GOOD ENOUGH ANSWER PLUG-IN COMPONENT

Coordinator: Pavel Shvaiko

with contributions from: Fausto Giunchiglia, Juan Pane and Paolo Besana

January 2008

Technical Report # DISI-08-004

OpenKnowledge

FP6-027253

Good enough answer plug-in component

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

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

Report Version: final
Report Preparation Date: 23.12.2007
Classification: deliverable 4.6
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

Abstract

This deliverable provides a brief documentation for the good enough answer (GEA) plug-in 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 combination of an interaction model and the peers assigned to each of its roles is called a *configuration*. The purpose of the good enough answer mechanism is to find good enough configurations that achieve the purposes of the peers with a reasonable investment of resources in their construction [4]. Specifically, given a set of peers the GEA component returns the best peer(s) for a role r in an interaction model IM [5]. It does this with the aid of two heuristic scores, namely:

- The matching score between the interaction model roles and the peers' capabilities [1, 3]. It is specified in the [0 1] range.
- Some score of trust in the peers, based on their historical behavior [2]. It is specified in the [0 1] range.

The matching and the trust scores are then combined into a single GEA score in the [0 1] range. Finally, based on the GEA scores, it is decided (e.g., by choosing top-1 peer) whether the peer is considered to be good enough or not for a given role in a given interaction model.

2 Usage example

The following web site `http://www.few.vu.nl/OK/wiki/` provides the Open-Knowledge (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`[1]. Here we provide only a usage example for the GEA component, located at openK>src>org.openk.core.OKC.impl within the SVN project. Let us discuss a simplified usage example, which is shown in Figure 1.

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

```
package org.openk.core.OKC.impl;

import java.util.LinkedList;
...
public class GEATest {
    public GEATest(int peerQty){
        this.peersQty = peerQty;
    }

    /** Define the number of peers to create */
    private int peersQty = 10;

    public static void main(String[] args) {
        PeerSelectionStrategy geaThreshold = new GEAThresholdSelStrategyImpl();
        PeerSelectionStrategy geaInterval = new GEAIntervalSelStrategyImpl();

        //create a test with 20 peers
        GEATest test = new GEATest(20);
        List<SubscriptionSpec> generatedPeers = test.createPeers();
        test.printPeers(generatedPeers,"Generated peers");

        List<SubscriptionSpec> selectedT = geaThreshold.selectPeer(generatedPeers);
        test.printPeers(selectedT, "selected by GEAThresholdSelStrategyImpl");

        List<SubscriptionSpec> selectedI = geaInterval.selectPeer(generatedPeers);
        test.printPeers(selectedI, "selected by GEAIntervalSelStrategyImpl");
    }
    ...
}
```

Figure 1: Usage example.

In particular, both GEAThresholdSelStrategyImpl and GEAIntervalSelStrategyImpl implement the PeerSelectionStrategy interface [5]. Specifically, the first class implements the thresholds & weighted sum strategy, while the second one implements an interval-based approach, see for details [5, 2].

In the example of Figure 1, we first instantiate implementations of GEAIntervalSelStrategyImpl and GEAIntervalSelStrategyImpl. We create a number of peers (e.g., 20) along with their trust and matching scores in order to compute GEAs. Then, we call the selectPeer() method to use the thresholds & weighted sum strategy implementation (geaThreshold.selectPeer()) and the interval-based strategy implementation (geaInterval.selectPeer()), respectively. We use default values for attributes of the GEAIntervalSelStrategyImpl and GEAIntervalSelStrategyImpl classes. Specifically, default values for weight and matchThresh used in GEAIntervalSelStrategyImpl are respectively 0.5 (thus, equal importance is given to both the matching and the trust dimensions) and 0.5 (which is a matching cut-off threshold). Default values for bandWidth and trustThresh used in GEAIntervalSelStrategyImpl are respectively 0.2 (which is an interval step) and 0.5 (which is a trust cut-off threshold), see for details [5].

The results of running the example are shown in Figure 2. Initially, 20 peers where created along with their matching and trust scores. These scores are in [0 1] and are generated randomly in order to exemplify the GEA computation. Figure 2 indicates that different strategies choose different peers. The GEAThresholdSelStrategyImpl approach selected the peer with peerID 15, which has the highest weighted average of its trust and matching scores. In turn, GEAIntervalSelStrategyImpl selected the peer with peerID 3, whose matching score turns out to be in the highest matching band [0.8 1] and which possesses the highest trust score of 1.

```
Generated peers
    PeerID   trust score | matching score
    0            0.37     |      0.66
    1            0.81     |      0.67
    2            0.30     |      0.25
    3            1.0      |      0.81
    4            0.44     |      0.11
    5            0.055    |      0.93
    6            0.37     |      0.14
    7            0.88     |      0.76
    8            0.56     |      0.82
    9            0.56     |      0.36
    10           0.52     |      0.19
    11           0.79     |      0.32
    12           0.96     |      0.021
    13           0.77     |      0.29
    14           0.67     |      0.40
    15           0.98     |      0.87
    16           0.88     |      0.72
    17           0.036    |      0.65
    18           0.85     |      0.54
    19           0.24     |      0.34

selected by GEAThresholdSelStrategyImpl
    PeerID   trust score | matching score
    15           0.98     |      0.87

selected by GEAIntervalSelStrategyImpl
    PeerID   trust score | matching score
    3            1.0      |      0.81
```

Figure 2: The usage example result.

3 Future work

Future work proceeds at least along the following directions: (i) adding more strategies for combining trust and matching scores, (ii) extensive GEA testing, and (iii) smooth integration of the component into the OK system.

References

[1] 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.

[2] Fausto Giunchiglia, Carles Sierra, Fiona McNeill, Nardine Osman, and Ronny Siebes. *OpenKnowledge Deliverable 4.5: Good Enough Answer Algorithms*. `http://www.cisa.informatics.ed.ac.uk/OK/Deliverables/D4.5.pdf`, 2007.

[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] Pavel Shvaiko, Fausto Giunchiglia, Alan Bundy, Paolo Besana, Carles Sierra, Frank van Harmelen, and Ilya Zaihrayeu. *OpenKnowledge Deliverable 4.2: Benchmarking methodology for good enough answers*. `http://www.cisa.informatics.ed.ac.uk/OK/Deliverables/D4.2.pdf`, 2007.

[5] Pavel Shvaiko, Fausto Giunchiglia, Juan Pane, and Paolo Besana. *OpenKnowledge Deliverable 3.5: Specification of the good enough answer component*. `http://www.cisa.informatics.ed.ac.uk/OK/Deliverables/D3.5.pdf`, 2007.