Prison Break: A Generic Schema Matching Solution to the Cloud Vendor Lock-in Problem

Mohammad Hamdaqa, Ladan Tahvildari
University of Waterloo
Waterloo, Ontario, Canada
mhamdaqa@uwaterloo.ca

November 03, 2014
Prison Break: A Generic Schema Matching Solution to the Cloud Vendor Lock-in Problem

Teaser Video

https://www.youtube.com/watch?v=QhdaeGlb7pU
Outline

• Motivation
  – The Vendor Lock-in Problem
  – Schema Matching Role and Shortcomings

• The Prison Break Approach
  – Web-Similarity Matcher
  – Concept Matching Linguistic vs. Web Similarity
  – Realization

• Case Study

• Concluding Remarks
Motivations (1/3)

Cloud Computing Inhibitors

2014 Future of Cloud Computing  4th Annual Survey Results
Motivations (2/3)
The Vendor Lock-in Problem

- Dependency of an application on a particular cloud vendor that prevents porting the application from one vendor to another
- Porting applications involves data, services, and binaries
- Research Focus: How to deploy an application on multiple cloud platforms
Motivations (3/3)
Examples of Deployment Artifacts

Windows Azure

```
<WorkerRole name="ShoppingCartProcessing" vmsize="Small">  
  <ConfigurationSettings>  
    <Setting name="DataConnection" />  
  </ConfigurationSettings>  
</WorkerRole>  
</ServiceDefinition>
```

Listing 1: Example of Azure Service Definition File.

```
<ServiceConfiguration>
  <Role name="ShoppingCartProcessing">  
    <Instances count="2" />  
    <ConfigurationSettings>  
      <Setting name="DataConnection" value="UseDevelopmentStorage=true" />  
    </ConfigurationSettings>  
  </Role>
</ServiceConfiguration>
```

Listing 2: Example of Azure Service Configuration File.

Google App Engine

```
<application>
  <description>Demo Java EE App</description>
  <display-name>My App</display-name>
  <module>
    <web>
      <web-uri>ShoppingCartProcessing</web-uri>
      <context-root>ShoppingCartProcessing</context-root>
    </web>
  </module>
</application>
```

Listing 3: Example of GAE Deployment Descriptor File.

```
<appengine-web-app>
  <application>My App</application>
  <module>ShoppingCartProcessing</module>
  <version>uno</version>
  <threadsafe>true</threadsafe>
  <instance-class>F4</instance-class>
  <manual-scaling>
    <instances>2</instances>
  </manual-scaling>
</appengine-web-app>
```

Listing 4: Example of GAE Module Configuration File.
The information required to specify a cloud application deployment is essentially the same.

The mismatch is usually related to the concepts and structure of the required artifacts.
The Role of the Schema Matching in Models Migration

<table>
<thead>
<tr>
<th>Platonic Standardization</th>
<th>Oligopoly</th>
<th>Modest Hybrid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>Process</td>
<td>Model</td>
</tr>
<tr>
<td>Meta-Model</td>
<td>Data Flow</td>
<td>Transformation</td>
</tr>
</tbody>
</table>

Schema Matching

Model Transformation

All Providers Specifications

Provider X Specifications

Provider Y Specifications

Mismatch
Schema Matching Shortcomings

- Schema matching is **harder** when concepts are **vague** and **ill-defined**
- Cloud schemas concepts do not **share linguistic or structural similarity**
  - lack of **coordination** between the providers
  - continuous **updating** of schemas
  - marketing campaigns
- Purely linguistic semantic similarity techniques **fail** in matching elements across cloud schemas.

<table>
<thead>
<tr>
<th></th>
<th>Azure</th>
<th>GAE</th>
<th>AWS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terms</td>
<td>Role</td>
<td>Module</td>
<td>Beanstalk</td>
</tr>
<tr>
<td>Azure</td>
<td>Role</td>
<td>1.0</td>
<td>0.12</td>
</tr>
<tr>
<td>GAE</td>
<td>Module</td>
<td>0.12</td>
<td>1.0</td>
</tr>
<tr>
<td>AWS</td>
<td>Beanstalk</td>
<td>0.07</td>
<td>0.1</td>
</tr>
</tbody>
</table>

**Path length similarity**
- A simple node-counting scheme (path).
- The **relatedness** score is inversely proportional to the number of nodes along the shortest **path** between the synsets.
The Research Challenges

• Problem Statement
  Schema matching typically relies on linguistic and structural similarities, and cloud schema terms diverge so much that such matching is impossible. There is a need to incorporating domain knowledge in order to disambiguate the similar schema concepts.

• Research Questions
  – RQ1. Is it possible to pull domain knowledge into schema matching processes without sacrificing generality?
  – RQ2. How will this generic approach to schema matching perform in comparison to existing approaches?
Platform Independent Meta-Model (e.g., StratusML MM)

Platform Specific Meta-Model (e.g., Azure Schema)

Platform Specific Models (e.g., Azure Config.)

Platform Independent Models

Model Migration

Matching Recommendations

Selection

Instance Model

Meta-Model

Process

Decision Maker

Data Flow

Model Driven Reverse Engineering

Step 1: Element-Based Similarity Computation

Step 2: Structure-Based Similarity Computation

Elements Mapping

The Prison Break: A Novel Schema Matching Solution
Web Similarity Matcher (1/2)
The Normalized Google Distance

- Element Schema Matching
- Based on the Normalized Google Distance (NGD)
  - Measure concepts relatedness based on search engine results
  - Measure distance not similarity (zero $\rightarrow$ infinity)

$$\text{NGD}(\tau_1, \tau_2) = \frac{\max\{\log f(\tau_1), \log f(\tau_2)\} - \log f(\tau_1, \tau_2)}{\log N - \min\{\log f(\tau_1), \log f(\tau_2)\}}$$

$N$: Total number of pages in Google index $\sim$ 30 trillion.
Web Similarity Matcher (WSM) (2/2)

- To use the NGD as a semantic similarity measure:
  - Convert distance to similarity
  - Normalize unbounded results
  - Generalize NGD to any search engine (SE) \( \rightarrow \mathbf{NWD}_{SE} \)
  - (N) different for different search engines \( \rightarrow \) Using a common term (e.g., “the”), assume certain frequency results/page, we find N

\[
\begin{align*}
WSM_{SE}(\tau_1, \tau_2) &= e^{-\mathbf{NWD}_{SE}(\tau_1, \tau_2)} \\
(g \circ \text{NGD}) : [0, \infty) &\rightarrow (0, 1) \\
g(0) &= 1 \\
\lim_{x \rightarrow \infty} g(x) &= 0 \\
g(x) \text{ is decreasing, } \\
\text{if } x_1 > x_2 \text{ then } g(x_1) < g(x_2)
\end{align*}
\]
## Concept Matching
Linguistic vs. Web Similarity

<table>
<thead>
<tr>
<th>Matching Technique</th>
<th>Terms</th>
<th>Role</th>
<th>Module</th>
<th>Beanstalk</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Path Length</strong></td>
<td>Role</td>
<td>1.0</td>
<td>0.12</td>
<td>0.07</td>
</tr>
<tr>
<td></td>
<td>Module</td>
<td>0.12</td>
<td>1.0</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>Beanstalk</td>
<td>0.07</td>
<td>0.1</td>
<td>1.0</td>
</tr>
<tr>
<td><strong>Matching Technique</strong></td>
<td>Terms</td>
<td>Role</td>
<td>Module</td>
<td>Beanstalk</td>
</tr>
<tr>
<td><strong>Web Similarity</strong></td>
<td>Role</td>
<td>1.0</td>
<td>0.75</td>
<td>0.78</td>
</tr>
<tr>
<td></td>
<td>Module</td>
<td>0.75</td>
<td>1.0</td>
<td>0.89</td>
</tr>
<tr>
<td></td>
<td>Beanstalk</td>
<td>0.78</td>
<td>0.89</td>
<td>1.0</td>
</tr>
</tbody>
</table>
The Prison Break Realization

- Realized as an extension to the OpenII Harmony framework
- **OpenII Harmony**: open source schema matching tool
  - Extensibility
  - Multiple Importers
  - Multi-Format Exporter
  - High-end GUI
  - Composite: allow combining and comparing the results of different matchers.

- **Design Decisions**
  - Response time of online requests
    - $3 \times n^2$ vs. $n \times (n + 2)$
  - Number of API requests allowed by the search engine provider
    - Multi license key
    - Multi search engines
Revisiting Research Questions

• Research Questions

  – **RQ1.** *Is it possible to pull domain knowledge into schema matching processes without sacrificing generality?*

    **Generality**

  – **RQ2.** *How will this generic approach to schema matching perform in comparison to existing approaches?*

    **Performance**
Evaluation – RQ1

• Evaluation criteria: **evaluate input constraints.**

• **Hypothesis:**

  “The more generic the approach, the fewer constraints it imposes on its input”

• Linguistic approaches **require** extra sources of knowledge
  – It requires the terms to have some kind of linguistic similarity
    • More advanced techniques requires: Thesauri, acronym lists, dictionaries, and mismatch lists.

• Prison Break **does not impose these constraints nor require** such information.
  – Search engine results are obtained automatically at runtime

• ➔ **Prison Break is a more generic approach**
Evaluation – RQ2

- **Evaluation criteria:** Quality of results
- **Goal:** Compare Prison Break with schema matching techniques implemented within OpenII
- **Experiment Setup:**
  - Schemas from two providers (Microsoft and Google):
    - Azure definition and configuration schemas vs. GAE “appengine-web” schema
    - No additional domain information
  - Prison Break WSM_{Bing} vs. Name Similarity vs. WordNet
  - Time between requests 0.25 sec / avoid DoS
  - Filtered the results based on different evidence threshold
  - Compared with a set of manually created reference alignments
Evaluation – RQ2
Quality of Matching

Precision

- Prison Break Approach
- Wordnet
- Name Similarity

Recall

- Prison Break Approach
- Wordnet
- Name Similarity

Evident threshold

Precision Value

Recall Value
Evaluation – RQ2
Quality of Matching

F-Measure

- Prison Break Approach
- Wordnet
- Name Similarity

F-Measure Value

Evident threshold
Drawbacks

• Execution time
  – Pure linguistic approaches are measured in seconds \(\sim(< 60)\).
  – Prison Break approach is measured in minutes \(\sim(10\text{-}30)\).
Wrap Up

Cloud Vendor Lock-in

Model Transformation

Elements Mapping

Schema Matching

Domain Knowledge

Web Similarity Metrics

Prison Break
Conclusions

• Summary:
  – Despite its potential, schema matching has never been well exploited to address vendor lock-in problem
    • Cloud schemas imposes new challenges on the existing schema matching approaches
  – Prison Break approach uses schema matching techniques based on web-similarity
    • It contribute to solving the vendor lock-in problem, by reducing the manual efforts needed for creating a pivot meta-model and keep it up-to-date as schemas evolve.

• Ongoing Works:
  – To improve the implementation and apply the approach to wider range of cloud schemas
  – To investigate using structural matching techniques and generalize it for the bigger problem of model driven migration on the fly.
Prison Break: A Generic Schema Matching Solution to the Cloud Vendor Lock-in Problem

Mohammad Hamdaqa, Ladan Tahvildari
University of Waterloo
Waterloo, Ontario, Canada
mhamdaqa@uwaterloo.ca

November 03, 2014