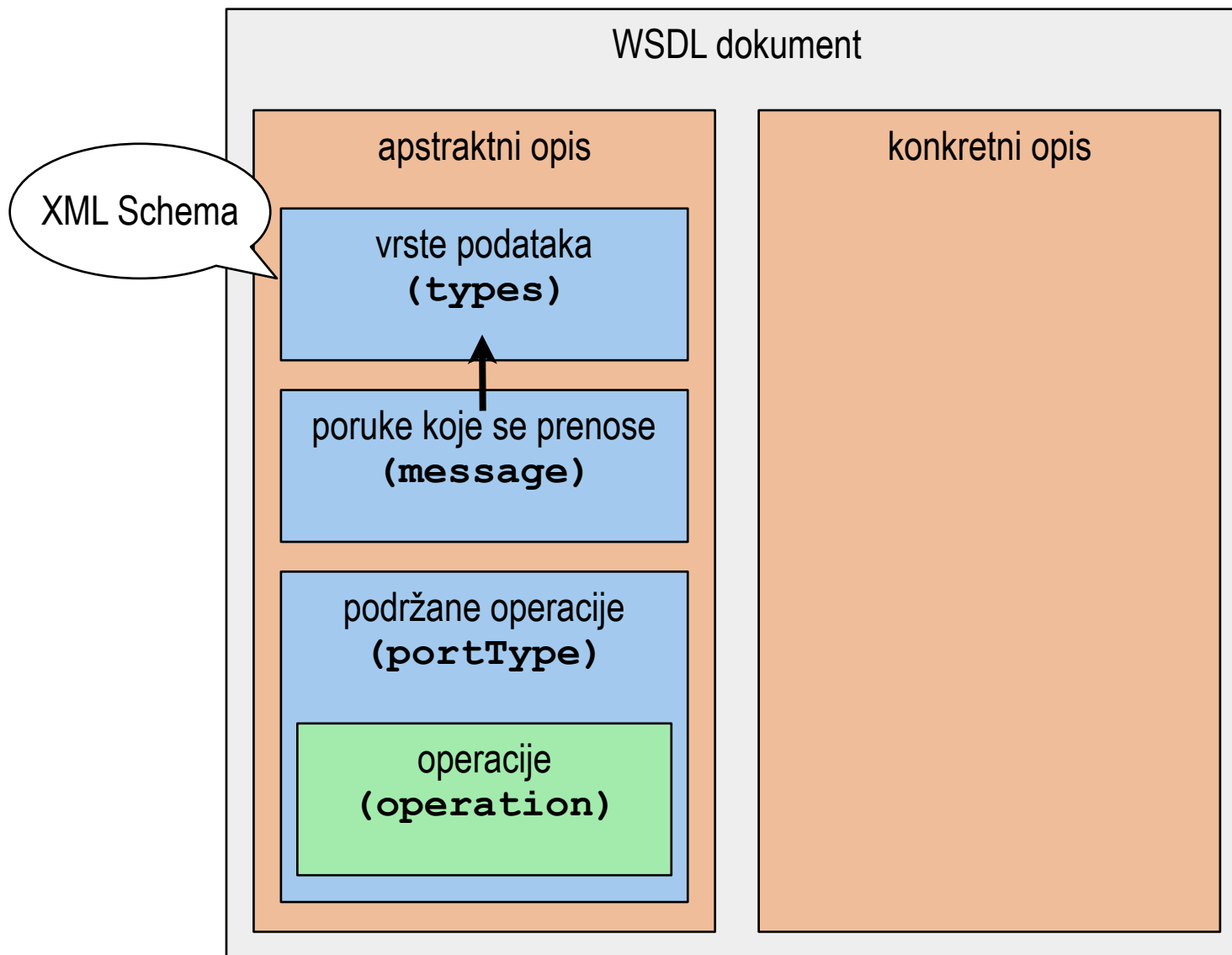


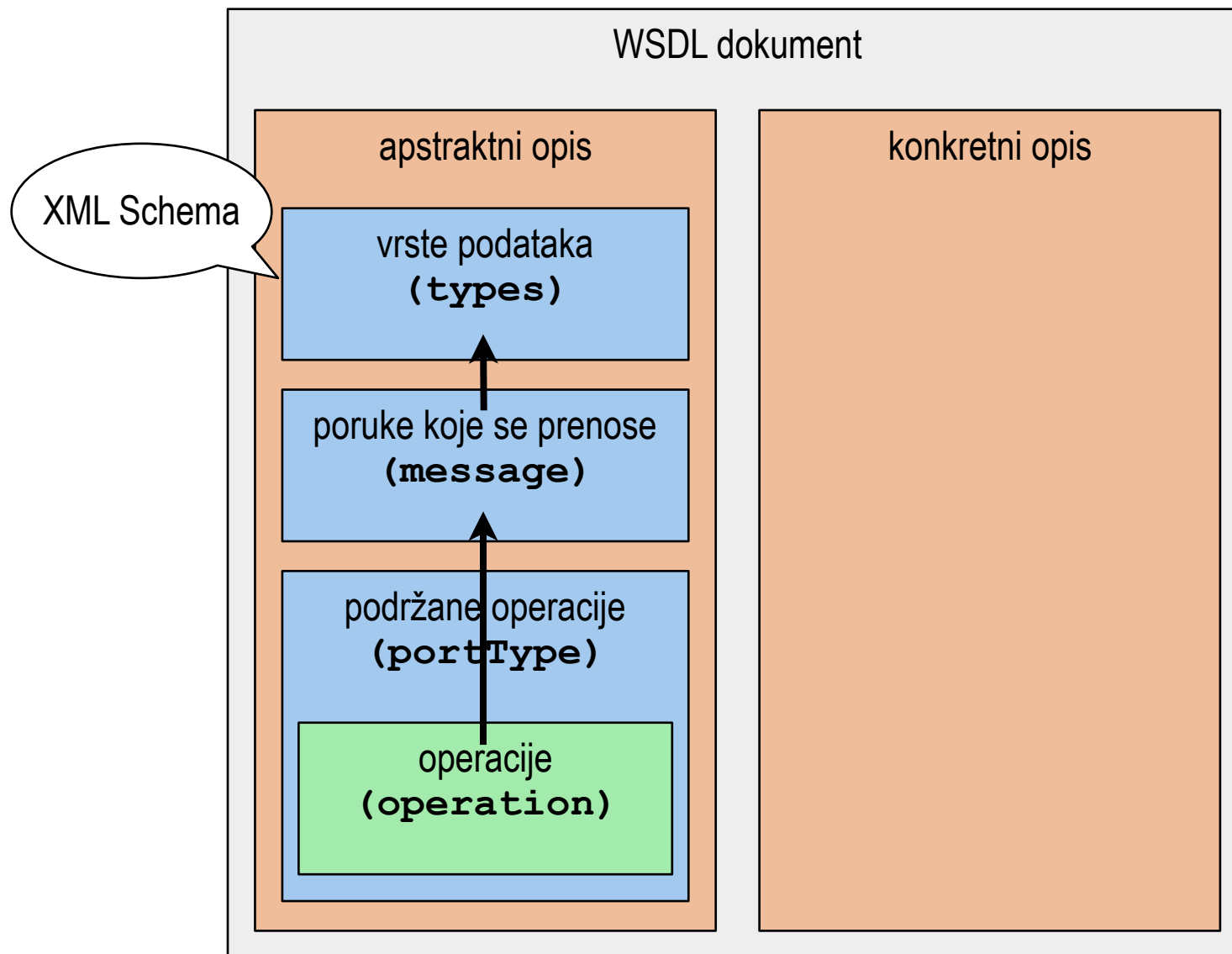


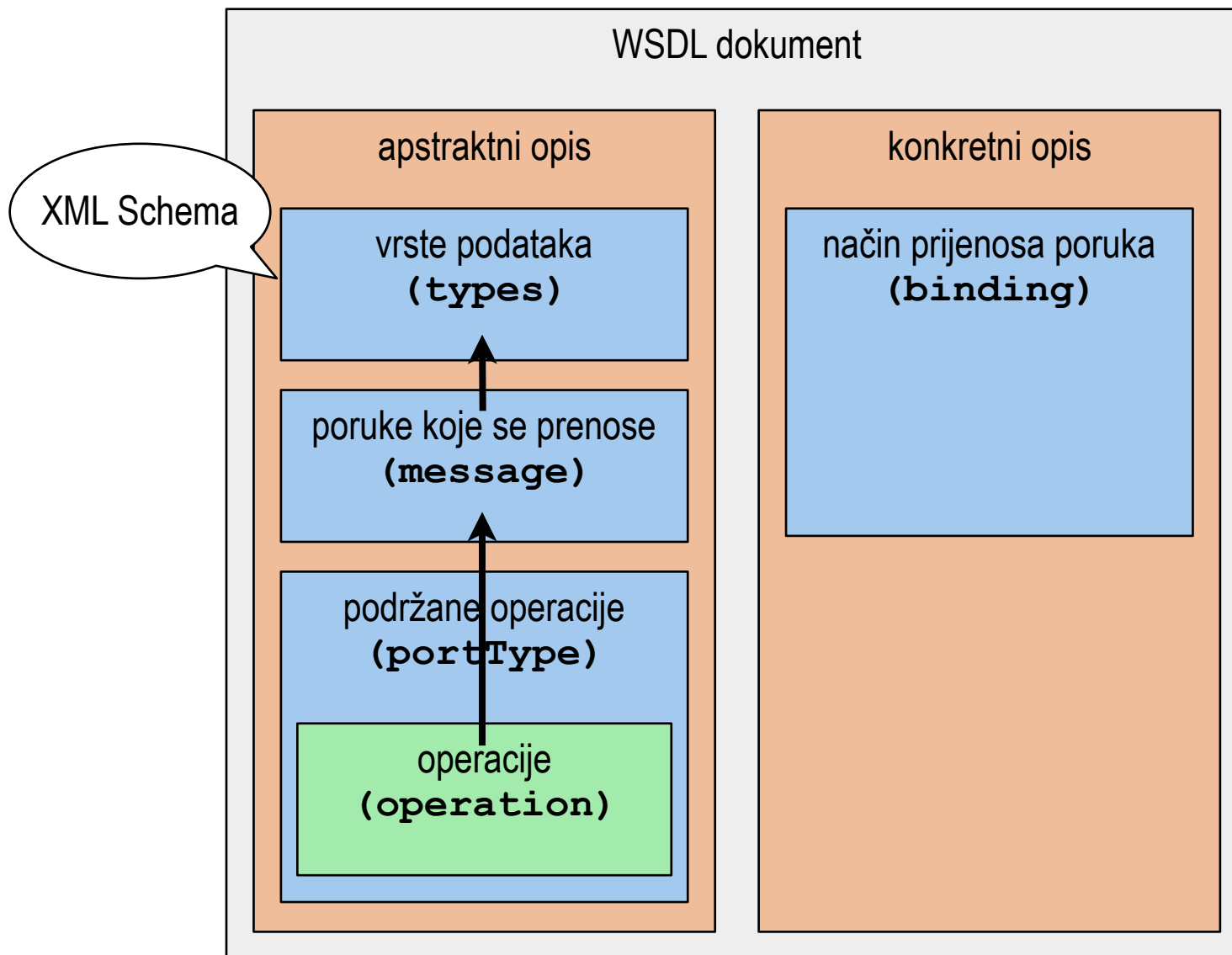


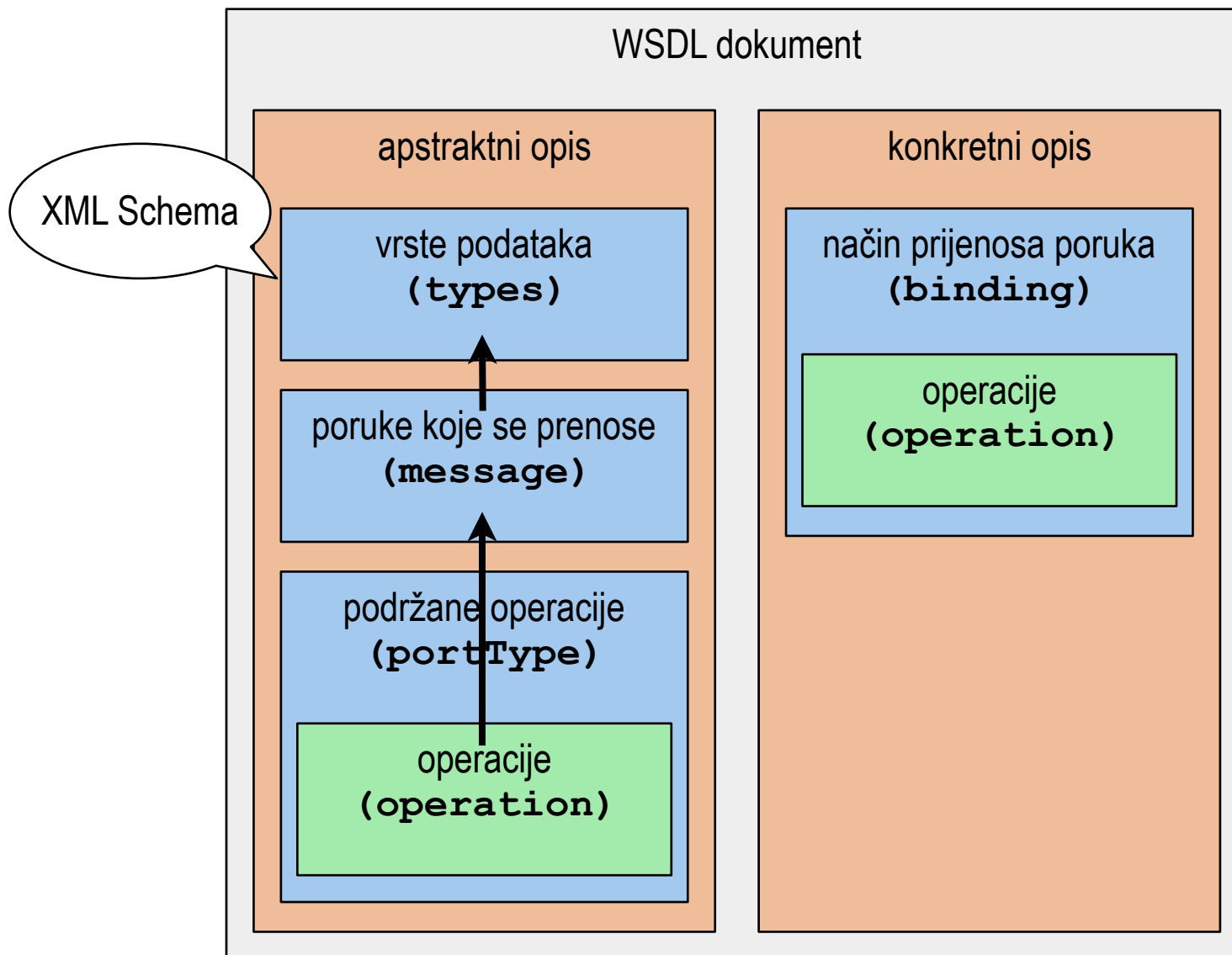


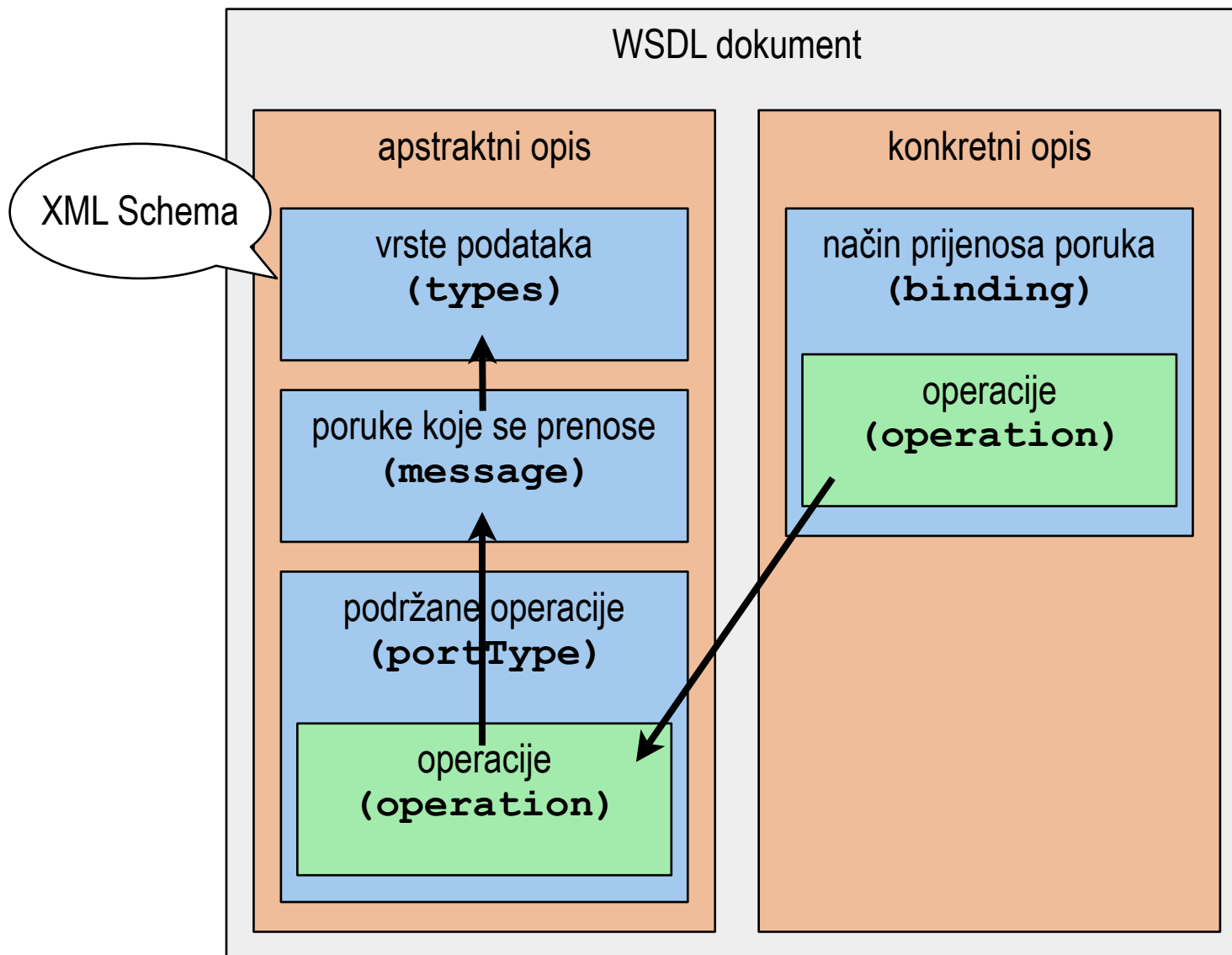


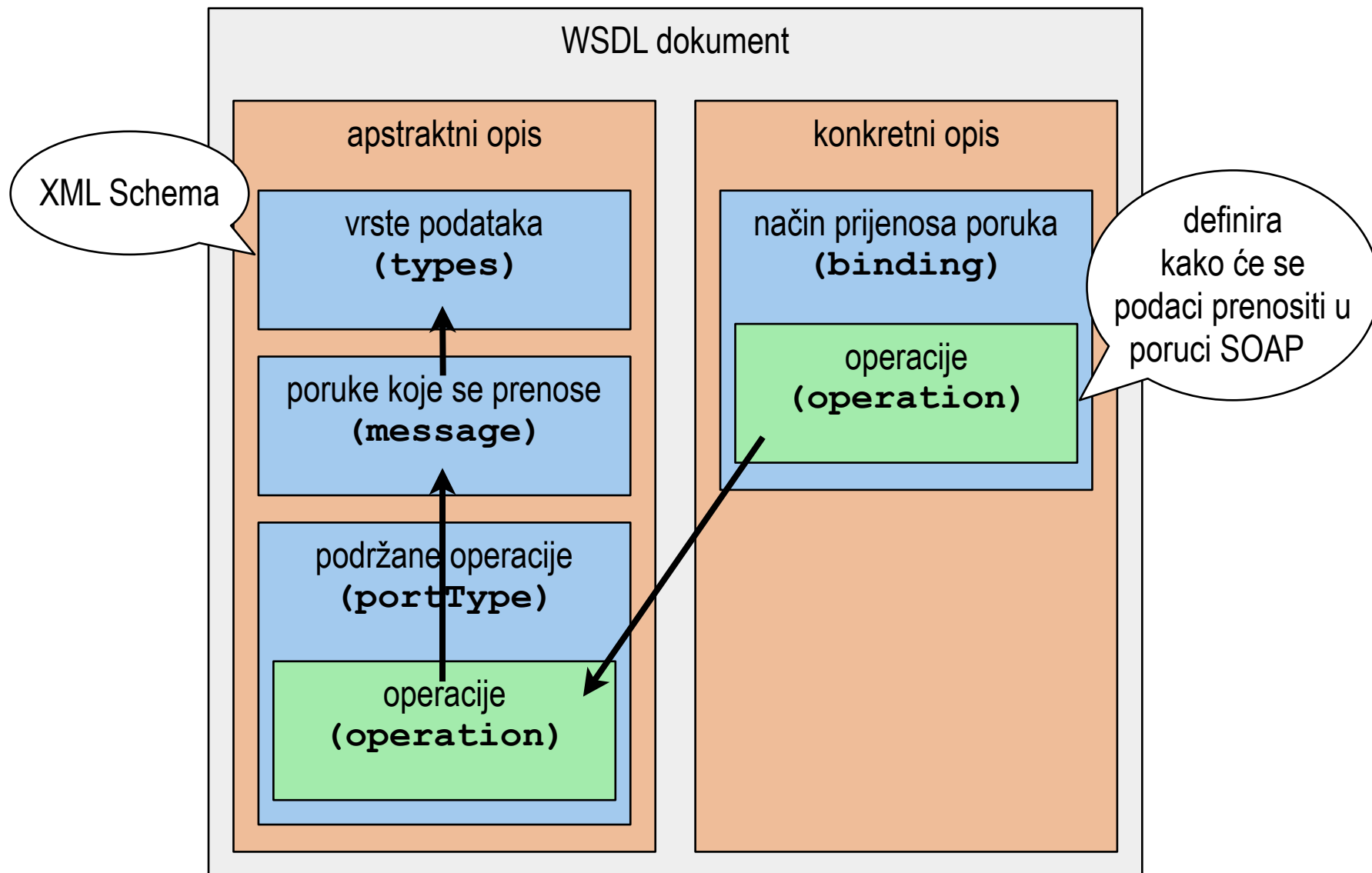


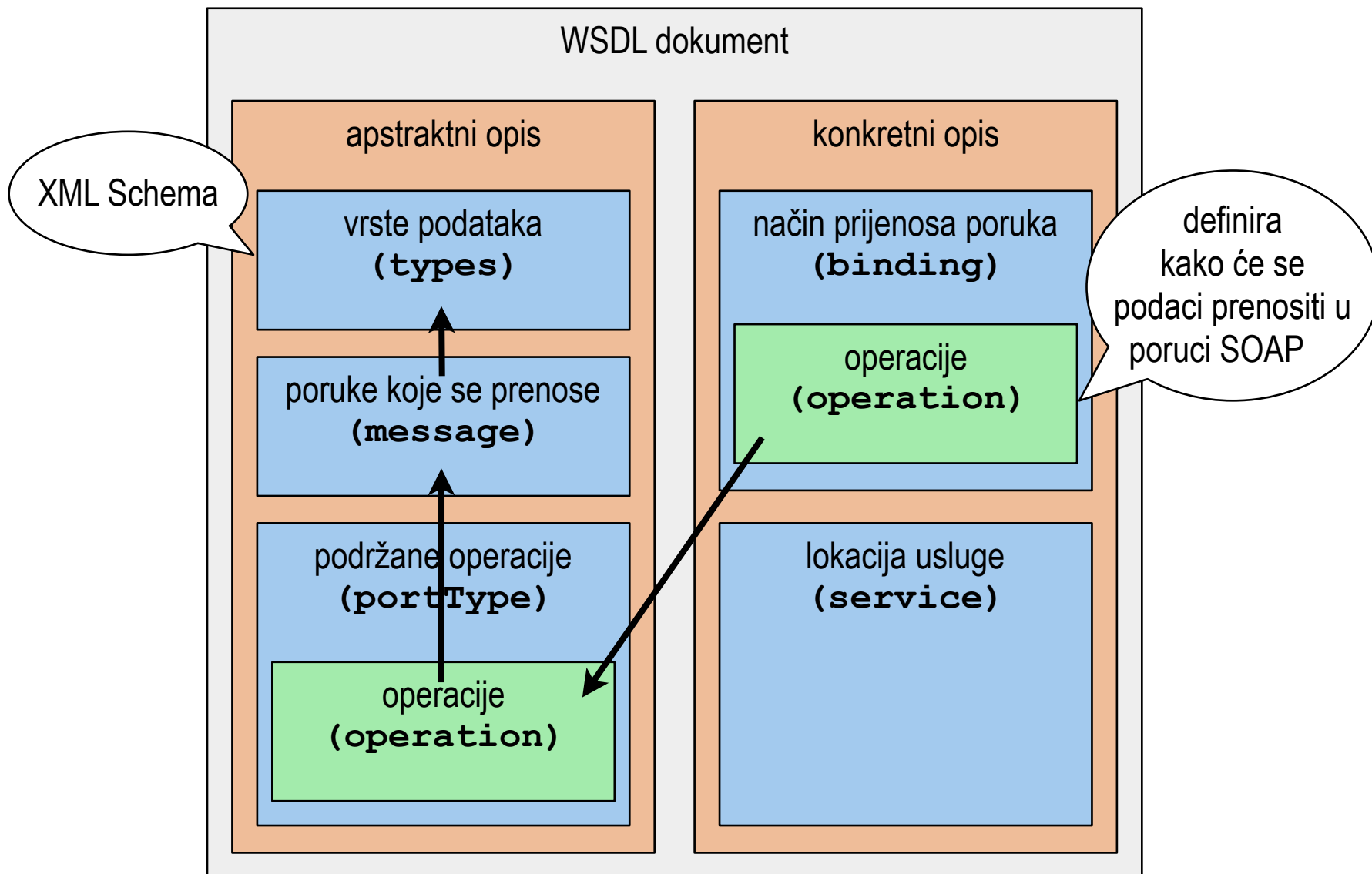


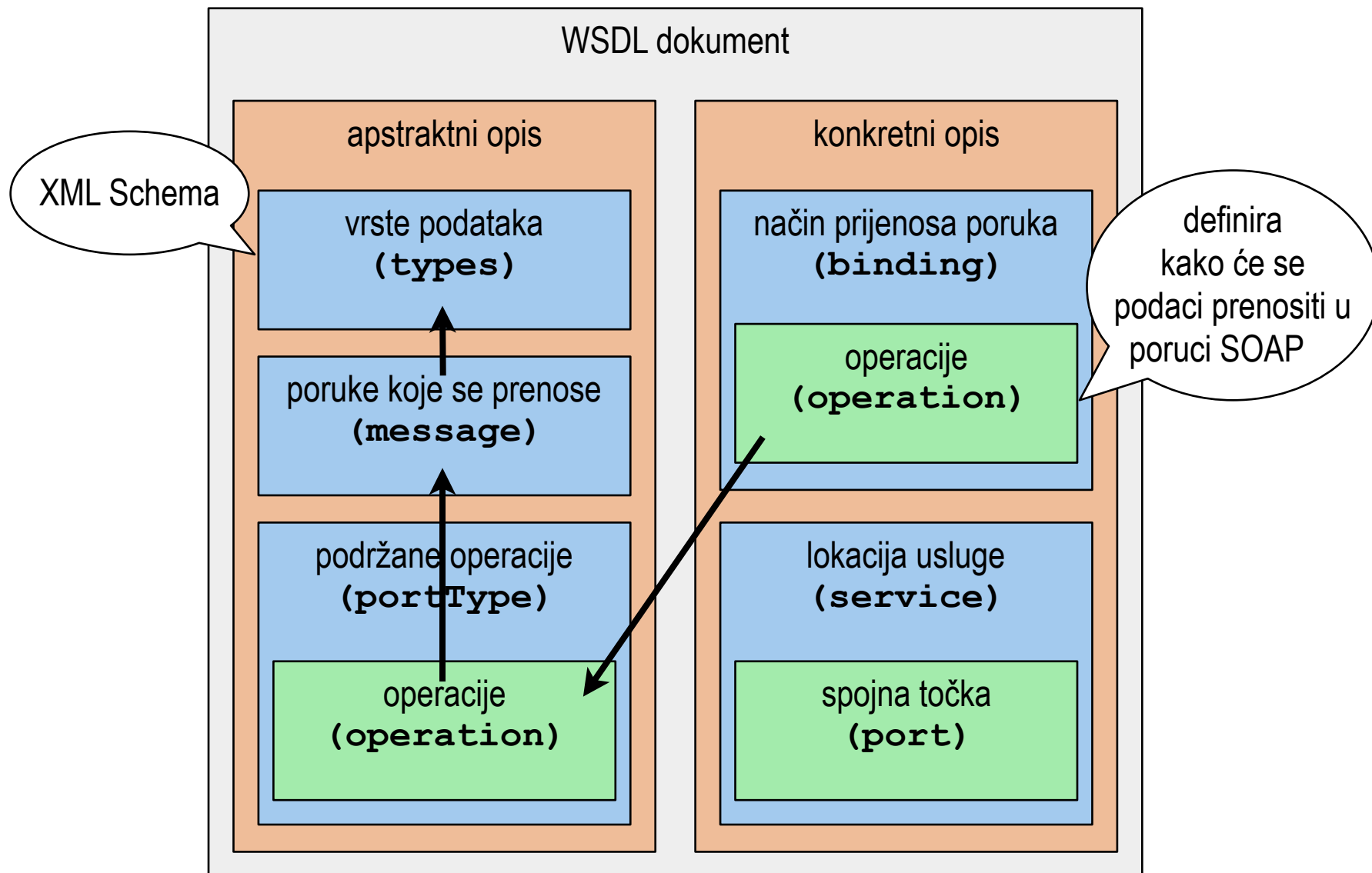


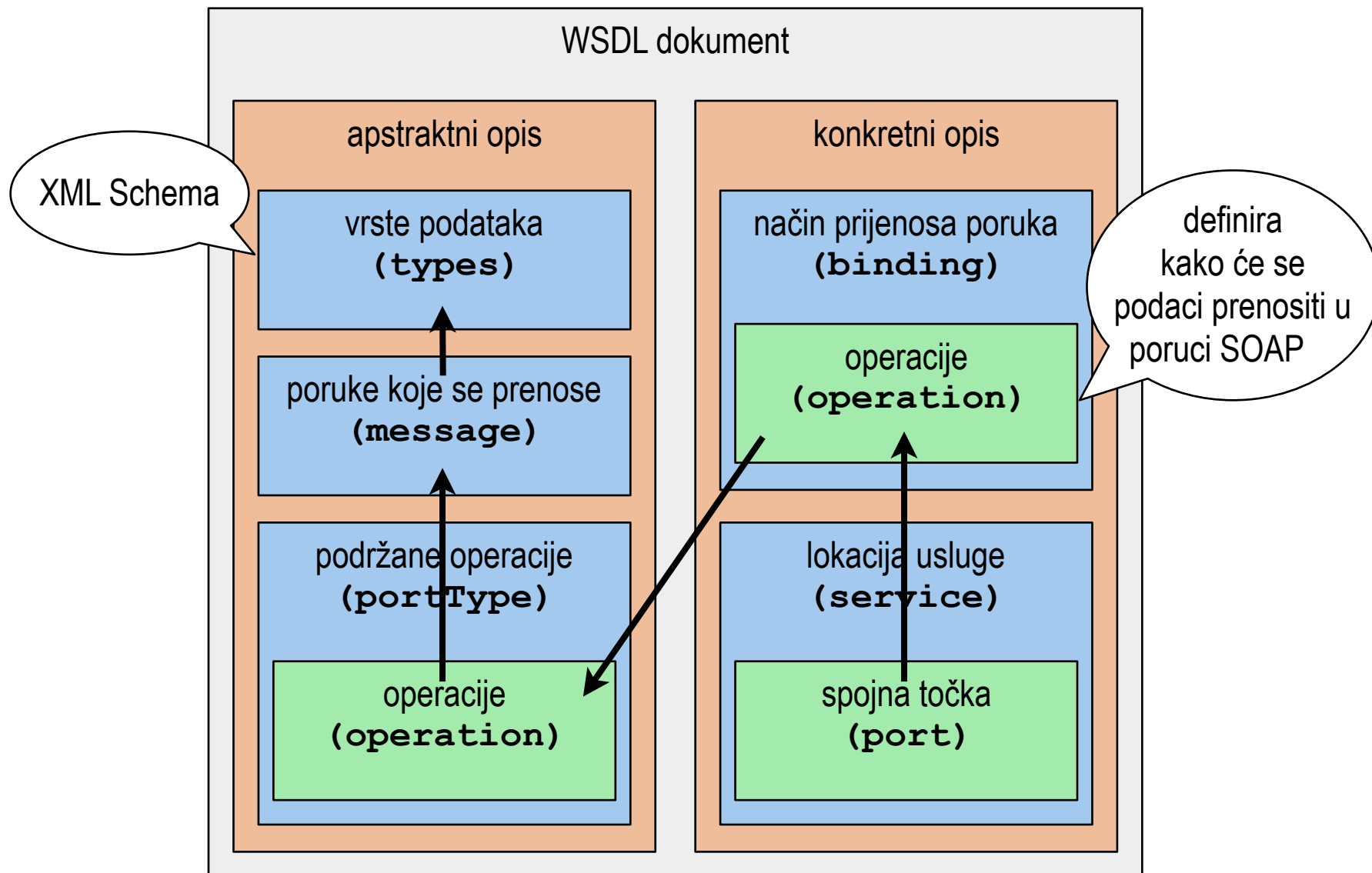












◆ **definitions**

- osnovni element WSDL dokumenta

◆ **types**

- opis podataka pomoću XML Scheme
- nepotreban ako se koriste osnovni podaci iz Scheme

◆ **message**

- opis jednosmjerne poruke
- definira ime poruke
- poruka se sastoji od dijelova (**part**)
- svaki dio se referencira na vrste podataka iz dijela **types**

◆ **portType**

- definira operacije
- svaka operacija se sastoji od poruka (reference na poruke)
- poruke koji mogu biti:
 - parametri (ulazne poruke)
 - vrijednosti koje se vraćaju (rezultati)

◆ **binding**

- definira kako će se poruke iz operacije prenositi
- definira koje transportne protokole će koristiti (HTTP GET, HTTP POST, SOAP, SMTP)
- stil definira vrstu čitave poruke:
 - **rpc** - zahtjev će imati omotač u kojem će pisati naziv funkcije koja se poziva
 - **document** - zahtjev i odgovori će imati “obične” XML dokumente
- poruke mogu koristiti dvije vrste pakiranja poruka:
 - **literal** - onako kako definira Schema
 - **encoded** - kodirano pravilima u SOAP-u

◆ **services**

- definira lokaciju usluge (URL)

◆ **import**

- koristi se za uključivanje drugih WSDL ili XML Schema dokumenata

Primjer jedne usluge opisane u WSDL-u (1)



```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="myWSDL"
  targetNamespace="http://j2ee.netbeans.org/wsdl/myWSDL"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://j2ee.netbeans.org/wsdl/myWSDL"
  xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/
```

Primjer jedne usluge opisane u WSDL-u (2)



Zavod za komunikacije

```
<binding name="myBinding" type="tns:myPortType">
  <soap:binding style="rpc"
    transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="myOperation">
    <soap:operation/>
    <input name="input1">
      <soap:body use="literal"
        namespace="http://j2ee.netbeans.org/wsdl/myWSDL"/>
    </input>
    <output name="output1">
      <soap:body use="literal"
        namespace="http://j2ee.netbeans.org/wsdl/myWSDL"/>
    </output>
  </operation>
</binding>
<service name="myService">
  <port name="myPort" binding="tns:myBinding">
    <soap:address
      location="http://ws.tel.fer.hr:8080/myService/myPort"/>
  </port>
</service>
<plnk:partnerLinkType name="myWSDL">
  <plnk:role name="myPortTypeRole" portType="tns:myPortType"/>
</plnk:partnerLinkType>
</definitions>
```

- ◆ UDDI (*Universal Description, Discovery and Integration*)
- ◆ osigurava platformu za otkrivanje usluga na Internetu
- ◆ sastoji se od 3 dijela:
 - imenik (*White pages*)
 - ▶ adrese, kontakti i identifikatori
 - poslovni imenik (*Yellow pages*)
 - ▶ kategorizacija područja, usluga i proizvoda te lokacije
 - tehničke informacije (*Green pages*)
 - ▶ sadrži tehničke informacije o uslugama
- ◆ nema centralnog registra
- ◆ više informacija: <http://www.uddi.org/>

Što nam je potrebno za izradu usluge weba



Zavod za telekomunikacije

♦ trebamo imati:

■ alat

- npr. NetBeans 6.x
 - ▶▶ <http://www.netbeans.org/>



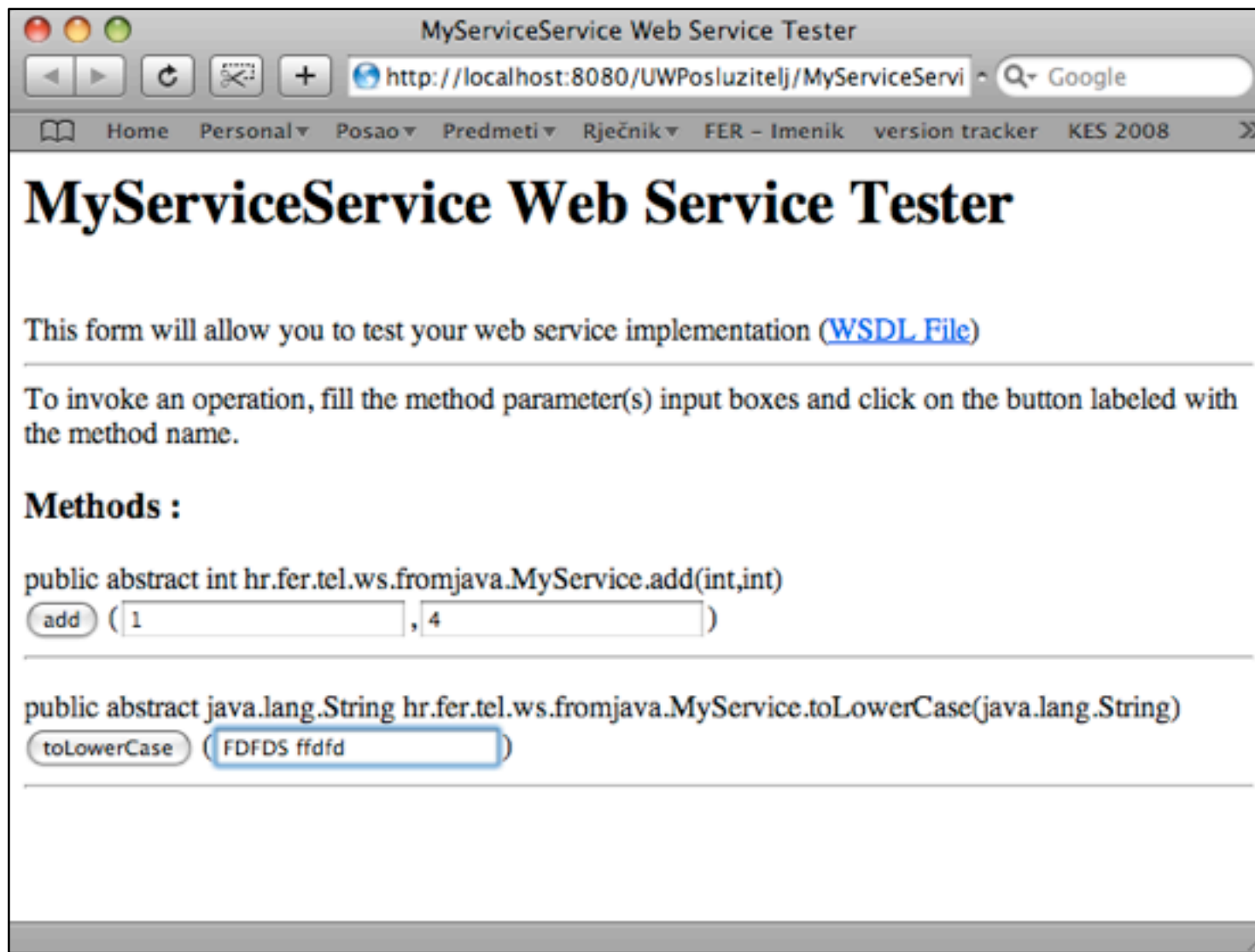
■ aplikacijski poslužitelj

- npr. GlassFish
 - ▶▶ <https://glassfish.dev.java.net>



- ◆ napravimo novi projekt
- ◆ stvorimo novu uslugu weba i stvorimo operacije:

```
@WebService()  
public class MyService {  
    @WebMethod(operationName = "add")  
    public int add(@WebParam(name = "a") int a,  
                  @WebParam(name = "b") int b) {  
        return a + b;  
    }  
  
    @WebMethod(operationName = "toLowerCase")  
    public String toLowerCase(  
        @WebParam(name = "text") String text) {  
        return text.toLowerCase();  
    }  
}
```



MyServiceService Web Service Tester

http://localhost:8080/UWPosluzitelj/MyServiceServi

Home Personal Posao Predmeti Rječnik FER – Imenik version tracker KES 2008

MyServiceService 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 int hr.fer.tel.ws.fromjava.MyService.add(int,int)

add (1 , 4)

public abstract java.lang.String hr.fer.tel.ws.fromjava.MyService.toLowerCase(java.lang.String)

toLowerCase (FDFDS ffdfd)

- ◆ treba stvoriti novi *Web Service Client*
 - WSDL možemo preuzeti iz: drugog projekta, datoteke, URL-a na WSDL
- ◆ u *Web Service References* djelu je sada prikazana usluga
- ◆ mišem treba prevući operaciju u kod gdje želimo pozvati uslugu
- ◆ tamo se generira sljedeći kod:

```
try {  
    MyServiceService service = new MyServiceService();  
    MyService port = service.getMyServicePort();  
    int a = 5;  
    int b = 4;  
  
    int result = port.add(a, b);  
    System.out.println("Result = " + result);  
} catch (Exception ex) { }
```

- ◆ složeni parametar je instanca klase (npr. Person)

- ◆ postupak:

- napraviti klasu

```
public class Person {  
    private String firstName, lastName;  
    private int age;  
  
    public int getAge() {  
        return age;  
    }  
  
    public void setAge(int age) {  
        this.age = age;  
    }  
  
    ...  
}
```

- napraviti uslugu weba s operacijom koja ima parametar klasu Person

complexOperation			
Parameters	Output	Faults	Description
Parameter Name	Parameter Type		
parameter	hr.fer.tel.ws.fromjava.Person		

◆ implementacija usluge

```
@WebMethod(operationName = "complexOperation")
public String complexOperation(@WebParam(name = "parameter")
                               Person parameter) {
    if(parameter.getAge() > 25)
        return parameter.getFirstName() + " is older than me.";
    else
        return parameter.getFirstName() + " is younger than me.";
}
```

◆ pozivanje usluge iz programa

```
try { // Call Web Service Operation
    MyServiceService service = new MyServiceService();
    MyService port = service.getMyServicePort();
    Person parameter = new Person();
    parameter.setAge(30);
    parameter.setFirstName("Ivan");
    parameter.setLastName("Horvat");
    String result = port.complexOperation(parameter);
    System.out.println("Result = " + result);
} catch (Exception ex) {
    // TODO handle custom exceptions here
}
```

Primjer korištenja gotovih usluga (1)



Zavod za komunikacije

- ◆ pod Service dijelom imamo već postavljene usluge
- ◆ možemo dodati neke druge usluge
- ◆ korištenje postojećih usluga - npr. *Yahoo News Search Service* (<http://developer.yahoo.com/search/news/V1/newsSearch.html>)
 - operaciju prevučemo u program koji ju poziva

```
try {
    String query = "distributed programming";
    String type = "all";
    Integer results = 10;
    Integer start = 1;
    String sort = "rank";
    String language = "";
    String output = "xml";
    String callback = null;

    RestResponse result = YahooNewsSearchService.search(query, type, results,
        start, sort, language, output, callback);
    ResultSet resultObj = result.getDataAsObject(ResultSet.class);
    System.out.println("The SaaSService returned: " +
        result.getDataAsString());
} catch (Exception ex) {
    ex.printStackTrace();
}
```

Primjer korištenja gotovih usluga (2)



Zavod za telekomunikacije

- u projektu su generirana dva paketa:
 - `org.netbeans.saas` i `org.netbeans.saas.yahoo`
- u drugom paketu se nalazi datoteka:
`yahoonewssearchserviceauthenticator.properties`
 - u nju treba staviti ključ koji se može dobiti registracijom (besplatno)
- pokrenuti program

Primjer korištenja gotovih usluga (3)



Zavod za telekomunikacije

```
<?xml version="1.0" encoding="UTF-8"?>
<ResultSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="urn:yahoo:yn" xsi:schemaLocation="urn:yahoo:yn http://
api.search.yahoo.com/NewsSearchService/V1/NewsSearchResponse.xsd"
totalResultsAvailable="146" totalResultsReturned="10"
firstResultPosition="1">
```

```
<Result>
```

```
  <Title>Nero Delivers on Vision for &quot;Liquid&quot; Content
Creation and Distribution with Launch of Three New Products</Title>
```

```
  <Summary>KARLSBAD, Germany, BUSINESS WIRE -- Nero , creators of
Liquid Media technology, today announced breakthrough new software
offerings that will enable content to be easily created and
distributed anytime, anywhere and on any device.</Summary>
```

```
  <Url>http://www.broadcastnewsroom.com/articles/viewarticle.jsp?
id=531798</Url>
```

```
  <ClickUrl>http://www.broadcastnewsroom.com/articles/
viewarticle.jsp?id=531798</ClickUrl>
```

```
  <NewsSource>Broadcast Newsroom</NewsSource>
```

```
  <NewsSourceUrl>http://www.broadcastnewsroom.com/</NewsSourceUrl>
```

```
  <Language>en</Language>
```

```
  <PublishDate>1222692949</PublishDate>
```

```
  <ModificationDate>1222692950</ModificationDate>
```

```
</Result>
```

...

◆ RPC

- implicira se poziv procedure/funkcije/metode
- obično su sinkrone
- sadržaj su parametri poziva procedure i rezultat izvršavanja

◆ Usluge temeljene na dokumentima

- razmjenjuju se dokumenti
- dogovor o dokumentima (XML Schema)
- obično su asinkrone
- parametri kod povezivanja u WSDL-u su: Document-literal

◆ Postupak:

1. definiranje XML Scheme dokumenata

- obično u posebnoj datoteci

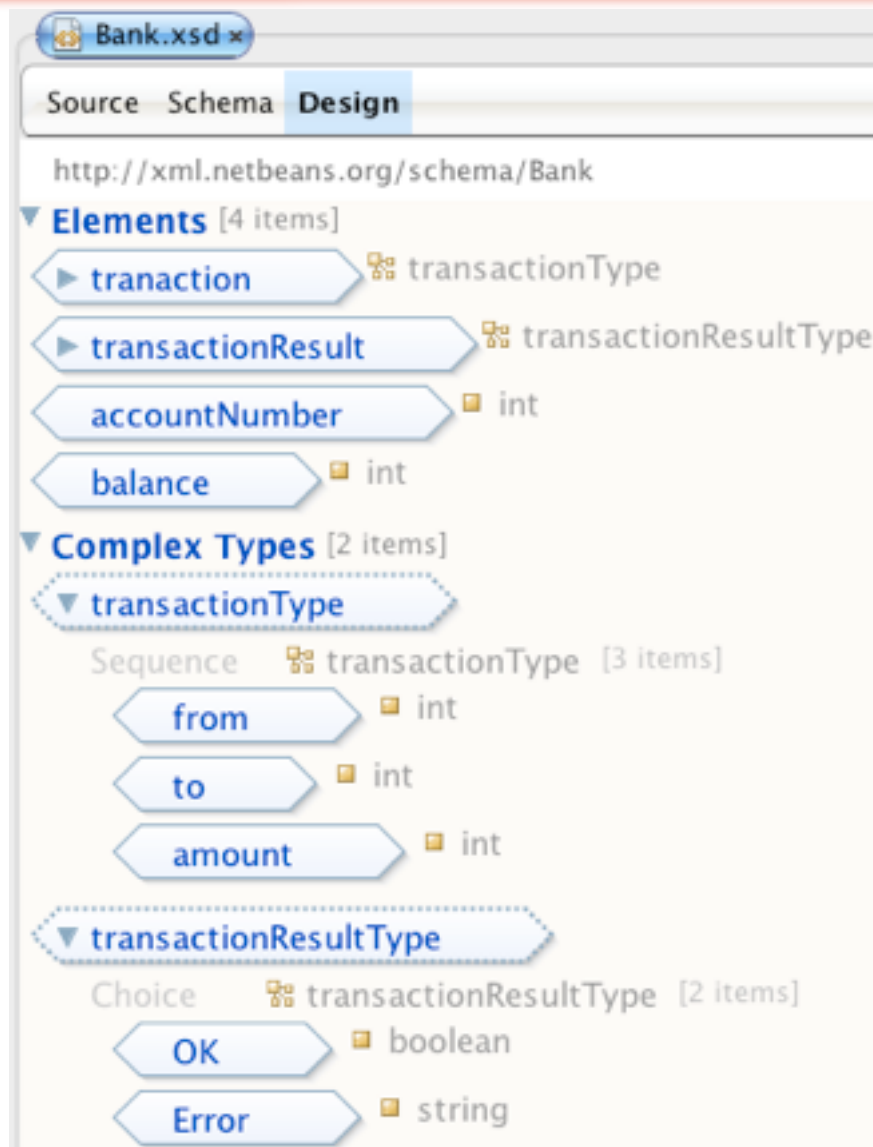
2. definiranje WSDL-a

- uključuje XML Scheme

Izrada scheme (1)



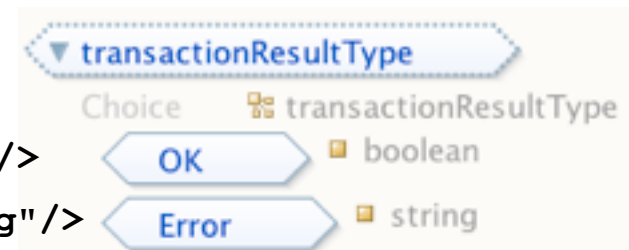
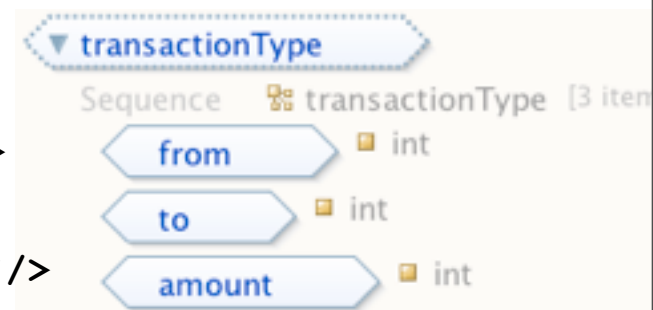
Zavod za telekomunikacije



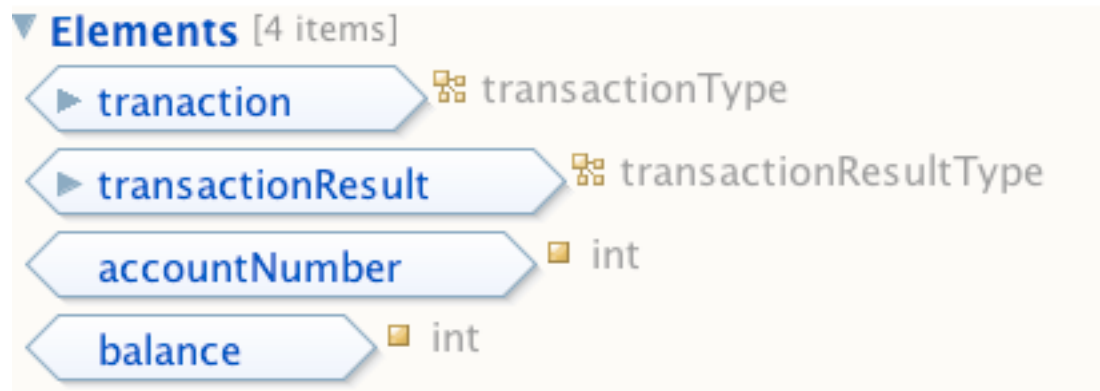
Izrada scheme (2)



```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://xml.netbeans.org/schema/Bank"
  xmlns:tns="http://xml.netbeans.org/schema/Bank"
  elementFormDefault="qualified">
  <xsd:complexType name="transactionType">
    <xsd:sequence>
      <xsd:element name="from" type="xsd:int"/>
      <xsd:element name="to" type="xsd:int"/>
      <xsd:element name="amount" type="xsd:int"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="transactionResultType">
    <xsd:choice>
      <xsd:element name="OK" type="xsd:boolean"/>
      <xsd:element name="Error" type="xsd:string"/>
    </xsd:choice>
  </xsd:complexType>
  ...
</xsd:schema>
```



```
...  
<xsd:element name="transaction" type="tns:transactionType"/>  
  <xsd:element name="transactionResult" type="tns:transactionResultType"/>  
  <xsd:element name="accountNumber" type="xsd:int"/>  
  <xsd:element name="balance" type="xsd:int"/>  
</xsd:schema>
```



Izrada WSDL-a (1)



Zavod za telekomunikacije

Bank.wsdl

Source WSDL Partner

<http://j2ee.netbeans.org/wsdl/Bank>

- Types
 - <http://j2ee.netbeans.org/wsdl/Bank>
 - Attributes
 - Attribute Groups
 - Complex Types
 - Elements
 - Groups
 - Referenced Schemas
 - import {http://xml.netbeans.org/schema/Bank}
 - Simple Types
- Imports
- Messages
 - TransferRequest
 - transaction1 ns:transaction
 - TransferResponse
 - transactionReport1 ns:transactionResult
 - BankOperationRequest
 - number ns:accountNumber
 - BankOperationResponse
 - return ns:balance
- Port Types
 - BankPortType
 - Transfer
 - input1 Message="TransferRequest"
 - output1 Message="TransferResponse"
 - Balance
 - input2 Message="BankOperationRequest"
 - output2 Message="BankOperationResponse"

Bindings

- BankBinding PortType="BankPortType"
 - soap:binding
 - Transfer
 - soap:operation
 - input1
 - soap:body
 - output1
 - soap:body
 - Balance

Services

- BankService
 - BankPort Binding="BankBinding"
 - soap:address

Extensibility Elements

- Bank
 - BankPortTypeRole PortType="BankPortType"

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="Bank" targetNamespace="http://j2ee.netbeans.org/wsdl/Bank"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://j2ee.netbeans.org/wsdl/Bank"
  xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
  xmlns:ns="http://xml.netbeans.org/schema/Bank"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
  <types>
    <xsd:schema targetNamespace="http://j2ee.netbeans.org/wsdl/Bank"
      <xsd:import namespace="http://xml.netbeans.org/schema/Bank"
        schemaLocation="Bank.xsd"/>
    </xsd:schema>
  </types>
  <message name="TransferRequest">
    <part name="transaction1" element="ns:transaction"/>
  </message>
  <message name="TransferResponse">
    <part name="transactionReport1" element="ns:transactionResult"/>
  </message>
  <message name="BankOperationRequest">
    <part name="number" element="ns:accountNumber"/>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="Bank" targetNamespace="http://j2ee.netbeans.org/wsdl/Bank"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://j2ee.netbeans.org/wsdl/Bank"
  xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
  xmlns:ns="http://xml.netbeans.org/schema/Bank"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
  <types>
    <xsd:schema targetNamespace="http://j2ee.netbeans.org/wsdl/Bank">
      <xsd:import namespace="http://xml.netbeans.org/schema/Bank"
        schemaLocation="Bank.xsd"/>
    </xsd:schema>
  </types>
  <message name="TransferRequest">
    <part name="transaction1" element="ns:transaction"/>
  </message>
  <message name="TransferResponse">
    <part name="transactionReport1" element="ns:transactionResult"/>
  </message>
  <message name="BankOperationRequest">
    <part name="number" element="ns:accountNumber"/>
  </message>
</definitions>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="Bank" targetNamespace="http://j2ee.netbeans.org/wsdl/Bank"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://j2ee.netbeans.org/wsdl/Bank"
  xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
  xmlns:ns="http://xml.netbeans.org/schema/Bank"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
  <types>
    <xsd:schema targetNamespace="http://j2ee.netbeans.org/wsdl/Bank">
      <xsd:import namespace="http://xml.netbeans.org/schema/Bank"
        schemaLocation="Bank.xsd"/>
    </xsd:schema>
  </types>
  <message name="TransferRequest">
    <part name="transaction1" element="ns:transaction"/>
  </message>
  <message name="TransferResponse">
    <part name="transactionReport1" element="ns:transactionResult"/>
  </message>
  <message name="BankOperationRequest">
    <part name="number" element="ns:accountNumber"/>
```

```
<xsd:import namespace="http://xml.netbeans.org/schema/Bank"
              schemaLocation="Bank.xsd"/>
```

```
</xsd:schema>
```

```
</types>
```

```
<message name="TransferRequest">
```

```
  <part name="transaction1" element="ns:transaction"/>
```

```
</message>
```

```
<message name="TransferResponse">
```

```
  <part name="transactionReport1" element="ns:transactionResult"/>
```

```
</message>
```

```
<message name="BankOperationRequest">
```

```
  <part name="number" element="ns:accountNumber"/>
```

```
</message>
```

```
<message name="BankOperationResponse">
```

```
  <part name="return" element="ns:balance"/>
```

```
</message>
```

```
<portType name="BankPortType">
```

```
  <operation name="Transfer">
```

```
    <input name="input1" message="tns:TransferRequest"/>
```

```
    <output name="output1" message="tns:TransferResponse"/>
```

```
  </operation>
```

```
  <operation name="Balance">
```

```
    <input name="input2" message="tns:BankOperationRequest"/>
```

```
    <output name="output2" message="tns:BankOperationResponse"/>
```

```
  </operation>
```

```
</portType>
```

```
<binding name="BankBinding" type="tns:BankPortType">
```



```

</message>
<message name="BankOperationResponse">
  <part name="return" element="ns:balance"/>
</message>

<portType name="BankPortType">
  <operation name="Transfer">
    <input name="input1" message="tns:TransferRequest"/>
    <output name="output1" message="tns:TransferResponse"/>
  </operation>
  <operation name="Balance">
    <input name="input2" message="tns:BankOperationRequest"/>
    <output name="output2" message="tns:BankOperationResponse"/>
  </operation>
</portType>

<binding name="BankBinding" type="tns:BankPortType">
  <soap:binding style="document"
    transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="Transfer">
    <soap:operation/>
    <input name="input1">
      <soap:body use="literal"/>
    </input>
    <output name="output1">
      <soap:body use="literal"/>
    </output>
  </operation>

```

```

        <output name="output2" message="tns:BankOperationResponse"/>
    </operation>
</portType>

<binding name="BankBinding" type="tns:BankPortType">
    <soap:binding style="document"
        transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="Transfer">
        <soap:operation/>
        <input name="input1">
            <soap:body use="literal"/>
        </input>
        <output name="output1">
            <soap:body use="literal"/>
        </output>
    </operation>
    <operation name="Balance">
        <soap:operation/>
        <input name="input2">
            <soap:body use="literal"/>
        </input>
        <output name="output2">
            <soap:body use="literal"/>
        </output>
    </operation>
</binding>

<service name="BankService">

```

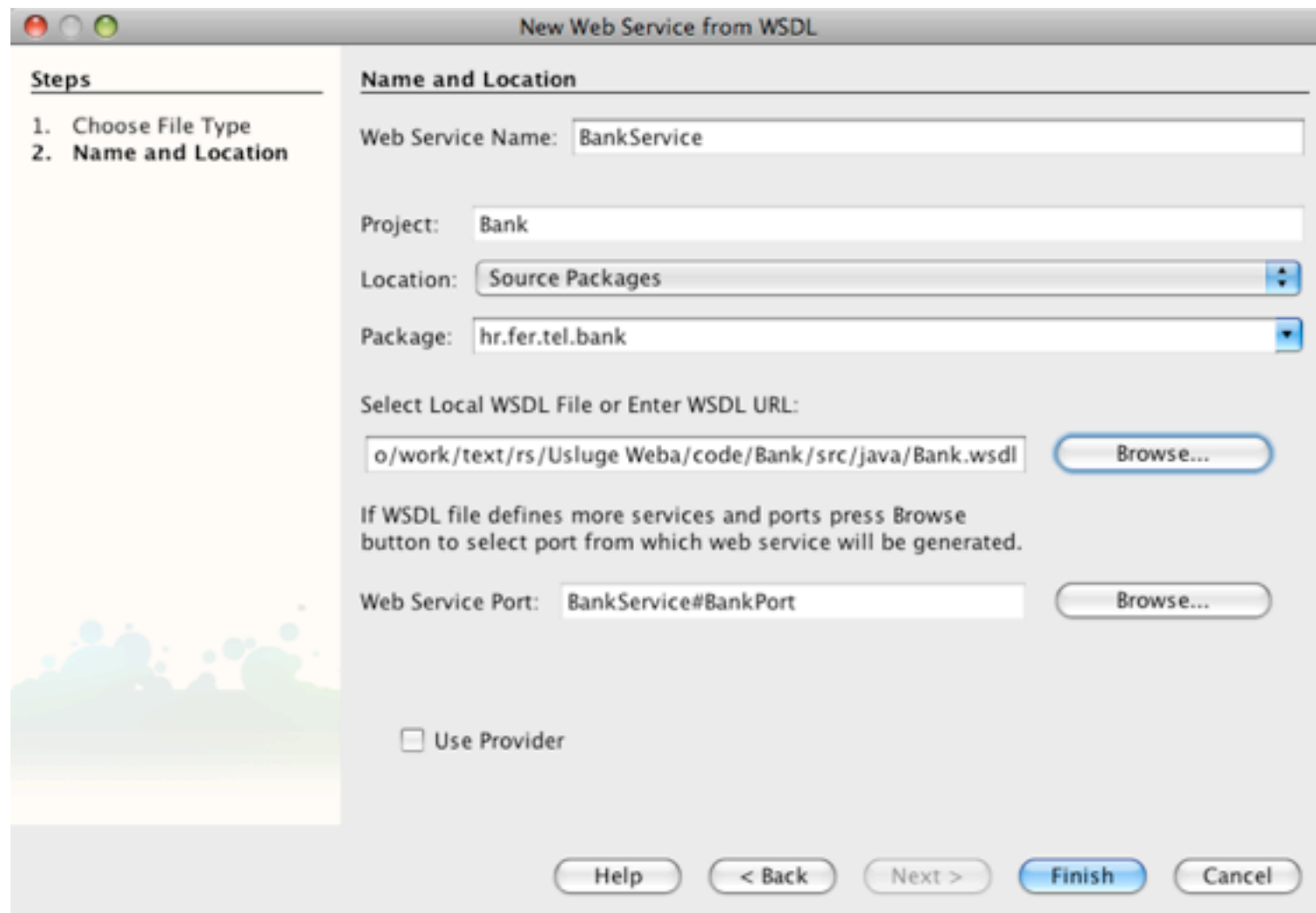
```
</input>
    <output name="output2">
        <soap:body use="literal"/>
    </output>
</operation>
</binding>

<service name="BankService">
    <port name="BankPort" binding="tns:BankBinding">
        <soap:address location="http://localhost:${HttpDefaultPort}/
BankService/BankPort"/>
    </port>
</service>

<plnk:partnerLinkType name="Bank">
    <plnk:role name="BankPortTypeRole" portType="tns:BankPortType"/>
</plnk:partnerLinkType>
</definitions>
```

```
</input>
    <output name="output2">
        <soap:body use="literal"/>
    </output>
</operation>
</binding>
<service name="BankService">
    <port name="BankPort" binding="tns:BankBinding">
        <soap:address location="http://localhost:${HttpDefaultPort}/
BankService/BankPort"/>
    </port>
</service>
<plnk:partnerLinkType name="Bank">
    <plnk:role name="BankPortTypeRole" portType="tns:BankPortType"/>
</plnk:partnerLinkType>
</definitions>
```

- ◆ File → New File → Web Services → Web Service From WSDL



Steps

1. Choose File Type
2. Name and Location

Name and Location

Web Service Name:

Project:

Location:

Package:

Select Local WSDL File or Enter WSDL URL:

If WSDL file defines more services and ports press Browse button to select port from which web service will be generated.

Web Service Port:

☐ Use Provider

Stvaranje usluge (2)



Zavod za telekomunikacije

BankService.java

Source Design 100%

BankService [BankPort]

Operations (2) Add Operation... Remove Operation

Transfer

Parameters	Output	Faults	Description
Parameter Name	Parameter Type		
transaction	org.netbeans.xml.schema.bank.TransactionType		

Balance

Parameters	Output	Faults	Description
Parameter Name	Parameter Type		
accountNumber	int		

Quality Of Service

- ☐ Optimize Transfer Of Binary Data (MTOM)
- ☐ Reliable Message Delivery
- ☐ Secure Service

Advanced ...

Stvaranje usluge (3)



Zavod za telekomunikacije

```
package hr.fer.tel.bank;

import ...

@WebService(serviceName = "BankService", portName = "BankPort",
    endpointInterface = "org.netbeans.j2ee.wsdl.bank.BankPortType",
    targetNamespace = "http://j2ee.netbeans.org/wsdl/Bank",
    wsdlLocation = "WEB-INF/wsdl/BankService/Bank.wsdl")

public class BankService implements BankPortType {

    public TransactionResultType transfer(TransactionType transaction1) {
        //TODO implement this method
        throw new UnsupportedOperationException("Not implemented yet.");
    }

    public int balance(int number) {
        //TODO implement this method
        throw new UnsupportedOperationException("Not implemented yet.");
    }
}
```

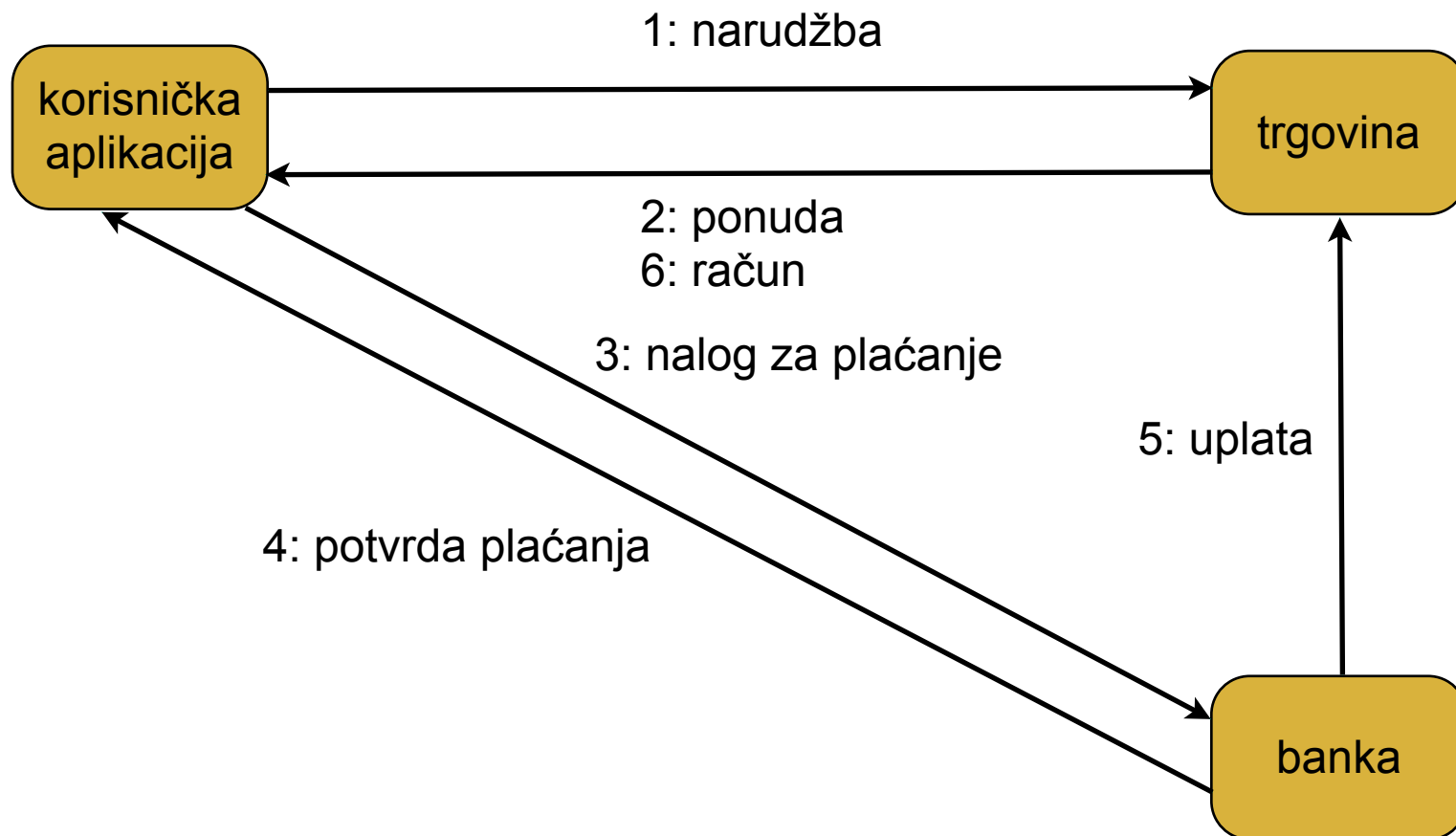
```
public TransactionResultType transfer(TransactionType transaction1) {  
    TransactionResultType result = new TransactionResultType();  
  
    if(transaction1.getAmount() > 1000) {  
        result.setOK(false);  
        result.setError("The amount is out of limits.");  
    } else {  
        result.setOK(true);  
    }  
  
    return result;  
}
```

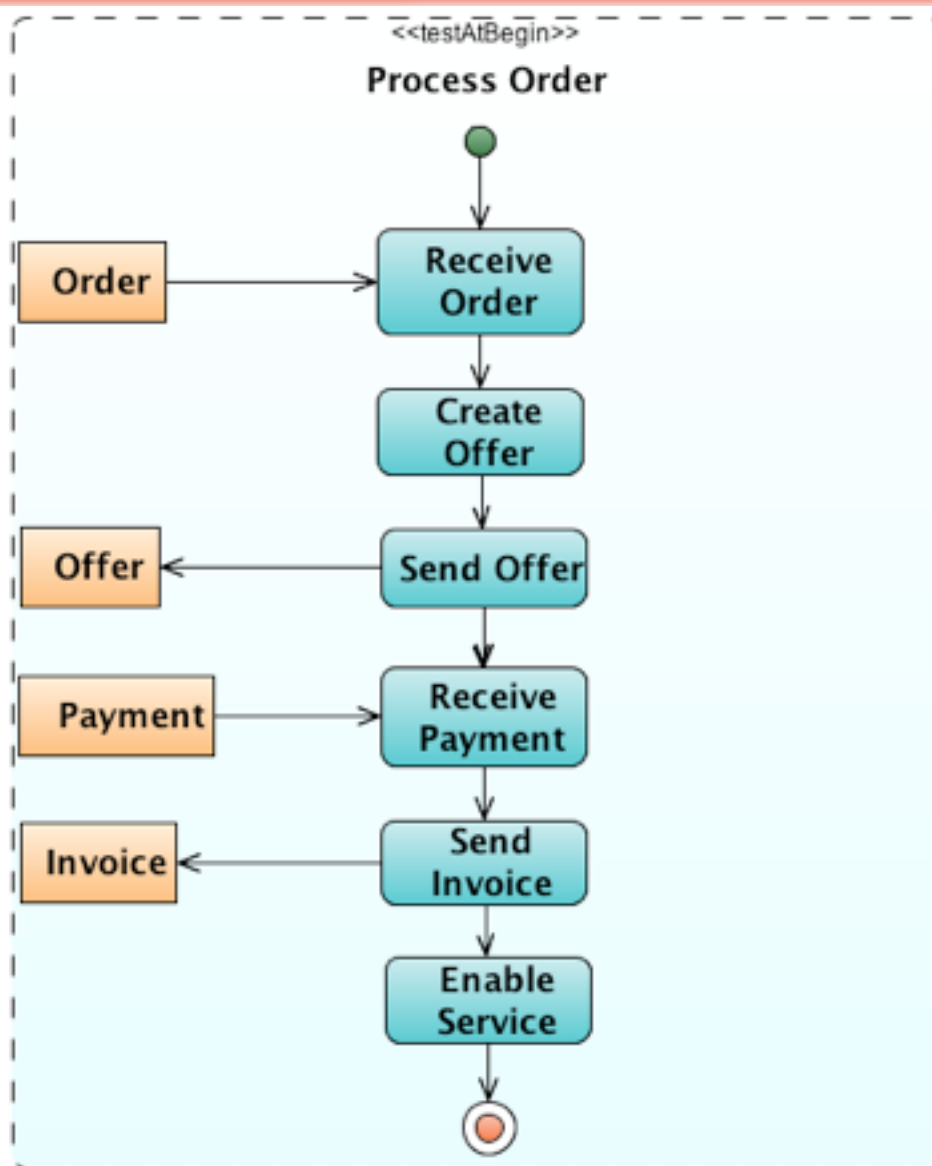

- ◆ isto kao i prijašnje usluge
- ◆ kod:

```
try { // Call Web Service Operation
    BankService service = new BankService();
    BankPortType port = service.getBankPort();

    TransactionType transaction1 = new TransactionType();
    transaction1.setAmount(500);

    TransactionResultType result = port.transfer(transaction1);
    if(result.isOK())
        System.out.println("Transaction is successful.");
    else
        System.out.println("Transaction failed. Reason: " +
            result.getError());
} catch (Exception ex) {
    // TODO handle custom exceptions here
}
```





- ◆ SOA (*Service Oriented Architecture*)
- ◆ to je pristup/način izrade usluga
- ◆ razdvajanje interesa (*separation of concerns*)
- ◆ raspodijeljene usluge (fizički, pružatelj)
- ◆ svaka usluga se može nadograđivati bez obzira na druge usluge
- ◆ jedna usluga koristi drugu i mora znati kako ju koristiti (lokacija, opis, poruke)
- ◆ SOA nije nužno vezana uz usluge weba
 - ▶ najlakše napraviti pomoću usluga weba

- ◆ slaba povezanost usluga
- ◆ uslužni ugovori
 - dogovor o komunikaciji (opis usluge – WSDL, XML Schema, *policy*, pravni dokumenti)
- ◆ autonomnost
 - usluga ima kontrolu nad logikom koju enkapsulira
- ◆ apstrakcija
 - izvana se vidi samo ono što je u ugovoru
- ◆ ponovna iskoristivost
 - uslugu mogu koristiti druge usluge
- ◆ mogućnost slaganja u složene usluge
- ◆ usluge bez stanja su skalabilne
- ◆ mogućnost otkrivanja usluga

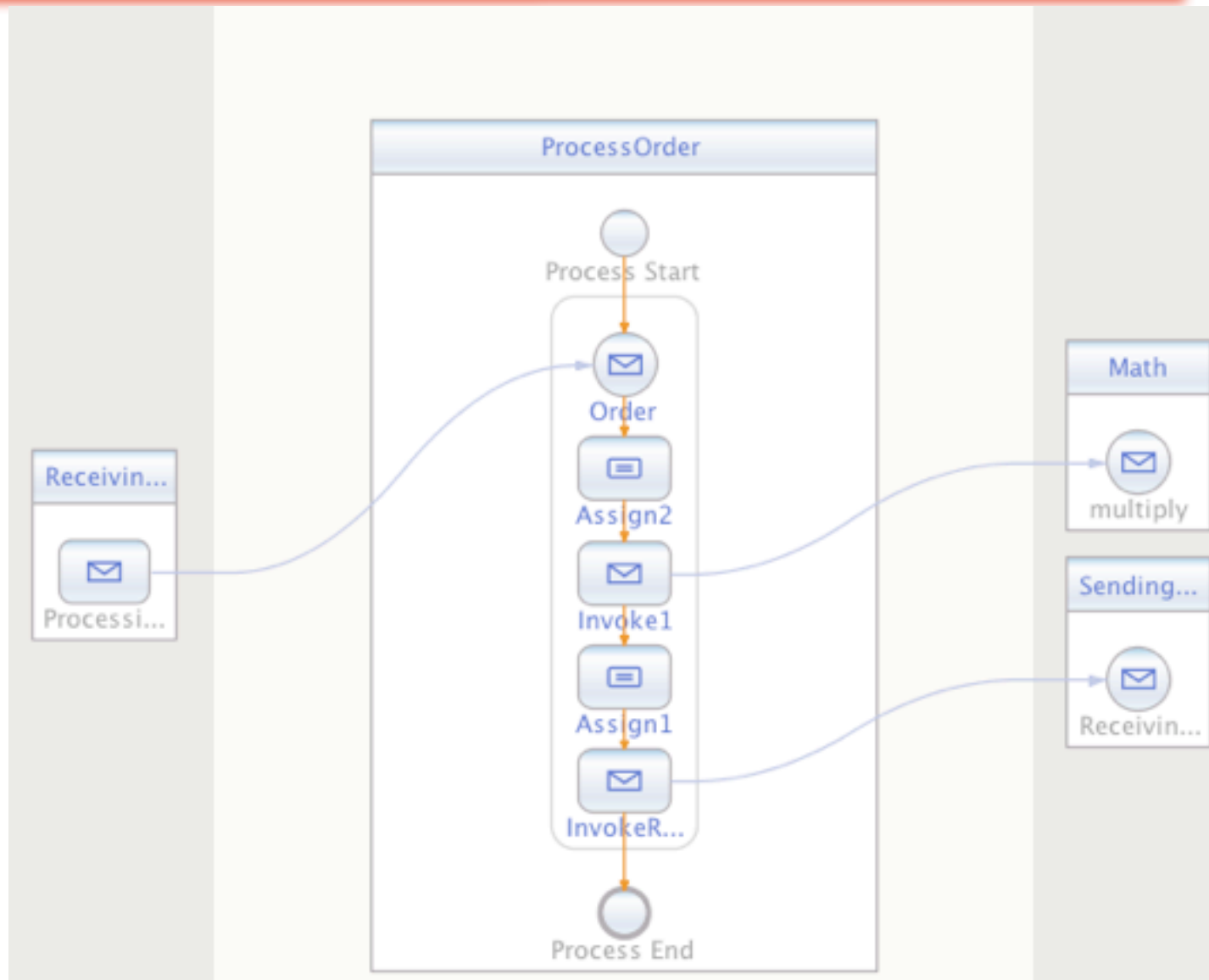
- ◆ *Business Process Execution Language* (BPEL)
- ◆ opis poslovnog procesa pomoću XML-a
- ◆ interakcija s uslugama weba
- ◆ nije metodologija ili proces
- ◆ definira ponašanje jednog entiteta
- ◆ omogućuje jednostavnu manipulaciju podacima
- ◆ ima elemente programskog jezika
 - petlje, uvjete, varijable, pozivanje usluga, čekanje poziva usluge, ...
- ◆ postoje “procesori” koji mogu izvršavati BPEL procese
 - npr. *Java Business Integration (JBI) runtime environment*

- ◆ stvaranje BPEL modul projekta
 - stvaranje XML Scheme
 - stvaranje WSDL-a
 - stvaranje BPEL procesa: dodavanje partnerskih linkova, povezivanje usluga, mapiranje podataka
- ◆ izrada kompozitne aplikacije
 - dodavanje BPEL modula aplikaciji
- ◆ isporuka kompozitne aplikacije

Primjer BPEL procesa (1)



Zavod za telekomunikacije



- ◆ REST (*Representational State Transfer*)
- ◆ pojmovi: *The REST Way* ili *RESTful services*
- ◆ pojam je skovao Roy Fielding u svojoj doktorskoj disertaciji
- ◆ nije standard već arhitekturni stil
- ◆ sve se temelji na resursima koji su predstavljeni URL-ovima:
 - <http://tel.fer.hr/users/> - popis svih korisnika
 - <http://tel.fer.hr/users/1> - korisnik s identifikatorom 1
 - <http://tel.fer.hr/users/1/firstName> - ime korisnika
- ◆ koristi protokole: HTTP (GET, POST, PUT, DELETE), XML, XLink
- ◆ bez stanja (*stateless*), privremeno spremanje (*cache*)

- HTTP GET <http://tel.fer.hr/users>

```
<?xml version="1.0"?>
<t:Users xmlns:t="http://tel.fer.hr"
          xmlns:xlink="http://www.w3.org/1999/xlink">
  <User xlink:href="http://tel.fer.hr/users/1">Mario Kušek</User>
  <User xlink:href="http://tel.fer.hr/users/2">Maja Matijašević
</User>
  ...
</t:Users>
```

- HTTP GET <http://tel.fer.hr/users/2>

```
<?xml version="1.0"?>
<t:User xmlns:t="http://tel.fer.hr"
        xmlns:xlink="http://www.w3.org/1999/xlink">
  <FirstName>Maja</FirstName/>
  <LastName>Matijašević</LastName/>
  ...
</t:User>
```

- ♦ za *Create, read, update and delete* (CRUD)

HTTP	CRUD
PUT	Create, (Overwrite/Replace)
GET	Read
POST	Update, (Create, Delete)
DELETE	Delete

- ♦ primjer gotove usluge: *Flickr Services*