ONTOENVIRONMENT: AN INTEGRATION INFRASTRUCTURE FOR DISTRIBUTED HETEROGENEOUS RESOURCES

Oleksiy Khriyenko[*1], Oleksandr Kononenko[*2], Vagan Terziyan[*3],

Department of Mathematical Information Technology,
University of Jyväskylä,
P.O. Box 35, 40014, Jyväskylä,
FINLAND
[*1]olkhriye@cc.jyu.fi, [*2]olkonone@cc.jyu.fi, [*3]vagan@it.jyu.fi

ABSTRACT

A new age of heterogeneous resource integration has begun. Next generation of integration systems will utilize different methods and techniques to achieve the vision of ubiquitous knowledge: Semantic Web and Web Services, Agent Technologies and Mobility. In this paper we overview the approach to heterogeneous resources integration base on OntoShell concept and Semantic Web-enabled integration environment (OntoEnvironment). We describe OntoEnvironment architecture, interaction models and business model for it.

KEY WORDS

Resource integration, Semantic Web, Peer-to-Peer.

1. Introduction

Nowadays, knowledge is one of the most valuable resources of enterprises and an important productivity and competitiveness factor. Therefore, in global and growing market the optimal usage of existing knowledge represents a key factor for the future enterprises. Knowledge-based assets, or intellectual capital is the sum of accumulated values of the company's shareable knowledge and expertise. Information sharing is critically important, because intellectual assets, unlike physical assets, increase in value while used; knowledge and intellect grow when shared. Information stored in archives is useless if it is not available as raw material for making decisions, improving quality, or enhancing productivity.

A new age of integration has begun. Gone are the days where integration consisted of tactical, point-to-point connections between disconnected applications. Today, integration is a critical and strategic factor in company's ability to compete. A successfully deployed integration network can: provide the agility for company to respond quickly and effectively to capture business opportunities, simplify business process and shorten business cycles to drive down costs, leverage company's vast ICT expenditures to realize real return on these investments. Integration is the unrestricted sharing of business processes and data among connected applications and data sources within an enterprise and between trading partners. According to [iPlanet, 2002], without integration, enterprises are left with stovepipe applications, inconsistent data, and inefficient business processes.

Talking about resource integration (in common case), we have a deal with heterogeneous in many aspects resources. Concerning this problem, ontology provides a common language at a human and a machine level to enable knowledge exchange and resource integration. Ontologies are the key technology used to describe the semantics of information exchange. They provide a shared and common understanding of a domain that can be communicated across people and application systems, and thus facilitate knowledge sharing and reuse. The underlying technology that enables main desired features is Semantic Web [Ankolekar et al., 2002], [Paolucci et al., 2002]. Semantic Web uses ontologies to create a comprehensive environment, in which intelligent agents (software applications) can access annotated resources, communicate and perform collaborative activities.

Next generation of integration systems will utilize different methods and techniques to achieve the vision of ubiquitous knowledge: Semantic Web and Web Services, Agent Technologies, Mobility [Curbera et al., 2002], [Clabby, 2002], [WEBSERVICES], [Ankolekar et al., 2002], [Paolucci et al., 2002], [FIPA, 2001]. In this paper we overview the approach to heterogeneous resources integration base on OntoShell concept and Semantic Web-enabled integration environment (OntoEnvironment). This idea comes from OntoServ.Net concept [IOG, 2003] developed by "Industrial Ontologies Group"[1]. We describe OntoEnvironment architecture, interaction models and business model for it.

[1] http://www.cs.jyu.fi/ai/Ontogroup

2. The approach to heterogeneous resources integration

As a matter of fact there is not yet an agreement about a service development standard, which allows automated interaction between services in heterogeneous environment. However complex environments that have to combine a variety of existing systems need such a standard to achieve maximal integration performance. When we have heterogeneous resources (services) and need to enable their autonomous integration over the Web, then we have to provide common language for their interactions to make them semantically enabled. We need to describe them in a common way based on a common ontology.

An OntoShell is a software shell, which has a deal to make resource (service) semantically enabled. The OntoShell is configured for a concrete resource based on ontology of it. The OntoShell represents a resource and carries its ontology-based description. It plays a role of a mediator, which provides interoperability between a resource and world of other OntoShells (other resources), where they have common interaction mechanisms and common language (Figure 1). Depending on resource domain, an ontology-based annotation must comprise not only a resource's description (inputs, outputs, parameters), but also many other aspects, which concern resources' goals, intentions, interactions aspects, etc.

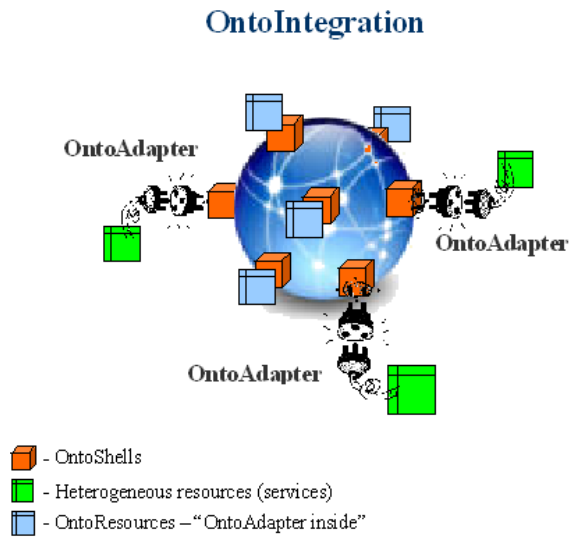

Figure 1. Environment-mediator

One of the important OntoShell's parts is an OntoAdapter for resources. When we develop service based on OntoShell approach (when we support interaction interface with OntoShell), we just need to adapt our service on semantic level via the visual interface of the OntoShell. On the other hand, if we need to transform an existing resource to a semantically enabled one, then we have to develop mechanisms for accessing that resource. Since the resources are developed according to different standards for both content (WSDL, C/C++ DLL, Java classes or applications, SQL Server, DCOM, CORBA, etc.) and transport protocols (TCP, HTTP, RMI, etc.) we need to design and develop respectively resource (services) transformation modules (OntoAdapters) for semantic, content and transport levels. It will be construction blocks, which will fill OntoShell depending on resource's description (Figure 2).

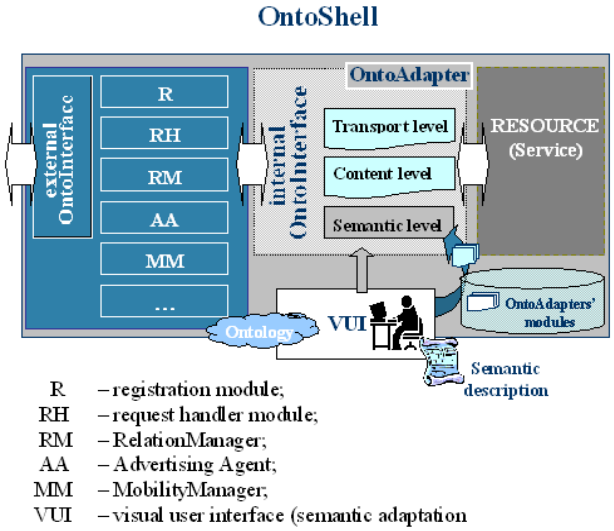

Figure 2. Structural schema of the OntoShell

OntoAdapters are ontology-based modules supplied with both interaction interfaces for the OntoShells and concrete class of the resources. For example, there are many services, databases, smart-devices (software interfaces for them), human, etc. "Ontology-based" means that we have to create all of the resources' Ontologies in advance. Ontologies building phase includes development of upper-ontology and development of ontologies themselves, which include data about resources (services) and environment domains. Concrete data will be annotated (marked up) in terms of upper- and common ontology. Here, ontology provides a basis for a well-understood "common language" to be used between system's elements.

3. OntoEnvironment for Semantic Web-enabled services

Considering the distributed resource integration, we propose architecture of ontology-based distributed integration environment for Semantic Web Services based on OntoShell concept (OntoEnvironment).

3.1 Environment architecture

OntoShell is the main structural component of the OntoEnvironment. As it was mentioned, OntoShell is based on a mechanism of making ontological description and providing interoperability for resources. So, we have environment with many OntoShells, which can interact with each other via common language. But it isn't enough, because these OntoShells need also the interaction, advertising and registration mechanisms, possibility to be mobile (movable), etc. That is why an OntoEnvironment is an organized set of the OntoShell-enabled elements (services) (Figure 3), such as:

- OntoAdapter for the resources;
- OntoShellContainer;
- OntoMeetingPlatform;
- OntoMobilityService.

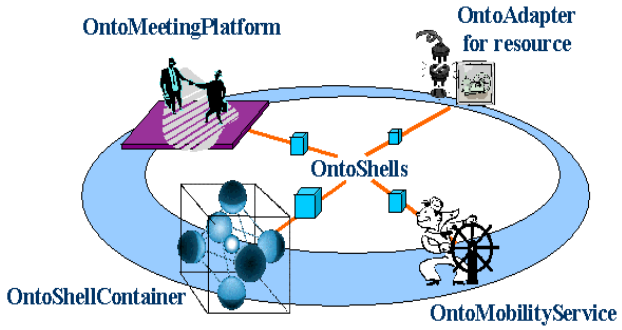

Figure 3. Elements of an OntoEnvironment

So, we observe the modular approach to constructing universal resource integration based on OntoShells. It assumes that resources can be nested to arbitrary levels via such shells for modeling multilevel cluster architecture. In the OntoEnvironment, services can be organized into a cluster (OntoShellContainer), which represents services wrapped within OntoShells. In order to share their information, OntoShellContainers must be also integrated into a higher-level network like a resource. Each element of the OntoEnvironment can be connected to several others. Finally the integrated elements form a decentralized environment of resources – Peer-to-Peer network. In such context, the OntoShellContainers become representatives of local resources at the appropriate level of the network. Resource clusters will reduce the cost of resource searches. Such consolidation into clusters may be organized according to various principles, such as:

- Location in the concrete server;
- Membership in a concrete domain;
- One-target federation of the resources (services);
- Geographical location (e.g. in cases, when a human is a resource, or a resource is a movable device, for example).

3.2 Hybrid interaction model

In a centralized interaction model, each OntoShell has a mechanism for registration to shell, which represents a cluster – aggregate of OntoShells. Thus, whole interaction will be realized via "mother shell" – ***OntoShellContainer*** (that is requests for searching of necessary resource and advertising yourself in "mother shell", what results in further discovery of registered resource). In such case we have a need to realize a special demountable (adapter) module for OntoShell representation in role of the OntoShellContainer for cluster. Such demountable module has to be configurable in a detailed way (especially in business model realization). It has to be responsible for observation of registration agreements, quality of provided search service, etc.

We may consider two main reasons for cluster organization:

- Cluster organization with a goal of useless traffic decreasing during searching the resource. In this case, cluster is organized in a way of hierarchical relation of "class-subclass" type based on resource ontology. A "mother shell" may register just such elements, which are members of its subclasses. Example of such clusterization is presented in Figure 4.

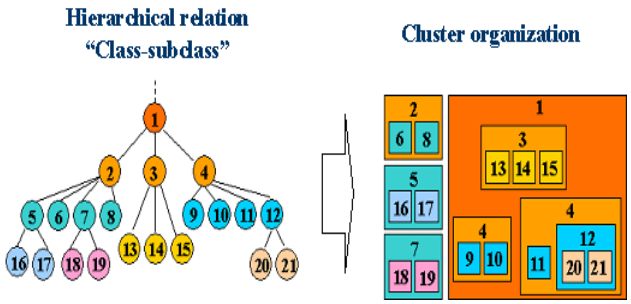

Figure 4. "Class-subclass" clusterization model

Since organization of such clusters will be carried out spontaneously and shells of some level may not register in a "mother shell", than we are not talking about centralized shells' management architecture.

- Cluster is organized to behave based on a community goal of closed set of functioning resources (components), which compose it. A cluster can be used to cover concrete domain with a set of different resources without relation to same class (for example maintenance platform with a set of services such as: main maintenance service, device alarm service, set of classifiers, etc.). In this case, "mother shell", which represent some cluster, provides search and interaction organization for registered resources. However a mother shell cannot always represent all of its elements the same way as a (sub)class in a hierarchical model because we are not assuming that an aggregation of heterogeneous components covers

a separate class. Organization of such heterogeneous cluster organization is represented in Figure 5.

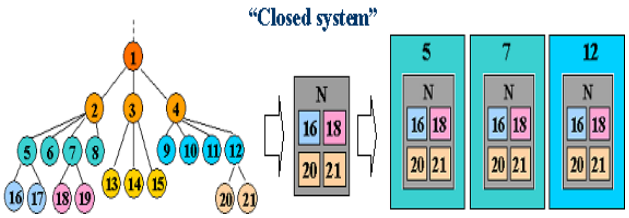

Figure 5. "Closed system" clusterization model

Because of impossibility of whole hierarchical clusters' nesting, which covers all levels of "class-subclass" type ontology, we cannot provide a guaranteed resource search via the "mother shells". Also, search within a cluster-tree (formed at some level) provides both centralized top-down search and non-effective bottom-up rising at the same time.

For resolving these two problems the OntoEnvironment introduces additional possibility of interaction between elements without "mother shell". This can be considered as a P2P interaction model. The main challenge here is own "record book" keeping by each OntoShell. This "record book" has to contain list of useful resources. In that way each shell (resource) can use own "record book" directly. Replenishment and modification of resource's "record book" is executed during interaction establishment with other resources. Such direct interaction model is represented in Figure 6.

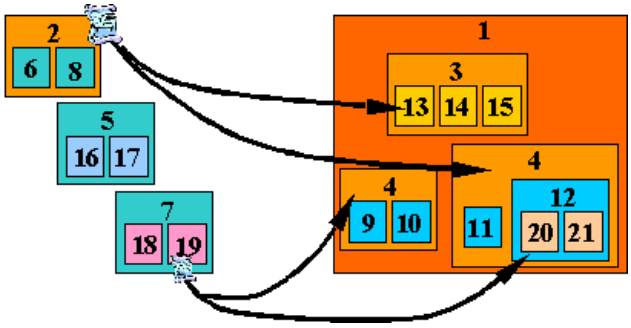

Figure 6. Direct interaction model

Lets consider some variants for resource search in hybrid interaction model:

1. Interaction organization via OntoShellContainer ("mother shell")
2. Records exchange during interaction between resources.
3. Using OntoMeetingPlatforms – places, where shells (more precisely, shell's Advertising Agents) can meet each other and exchange their "record books" (fill up them).
4. Using special search services.

During of each records exchange case (cases 2 and 3) a negotiation mechanism may be used.

OntoMeetingPlatform is a service, which provides possibility for shell's publicity agent (*PublicityAgents*) to meet each other and exchange records in "record book". This service may be placed into OntoShell or may be elaborated like service of new generation in OntoEnvironment and supplied with the same interaction interface like OntoShells. Such OntoMeetingPlatforms may be attached to some class of service classification tree in ontology and cover some specific resource domain. Such relation to the concrete domain may be fixed on OntoMeetingPlatform's annotation (description) and used by OntoShells' PublicityAgents.

Since amount of records will increase very fast, we have a need to supplement an OntoShell structure with "record book's" management block – *RelationManager*. Thus, we insert two additional elements into an OntoShell for management of relations. There are RelationManager and PublicityAgent blocks. These blocks have to be configurable. RelationManager has to be responsible for rectification of the "record book" depending on useless and useful records. PublicityAgent has to be responsible for visiting necessary OntoMeetingPlatforms, negotiation with other agents for exchange of the records, etc.

3.3 Mobility

While considering distributed environment for resources, the necessity of resource mobility emerges in a number of cases. In other words, there is sometimes a need to move a resource with its necessary "equipment" from one machine (computing system) to another one. Realization of such movement is a duty of special service (OntoMobilityService), which will provide mobility in OntoEnvironment. Thus, party (player), in case of need to provide mobility for resources, has to supply its computing system with such specific service.

To be a "player" within a mobile environment, elements of OntoEnvironment have to be supplied with MobilityManager module. This module has to be configured in conformity with a policy system (concerning mobility). Resource can be configured in both way like movement initiator or like available resource to be moved. All resources of a mobile environment, which support an OntoMobilityService and accordingly support mobility, have to provide necessary data for this service, such as: location, final point of destination, residence time, etc. Thus, we have a need to design respective ontology for messages between elements of mobile environment and ontology concerning behavior and relations of these elements.

3.4 Business model

Considering implementation issues of a distributed integration environment based on OntoShell approach, we have to consider also related business environment. In such environment service providers are interested in frequent use of their services; that is why service advertising and search plays an important role. Also, within such business environment some mediation elements, which provide necessary services for players, have to be embedded.

3.4.1 Patterns of behavior for elements of OntoEnvironment

OntoShell. At the very beginning of its appearance an OntoShell needs to advertise its resource. For realization of this goal we may consider two ways: registration in a "mother shell" and delegating responsibility for advertising duties to it; or self advertising during the life cycle by visiting OntoMeetingPlatforms. In case of need to interact with some resource (which is not available in a "record book"), an OntoShell has to use search process via "mother shell" or special search service. Also, alternative solution is stay on an OntoMeetingPlatform with a goal of meet necessary resource or find reference to it. During establishment of a link with environment element for records (from "record book") exchange or registration in a cluster, some negotiation mechanism is used. Thus, various aspects of behavior have to be configured in advance via a respective software visual interface module. Such configuration plays important role especially in business environment, where "service" costs "money".

OntoMeetingPlatform. We may consider two ways of OntoMeetingPlatforms providing. If they will be provided in a centralized way, then they will be advertised in one central point. But if they will be provided without centralization, then they will need to advertise themselves in the same way like OntoShells. In general case, OntoMeetingPlatform as a resource in OntoShell plays its (OntoShell's) role. It may register in a cluster, visit another OntoMeetingPlatforms, use search services, etc.

OntoShellContainer. OntoShellContainer provides a mechanism with more complicated behavior especially in Business Environment, where it plays a role of commercial mediation element. Loose configuration of such element may result to a negative profit. From the moment of an OntoShellContainer emergence in the same way as an OntoShell, it needs to advertise itself. Then in role of "mother shell" an OntoShellContainer has two main goals:

- Advertising of the "daughter shells" via itself advertising.
- Supplying with a search mechanism.

Registration in a cluster allows OntoShell to share its "record book" within whole OntoShellContainer for advertising purposes. This information allows execution of a more effective search and allows removal useless ascent (bottom-up rise) in a cluster-tree during search, which has been described in chapter #3.2. In case of further refresh of OntoShell's "record book", an OntoShell may proceed with its sharing within OntoShellContainer ("mother shell"). Depending on amount of new records (references) an OntoShellContainer updates its profile used for advertising of the whole cluster. There is a competition between "daughter shells" to get more queries from a mother shell based on advanced personal profile. In the same time, there is a competition between OntoShellContainers based on updated community profile.

3.4.2 Business relations between players

In our business model we may highlight the set of following "players":

A – provider of OntoShells, OntoShellContainers and OntoMeetingPlatform, OntoMobilityService;
B – OntoAdapters' blocks developers;
C – Owner of an OntoShell with resource;
D – Owner of an OntoShellContainer;
E – Owner of an OntoMeetingPlatform;
F – Owner of some search service.

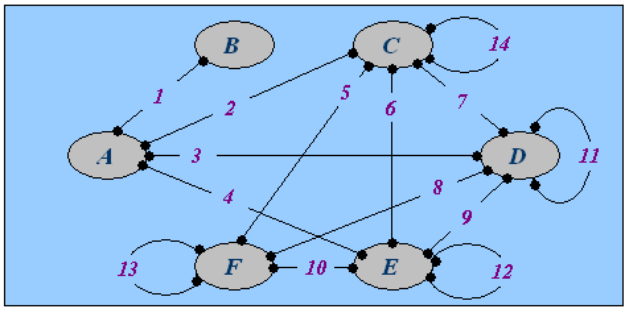

Figure 7. Inter-players interaction

Figure 7 shows business relations between players:

1 – Player "A" is a customer of player "B" for adaptation modules development (OntoAdapter's modules);
2 – Player "A" supplies OntoShell with necessary adaptation modules and OntoMobilityService (in case of need) to player "C" for inclusion of its resource into OntoEnvironment;
3 – Player "A" supplies OntoShellContainer and OntoMobilityService (in case of need) to player "D" for cluster organization;
4 – Player "A" supplies OntoMeetingPlatform and OntoMobilityService (in case of need) to player "E";
5 – Player "C" pays player "F" in case of need to search necessary resource;

6 – Player "C" pays player "F" in case of need to find someone or refresh "record book" during stay on an OntoMeetingPlatform;
7 – OntoShell registers itself in OntoShellContainer based on some agreements and advertises itself for further discovery. Additionally an OntoShellContainer provides search service for registered OntoShells. Player "C" pays player "D" namely for that search service;
8 – In a similar manner like in case #5, an OntoShellContainer may have a need to search some resource for guaranteeing a high-level quality of its services (in that way, increases its competitiveness). In case of search services use, a player "D" pays player "F". In same time player "F" plays a role of player "C" and may have a need to register in OntoShellContainer (case #7), then player "F" pays player "D";
9 – Player "D" pays player "E" for use an OntoMeetingPlatform by OntoShellContainer. On the other hand, OntoMeetingPlatform is a service, which needs to advertise itself. In that case, OntoMeetingPlatform may be registered in respective OntoShellContainer;
10 – Player "F" pays player "E" for use an OntoMeetingPlatform with a goal to supplement resource database of search service. On the other hand, OntoMeetingPlatform may use search service for find necessary resource (another OntoMeetingPlatform, OntoShellContainer). In that case, player "E" plays a role of player "C" and pays player "F" (case #5);
11 - In a similar manner like OntoShell, OntoShellContainer may register itself within other OntoShellContainer for advertising and additionally for search via a "mother shell". So, in that case, player "D" pays player "D" namely for that search service.
12 - OntoMeetingPlatform may visit another necessary OntoMeetingPlatform in case of need to advertise itself for concrete resources. Then player "E" pays to another player "E".
13 – One player "F" plays a role of player "C" in case of need to use a search service with a goal to supplement its resource database and increase its quality. Then this player "F" pays to another player "F".
14 – If we consider real business environment, we have commercial services, which require payment for its service. Then player "C" pays to another player "C".

4. Conclusion

Nowadays world is overcrowded by information, which is decentralized and non-shared (i.e. not available) for wide community of users, who would need this information. The Semantic Web approach based on creation and using common ontologies seems to be appropriate solution for integration and sharing useful information, knowledge, services and in general sense – Web resources.

Resources and services (like subclass of the resources) are heterogeneous and need to be preliminarily adapted via common ontology. According to this problem, we consider an OntoShell approach to heterogeneous resource adaptation and Ontology-based universal integration environment - OntoEnvironment. It allows transforming all resources (already existing and being developed) to semantically enabled resources for their integration. We consider environment, which supports mobility of the elements to enable effective integration of distributed resources.

Such environment provides integration within enterprise, as well as with trading partners, suppliers, and customers, by offering latest technology and open standards. This integration solution provides possibility to create a cost-effective, extended enterprise and get more return on information assets from existing ICT investments.

REFERENCES

[Ankolekar et al., 2002] A. Ankolekar, M. Burstein, J.R. Hobbs, O. Lassila, D.L. Martin, D. McDermott, S.A. McIlraith, S. Narayanan, M. Paolucci, T.R. Payne, K. Sycara, DAML-S: Web Service Description for the Semantic Web, URL: http://www-2.cs.cmu.edu/~terryp/Pubs/ ISWC2002-DAMLS.pdf, 2002.

[Clabby, 2002] J. Clabby, Web Services Executive Summary, URL: http://www-106.ibm.com/developerworks/webservices/library/ws-gotcha/?dwzone=webservices, 2002.

[Curbera et al., 2002] F. Curbera, M. Dufler, R. Khalaf, W. Nagy, N. Mukhi, S. Weerawarana, Unravelling the Web Services Web: An introduction to SOAP, WSDL and UDDI, Internet computing, March/April, 2002.

[FIPA, 2001] FIPA Interaction Protocol Library Specification Specification, FIPA00025. URL: http://www.fipa.org/specs/fipa00025/, 2001.

[IOG, 2003] Industrial Ontologies Group *"Semantic Web Enabled Network of Maintenance Services for Smart Devices",* Tekes project proposal, Agora Center, University of Jyväskylä, URL: http://www.cs.jyu.fi/ai/Metso_Maintenance.ppt, March 2003

[iPlanet, 2002] "iPlanet Application Server Enterprise Connector for CICS", URL: http://docs-pdf.sun.com/806-5504/806-5504.pdf, 2002

[Paolucci et al., 2002] M. Paolucci, T. Kawamura, T. R. Payne, K. Sycara, *Importing the Semantic Web in UDDI*, URL:http://www-2.cs.cmu.edu/~softagents/papers/Essw.pdf, 2002.

[WebServices] URL: http://www.webservices.org.