OpenKnowledge

FP6-027253

Implementation of the Ontology Matching Component

Coordinator: Pavel Shvaiko[1]

with contributions from

Fausto Giunchiglia[1], Mikalai Yatskevich[1], Juan Pane[1], and Paolo Besana[2]

[1] Dept of Information and Communication Technology, University of Trento, Italy

[2] School of Informatics, University of Edinburgh, UK

Report Version: final

Report Preparation Date: 14.12.2007

Classification: deliverable D3.6

Contract Start Date: 1.1.2006 Duration: 36 months

Project Co-ordinator: University of Edinburgh (David Robertson)

Partners: IIIA(CSIC) Barcelona

Vrije Universiteit Amsterdam

University of Edinburgh

KMI, Open University

University of Southampton

University of Trento

OpenKnowledge

FP6-027253

Implementation of the ontology matching component [1]

Coordinator: Pavel Shvaiko[1]
with contributions from
Fausto Giunchiglia[1], Mikalai Yatskevich[1], Juan Pane[1], and Paolo Besana[2]

[1] Department of Information and Communication Technology (DIT),
University of Trento, Povo, Trento, Italy
{pavel|fausto|yatskevi|pane}@dit.unitn.it
[2] The University of Edinburgh, Edinburgh, UK
p.besana@ed.ac.uk

Report Version: final
Report Preparation Date: 14.12.2007
Classification: deliverable 3.6
Contract Start Date: 1.1.2006 Duration: 36 months
Project Co-ordinator: University of Edinburgh (David Robertson)
Partners: IIIA(CSIC) Barcelona
Vrije Universiteit Amsterdam
University of Edinburgh
KMI, Open University
University of Southampton
University of Trento

[1]The originally planned title of this deliverable as from the project proposal was "Implementation of composite mapping engine". However, this new title better reflects the actual implementation and needs of the project, and therefore, is used here.

Abstract

This deliverable provides a brief documentation for the ontology matching component implementation. Specifically, it discusses (*i*) the purpose and functionality of the component, (*ii*) its usage example, and finally (*iii*) plans for its future development.

1 Purpose and functionality

The purpose of the ontology matching component is to reduce the semantic heterogeneity in peer role descriptions [7, 2], formalized as lightweight coordination calculus (LCC) constraints [5]. The heterogeneity is reduced in two steps: (*i*) match the constraints to determine correspondences and (*ii*) execute correspondences according to an application needs, such as query answering. In this deliverable we focus only on the first, i.e., matching step, while the query answering step is discussed in [6].

The ontology matching component takes LCC constraints and functionalities in the OpenKnowledge components (OKCs) [1], for example, Get_Wine(Region, Country, Color, Price) and Get_Wine(Region(Country, Area), Colour, Cost, Year), and returns a global similarity coefficient in the [0 1] range between these constraints (e.g., 0.57) as well as the set of one-to-one correspondences between the semantically related elements of these constraints (e.g., that Color in the first description corresponds to Colour in the second one). The following two structural properties are preserved: (*i*) functions are matched to functions and (*ii*) variables are matched to variables.

2 Usage example

The following web site `http://www.few.vu.nl/OK/wiki/` provides the OpenKnowledge (OK) client installation guidelines, while the source code is available at the project revision (subversion) control system - SVN: `http://fountain.ecs.soton.ac.uk/ok/repos`[2]. Here we provide only a usage example for the ontology matching component, located at openK>src>org.openk.core>module>matcher within the SVN project. The ontology matching component uses the S-Match library (also available under SVN), specifically its label and node matchers [4]. Let us discuss a simplified usage example, which is shown in Figure 1.

[2]Authorization required, contact David Dupplaw (dpd@ecs.soton.ac.uk) for an account set up.

```
package org.openk.core.module.matcher.impl;
import java.util.Properties;
...

public class Example{
  public static void main(String[] args) {
    DefaultMatchingComponent mc = new DefaultMatchingComponent();
    mc.init();
    exampleLCC(mc);
 }

public static void exampleLCC(DefaultMatchingComponent mc){
  Properties p = new Properties();
  p.put(Matcher.THRESHOLD_VALUE, "0.55");

  matchLCC("get_wine(region, country, color, price, amount)",
           "get_wine(region(country, area), colour, cost, year, quantity)",
           mc, p);
}

public static void matchLCC(String source, String target,
                            DefaultMatchingComponent mc, Properties p){
  TreeMapping tm = mc.treeMatch(null, source, target,
                                Matcher.STRUCTURE_TYPE.LCC_CONSTRAINT, p);
  ...
}
```

Figure 1: Usage example.

In particular, DefaultMatchingComponent implements the Matcher interface and provides the basic functionality of the matching component. It converts the input constraints into trees and performs the structure preserving semantic matching as described in [3]. The Properties parameter contains the matcher specific information, such as threshold values to use. THRESHOLD_VALUE specifies an experimentally established threshold (0.55) above which the constraints are considered as globally similar and dissimilar otherwise. Get_Wine(Region, Country, Color, Price) and Get_Wine(Region(Country, Area), Colour, Cost, Year) are the two constraints to be matched by the matchLCC function.

The result of running this example is shown in Figure 2. Initially, the two input constraints (SOURCE and TARGET) as well as the global similarity (SIM) between them are reported. Then, the set of correspondences that hold between the elements of the input constraints is shown, starting from the root nodes. Statements in curly brackets (e.g., {EQUIVALENT | 1.0}) express the relation holding between the entities under consideration and the confidence in the [0 1] range that this

relation holds. Numbers in square brackets (e.g., [0]) are used to index the elements of the constraints, which are further exploited during the query answering phase, see [6].

```
SOURCE: get_wine(region, country, color, price)
TARGET: get_wine(region(country,area), colour, cost, year)
SIM: 0.5714285714285714

//get_wine <-> //get_wine {EQUIVALENT | 1.0}
/get_wine/region[0] <-> /get_wine/region[0]/area[1] {EQUIVALENT | 1.0}
/get_wine/country[1] <-> /get_wine/region[0]/country[0] {EQUIVALENT | 1.0}
/get_wine/color[2] <-> /get_wine/colour[1] {EQUIVALENT | 1.0}
/get_wine/price[3] <-> /get_wine/cost[2] {EQUIVALENT | 1.0}
```

Figure 2: The usage example result.

The execution of this example (on a standard laptop: 2Ghz, 2Gb RAM) took 1292 ms., out of which initialization of the matching component (mc.init) required 1077 ms., while the matching operation was performed in 41 ms.

3 Future work

Future work proceeds at least along the following directions: (*i*) making implementation of the ontology matching component robust and (*ii*) smooth integration of the component into the OK system.

References

[1] David Dupplaw, Uladzimir Kharkevich, Spyros Kotoulas, Adrian Perreau de Pinninck, Ronny Siebes, and Chris Walton. *OpenKnowledge Deliverable 2.1: Architecting Open Knowledge*. `http://www.cisa.informatics.ed.ac.uk/OK/Deliverables/D2.1a.pdf`, 2006.

[2] Fausto Giunchiglia, Fiona McNeill, Mikalai Yatskevich, Zharko Alekovski, Alan Bundy, Frank van Harmelen, Spyros Kotoulas, Vanessa Lopez, Marta Sabou, Ronny Siebes, and Annette ten Tejie. *OpenKnowledge Deliverable 4.1: Approximate Semantic Tree Matching in OpenKnowledge*. `http://www.cisa.informatics.ed.ac.uk/OK/Deliverables/D4.1.pdf`, 2006.

[3] Fausto Giunchiglia, Mikalai Yatskevich, and Fiona McNeill. Structure preserving semantic matching. In *Proceedings of the ISWC+ASWC International workshop on Ontology Matching (OM)*, pages 13–24, Busan (KR), 2007.

[4] Fausto Giunchiglia, Mikalai Yatskevich, and Pavel Shvaiko. Semantic matching: Algorithms and implementation. *Journal on Data Semantics*, IX:1–38, 2007.

[5] Sindhu Joseph, Adrian Perreau de Pinninck, Dave Robertson, Carles Sierra, and Chris Walton. *OpenKnowledge Deliverable 1.1: Interaction Model Language Definition.* `http://www.cisa.informatics.ed.ac.uk/OK/Deliverables/D1.1.pdf`, 2006.

[6] Pavel Shvaiko, Fausto Giunchiglia, Juan Pane, and Paolo Besana. *OpenKnowledge Deliverable 4.3: Plug-in component supporting query answering.* `http://www.cisa.informatics.ed.ac.uk/OK/Deliverables/D4.3.pdf`, 2007.

[7] Mikalai Yatskevich, Fausto Giunchiglia, Fiona McNeill, and Pavel Shvaiko. *OpenKnowledge Deliverable 3.4: Specification of ontology matching component.* `http://www.cisa.informatics.ed.ac.uk/OK/Deliverables/D3.4.pdf`, 2007.