SOAP eta Web-Zerbitzuak

XML teknologiak, Web-zerbitzuak SOAP bidez

Iruñea, 2004ko uztailaren 22a

Alberto Silva (alberto@nmp.es)

Aurkibidea

1. Sarrera: ikuspegi orokorra

- 1.1. Zer da SOAP?
- 1.2. Zergatik behar da SOAP?
- 1.3. Nola funtzionatzen du? (I)
- 1.4. Nola funtzionatzen du? (II)
- 1.5. Funtzionamenduaren grafikoa
- 1.6. SOAPen onurak

1.1. Zer da SOAP?

- Leku eta zerbitzari desberdinetan dauden programak komunikatzeko protokoloa
- Lengoaiaren independentzia: programak lengoaia desberdinetakoak izan daitezke
- Sistemen independentzia: sistema eragileak ere guztiz ezberdinak izan daitezke
- XMLn oinarrituta
- HTTP sare-protokolo gisa
- Urruneko konputazioa

1.2. Zergatik behar da SOAP

- SOAP eta Web-zerbitzuak programen “**Babel dorrea**” gainditzeko sortuak dira
- Egun, programak egiteko zenbait lengoaia daude: C, Java, Perl, PHP, Python, e.a.
- Programa horiek sistema eragile desberdinetan funtzionatzen dute
- Programa ezberdinak konektatzea eta komunikatzea zaila eta konplexua da
- Komunikazio hori errazteko, programen “autobideak” eraiki behar dira: Web-zerbitzuak “autoak” dira, eta SOAP “errepidea”

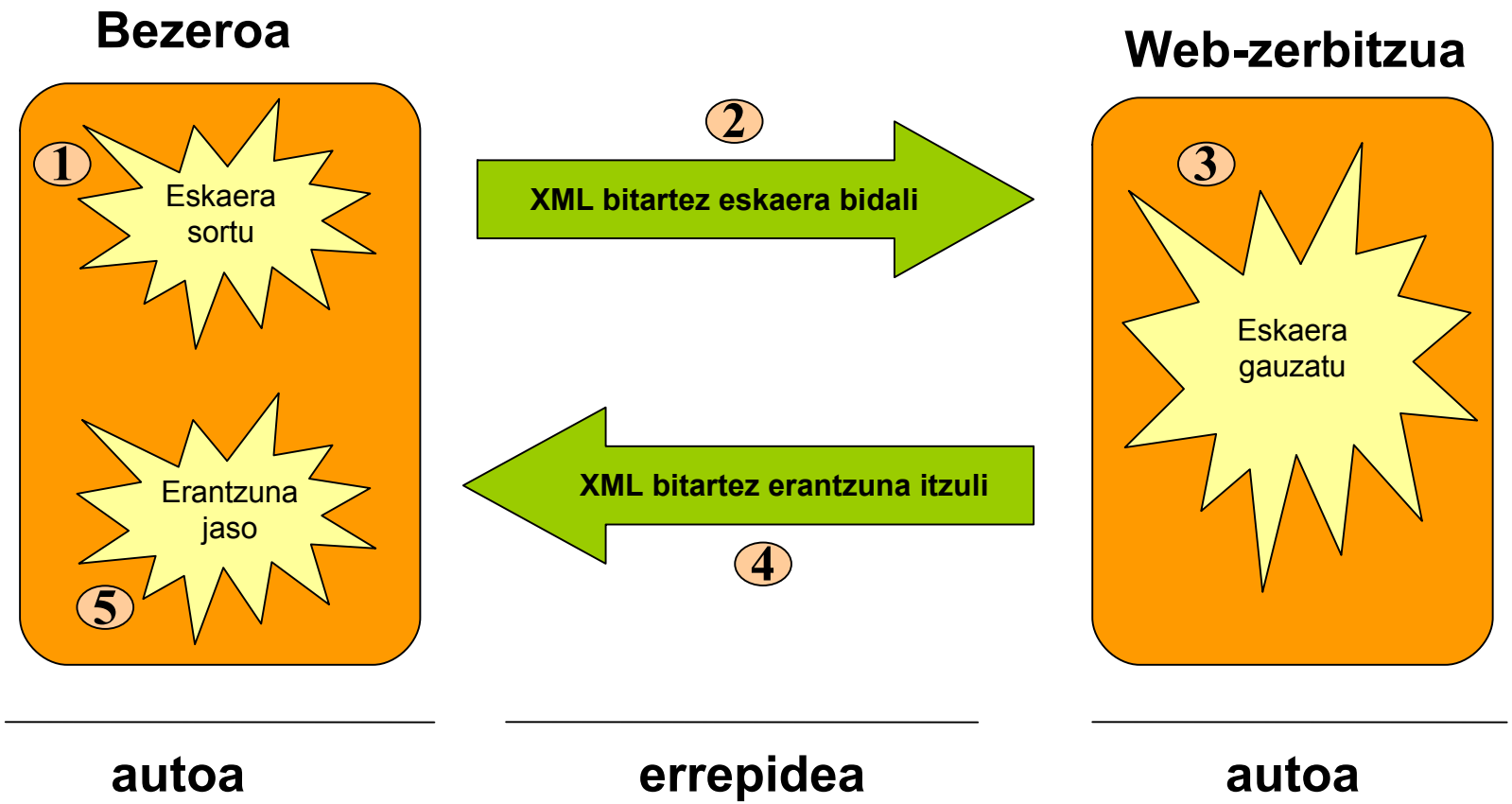
1.3. Nola funtzionatzen du? (I)

- Web-zerbitzua programa normala da, zerbitzarian instalatua
- SOAP erabiliz (HTTP + XML), bezeroak programa horri zerbait egiteko eskatzen dio
- Hori eskatzeko, HTTP bitartez, XML testu bat, formatu jakin batekin, bidaltzen dio

1.4. Nola funtzionatzen du? (II)

- Web-zerbitzuak (programa) eskatutakoa burutzen du
- Bezeroari erantzuteko, SOAP ere erabiltzen du
- HTTP bitartez, XML bat, formatu jakin batekin, bezeroari itzultzen dio
- Bezeroak XML hori irakurri eta Web-zerbitzuari eskatutakoa jasotzen du

1.5. Funtzionamenduaren grafikoa



1.6. SOAPen onurak

- Munduko beste puntan pertsona batek garatu duen programa erabil daiteke ezer instalatu gabe
- Programa konplexuak edo baliabide asko behar dituztenak urrunetik erabil daitezke, bezeroari zama kenduz
- Hasieran gizakiek Internet erabili dute informazioa lortzeko. Orain, programak ere haien artean automatikoki konektatzen dira, gizakien parte-hartzerik gabe: “makinen Internet”.
- Laburbilduz, urruneko konputazioa

2. Zer da SOAP?

- 2.1. SOAPen definizioa
- 2.2. Protokoloaren historia
- 2.3. Onarpen zabala
- 2.4. RPC oinarri
- 2.5. XML: komunikatzeko formatua
- 2.6. HTTP: sare-protokoloa
- 2.7. SOAPen pertsonalizazioa
- 2.8. Web-zerbitzuak garatzeko tresna

2.1. SOAPen definizioa

- SOAP: **S**imple **O**bject **A**ccess **P**rotocol:
  - **Protocol**: Protokolo bat da, irekia eta ezaguna
  - **Access**: sarbidea ematen du, bi partaideren arteko komunikazioa errazten duena
  - **Object**: bi partaideak objektuak dira, eta haien metodoak dei daitezke. Objektuen tipoak ere mantentzen dira.
  - **Simple**: oso erraza da, HTTP eta XML erabiltzen baitu. Biak estandarrak eta irekiak dira.

2.2. Protokoloaren historia

- Microsoft-ek eta Userland-ek asmatutako protokoloa

- Protokoloaren bertsioak (http://www.w3.org/TR/soap/):
  - SOAP 1.1. (2000/05/08). W3Ck onartua, baina ez espezifikazio gisa, proposamen baizik
  - SOAP 1.2. (2003/06/24). W3Ck onartua espezifikazio gisa

2.3. Onarpen zabala

- Estandarra da (W3C)

- Oso azkar ari da zabaltzen eta gero eta gehiago erabiltzen da:
  - Microsoft: .NET
  - Apache: SOAP, Axis
  - Google: Google Web API
  - e.a.

2.4. RPC oinarri

- SOAP RPCtik dator
- *Remote Procedure Call*: beste leku batean dagoen zerbitzari bateko aplizakio bati ekintza bat burutu dezan eskatzea da
- Bezeroak programa bati eskaera bat bidaltzen dio, protokolo bat erabiliz
- Hau da, urruneko funtzioak erabiltzeko balio du
- Inplementazio desberdinak erabil daitezke: DCOM, CORBA, XML-RPC, SOAP, e.a.

2.5. XML: komunikatzeko formatua

- Bezeroa eta zerbitzariaren arteko komunikazioa XMLrekin egiten da
- Eskaera eta erantzuna XML dira
- XML ez da formaturik azkarrena ezta trinkoena ere, baina oso sinplea da
- Formatua oso ulerterraza da, eta arazoak sortuz gero, erraz jakin daiteke, XMLa begiratuz, non dagoen akatsa

2.6. HTTP: sare-protokoloa

- Komunikatzeko XML egitura HTTP bitartez bidaltzen eta jasotzen da
- HTTP ez da protokolorik azkarrena ezta aproposena ere, baina oso sinplea da (XML bezala)
- HTTPk oso azpiegitura xume eta sinple behar du
- Sistema eragile gehienetan HTTP badago
- Adibidea: http://www.nanonull.com/TimeService/TimeService.asmx

2.7. SOAPen pertsonalizazioa

- SOAPek erabiltzen duen XML pertsonaliza daiteke
- XMLk elementu motak eta tipoak zehazten ditu
- Hala ere, tipoak pertsonaliza daitezke
- Beraz, XMLren edukia guztiz doi daiteke, bidaltzen dena zehatz-mehatz deskribatuz

2.8. Web-zerbitzuak garatzeko tresna

- Beste programekin komunikatzeko HTTP eta XML erabiltzen dituzten programak Web-zerbitzuak dira
- Bai XML-RPCek bai SOAPek HTTP eta XML erabiltzen dituzte
- SOAP aukeratzeko arrazoiak:
  - Aukera gehiago ematen ditu
  - W3C espezifikazioa da
  - Etorkizuneko protokoloa da
  - Enpresa handien sostengua dauka

3. SOAP beste teknologiekin alderatuta

- 3.1. Aurrerapauso sendoa
- 3.2. Orain arteko aukerak
- 3.3. *COM/DCOM*
- 3.4. *CORBA/IIOP (Internet Inter-ORB Protocol)*
- 3.5. *JAVA/RMI*
- 3.6. Aurreko protokoloen arazoak
- 3.7. Bateragarritasun eza
- 3.8. Enpresen Aplikazioen Bateragarritasuna
- 3.9. Negozio-Negozio Bateragarritasuna
- 3.10. Irtenbidea: SOAP

3.1. Aurrerapauso sendoa

- Beste teknologien arazoak gainditzen ditu
- Sinple, irekia eta estandarra
- Lengoaia eta sistemekiko independentea
- Informatikagintzaren sostengua

3.2. Orain arteko aukerak

- *COM/DCOM*
- *CORBA/IIOP*
- *JAVA/RMI*

3.3. *COM/DCOM*

- Alde:
  - Lengoaiaren independentea
- Kontra:
  - Mezuen formatua: bitarra
  - Teorian, sistema eragilearen independentea; baina bakarrik *Windows-en* funtzionatzen du
  - *Firewall*-en “etsaia”

3.4. *CORBA/IIOP(Internet Inter-ORB Protocol)*

- Alde:
  - Lengoaiaren independentea
  - Sistemaren independentea

- Kontra:
  - Mezuen formatua: bitarra
  - Azpiegitura zaila eta ulergaitza: *IIOP protokoloa (Internet Inter-ORB Protocol)*
  - Enpresen inplementazio batzuk ez dira bateragarriak
  - *Firewall*-en “etsaia”

3.5. *JAVA/RMI*

- Alde:
  - Sistemaren independentea
- Kontra:
  - Programak bakarrik Javaz egin daitezke
  - Mezuen formatua: bitarra
  - Azpiegitura zaila: aplikazio-zerbitzariak, Java, J2EE, e.a.
  - Protokoloa: *RMI IIOP gainean*. *RMI (Remote Method Invocation)* protokoloak soilik Javarekin funtzionatzen du
  - *Firewall*-en “etsaia”

3.6. Aurreko protokoloen arazoak

- Bateraezinak dira
- Batzuk oso mugatuak dira, bakarrik lengoaia edo sistema baterako (COM eta RMI, adibidez)
- Enpresen garapenak dira, ez dira software librea
- Portu bereziak erabiltzen dituzte, *firewall*-ekin arazoak sortuz
- Orokorrean, oso azpiegitura konplexua behar dute, parte hartzen duten zerbitzari guztietan
- UNIXen filosofiaren kontrakoa: gauza bat ondo egin beharrean, gauza asko saiatzen dira egiten, baina ez dute lortzen: segurtasuna, zabor-bilketa (*garbage collection)*, saioen kudeaketa, transakzioak, e.a.

3.7. Bateragarritasun eza

- Aurreko arazoen ondorioz, orain arte agertu diren sistemak ez dira bateragarriak
- Sistema eragile desberdinak erabiltzen dituzte
- Lengoaiaren dependenteak
- Bi alorretan Sistemen Bateragarritasuna oso garratzitsua da:
  - Enpresen Aplikazioen Bateragarritasuna: *Enterprise Application Integration* (EAI)
  - Negozio-Negozio Bateragarritasuna: *Business-to-Business Integration* (B2Bi):

3.8. Enpresen Aplikazioen Bateragarritasuna

- Enpresek hainbat sistema ezberdin izan ditzakete
- Sistema horien artean komunikatu behar dira, datuak eta informazioa trukatzeko
- Adibidez, Linux zerbitzari bat datu-base baterako, Windows email zerbitzarirako, eta Solaris *LDAP* zerbitzari baterako
- Bateragarritasun ezaren oztopoa gainditzea enpresen erronka izan da beti
- Horren aurrean, askok bateragarritasuna inprimagailu eta faxekin lortzen dute

3.9. Negozio-Negozio Bateragarritasuna

- Enpresa desberdinen arteko elkarrekintza
- Haien artean, negozioak egiteko, datuak eta informazioa trukatu behar dituzte
- Adibidez, bidai-agentzia bat eta hegazkinen bileteen salerosketa
- Normalean, sistemak desberdinak dira, negozioa eta enpresen jarduera oztopatuz

3.10. Irtenbidea: SOAP

- Oso sinplea
- XML formatua, eta ez bitarra
- HTTP sare protokolo gisa, eta ez IIOP edo RMI
- *Firewall*-en laguna
- Ez dago ezer arrarorik instalatu behar
- Objetuak eta datu-tipo konplexuak igorri daitezke.

4. SOAP vs XML-RPC

- 4.1. Aukeratu beharra
- 4.2. Erraztasuna
- 4.3. Helburuak
- 4.4. Nork bultzatua eta egina
- 4.5. Erabakia: SOAP

4.1. Aukeratu beharra

- Bi aukera daude: XML-RPC edo SOAP
- Biek XML eta HTTP erabiltzen dituzte
- Haien artean bateraezinak
- Beraz, bat aukeratu behar

4.2. Erraztasuna

- XML-RPC:
  - Oso sinplea da
  - Espezifikazio sinple eta ulerterraza (7 orri)
  - Programatzaileak ez direnek ere uler dezakete bere XML
- SOAP:
  - Konplexuagoa
  - Espezifikazio osatuagoa (40 orri)
  - Nahiko teknikoa, XSLren izen-eremuak maiz erabiltzen ditu

4.3. Helburuak

- XML-RPC:
  - Ez ditu arazo guztiak konpondu nahi
  - Informazioa trukatzeko sistema eraginkorra eta sinplea nahi du izan
  - Edozein pertsona bere XML ulertzeko gai izan daiteke
  - Protokoloa beste lengoaia edo sistemetara pasatzea erraza izan behar da

- SOAP:
  - XML-RPC motz geratzen den arloak bete nahi ditu
  - Arazo eta egoera guztientzako baliagarria izan nahi du
  - Motz ez geratzeko, erabiltzaileak datu-tipoak ete beste hainbat xehetasun doi ditzake (adibidez, karaktere-kodea: US-ASCII, UTF-8, UTF-16)

4.4. Nork bultzatua eta egina

- XML-RPC:
  - UserLand Software
  - Enpresa bat da, beraz, bere espezifikazioa ez da batzorde ireki baten esku
  - Hala ere, iradokizuneei irekita dago
- SOAP:
  - W3C: *XML Protocol Working Group*
  - Espezifikazio irekia da
  - Microsoft, IBM, Apache, e.a.

4.5. Erabakia: SOAP

- SOAP da etorkizun gehien duena
- Zailagoa da, baina aukera gehiago ematen ditu
- Irekiagoa da
- W3Cren espezifikazio bat da
- Enpresa handi gehienek sostengatzen dute

5. SOAPen funtzionamendua

- 5.1. Funtzionamendu orokorra
- 5.2. SOAP mezuen egitura
- 5.3. Sintaxiaren arauak
- 5.4. SOAP mezuen adibidea
- 5.5. *Envelope* osagaia
- 5.6. *encodingStyle* atributua
- 5.7. *Header* osagaia
- 5.8. *role* atributua
- 5.9. *mustUnderstand* atributua
- 5.10. *relay* atributua
- 5.11. *Body* osagaia (I)
- 5.12. *Body* osagaia (II)
- 5.13. *Fault* osagaia
- 5.14. *Fault* adibidea
- 5.15. HTTP komunikazioa: eskaera
- 5.16. HTTP komunikazioa: erantzuna

5.1. Funtzionamendu orokorra

- SOAP erabiltzeko zerbitzu bat eta bezero bat programatu behar dira
- Zerbitzuak funtzio batzuk izango ditu, parametro batzuk onartzen dituenak.
- Bezeroak XML dokumentu bat, formatu berezi batekin, bidali behar dio zerbitzuari
- XML horretan erabili nahi diren zerbitzuaren funtzioak agertzen dira, baita parametroak ere
- Zerbitzuak XML jasotzen du, eta agindutako funtzioak (beren parametroekin) abiarazten ditu
- Zerbitzuak erantzuna bidaltzen dio bezeroari, XML bitartez

5.2. SOAP mezuen egitura

- XML dokumentu arrunta da, ondorengo osagaiekin:
  - *Envelope*: mezuaren gutun-azala, XML dokumentua SOAP mezu bat dela ezagutarazten duena. Beharrezkoa.
  - *Header*: mezuaren burukoa. Aukerakoa.
  - *Body*: SOAP mezuaren edukia. Eskaeren eta erantzunen informazioa duena. Beharrezkoa.
  - *Fault*: errore-osagaia, hutsegiteei buruzko informazioa ematen duena. Aukerakoa.
- Osagai guztiak ondorengo izen-eremuarekin (namespace) finkatzen dira:
  - http://www.w3.org/2003/05/soap-envelope
- Espezifikazio osoa: http://www.w3.org/TR/soap/

5.3. Sintaxiaren arauak

- SOAP mezua beti XMLrekin egin behar da
- SOAP mezuak beti “SOAP Envelope” eta “SOAP Encoding” izen-eremuak erabili behar ditu
- SOAP mezuak ezin du DTDrik izan
- SOAP mezuak ezin du “XML PI”rik izan (XML *Processing Instruction*)

5.4. SOAP mezuen adibidea

```
<soap:Envelope
xmlns:soap=http://www.w3.org/2003/05/soap-envelope
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
   <soap:Header>
    <n:alertcontrol xmlns:n="http://example.org/alertcontrol">
        <n:priority>1</n:priority>
        <n:expires>2004-07-22T09:30:00-05:00</n:expires>
    </n:alertcontrol>
   </soap:Header>
   <soap:Body>
    <m:alert xmlns:m="http://example.org/alert">
        <m:msg>XML ikastaroa Iruñean eman behar duzu</m:msg>
    </m:alert>
   </soap:Body>
</soap:Envelope>
```

5.5. *Envelope* osagaia

- SOAP mezuaren osagai nagusia
- XML dokumentua SOAP mezu gisa zehazten du
- Gomendagarria:
  - *xmlns:soap* izen-eremua “`http://www.w3.org/2003/05/soap-envelope`” izan behar da

5.6. *encodingStyle* atributua

- Dokumentuan erabiltzen diren data-tipoak zehazteko balio du
- Edozein SOAP osagaitan ager daiteke, eta osagai horren barruan diren beste osagaitan eragina du
- SOAP mezuek ez dute kodeketa lehenetsirik

5.7. *Header* osagaia

- Aukerakoa da, eta mezuari buruzko informazio pertsonalizatua dauka: autentifikazioa, ordainketa, e.a.

```
<soap:Header xmlns:soap="http://www.w3.org/2003/05/soap-envelope" >
   <t:Transaction
   xmlns:t="http://example.org/2001/06/tx"
   soap:mustUnderstand="true">
   5
   </t:Transaction>
</soap:Header>
```

- SOAPek, bere izen-eremuan ("http://www.w3.org/2003/05/soap-envelope"), 3 atributu zehatzen ditu “Header” osagairako:
  - *role:* zein bitartekarik irakurriko dute Header osagaia
  - *mustUnderstand:* elementuak beharrezkoak ala aukerakoak diren zehazteko
  - *relay:* hurrengo bitartekariri mezua pasatzen zaion ala ez zehazteko

5.8. *role* atributua

- SOAP mezu bat igortzen denean, puntu eta leku desberdinetatik igaro daiteke
- Role atributuarekin zein bitartekarik irakurriko dute Header osagaia zehaztu daiteke:
  - http://www.w3.org/2003/05/soap-envelope/role/next
    Bitartekari guztiak, azkena barne, irakurriko dute Header
  - http://www.w3.org/2003/05/soap-envelope/role/none
    Bitartekariek ezin dute Header aintzat hartu
  - http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver
    Azken bitartekariak bakarrik (hartzailea) hartuko du Header osagaia

5.9. *mustUnderstand* atributua

- *Header* osagaiaren barruan dauden elementuak beharrezkoak ala aukerakoak diren zehazteko balio du
- `mustUnderstand="true"` jarriz gero, mezuaren hartzaileak elementua aintzat hartu behar du
- Elementua ez onartzekotan, hartzaileak errore bat eman behar du

5.10. *relay* atributua

- *Header* osagai bat bitartekari batera heltzen denean, bitartekariak irakur dezake ala ez

- Bitartekari horretarako ez denean, hurrengora pasatzen zaion ala ez zehazten du *relay* atributuak

5.11. *Body* osagaia (I)

- Beharrezkoa da
- SOAP mezuaren edukia “body” barruan dago
- Produktu baten prezioa eskatzeko adibidea (eskaera egitea):

```
<soap:Body>
 <m:GetPrice xmlns:m="http://example.org/prices">
     <m:Item>Sagarrak</m:Item>
 </m:GetPrice>
</soap:Body>
```

5.12. *Body* osagaia (II)

- Aurreko eskaeraren erantzuna:

```
<soap:Body>
 <m:GetPriceResponse xmlns:m="http://example.org/prices">
     <m:Price>1.90</m:Price>
 </m:GetPriceResponse>
</soap:Body>
```

(“m” izen-eremua ez da SOAP estandarraren parte. Aplikazioaren elementu espezifikoak dira)

5.13. *Fault* osagaia

- SOAPen erroreak *Fault* osagaiaren barruan doaz
- Bakarrik *Fault* osagai bat ager daiteke SOAP mezu bakoitzeko
- Ondorengo elementuak dauzka:
  - *Code* (beharrezkoa): *VersionMismatch, MustUnderstand, DataEncodingUnknown, Sender, Receiver*
  - *Reason* (beharrezkoa)
  - *Node* (aukeran)
  - *Role* (aukeran)
  - *Detail* (aukeran)

5.14. *Fault* adibidea

```
<soap:Body>
   <soap:Fault>
    <soap:Code>
        <soap:Value>env:Sender</soap:Value>
        <soap:Subcode>
            <soap:Value>m:MessageTimeout</soap:Value>
        </soap:Subcode>
    </soap:Code>
   <soap:Reason>
    <soap:Text xml:lang="en">Sender Timeout</soap:Text>
   </soap:Reason>
   <soap:Detail>
    <m:MaxTime>P5M</m:MaxTime>
   </soap:Detail>
   </soap:Fault>
</soap:Body>
```

5.15. HTTP komunikazioa: eskaera

```
POST /InStock HTTP/1.1
Host: example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: xxx

<soap:Envelope
xmlns:soap=http://www.w3.org/2003/05/soap-envelope
soap:encodingStyle="http://www.w3.org/2003/05/soap-
   encoding">
<soap:Body>
    <m:GetPrice xmlns:m="http://example.org/prices">
        <m:Item>Sagarrak</m:Item>
    </m:GetPrice>
   </soap:Body>
</soap:Envelope>
```

5.16. HTTP komunikazioa: erantzuna

```
HTTP/1.1 200 OK
Content-Type: application/soap; charset=utf-8
Content-Length: xxx

<soap:Envelope
xmlns:soap=http://www.w3.org/2003/05/soap-envelope
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
   <soap:Body>
    <m:GetPriceResponse xmlns:m="http://example.org/prices">
        <m:Price>1.90</m:Price>
    </m:GetPriceResponse>
   </soap:Body>
</soap:Envelope>
```

6. Web-zerbitzuak

- 6.1. Zer dira Web-zerbitzuak?
- 6.2. Sistema eta lengoaien batasuna
- 6.3. Komunikazio automatikoa
- 6.4. Berrerabilgarritasuna
- 6.5. Urruneko konputazioa
- 6.6. Eguneratze gardena
- 6.7. Noiz dira gomendagarria?
- 6.8. Noiz ez dira gomendagarria?
- 6.9. Web-zerbitzuen adibideak

6.1. Zer dira Web-zerbitzuak?

- Edozein programa, SOAP (edo XML-RPC) bitartez komunikatzen dena
- Web-zerbitzari batean instalatuta daude (Apache, IIS, e.a.)
- HTTP protokoloari erantzuten diote (SOAP edo XML-RPC bitartez)

6.2. Sistema eta lengoaien batasuna

- Sistema eragile eta lengoaia ezberdinak batzen ditu
- Web-zerbitzuek edozein sistema eragiletan funtzionatzen dute
- Edozein lengoaiatan izan daitezke programatuta
- Soilik SOAP protokoloa inplementatu behar dute haien artean komunikatzeko

6.3. Komunikazio automatikoa

- Programak haien artean komunika daitezke, automatikoki, gizakien parte-hartzerik gabe
- Web-zerbitzaritan 2 baliabide daude:
  - Web-orriak: gizaki bat bisitatu ahal izateko egina daude
  - Web-zerbitzuak: programa bat bisitatu ahal izateko egina daude
- World Wide Web paraleloa: makinen Internet

6.4. Berrerabilgarritasuna

- Berrerabilgarriak dira, mundu osoko pertsonek eta programek erabil ditzakete
- Web-zerbitzu ezberdinak batuz programa osoa egin daiteke
- Sarea funtzio-liburutegi bihurtzen da
- Arazoa: programen funtzionamendua besteen esku uzten da

6.5. Urruneko konputazioa

- Konputazio-lana Interneten banatzen da
- Ekintza bat oso zaila bada, edo baliabide asko behar baditu, Web-zerbitzu batera bidal daiteke
- Exekuzioa: paraleloa eta azkarragoa

6.6. Eguneratze gardena

- Web-zerbitzuaren funtzionamendua alda daiteke, bezeroa abisatu gabe
- Interfazea mantentzea: bakarrik metodoen izenak eta parametroak mantendu behar dira

6.7. Noiz da gomendagarria

- Gure esku ez dauden gauzekin ekintza bat burutu behar dugunean:
  - liburu baten prezioa jakin
  - burtsako kotizazio bat lortu
  - beste zerbitzari batean dagoen datu-basean datu bat lortu
  - e.a.
- Oso ekintza konplexua denean: berrerabilgarritasuna.
- Baliabide asko behar direnean
- Segurtasunagatik, zerbitzari batekin bakarrik 80 portutik konekta gaitezkenean

6.8. Noiz ez da gomendagarria

- Ekintza sinplea denean
- Bezeroa eta Web-zerbitzariaren artean datu gehiegi trukatu behar direnean: bideo trinkotzea
- Programaren zati garrantzitsu bat besteen esku utzi nahi ez denean

6.9. Web-zerbitzuen adibideak

- *Google API* (google.com/apis): bilaketak eta zuzentzailea
- *Barnes and Noble*: liburuen prezioa ISBN emanda
- *Network Solutions*: domeinu baten jabea ezagutzea
- *Flash-db.com*: flash irudiak eta grafikoak sortzea
- http://live.capescience.com/GlobalWeather (KJFK parametro gisa)
- *services.xmethods.net*: web-zerbitzuen sailkapena

7. WSDL: SOAP automatizatzen

- 7.1. Zer eskaintzen du zerbitzuak?
- 7.2. Interfazea adierazteko estandarra
- 7.3. WSDL adibidea
- 7.4. Kodeketa errazten du

7.1. Zer eskaintzen du zerbitzuak?

- Web-zerbitzuak "kutxa beltzak" dira: bezeroak ez daki barruan nola funtzionatzen duten
- Bezeroak jakin behar du zein funtzio dauzka zerbitzuak eta zein parametro onartzen ditu
- Informazio hori emateko modu estandarra behar da

7.2. Interfazea adierazteko estandarra

- *Web Services Description Language (WSDL)* edo Web-zerbitzuak Deskribatzeko Lengoaia (http://www.w3.org/TR/wsdl)

- XMLrekin egina

- Web-zerbitzu bati nola konektatu eta zein ekintza egin ahal izango dituen adierazten du

7.3. WSDL adibidea

```
<definitions name="GoogleSearch" targetNamespace="urn:GoogleSearch">
    <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"/>
    </message>
    <service name="GoogleSearchService">
      <port name="GoogleSearchPort" binding="typens:GoogleSearchBinding">
            <soap:address location="http://api.google.com/search/beta2"/>
      </port>
    </service>
</definitions>
```

- http://api.google.com/GoogleSearch.wsdl
- http://www.nanonull.com/TimeService/TimeService.asmx?WSDL
- http://www.xmethods.com (edozein Web-zerbitzuren adibidea)
- http://www.w3.org/2000/06/webdata/xslt?xslfile=http://www.capescience.com/simplifiedwsdl.xslt&xmlfile=http://api.google.com/GoogleSearch.wsdl&transform=Submit

7.4. Kodeketa errazten du

- WSDLk Web-zerbitzuen erabilera errazten du
- Zenbait programek WSDL fitxategia hartuz konektazeko programa egin dezakete
- Tresnak: .NET, Apache Axis, 4s4c.com, e.a.

8. UDDI: Web-zerbitzuen aurkibidea

- UDDIk Universal Discovery, Description and Integration (UDDI) esan nahi du
- Aurkibide bat da, WSDL eta Web-zerbitzuez osatuta
- Enpresek haien Web-zerbitzuak UDDI-n erregistratzen dituzte, eta beste enpresa batzuek Web-zerbitzuak aurki ditzakete
- Lehen ez zegoen enpresen artean produktuak eta zerbitzuak aurkitzeko modu estandarrik
- Ia enpresa handi guztiak UDDI barruan daude
- Web-zerbitzuak aurkitzeko balio du, milaka eta milaka baitaude
- UDDI agertu zenean, programazioa desagertuko zela zabaldu zen, dena automatikoa izango baitzen (UDDI + UML erabiliz). Gaur, aurreikuspen horiek ez dira bete.
- http://www.uddi.org/find.html

9. Adibideak

- 9.1. PHP: Web-zerbitzuaren kodea
- 9.2. PHP: bezeroaren kodea
- 9.3. Perl: Web-zerbitzuaren kodea
- 9.4. Perl: bezeroaren kodea

9.1. PHP: Web-zerbitzuaren kodea

- Nusoap programategia erabiliz (http://dietrich.ganx4.com/nusoap/)

```
<?php
function euskaratu($erdarazko_hitza) {
// Hiztegirako baliokideak zehaztuko ditugu
  $hiztegia["house"] = "etxe";
  $hiztegia["sun"] = "eguzki";
  $hiztegia["moon"] = "ilargi";

  if ($hiztegia[$erdarazko_hitza]) { return $hiztegia[$erdarazko_hitza]; }
  else { return new soap_fault('Sender','',"'$erdarazko_hitza' hitza ez dugu aurkitzen"); }
}

// SOAP klasea txertatuko dugu
require_once('lib/nusoap.php');
// Zerbitzu objektua sortuko dugu
$zerbitzua = new soap_server;
// Zeregina erregistratu
$zerbitzua->register('euskaratu');
// erantzuna bidali
$zerbitzua->service($HTTP_RAW_POST_DATA);
?>
```

9.2. PHP: bezeroaren kodea

- Nusoap programategia erabiliz (http://dietrich.ganx4.com/nusoap/)

```
<?php
// SOAP klasea txertatuko dugu
require_once('lib/nusoap.php');

// Parametroak zehatzuko ditugu
$parametroak = array('erdarazko_hitza'=>'house');
$helbidea ='http://domain.com/service.php';

// Bezero objektua sortuko dugu
$bezeroa = new soapclient($helbidea);

// Web-zerbitzua deituko dugu
$euskarazko_hitza = $bezeroa->call('euskaratu',$parametroak);

// Hutsegiteak kontrolpean
if ($bezeroa->fault) { echo $bezeroa->getError(); }
else { echo "'" . $parametroak["erdarazko_hitza"] . "' hitza euskaraz: $euskarazko_hitza"; }
?>
```

9.3. Perl: Web-zerbitzuaren kodea

- SOAP::Lite programategia erabiliz (http://www.soaplite.com)

```
#!/usr/bin/perl
use SOAP::Transport::HTTP;
SOAP::Transport::HTTP::CGI
  -> dispatch_to('Hiztegia')
  -> handle;

package Hiztegia;
# Zerbitzuaren zeregina zehaztuko dugu
sub euskaratu {
  my ($class, $hitza) = @_;

  $hiztegia{"house"} = "etxe";
  $hiztegia{"sun"} = "eguzki";
  $hiztegia{"moon"} = "ilargi";

  # Hitza itzuliko dugu
  return $hiztegia{$hitza};
}
```

9.4. Perl: bezeroaren kodea

- SOAP::Lite programategia erabiliz (http://www.soaplite.com/)

```
#!/usr/bin/perl

# Programategia txertatuko dugu
use SOAP::Lite;

# Erantzuna erakutsiko dugu
print "Content-Type: text/html; charset=ISO-8859-1\n\n";
print SOAP::Lite
  -> uri('http://www.soaplite.com/Hiztegia')
  -> proxy('http://domain.com/server.cgi')
  -> euskaratu("house")
  -> result;
```