

Abstract Next generation of knowledge management systems will utilize different methods and techniques from the following communities to achieve the vision of ubiquitous knowledge: Semantic Web and Web Services, Agent Technologies, Mobility. A knowledge asset (Web resource or service) to become an intellectual capital must be shared; it increases in value while being used. We consider an infrastructure of distributed Web service components, which can be discovered in the Web based on semantic annotations, move to any target platform carried by mobile agents and perform their tasks locally and cooperatively. The challenge to use agents allows not only mobility of service components but also their learning while performing tasks locally. We are implementing this concept for automated monitoring and maintenance of field devices. A Model of Distributed Industrial Product Maintenance System based on interaction of heterogeneous distributed mobile Web services is described.

Mobile Agent-Based Web Service Components in Semantic Web

Vagan Terziyan, Oleksiy Khriyenko

*Industrial Ontologies Group, Agora Center, University of Jyväskylä, P.O. Box 35 (Agora), FIN-40014 Jyväskylä, Finland
E-mail: vagan@it.jyu.fi, olkhriye@cc.jyu.fi*

1. Introduction

The challenges for today's enterprise information integration systems are emerging. In order to manage and use information effectively within the enterprise, three barriers that increase the complexity of managing information have to be overcome; namely the diverse formats of content, the disparate nature of content and the need to derive 'intelligence' from this content [Sheth, 2003]. Indeed, the next generation of the Web is termed the Semantic Web, where semantic metadata plays a fundamental role. By annotating resources with semantic metadata, software can automatically understand the full context of what the resource (document) means and can make decisions about who and how these resources should be used. 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], without integration, enterprises are left with stovepipe applications, inconsistent data, and inefficient business processes. Integration is a must to gain and retain a competitive edge in today's business climate. It is not surprising that most companies plan to spend a large portion of their ICT budget on application integration. To build Web services through integration requires an infrastructure that enables end-to-end business processes. Applications should be integrated easily and painlessly and a solution must be built based on standards.

The world of services is evolving towards 'web-services', a simple concept where applications advertise their own capabilities, search for other applications on the web and invoke

their services without prior design. Web services represent a new breed of Web applications development [Curbera et al., 2002], [Clabby, 2002], [WebServices]. The full advantage of the power of Web services lies in the possibility for the user to dynamically discover and invoke a Web service. Web Services represent a new kind of web application that is characterized as self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. These services provide means of communication among different software applications involved in presenting information to the user or allow these applications to be combined in order to perform more complex operations [Clabby, 2002].

Web services are rapidly emerging as important building blocks for business integration. They are finding important applications in business-to-business, business-to-consumer, and enterprise application integration solutions. As such, Web services form a critical aspect of e-business architecture and, in that role; their reliable execution must be assured. Reliability must be a first-rank consideration for organizations deploying such solutions [Farrell & Kreger, 2002]. A fundamental aspect of Web service design is interoperability. For a company's Internet applications to be most effective, Web services must interface in seamless way internally and, potentially, externally with partners, suppliers, and customers [Peltz, 2003].

An XML-based standard, UDDI, provides registry of business and web services [UDDI, 2002]. According to [Ankolekar et al., 2002] and [Ankolekar et al., 2001], UDDI provides poor search facilities as it relies on pre-defined categorization through keywords and does not support semantic

description of search and does not implement semantic search of the services' advertisement. DAML-S [DAML-S, 2002] is adopted as a service description language, which provides capability to semantically annotate web services. The current version of DAML-S supports automated web services invocation, composition and interoperation. This is done under the set of ontologies that specifies a service as a process with inputs and outputs. DAML-S ontology provides classes and properties to describe content and capabilities of the Web Services. The advantages that UDDI gains when integrating DAML-S capabilities are described in [Paolucci et al., 2002]. DAML-S ontology of services provides enough knowledge that can be used by intelligent software agent to determine whether the service meets the agent's demands and the means by which the service can be accessed (inputs and outputs).

Agent technology lies in the intersection of distributed computing and artificial intelligence [Wooldridge, 2002]. Whatever is the definition, the main point is that an agent can carry out tasks without human supervision. Thus, an agent is a computer system capable of autonomous action in some environment controlling its own internal state. One can say also that an agent is autonomous only if it is capable of learning from experience and its behavior is determined by this experience. Agents are best suited for applications that are modular, decentralized, changeable, badly structured and complex [Parunak, 1998]. In particular, agents will turn the web-services into proactive entities working as peers to serve the end-user, representing him/her and defending his/her interests in a competitive world where services are negotiated and composed dynamically. Some initial experimentation on automatic generation of contracts shows encouraging results towards contractual web-services [Rodriguez & Sallantin, 1998]. According to [Burg, 2002], agents introduce an unparalleled level of autonomy into future systems so that users can delegate high-level tasks in a generic manner. Agents can now migrate to discover the resources and represent their user. Mobility of agents is an important property, which has not been fully utilized so far.

Now that agents have a foundation for interoperability, are getting deployed, the agent community has to reassess its position with regard to other initiatives, such as UDDI, SOAP, DAML, OIL and the semantic web, each of which is bringing answers to the problems initially addressed by the agent community. It is clear that these questions were not specific to agent technology and needed generic solutions of their own. Therefore the agent community needs to evolve from its insular agent-centric vision towards an agent-integrated ecosystem of technologies, embracing all relevant standards into an operational and deployable world. This evolution defines the charter for the Agentcities Task Force, an organization leveraging the efforts of the Agentcities around the world towards this freely accessible ecosystem for experimentation on the future active web-services [Burg, 2002].

In this paper, we present one of possible applications of mobile agent technology to management of Web Services (resources). We consider the case of industrial product's maintenance domain, where integration of distributed knowledge plays an important role in effective product's maintenance activities. We discuss the Distributed Industrial Maintenance System based on Semantic Web approach and network of platforms for agent-carriers of mobile service components.

2. Ontology-Based Integration Environment for Heterogeneous Resources (OntoShell)

How to make semantically enabled resources, and more important, how to transform already existing heterogeneous resources to semantically enabled?

To provide autonomous integration of heterogeneous resources over the Web, we need to describe them in a common way based on a common ontology. For example, in the domain of industrial product maintenance, we distinguish such resources as: smart devices, which can be considered as services because of their alarm or control systems (or some other software interface); set of diagnostic services or classifiers; platforms, which are represented by clusters or collections of various resources; humans, which can be considered as some special services; large enterprise information systems; etc. An ontology-based annotation must comprise not only a resource's description (parameters, inputs, outputs), but also many other necessary aspects, which concern their goals, intentions, interaction aspects, etc. Concerning this problem, we propose an OntoShell concept within an ontology-based universal integration environment (Fig. 1). Such an environment allows resources (services) to be designed and developed independently of other resources (services). This approach implies integration of heterogeneous resources (based on a specific standard) via attuned OntoShells, which interact with one another based on a common Ontology-based standard (environment-mediator) (Fig. 2).

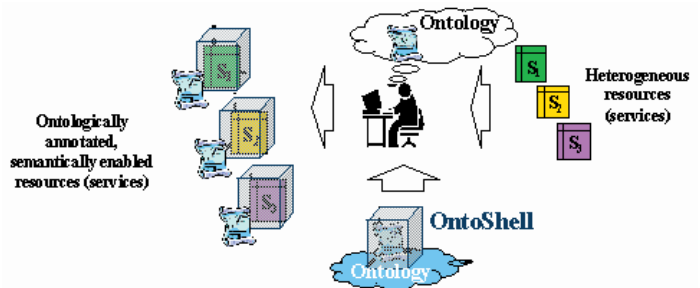


Figure 1. OntoShell concept

OntoShell is a software shell, which carries an ontology-based semantic description of a resource and plays the role of mediator (which knows a resource's goals and needs). This shell is configured for a concrete resource based on an ontology, which contains the resource's description. That is why it is important to elaborate on the details of an ontology.

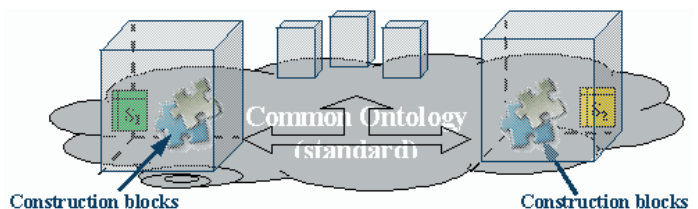


Figure 2. OntoEnvironment – “environment-mediator”

The structural schema of one such OntoShell is showed in Fig.3. 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 transportation protocols. They will be construction blocks, for OntoShells, and will be defined depending on resource's description (Fig.2). There are RCA modules for resource adaptation on the content level and RTA modules for resource adaptation on the transportation level (Fig.3).

A new generation of push services, which have an interface to interact with OntoShells, will also be based on this environment. If we have to cope with existing push services, we can develop transformation modules only for services, which are defined to configure a service's output interface. They are similar to RCA and RTA modules, but they work in the opposite direction (Fig.3).

A human executes an initial description of a resource via the visual user interface (VUI) (Fig.3) based on a common ontology and dynamically changeable windows. This process extensively plays a role in resource adaptation on a semantic level, and also gives necessary information to a linker module (L) (Fig.3) for the selection of construction blocks for concrete resources. An OntoShell's configuration is performed via the same visual interface, which indicates its active features (interaction methods).

Such OntoShells may be organized into a cluster, which also can be nested within another OntoShell, since an OntoShell can be considered a resource and has to be represented within the ontology.

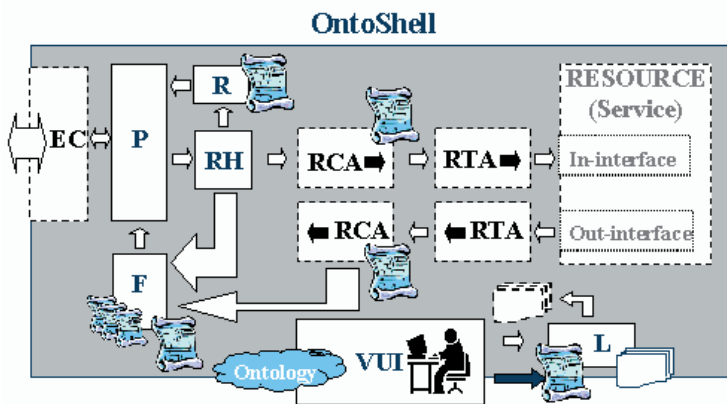





Figure 3. OntoShell's structural schema

- L – linker;
- P – packer/unpacker;
- R – registration module;
- F – forwarding module;
- RH – request handler module;
- EC – external connection (transportation) module;
- VUI – visual user interface (semantic adaptation level);
- RCA – block of resource content adaptation level;
- RTA – block of resource transport adaptation level (internal connection to the resource);

-  – resource and OntoShell description;
-  – description list of neighbors in P2P interaction model;
-  – demountable construction block.

The work of a registration module (R – shell's registration into the environment), request handler module (RH) and forwarding module (F – includes a description search engine of necessary resource) depends on the respective shell's configuration (inter-shell interaction architecture, class of internal resource, etc.) and the class of the request. Such classification of requests is described using an ontology for requests, very much like an interaction language between OntoShells. A packer/unpacker module (P) simply provides packing and unpacking for a message. But physical message transportation is performed by an external connection module (EC), which is a demountable construction block, because there are many methods for interaction on the transport level between OntoShells. This block is hence a block at the transport adaptation level for OntoShells.

So, we observe the modular approach to constructing a universal resource integration environment based on OntoShells. We can nest resources to arbitrary levels via such shells for modeling a multilevel cluster architecture (Fig.4). Resource clusters will reduce the cost of resource searches. Such amalgamation into clusters may be organized according to various principles, such as:

- Membership in a concrete domain;
- Location on the concrete server;
- Geographical location (in cases, when a human is a resource, or a resource is a movable device, for example).

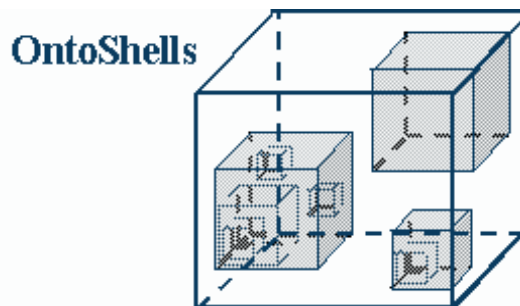


Figure 4. Multilevel cluster architecture

Interaction between OntoShells can be organized via either a centralized or decentralized (P2P) interaction architecture.

Centralized interaction architecture. For each shell in the cluster, the “mother-shell”, which represents a cluster of adapted resources, is highlighted. During the registration of an OntoShell with its “mother-shell”, the change (addition) of the cluster's description to a summary “daughter-shells” description is made. This registration list with descriptions of all internal resources is duplicated for each “daughter-shell”. Discharge is organized in the same way. In this case, the search of the necessary resource in the cluster may be organized by each “daughter-shell” or “mother-shell” (in case of need). Resources, which are registered not at one cluster, but at many clusters, have a more comprehensive list of the accessible resources and provide additional possibility to search resources in a through level

way out of the cluster (Fig.5). Such additional opportunity can speed up the resource search.

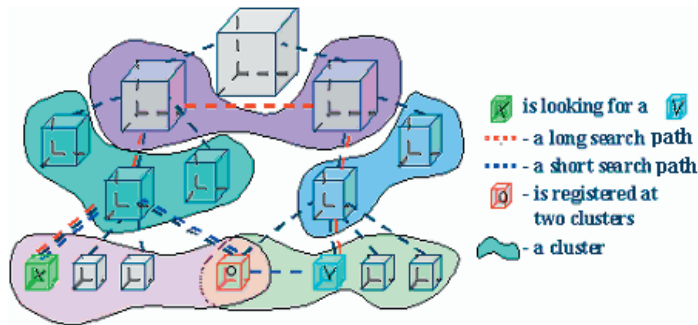


Figure 5. Through level search

Decentralized interaction architecture. In such architecture, there is no registration at the “mother-shell”, but there is an initial tune up for an OntoShell with the indication of the “neighbor-shells” list. The further changing of the list is carried out during the resources’ interaction (“life”). This list may be supplemented with a resource, which was used (was useful) and in a similar way may be lessened with a useless one.

Depending on an environment’s interaction architecture, to which a resource will be embedded in, an OntoShell can use both centralized and decentralized interaction architectures.

3.Mobile (Movable) Semantic Web Services

Why Mobile (movable) Web Services? First of the reasons is the utilized capacity of the server (which provides a service), shortage of resources when it should serve a huge stream of online queries. That problem concerns a service provider, and can be solved by means of service reproduction and distribution of its copies to other servers in the Web. In this case it is possible to decrease the utilized capacity of the concrete source. That will also improve service discovery among a large amount of the services.

Side by side with a provider a service requestor also needs Mobile (movable) Web Services. Imagine a situation, when a client of a service needs to use this service very often as such or as a part of a more complicated transaction involving several services. In this case we have frequent use of the network for service access. Besides, we cannot guarantee such important characteristics like:

- Minimal service execution time.
- Guaranteed, permanent connection with service.
- Guaranty of confidentiality and secure private information exchange.

In this case, it would be more effective to place all frequently used services at the client side. In this case we need to take into account the storage capacity of a client.

Another important concern is that Web service is often a business unit, which is being paid for its service. This means that a service that is transferred to a client side should keep business interests of its creator (owner). So we have here “self-interested” movable services. In this case, the mobility of services plays a very important role allowing “inviting” a service to a client side (platform) to serve locally.

Who and how will provide mobility of services? One solution to this problem might be the implementation of “Agent-Shell Platforms”.

Agent-Shell Platform (ASP) is an environment for a number of (mobile) Agent-Shells, which are assumed to be carriers of different Web Services (Fig.6). “Platform Steward” represents ASP. Concerning the OntoShell approach, “Platform Steward” is represented by the OntoShellContainer (“mother shell” of the second type). “Platform Steward” provides connection with a network of other ASPs (OntoShellContainers), registration of new agents on the platform, shares information with local agents. In the context of ASP, which supports agents’ migration between platforms, “Platform Steward” is rated like a cluster supplied with OntoMobilityService. P2P management tools for information movement via the network equip the platform.

Network of Agent-Shell Platforms

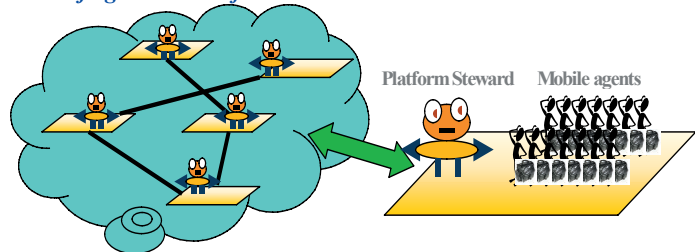


Figure 6. Agent-Shell’s Platform

Agent-Shell (AS) is the carrier of a web service (resource). In the context of the OntoShell approach, Agent-Shell is an OntoShell. It contains a mechanism of interaction with the platform and other agents, service engine. But why is it an agent? When we equip an OntoShell with a behavior mechanism, a goal, a set of mechanisms for participation in business environment, then it will become an agent. An agent, like a service representative, has to be responsible for the business interests of its service. An agent has to support service policy and certification. The mobility of the service and its agent-based implementation provides a possibility to a Web Service to learn during the execution on a service requestor site.

Considering both decentralized and centralized approaches to the management of our service network, it is possible to pick out following service network types:

- **Centralized platforms – centralized agents.** Each platform registers its services (provides descriptions) at some central (mediator) platform of the network. This platform (“Network Center”) gets direct requests for services from clients and its “Platform Steward” decides to which platform forward this request. Similarly, when a local platform steward gets a forwarded request, it analyzes the request and decides to which agent (service) on the platform to forward it to serve (Fig.7).

- **Centralized platforms – decentralized agents.** In this case, like in the previous one, the central point of the network selects the platform, which is assumed to be able to serve the request, but inside the platform, which finally gets the request, the right servant will be found based on a peer-to-peer (P2P) (semantic) search within the platform (Fig.8).

- **Decentralized platforms – centralized platform’s agents.** This case is similar to the first one, but interopera-

tion between platforms is based on a peer-to-peer semantic service discovery (Fig. 9).

• **Decentralized platforms – decentralized agents.** This is the case, when peer-to-peer interaction is considered within both: network of platforms as a whole and locally within each platform (Fig. 10).

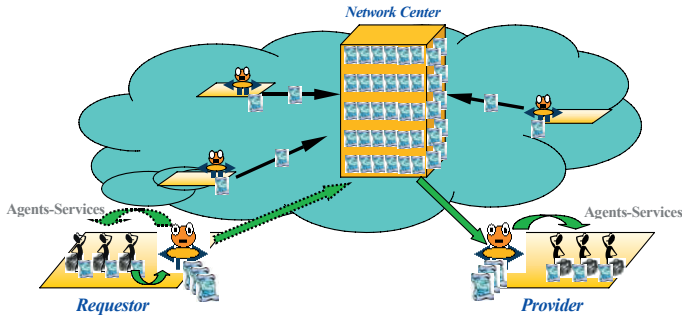


Figure 7. Centralized platforms – centralized agents

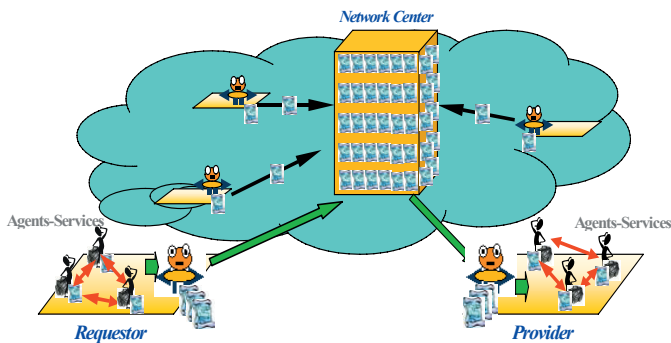


Figure 8. Centralized platforms – decentralized agents

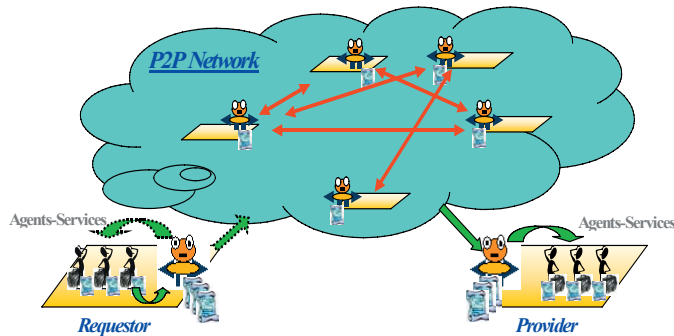


Figure 9. Decentralized platforms – centralized agents

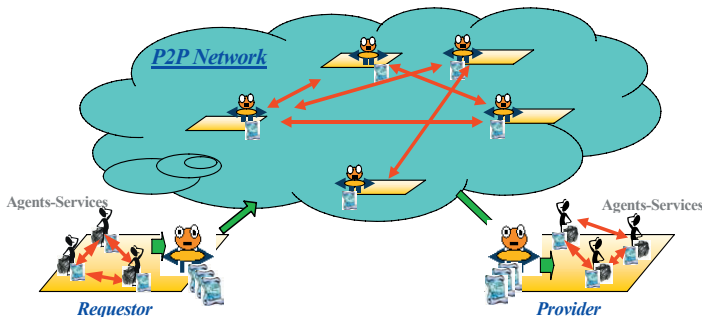


Figure 10. Decentralized platforms – decentralized agents

In a typical case we have compound services, which combine a set of distributed (atomic) service components into one service to provide more complex service for requestors. This complex service when created “on the fly” decides, which of its sub-services (up to components) corresponds to a request and how they should interact to resolve it. Outputs provided by some components could themselves be considered as requests for some other components etc. like in multi-agent systems.

Thus atomic service components are organized in a HAS_PART – PART_OF hierarchy from a service as a whole (abstract object) via (sub) services (abstract objects) up to concrete components, which form a “MegaHybrid” structure of a service network (Fig.11). Interaction between elements on each level may be organized in either centralized or decentralized way.

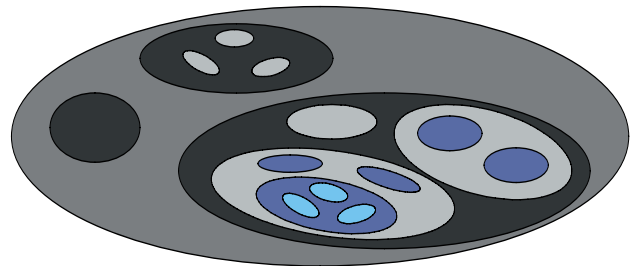


Figure 11. MegaHybrid network structure

Consider the case, when such complex service receives a request and provides another request as an output of one of its components. Assume that there are no other components in its platform, which can resolve this request. The service queries the network. As a result, such service will be found, and the request will be resolved. Evidently, it would be better for the service to accumulate its own set of links to services, which satisfy the requirements, and use them in violation of the standard search scheme in case of need. Then we will have a direct interaction between services (peer-to-peer interaction), not only between elements on some level, but also in the “vertical” and the “horizontal” plane of the service network hierarchy (Fig.12).

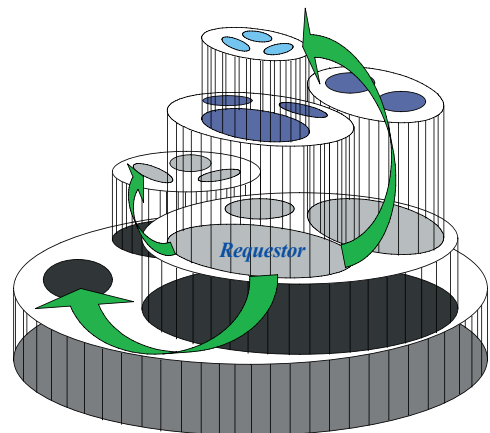


Figure 12. “Via-Level” Peer-to-Peer interaction

Nowadays there is already a large amount of existing Web Services. They differ not only by types of

service, but also by types of concrete physical objects that provide and consume the service. While previously services were meant to be consumed by humans, now industry needs services for another group of customers like various software applications and even **smart industrial field devices**. On the other hand, both humans and artificial objects (software or devices) can finally provide the service, which was discovered in the Web.

The evolution of the Semantic Web technology allows the description of Web Services based on a service domain ontology. Now we have a new phase in the Web Service evolution, when autonomous service interoperability plays a main role. However, the human component is still left and will stay in the Web Service environment both as service-consumer and service-provider, because many of the services provided by humans cannot be provided by software components.

Let's discuss both sides of human participation in the environment of Semantic Web Services:

- Human components as consumers of a new Web Service generation.
- Human component as providers of Semantic Web enabled Web Services.

A human component, when it is a user of a semantically annotated service, cannot and does not need to know the ontological service's description and specific query languages. He has to know exactly what he wants. To provide such "simple" interface between a human component and a network of Web Services, Agent-mediator is used. Agent-mediator is something like user-wrapper or layer between the human component and the services network, which knows how to handle both user queries and Web Services formal descriptions (Fig.13).

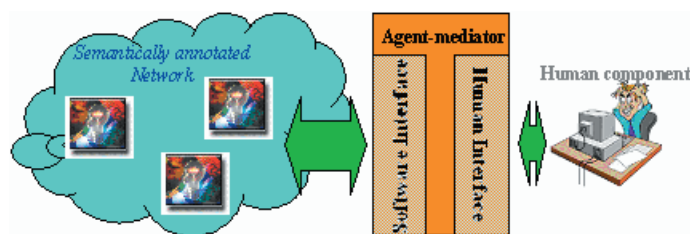


Figure 13. Agent-mediator - intelligent layer between Software and Human components

The main requirement to such user-wrapper is the provisioning of a simple, friendly human user-interface:

- Simple mechanism to choose the necessary type or class of service;
- Dynamic interface provision when filling the desired service's characteristics;
- Service's result representation in a human understandable form.

To satisfy this kind of requirements we have to describe the nodes in the ontology both in software understandable and human understandable form. Information representation in a human readable form generates a new problem. This problem is the heterogeneity of human languages. In this situation there are at least three choices:

- Having the ontology nodes' description in many languages (more space needed). Human component uses the description in his language (Fig. 14).

- Implementing translation services for information adaptation (Fig. 15).

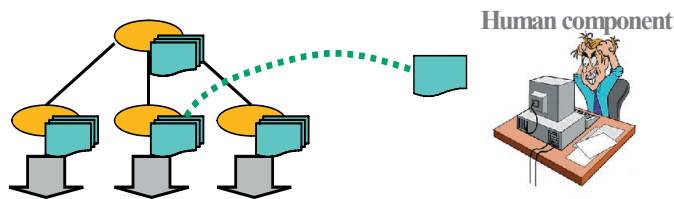


Figure 14. Multilingual node's description

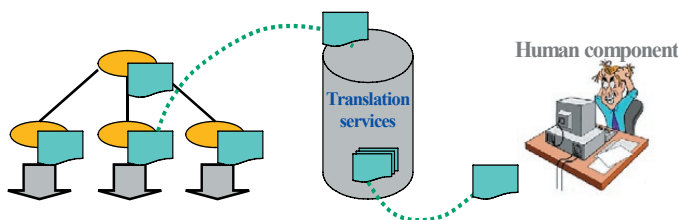


Figure 15. Information adaptation via translation service

- Using information visualization methods, different from language description:
- Graphical (visual) representation of information;
- Multimedia (video and audio) data representation.

Above methods may be used jointly. Of course, we have to take into account that the human component may use different devices for accessing the information. It may be a stationary device with more functional capability or a mobile device with limited resources.

This kind of relation between information type, access device type and information representation format for human interface can be semantically annotated. For that it is reasonable to elaborate an appropriate ontology.

What is the role of a human as a service provider within a network of semantically annotated Web Services? In fact, a service represented by a human component, it is the same Web Service as others and is described in the same way as other Web Services. Just like the human component in the service consuming case, in the service providing case a human component interacts with a network of other Semantic Web services via Agent-mediator (Fig.13).

A mobile agent-carrier of a Web Service represents it whenever it moves. However in the case of human service, the agent-carrier can hardly be movable within a network, because its burden – "human-service" can be attached to some location and cannot be moved to the service consumer side. In such context, the burden of this agent-carrier is a human provider-interface. In other words, an agent-mediator for such case is a combination of Agent-Sell, which is carrier of service, and human provider-interface. Human provider-interface should not only adapt formalized information for the human component, but at the same time has to make the opposite, i.e. formalize information from the human component in a software understandable form.

4. Global Industrial Maintenance Network based on Mobile Web Services

4.1. Industrial Product's Maintenance

Comprehensive maintenance systems play a key role in the maintenance of process equipment, field instrumentation, power supply, automation and information networks as well as automation applications. These maintenance systems aim at optimized maintenance for the entirety of a plant's life cycle. In a multi-dimensional and widely dispersed paper-producing company like Metso Oy (www.metso.com), many people in different divisions and in different places have specific expertise and experience. Bringing that diverse knowledge together is essential to effectively solve many problems that involve process control, quality control and paper process technology.

The maximization of productivity, usability and safety can be regarded as the main goal of automation in general. Other important aspects in the process industry are an increasing demand for quality and flexibility and emphasizing environmental aspects. Maintenance plays a very important role in achieving these goals.

One quite commonly used sales argument for smart (with embedded intelligence) field devices has been advanced diagnostics and preventive/predictive maintenance capabilities. In most cases these devices only give the possibility to perform maintenance rather than providing complete solutions for it. The challenge is, therefore, to develop a diagnostic system that automatically follows up the performance and maintenance needs of field devices offering also easy access to this information. Modern smart field devices with advanced on-line diagnostics provide a lot of diagnostic information during the field device lifetime. Effective management and analysis of this information is a key to success in future field device management [Pyotsia & Cederlof, 1999], [Ojala, 2001].

It is also very important that the diagnostic system is easy-to-use and results are easy-to-interpret. System users in a plant do not want to have yet another application interface to learn. That is why this diagnostics concept should utilize as much as possible the existing tools. In fact a user-friendly diagnostic system should not be visible to the user at all as a separate user interface. The system only notifies the user when needed [Riihilahti & Ojala, 2000], [Nikunen et al., 2001], [Pyotsia & Cederlof, 1999], [Pyotsia & Cederlof, 2000].

Previously, when the communication between field devices and control system was just analog signals, there was no possibility to acquire any diagnostics or operational information from the field devices, even if they were 'smart'. The operational information of a smart device can reduce maintenance costs and unnecessary process shutdowns, thus increasing plant throughput via increased control loop and field device operation knowledge.

A complete field device management and condition monitoring system consists of two software packages, Neles FieldBrowser™ and Valve Manager™. Neles Field-Browser™ is a maintenance tool to monitor the condition of the field devices continuously on-line. When Neles FieldBrowser detects some exceptional event on the field device, the maintenance staff of the factory is alerted. Valve Manager can be used to diagnose and configure the situation. These actions are taken when a field device is diagnosed for faults and needs to be repaired. A database viewer module enables to view diagnostic data with a browser [METSO, 2002], [METSO, 2003].

4.2. Distributed Mobile Maintenance System for smart-device

As was previously mentioned, there are two big classes of services' users – human components and software components. The class of software service requestors is extended with a new group of service users – smart devices. They should be able to access Web services in case of need. The semantic-enabled description of services is important to facilitate automated search and use of services by smart-devices.

This is the state of the product maintenance domain today:

- Every product is supported by some maintenance center;
- Maintenance is performed by humans, with poor automation (most of solutions cover only a part of the automation problem);
- Communications between centers are minimal, if they exist at all.

As site wide condition monitoring solutions are already widespread, the next logical step is breaking out from the site-oriented view and gaining the benefits of more large-scale solutions. If all information available in different industrial sites could be collected and analyzed together, significant improvements could be made to the accuracy of the analysis [Ojala, 2001]. A global maintenance web service network, which provides condition monitoring, fault prediction and recovery maintenance activities, integrates the maintenance experience from industrial sites. This scenario leads to a situation where the information management of tens of thousands of field devices is both distributed and centralized at the same time.

From a variety of Maintenance Services we may choose 3 main types:

- **Product-based Maintenance Service.** There are services, which provide all types of maintenance activities for specific products.
- **Profile-based Maintenance Service.** These services are specialized on specific maintenance activities for a wide class of products.
- **Location-based Maintenance Service.** This type of services combine Maintenance Services based on a location where products are used.

Actually each node related to a maintenance center may combine all of these three types of maintenance.

The next step of maintenance improving implies:

- Products' connection through one maintenance center to a maintenance network formed by maintenance services;
- Automated interaction between product and network for maintenance query;
- Discovery and utilization of maintenance resources and services within the whole network;
- Experience accumulation of service providers during interaction with clients.

As a result, every Maintenance Center in the Maintenance Network provides specific services. When a problem appears, the Maintenance Center with the most relevant knowledge for resolving that request must be found in the Maintenance network. Experiences are accumulated independently by each Maintenance Center during interaction between Maintenance Agents (agents which represent maintenance service) and client points with a possibility to be integrated together when needed.

Field agents are already considered to be useful for condition monitoring. Intelligent agents have found their place also in distributed web-services. The next step would be to embed smart-agents to the maintenance system for enabling machines to communicate and cooperate with each other. In case of mobile service agents, some Maintenance Service agent or agents can be selected for the specific emergency situation, based on the online diagnostics, and can be moved to the embedded platform to help the host agent to manage it and to carry out the predictive maintenance activities.

Structure of Maintenance Service Platforms. In the beginning of its lifecycle, each field device is registered to a fixed Maintenance Center, which is the responsible point for this device. Exactly that Maintenance Center is like a bridge, which ties together the field device and the Network of Maintenance Centers (Maintenance Network). For interaction between the field device and the maintenance service we have to provide service platforms to both the field device and the Maintenance Center (Fig. 16a,b).

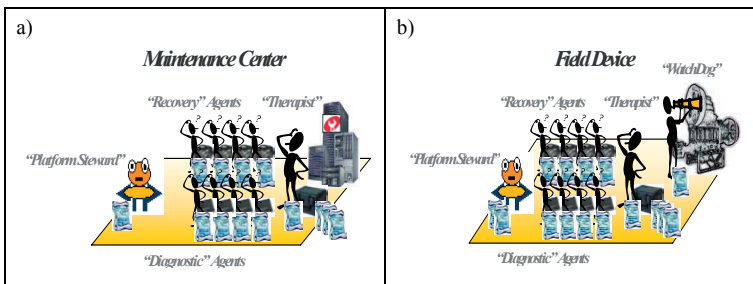


Figure 16. (a) External and (b) Internal Maintenance platforms

As we see, a Maintenance Center (Fig.16a) is a Maintenance Service based on Web service Platform. A “Therapist” agent represents this Maintenance Service. It has a set of subordinate agents. These are “Diagnostic” and “Recovery” agents, which represent two classes of services: Maintenance Diagnostic Service and Maintenance Recovery Service.

- **“Therapist” agent:** classifies input data by classes of maintenance diagnosis and checks conformity of incoming requests with the profiles of local agents;
- **“Diagnostic” agent:** returns the diagnosis given device condition parameters;
- **“Recovery” agent:** performs remediation given diagnoses.

All of these agents can learn and accumulate experience during their work.

A field device local maintenance service (Fig.16b) is based on an internal (embedded) service platform. Such platform can also host “Therapist”, “Recovery” and “Diagnostic” agents like an external service platform, however these agents have weaker knowledge and abilities than the agents in a Web-based service platform naturally having less experience and resources. Specific for a local device-based platform is a “WatchDog” agent (service). This agent is usually provided by the field device manufacturer. Its goal is to monitor some subset of critical system state parameters, detect relevant changes and query the internal “Therapist” agent for the Maintenance

Service. The “Therapist” agent examines the condition of the device and makes decisions about further actions. If a problem is detected, an action can be:

- Allowing local agents to be used for recovery (if appropriate);
- Requesting support from the Maintenance Network;
- Calling the maintenance center for the Emergent First Aid maintenance;
- Requesting for human intervention.

Human components in the Distributed System of Mobile Maintenance Services. Despite intense efforts to fully automate the maintenance activities, human involvement is still important. In the existing system of field device monitoring, information about device condition state is delivered to a human at the control panel, for further analysis and decision-making. In the proposed maintenance system, such a component like control panel exists also. This panel represents information about all processes, which take place within devices. This kind of a panel may be represented as a Maintenance Process Monitoring Service. This service, represented by a human, is a bridge between services that are responsible for interaction with field device (e.g. WatchDog), and maintenance services (diagnostic, recovery, etc.). The human can influence the processes in the field device via this service. Communication with the human component will be enabled via both the wire and wireless communications (Fig. 17).

Certainly a Maintenance System cannot perform without human resource execution especially in cases when a maintenance activity involves physical actions over a field device. Maintenance Crews can be located both in immediate proximity to a field device or in a remote Maintenance Center in a physical world and it is represented by human components (Fig.18). A human component like an agent component can provide services such as “diagnostic” and “recovery” however as it was mentioned above humans need adaptive interface to the Semantic Web environment.

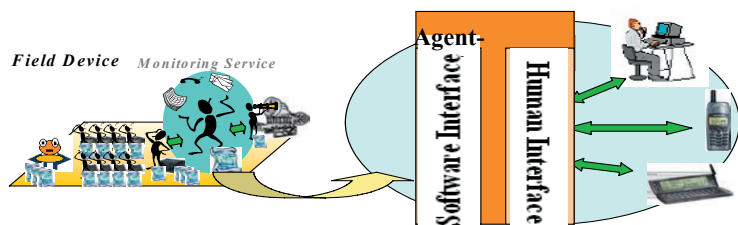


Figure 17. Human Monitoring Service

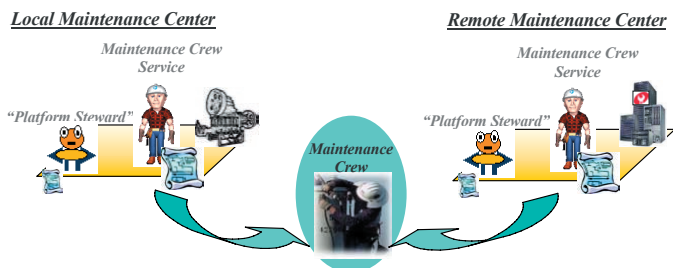


Figure 18. Human Maintenance Crew Service

Maintenance Network Management. Knowledge integration is an important requirement in industry as a whole and particularly in the product maintenance domain. Actually, the network of Maintenance Centers (in a common case, it is a network of Maintenance Services) provides such integration. Existing knowledge, which was previously isolated and inaccessible, now may be shared and reused based on a distributed environment of mobile (movable) semantically annotated services.

Maintenance Network services can be provided not just by the product's producers, but also by other knowledge providers in that domain. In this context we have to consider such questions as: how to launch a system of knowledge (service, experience) certification and how to manage business processes related to the utilization of commercial services. Network services have to be certified by a respectable and trusted certification instance for both: to perform specific maintenance activities for different products or to perform wide spectrum of maintenance activities for specific products. A certification system is a basis for guaranteed maintenance quality (Fig. 19).

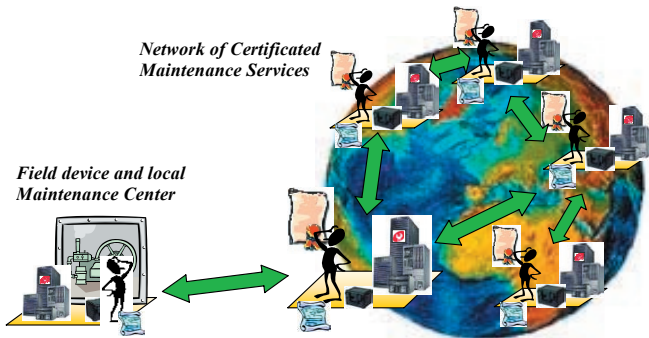


Figure 19. Network of certified Maintenance Services

Generally "Therapists" agents perform interactions in the Maintenance Network. Requirements to a "Therapist" agent as to a transaction manager include:

- Matchmaking between received service queries and profiles of the service components (agents) available at the platform;
- Targeted forwarding of the query to other platforms at the network, if the request cannot be served locally;
- Enabling peer-to-peer semantic search in the Maintenance Network.

We consider five types of product maintenance services and appropriate interaction scenarios between Field Device and Maintenance Center platforms:

- Service 1: Remote diagnostic
- Service 2: Recovery and predictive maintenance
- Service 3: Preventive inspection
- Service 4: Emergency service
- Service 5: Human resource execution

Remote diagnostics. Remote diagnostic is a case, when monitored parameters of a device differ from a normal state, and the local maintenance center does not have enough expertise to make a diagnosis itself. In this case the request with parameters will go from an internal platform to the Maintenance Center (MC). As a result, MC returns the diagnosis back to the internal platform. However if similar requests for diagnosis are sent very often, then it is considered to move an appropriate diagnostics agent, which is expert in

this repeating problem, permanently or for a certain time period to operate locally in the internal (embedded) platform (Fig. 20).

Recovery and predictive maintenance. Assume that a

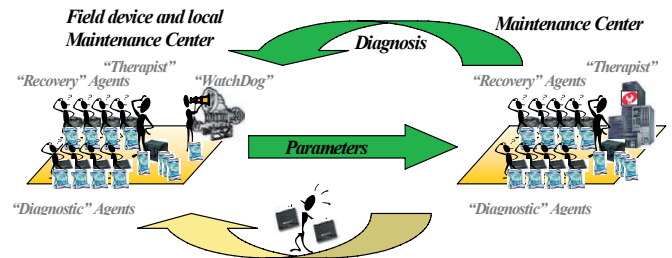


Figure 20. Remote diagnostics

local maintenance centre makes a diagnosis, but cannot recover the situation itself (e.g. there is no qualified "Recovery" agent). In this case, the internal platform sends a request with parameters and diagnosis to the Maintenance Center (MC). As a result, MC sends the appropriate "Recovery" agent to the internal platform, which can resolve the problem. This agent can accumulate experience during its work at the internal platform. If similar requests are sent very often, then it is also considered to send an experienced agent to the embedded platform for a permanent "job" if internal resources allow (Fig. 21).

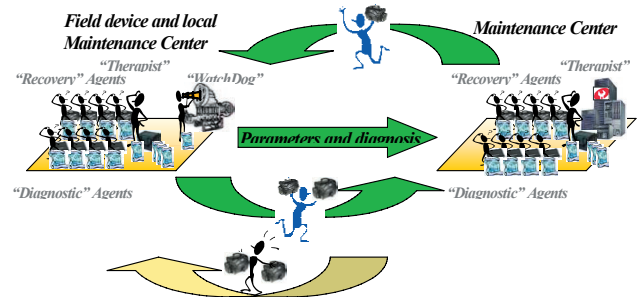


Figure 21. Recovery and predictive maintenance

Preventive inspection:

Case 1: Sometimes, when the Internal System requires preventive inspection, it sends this type of request and all necessary state data to the Maintenance Center. As a result, the MC sends its decisions from a set of "Diagnostic" agents, which are experts in all necessary fields for preventive inspection, to the internal platform (Fig. 22).

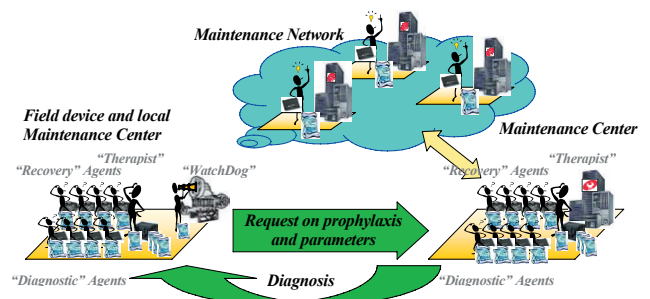


Figure 22. Remote preventive inspection

Case 2: The Internal System requests preventive inspection from the Maintenance Center and send the parameters. As a result, the “Therapist” in the MC gathers a group of agents (experts in the necessary fields) for preventive inspection and sends this brigade of “Diagnostic” agents to the Internal System. Locally they inspect Product and can reveal some troubles (Fig. 23).

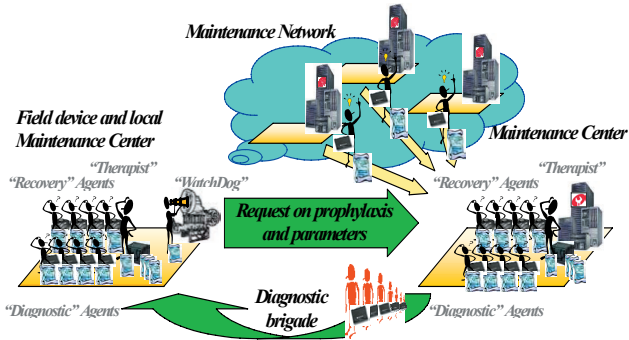


Figure 23. Local preventive inspection.

Emergency service. There is “First Aid” maintenance. If the diagnosis shows necessity of the emergency works (in critical states), the “Therapist” in the MC calls a group of “Recovery” agent(s) on-duty, using as much as possible the maintenance resources of its own MC, and sends this brigade to the Internal System as soon as possible. Also, it must continue to look for better experts for this problem in the Maintenance Network (Fig. 24).

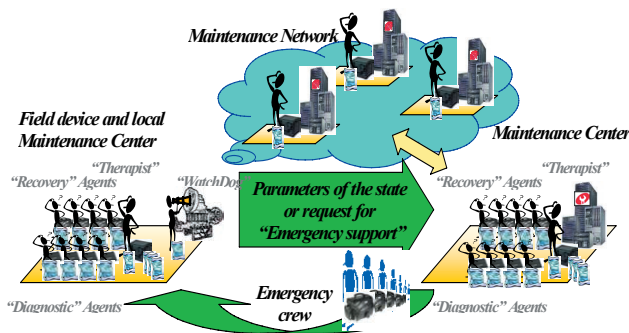


Figure 24. Emergency service

Human resource execution. There is a case of maintenance activities with people involved. If the “Recovery” agents cannot provide appropriate maintenance activities without human participation, then the “Therapist” checks the possibility of the local (human) Maintenance Crew to execute this type of activities or makes request for human advise to the Maintenance Network. The search is based on the profile of the required Maintenance Crew. Actually, some Maintenance Centers probably do not have their own crews. One of the important factors for a Maintenance Crew of humans to be taken into account is its location (Fig. 25).

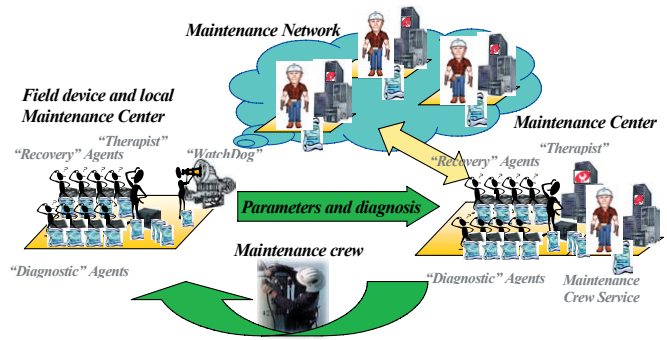


Figure 25. Human resource execution

5. Conclusions

In this paper we consider an infrastructure of distributed Web service components, which can be discovered in the Web based on semantic annotations, move to any target platform carried by mobile agents and perform their tasks locally and cooperatively. The challenge to use agents allows not only mobility of service components but also their learning while performing tasks locally. We are implementing this concept for automated monitoring and maintenance of field devices. A Model of Distributed Industrial Product Maintenance System based on interaction of heterogeneous distributed mobile Web services is described.

Resources and services (like subclass of the resources) are heterogeneous and need to be preliminarily adapted via a common ontology. According to this problem, we consider an OntoShell approach to an Ontology-based universal integration environment creation. It allows transforming all resources (already existing and being developed) to semantically enabled resources for their integration. Also, resources are distributed in the Web. Along with detached resources, there are also modular resources, which are components of other more complex. We consider services as mobile components to enabling effective integration of distributed resources. Mobile resources (services) are expected to be applied in domains where sharable semantically annotated distributed resources are utilized, i.e. for Semantic Web applications, particularly in industrial context. Field devices having the explicit physical contact to industrial processes are extremely important players to solve the productivity and quality tasks. It is very important to develop intelligent diagnostic solutions for automated monitoring and analysis of the field device needs. Effective utilization of existing and distributed knowledge in maintenance domain is one of emerging industry concerns. A Model of Industrial Maintenance System utilizes Semantic Web technology (ontological description and semantic annotation of service components); mobile agents approach with agents that are carriers of resources (services). Such system of mobile components integration (in the general case) provides a comprehensive approach to integration within enterprise, as well as with trading partners, suppliers, and customers, by offering latest technology and open standards. It provides possibility for organizations to create a cost-effective, extended enterprise by using an integration solution to get more return on information assets from existing ICT investments.

Acknowledgements. Authors are grateful to Dr. Jouni Pyotsia and his colleagues from Metso Corporation and

Metso business units for useful consultations and materials. Also we would like to thank our colleagues from Industrial

Ontologies Group (Oleksandr Kononenko and Andriy Zharko) for useful discussions within the scope of this paper.



Vagan Terziyan 1958, received his Engineer-Mathematician degree on Applied Mathematics from Kharkov National University of Radioelectronics in 1981. He became Candidate of Technical Sciences (Dr. Tech. equivalent) in 1985 and Doctor of Technical Sciences in 1993 (Dr. Habil Tech. equivalent) at the Software Engineering Department. He is acting as Professor on Software Engineering since 1994 and as the Head of the Department of Artificial Intelligence since 1997. Area of research

interests and teaching includes but not limited by the following: Intelligent Web Applications, Distributed AI, Agents, Multiagent Systems and Agent-Oriented Software Engineering, Semantic Web and Web Services, Peer-to-Peer, Knowledge Management, Knowledge Discovery and Machine Learning, Mobile Electronic Commerce. Recently he is working as Associate Professor in MIT Department, University of Jyväskylä (Finland) and as Senior Researcher at the InBCT (Innovations in Business, Communication and Technology) TEKES Project in Agora Centre and Head of "Industrial Ontologies Group". He has more than 100 scientific publications, more than half of them in internationally recognised magazines and conferences.



Oleksiy Khriyenko 1981, obtained his Engineer's degree in Computer Sciences on Intelligent Decision Support Systems in June 2003 from the Kharkov National University of Radioelectronics in Ukraine. Also he got Master of Science Degree on Mobile Computing at Department of Mathematical Information Technology, University of Jyväskylä in (Finland) in December 2003. He is a member of the "Industrial Ontologies Group" since January 2003 and researcher in Agora Center (Jyväskylä, Finland).

His research interests include: Artificial Intelligence, Semantic Web, Agent Technology, Web-Services, Distributed Resource Integration, and the application of these and new technologies. (<http://www.cc.jyu.fi/~olkhriye>).

References

1. A. Ankolekar, M. Burstein, J.R. Hobbs, O. Lassila, D.L. Martin, S.A. McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara, H. Zeng, DAML-S: Semantic Markup For Web Services, URL: <http://www.semanticweb.org/SWWS/program/full/paper57.pdf>, 2001.
2. 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.
3. B. Burg, "Agents in the World of Active Web-services" Hewlett-Packard Laboratories, 1501 Page Mill Road, MS 1U-16, Palo-Alto, CA 94304, URL: <http://www.hpl.hp.com/org/stl/maas/docs/HPL-2001-295.pdf>, 2002.
4. J. Clabby, Web Services Executive Summary, URL: <http://www-106.ibm.com/developerworks/webservices/library/ws-gotcha/?dwzone=webservices>, 2002.
5. 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.
6. "DAML-S: Semantic Markup for Web Services", The DAML Services Coalition 2002, URL: <http://www.daml.org/services/daml-s/0.7/daml-s.pdf>
7. J. A. Farrell, H. Kreger, Web services management approaches, IBM Systems Journal, Vol. 41, No. 2, 2002, URL: <http://www.research.ibm.com/journal/sj/412/farrell.html>.
8. "iPlanet Application Server Enterprise Connector for CICS", 2002, URL: <http://docs-pdf.sun.com/806-5504/806-5504.pdf>
9. Metso Automation's Customer Magazine Vol. 1, 2002 "Automation for the pulp & paper, energy, hydrocarbon and chemical industries, and rock and minerals processing", URL: [http://www.metsoautomation.com/automation/magazinebank.nsf/Resource/wwwAutom1_2002all/\\$File/wwwAutom1_2002all.pdf](http://www.metsoautomation.com/automation/magazinebank.nsf/Resource/wwwAutom1_2002all/$File/wwwAutom1_2002all.pdf)
10. Metso Automation Customer Magazine Vol. 10, Issue 1, 2003, "Automation. Metso Future Care is here today", URL: [http://www.metsoautomation.com/Automation/magazinebank.nsf/Resource/wwwAu103/\\$File/wwwAu103.pdf](http://www.metsoautomation.com/Automation/magazinebank.nsf/Resource/wwwAu103/$File/wwwAu103.pdf)
11. J. Nikunen, M. Salmenpera, H. Koivisto, Global Condition Monitoring Network, Automaatiopaivat 2001.
12. M. Ojala, SW agent technology in global field device management, URL: http://www.automationit.hut.fi/julkaisut/documents/seminars/sem_a01/Ojala.pdf, 2001.
13. 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.

14. H. Parunak, Practical and Industrial Applications of Agent-Based Systems, 1998.
15. C. Peltz, Applying Design Issues and Patterns in Web Services, URL: <http://www.devx.com/enterprise/Article/10397/0/page/1>, 7 January 2003.
16. J. Pyotsia, H. Cederlof, Advanced Diagnostic Concept Using Intelligent Field Agents, ISA Proceedings, 1999.
17. J. Pyotsia, H. Cederlof, Remote Wireless Presence in Field Device Management, ISA Proceedings, 2000.
18. J. Riihilahti, M. Ojala, On-line Diagnostics for Maintenance of Smart Field Devices, ISA Proceedings 2000.
19. J.M. Rodrigez, J. Sallantin, A system for Document Teleneegotiation (negotiator agents), COOP'98: 3rd International Conference on the Design of Cooperative Systems, Cannes, France, May 26-29, 1998, pp. 61-66.
20. A. Sheth, Semantic Metadata For Next-Gen Enterprise Information Integration: Gaining Industry-Specific Understanding of Heterogeneous Content, DM Review Magazine, April 2003, URL: <http://www.semagix.com/pdf/DM-Review-Final.pdf>.
21. UDDI. The UDDI Technical White Paper, URL: <http://www.uddi.org/>, 2002.
22. URL: <http://www.webservices.org>.
23. M. Wooldridge, An Introduction to Multiagent Systems, John Wiley & Sons, 340 pp.

УДК 020.23.25

МЕНЕДЖМЕНТ ЗНАНИЙ В СИСТЕМЕ ОРГАНИЗАЦИИ И ТЕХНОЛОГИИ ПОДГОТОВКИ СПЕЦИАЛИСТОВ НА ПРОФИЛИРУЮЩЕЙ КАФЕДРЕ

В.М. Левыкин*, В.Я. Терзиян,
А.Ю. Шевченко****

** институт компьютерных и
информационных технологий,
кафедра информационно-управляющих
систем Харьковского национального
университета радиоэлектроники,
пр. Ленина, 14, г. Харьков, Украина, 61166.*

*** кафедра искусственного интеллекта
Харьковского национального университета
радиоэлектроники, пр. Ленина, 14,
г. Харьков, Украина, 61166.
e-mail: vagan@it.jyu.fi
e-mail: shevchenko@sa.net.ua*

В работе рассматриваются вопросы проведения менеджмента знаний в системе организации и подготовки специалистов на профилирующей кафедре. Рассмотрены вопросы, связанные с построением и использованием онтологии предметной области. Данная работа является тематическим продолжением статьи "Комплексная система организации и технологии подготовки специалистов: проблемы реализации"

Введение

Растущие требования к повышению качества обучения обуславливают поиск новых подходов в организации учебного процесса, создание новых, более эффективных систем подготовки специалистов. Важным моментом в формировании таких систем является учёт

требований конечного "Заказчика" (предприятия, организации, фирмы). Именно "Заказчик" и определяет тот уровень подготовки специалистов, который ему требуется. Необходимо создать систему, способную гармонично учесть пожелания заказчика, существующие стандарты высшего образования и особенности конкретного ВУЗа. Одним из методов решения этой зада-