Infrastructure for Ontological Resource Matching in a Virtual Organization

Michał Szymczak[1], Grzegorz Frąckowiak[1], Maria Ganzha[1], Marcin Paprzycki[1], Sang Keun Rhee[2], Myon Woong Park[2], Yo-Sub Han[2], Young Tae Sohn[2], Jihye Lee[2], and Jae Kwan Kim[2]

[1] Systems Research Institute Polish Academy of Sciences, Warsaw, Poland, marcin.paprzycki@ibspan.waw.pl
[2] Korea Institute of Science and Technology, Seoul, Korea, greyrhee@kist.re.kr

Summary. In our earlier work we have outlined general approach to ontological matchmaking in an agent-based virtual organization. The aim of this paper is to describe in details how matchmaking is to take place within the system under construction. The Grant Announcement application is used to illustrate the proposed approach. Questions concerning efficiency of matchmaking will be addressed and in this context a distinction between asynchronous and synchronous matchmaking will be proposed.

1 Introduction

As the amount of available information increases rapidly, sometimes the efficient searching method alone is not enough to obtain necessary information in timely manner. Therefore support is needed to share the burden of searching for and filtering information. In the era of ubiquitous computing, computer systems existing everywhere should be able to proactively provide information just in time. Resource matching is essential in order to develop system searching and recommending information required for a user in a specific context. This paper describes the infrastructure and methodology of resource matching in the environment of a Research Institute where most of paperwork is carried out through an intranet. System requirements which are set by the Research Institute include facilitating user specific suggestion based on the knowledge model, actual data and geospatial information about objects. In the following sections we discuss how these factors can be included in a single suggestion request processing. In order to do so, we specify matching functionality, its possible processing modes and system building blocks which allow to realize the requirements. Resource matching utilized in forwarding notices about new research grants to appropriate users is used to illustrate the proposed

approach. This work can be generalized and expanded to become a kernel of a smart information provider.

2 Defining matching

Let us start by defining matchmaking in the context of our work. By *matching* we mean establishing closeness between ontology class instances (object(s)) and (an)other selected object(s) for which certain *Matching Criteria* are met. *Calculating Relevance* is a method for finding a degree of relevance (closeness) between objects which are related through their properties. Our approach to calculating relevance was outlined in [9]. In the case considered thus far in our work, *Matching Criteria* is an ordered quadruple $\{x, q, a, g\}$, where:

- x is the selected ontology class instance (the *source object*)
- q is a SPARQL query [5] which defines a subset of objects that are considered relevant and will be matched against the source object x
- a is numeral $(0, 1)$ which is the *relevance threshold*—value above which objects will be considered relevant
- g is a sub-query processed by the GIS subsystem; this part of the system is responsible for finding cities which are located close to others (part of the *Duty Trip Support* application, see [7]). This sub-query is a triple $\{gc, gr, ga\}$, where:
  - gc is an URI of a city demarcated with the *City* class properties of the system ontology
  - gr is an operator which allows to either limit returned number of cities of interest ($AMOUNT$ condition) or to limit the maximum distance between the gc and the returned cities ($RADIUS$ condition)
  - ga is the parameter of the gr operator, it either limits the number of cities that can be returned or the maximum distance between the gc and the returned cities

To make the idea of the GIS sub-query clear we can consider its following two instances:

$$\{gc, gr, ga\} = \{geo{:}WarsawCity,\ RADIUS, 100\}$$
$$\{gc, gr, ga\} = \{geo{:}WarsawCity,\ AMOUNT, 50\}$$

As a result of the first query the GIS module should return all cities which are located not further than 100 km from the city of Warsaw (represented by the RDF resource *geo:WarsawCity*). On the other hand, the second request means that a maximum of 50 cities should be found (that are located closest to Warsaw).

Note that, in general, the GIS sub-query can be omitted, or replaced with a different criterion (or a group of criteria). Therefore, due to the lack of space, it will be left to be discussed in more details in subsequent publication.

2.1 Grant Announcement-based matching example

In our earlier work [7] we have provided an example of a scientist employed in a Science Institute in North-east Asia.

When represented in the system, this sample employee (Prof. Chan), can have several profiles assigned. Below we depict an example of the general *Employee* profile, which consists of a *Personal Profile* and an *Experience Profile*:

```
:Employee\#1 a onto:ISTPerson;
     onto:id "1234567890"^^xsd:string;
     onto:hasProfile (:Employee\#1PProfile,
                      :Employee\#1EProfile),
     onto:belongsToOUs (:GOU).
:ResearchOU a onto:OrganizationUnit;
     onto:name ''Researchers Organization Unit''^^xsd:string.
```

In this example the *Employee#1PProfile—Personal Profile*, which describes the "human resource properties" of an employee. In what follows we use basic properties (however, our system supports a complete list of needed HR-related properties): *fullname*, *gender* and *birthday*. Furthermore, the *belongsToOUs* property indicates Prof. Chan's appointment in the organization.

```
:Employee\#1PProfile a onto:ISTPersonalProfile;
  onto:belongsTo :Employee\#1;
  person:fullname "Yao␣Chan"^^xsd:string;
  person:gender person:Male;
  person:birthday "1982-01-01T00:00:00"^^xsd:dateTime.
```

The second possible profile of *Employee#1* (Prof. Chan) is an *Experience Profile*. It demarcates human resource specialization in terms of fields of knowledge and project experience. Note that codes for the specification of fields of knowledge originate from the KOSEF (Korea Science and Engineering Foundation) [3].

```
:Employee\#1EProfile a onto:ISTExperienceProfile;
  onto:belongsTo :Employee\#1;
  onto:doesResearchInFields
    scienceNamespace:Volcanology-13105,
    scienceNamespace:Paleontology-13108,
    scienceNamespace:Geochronology-13204;
  onto:knowsFields
    [a onto:Knowledge;
    onto:knowledgeObject scienceNamespace:Volcanology-13105;
    onto:knowledgeLevel "0.75"^^xsd:float],
    [a onto:Knowledge;
    onto:knowledgeObject scienceNamespace:Paleontology-13108;
    onto:knowledgeLevel "0.40"^^xsd:float],
    [a onto:Knowledge;
    onto:knowledgeObject scienceNamespace:Geochronology-13204;
    onto:knowledgeLevel "0.90"^^xsd:float];
  onto:managesProjects (:Project1).
```

According to the *Experience Profile*, Prof. Chan specializes in *Volcanology*, *Paleontology* and *Geochronology*. Level of knowledge in each of these areas is expressed as a real number from the interval $(0, 1)$, respectively:

0.75, 0.4, 0.9. Additionally, *Employee#1* who is described with that profile manages a project *Project1*. It is a scientific project in *Volcanology*, which has its own profile:

```
:Project1 a onto:ISTProject;
  onto:managedBy :Employee\#1;
  onto:period
    [a onto:Period;
    onto:from "2008-06-01T00:00:00"^^xsd:dateTime;
    onto:to "2009-05-31T00:00:00"^^xsd:dateTime];
 onto:fieldsRef scienceNamespace:Volcanology-13105;
 onto:projectTitle ''Very Important Volcanology
        Scientific Project''^^xsd:string.
```

The listings introduced above set a *context* within which we place a member of an organization. In order to make the matching definition clearer, let us now introduce an instance of a *SampleGrant*. Note that the sample grant could be replaced by any resource that is delivered to the organization and information about which has to be delivered to the right employees (e.g. a book, or a transport of copy paper). Profile of the proposed *SampleGrant* specifies its domain as *Geochemistry*. Obviously a resource could be demarcated using more complicated structure of covered areas and the proposed approach would work as well.

```
:SampleGrant a onto:ISTAnnouncement;
 onto:hasDescription
        ''Description of the exemplary grant announcement.
         It should be really interesting.''^^xsd:string;
 onto:refScientificFields (<scienceNamespace:
                                        Geochemistry-13200>).
```

In order to match the *SampleGrant* announcement with a human resource represented by the *ISTPerson* class instance, the following process has to be executed:

1. Construct a set of *Matching Criteria* x, q, a, g:
   a) $x =: SampleGrant$ (comparing against $:SampleGrant$)
   b) $q =$

```
PREFIX onto:
  <http://rossini.ibspan.waw.pl/Ontologies/KIST/KISTVO>
SELECT ?person
WHERE {
  ?person isa onto:ISTPerson.
  ?profile isa onto:ExprienceProfile.
  ?person onto:hasProfile ?profile.
  ?person onto:belongsToOU :ResearchOU}
```

   c) $a = \frac{1}{40}$ (sample value)
   d) $g = NULL$, since *Grant Announcement* scenario does not require any geo-spatial support

2. Execute thus generated SPARQL query. In the case of the *Grant Announcement* scenario [8, 7] this query is going to filter employee' experience profiles leaving only researchers, i.e. employees that belong to the *Organization Unit* which is specific to all researchers (more on *Organization Units*, their role in the knowledge base and examples can be found in [7, 12, 9]). In our example, the result of the query is the *Employee#1EProfile*.
3. Perform *Relevance Calculations* for (in our example):
   a) source object $URI =: SampleGrant$
   b) target objects $URI's = [: Employee\#1EProfile]$
   c) allowed minimum relevance value: $R = \frac{1}{40}$

In Figure 1 we depict relations between *Employee#1* and *SampleGrant* objects. Unfortunately, due to the complexity of the example, listings including all relations between these objects would make it illegible. However, note that all objects are linked with properties which are included in the listings and refer to ontology classes and properties described in [9, 7, 12]. For instance, path between the *SampleGrant* and the *Employee1* is composed through the following intermediate nodes: *GeologicalScience13100*, *Volcanology13105* and *Employee1EProfile*. Additionally, following property weights are set in the ontology:

$$
\begin{aligned}
&voPropertyWeight(doesResearchInFields) = 2\\
&voPropertyWeight(isSubfieldOf) = 5\\
&voPropertyWeight(hasSubfield) = 3\\
&voPropertyWeight(refScientificField) = 2\\
&voPropertyWeight(invRefScientificField) = 8\\
&voPropertyWeight(hasProfile) = 1\\
&voPropertyWeight(belongsToResource) = 1
\end{aligned}
$$

These weights are given just sample values. Thus far we have not designed the mechanism for weights set up. Based on the weights and relations presented in Figure 1, closeness between the two objects can be computed. The computed relevance, according to the algorithm proposed in [10, 11], is $\frac{1}{36}$. Depending on the threshold used in the application, this degree of closeness may or ma not be considered "close enough" for the grant information to be delivered to Professor Chan.

3 Matching Request Processing

Matching Criteria defined in the previous section require adequate computations to be performed by:

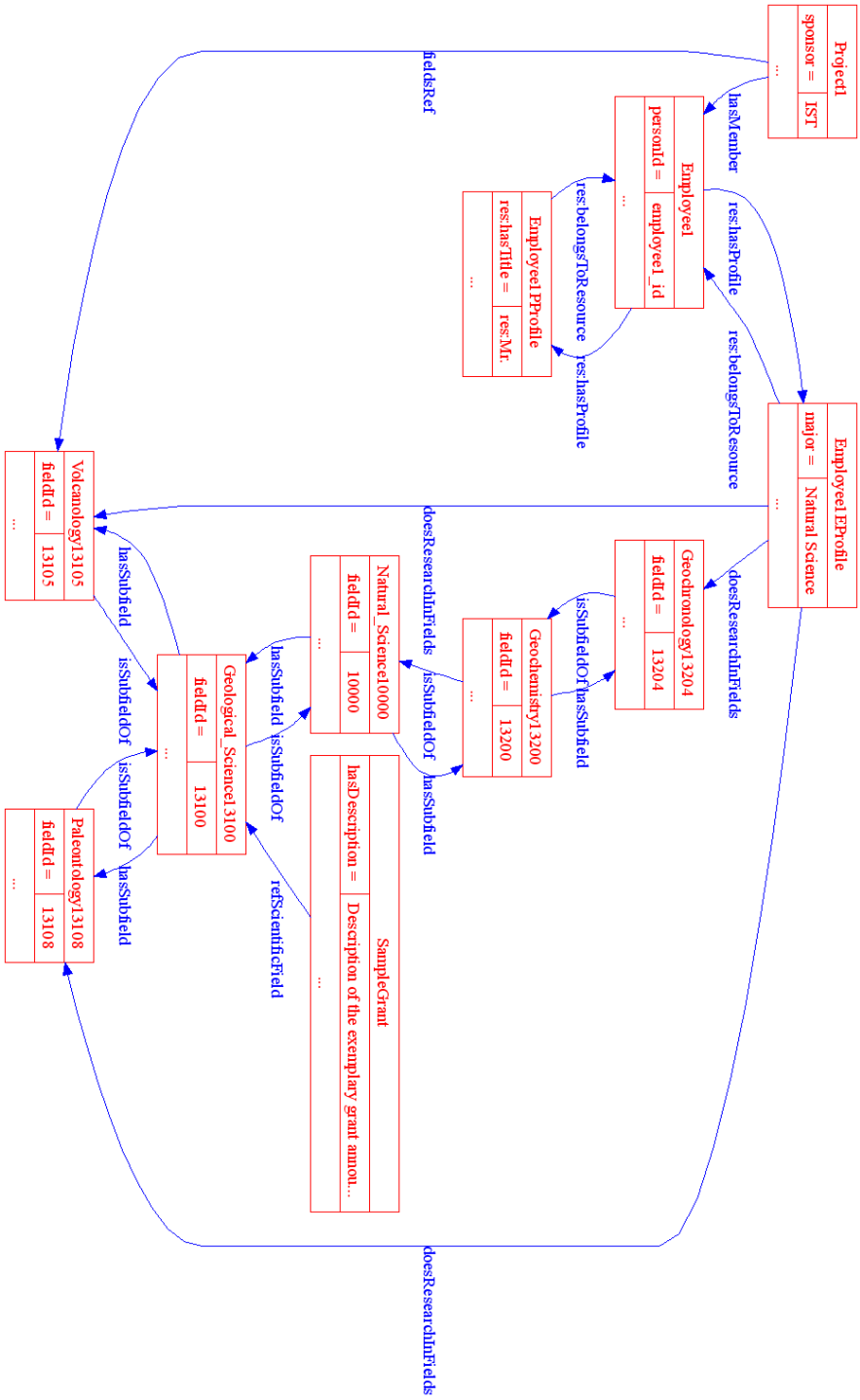

Fig. 1. Relations between sample Employee and sample Grant Announcement objects

- GIS Subsystem (mostly omitted in this paper)
- SPARQL engine
- Relevance Graph matching

All these operations are by default heavily resource consuming. Therefore we distinguish two basic matching modes to be supported by the system: *synchronous* and *asynchronous*.

3.1 Synchronous matching request processing

Synchronous method of matching request processing might be highly valuable for applications or even single components that require short response time. Possible usages of this method include but are not limited to (1) matching in the case of a process which has to deliver the result to a web page in a synchronous mode, or (2) requesting a single result based on the current state of objects stored in the system.

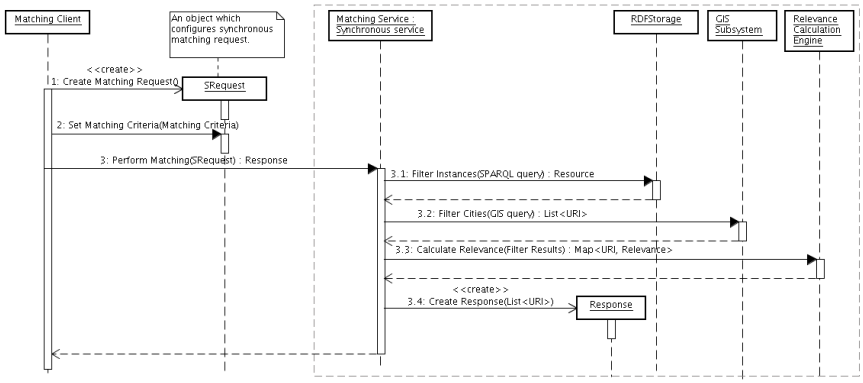

Fig. 2. Synchronous matching request processing

This approach has been represented in Figure 2 for the agent-based information processing. Specifically we can see there that:

1. A *Matching Client* (a role which can be realized by any agent capable of performing it) creates a synchronous request *SRequest* object.
2. The *Matching Client* fills *Matching Criteria* of the matching request.
3. The service which is responsible for processing synchronous matching requests (due to the fact that it extends abstract *SynchronousService* class) receives the new request and orders the *RDF Storage* (e.g. Jena) to provide results of SPARQL matching request part (*SPARQL Filtering*).
4. *GIS Subsystem* may be requested to perform *Cities Filtering* based on the GIS request.

5. Steps 3 and 4 filter objects which meet SPARQL (and GIS) *Matching Criteria*. These objects are processed by the *Relevance Calculation Engine* which is based on the *Relevance Graph* (see below).
6. Results of relevance calculation are wrapped in the *Response* object and sent back to the requesting client.

3.2 Asynchronous matching request processing

Asynchronous method of request processing might be more suitable for low priority relevance calculations and for calculation which have to be repeated within a certain time frame. For instance, consider search of employees who should be informed about a *Grant Announcement* (GA). Here, we can assume in that the *GA* is valid for some predefined time (until its deadline) and during that time there may appear "new" individuals who meet the *Matching Criteria*; e.g. due to their profile update. Since the asynchronous mode supports repeating request, results which include resource with changed profiles will be returned.

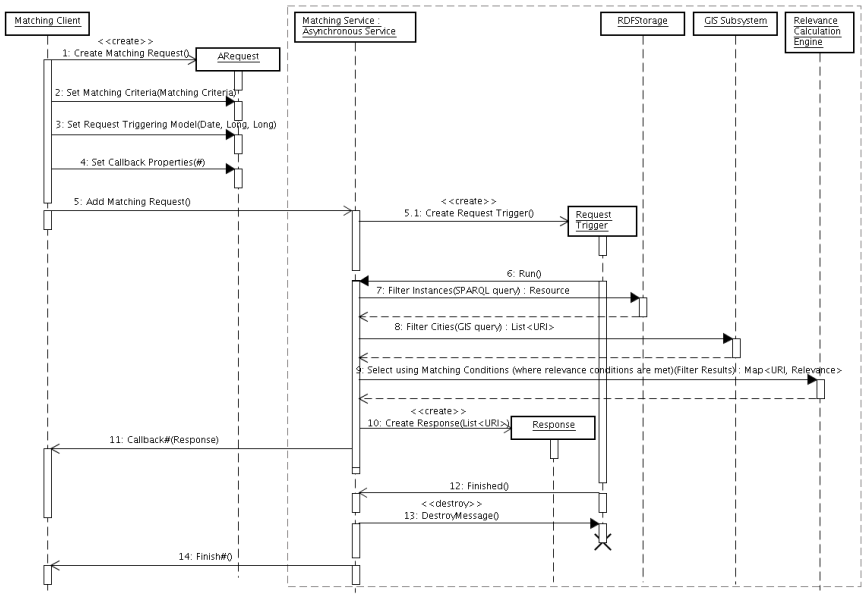

Fig. 3. Asynchronous matching request processing

Figure 3 represents the sequence diagram of the asynchronous matching process, which can be described as follows:

1. A *Matching Client* (a role similar to the *Matching Client* described in the section above) creates an asynchronous request *ARequest* object.

2. The client fills *Matching Criteria* of the matching request.
3. The client sets request triggering conditions.
4. The client sets request callback properties.
5. The service which is responsible for processing asynchronous matching requests (service which extends abstract *AsynchronousService* class) receives the new request and creates a trigger object which is set in accordance with triggering conditions specified in the *ARequest* object.
6. Each time the trigger executes its *Run* function, it starts a process which is similar to the *Synchronous Request Processing*. Function calls 6.1–6.3 correspond to the function calls 4.1–4.3 presented in Figure 2.
7. Results of relevance calculation are wrapped in the *Response* object and sent back to the client using the callback function defined in the *ARequest* object (callback properties).
8. If the trigger notifies that it has finished the scheduled work, the service is informed and similar request processing finish notification is sent to the client using the same callback settings.

4 System building blocks

Let us now describe in some details the main building block involved in relevance calculations.

4.1 Relevance Calculation Engine

The main role of the *Relevance Calculation Engine* (*RCE*) is, given a resource, produce a list of related resources with their relevance values. This module is designed in Java, with additional libraries from the Jena API [6] for ontology model handling, and the Structure Package [2] for dealing with the graph structure. The relevance measure algorithm applied here was first introduced in [10], and its initial application was described in [11].

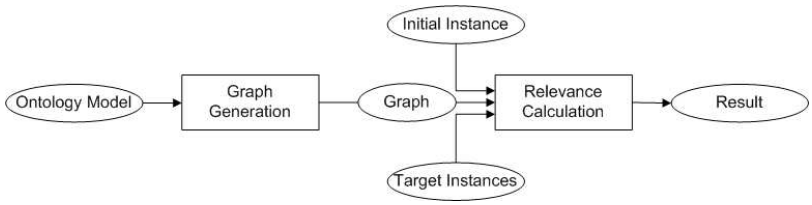

Fig. 4. Relevance calculation process

The process illustrated in Figure 4 includes objects that play key roles in calculating relevance between the *Initial Instance* and the *Target Instances*.

Relevance Graph

The core of the *RCE* is based on a graph structure that represents the underlaying Jena Ontology Model. The Model is interpreted as a directed graph $G = (V, E)$ so that:

V : set of nodes, representing all instances (or individuals)

E : set of edges, representing all object properties.

Note that reflexive relations are ignored. Upon creating edges, the value of the annotation property *voPropertyWeight* of each object property becomes the label of the edge, representing the distance between two nodes. The relevance value between two nodes is the inverse of the distance value. Graph creation function results in creating a weighted, directed graph on ontology resources and their object properties.

Relevance Calculation Interface

The *Relevance Calculation Engine* establishes closeness between a specific (*Initial*) resource and a given list of (*Target*) resources. The result is returned as an instance of the Java $Map\langle Key, Value\rangle$ interface, where *Key*s are the URI's of resources and *Value*s are the relevance results for these resources and the *Initial* node computed by the engine.

4.2 GIS Sub-system

In [8, 7] we have outlined utilization of the GIS module. The state of the art research shown that we can provide reliable geospatial backend for our system by using the following components: (1) GeoMaker [1] for collecting geographic coordinates of cities in the world, (2) PostgreSQL database [4] for storing that information and calculating distance between cities on demand, finally for caching the result, (3) Java for interfacing the GIS module with the rest of the system. As the GIS based calculation details were omitted in the text, we only sketch the description of the GIS. We will provide detailed reports on the architecture and efficiency of the GIS module in our subsequent publications.

5 Concluding Remarks

In the text we have outlined how the resource matching and relevance calculations will be facilitated in our agent-based virtual organization. We are currently implementing the proposed approach. Our subsequent immediate research goals are: property weights setup, stress and performance tests, matching including time frame constraints.

Acknowlegement

Work was partially sponsored by the KIST-SRI PAS "Agent Technology for Adaptive Information Provisioning" grant.

References

1. Geomaker. http://pcwin.com/Software_Development/GeoMaker/index.htm.
2. Java structure. http://www.cs.williams.edu/\~bailey/JavaStructures/Software.html.
3. Korea science and engineering foundation. http://www.kosef.re.kr/english_new/index.html.
4. Postgis:home. http://postgis.refractions.net/.
5. Sparql query language for rdf. http://www.w3.org/TR/rdf-sparql-query.
6. Jena—a semantic framework for java. http://jena.sourceforge.net, 2008.
7. G. Frackowiak, M. Ganzha, M. Gawinecki, M. Paprzycki, M. Szymczak, M.-W. Park, and Y.-S. Han. *Considering Resource Management in Agent-Based Virtual Organization*. LNCS. Springer, 2008. in press.
8. G. Frackowiak, M. Ganzha, M. Gawinecki, M. Paprzycki, M. Szymczak, M.-W. Park, and Y.-S. Han. On resource profiling and matching in an agent-based virtual organization. In *Proceedings of the ICAISC'2008 conference*, LNCS. Springer, 2008.
9. M. Ganzha, M. Gawinecki, M. Szymczak, G. Frackowiak, M. Paprzycki, M.-W. Park, Y.-S. Han, and Y. Sohn. Generic framework for agent adaptability and utilization in a virtual organization—preliminary considerations. In J. Cordeiro et al., editors, *Proceedings of the 2008 WEBIST conference*, pages ISâĂŚ17âĂŚ–ISâĂŚ25. INSTICC Press, 2008. to appear.
10. S. Rhee, J. Lee, and M.-W. Park. Ontology-based semantic relevance measure. *CEUR-WS*, 294(1613-0073), 2007.
11. S. Rhee, J. Lee, and M.-W. Park. Riki: A wiki-based knowledge sharing system for collaborative research projects. In *Proceedings of the APCHI 2008 Conference*, LNCS. Springer, 2008.
12. M. Szymczak, G. Frackowiak, M. Gawinecki, M. Ganzha, M. Paprzycki, M.-W. Park, Y.-S. Han, and Y. Sohn. Adaptive information provisioning in an agent-based virtual organization—ontologies in the system. In N. Nguyen, editor, *Proceedings of the AMSTA-KES Conference*, volume 4953 of *LNAI*, pages 271–280, Heidelberg, Germany, 2008. Springer.