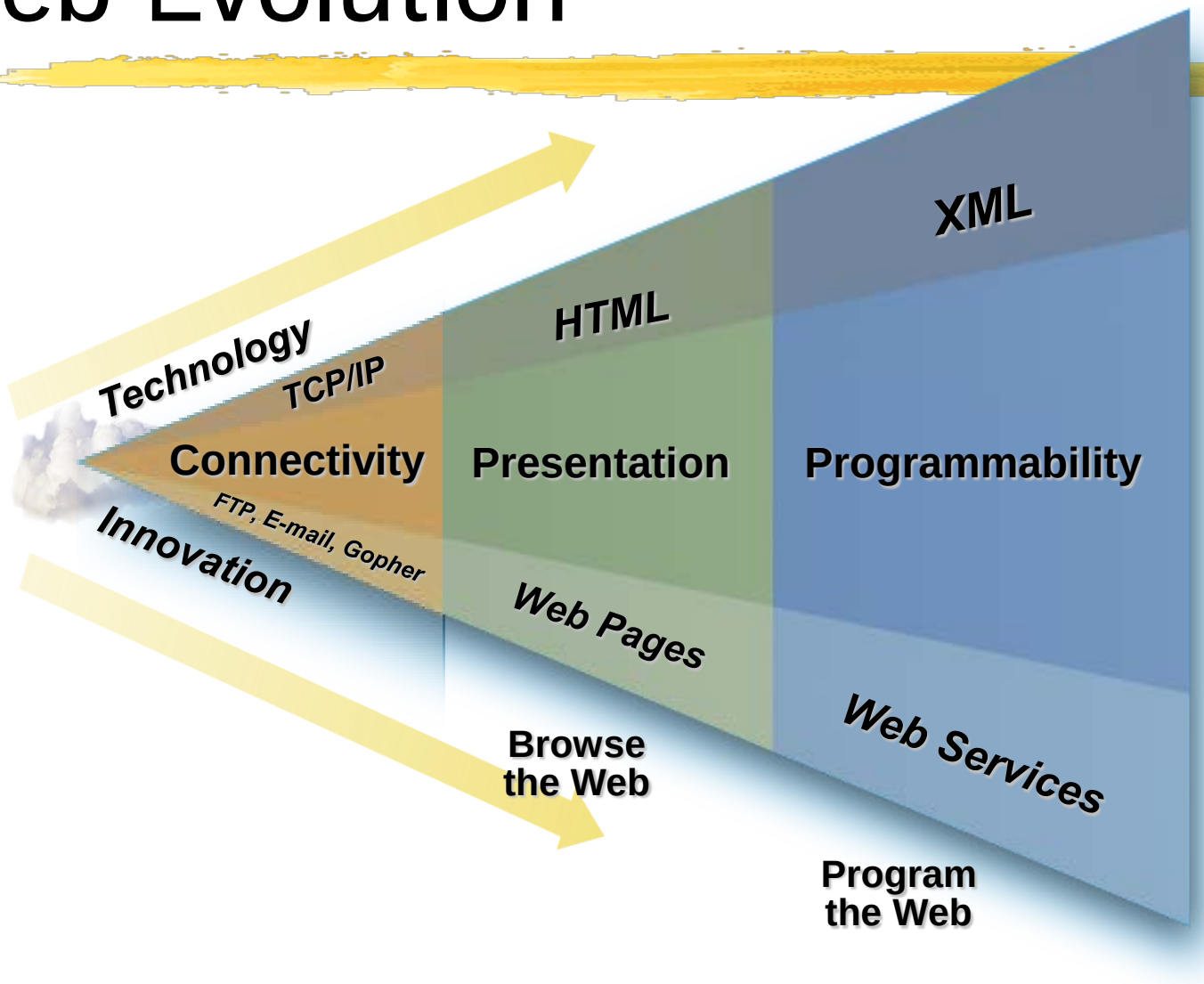
Introduction to Web Services

Outline

- What are Web Services?
- Why Web Services?
- Enabling Technologies?
- What is Web Service Composition?
- Main Issues concerning the composition?

Web Evolution



Service Oriented Architecture

- Service Oriented Architecture (SOA)
  - Type of distributed system
  - Services must cooperate to implement a single functionality.
  - Enables remote objects and services (functions) invocation
  - Provides tools for dynamic service discovery, placing emphasis on interoperability.
  - Components publish their functionality in a centralised registry, and export a network-level API so that other services can programmatically access them
- Decouple business logic from presentation logic.
- Design an application as a set of services.
- Next generation applications dynamically resolve service needs.

What are Web Services?

- Definition from W3C

  "A Web service is a software system identified by a *URI*, whose public interfaces and bindings are *defined* and *described* using XML. Its definition can be *discovered* by other software systems. These systems may then *interact* with the Web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols".

What are Web Services?

- Every component that
  - works in a network,
  - is modular,
  - is self-descriptive,
  - provides services independent of platform and application,
  - conforms to an open set of standards and
  - follows a common structure for description and invocation.
  - presents an API for communication using established protocols.

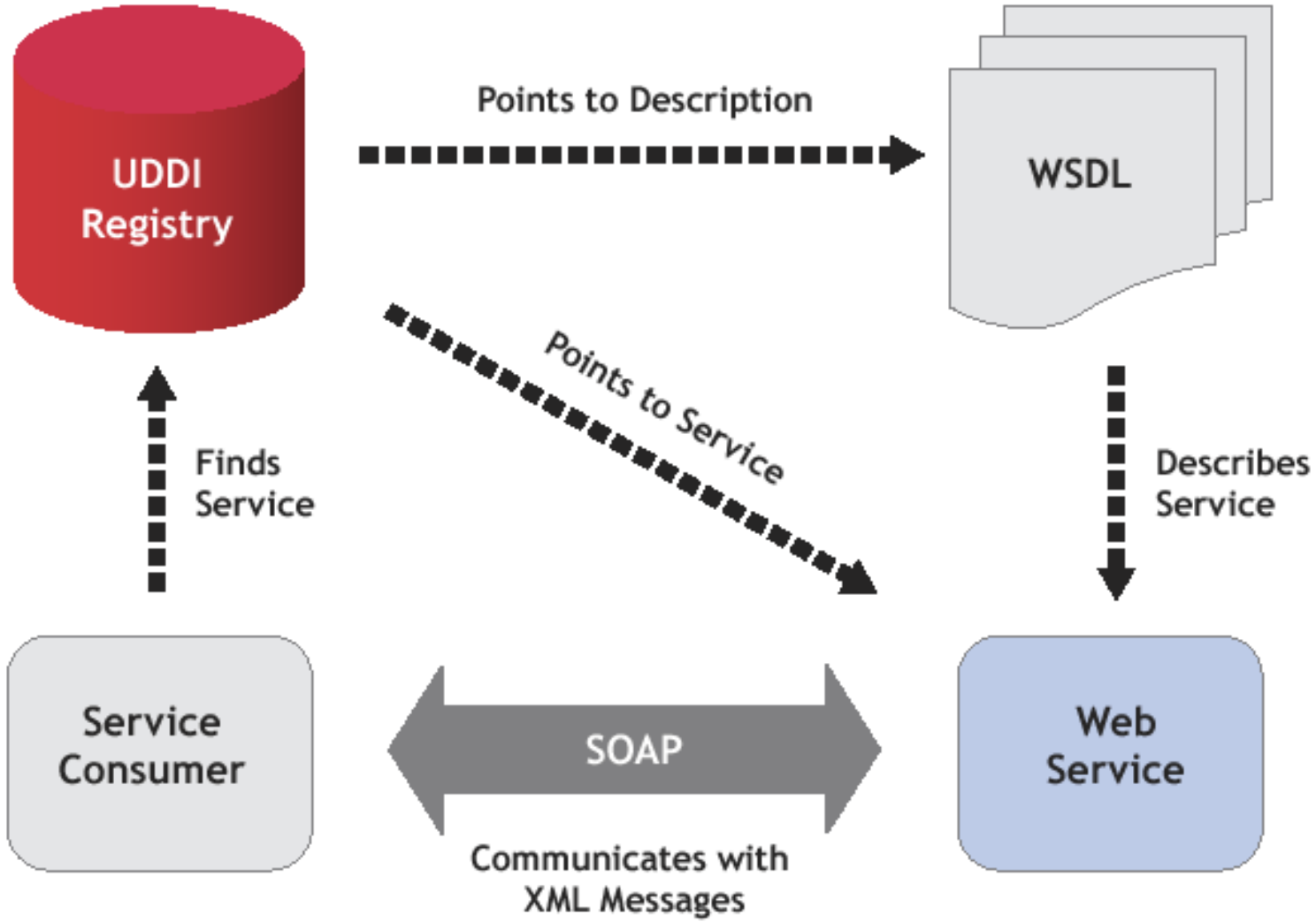
Why Web Services

- Interoperability.
  - Any WS can interact with any other WS.
- Ubiquity.
  - Any device which supports HTTP + XML can host and access WS.
- Effortless entry in this concept.
  - easily understood + free toolkits
- Industry Support.
  - major vendors support surrounding technology.
- Platform, transport, language independence

Web Services Roles

- Components
  - Service Providers
  - Service Brokers
  - Service Requestors
- Operations
  - Publish / Unpublish
  - Find
  - Bind

Web Services Technologies



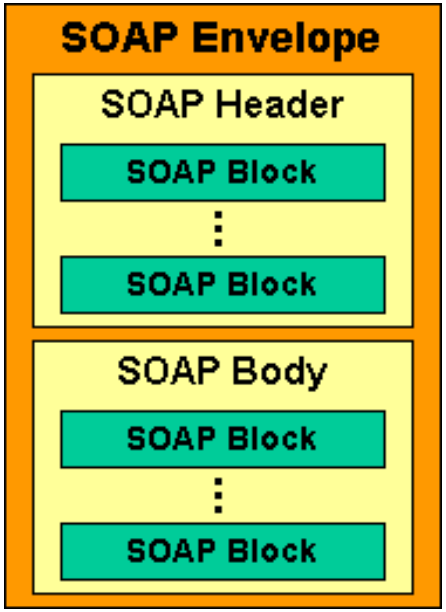
Enabling technologies

- They encapsulate a set of standards that allow the developers to implement distributed applications.
- XML (eXtensible Markup Language)
  - PHTML successor. Provides a neutral format for data.
- SOAP (Simple Object Access Protocol),
  - XML messaging protocol for basic service interoperability
- WSDL (Web Service Description Language)
  - Common grammar for describing services
- UDDI (Universal Description Discovery and Integration)
  - infrastructure required to publish and discover services.

SOAP

- Uniform way of
  - passing XML-encoded data.
  - simulates RPCs over SMTP, FTP, TCP/IP, HTTP

1. The requestor sends a msg to the service
2. The service processes the msg.
3. The service sends back a response.



The requestor has **no** knowledge of how the service is implemented.

SOAP Example (travel reservation)

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">

    <env:Header>
        <m:reservation xmlns:m=http://travelcompany.example.org/reservation
                       env:role=http://www.w3.org/2003/05/soap-envelope/role/next
                       env:mustUderstand="true">
            <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
            <m:dateAndTime>2001-11-29T13:20:00.000-05:00</m:dateAndTime>
        </m:reservation>

        <n:passenger xmlns:n="http://mycompany.example.com/employees" ...>
            <n:name>Åke Jógvan Øyvind</n:name>
        </n:passenger>
     </env:Header>

    <env:Body>
          <p:itinerary xmlns:p="http://travelcompany.example.org/reservation/travel">
              <p:departure>
                    <p:departing>New York</p:departing>
                    <p:arriving>Los Angeles</p:arriving>
                    <p:departureDate>2001-12-14</p:departureDate>
                    <p:departureTime>late afternoon</p:departureTime>
              </p:departure>
          </p:itinerary>
    </env:Body>
</env:Envelope>
```

SOAP - RPC

- Must define an RPC protocol
  - How will types be transported (in XML) and how application represents them.
  - RPC parts (object id, operation name, parameters)

→SOAP assumes a type system based on XML-schema.

SOAP Example - doGoogleSearch

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV= http://schemas.xmlsoap.org/soap/envelope/
    xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/1999/XMLSchema">

    <SOAP-ENV:Body>
    <ns1:doGoogleSearch xmlns:ns1="urn:GoogleSearch" SOAP-
    ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
                <key
    xsi:type="xsd:string">00000000000000000000000000000000</key>
                <q xsi:type="xsd:string">my query</q>
                <start xsi:type="xsd:int">0</start>
                <maxResults xsi:type="xsd:int">10</maxResults>
                <filter xsi:type="xsd:boolean">true</filter>
                <restrict xsi:type="xsd:string"/>
                <safeSearch xsi:type="xsd:boolean">false</safeSearch>
                <lr xsi:type="xsd:string"/>
                <ie xsi:type="xsd:string">latin1</ie>
                <oe xsi:type="xsd:string">latin1</oe>
            </ns1:doGoogleSearch>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP Example - doGoogleSearchResult

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" ……
    <SOAP-ENV:Body>
    <ns1:doGoogleSearchResponse xmlns:ns1="urn:GoogleSearch" SOAP-
            ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
            <return xsi:type="ns1:GoogleSearchResult">
            <documentFiltering xsi:type="xsd:boolean">false</documentFiltering>
            <estimatedTotalResultsCount
                    xsi:type="xsd:int">3</estimatedTotalResultsCount>
            <directoryCategories xmlns:ns2="http://schemas.xmlsoap.org/soap/encoding/"
                    xsi:type="ns2:Array" ns2:arrayType="ns1:DirectoryCategory[0]"/>
            <searchTime xsi:type="xsd:double">0.194871</searchTime>
            <resultElements xmlns:ns3="http://schemas.xmlsoap.org/soap/encoding/"
                    xsi:type="ns3:Array" ns3:arrayType="ns1:ResultElement[3]">
            <item xsi:type="ns1:ResultElement">
            <cachedSize xsi:type="xsd:string">12k</cachedSize>
            <directoryCategory xsi:type="ns1:DirectoryCategory">Category</directoryCategory>
            <relatedInformationPresent
                    xsi:type="xsd:boolean">true</relatedInformationPresent>
            <directoryTitle xsi:type="xsd:string"/>
            <summary xsi:type="xsd:string"/>
            <URL xsi:type="xsd:string">http://hci.stanford.edu/cs147/example/shrdlu/</URL>
            <title xsi:type="xsd:string">&lt;b&gt;SHRDLU&lt;/b&gt;</title>
            </item>
```

WSDL

- IDL of Web Services in XML format.
- Uses these elements to define network services:
  - Types: Container for data type definitions (e.g. XSD)
  - Message: Definition of the communicated data
  - Operation: Description of an action supported by the service
  - Port Type: Set of operations supported by endpoints
    - ✓ Req/resp, one-way, solicit-response, notification
  - Binding: Concrete protocol & data format specification for a particular port type.
  - Port: Single endpoint (Binding + network address)
  - Service: Collection of related endpoints

Vocabulary

```
<wsdl:types>
  <xsd:schema xmlns="http://www.w3.org/2001/XMLSchema"
    targetNamespace="urn:GoogleSearch">
  <xsd:complexType name="GoogleSearchResult">
    <xsd:all>
      <xsd:element name="documentFiltering" type="xsd:boolean"/>
      <xsd:element name="searchComments" type="xsd:string"/>
      <xsd:element name="estimatedTotalResultsCount" type="xsd:int"/>
      <xsd:element name="estimateIsExact" type="xsd:boolean"/>
      <xsd:element name="resultElements"
    type="typens:ResultElementArray"/>
      <xsd:element name="searchQuery" type="xsd:string"/>
      <xsd:element name="startIndex" type="xsd:int"/>
      <xsd:element name="endIndex" type="xsd:int"/>
      <xsd:element name="searchTips" type="xsd:string"/>
      <xsd:element name="directoryCategories"
                   type="typens:DirectoryCategoryArray"/>
      <xsd:element name="searchTime" type="xsd:double"/>
    </xsd:all>
</xsd:complexType>
```

Message

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

Interaction

```
<binding name="GoogleSearchBinding" type="typens:GoogleSearchPort">
    <soap:binding style="rpc"
    transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="doGetCachedPage">
      <soap:operation soapAction="urn:GoogleSearchAction"/>
      <input>
          <soap:body use="encoded"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="urn:GoogleSearch"/>
      </input>
      <output>
          <soap:body use="encoded"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                    namespace="urn:GoogleSearch"/>
      </output>
    </operation>
```

UDDI

- Global business registry
- Root under www.uddi.org
- Relies on:
  - TCP/IP, HTTP, HTTPS
  - XML, XML Schemas, WSDL
  - SOAP

Web Service Composition

- Definition: Technique of composing the functionalities of relatively simpler services to produce a 'meaningful' arbitrarily complex application.

WS composition - Classification

- Proactive Composition & Reactive Composition
  - Proactive: offline composition of available services
    - ✓When: services are stable and always running
    - ✓Example: ticket reservation service
  - Reactive: dynamically creating a composite service.
    - ✓When: composite service not often used and service processes not stable.
    - ✓Example: tour manager where the itinerary is not predefined

WS composition – Classification (2)

- Mandatory & Optional-Composite Services
  - Mandatory: all subcomponents must participate to yield a result
    - ✓Example: service that calculates the averages of stock values for a company.
  - Optional: subcomponents are not obligated to participate for a successful execution.
    - ✓Example: services that include a subcomponent that is an optimizer.

Important issues on WS composition

- **Service Discovery**
- Service Coordination and Management
- Uniform Information Exchange Infrastructure
- Fault Tolerance and Scalability
- Adaptiveness
- **Reliability** & **Transactions**
- **Security**
- Accountability
- Testing

Service Discovery

- An efficient discovery structure should be able:
  - find out all services implementing some functionality (ontology)
  - semantic level reasoning (discover most appropriate service).
  - scalable.
- Most of existing discovery infrastructures use a central lookup server (Jini, UPnP)
- Semantic Language: DAML-S, a process modelling language for computer-interpretable description of services.
  - AI inspired description logic-based language, built on top of XML + RDF for well-defined semantics and a set of language constructs and properties.

Service Discovery - DAML-S

- Enables automatic Web Service discovery. =automatic location of services with required functionality.
- Currently performed manually
- DAML-S: expressed in computer interpretable semantic markup.

Service Discovery - Example of DAML-S

```
<daml:Class rdf: ID="CompositeProcess">
<daml:intersectionOf rdf:parseType = "daml:collection">
    <daml:Class rdf:about="#Process"/>
    <daml:Restriction daml:minCardinality="1">
        <daml:onProperty rdf:resource="#composedOf"/>
    </daml:Restriction>
</daml:intersectionOf>
</daml:Class>

<rdf:Property rdf:ID="composedOf">
    <rdfs: domain rdf:resource="#CompositeProcess"/>
    <rdfs: range rdf:resource="#ControlConstruct"/>
</rdf:Property>
```

Reliability & Transactions

- How we can measure reliability?
- WS descriptions may lie!
- Transactions are fundamental to reliable distributed computing.
- Traditional transaction systems support *ACID* semantics, use a two-phase commit approach: all participating resources are locked until entire transaction is completed.
  - Only in close environments where transactions are short-lived
  - Not on an open environment (flexibility in how it is attained)
    - ✓ MS XLANG: compensating transactions.
    - ✓ Split the model into concurrent sub-transactions that can commit independently (requires compensation over committed sub transactions in case of abortion).

Security

- Basic security: HTTP over SSL
- Authorisation control.
  - Existing authorisation control frameworks not applicable to WS (designed for some services e.g. network access control (DIAMETER) or not well designed to access different administrative domains (.NET Passport))
  - Proposal: generic authorisation control protocol based on SOAP/XML. Supports credential transformation.
    - ✓ Need for CA in each domain. It will issue users and services with certificate and secret key pairs used for user authentication and request signing.
    - ✓ Credentials described in an XML-based language. Authorisation server validates the certificate, credentials etc. If everything is successfully validated, the authorisation server sends back a SOAP response containing the result.

References

1. Dipanjan Chakraborty, *Service Composition in Ad-Hoc Environments*. Ph.D Dissertation Proposal, University of Maryland, Baltimore County, 2001.
2. Dipanjan Chakraborty, Technical Report TR-CS-01-19: *Dynamic Service composition: State-of-the-Art and Research Directions.* University of Maryland, Baltimore County, 2001.
3. Anans Rajamam, "Overview of UDDI", Online, 2001.
4. F.Curbera and al, "Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI". IEEE Internet Computing March-April 2002, p.86-93.
5. Takashi Suzuki, Randy H.Katz, *An authorization control framework to enable service composition across domains*. University of California, Berkeley.
6. DAML Service Coalition, *DAML-S Semantic Markup for Web Services*. Online at http://www.daml.org/services/daml-s/2001/10/daml-s.html, 2001.
7. WSDL Specification, Online at http://www.w3c.org/TR/wsdl.
8. Steve Vinoski, *Web Services and Dynamic Discovery*, Online at http://www.webservices.org/article.php?sid=389, 2001.
9. UDDI Specification, Online at http://uddi.org/.
10. UDDI Technical White Paper, Online at http://uddi.org/, 2000.
11. Sheila A. McIlaith, Tran Cao Son, Honglei Zeng, *Semantic Web Services*, IEEE Intelligent Systems, 2001
12. Vladimir Tosic, Bernard Pagurek, Babak Esfandiari, Kruti Patel, *On the Management of Composition of Web Services*, Carleton University, Canada.
13. Tom Clements, "Overview of SOAP". Online at: http://dcb.sun.com/practices/webservices/overviews/overview_soap.jsp
14. Deitel,"Web Services: A technical Introduction", Prentice Hall, 2002.
15. W3C Web Services Architecture. Online at: http://www.w3.org/TR/2003/WD-ws-arch-20030514/. W3C Working Draft 14 May 2003.