Unspecified Ontologies for eGovernment Web Services Composition and Orchestration

Shlomo Berkovsky, Yaniv Eytani
University of Haifa
{slavax,ieytani}@cs.haifa.ac.il

Avigdor Gal
Technion – Israel Institute of Technology
avigal@ie.technion.ac.il

Abstract

Ontologies and Web services are the emerging technologies for service interoperability in eGovernment applications. This work proposes to adapt the notion of “UNSpecified Ontology” (UNSO) to the realm of eGovernment Web services. UNSO allows different users to specify their own personal ontologies, thus eliminating the need for an all-inclusive shared ontology and enhance the users' privacy. In the context of eGoverment Web services, it allows users to describe their need in a relatively free form manner and supports smart semantic matching capabilities which simplify use by inexperienced users. In addition, we propose to use these capabilities for the purpose of composition and orchestration of eGoverment Web services to enable specific behaviors tailored to user needs.

1. Introduction

Improved technology and secured communication enabled the development of eGovernment, defined by Richard Heeks as "the use of information and communication technologies (ICTs) to improve the activities of public sector organizations". To enable governments to fully benefit from eGovernment technology, regional governments progressively offer reference architecture and standards for the development of secure and interoperable services. Provision of generic governmental services is increasingly ensured by regional services with the support of technical deployment interoperability standards, such as IDA at the European level, and national government departments (e.g., ATICA in France or eEnvoy in the UK).

Although the deployment of such services might take several years, the emerging technologies to be used to ensure service interoperability are clearly ontologies and Web services. The latter serves as the main vehicle of describing generic governmental services, while the former aims at ensuring semantic consistency among governments, in services within a single government, and between a government and service providers.

Web services allow universal connectivity and interoperability of applications and services, using well-accepted standards (as UDDI, WSDL, and SOAP). Having this technology in mind, an eGovernment platform should enable accredited actors (representing government services) to enrich the set of Web Services available by publishing new eProcedures as custom-made Web Services. These services would embed access to existing legacy systems, invoke other Web Services and should be packaged in such a way, that they can themselves be invoked by other Web services.

To do so, governmental processes should enable composition and orchestration of Web services, as opposed to the existing monolithic approach, in which a process is defined as a single workflow unit. The nature of Web Services enables dynamic reconfiguration of workflows, based on the available Web services, relying on the establishment of business agreements between Web Service providers (governmental agencies and possibly outsourcing vendors).

In current eGovernment systems, a user publishing or searching a Web service is usually required either to fill-in a predefined form describing the matter of the service or to choose one of the available services. Exploiting predefined forms having a set of attributes describing the service is referred as an ontology-based approach. The set of attributes related to forms from a particular domain comprise the ontology of that domain. However, in a dynamic environment, where the set of functions are unknown a-priori, requiring predefined ontologies would be inappropriate.

Possible solutions, such as creating new forms to handle specific functionalities for each service, or developing a comprehensive ontology composed of many smaller ontologies, may be inadequate. The former might not be flexible and may well flood the system with multiple, partially similar and overlapping ontological descriptions of available services. This, in turn, will aggravate the process of matching the user's needs with the appropriate Web services. In a frequently changing and dynamic environment the latter could result in a barely manageable structure. Moreover, since a single ontology must be shared by all the services and users, might well obstruct it from being expanded in short-time intervals.

In this work we propose to overcome these restrictions by employing the notion of "UNSpecified Ontology" (UNSO [2]). In UNSO, there is no single globally known "master" ontology as the users dynamically specify parts of it. Thus, it allows for a wider range of functionalities, rather than just applications for a single domain dictated by a-priori ontology. In this way UNSO provides a novel technique for management of a dynamic set of ontology-based Web services and supports matching functionality, essential in many applications. Since the users are not forced to share any predefined ontology it simplifies the use for inexperienced users and increases the likelihood of successful matching identification and enhances users' privacy.

In addition, we show the possibility of automatically supporting Web services composition and orchestration using this flexible platform for dynamic smart matching. This is accomplished by either identifying the appropriate Web services (i.e., services that satisfy sub-descriptions of the original problem description), or by composing multiple Web services in a way that yields a solution to the original problem.

The rest of the paper is organized as follows. Section 2 discuses ontology-based data management. Section 3 presents semantic routing and matching. Section 4 discuses the UNSO approach. In Section 5 we propose ways for Web services composition and orchestration. Finally, in Section 6 we list our conclusions and directions of further research.

2. Ontology based data management

A major issue for the deployment of interoperable Web Services is the capability of service providers to provide a clear specification of a Web service, to enable mutual understanding of the delivered services. Current Web service standards focus on syntactic, operational details for implementation and execution, rather than semantic capability description. The recent introduction of the semantic Web [4] enables the specification of semantic Web services.

The semantic Web aims to extend the World-Wide-Web by representing data on the Web in a meaningful and machine-interpretable form. The semantic Web is based on a set of languages that provide well-defined semantics and enable the markup of complex taxonomic and relations between entities on the Web. Ontologies, commonly defined as specifications of a conceptualization [8], serve as the key mechanism for the semantic Web by allowing concepts to be globally defined and referenced.

In this work we propose to extend the use of semantic capabilities description of Web services using the concepts of semantic routing and unspecified ontologies. We build upon two threads of previous works. First is HyperCup [12], which proposed a flexible ontology-based hypercube topology for distributed decentralized data management. A global predefined ontology was used in HyperCuP to classify users as providers of particular information associated with the ontology slots. This classification determined the position of the user in a hypercube graph topology and location of any desired information in a bounded number of steps through a semantic routing, (see [1] and [11] for elaborate discussions).

For the problem at hand, available Web services are classified as performers of particular functionalities. These are associated with the ontology slots and mapped accordingly to the hypercube space. For example, consider the following simple ontology of recreation accommodation domain, provided by a local government tourism bureau. The example's ontology slots construct a 3-dimensional HyperCuP (figure 1) in the following manner: dimension 0 distinguishes between B&B(0) and hotel(1) accommodation, dimension 1 stands for cheap(0) or expensive(1) room, and dimension 2 for breakfast only(0) or all (1) meals. Clearly, a query for a hotel room including all meals will be routed to nodes 1 and 5, as only these nodes might possibly store accommodation offers relevant to the request.

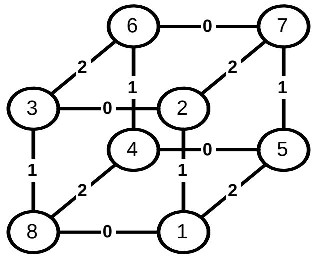

Fig.1 HyperCuP Example of 3-dimensional HyperCuP structure with 8 connected nodes

3. Semantic routing and matching

Ontological metadata facilitates an access to the domain knowledge. Existing approaches of ontology-based data management assume that the data sources share a single ontology allowing the access to the data. HyperCuP implements such technique and proposes a set of algorithms for maintenance of ontology-based hypercube graph topology supporting semantic data management and search. Hypercube graph topology was chosen due to its logarithmic diameter, increased fault tolerance and the symmetry guaranteeing equal load in the system.

The hypercube dimension *d* and the range of possible values in the dimensions *k* determine the maximal number of users that can be connected to the hypercube. A complete hypercube contains at most $N_{max}= k^d$ users, where every user is connected to *k-1* logical neighbors in each dimension, resulting in $N_n=(k-1)\cdot d$ neighbors. A user, for the problem at hand, is either Web service provider, or a user searching for a Web service represented by the description of the requested service.

Any edge in the hypercube, connecting a pair of adjacent nodes *X* and *Y*, is assigned a numeric value, referred as rank. When node *Y* is a logical neighbor of node *X* over dimension *i* of the hypercube, the rank of the edge, connecting *X* and *Y*, is *i*. Thus, the edges ranks range from *0* to *d-1* (dimensions numbering is arbitrary).

Any user *T* in the hypercube can act as an initiator of search or broadcast operation, which is performed as follows. The message, jointly with the rank of the connecting edge, is sent to all the neighbors of the initiating user *T*. Upon receiving a message, the nodes forward it only over the edges, whose ranks are higher than the rank of the edge the message was received from. This guarantees that each node in the hypercube will receive a message exactly once, and that any connected node will be reached in *O(d)* routing hops.

Consider the following example hypercube with a dimension *d=3* and the coordinates range *k=2* (figure 1). Eight nodes, numbered from *1* to *8*, are connected and form a complete hypercube. Each node is connected to exactly *d=3* logical neighbors and the ranks of the connecting edges are *0*, *1*, or *2*. For example, in respect to node *8*, node *1* is regarded as *0*-neighbor, node *3* is *1*-neighbor, and node *4* is *2*-neighbor. Let node *8* be the initiator of the broadcast operation. The messages are sent to the neighbors, i.e., nodes *1*, *3* and *4*. Upon receiving a message over *0*-rank edge, node *1* forwards it to its *1*- and *2*-neighbors, i.e., nodes *2* and *5*. Node *3* forwards it to its *2*-neighbor, node *6*, and this broadcast through “higher-rank forwards” continues until all the nodes in the hypercube are reached (figure 2). Obviously, in this case no node receives the broadcasted message more than once, and the longest path in this hypercube is *d=3* hops long.

HyperCuP also proposes a dynamic algorithm for hypercube construction and maintenance in a distributed environment. The algorithm is based on the idea that each user can manage not just a single, but also a number of nodes in the hypercube graph. This is required in order to simulate the missing nodes in the topology of the next complete hypercube, which is implicitly maintained in any state. For example, consider node *4*, which simulates three missing nodes of the hypercube (figure 3a). The simulations are noted by the dashed edges *1-4*, *2-4* and *3-4*, as node *4* acts as a logical neighbor of nodes *1*, *2* and *3*.

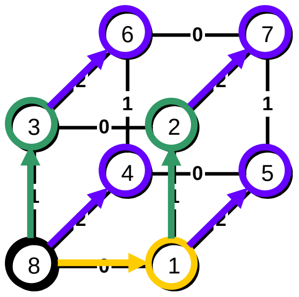

Fig.2 Broadcast procedure stages over the example HyperCuP structure.

When a new user connects to the network, its takes his place (according to the values of the ontology slots) in the next complete hypercube, releases the user that previously managed the node and starts functioning as a real hypercube node. For example, if a new user, that should be positioned at node *5*, is connected, it is routed to one of the logical neighbors of node *5*, i.e., nodes *1* or *4* (figure 3b). As the location of node *5* is simulated by the user maintaining node *4*, the new user contacts node *4*, builds a real edge between them and takes part of its functionality, i.e., builds a real edge with node *1* and starts simulating the neighbor of node *2*.

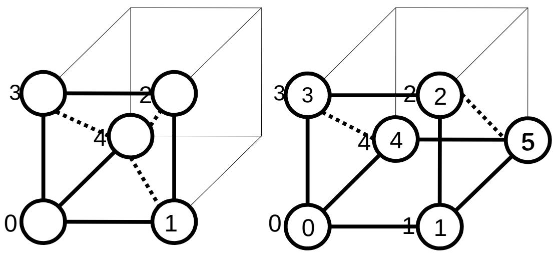

Fig. 3. (a) Implicitly preserved topology of the next complete hypercube; (b) Join of a new node

When a user disconnects from the hypercube, one of the remaining neighbors takes the responsibility for the node, previously managed by the leaving user. Since the next complete hypercube is constantly maintained, previously discussed broadcast and search operations are not affected by sporadic joins and departures of users. Elaborate description of the way the HyperCuP topology is maintained, jointly with the examples of such departures and joins, can be found in [12].

As we already stated, users are classified as providers of a particular service. A single predefined ontology defining the domain semantics, inherently organizes the users providing the same or similar contents, in concept clusters. This facilitates effective querying of the generated topology using the above routing algorithm. Note that a query is built as a logical combination of the ontology slots, and is routed only to the users that can potentially answer it.

Using a predefined ontology constitutes the main drawback of this approach. General-purpose eGovernment applications support various types of transactions. Thus, general-purpose applications will require all-inclusive and global ontology, embracing as many as possible application domains. However, state-of-the-art ontologies are usually limited to a single domain. Moreover, even ontologies from the same domain are heterogeneous due to existence of different views on the domain. Recent research efforts, such as [7] and [9], focus on the issue of merging ontologies to generate a global ontology. However, this issue is a controversial one [14].

Using a global shared ontology may lead to an unacceptable situation where updates of the ontology involve a central point of management. This happens when many services and users share the ontology (as in the case of eGovernment). To solve this issue, we propose employing UNSO, data management platform supporting organization of users in HyperCuP-like graph structure not requiring a single master ontology. We discuss it in details in the following section.

4. Using the unspecified ontology

UNSO (UNSpecified Ontology) was suggested as a way to overcome restrictions associated with sharing of a globally predefined master ontology. It proposes a data structure that allows users to provide relatively free description of objects (in a form of <$attribute_i$:$value_i$> pairs list) and allows the ontology to grow dynamically with no limited range**.** In this approach, no master ontology exists and adding new objects (and attributes) does not require constant updates to such a shared ontology.

In UNSO, the Ontology is represented as a vector, whose slots correspond to the attributes of the object being described. The range of values of each slot corresponds with the set of possible values of the related attribute.

For example, consider a simple ontology that resembles the example presented in section 2. This ontology is represented by a vector containing three slots *[accommodation_type | price_type | meals]*. These slots have the following range of predefined values: *[{B&B, hotel, house} | {20-50, 50-100, 100-150} | {none, breakfast only, breakfast + lunch + dinner}]*. Clearly, HyperCuP-like semantic system based on this ontology will generate a hypercube with at most 27 nodes, each having up to 6 neighbors.

Generalization to Unspecified Ontology is performed as follows. The range of possible slot values is made unlimited by operating a hashing mechanism on these slots (instead of using a set of predefined values). For the above example we use hashing to a range of size *3*, mapping new values to their corresponding positions in the hypercube. Thus the vector: *[accommodation_type: B&B | price_type:40 | meals: breakfast+lunch]* is mapped to *[hash(B&B) | hash(40) | hash(breakfast+lunch)]*. The same *3*-dimensional hypercube is used to hold objects However, in this hypercube the values of the ontology slots are not anticipated by a predefined ontology. Thus, the users can independently insert new objects in a decentralized way, using the hashing mechanism (without a need to use a master ontology). To eliminate ambiguity, the terms used in the vectors undergo simple semantic standardization using WordNet [6] before the hashing process. (thus, for example, hash(auto) = hash(car)).

A search operation is done by inserting the desired description of the object and performing a broadcast search in a similar manner to HyperCup (figure 2). Users can specify a different number of unspecified slots. When an unspecified vector is mapped to an existing hypercube, its slots are extended to the dimension of the hypercube. For example, the unspecified vector *[accommodation_type:hotel]* is extended to *[accommodation_type:hotel | price_type:*]*, where * stands for all possible hashing values. Thus the returned hotel rooms might be both of price *40* or *70*.

4.1. Multi-layered hypercube (MLH)

UNSO extends the original hypercube graph of HyperCuP into a multi-layered hypercube (MLH). It is a hypercube where each node is potentially constructed of another hypercube. Thus, in practice, more than one vector is used in the ontological description of an object to obtain a multi-layered ontology. For example, consider a *3*-layered ontology with three vectors *[a | b | c] + [d | e | f] + [g | h | i]* with two possible values for each slot. It generates a hypercube with 8 nodes, where each node contains another cube (figure *4*). This should be compared to *512* nodes hypercube, had we used one flat vector for the whole ontology. Obviously, this results in an hierarchical set of small and dense hypercubes, which are easier to handle in comparison to the hypercubes constructed over flat ontology.

The vector can dynamically grow by letting the users to add levels to the ontology. An unspecified vector (ontology-based description) is formed by <*attribute:value*> pairs. Two different hash functions

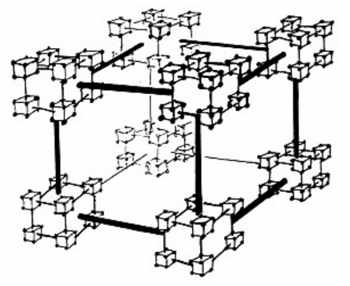

Fig. 4. Multi-Layered Hypercube (MLH)

are used to map the unspecified vector to the MLH. The first maps the attributes to the appropriate ontology slots, while the second maps the values to the numeric values of the slots. Thus, *$hash_1$(attribute)* determines the relevant dimension in the MLH, while *$hash_2$(value)* determines the numeric value in the dimension.

For example, consider the following description *[accommodation_type:hotel | price_type:70]*. It is mapped to the underlying MLH by applying *$hash_1$(accommodation_type)* and *$hash_1$(price_type)* to obtain the slot numbers (i.e., respective MLH dimensions), and *$hash_2$(hotel)* and *$hash_2$(70)* to determine the numeric values within these dimensions.

Multi-layered structure provides an efficient mean to resolve hashing collisions, since the attributes and their values are used to distinguish between objects from different domains. If a new object contains more slots than the number of dimensions in the current hypercube, the vectors in the relevant hypercube are updated to accordingly contain more slots. We assume that these updates are not frequent, thus they are feasible (in [2] this assumption was validated by the experiments).

4.2. The generalization process

In this section we describe in details the process for extending a fixed specified ontology to an Unspecified Ontology (illustrated in figure 5). As opposed to mapping of predefined attributes and values of a fixed ontology to an underlying hypercube graph, in UNSO the number of *<$attribute_i$:$value_i$>* pairs in the unspecified description is unlimited. UNSO dynamically generates a MLH that acts as a platform for the operations proposed in HyperCuP. During the insertion of an object to the MLH, it is forwarded towards its proper position. The semantic routing in the MLH is performed in a similar manner to the HyperCuP. It consists of a series of routings to the appropriate node in the current-layer hypercube routing.

Using the UNSO approach, the mapping of an object to its position in the MLH is performed through hashing mechanism, instead of using a fixed predefined ontology. Thus, new types of objects and new attributes can be easily added to the system, not requiring any updates of the ontology. The unspecified part of the ontology allows the users to mention relatively free list of attributes and values in the ontological descriptions of the objects and does not require any order of the mentioned attributes.

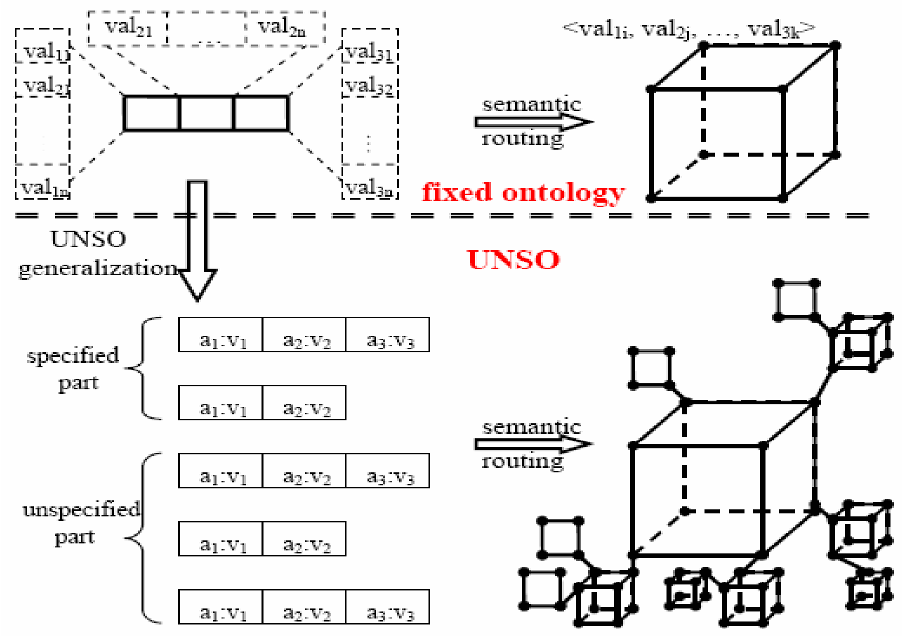

Fig. 5. Generalization of the fixed ontology to unspecified ontology

As a result, similar descriptions (i.e., similar entities), are mapped to adjacent positions within the MLH. Thus, similar entities are autonomously clustered without any explicit clustering mechanism. This implicit locality property in UNSO allows employing approximated searches to find entities that are similar to the entity being searched for (for an example, see also [3]). A search, containing a subset of possible entity attributes can be conducted, to find a wider range of entities.

5. Composition and orchestration of eGoverment Web services

Automatic composition and orchestration of Web services has drawn a great deal of attention recently. By *orchestration* we mean satisfying a user request for a desired Web service using a number of available services that each satisfies only parts of the request. Thus, in order to perform the more complex service, we use a number of semantically related simpler Web services, and execute them in parallel, such that the whole set services effectively provides the desired service. By *composition*, we mean taking advantage of a set of currently existing Web services to provide a new more complex Web service that does not exist on its own. This is done by sequentially composing the inputs and outputs of available services to achieve a final output that is desired by a requesting user.

There are Web service specification languages that specify semantic properties of web services. These languages are helpful in searching for the Web services that can participate in a composition. In a recent paper [13], the authors have provided a conceptual framework for a model-driven design using semantic Web services. Taking a conceptual modeling approach, inter-relationships between ontology concepts and syntactic Web services were defined and a generic algorithm for ranking Web services in a decreasing order of their benefit vis-á-vis a semantic Web service was proposed.

In this work we aim to explore a different venue for performing such operations. We propose methods that exploit semantic capabilities of UNSO by using description of Web services for the purposes of matching between similar Web services. This allows for the composition and orchestration of these services and enables specific behavior tailored to the users' needs. Web services found during the search process can be either parallelized or incrementally composed together to provide a new service that realizes that behavior. In the next two sub-sections we describe the proposed methods for both the orchestration and the composition of Web services.

5.1. Services orchestration

Web services orchestration is the process of satisfying a user's request for a desired service using a number of available services, where each of these services suffices only parts of the user's request. Thus, we aim to use the semantic matching mechanism of UNSO to relate simpler Web services and execute them in parallel such that the whole set will provide the desired service. It might be possible that no exact set of Web services providing tougher the requested service can be found. Our approach can be also used to allow finding other sets of services that have a close functional proximity to the desired Web service (if such sets are available).

As an example scenario we describe an application of booking both a hotel room and rented car. The booking process is done using two different Web services, one for ordering the hotel and one for the car rental. Note that the user specifies her needs through a set of $<attribute_i:value_i>$ pairs. The user's request contains combined information regarding both the desired accommodation and the desired rent car. Some of the pairs may relate to both desires such as location, start date, end date etc.

UNSO uses the described matching mechanism to find all available matches (including partial matches). Finding the "right" set of Web service is done by verifying for each group of matches whether it can achieve the desire Web service functionality.

5.2. Services composition

Web services composition is a process that, given a description of a desired Web service and its inputs and outputs, takes advantage of currently existing Web services to provide a new service that does not exist on its own. We use the semantic matching mechanism of UNSO to find a set of available Web services that can be executed in a sequential manner where each service uses its predecessor outputs as inputs. The resulted output after executing the whole set of services is such that the output of the last service will provide the desired service (given the input). If no exact set of Web services can together provide the requested Web service, UNSO allows to find other sets of services in a close functional proximity to the desired Web service (if such sets are available).

As an example scenario we describe an application of traveling from one location to another using two different transportation methods. Reserving tickets for each is done using different Web services, one for ordering a bus ticket and the other for ordering a train ticked. The user specifies her needs through a set of $<attribute_i:value_i>$ pairs. The user's request contains information regarding the starting location (i.e. the service input) and the end location (i.e. the service output).

To achieve service composition capability by UNSO, we differentiate between service inputs and its outputs when mapping the service to the MLH. We ask both the service providers and the users to provide semantic information about the input and the output pairs. Finding the right composition of services is done by finding a path in the MLH graph topology from the input to the desired output by a BFS search starting from the users input location in the MLH. Such a path constitutes the desire Web service functionality.

Providing the users with approximated similar functionally Web services requires additional research. A related research direction is finding a composed Web service that complies with a given constraint. For example, consider the previous example when adding a “price small than” constraint.

6. Conclusions and further work

Ontologies and Web services are the emerging technologies for service interoperability in eGovernment applications. eGovernment platform should enable accredited actors to enrich the set of Web Services available by publishing new eProcedures as custom-made Web Services. These services would embed access to existing legacy systems, invoke other Web Services and should be packaged in such a way they can themselves be invoked by other Web services. To do so, business processes should enable composition and orchestration of Web services, as opposed to the existing monolithic approach.

In the current eGovernment systems, publishing or searching Web services usually requires either to fill-in a predefined form or to choose one of the available services using an ontology-based approach. For a dynamic environment such as the general-purpose eGovernment systems a predefined ontologies may well be inappropriate. To overcome these restrictions, we propose to employ the notion of “UNSpecified Ontology”. In UNSO there is no master ontology, and parts of it are dynamically specified by the users. It allows exploiting a wider range of functionalities, rather than just one application domain defined by a-priori ontology and enhances the users' privacy.

The UNSO platform allows both to simplify usage by inexperienced users by letting them specify their own ontology and to support Web services composition and orchestration. The latter is accomplished by identifying available services that satisfy sub-descriptions of the desired Web service and performing them in a way that will produce the desired solution.

In the future, we would like to test our approach with real-life scenarios. We would also like to extend our approach by combining it with two other relevant approaches. First we would like to use available language for ontology modeling for the semantic Web such as Web Ontology language (OWL) [10] to provide standard a semantic markup for the definition of concept classes, relationships among them, and their instances.

Another interesting direction is to enhance the MLH hashing mechanism using Schema matching tools [5]. These tools allow matching between concepts describing the meaning of data in heterogeneous, distributed data sources such as in the case of the eGovernment environment.

7. Acknowledgements

We thank Edward Furman for his comments during the preparation of this paper.

8. References

[1] S.Ayyasamy, C.Patel, Y.Lee, “Semantic Web services and DHT-based Peer-to-Peer Networks: A New Symbiotic Relationship”, In proceeding of the Workshop on Semantics in Peer-to-Peer and Grid Computing, Budapest, Hungary, 2003.

[2] Y.Ben-Asher, S.Berkovsky, "UNSO: Unspecified Ontologies for Peer-to-Peer E-Commerce Applications", proceedings of the International Conference on Informatics, ICI, September 2004, Cesme, Turkey.

[3] S.Berkovsky, T.Kuflik, F.Ricci, "P2P Case Retrieval with an Unspecified Ontology", proceedings of the International Conference on Case-Based Reasoning (ICCBR), August 2005, Chicago, IL.

[4] T.Berners-Lee, J.Hendler, O.Lassila, “The Semantic Web”, in Scientific American, 2001.

[5] A. Doan, J. Madhavan, R. Dhamankar, P. Domingos, A. Y. Halevy. “Learning to match ontologies on the Semantic Web”, VLDB Journal, 12(4): 303-319 (2003).

[6] C.Fellbaum, “WordNet - An Electronic Lexical Database”, MIT Press, 1998.

[7] N.Fridman Noy, M.A.Musen, “PROMPT: algorithm and tool for automated ontology merging and alignment”, In proceedings of the National Conference on Artificial Intelligence (AAAI'2000), Austin, TX, 2000.

[8] T.R. Gruber ,"A Translation Approach to Portable Ontology Specifications", Knowledge Acquisition, 5(2): 199-220 (1993).

[9] E.Hovy, “Combining and standardizing large-scale, practical ontologies for machine translation and other uses”, In proceedings of the International Conference on Language Resources and Evaluation, Granada, Spain, 1998.

[10] http://www.w3.org/TR/2003/WD-owl-ref-20030331

[11] S.Ratnasamy, S.Shenker, I.Stoica, “Routing Algorithms for DHTs: Some Open Questions”, In proceedings for the

International Workshop on Peer-to-Peer Systems, Cambridge, MA, 2002.

[12] M.Schlosser, M.Sintek, S.Decker, W.Nejdl, “A scalable and ontology-based P2P infrastructure for semantic Web services”, In proceeding of the IEEE International Conference on Peer-to-Peer Computing, Linkoping, Sweden, 2002.

[13] E. Toch, A. Gal, D. Dori “Automatically Grounding Semantically enriched Conceptual Models to Concrete Web Services”, Proceedings of the 24th International Conference on Conceptual Modeling (ER'05). Klagenfurt, Austria, 2005

[14] M.Uschold, “Creating, integrating and maintaining local and global ontologies”, In proceedings of the European Conference on Artificial Intelligence, Berlin, Germany, 2000.