Seminar on

Web services retrieval: URBE approach

Pierluigi PLEBANI

Dipartimento di Elettronica ed Informazione - Politecnico di Milano
plebani@elet.polimi.it
Before starting ...

- Web services retrieval is only the last problem
- We have had:
  - plumber retrieval
  - data retrieval
  - document retrieval
  - software component retrieval
  - ... and now Web service retrieval
... a look at the real world

- Once upon a time...
  - Friends of mine
  - Friends of friends of mine
  - ... (Friends of )\(^n\) mine with 1 ≤ n ≤ 6

- Advertising rules!
  - White pages
  - Yellow pages

- e-Advertising rules!
  - http://www.whitepages.com
  - http://www.yellowpages.com
Agenda

• All about Web services retrieval
• UDDI (Universal Descriptor Discovery and Integration)
• URBE (UDDI Registry By Example)
• Related work
• Concluding remarks
Who does retrieve Web services?

- Web services retrieval is one of the fundamental steps in SOA
- Final users need to retrieve Web services
- We need to consider Web service providers as well
What do we retrieve? 1/2

- We need to find a Web service (obviously)
- But, which one? The one:
  - able to perform what we need
  - accessible in a way we need
  - working in a way we need
What do we retrieve? 2/2

• A shared model for both Web service providers and Web service users is required

• This model must consider:
  - functionalities
  - conversation
  - quality

• Lot of specifications are available today:
  - WSDL
  - WS-CDL
  - WS-BPEL
  - WS-Policy
  - ... and many others
Where do we retrieve Web services?

- All the information should be collected and stored in well known places:
  - centralized solution
  - distributed or peer-to-peer solution

- Who has the ownership on this information?
  - registry
  - repository
When do we retrieve Web services?

- At design-time
  - we can code the client-side

- At deployment-time
  - we need a declarative model

- At run-time
  - we need... something
Why do we retrieve Web services?

- Only for a single invocation
- For building a partnership
- As a part of my application
- As the whole application
How do we retrieve Web services?

- (Friends of )$^n$ mine with $1 \leq n \leq 6$
- Browsing the Web (XMethods, SALCentral (?) )
- Googling
- White pages
- Yellow pages
Service registries

- Web service registries support:
  - the publication of new Web services
  - the retrieve of desired Web services

- Some tools:
  - directory: LDAP
  - basic registry: RMI, UDDI
  - advanced registry: ebXML
UDDI
(Universal Description Discovery and Integration)

Seminar on “Web services retrieval: URBE approach”
Pierluigi Plebani
May, 3rd, 2007 - University of Trento
• OASIS standard (v. 3)
  ‣ Formerly a joint proposal of (IBM, SAP, BEA, ...)

• UDDI Service discovery is driven by:
  • Keyword-based query
  • Pre-defined taxonomies browsing
    • UNSPSC
    • ISO 3166
    • NTIS - NAICS

• UDDI supports publication of generic services, not necessarily Web services
UDDI Publication model

- A service provider or a group of them agree on service specifications:
  - what the service does
  - how it can be used
- Everyone now can offers a service according to such specifications
- Both service specifications and service instances must be accessible
The UDDI acronym

- **Universal Description**: UDDI does not rely on a specific approach for describe a service (WSDL is only one of them)

- **Universal Discovery**: service retrieval can be performed in several ways
  - white pages: by service provider
  - yellow pages: by service classification
  - green pages: by service type

- **Universal Integration**: services are described regardless of the underlying technologies
UDDI architecture

Interface for browsing both registries

API

UDDI Business Registry

- Business Registry
- Service Type Registry

Specification of real services
Specification of a class of services
UDDI Data Model

from C. von Riegen (ed), UDDI Version 2.03 Data Structure Reference
http://uddi.org/pubs/DataStructure-V2.03-Published-20020719.htm, 2002
UDDI query model

- Discovery process is mainly performed manually
  - browsing one of the available taxonomy
  - using keywords
- I need to exactly know how information are organized in the registry
URBE (Uddi Registry By Example)

Seminar on “Web services retrieval: URBE approach”
Pierluigi Plebani
May, 3rd, 2007 - University of Trento
Main features

- Interface matching
- Semantic matching
- Quality driven matching
- So far:
  - we have deeply studied the first point
  - we are going to validate the second and third point
- The main goal is: *retrieval for substitutability*
URBE

- Uddi Registry By Example
  - is compliant with UDDI (publishing, searching, data models)
  - performs content based query based
    - user submits a WSDL expressing the requirements
    - URBE returns a list of Web services close to the request
- Similarity function fSim is the core of URBE
  - semantic analysis
  - syntactic analysis

\[
\text{fSim}(\sigma_q, \sigma_p) = w_{\text{NamingSim}} \times n_{\text{Sim}}(\sigma_q, \sigma_p) + w_{\text{StructSim}} \times s_{\text{Sim}}(\sigma_q, \sigma_p)
\]
fSim in detail

- **wNamingSim, wStructSim**
  - weights the influence of semantic against to the structural similarity
  - allow tuning the function
  - influence the nature of result
- **fSim: \((\sigma_q, \sigma_p) \rightarrow [0..1]\)**
  - \(fSim(\sigma_q, \sigma_q) = 1\)
  - fSim is not symmetric
  - fSim relies on a linear programming model
• Assignment in bipartite graphs which compares:
  - terms, operation, services

\[
\text{opt}(\text{sim}(Q, P)) = \frac{1.0+0.7+1.0}{3} = 2.7 / 3 = 0.9
\]
fSim components

- **Semantic analysis:**
  - portType names
  - operations names
  - parameters names

\[ nSim(\sigma_q, \sigma_p) = w_{Step} \cdot nameSim(\sigma_q.name, \sigma_p.name) + (1 - w_{Step}) \cdot opt(opNSim(\sigma_q.op_i, \sigma_p.op_j)) \]

- **Structural analysis:**
  - data types
  - number of inputs
  - number of outputs
  - number of operations

\[ sSim(\sigma_q, \sigma_p) = w_{DataType} \cdot opt(opSSim(\sigma_q.op_i, \sigma_p.op_j)) + (1 - w_{DataType}) \cdot \min \left( \frac{|\sigma_q.op_i|}{|\sigma_p.op_j|}, \frac{|\sigma_p.op_j|}{|\sigma_q.op_i|} \right) \]

- **Two core elements:** Name and DataType similarity
Names similarity 1/2

• Quantifies how much two names are related
  car ←→ automobile
  money ←→ currency

• We need that to compare service, operation, and parameter names

• Our approach relies on two ontologies:
  ‣ domain specific ontology (few terms higher precision)
  ‣ general purpose ontology (more terms lower precision)

• Names similarity is based on path length between names in the ontology
Name similarity 2/2

- We assume that the WSDL is automatically generated
- Names reflect coding conventions
- Stemming and tokenization are required before comparing names
  - getData, currencyExchange
- Some terms have less meaning
  - body, result, parameters

<table>
<thead>
<tr>
<th>Rule</th>
<th>Original term</th>
<th>Tokenized version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case change</td>
<td>currencyExchange</td>
<td>currency, exchange</td>
</tr>
<tr>
<td>Case change</td>
<td>SendSMSTo</td>
<td>send, sms, to</td>
</tr>
<tr>
<td>Suffix numbers elimination</td>
<td>currency1</td>
<td>currency</td>
</tr>
<tr>
<td>Underscore separator</td>
<td>currency_exchange</td>
<td>currency, exchange</td>
</tr>
</tbody>
</table>

Roughly speaking, the names similarity depends on the maximum similarity among the terms composing the given names.
DataType similarity

• Inspired by Stroulia and Yang

<table>
<thead>
<tr>
<th>Group</th>
<th>Simple Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer group</td>
<td>int, integer, byte, short, long</td>
</tr>
<tr>
<td>Real group</td>
<td>real, float, double, decimal</td>
</tr>
<tr>
<td>String group</td>
<td>char, string, arrayofchar</td>
</tr>
<tr>
<td>Boolean group</td>
<td>boolean</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>typeSim</th>
<th>Integer</th>
<th>Real</th>
<th>String</th>
<th>Boolean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>1</td>
<td>0.8</td>
<td>0</td>
<td>0.1</td>
</tr>
<tr>
<td>Real</td>
<td>0.5</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>String</td>
<td>0.3</td>
<td>0.3</td>
<td>1</td>
<td>0.3</td>
</tr>
<tr>
<td>Boolean</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

• Table reflects the casting among different data types in programming language
Benchmark

- 35 WSDLs, 64 descriptors divided in 5 groups
  - Group A: Currency (4 WSDLs, 9 portTypes)
  - Group B: DNA (5 WSDLs, 5 portTypes)
  - Group C: SMS (12 WSDLs, 18 portTypes)
  - Group D: Weather (6 WSDLs, 18 portTypes)
  - Group E: ZIP (8 WSDLs, 14 portTypes)

- Machine
  - IBM xSeries, 2 CPU Intel XEON 3GHz, 2 GByte RAM

- For each group a member is selected as query
  - result always contains at least one item with fSim = 1
Average precision

Precision for different groups

Threshold

Precision

A
B
C
D
E
General recall

Recall for different groups

Threshold

Recall

A
B
C
D
E
Semantic vs Structural precision

**Precision with different weights**

![Graph showing precision with different weights for Names and Structure. The graph includes lines for different weights: 0%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, and 90%. Each line represents a different name or year, and the threshold ranges from 0.10 to 0.90.](image-url)
Semantic vs Structural recall

Recall with different weights

Threshold

Recall

Names
- 1-0
- 07-03
- 05-05
- 03-07
- 0-1

Structure
Groups B & E: precision & recall

Group B

Group E

Precision vs. Threshold

Recall vs. Threshold

Names: 1-0, 07-03, 05-05, 03-07, 0-1

Structure
Exploiting the annotations

- Recall can be improved if SAWSDL description is available
- In this case name similarity is based on the annotations
  - Annotation refers to concept in the domain-specific ontology
  - Similarity evaluation relies on path length
- Annotations similarity results
  - more precise than names similarity
  - more fast to calculate
- Annotation mainly influences the recall of fSim
About the quality

Class of users
Quality Tree

Domain application community
produce

read

User
produce

Quality Tree

User policy

Quality Tree

read

Provider

Service policy

produce
Quality tree

Quality of service

Video quality
- Resolution: {320*200; 800*600; 1024*768}
- Framerate: [10..30]
- Colordepth: {8; 16; 24; 32}

Sound quality
- Price: [0..∞]
- Encoding: {MP3; WMA; ALAC}
- Sampling: [64..192]
Related work

Seminar on “Web services retrieval: URBE approach”
Pierluigi Plebani
May, 3rd, 2007 - University of Trento
About the architecture

- Web service registries support:
  - the publication of new Web services
  - the retrieve of desired Web services

- Some tools:
  - directory: LDAP
  - basic registry: RMI, UDDI
  - advanced registry: ebXML
About the matchmaking

- Interface matching
  - Zaremski and Wang (Software components)
  - Stroulieva and Yang, Woogle (WSDL)

- Semantic matching
  - OWL-S MM, WSMO MM

- Hybrid matching
  - Lumina (SAWSDL)

- Quality driven matching
  - WSOI (WSOL), UDDle (Proprietary Language)

- What about behavior?
Concluding remarks

Seminar on “Web services retrieval: URBE approach”
Pierluigi Plebani
May, 3rd, 2007 - University of Trento
Conclusions 1/2

- **URBE:**
  - allows content-based query
  - is compliant with UDDI

- **Why use URBE**
  - to retrieve services
  - to support BPEL processes design (top-down approach)
  - to substitute services (autonomic computing, grid computing)

- **Try it:** [http://black.elet.polimi.it/urbeClient](http://black.elet.polimi.it/urbeClient)

- **Download it:** [http://black.elet.polimi.it/urbe](http://black.elet.polimi.it/urbe)
Conclusions 2/2

- Quality matching represents, at this stage, the biggest open issue
- Web service registry managing should be deeply investigated as well
- Semantic based approaches suffer of the need of services semantically described
- Web services retrieval must be, first of all, usable!