

ՅՈՒՐԱԿԱՆ ԵՎ ԲԵՐՈՒՄԱՆ ՎԵՐՈՒՄ

ԶԵՆՈՒԹՅԱՆ ԶԵՆՈՒԹՅԱՆ ԿԵՆՏՐԱԼԻԶԱՄ

ՈՒՆԱՅԻՆ ՄԻՋՈՒԹՅԱՆ

Otvoreno računarstvo

- Uslužno usmjerena arhitektura
 - Osnovni pojmovi
 - Sprega
 - Primjeri:
 - Eclipse
 - XML-RPC
 - SOA - SOAP

Mario Žagar





Osnovni pojmovi

Uslužno usmjerena arhitektura

- Arhitektura usmjerena prema uslugama
- "Vruća" tema posljednjih godina
- "Moda" u svijetu programskog inženjerstva?
- Odgovor na potrebe i probleme razvoja u svijetu poslovne programske podrške?
- *Budućnost* aplikacija?
 - Google Docs (*Office* preko Interneta)
 - Microsoft Office Live
 - Adobe Photoshop (on-line)
 - Amazon Cloud (virtualni poslužitelj na Internetu)



Što je SOA?



Service-Oriented Architecture (SOA) is an architectural style that supports service orientation.

An architectural style is the combination of distinctive features in which architecture is performed or expressed.

Service orientation is a way of thinking in terms of services and service-based development and the outcomes of services.

The Open Group, <http://opengroup.org>

Što je 'usluga' (Service)?

A service:

- *Is a **logical representation** of a **repeatable** business **activity** that has a specified outcome (e.g., check customer credit; provide weather data, consolidate drilling reports)*
- *Is **self-contained***
- *May be **composed** of other services*
- *Is a “**black box**” to consumers of the service*

Definition of SOA, The Open Group, <http://opengroup.org>

I što je onda SOA?

- Arhitekturni **stil** za razvoj aplikacija
- Orijehtiran na **poslovne** primjene (ne i ograničen)
 - Velike tvrtke
 - Složene sustave
 - Složene poslovne procese
 - Velike količine podataka
 - Veliki troškovi izgradnje i održavanja
- Uz sve ovo
 - Potreba za **komunikacijom** sa drugim tvrtkama
 - Velikim
 - Složenim
 - ...



Usluga?



- Logička cjelina
- Jasno definirane granice
 - Znamo kako postaviti zahtjev za uslugom
 - Znamo kako predati potrebne informacije
 - Znamo što možemo očekivati
 - Definirano sučelje
- Dosljedna
 - Daje predvidljive rezultate
- Skriva detalje funkcioniranja
 - Trebamo poznavati sučelje



S druge točke gledišta



*Services are anything sold in trade that cannot be
dropped on your foot*

The Economist

Još jedna...



An *architectural style* for building *distributed* systems that deliver application functionality as *services* to either user *applications* or other *services*

Service-Oriented Architecture expands the vision of Web services, IBM

- Naglasak na raspodijeljene sustave
- Aplikacija kao skup usluga
- Usluga kao skup usluga



Konačna definicija?

- Jedinstvena definicija ne postoji
- Zašto?
 - Arhitekturni stil je teško definirati
 - Arhitekturni stil je teško (izravno) primijeniti
 - Potrebno je prilagoditi i uskladiti sa svojim potrebama
- Svaka prilagodba mijenja definiciju
- Isto je sa svim paradigmama razvoja aplikacija
 - Proceduralna
 - OO
 - ...

Budimo praktični

- Problem:


Potrebno je dizajnirati (složenu) aplikaciju

- Kako organizirati strukturu aplikacije?
- Kako povezati/odvojiti dijelove aplikacije?
- Kako iskoristiti postojeće dijelove?
- Kako omogućiti ponovno korištenje gotovih dijelova?

- Dodatni problem:

Potrebno je povezati više postojećih aplikacija

- Koju metodu komunikacije upotrijebiti?
- Kako opisati sučelja?
- Kako uskladiti podatke?

The background features large, light blue stylized letters 'A' and 'R' that are partially visible, with the 'A' on the left and the 'R' on the right. The text 'Sprega (Coupling)' is centered in the middle of the image.

Sprega (*Coupling*)

Sprega

- Sprega (*coupling*) određuje razinu interakcije dijelova cjeline
 - Niska razina
 - Poznavanje interne organizacije
 - Pristupanje *tuđim* podacima
 - Visoka razina
 - Poznavanje javnog sučelja
 - Pristupanje javnim podacima

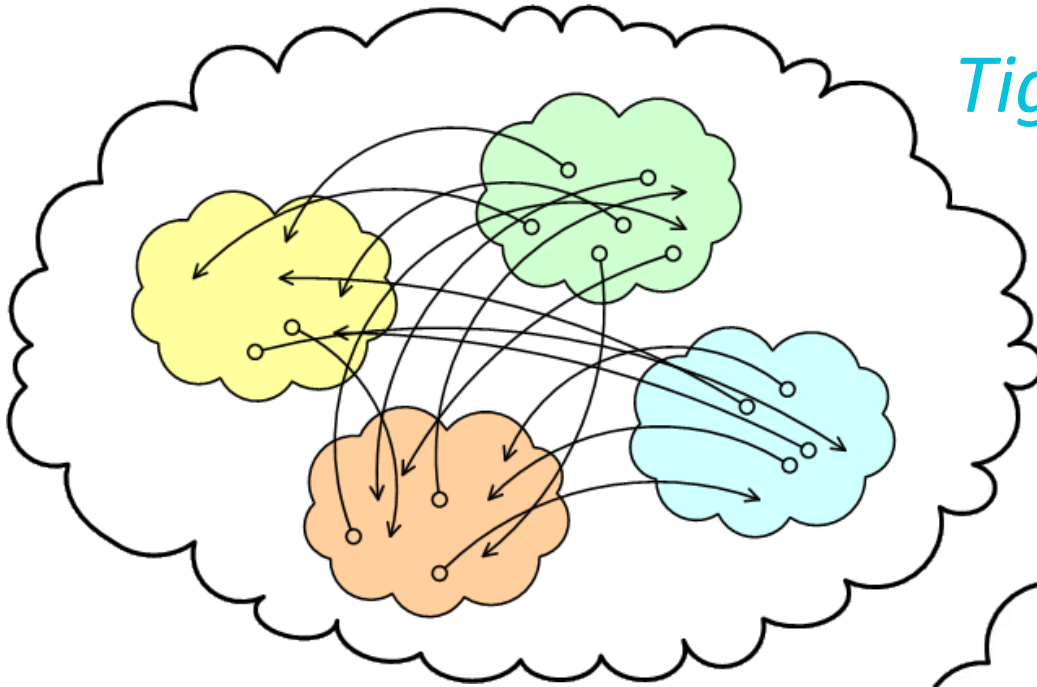
Čvrsta sprega

Labava sprega

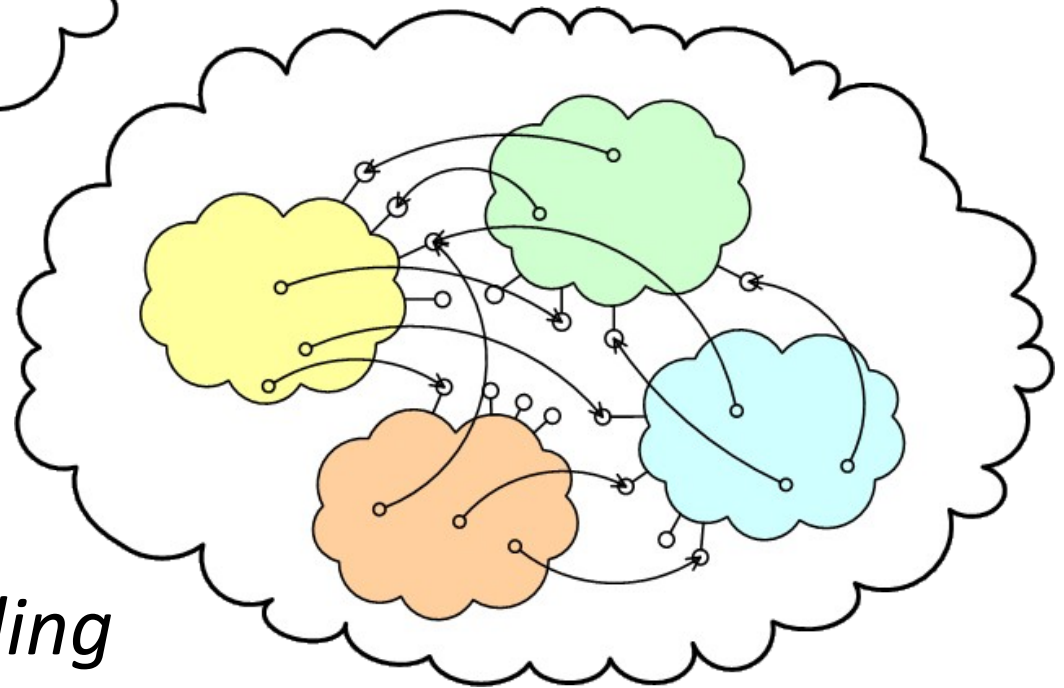
Čvrsta sprega - labava sprega



Tight coupling



Loose coupling



Čvrsta sprega



- Nastaje **prirodno**, bez posebnog truda
 - Raznorodni dijelovi aplikacije međusobno koriste funkcionalnosti
- Teško održavanje
 - Interne promjene bilo kojeg dijela aplikacije imaju široke posljedice
- Spore i skupe promjene
 - Potrebno mijenjati na više mjesta
- Višestruko složenije i skuplje u poslovnom svijetu
 - Promjena unutarnjeg dijela aplikacije tvrtke A zahtijeva promjene u aplikaciji tvrtke B

Labava sprega

- Zahtijeva **trud** od faze dizajna na dalje
- Lakše održavanje
 - Možemo zamijeniti samo dio aplikacije
- Jednostavnije i jeftinije promjene
 - Jasna funkcionalnost pojedinih dijelova
 - Jasne granice između pojedinih dijelova
 - Jasne ovisnosti između dijelova
- Ponovno korištenje
 - Iz jedne aplikacije uzmemo dio i ugradimo u drugu aplikaciju

* sprega

- Špageti-kôd
- Neprovojeravanje rubnih slučajevea
- Loše programerske navike
- Loš dizajn
- *Krpanje*
- Korištenje (umetni) mimo specifikacije

... **nemaju** veze s (čvrstom)
spregom unutar aplikacije
i jednostavno su **loši**.



Primjer

- Slična nastojanja u povijesti
- Biblioteke (*library*)
 - Niz funkcija i struktura podataka
 - Zaglavne (*header*) datoteke - što **smijemo** koristiti
 - Podaci?
- Objektno orijentirana paradigma
 - Enkapsulacija u objekte
 - **Prava pristupa** metodama i podacima (*public, private,...*)
 - Javni i privatni *život* objekata
- Ovo je sve na niskoj razini

Sprega i arhitektura

- Cilj: labava sprega unutar i između aplikacija

Aplikacija = skup labavo povezanih dijelova/aplikacija

- Uslužno usmjerena arhitektura vidi aplikaciju kao niz **povezanih** usluga



Pokušaji



- Složeni dokumenti
 - Više aplikacija izgrađuje isti dokument (*compound document, object embedding*)
 - OLE, COM, OpenDoc
- Posredničke tehnologije
 - Povezuju (raspodijeljene) **objekte**
 - CORBA, COM+, DCOM, RMI

} **Objekt** - manja funkcionalnost
- Arhitekture orijentirane na usluge
 - Povezuju (raspodijeljene) **usluge**
 - Web Servisi, SOA, OSGi

} **Usluga** - veća funkcionalnost

Razine sprege

- Dijelovi aplikacije

- Funkcije
- Objekti
- Usluge

Čvrsta sprega

Labava sprega

- Podaci

- Vlastiti oblik zapisa
- SQL (Structured Query language)
- XML

Čvrsta sprega

Labava sprega

Izgradnja temeljena na uslugama



- Postupci povezivanja (i povezivanja s) usluga(ma)
 - Otkrivanje usluge
 - Opis sučelja usluge
 - Opis podataka sučelja
 - Pristup usluzi
 - Kompozicija usluga
- Trebaju biti široko prihvaćeni
 - Želimo omogućiti drugima da koriste naše usluge i obratno, da možemo
 - Kupiti gotovo rješenje
 - Koristiti usluge koje ne možemo sami ostvariti
 - Kreditne kartice
 - Slanje SMS-a

Otkrivanje usluge

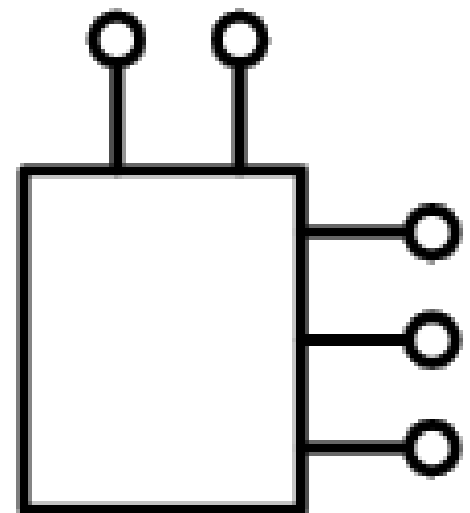


- Upisnik (registar, *registry*) usluga
 - Lokalni, centralni, distribuirani
- Adresa usluge
 - Poznata ili dolazi od upisnika
 - Dovoljna za **jedinstveno** raspoznavanje usluge



Opis sučelja

- Čest problem
- Postojeći jezik
 - Java RMI: Java (koristi se *interface*)
- Vlastiti jezik
 - CORBA: IDL (Interface Definition Language)
 - DCOM: MIDL (Microsoft Interface Definition Language)
- Dijalekt jezika
 - XML



Opis podataka sučelja

- Čest problem
- Postojeći jezik
 - Java RMI: Java (ugrađeni i složeni tipovi podataka)
- Vlastiti jezik
 - CORBA: IDL
 - DCOM: MIDL
- Dijalekt jezika
 - XML: XML Schema Definition Language (XSDL)
 - Dopunjuje praznine DTD-a
 - Opisuje strukturu podataka sadržanih kao vrijednost atributa i elemenata
 - Opisuje strukturu podataka sadržanih u više elemenata

Pristup usluzi

- Protokol
- Lokalna aplikacija
 - Bilo što (poziv metode)
- Raspodijeljena aplikacija
 - Mrežni protokol
 - Problem propuštanja vatrozida
 - Koristiti HTTP (prolazi svuda)
 - Vlastiti protokol na poznatom portu (port 80, npr. Skype)
 - Problem: široka prihvaćenost

Kompozicija usluga

- **Aplikacija** = niz povezanih usluga (kompozicija)
- **Usluga** = crna kutija
 - Može nastati korištenjem povezivanjem drugih usluga
- Isti mehanizam korištenja usluge za oba slučaja
- *"Ručno"*
 - Čovjek odabire koje će usluge i kojim redom upotrijebiti
- *Automatski*
 - Računalo odabire i koristi usluge prema potrebi i opisu posla

XML Schema Definition Language



- XSDL
 - Jezik za opis strukture XML dokumenta
 - Proširenje funkcije DTD-a
- XML shema (XSD)
 - Dokument pisan XSDL-om
 - Opisuje elemente i attribute koji se mogu pojaviti u XML dokumentu
 - Opisuje ograničenja sadržaja elemenata i atributa
 - Opisuje tipove podataka koji se mogu pojaviti u XML dokumentu

XSD dokument

```
<?xml version="1.0" encoding="UTF-8"?>
<schema ... >
  <simpleType name="VrsteGrupa">
    <restriction base="string">
      <enumeration value="Prijatelji"/>
      <enumeration value="Obitelj"/>
    </restriction>
  </simpleType>
  <element name="osoba">
    <complexType>
      <sequence>
        <element name="ime" type="string"/>
        <element name="prezime" type="string"
          minOccurs="1" maxOccurs="2" />
        <element name="dob" type="positiveInteger"/>
      </sequence>
      <attribute name="grupa" type="VrsteGrupa"
        use="required"/>
    </complexType>
  </element>
</schema>
```

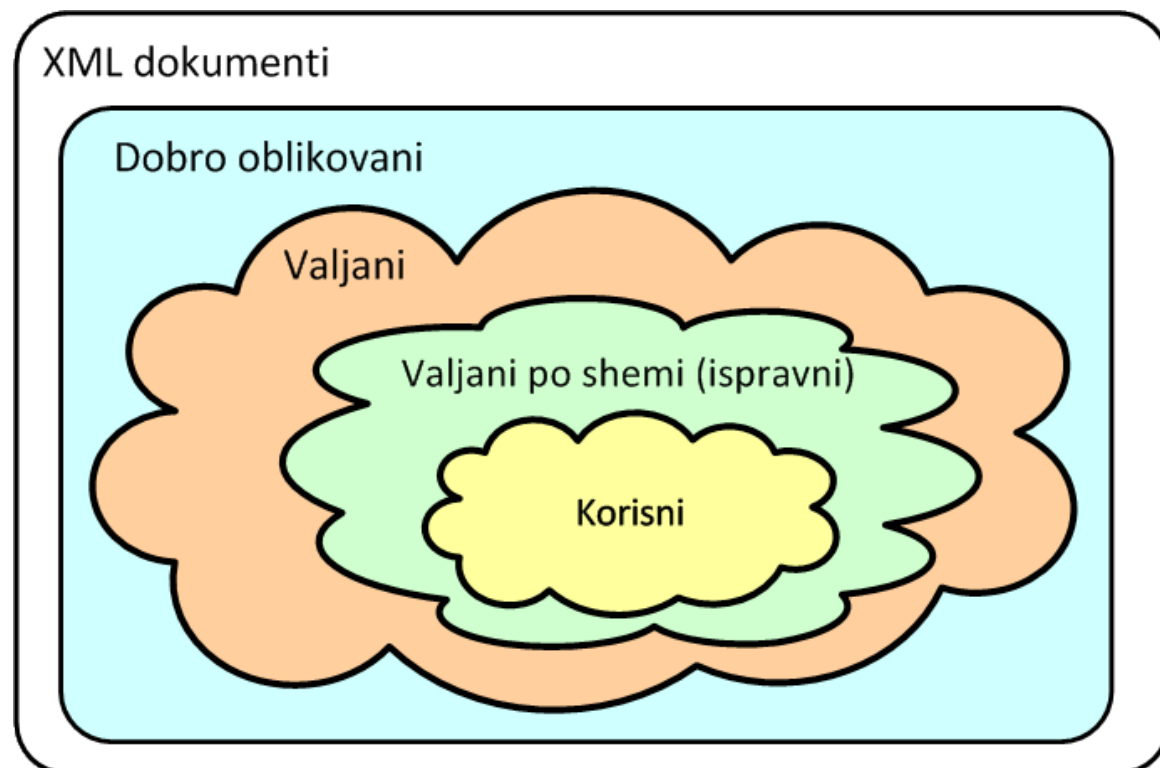
Osoba sa imenom, do 2 prezimena i dobi kao pozitivnim brojem, svrstana u grupu Prijatelji ili Obitelj

Jednostavni tipovi podataka

- Neki od tipova podataka definiranih u okviru XSDL-a
 - boolean
 - float
 - decimal
 - integer
 - nonPositiveInteger
 - nonNegativeInteger
 - string
 - time
- Moguće je definirati vlastite tipove podataka
 - Kao ograničenja postojećih tipova (npr. interval $\langle a, b \rangle$)
 - Kao složeni tip podataka

Provjera ispravnosti(Sukladnosti???)

- XML dokument može biti
 - Ispravno oblikovan (*well-formed*)
 - Valjan (*valid*)
 - Valjani po shemi - ispravni (*schema-valid*)
 - Koristan





Eclipse

Primjena arh. temeljene na uslugama



- Arhitekturni **stil** - može se primijeniti **svugdje**
- Najčešća primjena
 - Poslovna okolina (SOA)
 - **Složeni** sustavi
 - Potreba za izgradnjom sličnih aplikacija (**ponovno korištenje**)
 - Brz odgovor na **promjene**
 - Web (Web Services, WS)
 - **Komunikacija** sa drugima
- Što je sa *klasičnim* aplikacijama?

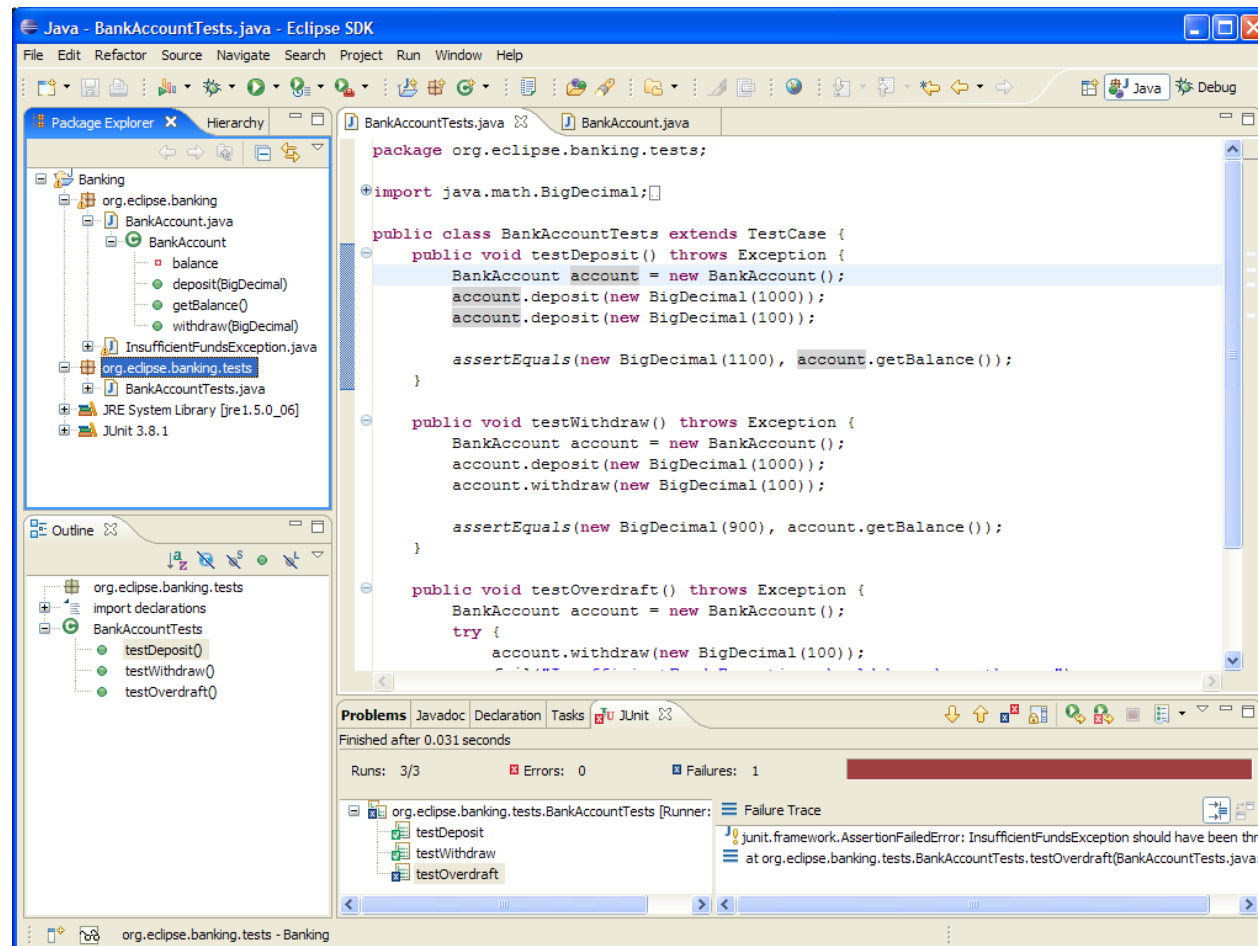
Primjer: Eclipse



Eclipse

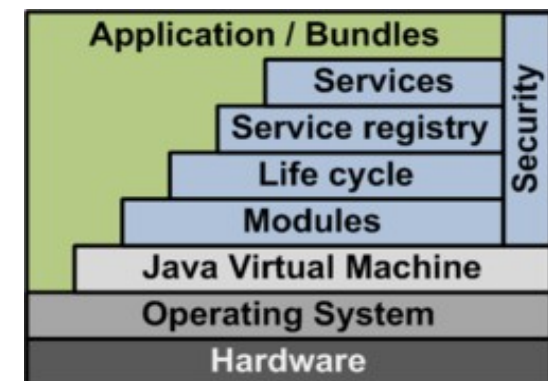


- Platforma za razvoj aplikacija
- Eclipse IDE - najpoznatija aplikacija na platformi



Eclipse - platforma

- U srcu platforme - Equinox
- Equinox - implementacija OSGi specifikacije
- OSGi - Open Services Gateway initiative
 - stari naziv, ali ostala skraćenica - danas OSGi Forum
<http://www.osgi.org>
- OSGi
 - *Service oriented lightweight component framework*
 - Aplikacija se sastoji od svežnja (*bundle*)
 - Svaki prilog donosi djelić funkcionalnosti aplikaciji



Eclipse - aplikacija

- Aplikacija
 - skup modula/cjelina koje se mogu dodavati/uključivati prema potrebi (*plug in*)
 - modul (u ovom kontekstu)
 - OSGi svežanj (*bundle*)
- Aplikacija se gradi
 - od niza dodataka
 - konfiguracije koja opisuje koje dodatke pokrenuti
 - dodaci su **ravnopravni**
- Dodatak - npr.
 - korisničko sučelje
 - Statusna traka, alatna traka, pogled, urednik teksta, ...
 - graditelj projekta (*builder*)
 - Java, C++, Perl, ...

Labava sprega

- Prilog (bundle)
 - Opisuje pakete (Java) koje treba (uvozi)
 - Opisuje pakete (Java) koje objavljuje (izvozi)
 - Opisuje koje druge priloge zahtijeva za svoj ispravan rad
 - Npr. tamo su definirani paketi (Java) koji su potrebni za rad

Import-Package: org.osgi.framework;version=1.3

Export-Package: hr.fer.rasip.or.sucelje

Require-Bundle: org.eclipse.equinox;
bundle-version="[3.3.0;4.0.0) "

Prednosti

- Prilozi se
 - Dinamički učitavaju kad su potrebni (*lazy loading*)
 - Pokreću kad trebaju nešto izvesti i zaustavljaju kad više nisu potrebni (*start/stop*)
- Prilozi mogu
 - Objaviti usluge koje nude drugima
 - Pratiti dostupnost pojedine usluge (dinamičko okružno)
- Prilozi se mogu za vrijeme rada zamijeniti novom verzijom
 - Labava sprega!

Proširivost

- Želimo omogućiti drugima da dodaju funkcionalnost našoj aplikaciji
 - Stvoriti mehanizam za dodatke (npr. Photoshop, ...)
 - Ugraditi skriptni/makro jezik (npr. VB for Applications)
 - ...

- Kako to radi Eclipse
 - Dodatak D1 definira točku proširenja
 - Dodatak D2 implementira proširenje za točku proširenja
 - **Doprinos** ukupnoj funkcionalnosti (*contribution*)
 - Sve ovo definirano u XML-u i XSD-u

U cjelini

- **Proširivost na više razina**
 - Dodatak objavljuje programsko sučelje
 - Dodatak definira točke proširenja
- **Dinamičnost**
 - Dodaci se učitavaju, pokreću, isključuju prema potrebi
 - Usluge se pojavljuju (objavljaju) i nestaju
 - Dijelovi aplikacije se mogu zamijeniti novim bez gašenja
- **Bez svega ovog**
 - Eclipse ne bi bio upotrebljiv
 - Eclipse ne bi stao u memoriju
 - Cjelokupna funkcionalnost projekta ne bi bila moguća

Primjer: Web Servisi

- Web Servisi (*Web Services*, WS) - pristup udaljenim uslugama
 - Adresiranje preko URI-ja
 - Poruke u obliku XML-a
 - (Korištenje HTTP-a kao transportnog protokola)
- Više inačica
 - XML-RPC - poziv udaljenih metoda
 - SOA - pristup udaljenim uslugama (protokol SOAP)
 - Usluge zasnovane na REST principu
 - ...?



XML-RPC

Primjer: XML-RPC

- Protokol poziva udaljenih procedura (RPC)
 - Koristi XML za kodiranje poziva procedure
 - Koristi HTTP kao prijenosni protokol
- Jednostavan protokol
 - Definira jednostavne tipove podataka
 - Jednostavni mehanizmi poziva, odgovora i grješke
- Pozivanje procedure
 - Slanje zahtjeva HTTP-om sa XML-om u sadržaju zahtjeva
 - Procedura odgovara XML-om

Tipovi podataka

- Opisani elementima XML-a
- Podržani osnovni tipovi
 - int - 32 bita s predznakom
 - string - ASCII string (neke implementacije i Unicode)
 - boolean
 - double
 - dateTime.iso8601 - datum u obliku YYYY-MM-DD bez TZ
 - base64
 - array
 - struct

XML-RPC klijent (PHP)

```
// Procedure su na adresi http://www.fer.hr/procedure.php
$server = new xmlrpc_client('procedure.php',
                           'www.fer.hr', 80);

// Priprema zahtjeva - or.udaljeniKvadrat(42)
$poruka = new xmlrpcmsg('or.udaljeniKvadrat',
                        array(new xmlrpcval(42, 'int')));
$rez = $server->send($message); // Pošalji poruku

// Provjera grješke i ispis rezultata
if (!$rez) {
    print "Grješka u spajanju.";
} elseif ($result->faultCode()) {
    print "XML-RPC grješka #" . $rez->faultCode() . ": " .
    $rez->faultString();
} else {
    print("Kvadrat: " . $rez->value());
}
```

XML-RPC klijent (PHP)

- U slučaju korištenja drugog API-ja još lakše

```
list($uspjeh, $rez) =
    xmlrpc_request('www.fer.hr',
                  '/procedure.php',
                  'or.udaljeniKvadrat',
                  array(xmlrpc_prepare(42)));
```

- Zahtjev sadrži
 - Ime procedure koju treba pozvati
 - Parametre koji se šalju
- Odgovor sadrži
 - Rezultat ili
 - Kôd i opis grješke

Zahtjev XML-RPC-a

- Transportni protokol je HTTP
- Zahtjev se prenosi u obliku XML poruke u tijelu HTTP zahtjeva

POST /procedure.php HTTP/1.0

Host: www.fer.hr

Connection: close

Content-Type: text/xml

Content-Length: ...

```
<?xml version="1.0" ?>
<methodCall>
<methodName>or.udaljeniKvadrat</methodName>
<params>
  <param>
    <value>
      <int>42</int>
    </value>
  </param>
</params>
</methodCall>
```

Odgovor XML-RPC-a



- Isti mehanizam kao zahtjev

HTTP/1.0 200 OK

Connection: close

Content-Type: text/xml

Content-Length: ...

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value>
        <int>1764</int>
      </value>
    </param>
  </params>
</methodResponse>
```

Grješka XML-RPC-a

HTTP/1.0 200 OK

Connection: close

Content-Type: text/xml

Content-Length: ...

```
<?xml version="1.0"?>
<methodResponse>
<fault>
  <value>
    <struct>
      <member>
        <name>faultCode</name>
        <value><int>101</int></value>
      </member>
      <member>
        <name>faultString</name>
        <value><string>Broj je premalen.</string></value>
      </member>
    </struct>
  </value>
</fault>
</methodResponse>
```



XML-RPC poslužitelj (PHP)

```
$zahtjev = xmlrpc_parse($HTTP_RAW_POST_DATA);    // Zahtjev
$imeMetode = xmlrpc_getMethodName($zahtjev);    // Metoda
$params = xmlrpc_getParams($zahtjev);           // Parametri


if($imeMetode=='or.udaljeniKvadrat'){           // Što pozvati?
    or_kvadrat($params);
} else {
    xmlrpc_error("100", "Nepostojeća metoda");    // Grješka
}

function or_kvadrat($params)
{
    $broj = $params[0];
    $kvadrat = $broj * $broj;

    if($broj > 1){
        xmlrpc_response(xmlrpc_prepare($kvadrat)); // Odgovor
    }else{
        xmlrpc_error("101", "Broj je premalen."); // Grješka
    }
}
```

XML-RPC

- Jednostavan (prednost)
- Pozivi metoda, ne usluga (niska razina)
- Veća sprega (u odnosu na ostale primjere)
- Otkrivanje usluge - nepostojeće
- Opis sučelja - nepostojeći, treba znati tip parametra
- Opis podataka sučelja - XML (bez prostora imena)
- Pristup usluzi - HTTP
- Kompozicija usluga - ručno, u programu



SOA - SOAP

Primjer: SOA

- Na višoj razini u odnosu na XML-RPC
 - arhitektura, ne samo protokol
- Komunikacija porukama SOAP-a (*Simple Object Access Protocol*)
- Opis sučelja usluge - WSDL (*Web Service Description Language*)
- Opis podataka - XSDL (*XML Schema Definition Language*)
- Otkrivanje usluge - UDDI, ... , upisnici pod kontrolom korisnika

Protokol SOAP

- W3C norma
- Može koristiti razne transportne protokole
 - SMTP, HTTP, HTTPS
- Neovisan o jeziku i platformi
- Zasnovan na XML-u
 - nastao kao nadogradnja XML-RPC-a
- Proširiv

Poruke SOAP-a

- Oblikovanje poruke za pristup usluzi

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

Poruka - zaglavlje

- Zaglavlje sadrži dodatne podatke o poruci

```
<?xml version="1.0"?>
<s:Envelope xmlns:s="http://www.w3.org/2001/12/soap-envelope"
    s:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
    <s:Header>
        <o:Podatak xmlns:o="http://www.fer.hr/rasip/or-usluge-podaci"
            s:actor="http://www.fer.hr/rasip/usluga1/">
            1234
        </o:Podatak>
    </s:Header>
    <s:Body>
        ...
        <s:Fault>
            ...
        </s:Fault>
    </s:Body>
</s:Envelope>
```

Na koga se odnosi ovo zaglavlje

Korisnik definira strukturu podataka

Poruka - tijelo zahtjeva

- Tijelo poruke sadrži opis zahtjeva ili odgovora

```
<?xml version="1.0"?>
<s:Envelope xmlns:s="http://www.w3.org/2001/12/soap-envelope"
    s:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
    <s:Header>
        ...
    </s:Header>
    <s:Body>
        <o:Kvadriraj xmlns:o="http://www.fer.hr/rasip/or-usluge-podaci">
            <o:Broj>
                42
            </o:Broj>
        </o:Kvadriraj>
    </s:Body>
</s:Envelope>
```



Proizvoljna, valjana
struktura XML-a

Poruka - tijelo odgovora

- Tijelo poruke sadrži opis zahtjeva ili odgovora

```
<?xml version="1.0"?>
<s:Envelope xmlns:s="http://www.w3.org/2001/12/soap-envelope"
    s:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <s:Header>
    ...
  </s:Header>
  <s:Body>
    <o:KvadrirajOdgovor xmlns:o="http://www.fer.hr/rasip/or-usluge-podaci">
      <o:Kvadrat>
        1764
      </o:Kvadrat>
    </o:KvadrirajOdgovor>
  </s:Body>
</s:Envelope>
```

Proizvoljna, valjana struktura XML-a

Poruka - grješka

- Grješku je moguće detaljno opisati

```

<?xml version="1.0"?>
<s:Envelope xmlns:s="http://www.w3.org/2001/12/soap-envelope"
  s:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <s:Header>
    ...
  </s:Header>
  <s:Body>
    ...
    <s:Fault>
      <faultcode> ... </faultcode>
      <faultstring> ... </faultstring>
      <faultactor> ... </faultactor>
      <detail> ... </detail>
    </s:Fault>
  </s:Body>
</s:Envelope>

```

Priroda grješke (klijent, server, različite verzije, nerazumijevanje zahtjeva)

Opis grješke

Gdje se dogodila grješka

Detalji o grješki (npr. trag iznimke)

Opis usluge - WSDL

- WSDL (Web Service Description Language)
 - Koristi XSDL za opis strukture podataka

- WSDL dokument sadrži
 - Opis **tipova** podataka (XML zapisa podataka) - XSDL
 - Opis strukture **poruka** (zahtjev, odgovor)
 - Opis **portova** (objekata)
 - Opis **operacija** (metoda)
 - Opis **veza** (*binding*)
 - Veže poziv metode i prijenosni protokol
 - Opis **usluge**
 - Veže adresu (URI) i portove (klase)

Primjer WSDL

- Usluga je dostupna na nekom URI-ju

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<wsdl:definitions ... >
```

```
...
```

```
  <wsdl:service name="KvadratServis">
```

```
    <wsdl:port binding="KvadratSoapBinding" name="Kvadrat">
```

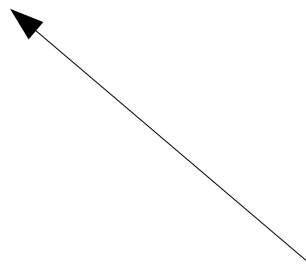
```
      <wsdlsoap:address location="http://www.fer.hr:80/ws" />
```

```
    </wsdl:port>
```

```
  </wsdl:service>
```

```
...
```

```
</wsdl:definitions>
```



Opis usluge

Primjer WSDL

- Usluga sadrži 1..n portova (analogno objektima)

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<wsdl:definitions ... >
```

```
...
  <wsdl:portType name="Kvadrat">
    <wsdl:operation name="kvadriraj" parameterOrder="broj">
      <wsdl:input message="kvadratZahtjev" name="kvadratZahtjev"/>
      <wsdl:output message="kvadratOdgovor" name="kvadratOdgovor"/>
    </wsdl:operation>
  </wsdl:portType>
```

```
...
```

```
</wsdl:definitions>
```

Opis **objekta**, pripadnih **metoda** i **poruka** koje se šalju metodama

Primjer WSDL

- Poruke usluzi šalju se transportnim protokolom

...

```
<wsdl:binding name="KvadratSoapBinding" type="Kvadrat">
  <wsdlsoap:binding style="rpc"
    transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="kvadriraj">
    <wsdlsoap:operation soapAction=""/ >
    <wsdl:input name="kvadrirajZahtjev">
      <wsdlsoap:body namespace="....." use="literal"/>
    </wsdl:input>
    <wsdl:output name="kvadrirajOdgovor">
      <wsdlsoap:body namespace="....." use="literal"/>
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
```

Sve metode HTTP-a, moguće
ograničiti (npr. samo POST)

Veza metoda i
transportnog protokola

...

```
</wsdl:definitions>
```

Primjer WSDL

- Poruke su XML i sadrže jedan ili više podataka

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions ... >
```

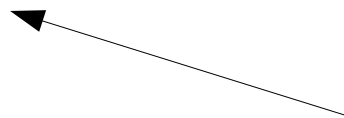
...

```
<wsdl:message name="kvadratZahtjev">
  <wsdl:part name="broj" type="xsd:int"/>
</wsdl:message>
```

```
<wsdl:message name="kvadratOdgovor">
  <wsdl:part name="kvadrat" type="xsd:int"/>
</wsdl:message>
```

...

```
</wsdl:definitions>
```



Poruke zahtjeva i
odgovora i dijelovi
poruka

Primjer WSDL

- Podaci koji se šalju mogu biti složeni

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions ... >
```

Struktura podataka

```
...
  <wsdl:types>
    <schema ... >
      <complexType name="Item">
        <sequence>
          <element name="ime" nillable="true" type="xsd:string"/>
          <element name="prezime" type="xsd:string"/>
          <element name="ulica" nillable="true" type="xsd:string"/>
          <element name="broj" nillable="true" type="xsd:int"/>
        </sequence>
      </complexType>
      <element name="Osoba" nillable="true" type="Osoba"/>
    </schema>
  ...
</wsdl:definitions>
```

Primjer WSDL

- Stvarni redoslijed u dokumentu je obratan
 - Tipovi podataka
 - Poruke
 - Dostupne operacije
 - Dostupni transportni protokoli
 - Usluge vezane na URI

Generiranje WSDL-a

- Postojeći opis klase → Web servis
- WSDL moguće generirati automatski
 - Opis usluge u programskom jeziku sadrži podatke potrebne za opis objekata i metoda u WSDL-u
 - podrazumijevan je SOAP preko HTTP-a
- Web servis opisan izravno u programskom jeziku
 - Dodatne informacije mogu biti opisane atributima u jeziku

Web servisi u .Net-u

- Atributi jezika (C#) pokreću generiranje kôda potrebnog za pretvorbu razreda u Web servis
- Aplikacijski poslužitelj (IIS) pozadinski generira posrednički sloj

```
<%@ WebService Language="C#" Class="WebServis.Kvadrat" %>
using System.Web.Services;
using System.Web.Services.Protocols;

namespace WebServis {
    [WebService(Namespace = "http://www.rasip.fer.hr/")]
    public class Kvadrat : System.Web.Services.WebService {
        [WebMethod]
        public int kvadriraj(int broj) {
            return broj*broj;
        }
    }
}
```

Web servisi u Javi

- Isti mehanizam postoji i u Javi

```
package hr.fer.rasip.ws;

import javax.jws.WebMethod;
import javax.jws.WebService;
import javax.jws.soap.SOAPBinding;

@WebService(name = "Kvadrat")
@SOAPBinding(
    style = SOAPBinding.Style.DOCUMENT,
    use = SOAPBinding.Use.LITERAL,
    parameterStyle = SOAPBinding.ParameterStyle.WRAPPED)
public class Kvadrat {
    @WebMethod()
    public int kvadriraj (int broj) {
        return broj*broj;
    }
}
```

Web servisi u Javi

- Sloboda u odabiru implementacije SOA platforme
 - Postoji više načina za implementaciju
 - Generatori kôda obavljaju mukotrpan dio posla
 - Neki proizvođači
 - Apache
 - IBM
 - Oracle
 - Sun
- Velika sloboda stvara problem
 - Koji pristup odabrati?



SOA - sažetak

- Složenija od XML-RPC-a
- Skup mehanizama
 - SOAP, WSDL, ...
- Labava sprega
 - Izdvojeni opis struktura podataka
 - Izdvojeni opis sučelja usluge
 - Mogućnost odabira transportnog protokola
 - Mogućnost otkrivanja sučelja usluga (WSDL definicije)
 - Usluge objavljuju svoj opis (WSDL)
- Kompozicija usluga
 - Postojanje opisa (WSDL) - podloga za generiranje kôda i automatizaciju

Primjer: REST

- Kritičari postojeće SOA-e smatraju da je moguće ostvariti još slabiju spregu
- REST - resursi, operacije nad resursima
 - Stvaranje, dohvat, promjena, brisanje
 - Dovoljno za ostvarenje bilo koje funkcionalnosti
- Može li se aplikacija ostvariti kao niz REST usluga?
- WSDL 2.0 donosi pomake u ovom smjeru
 - Trenutno rasprostranjena verzija - WSDL 1.1

Zaključak

- Uslužno orijentirana arhitektura
 - pristup izradi složenih aplikacija
 - sličan pristup kao OO, na višoj razini (usluga ima veću funkcionalnost od objekta)

- Usluga - dio funkcionalnosti, crna kutija
 - dobro definirano sučelje
 - samostalna funkcionalnost
 - komunicira porukama (sakriva detalje implementacije)
 - ne ovisi o stanju drugih usluga
 - pruža informacije ili mijenja stanje sustava iz jednog valjanog stanja u drugo (transaktivnost)

Zaključak

- Sprega - međusobni utjecaj djelova aplikacije
 - poželjna je labava sprega
- Labava sprega
 - lakši razvoj - podjela na manje dijelove
 - moguća interna promjena dijela bez utjecaja na cjelinu
 - moguće ponovno korištenje postojećih dijelova
- Web servisi
 - Dostupni preko URI-ja
 - Poruke u obliku XML dokumenta



Pitanja?