COMMUNITY FORMATION SCENARIOS IN PEER-TO-PEER WEB SERVICE ENVIRONMENTS

Olena Kaykova 1, Oleksandr Kononenko 2, Vagan Terziyan 3, Andriy Zharko 4

Department of Mathematical Information Technology, University of Jyväskylä, P.O.Box 35 (Agora), 40014, Jyväskylä, FINLAND
*1helen@it.jyu.fi, *2olkonone@cc.jyu.fi, *3vagan@it.jyu.fi, *4anzharko@cc.jyu.fi

ABSTRACT

Companies need integrated information systems to react to requirements of their customers, collaborate with partners, identify and exploit new opportunities quickly and effectively. A need for solid technology that will provide effective Knowledge Management within and across industrial enterprises is crucial. The main attention of this paper is concentrated on formation processes of global peer-to-peer networks of Web-based resources in the context of OntoServ.Net - an automated industrial environment for assets management being developed by Industrial Ontologies Group [9,10]. To increase the efficiency of service (resource) discovery and to facilitate service integration in a decentralized environment, we present scenarios of peer-to-peer network formation and OntoShell mechanism that joins specification of the formation processes with the technology for adaptation of heterogeneous services.

KEY WORDS
Intelligent Web Services, Peer-to-Peer network formation, Ontology, cluster, network community, Semantic Web.

1. Introduction

Very few business applications can live in isolation. Most often, applications have to be integrated with other applications inside and outside the enterprise. This integration is usually achieved using some form of "middleware". Middleware provides the "plumbing" such as data transport, data transformation, routing etc. [1]. Enterprise integration can also help to reduce costs, increase operational efficiencies, expedite time to market, and improve return on information technology investments. Without enterprise integration, various infrastructures will lack the robustness and flexibility in a dynamic economy [2]. Now integration is the direction of success for enterprises. They invest large portions of their ICT budget on informational integration.

The recent trends in industry connected with enterprise integration demand solid technology to provide effective Knowledge Management within and across industrial enterprises [3]. Combination of Semantic Web and Peer-to-Peer technologies can give, as it was shown in recent researches [3-7], many attractive and powerful features for this domain, but these great possibilities are not evident for companies still. There is a need in deep analysis of the potential of the Semantic Web and Peer-to-Peer synergy, carrying out case studies.

There is a strong interest in the development of reliable platform for support of cooperative knowledge management and flexible integration of various applications, Web Services and industrial resources. MIMOSA1 and PROTEUS2 are large initiatives launched recently in the USA and EU with the goal of providing such platform and technology for industry. OntoServ.Net environment is being developed by “Industrial Ontologies” Research Group3 as a pilot implementation of the innovative approach to the development of the Semantic Web based environment for industrial systems and provides a maintenance platform for automated management of industrial resources [8, 9]. Provision of efficient search mechanisms is one of the most important challenges in the peer-to-peer network for integration of Semantic Web services within OntoServ.Net.

Existing techniques of P2P search are based on very simple messages propagated among nodes, sharing computer resources [6, 11]. Such search queries contain logical conjunctions of keywords and does not have any agreement about using common terms in them. This approach is not suitable for search in P2P networks of web services considered in the OntoServ.Net, since search is performed based on the semantic descriptions of services and involves complicated semantic queries that have to be satisfied by search results (found services that can service desired actions). We use notion of service ontology as an agreed specification of service classes existing in the environment. Service ontology provides primitives and schema for services description so that taxonomies of services, their specific characteristics and integration-oriented knowledge can be modeled.

1 http://www.mimosa.org/
2 http://www.proteus-iteproject.com/
3 http://www.cs.jyu.fi/ai/OntoGroup/
DAML-S upper ontology is put in the basis of semantic-enabled service description representation.

Semantic descriptions are meant to be correctly interpreted by all applications in the environment. This is achieved via “semantic contracts” taken by software developers on the semantics of the specified by ontology concepts, which are referred from descriptions and used for tagging of data similarly to that in markup languages. Developed software should support meta-model, so-called upper-ontology, and possible models build over it. In the case of web services, upper-ontology provides model for building hierarchies of service classes, defining their properties and interrelations between them. The issue of storing created models and redistribution of them presents another great challenge for development of distributed ontology-based environments and goes out of the scope of this paper. Several options exist for that: centralized, semi-centralized, distributed onto-base (ontology-database middleware), etc.

Since the efficiency of message routing in Semantic Web-enabled peer-to-peer network highly depends on the logical structure of the network [12], the research must pay attention to these issues. The exact role of ontologies in intelligent query routing must be studied.

The structure of this paper is the following. Section 2 presents idea of OntoShell in OntoServ.Net framework, which is highly related to the network formation. Section 3 discusses the scenarios of creation of peer-to-peer relations between web service platforms and formation of communities based on common ontology. Conclusions summaries the importance of the stated ideas and describe scope of the future research.

2. OntoShell Model

Components of OntoServ.Net environment are (web) services, applications and user-agents that interoperate, exchange information, provide and access available services in dynamic peer-to-peer network. Required services are searched in the service network as soon as need appears. Service search is based on the descriptions of the desired service functionality. In OntoServ.Net environment, every service component has a semantic profile in form of an RDF-graph based on correspondent ontology.

OntoShell is a core concept of the integration mechanism provided for the network components in the OntoServ.Net. Implementation of OntoShell provides two main functions: adaptation of heterogeneous components (developed without supporting communication protocols of the OntoServ.Net) and formation of the service network structure via hierarchical aggregation of shells for the support of efficient search strategies.

The internal structure of the generic OntoShell is presented in Figure 1.

**Figure 1. Structure of OntoShell**

**Adapter.** This module adapts legacy software and resources to Semantic Web environment. It mediates between the service-specific data exchange, connection protocols and defined by OntoServ.Net specification of semantic messaging, an ontology-based representation of exchanged messages and OntoServ.Net protocols. Adapters are based on the generic adapter-component that can be configured to wrap a concrete software. It requires adaptation at various levels: connectivity, representation and semantics.

**Shell Manager.** It implements the logic of integration of different services and service consumers within a single OntoShell. The group of services and service consumers located in the same OntoShell interact with each other via the OntoShell Manager, which provides basic services for messaging and service registry supported by Discovery Module of a shell. The Manager is a replaceable module and can implement different integration algorithms.

**Discovery Module and Service Registry.** These modules are responsible for the registration of components within and outside the OntoShell and implement search mechanism used to locate services elsewhere in the service network.

**Resource.** Resource of an OntoShell is a general name for components in OntoServ.Net. It can be a software component (implementation of a service, application, database, etc.), hardware (e.g. industrial devices, considered in OntoServ.Net) or other kind of non-computational resources, such as humans, expert’s knowledge, etc. Resources are meant to be accessible as services in OntoServ.Net in unified, ontology-based form whether they can be used just for information retrieval or performing some actions on other resources. Another notion here is that resource itself in this environment is an agent, autonomous self-interested entity.

After introducing the structure of OntoShell, we see that Manager is responsible for interaction with external entities (actually other Managers, as OntoShell representatives) and tightly related to Discovery Module. A Manager generalizes the profiles of all components that belong to OntoShell and hides its actual internal structure. OntoShell Managers can exchange service components
(one OntoShell can “invite” to its site a service component from another OntoShell), they can co-operate forming communities and arrange in a tree hierarchy, solving complex tasks in a distributed manner.

If the service providers want to place their services in a P2P environment of OntoServ.Net, they must put their services into an OntoShell. However, the service can be so simple that it would not be profitable for the service provider to buy a whole OntoShell for it and maintain it further. In this case there can be a separate OntoShell provider – a new business player.

3. Formation of P2P Network in OntoServ.Net

Since discovery mechanism highly depends on the logical structure of the P2P network and its formation [12], these two issues have to be considered thoroughly.

Let us discuss the formation of partner communities or clusters in OntoServ.Net. Nowadays the registration of communities (Web services cataloging) is centralized based on UDDI registries [14] and the formation of partner relations between Web service providers in P2P network is quite uninvestigated.

Researchers have defined a P2P community as “a set of active members, who are involved in sharing, communicating and promoting a common interest”, unlike a notion group that is rather “a physical collection of objects”. “Communities are useful in structuring the information storage space, discovering resources and pruning the search space. It also aids in better dissemination of useful information” [11].

The partner relation in the real world can be described in a simple way like the following. Assume that two services – e.g. Region Map Service and City Map Service – are entering the OntoServ.Net-like environment. Since the have an intersection in provided services, but at different levels (of performance), they can establish a partnership with the purpose of providing integral service: clients would use only one of them but if the go out of service’s ability to serve, one service can forward the message to another. By allowing the nodes of a P2P network “to answer queries on behalf of other nodes” [11], the search of the appropriate service component can be made much more efficient. That means that every member of the P2P community having the generalized profile of the community is able to answer queries on behalf of this community.

It is interesting to know what other researchers think about the formation of a P2P network. Here is one of the examples: “A node X links to a node A if (i) A is a special node (server) designated by the domain for P2P links, (ii) A is a node, known to X that it trusts (iii) A is a node that belongs to many communities X is interested in. For a novice/new node, (i) may be the most appropriate link. As X ages, it finds other nodes and adding these links improves search speed, information access and such. The linkages are similar to friendships in real life, or http links in the Web and are human directed” [11].

Clustering or formation of P2P communities highly depends on the goals of the industrial partners. The goals can be the following:

1. Cooperation with service providers of different types for the composition of new services from the service components.
2. Providers of industrial services are interested in cooperation with providers of similar services for collaborative service of clients and resource sharing (industrial software).

3.1 Service Composition-Oriented Clustering

Let’s discuss the first type of clustering – service composition-oriented. The simplest service composition assumes that compound service is just a sum of some services, a list of services, which correspond to the lowest, terminal level on a hierarchy tree. An example can be a polyclinic (compound service), which provides a list of different medical services. The ontology of such compound services can look like it is shown in figure 2.

In figure 2 terminal service components are atomic services that can be used just as building blocks for services of higher levels of hierarchy. **Top-level services** cannot be used for the composition of services of higher levels. The relations between classes of services in this

![Figure 2. Compound services ontology](image-url)

ontoy are part-Of. If one service has relation part-Of with other service it means that the second service is compound and the first one belongs to it.

More complicated rules of service composition assume that a compound service is not just a sum of other services but their integration. For this type of service composition the absence of some service in the set covered by the compound service can make the latter nonfunctional. For instance, we declared in ontology a compound service ‘Conference visit manager’. It consists of 3 integrated services: a service that registers participants on a conference, a service that transfers money from one
account to another and a service, which arranges a transport to the given destination. The relations between these services can be described by part of Service Composition ontology given in figure 3.

As we can see in the picture, Registration service, Money transfer service and Transport manager service have relation part-Of with Conference visit manager service. Registration service as a software provides outputs: conference bank account and participation fee, which are instances of classes bank account and money accordingly.

After the registration service has provided its outputs some integration module of Conference visit manager service must pass these values to Money transfer service. Explicit relation requires provides interoperability between different services on the level of their inputs and outputs. Simultaneously with this operation Transport manager service can find an appropriate transport agency and order a ticket to the site of conference. This service acquires such data as transport agency bank account and price of a trip. Integration module will use this data to buy a ticket with the help of Money transfer service.

Now we can see that the absence of, for example, Money transfer service, will make impossible to arrange a conference visit. For this type of service the composition provider (its integration module) of compound service has to establish a partnership with other compound services that have the missing terminal services, or find these terminal services somewhere else. To support automated search of potential partners it is reasonable to provide service meeting places, which will be discussed in Section 3.3.

3.2 Service Communities Based on Similarity

As for clustering according to the similarity of services, the structure of ontology is the same as it was shown in figure 2 for integration of different services. But this time the relations between service classes are subclass-Of (see in [15]). Such service classifying ontology can reflect different aspects of services.

For industrial maintenance services there can be minimum 4 different classifying ontologies:

1. Location. The notions of this type classify geographical coverage of the service (or region of availability of a service consumer). Most likely the relations between notions will be part-Of. Type of service consumer (in maintenance domain – device, “patient”).

2. Classes of breakages (“diseases”) (example of class: service that tests electrical contacts of devices of any types).

3. Used methods (Bayesian networks, Neuro-fuzzy).

Other classifying ontologies are also possible:

4. Ontology of service quality.

5. Ontology of service costs.

And now when a service provider wants to put his/her service applications into an OntoShell, he/she must choose an appropriate class from the ontology. As a result, possible profiles can describe “regional policlinic” (restricted geographical coverage), or the profile can combine classes of different service (consumer) aspects. Assume, that types of services are arranged into the following hierarchy (figure 4).

Figure 3. Part of the Service Composition ontology

Figure 4. Classes ‘Dentist’ and ‘Policlinics’ on the hierarchy tree

When service providers buy an OntoShell, they choose their types, e. g. Dentist service. Another service provider can choose more general class for his/her service – Policlinics. That means that he/she provides different doctors. The chosen classes correspond to the following nodes on a hierarchy tree (filled nodes in the figure 4).
While choosing the appropriate class from the hierarchy for their services, service providers must remember some rules. If the chosen class is very general (like Policlinics), the provider will receive more client calls. But if he/she is not able to serve them very often the trust to such service will decrease. And on the other hand, if the chosen class is too specific, the probability of client call is very low.

3.3 Establishment of Partner Relations Between Peers

Now, when the service provider has put OntoShell on his/her service he/she can send his/her software agents to special meeting places to find partners. This meeting place is an agent platform that supports registration of agents with their profiles and also supports negotiation protocols. On such meeting place the above-mentioned Dentist service agent can meet the Policlinics service agent and they might negotiate the partnership. ‘If you’, - the Dentist agent says to Policlinics agent, - “are not able to serve the need of the client, forward the message to me and I will give you the one third of my profit. And I will do likewise”.

The meeting platform can apply some certification tests before the permission of a service agent on the board. In this case the platform guarantees that the professional level of each participant will not be lower than a given value.

Thus there can be minimum 4 types of meeting platforms correspondent to the ontologies mentioned in Section 3.2. Agents can exchange links after obtaining common location features (ontology #1) or methods they use (ontology #4). Agents can also visit a “conference” of a very general level (e. g. Medicine), but find partners on a more specific level (e. g. Dentists). In this case if a peer is not able to serve the query of the customer, he/she will check whether the query matches the profile of his partners. If so, the query will be forwarded (in P2P network) to the partners and the customer will think he/she deals with a single organization.

The formation of a community requires the formulation of its profile through generalization of the profiles of its members. This profile can be the nearest general class on a hierarchy tree for all components that belong to the community. To get such a class one must find a node on a tree, which is a point of intersection of all branches that correspond to the classes of the community members. Figure 5 shows how to determine on a hierarchy tree a class for a community that contains 3 members.

But in fact the shown community “lies” about the services it provides. The cluster of this class as it is shown on a tree must include at least 9 services of different classes. Such cluster can loose the trust of clients if it happens not to have some service that belongs to the set the cluster covers. So the community must be interested in joining the services that will complete the set.

Another scenario is also possible, where some OntoShells can act as “mother” shells that register “child” OntoShells (Bank registers different bank services). The maternal OntoShell refuses registration of services, whose class does not belong to the set of specific classes covered by the parent class on a tree. Figure 6 shows an example of services, which can be accepted and refused (crossed node) by the maternal OntoShell.

The “Mother” shell takes care about advertising her “children” and implements the query routing. Since every shell can be registered in other shells, there is a possibility to create complex service communities formed by shells
as grouping and mediation mechanism that supports efficient search and communication.

If the children of the mother shell cannot process the income query, mother shell forwards the query to the upper level – to the “grandmother”. The grandmother disseminates the query among her own children, except the child who forwarded the query. Obviously, on the highest level of such hierarchy of OntoShells there will be a “super grandmother” – top-level OntoShell. If her children cannot process the forwarded query, then it means that there is nobody in the network who can process the query.

In the described scenario URIs of all “mother” shells must be registered and stored somewhere, because a successive OntoShell must choose some “mother” shell for registration.

Also a scenario combining the two above schemata is possible. That is, if some mother shell does not have any appropriate child to process a query it can forward the query to her partner shells first and after negative response to an upper level. So, mother shells can find partners on the described meeting platforms and establish peer-to-peer links with them in addition to centralized ones. The number of partners found on meeting platforms is limited by the amount of money that the mother shell provider has. Every provider of meeting platform is expected to take money from visitors.

6. Conclusions

In this paper a problem of establishing partner relations between Intelligent Web Service platforms in decentralized manner was discussed. Methods of organization of Peer-to-Peer communities of Web Service platforms basing on common ontologies have been researched. The proposed scenarios of logical network arrangement involve ontologies in order to increase the efficiency of semantic search in such decentralized environments. The proposed scenarios are supposed to be worked out in detail, some pilot ontologies for industrial domain will be developed and the final solution will be tested in simulation environment of OntoServ.Net.

REFERENCES


