Web Services
Hope or Hype?

Richard P. Martin
Rutgers University

Outline

- Introduction
- Promise and vision
- Description and example
- Our Experiences
- Security
- Future Directions

Web Service Overview

- Definition: A set of representations and protocols to export methods over the Internet

- A Remote Procedure Call (RPC)
  - Call method **`foo(X,Y,Z)`** on your server
  - Your server computes something and sends a response

The Promise

- Sharing of data and programming effort on a scale not seen before
  - Software reuse not by sharing code, but by calling someone else's running program
    - Examples: Google, random number generator
- Order-of-magnitude reduction in time and effort to interconnect systems
  - E.g., 1 programmer hours/days vs. team 6-12 months

Our Experience

- Built a model corporate IT infrastructure using Web Services as part of a graduate course.

- Conclusions:
  - Decent infrastructure for tying systems together
  - Can deliver on the vision
  - Not yet seamless (but will be soon)
  - Requires an excellent network

The Skeptic

- Original RPC paper published in 1981
  - B. J. Nelson, Xerox PARC Tech Report CSL-81-9
- Long list of “failed” RPC technologies:
  - 1980’s Distributed Computing Environment (DCE)
  - 1990’s Common Object Request Broker (CORBA)
  - 1990’s Distributed Component Object Model (DCOM)
  - Those were just the well-known ones
- All had similar promises, lots of hype.
  - None ended up meeting the claims

Key differences from the past

- Platform neutral
- Simple to implement and use
  - E.g. My random perl script running on FreeBSD can invoke a method on your Microsoft exchange sever.
- Open Source/Free implementations that work
  - Don't need to spend big $ to get started
    - E.g., used it in class for free.
- Incremental deployment
  - Can layer on top of existing systems
  - Can incrementally scale up small islands

Similarities to other successes

Networking:

- Zoo of competing protocol stacks in 1970's-1980's
  - LU.6 (IBM), DECnet, OSI, TCP/IP
- Formatting nightmares
    - E.g., ASCII vs. EBCDIC, big vs. little endian, ASN.1 …
- No open implementations => expensive
  - OSI stack for a machine cost 100's of $
- Resulting "islands"
- Big promises, but hard to actually share data
  - Could do it, but with a lot of effort

Similarities to networking (cont)

- IP changed the environment
  - Single protocol to interconnect all existing islands
  - Open source implementation that worked (BSD)
    - Ported to everything, reference for new implementations
  - Standard API to access (BSD sockets)
- Resulting applications made it possible to share data at low cost
    - FTP, SMTP, NFS
- Realized vision of “internetworking”
  - Huge success, delivered on hype.

Information Retrieval Similarities

- HTTP/HTML:
  - Protocols to interconnect existing islands
    - Open FTP sites, Wais, Gopher
    - Usable open source implementations (apache)
    - Mosaic browser was free and widely ported, close enough
  - Standard APIs (e.g. CGI)
- Resulting applications made it possible to share data at low cost
- Realized vision of “Global Hypertext”
  - Huge success, delivered on hype.

Open Issues

- Will web-services deliver on a technical level?
  - Experience with class says “yes”

A sufficient environment for the vision of ‘ecologies’ of services?

  - Commercial
    - Inventory, HR, financial
  - Scientific
    - Simulations, monitoring, modeling, experiments
  - Medical
    - Records, diagnosis, patient care

- Can WS deliver on the hype?

Outline

- Introduction
- Promise and vision
- Description and example
- Our Experiences
- Security
- Future Directions
- Conclusions

Technologies and Protocols

- Data Representation:
  - eXensible Markup Language (XML)
- Transport:
  - Simple Object Access Protocol (SOAP)
  - XML-RPC
- Discovery and publication
  - Web Service Description Language (WSDL)
  - Universal Description Discovery and Integration (UDDI)

XML

- Simplification of Generalized Markup
  - Tag/property based format similar to HTML
  - Also can express correct document semantics with a Document Type Definition (DTD) (unlike HTML)
- A lowest-common denominator format for "semi structured" data
  - E.g. not database relations/tables

XML Example

```
<?xml version="1.0"?>
<tag1>
   <!-- comment -->
   <tag2 var1=value1 var2=value2>
   Info can go here
   </tag2>
</tag1>
```

SOAP

- Simple Object Access Protocol
- Method to invoke remote methods
- General idea is a header followed by data
- Why not use HTTP GET and POST?
  - Not enough structure
- Is SOAP too complicated?

XML-RPC

- Same goals as SOAP
- Very straightforward way to map RPCs into XML objects
- Simpler than SOAP
  - Just datatypes and methods

WSDL

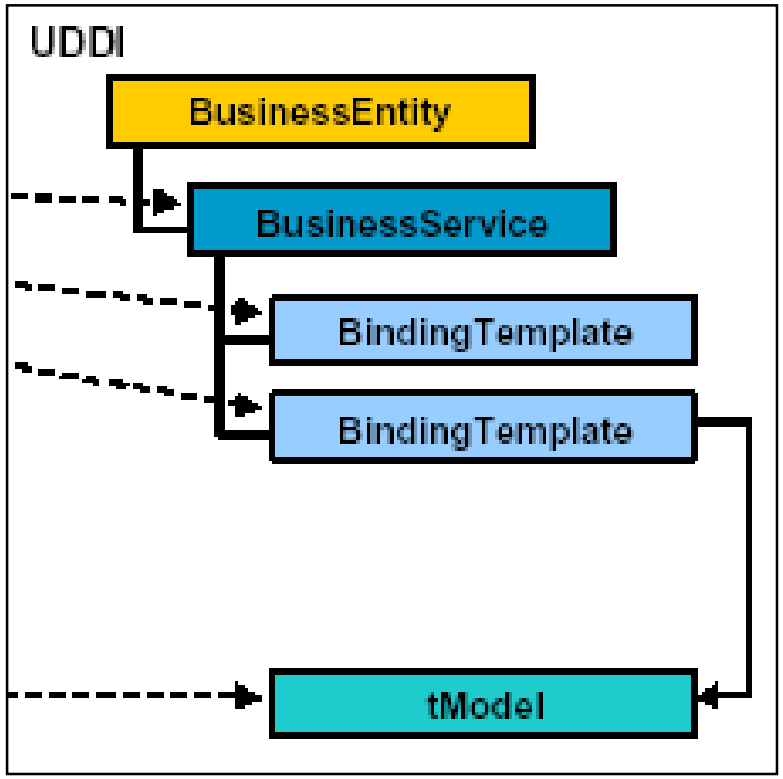
- Web Service Description Language
- Interface definition for a Web service
  - E.g. function signature
  - Similar function to header file in C, public interface/class definition in OO languages
  - Types of data which can be passed
  - “methods”
- Additional elements for:
  - What messages look like
  - Which transport to use
  - Where is the service

WSDL Structure

1. Type definitions,
   – used to describe the data being exchanged
2. Message definitions
   - What can be sent/exchanged
   - Can think of these as the methods
3. Operation definitions
   - Sets of messages involved in an exchange
4. Binding definitions
   - Map operations and types to actual transports
5. Service definition
   – defines the endpoint (URL) where the server can be found

UDDI

- Not necessary for WS
  - *White pages (business info)*
  - *Yellow pages (business categories following standard taxonomies)*
  - *Green pages (how to find services, pages, etc)*
- A bunch of browseable/searchable data structs running over SOAP implementing the above
  - Model can hold a WSDL like file



Google Example

- Google web service supports 3 functions
  - Search
  - Cached page lookup
  - Spelling suggestions
- We'll walk through some client code
  - Time permitting
    - WSDL definition
    - SOAP request and response

Example Google WS Client

```
public class GoogleAPIDemo {
public static void main(String[] args) {
String clientKey = args[0];
String directive = args[1];
String directiveArg = args[2];
// Create a Google Search object, set our authorization key
GoogleSearch s = new GoogleSearch();    // create the GS object
s.setKey(clientKey);  // Depending on user input, do search or cache query, then
                      // print out result
try {
   if (directive.equalsIgnoreCase("search")) {
      s.setQueryString(directiveArg);        // set the query
      GoogleSearchResult r = s.doSearch();    // calls the search engine
      System.out.println("Google Search Results:");
      System.out.println(r.toString());
   } else if (directive.equalsIgnoreCase("cached")) {
        … // cached page and spelling are similar to the search above
  } catch (GoogleSearchFault f) {
    System.out.println("The call to the Google Web APIs failed:");
   System.out.println(f.toString());
} }
```

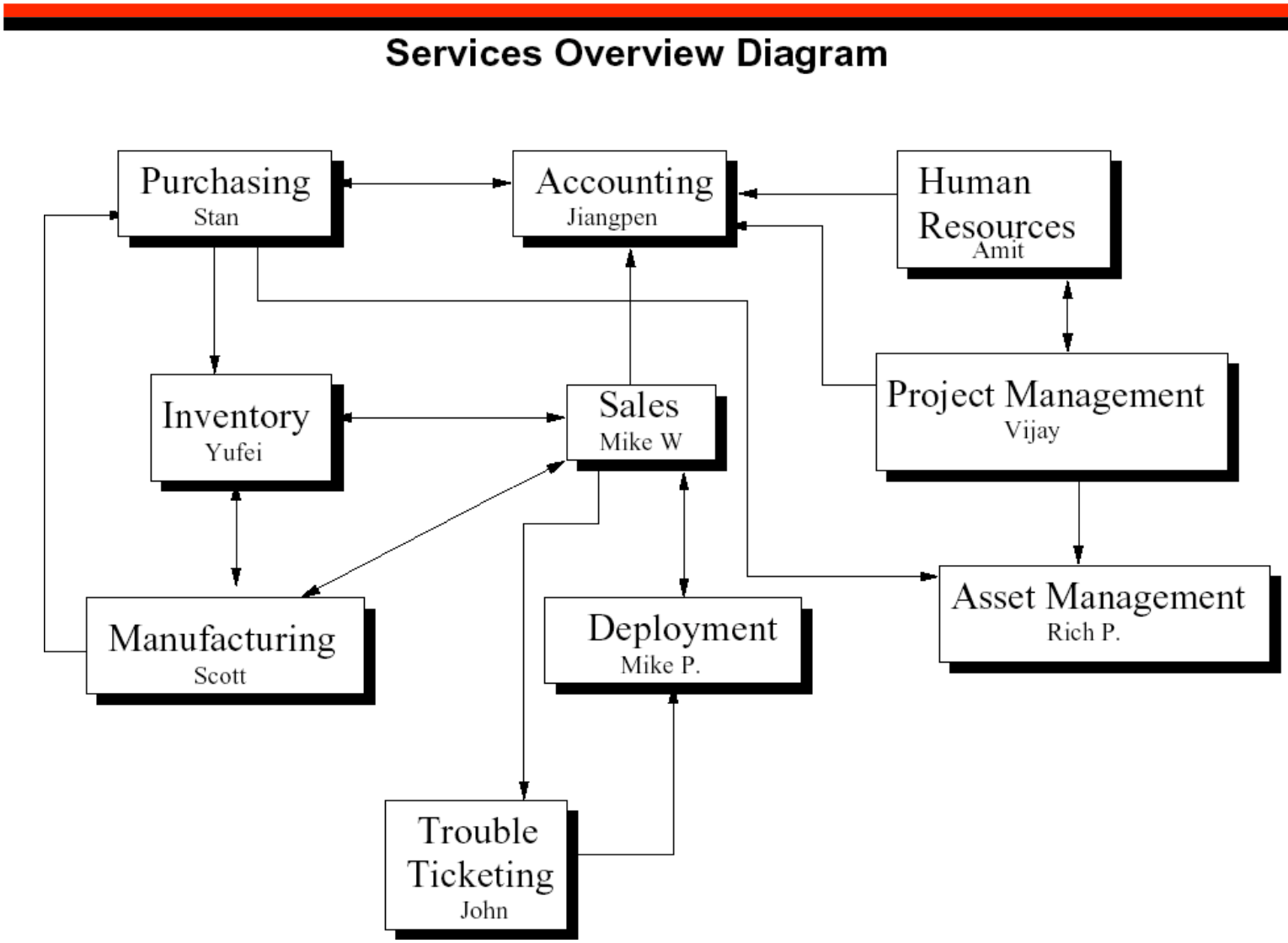
Outline

- Introduction
- Promise and vision
- Description and example
- Our Experiences
- Security
- Future Directions
- Conclusions

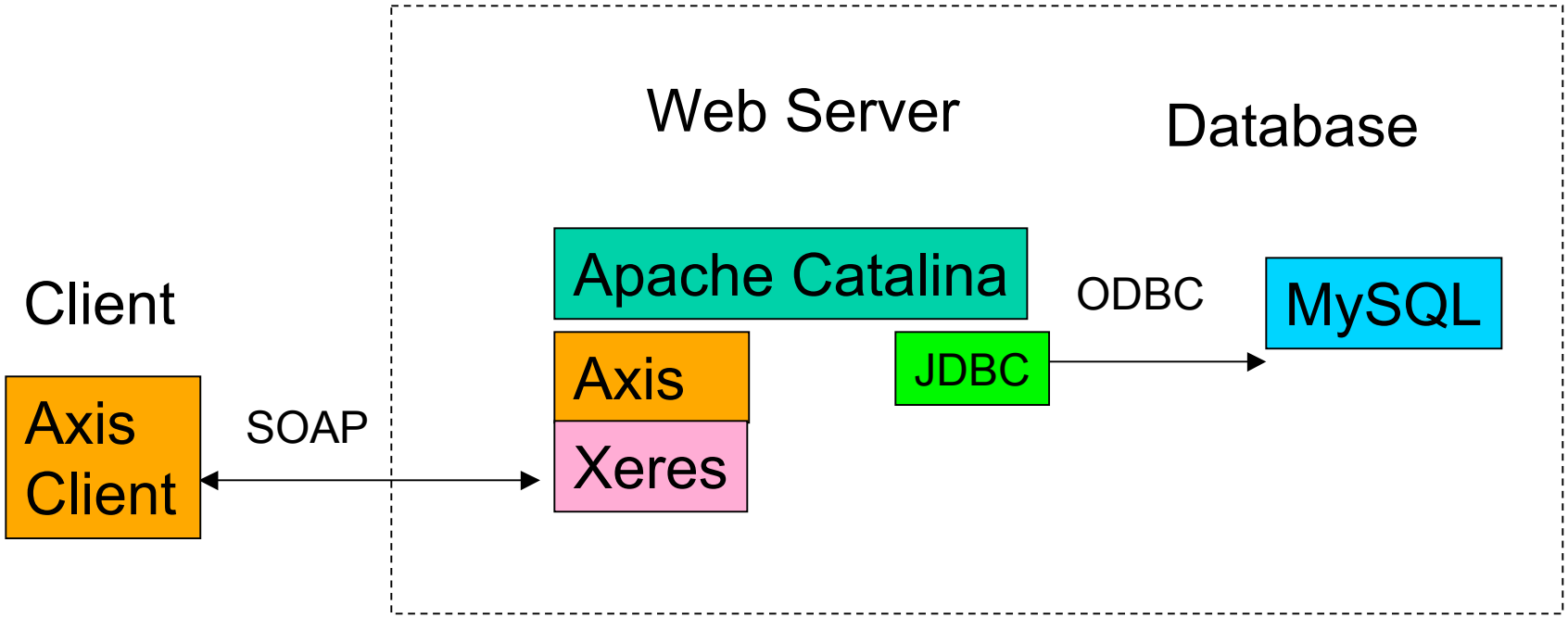
IT-only promise

- Interconnect Information Technology (IT) islands (vs. anything)
  - Typically a business application + databases
    - E.g. Manufacturing, accounting, HR
- Goals:
  - Each IT unit distinct, but calls other units as necessary
    - Contrast to monolithic application (SAP)
  - Business processes and reporting automated

Model Company IT Infrastructure



Typical WS Implementation



Class Experience

- Most students got WS up in 2-3 weeks
  - Windows, Linux and Solaris
  - All used apache/axis, some used access DB
- Network
  - Firewalls, wireless connections limit WS
  - Tried SSH tunnels, failed
- Several different styles of server
  - A “generic” client used Java reflection to implement a WS command interpreter!
- Consistency problem
  - Multiple versions of same data in different services
    - e.g. customers

Outline

- Introduction
- Promise and vision
- Description and example
- Our Experiences
- Security
- Future Directions
- Conclusions

Security Issues

- **Confidentiality**: can a 3rd party see it?
- **Authentication**: Who am I talking to?
- **Non-repudiation**: can you claim you didn't send it even if you really did?
- **Integrity**: was it altered before I got it?
- **Authorization**: Are you allowed to perform the action (method)?
- **Auditing**: what happened, when, by who?

WS security approaches

- SSL/HTTPS connection to Web Server
  - Pro: Simple, easy to add
  - Con: lose XML
- XML encryption, per element
  - Pro: keep XML structure, readability of “rest” of the document
- SOAP/XML based based encryption
  - Con: not standardized yet.

Likely Scenario

- SSL/HTTPS for transport
- Unwrapping layer inside server
  - Not quite here yet?
    - How do I get args to the methods, etc
- Use XML digital signatures for authentication
  - Distributed keys using emerging PKI
    - More likely files in emails…
- Ad-hoc logging for audit trails

Outline

- Introduction
- Promise and vision
- Description and example
- Our Experiences
- Security
- Future Directions
- Conclusions

Future Directions

- Automatic mapping, search and retrieval
  - Find all interconnected servers
- Automated consistency checking
  - Check DB tables for errors, consistency
- Fault tolerance and fail over
  - Multiple services at different sites

Conclusions

- Web Services are a promising technology
  - Can be used now as an interconnect
- Will need several years to grow
  - Will it move beyond a niche?
- Security, billing still unresolved

Backup Slides

- WSDL and XML examples

Google WSDL Example

```
<?xml version="1.0"?>

<definitions name="GoogleSearch"
             targetNamespace="urn:GoogleSearch"
             xmlns:typens="urn:GoogleSearch”
  <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"


. . .
```

Google WSDL Message part

```
<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>
```

•••

Google WSDL operations part

```
<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>
```

•••

Google WSDL binding part

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

SOAP Request Example

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

<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">shrdlu winograd maclisp teletype</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"></restrict>
      <safeSearch xsi:type="xsd:boolean">false</safeSearch>
      <lr xsi:type="xsd:string"></lr>
      <ie xsi:type="xsd:string">latin1</ie>
      <oe xsi:type="xsd:string">latin1</oe>
    </ns1:doGoogleSearch>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP Response Example

```
<?xml version='1.0' encoding='UTF-8'?>
<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: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]"></directoryCategories>
        <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>
            <hostName xsi:type="xsd:string"></hostName>
            <snippet xsi:type="xsd:string"> &lt;b&gt;...&lt;/b&gt; on a simple dialog (via
       &lt;b&gt;teletype&lt;/b&gt;) with a user, about a &lt;b&gt;...&lt;/b&gt;
       http://hci.stanford.edu/&lt;b&gt;winograd&lt;/b&gt;/&lt;b&gt;shrdlu&lt;/b&gt;&lt;br&gt; . It
       is written in &lt;b&gt;MacLisp&lt;/b&gt;, vintage 1970, and to
       &lt;b&gt;...&lt;/b&gt;</snippet>
        …
          </item>
          <item xsi:type="ns1:ResultElement">
```