

Khriyenko Oleksiy

DISTRIBUTED MOBILE WEB SERVICES BASED ON  
SEMANTIC WEB

Master's Thesis

Mobile computing

22.3.2005

University of Jyväskylä

Department of Mathematical Information Technology

**Author:** Oleksiy Khriyenko

**Contact information:** Roninmäentie 1G 23b, 40500 Jyväskylä, FINLAND,  
olkhriye@cc.jyu.fi

**Title:** Distributed mobile web services based on Semantic Web

**Work:** Master's Thesis

**Pages:** 100

**Study Program:** Mobile computing

**Keywords:** Semantic Web, mobile web service, service component, mobile agent, integration environment, information integration, imbedded platform, maintenance, monitoring, remote diagnostic.

**Abstract:** The 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. To become an intellectual capital, a knowledge asset (Web resource or service) must be shared; it increases in value while being used. The research work is devoted to a problem of integration and an effective utilization of the distributed information resources, in particular for the industry needs in the field of the automated support and maintenance of field devices. The architecture of a global intelligent system of mobile service's components based on a semantic Web-enabled distributed integration environment is offered. The model of the distributed product maintenance system is developed on the basis of global intelligent system of mobile semantically annotated Web-services. A Model of Distributed Industrial Product Maintenance System based on interaction of heterogeneous distributed mobile Web services is described. Materials of this Master's Thesis underlie the papers: "Global Maintenance Network of Mobile Agent-Based Web Service Components" submitted to International Journal "Knowledge and Information System" (Springer, ISSN:0219-1377) and "OntoEnvironment: an integration infrastructure for distributed heterogeneous resources" submitted to PDCN2004 (the IASTED International Conference on Parallel and Distributed Computing and Networks).

## ACKNOWLEDGEMENTS

I am grateful to my supervisors: Ass. Professor (Yliassistentti) Vagan Terziyan from MIT Department (University of Jyväskylä) and Dr. Jouni Pyötsiä from Metso Corporation for useful consultations and materials. I would also like to thank my colleagues from the Industrial Ontologies Group (Vagan Terziyan, Oleksandr Kononenko and Andriy Zharko) for useful discussions within the scope of this work.

Also, I am thankful to Agora Center and especially to professor Pekka Neittaanmäki for so important support during the term of writing the Master's Thesis. Extra, I would like to thank the leadership of the InBCT project (Agora Center) for provided assistance.

I would like to thank the Kharkov National University of Radioelectronics and the University of Jyväskylä for the possibility to be a participant of Master's program.

Especially, I would like to express my deep appreciation to Helen Kaykova and Vagan Terziyan for their support and useful advices, which are so important for me, and my living in Finland.

Also, I am thankful to Ritva Weber for so painstaking language check of my Master's Thesis.

Finally, I want to dedicate this Master's Thesis to my parents, who are far from me now, but always with me in my mind and my heart.

*University of Jyväskylä, 22.3.2005*

## TERMS AND ABBREVIATIONS

ACL	Agent Communication Language
AI	Artificial Intelligence
AS	Agent-Shell
ASP	Agent-Shell Platform
DAML	DARPA Agent Markup Language
DAML-S	DAML-based Web service ontology
FIPA	Foundation for Intelligent Physical Agents
HTTP	Hypertext Transfer Protocol
ICT	Information and Communication Technology
IOP	Internet Inter-ORB Protocol
KQML	Knowledge Query and Manipulation Language
LAN	A local area network
MC	Maintenance Center
Metadata	Data that describes other data. Often deals with the format or authorship of the underlying data
OIL	Ontology Inference Layer
OMG	Object Management Group
Ontology	A conceptual representation of the entities, meanings, and relationships within a specific domain of knowledge. See RDF, W3C, Metadata, OIL, Semantic Web, DAML

P2P	Peer-to-Peer – computing paradigm where each node in the network can be at the same time client and server
RDF	Resource Description Framework
RDFS	RDF Schema
Semantic Web	A conceptual web built on top of the World Wide Web in which all identified resources will be machine-processable
SOAP	Simple Object Access Protocol
SW Agent	Software Agent
UDDI	Universal Description, Discovery and Integration
W3C	Worldwide Web Consortium
WAP	Wireless application protocol
WSDL	Web Services Description Language
WSEL	Web Services Endpoint Language
WSFL	Web Services Flow Language
XLANG	An extension of WSDL
XML	Extensible Markup Language
UML	Unified Modelling Language

# CONTENTS

<b>1</b>	<b>INTRODUCTION .....</b>	<b>1</b>
1.1	VALUE OF THE PROBLEM DOMAIN.....	1
1.2	THESIS GOAL .....	6
1.3	RELATIONS WITH OTHER PROJECTS .....	6
<b>2</b>	<b>SEMANTIC WEB AND AGENTS APPROACHES IN WEB SERVICE TECHNOLOGY .....</b>	<b>8</b>
2.1	WEB SERVICE INFRASTRUCTURE .....	8
2.1.1	Web Service is ... ..	8
2.1.2	Service Properties and Types.....	10
2.1.3	Service Architectures .....	15
2.2	SEMANTIC WEB AND WEB SERVICES.....	17
2.2.1	SOAP, WSDL, UDDI technologies.....	17
2.2.2	Semantic Markup of Web Services .....	22
2.2.3	Semantic Web-enabled Services.....	24
2.3	AGENT APPROACH TO WEB SERVICES .....	24
2.3.1	Agent Technology.....	24
2.3.1.1	Software Agents.....	24
2.3.1.2	How Are Agents Built up?.....	27
2.3.1.3	Why Agents?.....	28
2.3.1.4	Multi-agent Systems .....	29
2.3.2	Agents and Semantic Web Services .....	29
2.3.2.1	The Semantic Layers of Agent-based Web-services Communications...30	
2.3.2.2	Active Web-services .....	31
2.3.2.3	Agent Standardization.....	33
2.3.2.4	Agent Environment.....	34
<b>3</b>	<b>ONTOLOGY-BASED DISTRIBUTED INTEGRATION ENVIRONMENT FOR HETEROGENEOUS RESOURCES .....</b>	<b>36</b>
3.1	ONTO SHELL APPROACH TO THE PROBLEM .....	36

3.2	INTERACTION MODELS .....	41
3.2.1	Centralized Interaction Model .....	41
3.2.2	Decentralized Interaction Model .....	42
3.2.3	Hybrid Interaction Model .....	42
3.3	MOBILITY .....	45
3.4	BUSINESS MODEL .....	46
3.4.1	Patterns of Behavior for Elements of OntoEnvironment.....	46
3.4.2	Business Relations between Players .....	48
<b>4</b>	<b>MOBILE (MOVABLE) WEB SERVICE BASED ON A SEMANTIC WEB .....</b>	<b>50</b>
4.1	NECESSITY OF MOBILE WEB SERVICES .....	50
4.2	EQUIPMENT FOR MOBILE (MOVABLE) WEB SERVICE.....	52
4.3	SERVICE NETWORKS BASED ON THE AGENT-SHELL PLATFORM APPROACH .....	53
4.4	WHO PROVIDES WEB SERVICES AND FOR WHOM .....	57
<b>5</b>	<b>GLOBAL INDUSTRIAL MAINTENANCE NETWORK BASED ON MOBILE</b>	
	<b>WEB SERVICES .....</b>	<b>62</b>
5.1	INDUSTRIAL PRODUCT’S MAINTENANCE .....	62
5.1.1	Necessity of Industrial Maintenance.....	62
5.1.2	Field Devices Maintenance.....	63
5.1.3	Increasing Information Flow .....	64
5.1.4	Existing Industrial Autonomous Maintenance Systems .....	64
5.1.4.1	Neles FieldBrowser Features .....	65
5.1.4.2	Central Management of Control Valves .....	67
5.1.4.3	Metso Automation’s Field Bus Products .....	68
5.2	DISTRIBUTED MOBILE MAINTENANCE SYSTEM FOR SMART-DEVICE .....	70
5.2.1	Service Requestor Is a Smart-Device .....	70
5.2.2	Intelligent Distributed Product’s Maintenance .....	70
5.2.3	Structure of Maintenance Web Service Platforms (Internal and External systems) .....	73
5.2.4	Human Component in the Distributed System of Mobile Maintenance Services .....	74

5.2.5	Maintenance Network.....	76
5.2.6	Maintenance Cases .....	77
5.2.6.1	Remote Diagnostic.....	78
5.2.6.2	Recovery and Predictive Maintenance .....	78
5.2.6.3	Preventive Inspection.....	79
5.2.6.4	Emergency Service .....	80
5.2.6.5	Human Resource Execution.....	81
<b>6</b>	<b>CONCLUSIONS .....</b>	<b>83</b>
	<b>REFERENCES.....</b>	<b>86</b>



# 1 Introduction

## 1.1 Value of the Problem Domain

Nowadays, knowledge is one of the most valuable resources of enterprises and an important production and competition factor. Therefore, in a globalizing and growing market the optimal usage of existing knowledge represents a key factor for enterprises of the future.

Knowledge assets are the knowledge regarding markets, products, technologies and organizations, that a business owns or needs to own and which enable its business processes to generate profits, add value, etc. Knowledge management is not only about managing these knowledge assets but also managing the processes that act upon the assets. These processes include: developing knowledge; preserving knowledge; using knowledge, and sharing knowledge. Therefore, Knowledge management involves the identification and analysis of available and required knowledge assets and knowledge asset related processes, and the subsequent planning and control of actions to develop both the assets and the processes so as to fulfil organizational objectives.

In knowledge-based industries, especially within the service sector, a company's primary assets are intellectual. Such organizations are more dependent upon ideas, concepts, and expertise than traditional capital assets: manufacturing plants, trucks, warehouses, or machinery. Knowledge-based assets, or intellectual capital is the sum total and accumulated value of the company's shareable knowledge and expertise. For knowledge to become intellectual capital, it must be shared. Information sharing is critically important, because intellectual assets, unlike physical assets, increase in value with use; 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.

*Why is knowledge sharing so important?* Expertise often lies in only a few individuals' heads and their expertise is not always easily accessible. Experts are busy. Newly hired

individuals may not know who the experts are. Companies can be temporarily crippled if key experts die, retire or resign. Valuable knowledge is in their minds and when they leave, they take their minds with them. Knowledge sharing takes an expert's tacit knowledge and makes it explicit.

Support for information and knowledge exchange is a key issue in the Information Society. The exponential growth of online information on intranets and the Web leads to information overload. To cut down on the time wasted in searching and browsing, and reduce associated user frustration, much more selective user access is needed. This is possible by automatic meaning-directed or semantic information processing of online documents.

Knowledge is becoming more and more important in our daily private and business life. The next generation of knowledge management systems will have to integrate different methods and techniques from the following different research communities and fields to achieve the vision of ubiquitous knowledge: Ontologies & Semantic Web, Knowledge Discovery & Business Intelligence, Mobility, Processes & Groupware [VISION]. Now it has been widely recognized that knowledge management systems must rely on a common knowledge structure, ideally in the form of ontology. Ontologies provide a common language on the human and machine level to enable knowledge exchange. 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.

To keep control of the enterprise's intellectual capital, companies have to organize its information resources, address knowledge workers' needs, and prepare for integrated business applications. In addition to the existing problems with management and the flow of intellectual capital, enterprises will face two new trends that risk furthering the erosion of enterprise control of intellectual capital. At a global level, we expect enterprises will accelerate the offshore sourcing of knowledge work wherever it is convenient and efficient to do it. At a local level, individual knowledge workers will depend increasingly upon their personal knowledge networks to increase their own productivity. Both trends will decrease

the level of direct control of intellectual capital by enterprises. To cope, they will need to improve both intellectual capital management and knowledge management processes dramatically.

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 a company's ability to compete. Nowadays, in time of the next generation of integration, enterprises have to utilize their in-place ICT assets to maximize the return on those investments and to streamline business processes. As a result, they will become more agile, efficient, and responsive – all key elements to success in the current business atmosphere. A successfully deployed integration network can: provide the agility for company needs to respond quickly and effectively to capture business opportunities, simplify the business process and shorten business cycles to drive down costs, leverage the company's vast ICT expenditures to realize real return on these investments [webMethods].

Enterprises are realizing how important it is to "know what they know" and be able to make maximum use of the knowledge. This knowledge resides in many different places such as: databases, knowledge bases, filing cabinets and peoples' heads and are distributed right across the enterprise. All too often one part of an enterprise repeats work of another part simply because it is impossible to keep track of, and make use of, knowledge in other parts. In addition, more and more companies are looking towards strategic partnerships and alliances to gain market share. As a result, it has become absolutely critical to integrate business processes and applications across the extended enterprise, which includes employees, customers, partners, and suppliers.

The benefits of the next generation of integration can only be realized with an integration solution that presents the leading edge of integration technology, a solution that provides the comprehensive, enterprise class integration capabilities necessary to power an enterprise [Tommila et al., 2001].

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 ('enhancing') 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. Metadata describes contextually relevant or domain-specific information about content based on a custom (e.g., industry-specific or enterprise specific) metadata model or ontology (is known as semantic metadata).

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. 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 solve the integration problem, there have been several point solutions in the market. To build Web services through integration requires an infrastructure that enables end-to-end business processes. Applications should integrate easily and painlessly. This means a solution built on standards. We have to make solutions that solve the current integration problems (integration of traditional applications, Web applications, and Web services) — as well as pave the way for future Web services.

Web services have the potential to fundamentally transform the way companies do business. Many companies are only now exploring Web services. The Web services-based solution provides seamless applications integration as services across the Internet. Web Services deliver a better integration solution because they are based on open standards, which are easy to use and widely supported. Web Services are the next logical extension of enterprise integration, because the technology standardizes communication, description, and discovery mechanisms. Web Services promise the ability to combine individual services into more complex, orchestrated services that will provide sophisticated business

process and workflow automation capabilities to the enterprise. However, such composition and orchestration is still on the drawing board.

It is possible to say, that Web Service adds up to, in the general case, knowledge, in a sense resource. Today, enterprises are interested exactly in integrating their resources via services' integration. Especially when they want to organize partnership, business combination or diversified group for using open to general use, shared resources for business keeping in new way, in way of reduction inputs (costs) and effective resource using. Today, when knowledge resources of business life participants are distributed across the Internet, necessity of shared resource integration for common using is emerging. Web Service using covers partially the solutions of this kind of problems. But as before, we need technologies and solutions, which can make possible distributed resources integration.

Exactly, one of such solutions is using the System of Platforms for Mobile Agent-carriers of Web Services (resources), which is based on the OntoShell approach (chapter #3) and described in chapter #4. It is a system of mobile integration of decentralized resources. This approach can easily find its use in both individual using of distributed resources and making system centralized using of decentralized resources, like in the case of making Industrial OntoHub for cooperative resource using by the members of an enterprise association.

This kind of knowledge resource integration can be used in many domains, and especially in the industrial domain. For example, it is a very important solution for the industrial product's maintenance domain, where the integration of distributed knowledge in maintenance field plays an important role in effective product's maintenance activities for sustenance effectiveness of industrial process. In chapter #5, Distributed Industrial Maintenance System based on the Semantic Web approach and network of platforms for agent-carriers of mobile service components (knowledge in product's maintenance domain) is described.

## 1.2 Thesis Goal

The goal of the present thesis is: model of the automated intelligent information exchange and information integration system development; model application for the domain, where this kind of system is called for nowadays. The task of the work is: development of a Semantic Web-enabled distributed integration environment, a distributed system of mobile service components architecture design, principles description of structural component's interactions and functioning in such system; considering of existing techniques in the industrial maintenance domain, model elaboration of an automated intelligent distributed maintenance system based on the distributed mobile semantically annotated Web Services architecture, description of the set of maintenance cases; review of the techniques used for the development of this kind of knowledge integration systems.

## 1.3 Relations with Other Projects

This idea comes from the OntoServ.Net concept developed by Industrial Ontologies Group. OntoServ.Net is a large-scale automated industrial environment for assets management. First of all, we consider the maintenance of assets, but, in general, this concept can be applied for process control, improvement of operating efficiency, field-performance diagnostics, plant-level management, etc., as well.

Better maintenance provided by OntoServ.Net considers maintenance information integration, better availability of operational data and shift from reactive and preventive maintenance towards predictive and proactive maintenance, which means, first of all, reduced Total Life Cycle Cost of machines.

OntoServ.Net is a network of industrial partners, which can share maintenance methods and information developed during the work of a separate machine (device, equipment, installation). Improved locally, maintenance experience can be shared. Also, it is assumed that there are special commercial maintenance centers supported either by manufacturers of machines, or by third parties. Browsing the internal state of a device is extended to an automatic diagnostics and recovery within a network of a maintenance centers. The role of

maintenance center, firstly, is to organize the gathering and integration of field data and maintenance methods improvement, and secondly, support its clients by providing better services (remote diagnostics, consulting) and upgrading local maintenance systems of devices.

This Master's Thesis is closely related to the theses of my colleagues from Industrial Ontologies Group (Oleksandr Kononenko and Andriy Zharko), which also concern the semantically-enabled resource integration approach. The Master's Thesis of Oleksandr Kononenko ("Ontological Support for Industrial Maintenance of Smart-Devices") concerns ontology application for information integration in industry. And the work of Andriy Zharko ("Peer-to-peer ontological discovery of mobile services components in Semantic Web") is devoted to a problem of resource (service component) search, discovery and routing in a peer-to-peer network.

## 2 Semantic Web and Agents Approaches in Web Service Technology

### 2.1 Web Service Infrastructure

#### 2.1.1 Web Service is ...

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. These web-services can reason about their capabilities to combine services and negotiate. Web Services is a set of standards that are being designed and specified by the Worldwide Web Consortium (W3C) to foster cross-platform application-to-application communications. 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 were supposed to provide a systematic and extensible framework for application-to-application interaction. However, due to the complexity of the web-services, there is a flurry of standards and software in competition and deployment becomes a key issue.

There are a lot of existing definitions for Web Services:

- It is software designed to be used by other software via Internet protocols and formats (Forrester).
- It is a self-describing component that can discover and engage other web services or applications to complete complex tasks over the Internet (Sun Microsystems, Inc).
- It is a loosely coupled software component delivered over the Internet via standards-based technologies like XML and SOAP (Gartner).
- It is a self-describing, self-contained, modular unit of application logic that provides some business functionality to other applications through an Internet connection... (UDDI.org)



- It is an Internet-based, modular application that performs a specific business task and conforms to a particular technical format (IBM).
- It is an application logic that is programmatically available, exposed using the Internet (Microsoft)

Web-services are flexible, Internet-based applications that allow companies to create new products and services faster than other existing methods which consist of dynamic assembly of loosely coupled components (e-services, legacy data...). This is very different from the traditional hard-wired approach for developing applications. Fixed applications tend to resist change, whereas web-services assume that change is ever present. Web-services require research in: explicit representations of e-services and their capabilities; their re-use in different contexts to form new and dynamic services; the creation of a heterogeneous and competitive environment; reputation networks, negotiation, contracts [Bernard, 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].

One of the benefits of the Web Services architecture implementation is the cost reduction for doing business electronically. Web Services allow companies to deploy, implement and integrate their new solutions faster because of the inculcation of the common application-to-application communication model. Web Services will enable Internet to become a global common platform where organizations and individuals will communicate to carry out various commercial activities and provide value added services [Fensel & Bussler, 2002].

A fundamental aspect of Web service design is interoperability. For a company's Internet applications to be most effective, Web services must interface seamlessly internally and, potentially, externally with partners, suppliers, and customers. But, these entities may not

have the same sophistication when developing their Web services, or they may have different XML representations of the same business data. With this in mind, interoperability must be designed into the architecture, not left up to chance [Peltz, 2003].

### **2.1.2 Service Properties and Types**

This section is based on materials from [Fethi, 2002].

Generally, a given resource can offer one or more services, and also access one or more services on another resource. Similarly, a service can access one or more services either on the local or a remote resource, and may be accessed by one or more services. Each service within this model can support two kinds of interfaces:

- A functional interface, which defines how the service is to be accessed and executed.
- A management interface, which defines parameters associated with service execution, licensing, cost etc. The management interface is used to differentiate between multiple resources offering a similar type of service, and generally corresponds to the non-functional attributes of a service.

Based on the *functional* and *management* interfaces to a service, we can define a number of ‘roles’ that may be performed as parts of a given service:

1. A Service user can request a service available at a local or remote host. The service user is responsible for initiating and terminating the service, and dealing with exceptions locally that are generated from the service.
2. A Service provider can generate service offers, and is responsible for establishing a service contract with a user. The service provider offers an interface for invoking a service, along with specification of parameters associated with managing the service.
3. A Service broker may be used to discover services based on one or more criteria. The broker acts as an intermediary between a service user and a service provider, and primarily supports service registration. The broker may also provide a ‘matchmaking’ service to help a user locate a service of interest. We use the term “broker” as a common intermediate service provider offering services of varying

complexity – ranging from security, service decomposition and service scheduling. An important part of this role is the provision of a discovery service interface – which enables a service provider to make itself known to a service user, and for a service user to identify its requirements. A broker may utilize the following properties to support discovery:

- Security: service security can vary from access rights (levels) to trust models that enable only a particular category of users to run the service.
- Cost: service cost can be associated with the management interface of a service, and correspond to computational time or access time, or access cost. Existing software tools such as Nimrod provide mechanisms to utilize such a parameter in selecting a service. Many resource providers at national centers also operate in this way, providing time on a computational resource or a percentage of a resource for a particular cost.
- Fairness: every service should be accessible from other services over a particular period of time. Service fairness issues arise when a particular service is prevented from being accessed (due to factors other than cost, security or performance issues). The issue of fairness is more complex when a service agent acts on the behalf of a service provider.
- Performance: the management interface of a service can be used to support service performance, and support a broker in discovering a service of interest. A service may also support additional levels of performance information, derived from analytical models of the service itself. For instance, if a service is to be run on a particular host, information about the host can be used to determine possible run times for a service with a given quantity of data. This information is also made available to a broker – and certain brokers may only select services, which publish such information. It is up to the broker to decide how to use this information.

As can be seen from some of the criteria discussed above, a broker undertakes many complex but related roles. A general system is likely to have many brokers, each undertaking roles determined by application user demands, and the differences in representation schemes employed within the system. Results generated by all the brokers are however coordinated through a central authority – generally the user

application service that initiated a request on the broker. Hence, there may be a broker to support service registration, a broker to support service discovery based on performance, service discovery based on cost etc.

4. A Service adapter is used to enable a service provider or user ‘wrap’ a given software library or application, and make this available as a single service. Many existing applications in Fortran/C, for instance, would need to be wrapped to enable them to be made available as a service. A service adapter is also used to ensure that the service provider, and the service requester respect the activation policy associated with a service. A service adapter must also ensure that functional dependencies between the called software libraries that constitute the service are followed. Hence, a call to a given service may include the execution of an initialization code, followed by the execution of a numeric solver, followed by the execution of code to record the results. The service adapter hides this level of detail from the service user. Service wrapping may be performed at the source code level (where this is available) or may involve adding an execution shell around an existing pre-compiled binary. Wrapping from source code is generally much more complex, as it involves a number of language specific issues, such as ensuring that type conversion does not modify the accuracy of the produced results. Wrapping from source code also requires the wrapping tool to have some knowledge of the structure of the application, also is likely to be a lengthy and error-prone process.
5. Service aggregators/decomposers are specialized brokers, which can decompose or combine a service request to sub-requests to find better matches for service providers. It is possible for a popular service provider to be overloaded with requests, or for there to be no single service provider, which can complete a given request. Service decomposers enable a given service request to be divided based on the available service providers, and for the results of these requests to then be combined before being returned to the user. Service decomposers can utilize domain specific ontologies to determine alternative service providers of interest, and enable these requests to be forwarded. The approaches adopted depend on the representation scheme used to encode the ontology, and the service definitions.

The aggregator/decomposer must use the same representation way as the service discovery agents – and confirmed through an initial message exchange.

6. Service discovery is the most important part of the process, and is responsible for finding a match between a service request and a service provider. This matchmaking can be supported through a number of possible criteria – as discussed previously – and it is possible for multiple service discovery agents to co-exist. A user may launch the same query to multiple such agents concurrently. A service discovery agent may utilize a syntax match, a context match, or a semantic match. A syntax match would involve an exact textual comparison of the request, with the advertisement from the service providers. The other two approaches are based on the availability of a domain or problem specific ontology, which may be used by the discovery agent to resolve a given request. A “context” match would involve finding some similarity between the request and service providers that the service discovery agent is aware of. A context match is based on analyzing other requests made by the same user previously in order to find a domain context for the current request. A domain context could enable a discovery agent to forward requests to particular service providers. A “semantic” match would involve navigating the domain ontology, or relaxing domain constraints, to find suitable candidates that could be queried to find the required service.
7. A Service optimizer enables a group of service providers to work collectively to improve their cost, security, or performance. A service optimizer may be used to improve the behavior of a resource cluster by sharing of common requests. A service optimizer may also be used to reserve a single service in advance, or make a reservation of a group of services, over a particular time frame. The service reservation mechanism is used to ensure that if a service aggregation/decomposition is to be performed, and then all sub-services will be available.
8. A Service execution agent works with the broker and the user application to launch one or more tasks on the identified computational resources. The service execution agent primarily acts as an interface to third party resource execution

systems – and does not directly undertake any scheduling on the remote resources itself.

9. A Reputation Service may be employed by a broker to rate a computational service. The rating function can be determined by the broker service, or it may be suggested by the application service needing resources. The rating service is used to provide each broker with a historical view of the available computational services, and enable a broker to filter service offers made by these resources. A broker undertaking service discovery may also query other brokers to determine similarities in their ratings of a given resource. Similarly, a computational resource may also wish to advertise its own ratings to a broker – which a broker may wish to ignore, or aggregate with its own results. To support a Reputation Service, each broker must be able to monitor the use of a resource by an application service, and be able to record these results locally. A broker utilizing this service must also be able to modify its database if a computational service migrates, or modifies its properties.
10. A Mobile Service is not tied to a particular host, and may be migrated on demand. Service migration shares many ideas with object migration – in object migration based on a call-by-value semantics, for instance, the state of an object is sent to a remote location to create a new instance of the object. The new instance now has a separate identity, and does not maintain links with the parent. As it is necessary for the receiving side to instantiate an instance, it must necessarily know something about the object's state and implementation – such as whether data members are private or public for instance. Service migration is defined at a coarser level of granularity to object migration, and consequently, may only involve the partial migration of state to the remote host. A service may be an aggregate of a number of other services, and consequently would require the migration of the complete dependency graph to the new host. By utilizing a combination of call-by-value and reference semantics, a mobile service is able to create a new instance at the remote site of the service.

### 2.1.3 Service Architectures

The standard Web Service architecture can be described as an interaction of the three main entities: service provider, service registry and service requestor. Figure 2.1 illustrates this interaction.

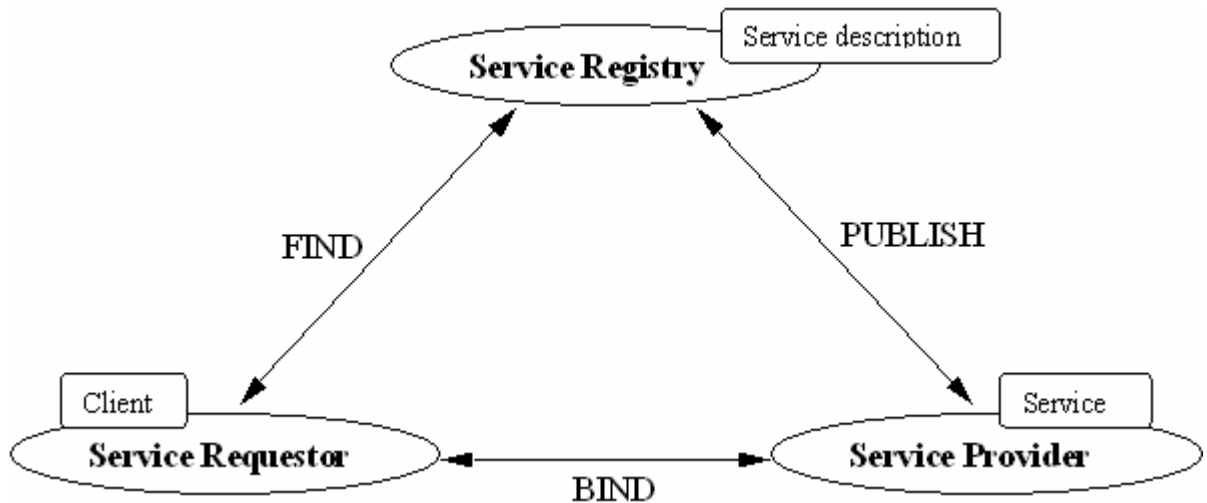


Figure 2.1. Web Services, entities interaction model.

The roles of the Web Services architecture nodes:

- Service Provider is the owner of a specific service. The Service Provider can also be described as a host that contains access to the service. The Service Provider tries to advertise its service so that the service requestor can find it. The place where the advertisement can be published depends on the concrete situation or application requirements. The standard way implies the Service Provider to publish some kind of a service description at the Service Registry.
- Service Registry is a searchable entry point where service providers publish their advertisements or service descriptions.
- Service Requestors are Web-based applications and services that are looking for interaction with the necessary service to obtain specific information or to fulfill a specific task. So, a Service Requestor queries the service registry for the specific type of service. When the necessary information is obtained, the Service Requestor invokes the service or performs interaction with the service. However, the Service Registry is an optional entity of the architecture, because the Service Provider can publish service

description at another accessible point or send it directly to the Service Requestor. Moreover, the Service Requestor can obtain a service description from other sources besides Service Registry, such as local file, web site, ftp site.

The operations of the nodes in the Web Services architecture:

- Publish – a service description needs to be published so that the service requestor can easily find it.
- Find – is an operation used by the service requestor to retrieve a service description and consume it. The Service requestor can retrieve a service description at design time or runtime from a service description repository, a simple service registry or a UDDI (Universal Description, Discovery and Integration) node [Kreger, 2001].

The Bind operation enables the service requestor to invoke or initiate interaction with the service at runtime using the details in the service description to locate, contact and invoke the service [Kreger, 2001].

Also, a Web Service instance can serve multiple roles simultaneously. In the peer-to-peer scenario, each peer Web Service instance serves in both the Service Requestor and Service Provider roles (Figure 2.2).

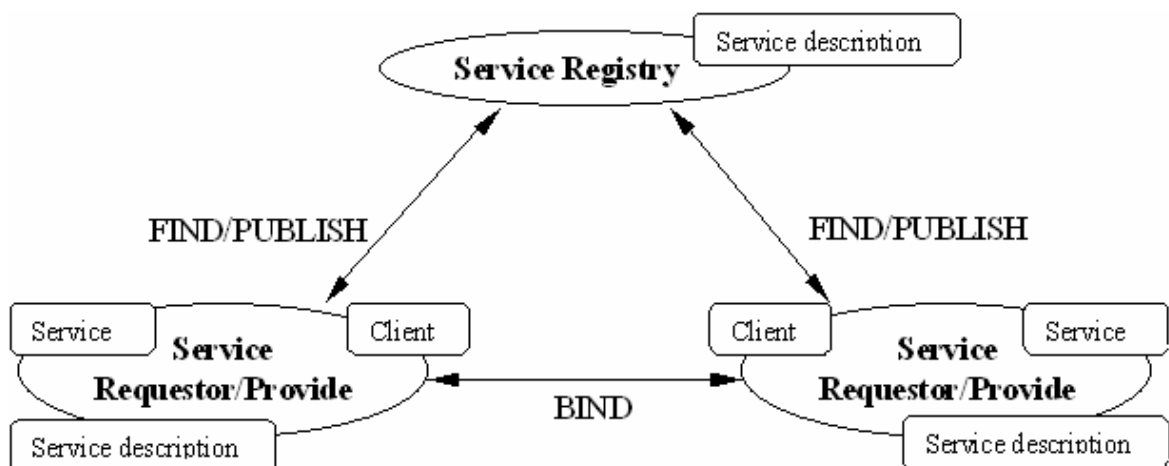


Figure 2.2. Web Services architecture – Peer-to-Peer derivative pattern.



Of course, we also have to consider the case, where the Service Requestor and the Service Provider interact directly (Figure 2.3).

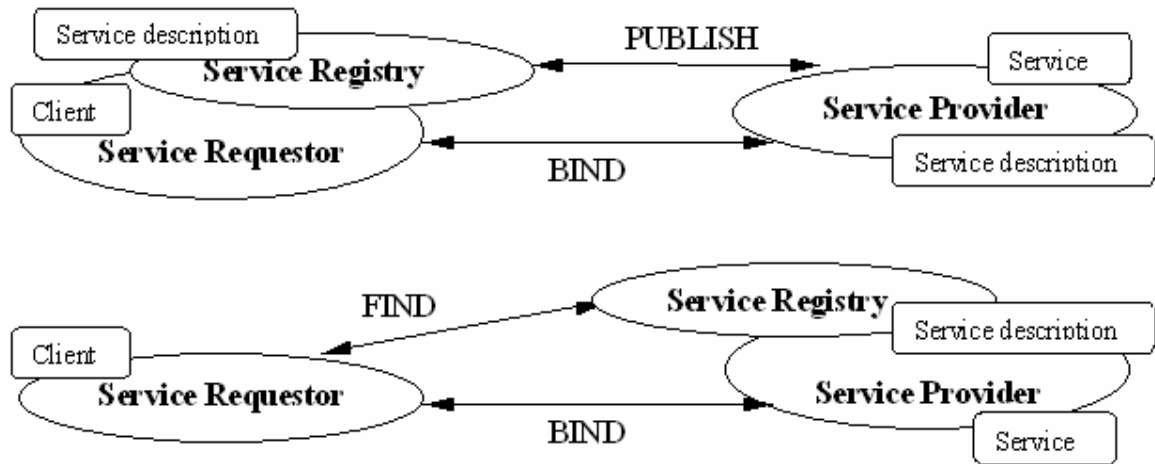


Figure 2.3. Web Services architecture – Direct Interaction derivative pattern.

## 2.2 Semantic Web and Web Services

### 2.2.1 SOAP, WSDL, UDDI technologies

Web services represent an evolution of the Web to allow applications to interact over the Internet in an open and flexible way. Important in this approach is the independence of the interactions from the platform, programming language, middleware, and implementation of the applications involved. Web services are self-contained, modular applications that are described, found, and called via a set of standards based on Extensible Markup Language (XML). Chiefly the World Wide Web Consortium (W3C) is formalizing these standards.

Semantic Web is widely regarded as the next step in the evolution of the World Wide Web. The main goal of Semantic Web is to make web documents content-explicit to computers. Semantic Web introduces a way to encode the content of pages in a machine-readable format and link this content to machine-understandable semantics using ontologies [Ding et al., 2002], [Fensel & Musen, 2001]. Ontologies will be the key for the semantic web development [Fensel et al., 2002]. The reason to use ontologies as it was mentioned in [Fensel et al., 2002] is largely due to what they promise: a shared and common

understanding of a domain that can be communicated between people and application systems. Thus one of the goals of the Semantic Web is to enable access to heterogeneous and distributed information all over the Web by enabling software agents to mediate between the user and the information [Davis, 2002].

The current situation on the Internet is that information is primarily in the form of pages composed of human-readable information. Semantic Web is aimed to augment this information with markup that enables machine understanding of the marketed information. This case faces the need for a markup language that supports defining data models or ontologies [Shah et al., 2002].

The Web Services framework is also represented by a set of standards: SOAP, WSDL and UDDI.

- **SOAP** Simple Object Access Protocol (SOAP) provides a simple and lightweight mechanism for exchanging structured and typed information between peers in a decentralized, distributed environment using XML. SOAP does not itself define any application semantics such as a programming model or implementation specific semantics; rather it defines a simple mechanism for expressing application semantics by providing a modular packaging model and encoding mechanisms for encoding data within modules. This allows SOAP to be used in a large variety of systems ranging from messaging systems to RPC. SOAP consists of three parts [W3C, 2000]:
  - The SOAP envelope construct defines an overall framework for expressing what is in a message; who should deal with it, and whether it is optional or mandatory.
  - The SOAP encoding rules define a serialization mechanism that can be used to exchange instances of application-defined data types.
  - The SOAP RPC representation defines a convention that can be used to represent remote procedure calls and responses.
- **WSDL** Web Services Description Language (WSDL) is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and

message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services). WSDL is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used to communicate, however, the only bindings described in this document describe how to use WSDL in conjunction with SOAP 1.1, HTTP GET/POST, and MIME [W3C(a)].

□ **UDDI** UDDI (Universal Description, Discovery and Integration) is an initiative proposed by Microsoft, IBM and Ariba to develop a standard for an online registry, and to enable the publishing and dynamic discovery of Web services offered by businesses. UDDI allows programmers and other representatives of a business to locate potential business partners and form business relationships on the basis of the services they provide. It thus facilitates the creation of new business relationships [Ankolekar et al., 2001].

There are building blocks for services. Figure 2.4 shows the relationship between these building blocks.

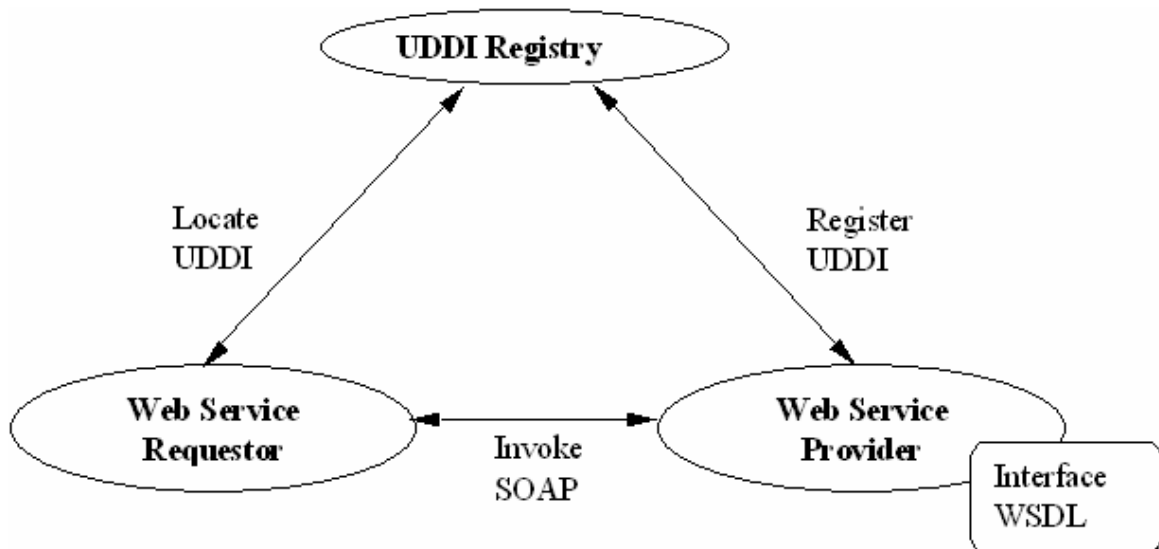


Figure 2.4. Building Blocks. This graphic shows the relationship between SOAP, WSDL, and UDDI, the building blocks of Web services.

The interface of a Web service is described in an XML format called the Web Services Description Language (WSDL). A WSDL file contains descriptions of one or more interfaces and binding information for one or more services [Farrell & Kreger, 2002].

WSDL is the key to managing interoperability. It provides contracts for describing the interfaces to both new and existing applications. This allows multiple organizations to standardize on an interface to a service, without having to worry about the underlying implementation. Another benefit of using WSDL is that it is the focal point for many of the Web service tools on the market today. Most tools provide a way to automatically generate client code from an existing WSDL document. This can end up saving developers development effort because they do not need to write any of the SOAP messaging code [Peltz, 2003].

The UDDI project, UDDI.org, is defining such a Web services registry. Once a service has been discovered, and a binding established based on the information in the registry, the interaction between the calling application and the Web service can begin.

This interaction is the most important aspect of Web services for our discussion. The invocation of a service involves sending an XML message to the service and receiving an XML message in return. These XML interactions are governed by another open standard called SOAP. SOAP defines a message header that describes the message and indicates which operation in the interface of the service is being invoked. The header is an envelope that contains an XML message body in which the parameters are passed. SOAP supports both a remote procedure call and a general XML document-passing paradigm. SOAP messages must be carried on a communications layer, which most often is the HyperText Transport Protocol (HTTP) [Farrell & Kreger, 2002].

Of course, SOAP is the layer that implements messaging between Web service components. As such, SOAP should be a transparent interoperability technology. Unfortunately, many SOAP implementations vary from the standard by either creating extended features or by only making a subset of the functionality available. With different levels of SOAP support, it makes it difficult for true interoperability. Clients wanting to use Web services that run on different platforms have to be aware of these issues and code

accordingly. If all vendors complied with the standards, the client would not have to be concerned about the underlying platform used.

Furthermore, interoperability between SOAP implementations can be difficult, as interpretations of the standard can diverge. But standards creation to resolve SOAP interoperability issues is being done by a number of working groups, such as SOAP Builders and WS-I. Today, however, developers are forced to create multiple variants of Web services to adequately interoperate with partners, which is expensive and labor intensive.

Also applicable to SOAP, specific Web services platforms may support an older version of a specification, which may not be interoperable with your clients. Choosing basic data types such as strings, integers, and standard array types can ease this problem, as the use of complex data types could limit what types of SOAP clients can talk with your service. The second best practice is to provide schema definitions for all data types. A schema definition provides a mechanism for defining the structure and content of an XML document.

SOAP and WSDL are fundamental technologies for an application to issue a request to a Web Service. However, they do not provide support for the application to compose multiple Web Services. With SOAP and WSDL, requests to multiple Web Services are issued individually, but a set of requests cannot be grouped into a single process flow across the web. Because of this shortcoming, additional technologies for web business process specification and management are currently being developed. These include new languages extending WSDL, for example, IBM's WSFL/WSEL and Microsoft's XLANG.

WSFL and XLANG propose language constructs for defining web processes that can involve requests to multiple Web Services [Mikalsen & Rouvellou, 2001]. However, the composition of Web Services as achieved with WSFL and XLANG observes no or only limited forms of reliability. We believe that this restricts their applicability unnecessarily. In many web systems for electronic commerce, for example, a Web Service composition must be reliable. Qualities of composition, such as atomicity of processing of a defined set of Web Services requests, or consistency of data transformations applied to the set of

composed Web Services, should be well-defined, observable, and guaranteed. For this purpose, both language (contractual specification) and system infrastructure support is needed.

Web services represent a new breed of Web applications development [Curbera et al., 2002], [Clabby, 2002], [WEBSERVICES]. As it was mentioned, 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.

### **2.2.2 Semantic Markup of Web Services**

As it was mentioned before, Web Services provide a new model of the Web, where services exchange information on demand to obtain the necessary result and this is especially important for the industrial community, because this opens a new way to conduct electronic business in a more efficient way. We can conclude that services need to efficiently find other services that provide solutions to concrete problems and services should interoperate to solve complex tasks. An XML-based standard, UDDI, provides a 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. In [Ankolekar et al., 2002] and [Ankolekar et al., 2001], DAML-S is adopted as a service description language. DAML-S provides capability to semantically annotate the web service. The DAML-S service description is richer than the representation of the service provided by UDDI or WSDL [Ankolekar et al., 2002]. The UDDI description does not evolve any service capability description. 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. The DAML-S ontology provides classes and properties to describe the content and capabilities of the Web Services [Ankolekar et al., 2002], [Ankolekar et al., 2001].

Structuring of the ontology of services is motivated by the need to provide three essential types of knowledge about a service (Figure 2.5), each characterized by the question it answers [DAML, 2002]:

- What does the service require of the user(s), or other agents, and provide for them? The answer to this question is given in the “profile”. Thus, the class Service presents a ServiceProfile;
- How does it work? The answer to this question is given in the “model.” Thus, the class Service is describedBy a ServiceModel;
- How is it used? The answer to this question is given in the “grounding.” Thus, the class Service supports a ServiceGrounding.

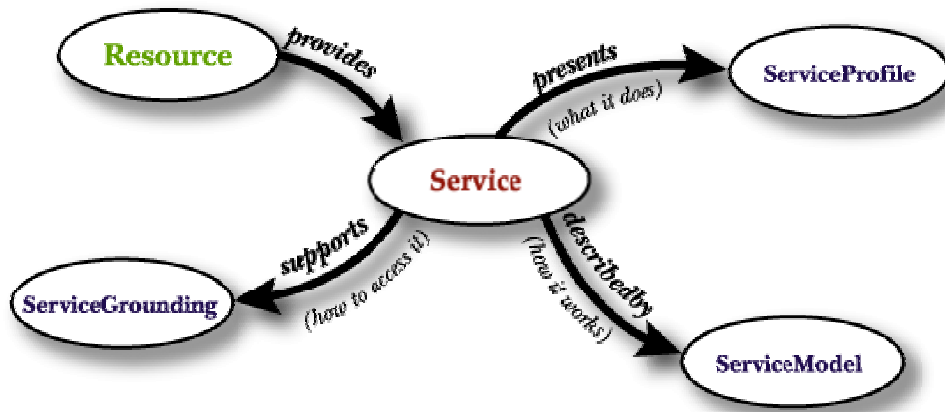


Figure 2.5. Top level of the service ontology.

DAML-S will enable users and agents to automatically discover, invoke, compose, and monitor the web service. [Ankolekar et al., 2001]. The advantages that UDDI gains when integrating DAML-S capabilities are described in [Paolucci et al., 2002].

The DAML-S ontology of services provides enough knowledge that can be used by an 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).

### 2.2.3 Semantic Web-enabled Services

Figure 2.6 provides a schema of steps of evolution to enrich the existing web infrastructure with semantically enabled web services. This leads to the integration of semantic web and web services architectures within one intelligent web.

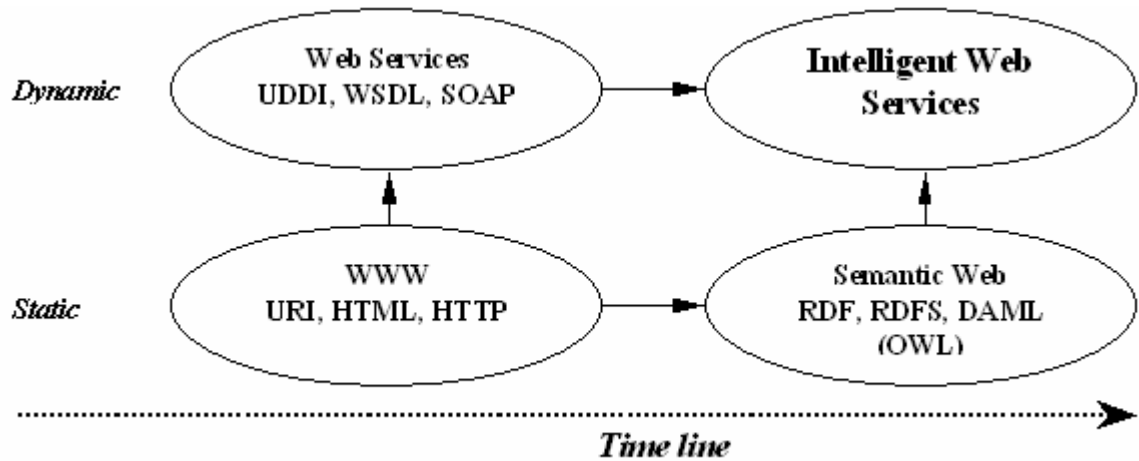


Figure 2.6. Evolution of the Web.

Semantic Web-enabled Web Services allow tackling the current problems of:

- Finding, extracting, representing, interpreting and maintaining information on the Web.
- Integrating web applications within a single international model.
- Bringing the web to its full potential.

## 2.3 Agent Approach to Web Services

### 2.3.1 Agent Technology

#### 2.3.1.1 Software Agents

The background of SW agent technology lies in the fields of distributed computing and artificial intelligence (especially distributed AI) [Wooldridge, 2003]. Agent technology is not however synonym to AI nor expert systems. Unlike many expert systems, agents are



situated in an environment and also act on that environment. Some real-time (typically process control) expert systems can though be regarded to be agents.

There are a lot of definitions for software agents but none universally accepted so far. Typical attributes of SW agents are listed later. Researchers of different background and personal interest emphasize different attributes in their definitions. Definitions vary from very generic like [Laamanen, 2001]:

- "An agent is a computational process that implements the autonomous, communicating functionality of an application."
- "Autonomous agents are computation systems that inhabit some complex dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks for which they are designed."

What is ever the definition, the main point is that an agent can carry out information-related tasks without immediate human intervention and supervision. Thus an agent is a computer system capable of autonomous action in some environment controlling its own internal state. Agents have an idea how to accomplish tasks. Ideally an intelligent agent takes always the best possible action in a situation. Some researchers define also that an agent is autonomous only if it is capable of learning from experience and its behavior is determined by this experience. This can however be considered too narrow-minded. With little or no experience at all an agent would act randomly in such a case. It is therefore reasonable to provide autonomous agents in most cases with an initial knowledge of their working environment.

From different definitions it is possible to summarize a list of attributes common to SW agents. Typical attributes of a SW agent [Wooldridge, 2003], [Seilonen, 2001], [Virtej, 1998]:

- *Reactivity* - an agent senses its dynamic environment and acts based on it. An agent maintains an on-going interaction with its environment and responds to changes that occur in it.
- *Autonomy* – an agent makes decisions over its own actions.

- *Proactiveness/Goal-orientedness* – an agent takes the initiative and recognizes opportunities in generating and attempting to achieve goals. It is not driven solely by events.
- *Social-ability/Communication* – Agents interact with each other (and possibly humans); typically by means of communication.
- *Cooperation* – an agent realizes that some goals can only be achieved by cooperating with others.
- *Learning/Adaptivity* – an agents' ability to learn from history and to adapt to changes means flexibility and improved performance over time.
- *Continuity* – an agent acts for an undefined time.
- *Mobility* – an agent is portable i.e. able to move from one machine to another or around an electronic network.
- *Rationality* - Agents will act in order to achieve their goals, and will not act in such a way as to prevent their goals from being achieved - at least as far as their beliefs permit it.

It is naturally the application and goal in question that determine which ones of these attributes dominate in each case. In a typical agent several attributes together make up the behavior of an agent. Attributes dominating too much may decrease the usability or performance of the system. For example an agent being too autonomous may in the worst-case result as an autistic agent. On the other hand social-ability with other agents and humans must not prevent an agent from having control over its own actions and internal state.

AI aims to build systems that can (ultimately) understand natural language, recognize and understand scenes, use common sense, think creatively, etc. all of which are very difficult tasks. We do not however need to solve all problems of AI to build a useful agent. When building an agent, we simply want a system that can choose the right action to perform, typically in a limited domain. A little intelligence goes a long way [Ojala, 2001].

Agents are best suited for applications that are modular, decentralized, changeable, ill-structured and complex. Humans must specify the task but we want to tell agents what to do without telling them how to do it.

### **2.3.1.2 How Are Agents Built up?**

This section is based on materials from [Ojala, 2001].

Programming has progressed from procedures and functions to abstract data types and object oriented practices and lately to agent oriented practices. Agents are sometimes thought to be just objects by another name because an object:

- Encapsulates some state.
- Communicates via message passing.
- Has a method, corresponding to operations that may be performed on this state.

To certain degree these characteristics resemble those of agents. Main differences in agents are however:

- Agents are autonomous; agents embody a stronger notion of autonomy than objects, and in particular, they decide for themselves whether or not to perform an action on request from another agent.
- Agents are smart; capable of flexible (reactive, pro-active, social) behavior. The standard object model has nothing to say about such types of behavior.
- Agents are active; a multi-agent system is inherently multi-threaded, in that each agent is assumed to have at least one thread of active control.

SW agents can be built in many different ways using different architectures. The right choice for agent architecture depends on many things like application (i.e. the problem in question) and agent environment. In most cases the agent environment is inaccessible, so that the agent cannot obtain complete, accurate, up-to-date information about the state of the environment. The physical world is also non-deterministic, so agents' actions have no single guaranteed effect; there is no certainty about the state that will result from performing an action. Furthermore, the physical world is a highly dynamic environment,

other processes operating on it, and which hence changes in ways beyond the agent's control.

Main types of agent architectures are:

- Deliberative (symbolic/logical) agents. An agent architecture that contains an explicitly represented, symbolic model of the world. Sensor data is used to update this model. Makes decisions (for example about what actions to perform) via symbolic reasoning.
- Reactive agents. Most everyday activities consist of routine actions and not of abstract reasoning. Some agents decide what to do without reference to their history; they base their decision-making entirely on the present, with no reference at all to the past. Perception from sensors is mapped to primitive actions. These agents work best where everything can be computed beforehand.
- Hybrid agents. The best properties of both combined.

### **2.3.1.3 Why Agents?**

This section is based on materials from [Wooldridge, 2003], [Seilonen, 2001], [Breger, 2003].

SW agent technology creates many new possibilities for example in forms of self-organizing modules and dynamic creation of solutions that have an adequate reaction to unforeseen events. Agent-based solutions will also be highly modular and easy to maintain. In general it can be said that SW agent technologies make new services and applications possible while old services and applications can be realized easier, faster and cheaper. Some ongoing trends that have driven the evolution of computing towards SW agents are:

- Ever more delegation to computers
- Ever more distributed systems
- Ever more intelligent systems
- Ever more human-oriented views

#### **2.3.1.4 Multi-agent Systems**

This section is based on materials from [Parunak, 1998], [Pitkänen & Shuling, 1998].

In multi-agent systems agents should inter-operate for giving solutions and for solving some specific problems. Individual agents may be identical or different from each other. Typically agents are heterogeneous in such a way that each agent has incomplete information or incomplete capabilities for solving the problem. In typical distributed systems there are interface agents, task agents and information agents. Agents however work towards a single global goal or separate goals that interact. Coordination and planning may be organized in a centralized as well as a distributed way. Communication in multi-agent systems is usually based on common protocols although interaction may also happen through environment only. Even the coordination of agents can be handled indirectly via environment. Most noteworthy characteristics of multi-agent systems are:

- Number of agents in a system (number of different kind or number of individuals). This may change as agents are created, destroyed, fused into single ones or single agent may divide into multiple ones.
- Communication means (medium, addressing, persistence, locality).
- Communication protocols (direct, voting, negotiation, speech acts).
- How agents are configured in relation to one another (set in advance or agents able to discover new relationships and configure themselves).
- How agents coordinate their activities as the system runs.

#### **2.3.2 Agents and Semantic Web Services**

Agents, as well as many other technologies around the semantic web, have shown an increased maturity through standards and open-source. These improvements have been very self-centered and led to the creation of silos. Time has come to integrate these improvements into an ecosystem, bringing a larger picture towards active web-services that is capable of serving each individual user personally.

### 2.3.2.1 The Semantic Layers of Agent-based Web-services Communications

Web-services require more infrastructures to realize all of their potential benefits than their existing static counterparts. There is a clear trend towards an explicit representation of web-services and the addition of semantic communications to the existing syntactic ones. We give below a short description of the extension of communication technologies towards semantics (see Figure 2.7) [Bernard, 2002].

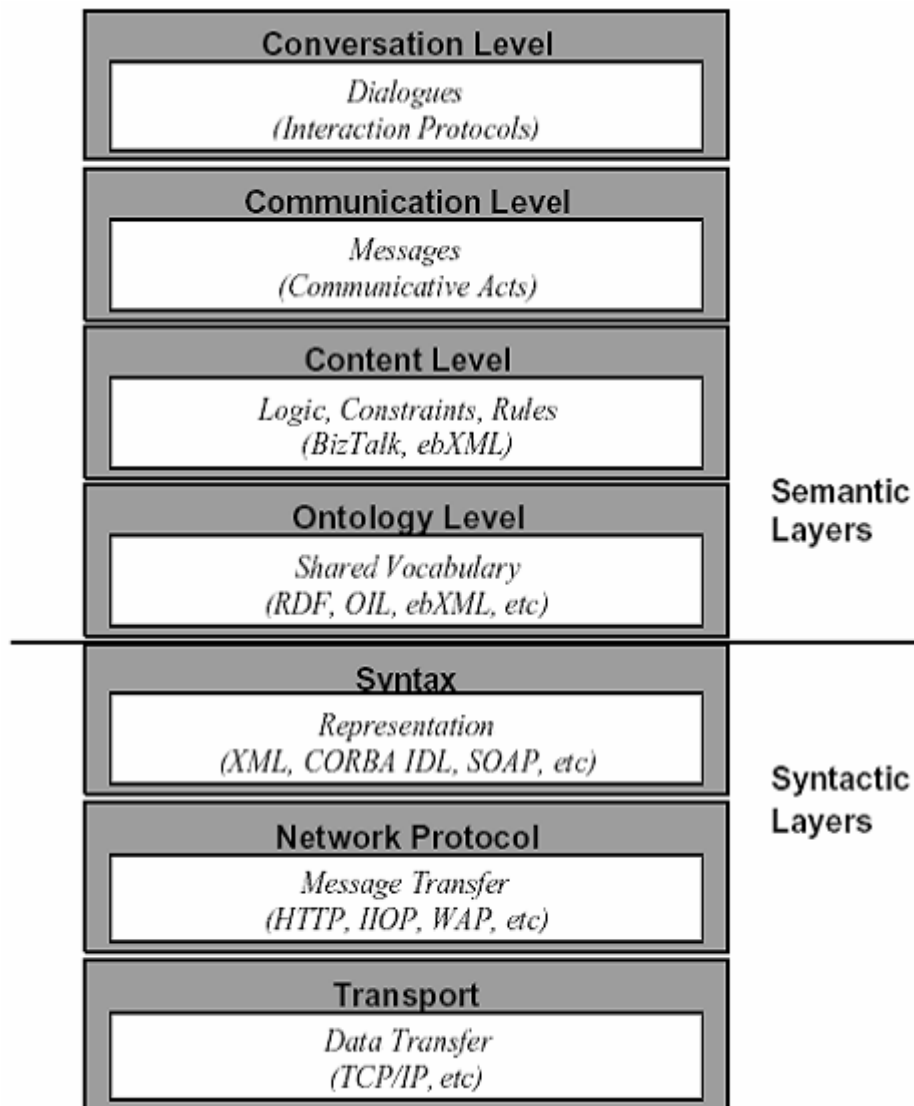


Figure 2.7. Communication technology stack.

*Dialogues and Interaction protocols* define message interactions between agents at the conversation level, that is, when two or more agents agree to exchange messages using a specific interaction protocol. The interaction protocol shows the messages that each agent can send and receive at each stage of the conversation. Interaction protocols can be simple, such as requesting an agent to complete an action and the agent agreeing or refusing; or complex, such as an auction or a call for proposals; *Communicative acts* define standard, application-independent methods for passing semantic messages between agents. A communicative act is a verb-utterance providing context for the contents of a message, for example, request, inform, etc. Communicative acts are application-independent, as opposed to services or actions, which are application-dependent, and can be reused across a wide range of application environments. Technically, an *Agent Communication Language* (ACL) provides a standard way of representing meta-information that can be associated with a message, such as the sender, receiver, the ontology used to express the message content, etc.; *Content languages* are used to express the actual content of a message. Some services may use a simple, fixed syntactic and semantic representation, although a number of agent-based systems use content languages such as predicate logic and constraint choice languages; *Ontology* is a vocabulary of terms and their definitions and relations that are applicable in the current problem domain. Ontologies supply the basis of semantics, by describing in a formal system the domain of discourse for a particular application, such as information retrieval or financial services.

### **2.3.2.2 Active Web-services**

When looking towards the future of web-services, we predict the breakthrough will come in the form of access to services. Added intelligent capabilities through semantic reasoning will be a secondary factor. Agents bring the most crucial capability to turn the entire web-services from the existing dormant mass of information where users need to surf and browse, into a dynamic set of capabilities deployed around and serving the user. Agents represent this great opportunity towards a new and completely different computing model, freeing humans from numbers of chores imposed by the contemporary Internet where users are sometimes enslaved by computers.

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. Agents introduce an unparalleled level of autonomy into future systems so that users can delegate high-level tasks in a generic manner. Therefore Agents need to get: mission statements, the definition of domains of competence, and a definition of autonomy through policies to be applied in these domains. These domains and policies need to support reasoning as they might be overlapped in real life; for example, the policies of multi-national companies have to comply with the policies of the nations in which these companies operate. These domain and policies mechanisms permit deployment and dynamic adaptation to any situation. In particular, they allow web-services to combine without prior design, negotiate end-to-end contracts to insure the final result of composed web-services, and monitor their execution on behalf of the user. Some initial experimentation on automatic generation of contracts show encouraging results towards contractual web-services [Rodriguez & Sallantin, 1998].

Agents will become the trusted intelligent interface between man and machine, allowing communications through speech acts and representing the interest of the user in any web transaction at any time, like a trusted friend or lawyer. Hence agent interfaces need to evolve towards ease of use, ease of delegation and monitoring of tasks, increased privacy, personalization and security, and user habits being acquired through learning. See overview on subject in [Dickinson, 2001].

Agents can now migrate to slim wireless appliances, and evolve in a multitude of micro worlds, typically the cells of wireless phones, malls, schools, a community of friends; and discover the resources and represent their user. This technology needs to be rolled out on a large scale to test the deployment capabilities as well as the usability of the technology in a mass market.

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 [Bernard, 2002].

### **2.3.2.3 Agent Standardization**

This section is based on materials from [Seilonen, 2001], [Breger, 2003], [Wooldridg, 2003], [FIPA, 2000], [Laamanen & Helin, 2001].

Standardization is always important from the interoperability and reusability point of view, having thus also an important effect on the extended use of any new technology. On the other hand standardization is always a compromise and the standardization process itself can be very slow. Therefore there is also a possibility that the standardization process itself is the obstacle preventing a broad use of a new technology. In the area of SW agent standardization the main problems are lack of consistent theory and definitions.

In this area the standardization work done by FIPA (Foundation for Intelligent Physical Agents) is by far the most important. FIPA was established 1996 and has nowadays over 60 member companies world-wide, almost equally represented in Asia, Europe and America with a strong representation of telecommunication and software companies. The FIPA standards follow an open process. FIPA meets quarterly for a one-week period in which Technical Committees and Work Groups develop the specifications. Members are invited to participate as well as anybody who wants to contribute to the technical progress of the work. Anybody can send technical proposals, comments or even submit work-plans to FIPA. In between meetings, the work progresses via email reflectors and sometimes through ad-hoc meetings. Information on FIPA specifications, proposals, meetings, registrations, activities.... and the standards themselves are accessible from the web site.

FIPA work is mainly done on a quite generic level. For example the FIPA abstract architecture specification is supposed to provide a framework in which services necessary to support the end-to-end interoperability of agents are specified. This abstract architecture permits many different realizations. Standards currently available (FIPA97, FIPA98 and FIPA2000) include guidelines to build multi-agent platforms:

- Agent system architecture, agent and system design (Agent UML)
- Agent management (creation, deletion, migration of agents)
- Agent communication (high abstraction level, knowledge sharing, interaction protocols, Agent Communication Language, Knowledge Query and Manipulation Language (KQML))
- Agent message structure and message transport (transport protocols IIOP, HTTP, WAP)
- Agent security, agent mobility

Agent UML is a result of the cooperation of FIPA and the Object Management Group (OMG). Up to now results include mechanisms to model interaction protocols (protocol diagrams and extending UML state and sequence diagrams in various ways), extension of class diagrams to model agents behavior and specification of ontologies. Agent UML is a part of the FIPA2000 specification and some extensions are part of the upcoming UML release. This kind of cooperation is very important but unfortunately only a good starting point, not much more.

#### **2.3.2.4 Agent Environment**

The Agentcities Network [AGENTCITIES] (hereafter referred to just as the Network) represents the first attempt to build an open, global and standards-based agent environment for research and future commerce on the Internet.

The objective of the Network is to bring together technologies from both agent and AI research (such as agent communication languages [Finin et al., 1992][FIPA, 2001(a)], conversation protocols [Burmeister et al., 1993][FIPA, 2001(b)], ontologies [Fensel, 2001], coordination [Wooldridge & Jennings, 1994], negotiation [Laasri et al., 1992] and open

systems theory found in the current Agentcities Network Architecture recommendation [AGENTCITIES, 2002]) with industry-led technology initiatives (such as Web Services [W3C(b)], JXTA [JXTA], XML [W3C(c)] and RDF and RDFS [W3C(d)]) to create a global, open, dynamic environment that enables:

- Rich, flexible communication between software entities deployed within it.
- Software entities to trade automatically with each other in a dynamic and flexible way without the constant intervention of humans.

### 3 Ontology-based Distributed Integration Environment for Heterogeneous Resources

#### 3.1 OntoShell Approach to the Problem

*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. "Ontology-based" means that we have to create all of the resources' Ontologies before.

Concerning this problem, we propose an OntoShell approach to an Ontology-based universal integration environment development (Figure 3.1).

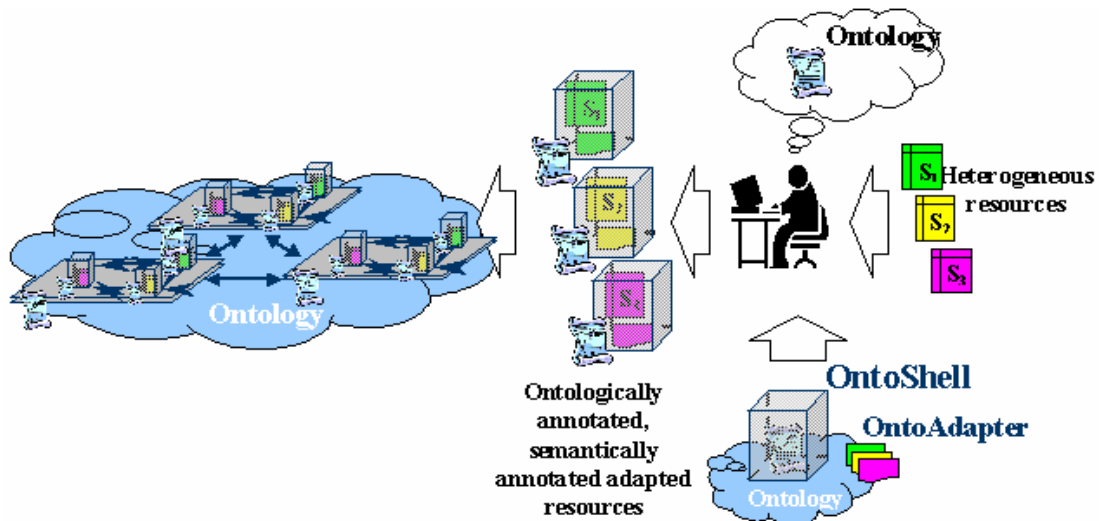


Figure 3.1. OntoShell approach.

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) (Figure 3.2).

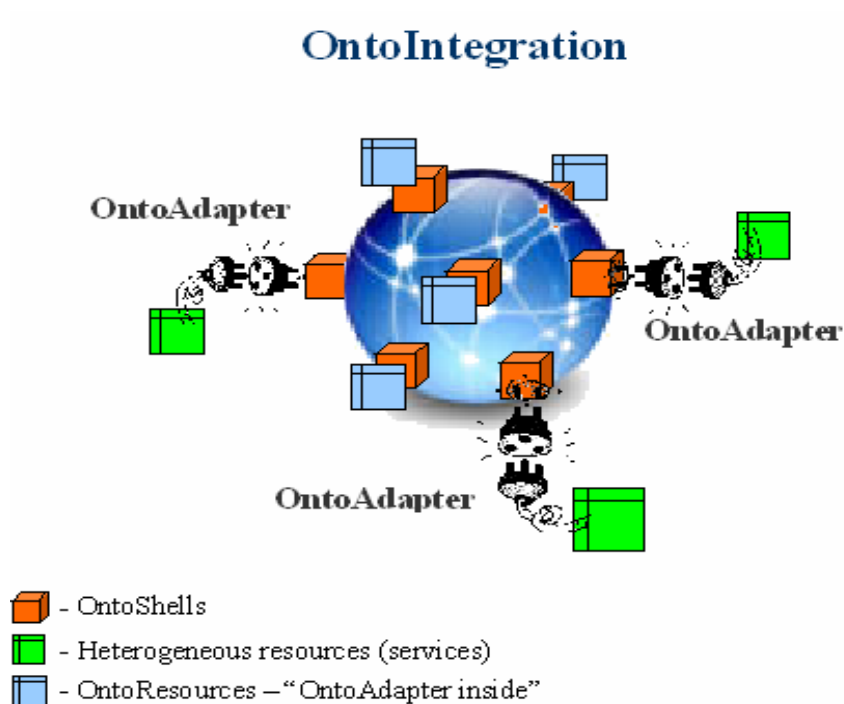


Figure 3.2. OntoEnvironment – “environment-mediator”.

**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.

The structural schema of one such OntoShell is showed in Figure 3.3. 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 transport levels. They will be construction blocks, for OntoShells, and will be defined depending on resource's description (Figure 3.2). There are RCA modules for resource adaptation on the content level and RTA modules for resource adaptation on the transportation level (Figure 3.3).

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 (Figure 3.3).

A human executes an initial description of a resource via the visual user interface (VUI) (Figure 3.3) 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) (Figure 3.3) 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.

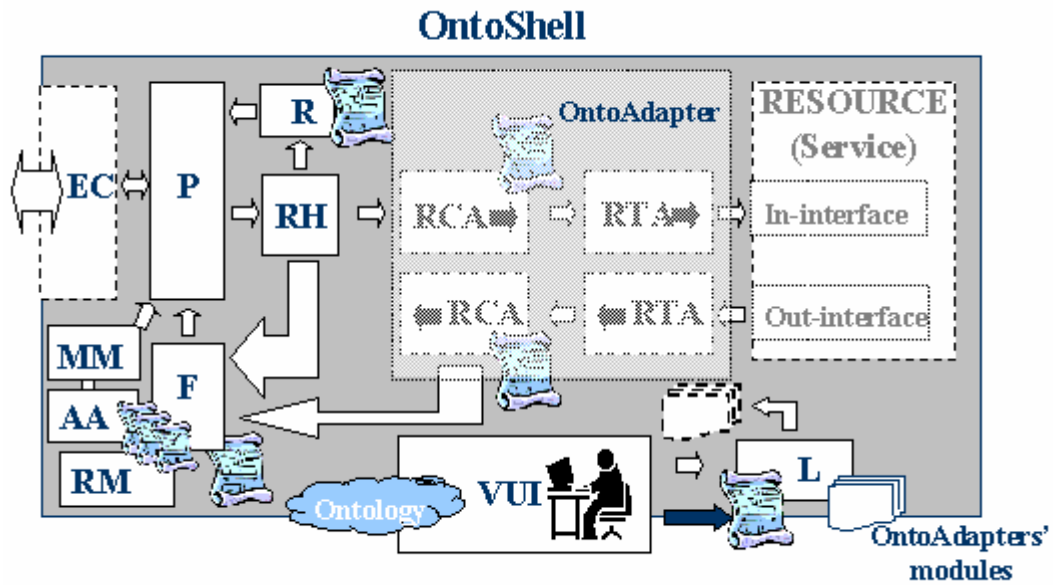





Figure 3.3. OntoShell's structural schema.

- L – linker;
- P – packer/unpacker;
- R – registration module;
- F – forwarding module;
- RH – request handler module;
- RM – RelationManager;
- AA – Advertising Agent;
- MM – MobilityManager;
- 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 cluster's members or neighbours 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 (Figure 3.4). 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).

Interaction between OntoShells can be organized via either a centralized or decentralized (P2P) interaction architecture depending on an environment's interaction architecture, to which a resource will be embedded in.

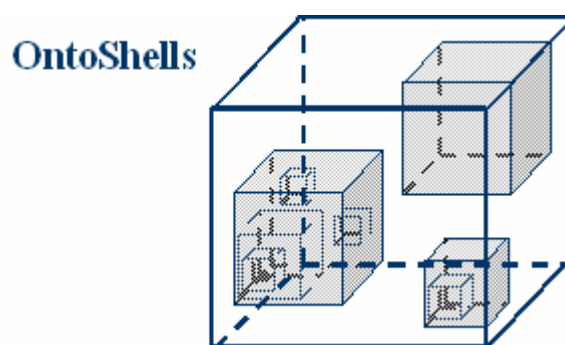


Figure 3.4. Multilevel cluster architecture.

The main element of this OntoEnvironment is an OntoShell. As was mentioned, an OntoShell is a mechanism for making an ontological description and providing interoperability for the resources. So, we have the environment with many OntoShells,



which can interact with each other via the common language. But it is not enough, because these OntoShells need the interaction, advertising and registration mechanisms, possibility to be mobile (movable), etc. That is why an OntoEnvironment is set of the OntoShell-enabled elements (services) (Figure 3.5), such as:

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

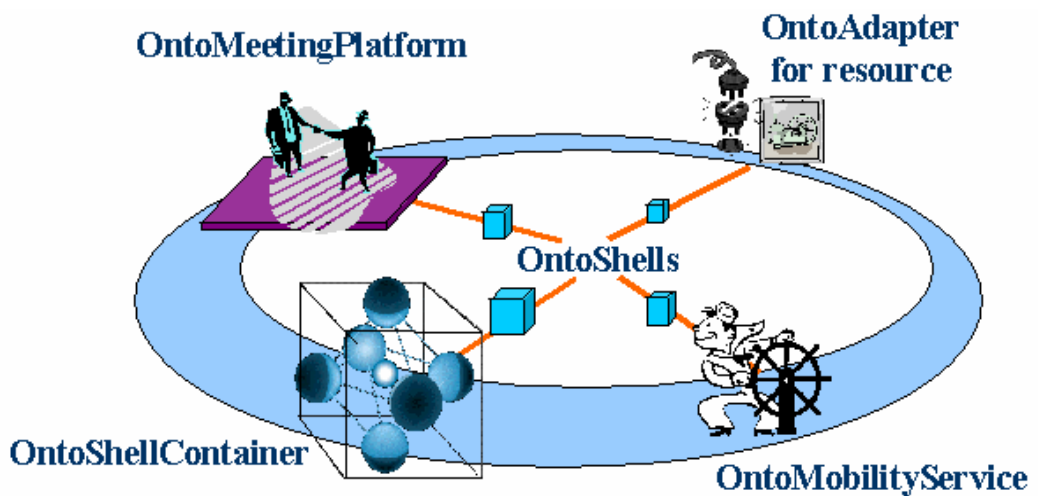


Figure 3.5. Elements of an OntoEnvironment.

## 3.2 Interaction Models

### 3.2.1 Centralized Interaction Model

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 (Figure 3.6). Such additional opportunity can speed up the resource search.

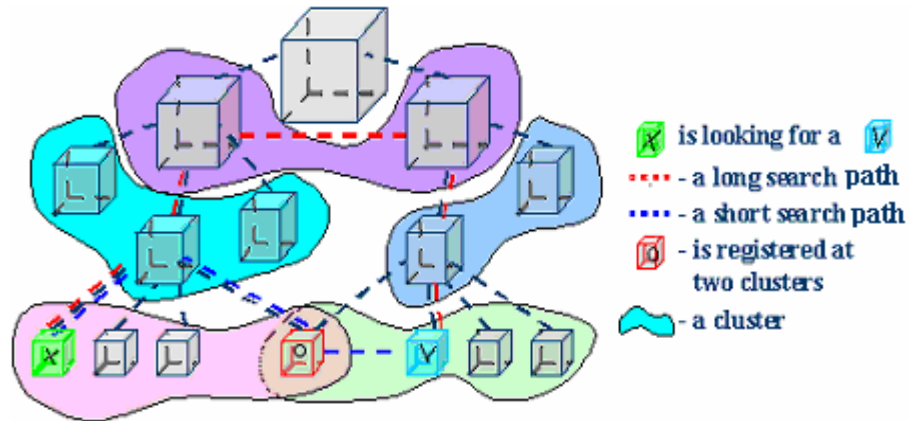


Figure 3.6. Through level search.

### 3.2.2 Decentralized Interaction Model

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.

### 3.2.3 Hybrid Interaction Model

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

for the observance of registration agreements, the quality of the provided search service, etc.

We may consider two main reasons for cluster organization:

- Cluster organization is chosen in order to decrease useless traffic during the search of a resource. In this case, a cluster is organized in a hierarchical relation of “class-subclass” type based on the resource ontology. In point of fact, a “mother shell” may register only elements which are its subclasses. An example of such clusterization is presented in Figure 3.7.

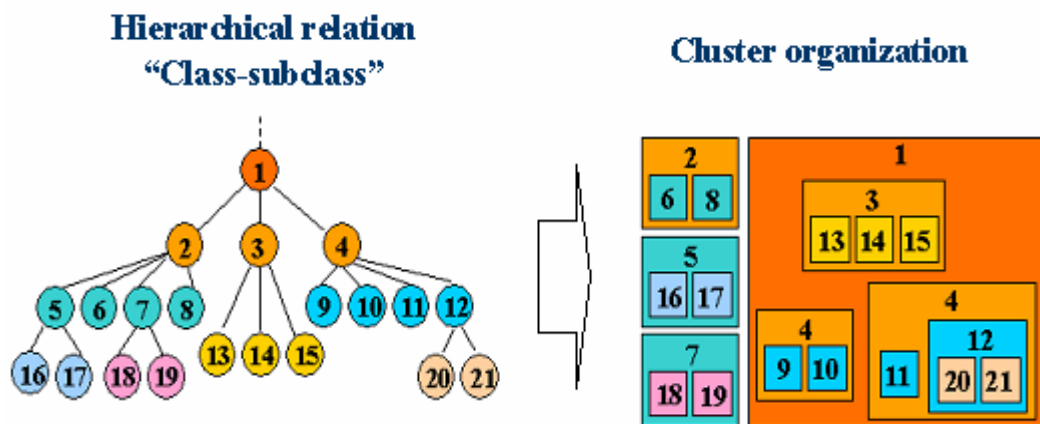


Figure 3.7. “Class-subclass” clusterization model.

Since the organization of such clusters will be carried out spontaneously and shells of some level may not register in the “mother shell”, the relations between the shells cannot be considered a totally centralized architecture.

- Cluster organization is chosen to build a closed set of functioning resources. It may be used for the organization of a cluster, which covers a concrete domain with a set of different resources without relation to the same class (for example a maintenance platform with a set of services such as: main maintenance service, device alarm service, set of classifiers, etc.). In this case, a “mother shell”, which represents some cluster, provides search and interaction organization for the registered resources. But it cannot represent all of them in a height level cluster as one element, because the aggregate of descriptions is not a subclass’s description of some height level class. There is only one way to go out to the height level cluster. This way is registration in

clusters, subclasses of which are separate elements of the concerned cluster. Such cluster organization is represented in Figure 3.8.

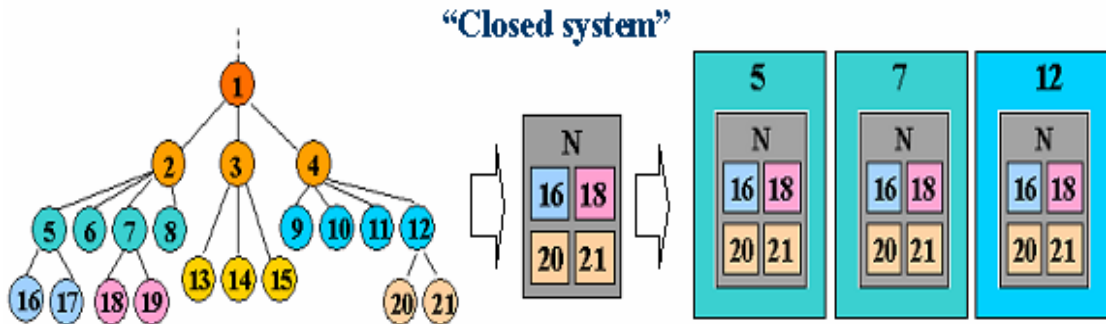


Figure 3.8. “Closed system” clusterization model.

Turning back to the impossibility of whole hierarchical clusters’ nesting, which covers all levels of a “class-subclass” type ontology, we cannot provide a guaranteed resource search via the “mother shells”. Also, search in a cluster-tree, formed on the some level, provides both centralized top-down search and non-effective bottom-up rise at the same time.

For resolving these two main problems we may introduce an additional possibility of interaction between the elements of an OntoEnvironment without “mother shell”. We may say that it is a P2P interaction model for an OntoEnvironment. The main idea is that each OntoShell keeps its own “record book”. This “record book” has to contain a list of useful resources. In that way each shell (resource) can use its own “record book” directly. Replenishment and modification of a resource’s “record book” are executed during interaction establishment with other resources. Such direct interaction model is represented in Figure 3.9.

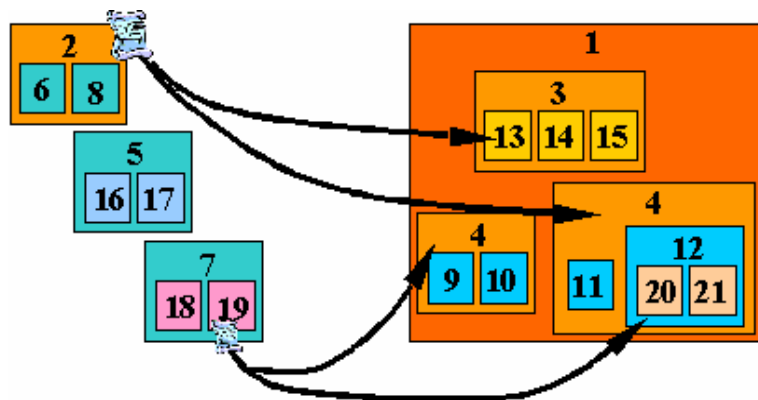


Figure 3.9. Direct interaction model.

Some variants for resource search in the hybrid interaction model are:

1. Interaction organization via *OntoShellContainer* (“mother shell”)
2. Records exchange during interaction between resources.
3. Using *OntoMeetingPlatforms* – places, where shells (more precisely, their 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 a possibility for a shell’s publicity agent (*PublicityAgents*) to meet other agents and exchange records in their “record books”. This service may be placed into an *OntoShell* or may be elaborated like a service of a new generation in the *OntoEnvironment* and supplied with the same interaction interface like the *OntoShells*. Such *OntoMeetingPlatforms* may be attached to some class of service classification tree in the ontology and cover a specific resource domain. Such relation to the concrete domain may be fixed on *OntoMeetingPlatform*’s annotation (description) and used by *OntoShells*’ *PublicityAgents*.

Since the amount of records will increase very fast, we have a need to supplement the *OntoShell* structure with a “record book’s” management block – *RelationManager*. Thus, we insert two additional elements in the *OntoShell* for the management of relations. There are *RelationManager* and *PublicityAgent* blocks. These blocks have to be configurable. *RelationManager* has to be responsible for the 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

Considering distributed environments for resources, the necessity of resource mobility emerges in a number of cases. In other words, we have a need to move a resource with its necessary “equipment” from one machine (computing system) to another. The realization

of such movement is a duty of a special service - *OntoMobilityService*, which will provide mobility in OntoEnvironment. Thus, a 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 of a mobile environment, elements of OntoEnvironment have to be supplied with a MobilityManager module. This module has to be configured in conformity with a policy system (concerning mobility). A resource can be configured to be both a movement initiator and an available resource for move. 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 a respective ontology for messages between elements of a mobile environment and an ontology concerning the behavior and relations of these elements.

### 3.4 Business Model

Concerning the use of the discussed distributed integration environment based on the OntoShell approach, we have to consider the use of it in a business environment. In such environment service providers are interested in a frequent use of their services, that is why advertising and search service play such an important role in this environment. Also, in such business environment there must be some mediation elements, which provide necessary services for the players.

#### 3.4.1 Patterns of Behavior for Elements of OntoEnvironment

*OntoShell*. From the moment it begins to exist, an OntoShell needs to advertise its resources. For the realization of this goal we may consider two ways: registration in a “mother shell” and delivery responsibility in OntoShell advertising; itself advertising during the life cycle and visiting OntoMeetingPlatforms. In case of need to interact with some resource (if it does not locate in its “record book”) an OntoShell has to use a search process via the “mother shell” or a special search service. Also, an alternative solution is to stay on an OntoMeetingPlatform with the goal of meeting the another necessary resource

or finding reference to it. During the establishment of a link with environment element for records exchange (from “record book”) or registration in a cluster, some negotiation mechanism is used. Thus, various aspects of behavior have to be configured beforehand via a respective software visual interface module. Such configuration plays an important role especially in the business environment, where “service” means “money”.

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

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

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

By registering in a cluster an OntoShell shares its “record book” with an OntoShellContainer in exchange for advertising service. This information allows executing more effective search and allows removing useless ascent (bottom-up rise) by cluster-tree during a search, which has been described in chapter #3.2.3. In case of a further refresh of the OntoShell’s “record book”, the OntoShell may proceed with sharing it with the OntoShellContainer (“mother shell”), because depending on the amount of new records (references) the OntoShellContainer shows preference for this particular OntoShell in advertising in case there are several OntoShells. Thus we have a competition between “daughter shells”. In the same time, we have a competition between OntoShellContainers. It may lead to the use of the OntoMeetingPlatforms or special search services for

increasing the quality level of service. In case of need, all elements of a mobile OntoEnvironment use OntoMobilityService.

### 3.4.2 Business Relations between Players

In the business model we may highlight a set of players, such as:

*A* – provider of OntoShells, OntoShellContainers and OntoMeetingPlatform;

*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.

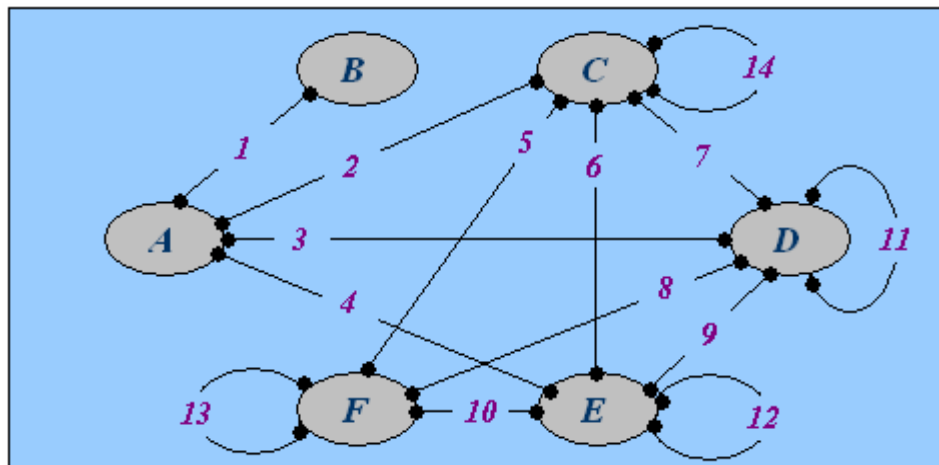


Figure 3.10. Inter-players interaction.

Figure 3.10 shows business relations between players. Detailed description of these relations will be presented below:

*1* – Player “A” is a customer of player “B” for adaptation modules development (OntoAdapter’s modules).

*2* – Player “A” supplies an OntoShell with the necessary adaptation modules to player “C” for the inculcation of its resource in the OntoEnvironment.

*3* – Player “A” supplies an OntoShellContainer to player “D” for cluster organization.

*4* – Player “A” supplies an OntoMeetingPlatform to player “E”.



- 5** – Player “C” pays player “F” in case of need to search a necessary resource.
- 6** – Player “C” pays player “F” in case of need to find someone or refresh its “record book” during its stay on an OntoMeetingPlatform.
- 7** – An OntoShell registers itself in an OntoShellContainer based on some agreements and in that way it advertises itself for further discovery. Additionally an OntoShellContainer provides a search service for the registered OntoShells. And 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, it increases its competitiveness). In case of use of a search service, player “D” pays player “F”. In the same time player “F” plays a role of player “C” and may have a need to register in an OntoShellContainer (case #7), then player “F” pays player “D”.
- 9** – Player “D” pays player “E” for the use of an OntoMeetingPlatform by an OntoShellContainer. On the other hand, an OntoMeetingPlatform is a service, which needs to advertise itself. In that case, the OntoMeetingPlatform may be registered in the respective OntoShellContainer.
- 10** – Player “F” pays player “E” for the use of an OntoMeetingPlatform with a goal to supplement the resource database of the search service. On the other hand, the OntoMeetingPlatform may use the search service to 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 an OntoShell, an OntoShellContainer may register itself in another 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** – An OntoMeetingPlatform may visit another necessary OntoMeetingPlatform in case of need to advertise itself for concrete resources. Then player “E” pays 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 another player “F”.
- 14** – If we consider a business environment, we have a great many of commercial services, which need a payment for their services. Then player “C” pays another player “C”.

## 4 Mobile (movable) Web Service based on a Semantic Web

### 4.1 Necessity of Mobile 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 (Figure 4.1). That will also improve service discovery among a large amount of the services.

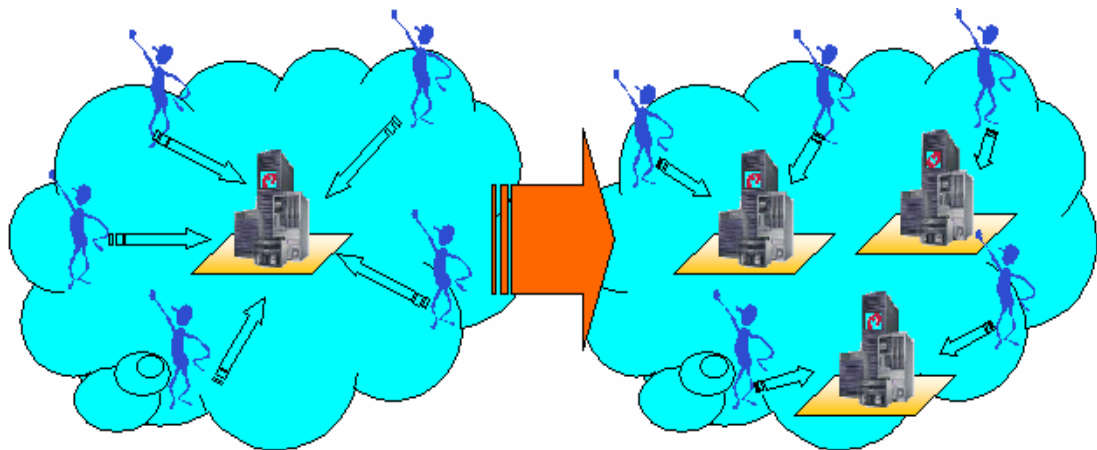


Figure 4.1. Decreasing the utilized capacity of the concrete source.

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 (Figure 4.2).

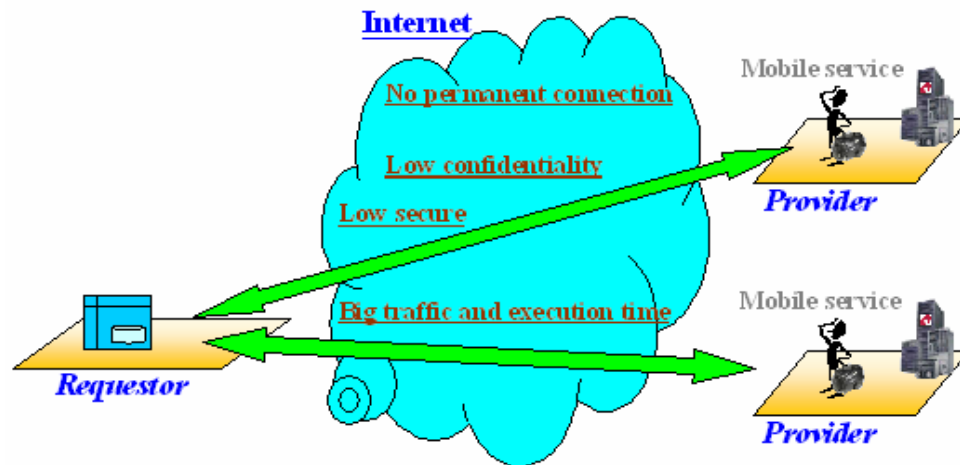


Figure 4.2. Remote service's activities.

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 (Figure 4.3). Of course, in this case we need to take into account the storage capacity of a client.

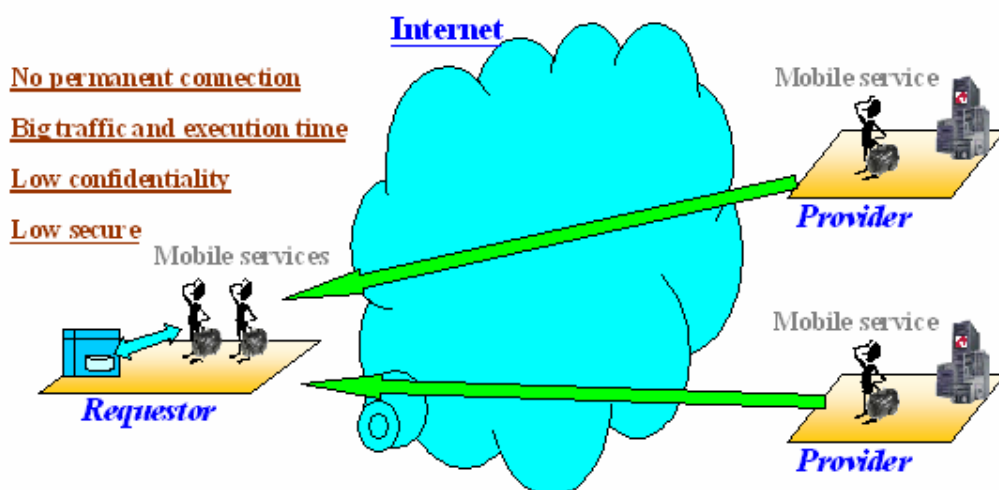


Figure 4.3. Local service's activities.

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.

## 4.2 Equipment for Mobile (movable) Web Service

*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 (Figure 4.4). “Platform Steward” represents ASP. Concerning the OntoShell approach, previously mentioned in chapter #3, “Platform Steward” is represented by the OntoShellContainer (“mother shell” of the second type (section #3.2.3). “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 (section #3.3). P2P management tools for information movement via the network equip the platform.

### Network of Agent-Shell Platform

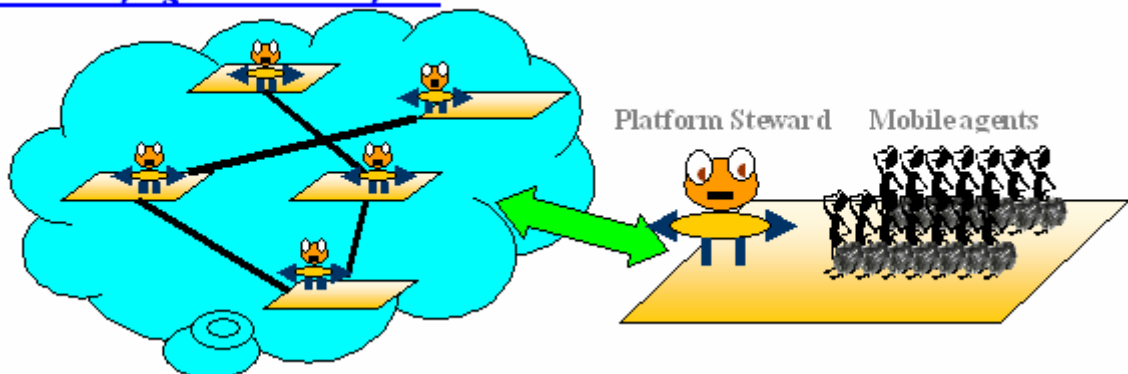


Figure 3.4. Agent-Shell's Platform.

*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.

### 4.3 Service Networks based on the Agent-Shell Platform Approach

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 (Figure 4.5).

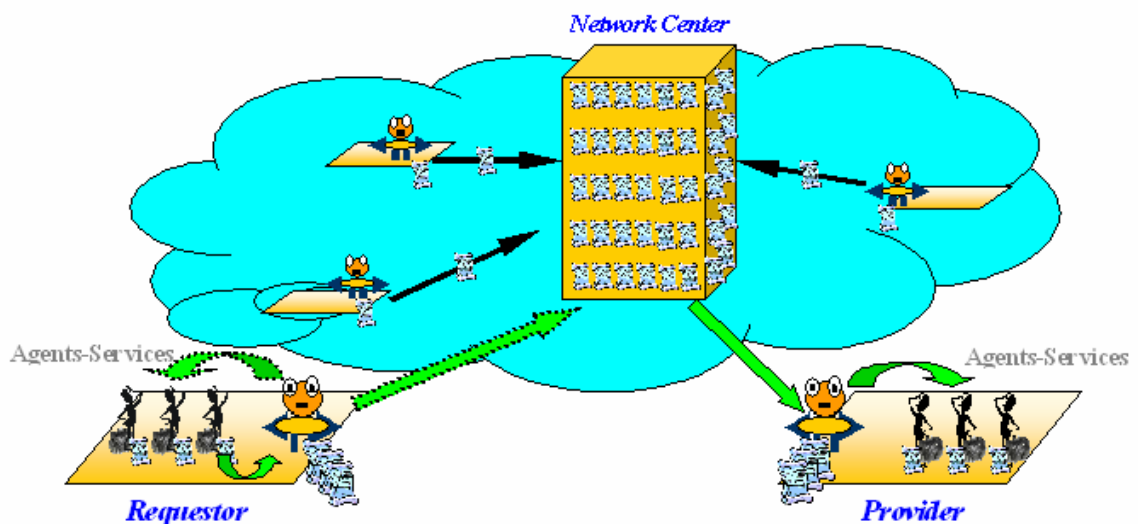


Figure 4.5. Centralized Platforms – centralized agents.

- 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 (Figure 4.6).

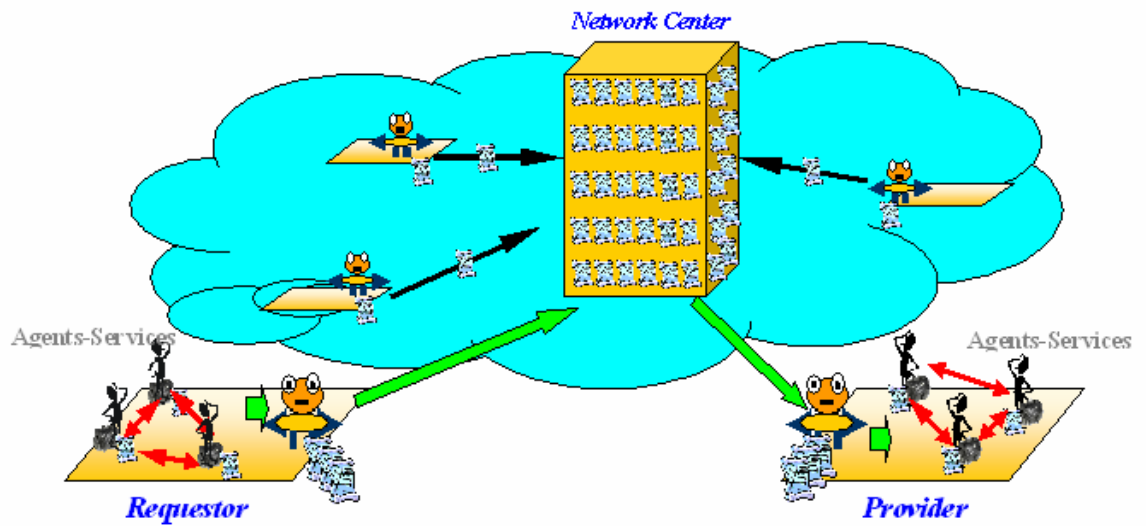


Figure 4.6. Centralized Platforms – decentralized agents.

- Decentralized platforms – centralized platform's agents. This case is similar to the first one, but interoperation between platforms is based on a peer-to-peer semantic service discovery (Figure 4.7).

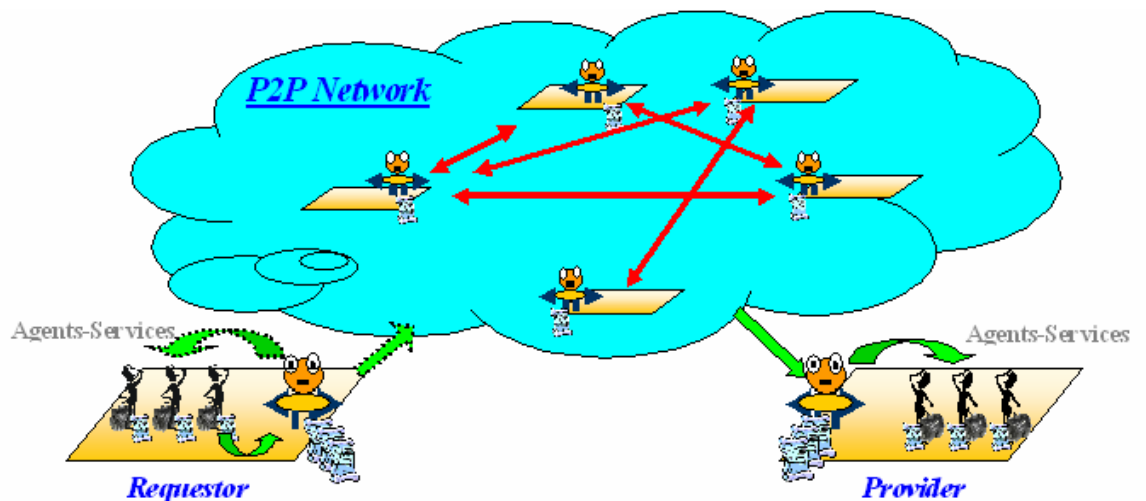


Figure 4.7. Decentralized Platforms – centralized agents.

- 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 (Figure 4.8).

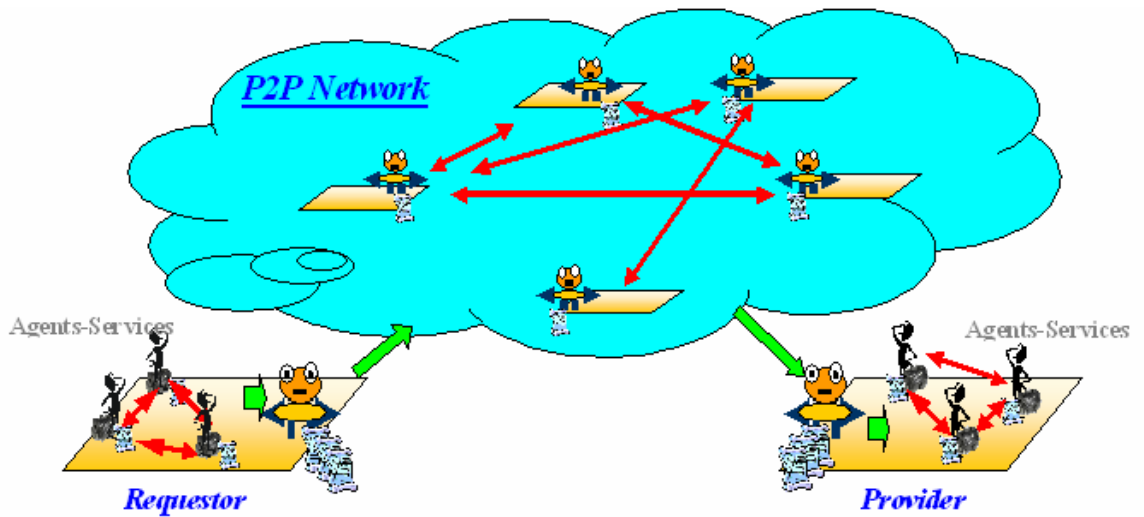


Figure 4.8. Decentralized Platforms – decentralized agents.

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 multiagent 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 (Figure 4.9). Interaction between elements on each level may be organized in either centralized or decentralized way.

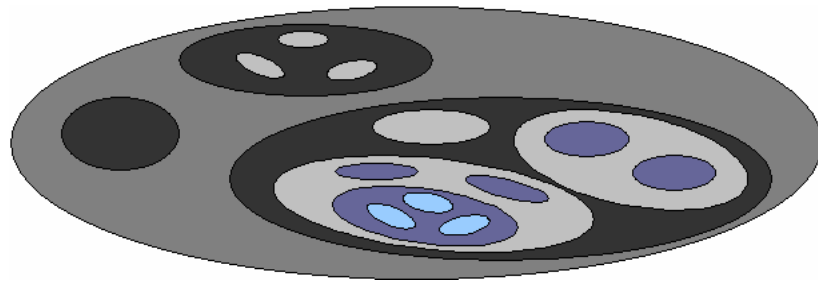


Figure 4.9. MegaHybrid network structure.

Let's 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. In such case 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 (Figure 4.10). In this context, simpler services would be considered like mobile (movable) components of the more complex services and they may be moved on the complex service side, in case of need.

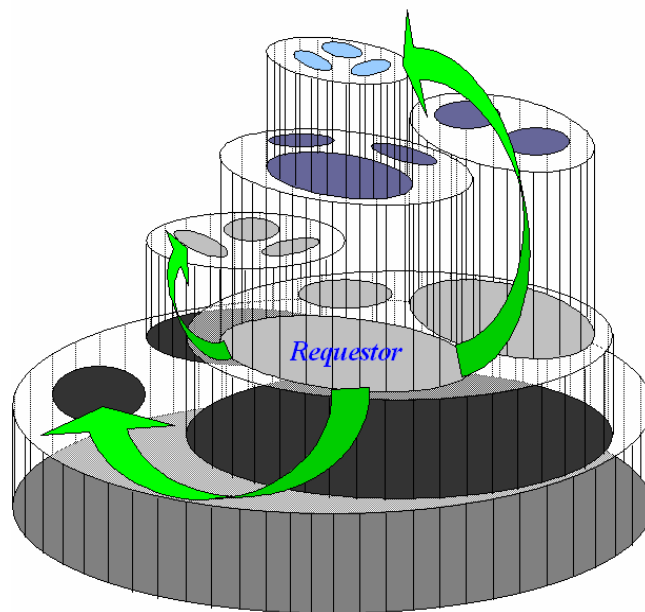


Figure 4.10. “Via-Level” Peer-to-Peer interaction.



## 4.4 Who Provides Web Services and for Whom

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.

So, in this case we have two big classes of service-users and service-providers (Figure 4.11):

- Human component
- Software component
  - smart-devise
  - compound (complex) service

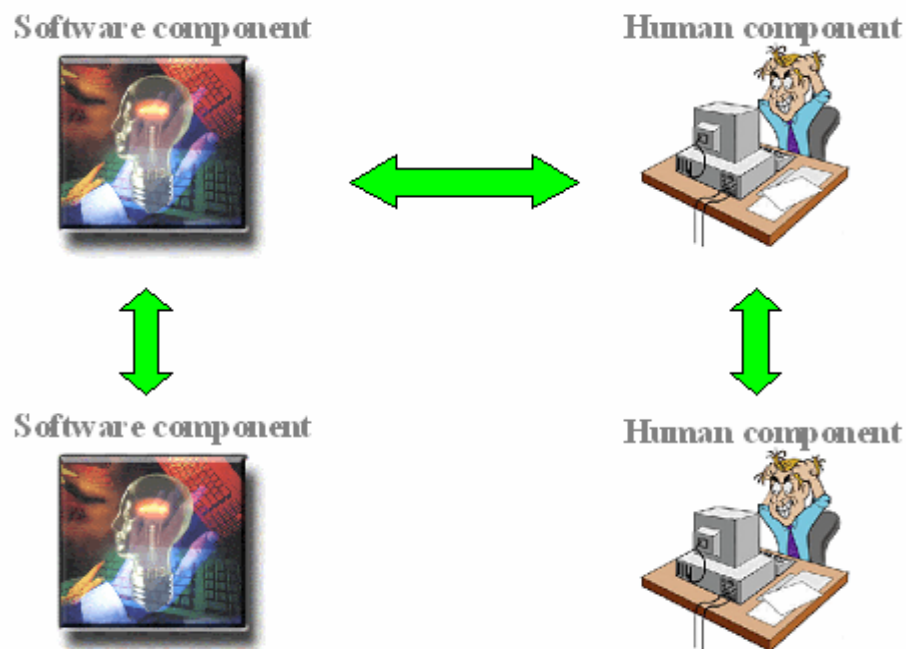


Figure 4.11. Between-component interactions.

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 (Figure 4.12).

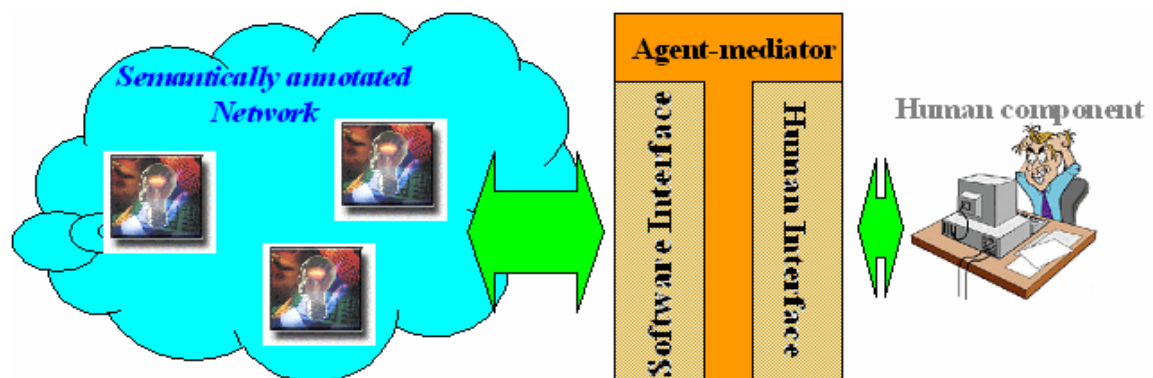


Figure 4.12. *Agent-mediator* - intelligent layer between Software and Human components.

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 a software understandable and a 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 storage space needed). Human component uses the description in his language (Figure 4.13).

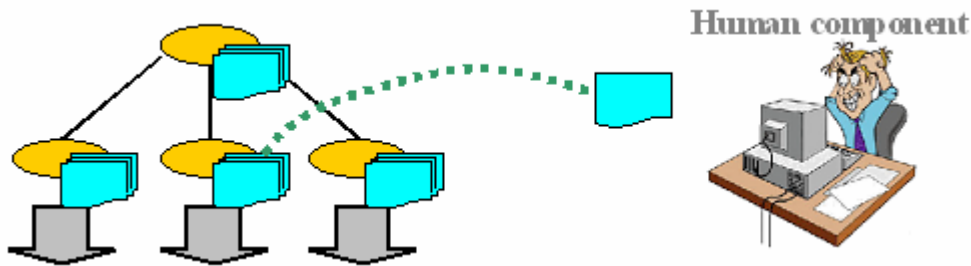


Figure 4.13. Multi-linguistic node's description.

- Implementing translation services for information adaptation (Figure 4.14).

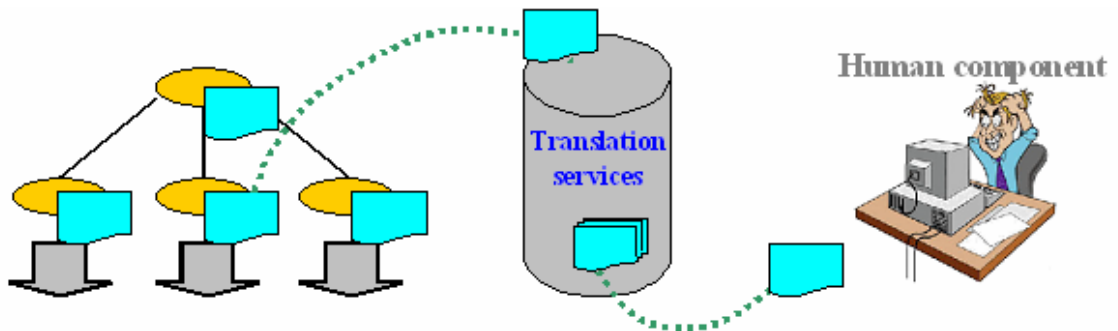


Figure 4.14. Information adaptation via translation service.

Such service may be based on an *Ontology Personalization* concept. An *Ontology Personalization* means development of the support mechanism for a double-sided ontology. Each player will be able to create a personal ontology, in other words it can describe each object from the common ontology (or often used part of it) in terms of

the own presentation way (language, terms, etc.) Thus, we have a double-sided ontology. From one side there is a common ontology, which is used by any semantically enabled resources (elements of an OntoEnvironment). And from other side there is a personal ontology in a concrete player understandable form. In this case we need some mechanism for terms interpretation (translation) from both sides of a double-sided ontology. The goal of such an *OntologyInterpreter* is a two-forked interpretation of the terms on the input and output of the human user interface (Figure 4.15). It is in a sense a dictionary of an ontology.

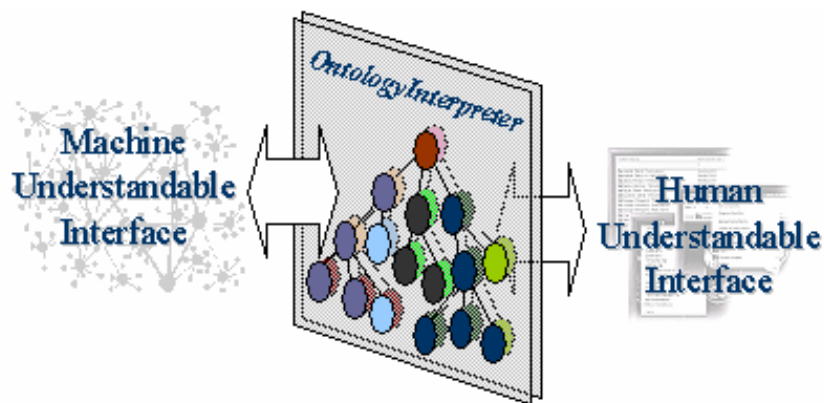


Figure 4.15. OntologyInterpreter – ontology personalization module.

A special tool for a personal ontology creation must supply such an *OntologyInterpreter*. It is not obligatory to create a whole personal ontology if player use just part of it, which concerns to the special domain of the player’s activities. Especially it has a sense in the case of a small storage space, for example in a personal mobile device. And in case of overrunning available part of an ontology, an ontology swap-in mechanism can be used (Figure 4.16).

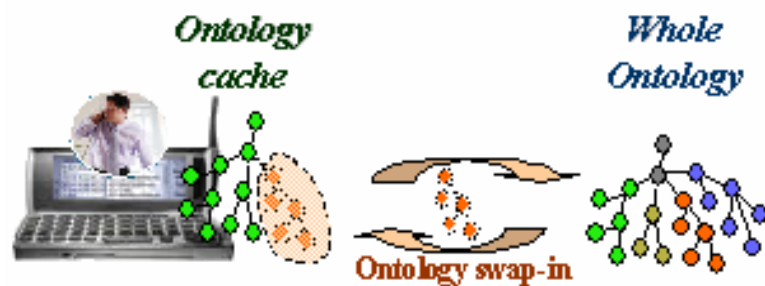


Figure 4.16. Ontology Caching – swap-in mechanism

- 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* (Figure 4.12).

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.

## 5 Global Industrial Maintenance Network based on Mobile Web Services

### 5.1 Industrial Product's Maintenance

#### 5.1.1 Necessity of Industrial Maintenance

Tougher competition in the industrial markets has forced plants to become more cost-effective: they must produce products at ever decreasing costs. As a result, production and maintenance personnel must increasingly be able to identify which maintenance operations are truly necessary and how to schedule them without significant production disturbances.

Ensuring operations at low costs sets many challenges for maintenance operations. Larger maintenance jobs, for example, should be planned according to need rather than at pre-determined intervals. And spare parts inventories should be kept low but also sufficiently large to ensure the right parts are available when needed. Automation provides the bridge between process and business management with real time access to valuable information. Innovative solutions and services in industrial domain improve product quality, process performance and environmental compliance, in addition to lowering operating costs.

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](http://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.

Ensuring reliable operations at minimum costs is a major challenge for today's maintenance operations. For instance, scheduling large upgrades should be based on real need rather than on standard intervals.

### 5.1.2 Field Devices Maintenance

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 [Pyötsiä & Cederlöf, 1999], [Ojala, 2001].

The development of more intelligent condition monitoring techniques provides automation of domain expert knowledge used in tasks of condition monitoring and fault predictions, eliminating human presence from the “*problem-solution*” chain and provides integration of maintenance experience, making it available and reusable world wide.

As a natural result of the life cycle cost kind of thinking, field device maintenance strategies are changing from corrective and preventive practices towards predictive maintenance. This helps performing maintenance functions better based on the actual need. As a result of this evolution, production losses caused by poor performance and unnecessary process down time are expected to decrease. As a bonus, also maintenance costs will decrease. In order to be successful in predictive maintenance a lot of new diagnostics information is needed during the field device lifetime and the importance of field device diagnostics will grow to a new extent. Modern smart field devices have great potential in meeting these challenges [Riihilahti & Ojala, 2000].

### **5.1.3 Increasing Information Flow**

This section is based on materials from [Riihilahti & Ojala, 2000], [Nikunen et al., 2001], [Pyötsiä & Cederlöf, 1999], [Pyötsiä & Cederlöf, 2000].

Processes tend to grow in size and complexity. At the very same time information flow from smart field devices increases continuously. Today smart field devices can collect huge amounts of information about device performance and operation. An increased amount of diagnostics information itself is however not a target. On the contrary, there is a danger that the information flow coming from smart field devices increases the complexity and workload of operators and maintenance people rather than simplifying it. Therefore, the management of this information flow is the key to success. This huge amount of raw information must be interpreted automatically because the plant staff cannot manage all the available information from different devices. Measuring critical values of the field devices and concluding the need for maintenance with the help of long-term statistics and an automatic analysis is the answer.

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.

Process equipment condition monitoring and maintenance operations have the primary goal of ensuring process reliability and high capacity utilization in a cost-efficient and environment-friendly way. In this regard, it will be better to develop new solutions to assist proactive maintenance from individual field units to entire processes.

### **5.1.4 Existing Industrial Autonomous Maintenance Systems**

This section is based on materials from [METSO, 2002], [METSO, 2003], [FIELD BROWSER].



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 web browser.

#### **5.1.4.1 Neles FieldBrowser Features**

The Neles FieldBrowser automatically monitors the condition and performance of the field devices on-line without any upsets to the process. Data such as trends diagnosis (travel deviation, load factor and valve travel), counters, error messages, etc. can be monitored. If one of the limits is exceeded or something unexpected happens, the Neles FieldBrowser (Figure 5.1) will notify you.

- ❑ Automatic condition monitoring and messaging.
- ❑ Automatic diagnostics data reading and analysis.
- ❑ Automatic alert monitoring for HART device.
- ❑ Simple 3-level alert priority: "OK-Warning-Alarm"
- ❑ Install and forget Neles FieldBrowser.

When a field device is about to fail or the performance is reduced, you will be automatically notified by e-mail or short message to your pager or mobile phone. You can

also use your favorite Internet browser to check the status of all field devices. The status of thousands of field devices can be viewed within just a few seconds.

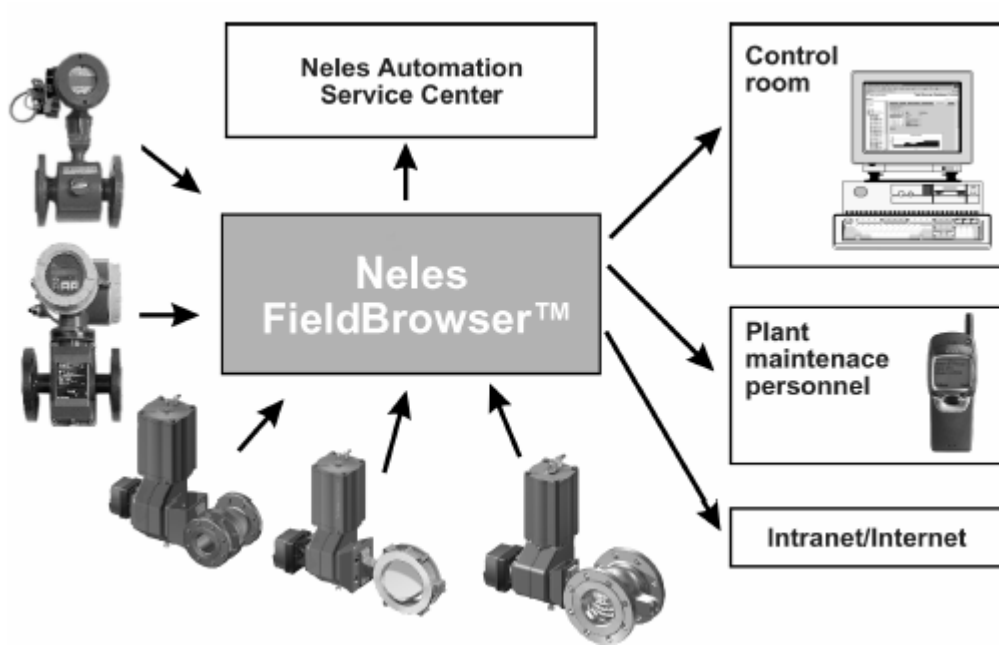


Figure 5.1. Neles FieldBrowser.

#### Remotely or locally

- ❑ Alerts via e-mail.
- ❑ Alerts via mobile or pager.
- ❑ Device status and diagnostic data via Inter/Intranet.
- ❑ Alerts for external database (ODBC)
- ❑ Diagnostics data sending via e-mail.

The diagnostics data retrieved from field devices can be automatically sent everywhere in the world as an E-mail attachment file. Check the performance from the office, home or leave it to Metso Automation. This feature allows you to analyze the alarm and preplan corrective actions.

### 5.1.4.2 Central Management of Control Valves

Metso Automation Valve Manager for HART Multiplexer Networks is a user-friendly and powerful tool to configure, diagnose and perform control valve calibrations and tests from a single workstation. Hazardous environments can be avoided because the valve diagnostics and configuration can be performed from a clean operator or maintenance room (Figure 5.2).

- ❑ Valve diagnostics data reading.
- ❑ Graphical diagnostics trends window.
- ❑ Diagnostics database.
- ❑ Valve configuration, configuration database.
- ❑ Valve testing, test result viewing, test database.
- ❑ On-line device variable monitoring and logging.
- ❑ Remote configuration of control valve characteristics.
- ❑ Process & valve info database.
- ❑ Three Security levels for users.
- ❑ Thousands of valves can be connected on-line.
- ❑ Simultaneous operation with Neles FieldBrowser

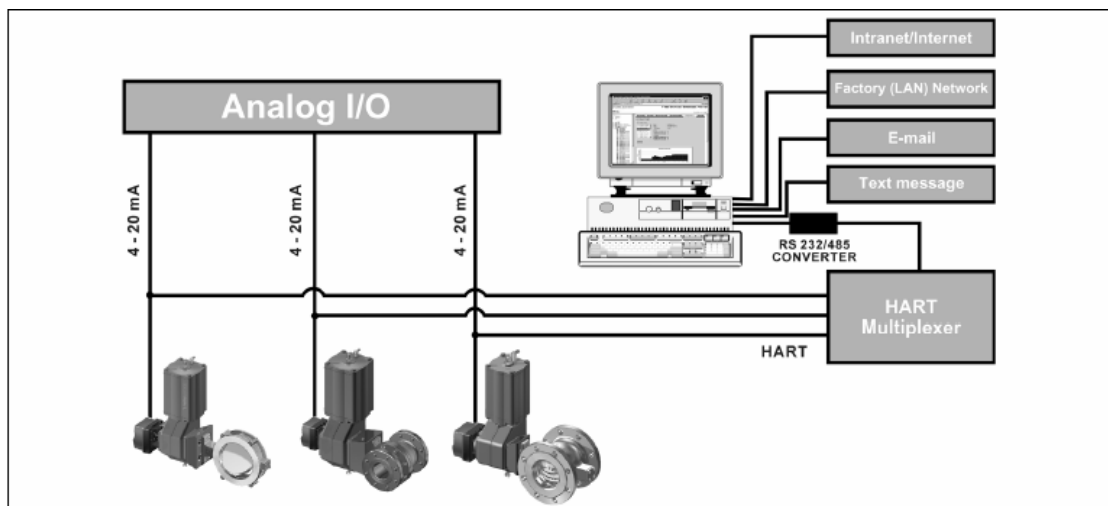


Figure 5.2. Central management of Control Valves.

### **5.1.4.3 Metso Automation's Field Bus Products**

A field bus (of one type or another) can be found across the whole plant structure. From Ethernet, through industrial LAN to control networks and discrete devices, it can bring a broad range of generic advantages. Field buses can also bring significant benefits to manufacturing systems in process control.

Field bus is an important technology in Metso's Future Care concept. Future Care is a long-term partnership providing technology know-how and resources to complement the plant owner's capabilities throughout the life cycle of the process and equipment. The integration of the plant assets and production information, which the field bus will ensure, is a key element in strengthening the plant competitiveness.

While the purchase of an asset or technology can be relatively simple, obtaining the best performance from it is more difficult. Future Care tools and solutions are designed to build long term knowledge out of the information the field bus devices provide. This knowledge is used to improve process performance, drive down maintenance costs and minimize shutdown time.

#### **Smart-Pulp PA**

Easy access for calibrating or reading diagnostics data is essential for most of the analytical transmitters in the pulp and paper industry. Often grade changes or unexpected process disturbances can cause headaches for operators.

For example, consistency transmitters on multi-grade machines may require several recipes to be configured into the transmitter memory for successful consistency control.

Field-bus-based solutions can provide calibration windows on the operator's monitor for recipe changes and diagnostics by clicking the mouse. This is a real benefit when real-time data is needed for decision-making.

Metso Automation has launched the first Profibus-based consistency transmitter, Smart-Pulp PA, to the market. The Foundation Fieldbus version will be launched soon. Those who are familiar with the analogue version of Smart-Pulp will not see any difference in

operating the transmitter. The same functionality is still available but users get data more easily than before.

### **ND800 FF/PA**

The ND800 Intelligent Field bus Positioner has evolved from Metso Automation's high integrity positioner platform. To ensure the optimum solution for specific applications, Metso Automation has developed ND800 for both Foundation Fieldbus H1 and Profibus PA. On-line valve diagnostics are continuously stored throughout the life of the device. The ND800 Intelligent Field bus Positioner utilizes sophisticated software to continuously receive accurate data related to the performance and condition of the field device. Essential on-line valve diagnostics maximize runtime performance, and can reduce valve maintenance expenditure by up to 50%.

The performance trends of the valve and its impending need for servicing are closely monitored and maintenance can be planned for when it is truly necessary. Unlike some other device vendors, Metso's Foundation Fieldbus device includes both AO and PID function blocks as standard. The AO control block application includes scaling and reversing options, while the PID control block application has advanced features including set-point ramp speed plus a derivation filter and selector. The closed loop response speed is easily adjustable and multiple tuning options are provided. The device can be used either as a basic device or a link master, allowing control in the field.

The ND800 Intelligent Field bus Positioner is just one element in an evolving dimension of process care solutions that promise to bring unprecedented process stability and efficiency.

Metso is playing an active role in developing open architecture device management software that provides the ability to integrate all products regardless of the protocol being used. By applying these latest technological developments to its long-established product reliability, Metso Automation is raising process plant efficiency to a new level.

## 5.2 Distributed Mobile Maintenance System for Smart-Device

### 5.2.1 Service Requestor Is a 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.

### 5.2.2 Intelligent Distributed Product's Maintenance

This is the state of the product maintenance domain today (Figure 5.3):

- 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.

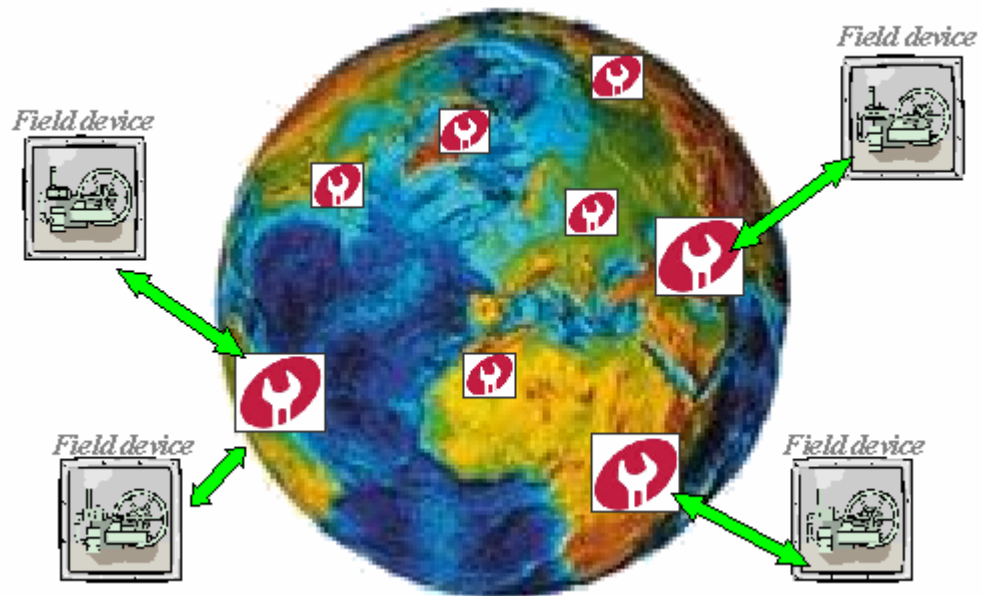


Figure 5.3. Maintenance today.

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 (Figure 5.4):

- 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.

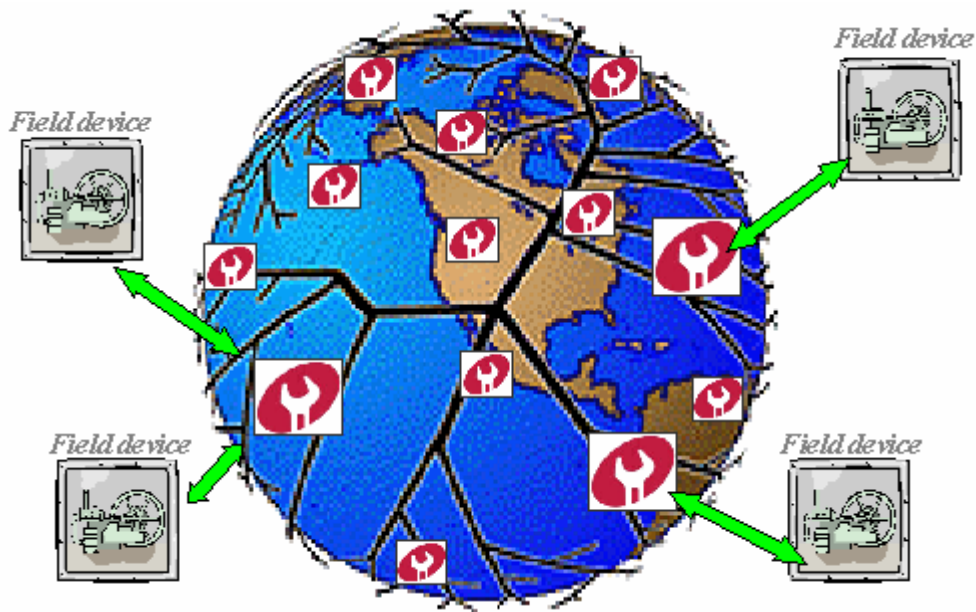


Figure 5.4. Maintenance tomorrow.

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.



### 5.2.3 Structure of Maintenance Web Service Platforms (Internal and External systems)

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 in our global maintenance web service network based on service platforms, we have to provide service platforms to both the field device and the Maintenance Center (Figure 5.5, Figure 5.6).

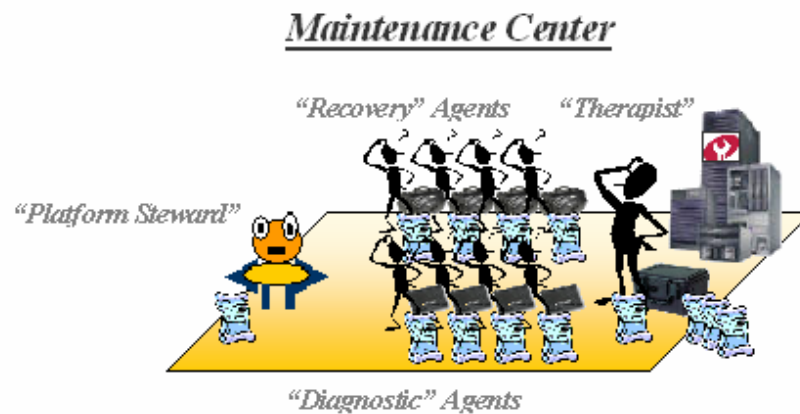


Figure 5.5. External System – Maintenance Center.

As we see, a Maintenance Center (Figure 5.5) 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.

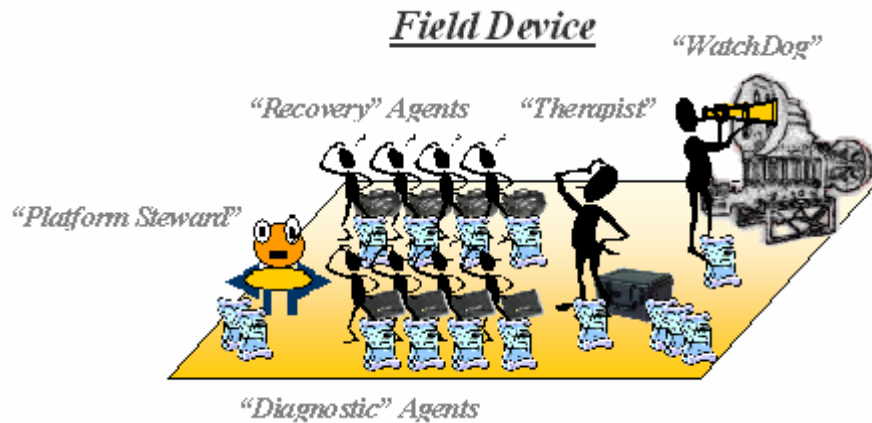


Figure 5.6. Internal System – Field Device.

A field device local maintenance service (Figure 5.6 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.

#### 5.2.4 Human Component 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 (Figure 5.7).

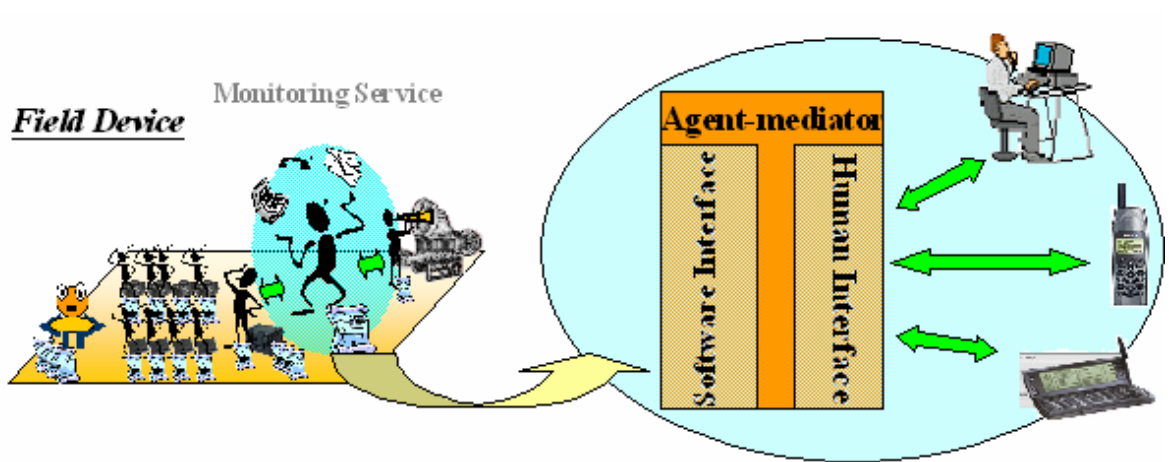


Figure 5.7. Human Monitoring Service.

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 (Figure 5.8). 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.

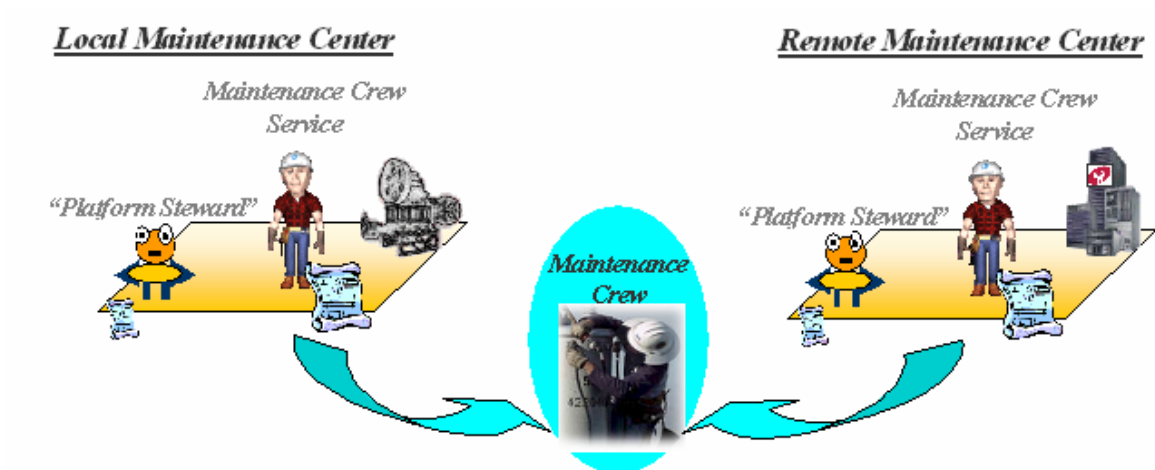


Figure 5.8. Human Maintenance Crew Service.

### 5.2.5 Maintenance Network

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 (Figure 5.9).

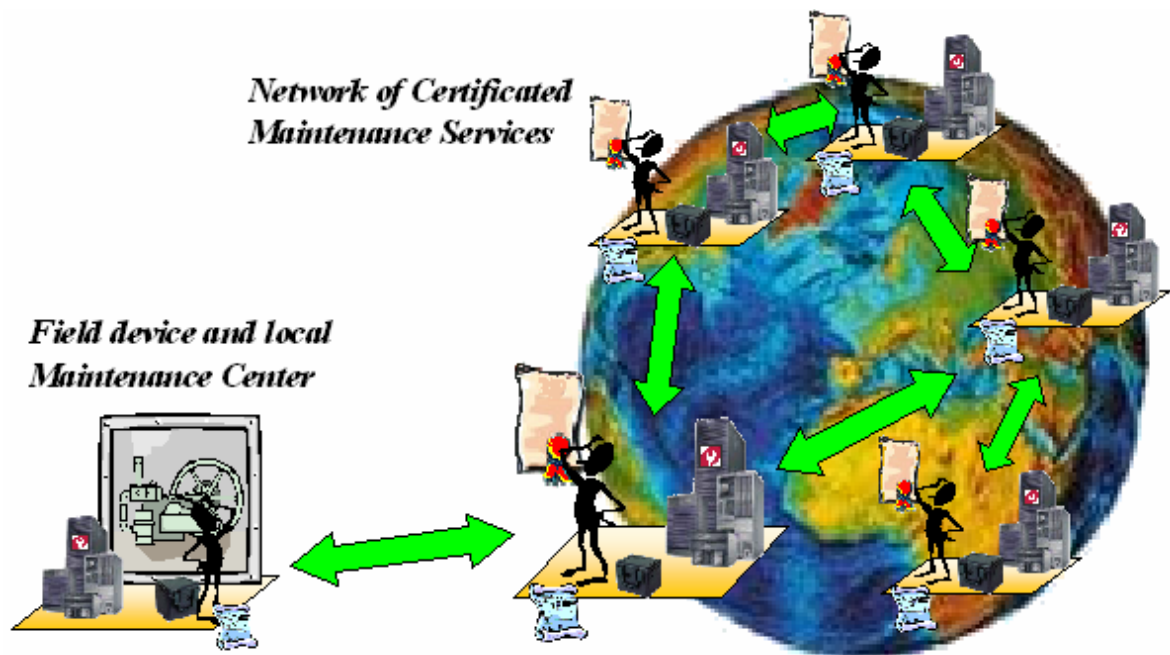


Figure 5.9. Network of certificated Maintenance Services.

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.

### 5.2.6 Maintenance Cases

Let’s 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

### 5.2.6.1 Remote Diagnostic

Remote diagnostic is a case, when some monitored parameters of a device differ from a normal state, however 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 the requests for diagnosis for similar cases 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 (Figure 5.10).

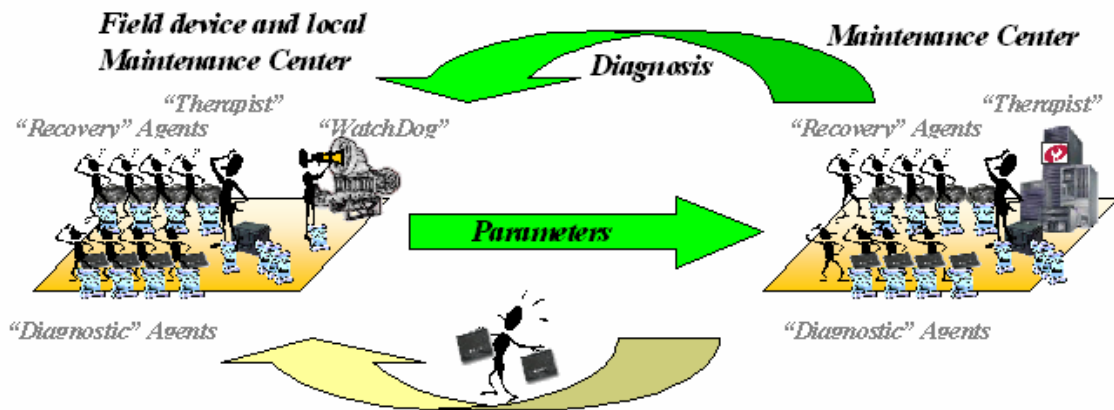


Figure 5.10. Remote diagnostic.

### 5.2.6.2 Recovery and Predictive Maintenance

Assume that a 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 (Figure 5.11).

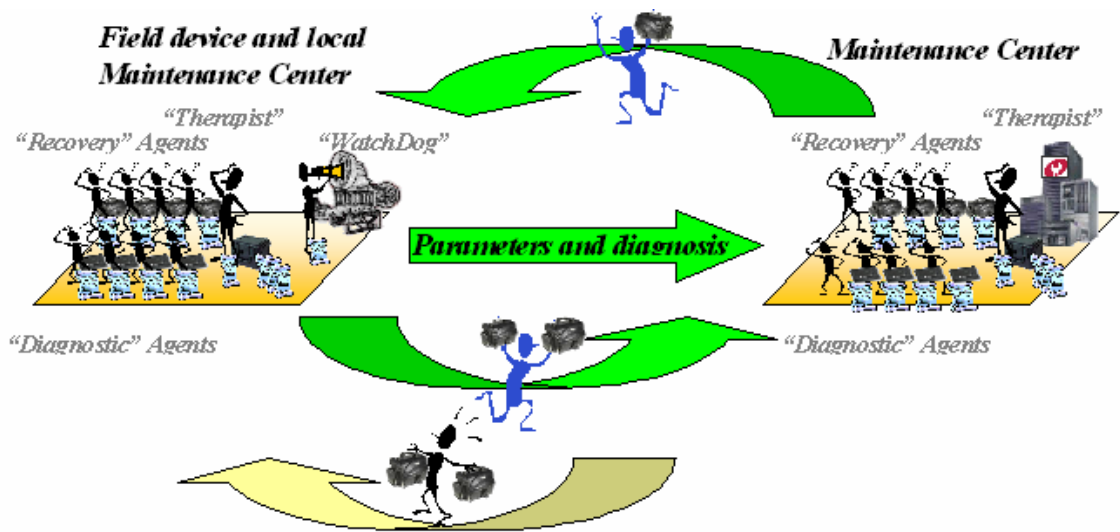


Figure 5.11. Recovery and predictive maintenance.

### 5.2.6.3 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 (Figure 5.12).

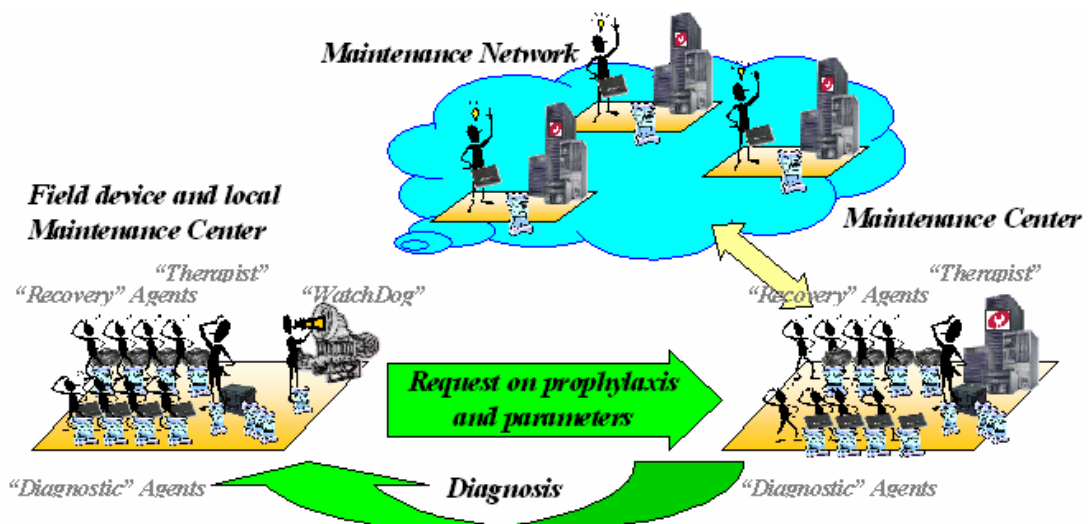


Figure 5.12. Remote preventive inspection.

**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 (Figure 5.13).

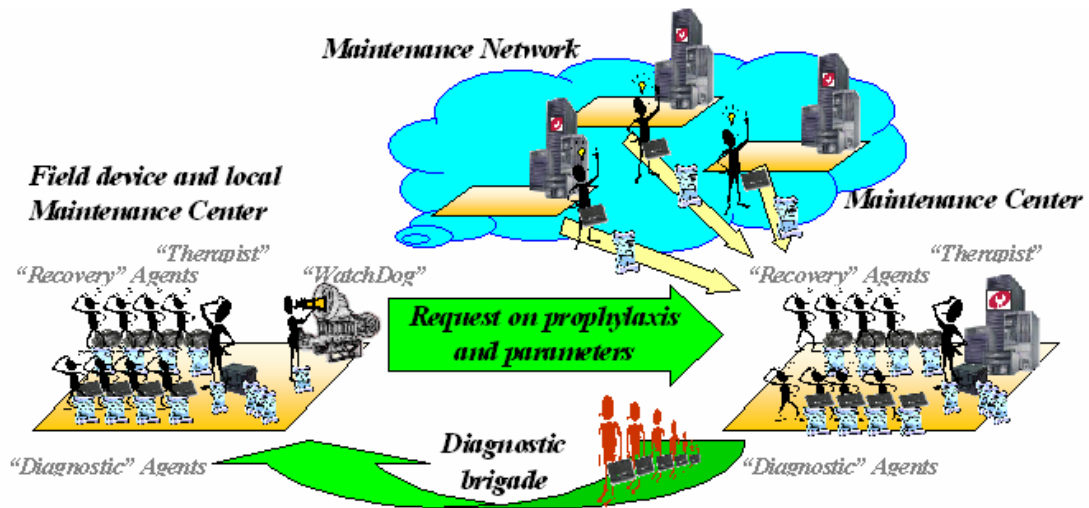


Figure 5.13. Local preventive inspection.

#### 5.2.6.4 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 (Figure 5.14).



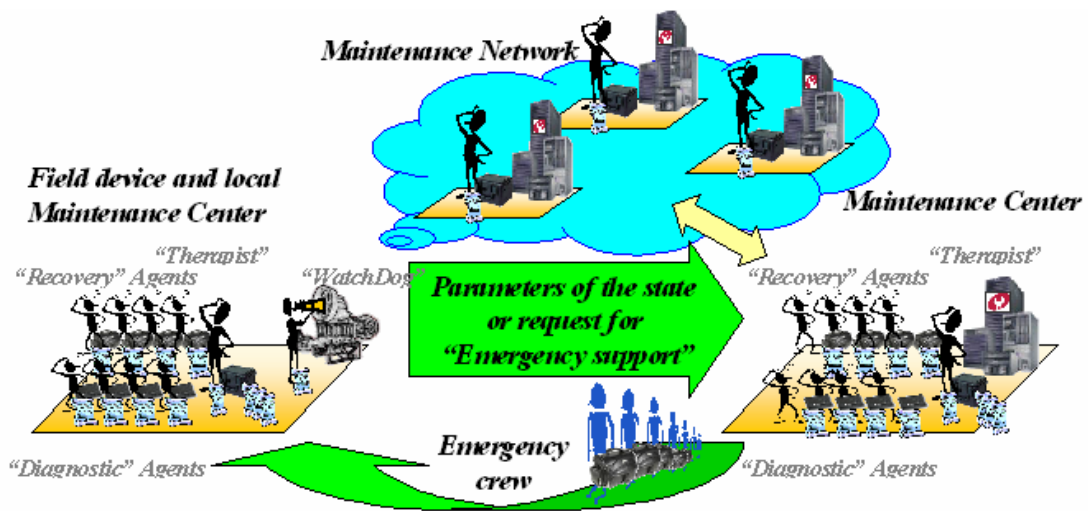


Figure 5.14. Emergency service.

#### 5.2.6.5 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 (Figure 5.15).

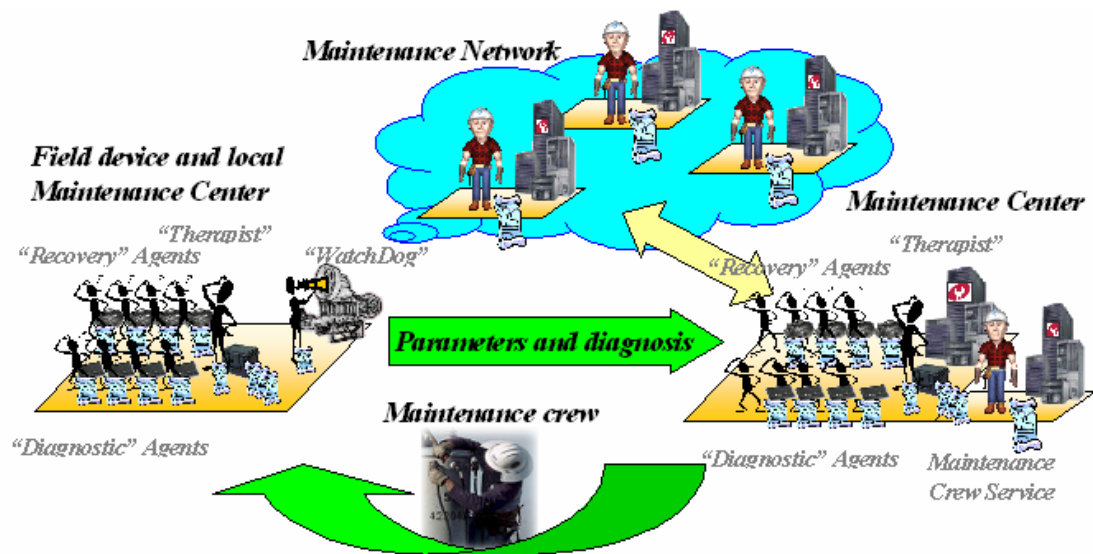


Figure 5.15. Human resource execution.

## 6 Conclusions

Nowadays the world is overcrowded by information, which is decentralized and non-shared for a wide circle of users, who need this information. Making the knowledge (information) available satisfies not only the users' requirements, but also it allows saving resources (money, human resources, etc.), which are expended for resources doubling. Thus the Semantic Web approach based on the creation and using of common ontologies is the more suitable solution for integration and shared using of information, knowledge, services (in one's own way they provide those information and knowledge) and, typically case resources. Using an ontology provides the context for creating accurate semantic metadata, which is the key to providing actionable information and business insight within the framework of information integration. The value of metadata has been long recognized, from data integration to application integration. It is only through semantic metadata that both humans and software can start to associate meaning with documents. This approach is accentuated especially now, when a new type of users like smart-device has appeared. They also need access to information and using of Web Services.

Nevertheless, resources and services (as a subclass of resources) are distributed via the Internet. Together with detached resources, there are modular resources, which are components of other more complex resource. Especially an approach to services (as a mobile components) is very good solution for organizing a shared using of this kind of a resource. Using of mobile resources (services) finds an application in various domains, when different services and sharable semantically annotated distributed resources are used. Particularly, this approach has a place in the industrial context, under organizing of corporative association (coalition) for shared using of common resources (services in context of the foregoing material) with the purpose of cost minimization and industrial process optimization. Organizing in a sense Industrial OntoHub (in the context of between-corporative cooperation), which accumulate information about stocked distributed shared resources, splendidly combine with the mobile resource's (service's) components approach and using network of platforms for mobile agent-carriers.

The emerging agent technology gives new opportunities to introduce mobility for resources (services), which in point of fact are represented by mobile agents. The agents are best suited for applications that are modular, decentralized, and changeable. The agent approach provides a useful means of integration and coordination of services.

In this work I consider an infrastructure of distributed Web service components, which can be discovered on 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. I am 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 (as subclass of resources) are heterogeneous and need to be preliminarily adapted via a common ontology. According to this problem, I propose an OntoShell approach to the creation of an Ontology-based universal integration environment. It allows transforming all resources (already existing and being developed) to semantically enabled resources for their integration. I propose both a centralized and a decentralized (P2P) interaction models for the components of this integration environment. Also, I consider a business model of such environment construction.

Now there is 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. We have a need to enable human presence in Semantic Web environment considering human to be a resource, not just a user in Semantic Web. Being naturally proactive, human can communicate with other resources and application acting as a web service. I consider a problem of a human representation in such environment of a resource-to-resource (service-to-service) communication, and propose approach to a resolving of this problem.

I 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 the industrial context. Field devices having an 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 the Semantic Web technology (ontological description and semantic annotation of service components); a 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 an enterprise, as well as between trading partners, suppliers, and customers, by offering the latest technology and open standards. It provides organizations with the possibility to create a cost-effective, extended enterprise by using an integration solution to get more return on information assets from existing ICT investments.

This system may be a reusable and strategic corporate asset, readily available to provide the next generation integration capability required to construct complex business processes.

It can help enterprises realize the benefits of the next generation of integration:

- Reduced total cost of integration
- Shortened reaction times: shortened development cycles translate to the time to deployment. This will allow an enterprise to be nimble and responsive in ways they never thought possible.
- Competitive differentiation: the system allows creating integrated; cross-functional business processes streamlined with intelligent workflow to effectively differentiate enterprise.
- Future profit: New technologies can be assimilated without the need for changes to the integration network or significant investments in additional software. And these new technologies will be able to interact with existing ICT assets.

## REFERENCES

- [AGENTCITIES] The Agentcities Network. URL: <http://www.agentcities.net/>
- [AGENTCITIES, 2002] Agentcities Network Architecture Recommendation, ACTF-REC-00001, 2002.
- [Ankolekar et al., 2001] 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.
- [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.
- [Burg, 2002] 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.
- [Breger, 2003] Breger Michael, Agent Technology @ Siemens, URL:<http://www.cs.helsinki.fi/u/hhelin/opetus/oat/>, 2003.
- [Burmeister et al., 1993] B. Burmeister, A. Haddadi, and K. Sundermeyer. *Generic configurable cooperation protocols for multiagent systems*. From Reaction to Cognition --5<sup>th</sup> European Workshop on Modeling Autonomous Agents in a Multi-Agent World (MAAMAW'93), 1993.

- [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.
- [DAML-S, 2002] “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>
- [Davis, 2002] J. Davis, D. Fensel, and F. van Harmelen, *Towards the Semantic Web: Ontology-Driven Knowledge Management*, Wiley, 2002.
- [Dickinson, 2001] Dickinson, The Interface as agent: a comparative review of human-agent interaction. In review, 2001.
- [Ding et al., 2002] Ying Ding, Dieter Fensel, Michel Klein, Borys Omelayenko, *The semantic web: yet another hip?*, Data and knowledge Engineering, 2002.
- [Farrell & Kreger, 2002] J. A. Farrell, H. Kreger, *Web services management approaches*, IBM Systems Journal, Vol. 41, URL:<http://www.research.ibm.com/journal/sj/412/farrell.html>, No. 2, 2002.
- [Fensel, 2001] D. Fensel. *Ontologies: A silver bullet for Knowledge Management and E-Commerce*. Springer 2001
- [Fensel & Bussler, 2002] D. Fensel, C. Bussler, *The Web services modelling framework WSMF*, URL:<http://www.cs.vu.nl/~dieter/wese/wsmf.paper.pdf>, 2002.
- [Fensel & Musen, 2001] Dieter Fensel, Mark A. Musen, *The Semantic Web: a brain for humankind*, Intelligent Systems, IEEE, March/April 2001.
- [Fensel et al., 2002] Dieter Fensel, Frank van Harmelen, Ying Ding, Michel Klein, Hans Akkermans, *On-To-Knowledge: Semantic Web Enabled*

- URL:[http://home.broadpark.no/~rhenge/rob/papers/2002\\_otk\\_ieee.pdf](http://home.broadpark.no/~rhenge/rob/papers/2002_otk_ieee.pdf), 2002.
- [Fethi, 2002] Fethi A. Rabhi, School of Information Systems University of New South Wales, Sydney 2052 (Australia) and Sergei Gorlatch, Technical University of Berlin Sekr, FR 5-6, Franklinstr. 28/29 D-10587 Berlin (Germany), “*Patterns and Skeletons for Parallel and Distributed Computing*”, URL:<http://www.cs.cf.ac.uk/User/David.W.Walker/papers/root.pdf>, October 2, 2002
- [FIELD BROWSER] “*Field browser system for Control Valve maintenance*”, 9 FB 20 Issue 4/99, URL:<http://www.strauja.lt/vt/pdf/9FB20EN.pdf>
- [Finin et al., 1992] Finin, T. et al. *Specification of the KQML Agent Communication Language*. The DARPA Knowledge Sharing Initiative, External Interfaces Working Group. 1992
- [FIPA, 2000] *The FIPA Standard for Interoperating Software Agents*. The Foundation for Intelligent Physical Agents, URL: <http://www.fipa.org/>, 2000.
- [FIPA, 2001(a)] *FIPA ACL Message Structure Specification*, FIPA00061, 2001. URL: <http://www.fipa.org/specs/fipa00061/>
- [FIPA, 2001(b)] *FIPA Interaction Protocol Library Specification Specification*, FIPA00025. URL: <http://www.fipa.org/specs/fipa00025/>, 2001
- [iPlanet, 2002] “*iPlanet Application Server Enterprise Connector for CICS*”, URL: <http://docs-pdf.sun.com/806-5504/806-5504.pdf>”, 2002.
- [JXTA] *The JXTA Project*. URL: <http://www.jxta.org/>
- [Kreger, 2001] Heather Kreger, IBM Software Group, *Web Services Conceptual Architecture*, URL:<http://www->



- 3.ibm.com/software/solutions/webservices/pdf/WSCA.pdf, 2001.
- [Laamanen, 2001] Laamanen Heimo, FIPA Agent Framework, URL:[http://aut-pc29.hut.fi/kurssit/as-116-140/sem\\_a01/FIPA\\_agent\\_framework.pdf](http://aut-pc29.hut.fi/kurssit/as-116-140/sem_a01/FIPA_agent_framework.pdf), 2001.
- [Laamanen & Helin, 2001] Laamanen Heimo, Heikki Helin, Software Agent Technology, FIPA Agent Framework, URL: <http://www.cs.helsinki.fi/u/hhelin/opetus/oat/>, 2001.
- [Laasri et al., 1992] B. Laasri, H. Laasri, and V. Lesser. *A generic model for intelligent negotiating agents*. International Journal on Intelligent Cooperative Information Systems, 1992
- [METSO, 2002] 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/$File/wwwAutom1_2002all.pdf), 2002.
- [METSO, 2003] Metso Automation Customer Magazine Vol. 10, Issue 1, 2003, “Automation. Metso Future Care is here today”, URL: [http://www.metsoautomation.com/Automation/magazinebank.nsf/Resource/wwwAu103/\\$File/wwwAu103.pdf](http://www.metsoautomation.com/Automation/magazinebank.nsf/Resource/wwwAu103/$File/wwwAu103.pdf), 2003.
- [Mikalsen & Rouvellou, 2001] Thomas Mikalsen, Isabelle Rouvellou, *Reliability of Composed Web Services From Object Transactions to Web Transactions*, Stefan Tai, IBM T.J. Watson Research Center, New York, USA, URL:[http://www.research.ibm.com/AEM/pubs/web\\_services\\_oopsla2001.pdf](http://www.research.ibm.com/AEM/pubs/web_services_oopsla2001.pdf), 2001
- [Nikunen et al., 2001] J. Nikunen, M. Salmenperä, H. Koivisto, Global Condition Monitoring Network, Automaatiopäivät 2001.

- [Ojala, 2001] M. Ojala, *SW agent technology in global field device management*, URL: [http://www.automationit.hut.fi/julkaisut/documents/seminars/sem\\_a01/Ojala.pdf](http://www.automationit.hut.fi/julkaisut/documents/seminars/sem_a01/Ojala.pdf), 2001.
- [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.
- [Parunak, 1998] H. Parunak, *Practical and Industrial Applications of Agent-Based Systems*, 1998.
- [Peltz, 2003] C. Peltz, *Applying Design Issues and Patterns in Web Services*, URL: <http://www.devx.com/enterprise/Article/10397/0/page/1>, 7 January 2003.
- [Pitkänen & Shuling, 1998] Pitkänen Hannu, Zhang Shuling, *Introduction to agent theory and review on agent applications in robotic and automation systems*, URL:<http://www.automation.hut.fi/edu/340autumn98/>
- [Pyötsiä & Cederlöf, 1999] J. Pyötsiä, H. Cederlöf, *Advanced Diagnostic Concept Using Intelligent Field Agents*, ISA Proceedings, 1999.
- [Pyötsiä & Cederlöf, 2000] J. Pyötsiä, H. Cederlöf, *Remote Wireless Presence in Field Device Management*, ISA Proceedings, 2000.
- [Riihilahti & Ojala, 2000] J. Riihilahti, M. Ojala, *On-line Diagnostics for Maintenance of Smart Field Devices*, ISA Proceedings 2000.
- [Rodriguez & Sallantin, 1998] J.M. Rodriguez, J. Sallantin, *A system for Document Telenegotiation (negotiator agents)*, COOP'98: 3rd International Conference on the Design of Cooperative Systems, Cannes, France, May 26-29, 1998, pp. 61-66.
- [Seilonen, 2001] Seilonen Ilkka, *Introduction to agent technology*, URL:[http://aut-pc29.hut.fi/kurssit/as-116-140/sem\\_a01/Introduction.pdf](http://aut-pc29.hut.fi/kurssit/as-116-140/sem_a01/Introduction.pdf), 2001.

- [Sheth, 2003] 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>, 2003.
- [Tommila et al., 2001] T. Tommila, O. Ventä, K. Kostinen, “Automation technology review 2001: Next Generation Industrial Automation – Needs and Opportunities”, URL: [www.vtt.fi/tuo/projektit/ohjaava/ohjaava1/atr\\_2001.pdf](http://www.vtt.fi/tuo/projektit/ohjaava/ohjaava1/atr_2001.pdf), 2001.
- [UDDI, 2002] UDDI. *The UDDI Technical White Paper*, URL: <http://www.uddi.org/>, 2002.
- [Virtej, 1998] Virtej Iuliana, Software agent concepts and technologies, URL:<http://www.automation.hut.fi/edu/340autumn98/virtej/virtej.htm>, 1998.
- [VISION] VISION project – description. “Next generation knowledge management”, EU project IST-2002-2.1.2, URL: [www.cognit.no/home\\_multi/html/VISIONProjectDescription.pdf](http://www.cognit.no/home_multi/html/VISIONProjectDescription.pdf), 2002.
- [W3C(a)] W3C org, WSDL (Web Services Description Language) URL:<http://www.w3.org/TR/wsdl>.
- [W3C(b)] W3C Web Services Activity. URL: <http://www.w3.org/2002/ws/>, 2002.
- [W3C(c)] W3C Extensible Mark-up Language (XML) Activity. URL: <http://www.w3c.org/XML/>
- [W3C(d)] W3C Resource Description Framework (RDF) Activity. URL: <http://www.w3c.org/RDF/>
- [W3C, 2000] Simple Object Access Protocol (SOAP) 1.1, W3C Note 08 May 2000 URL:<http://www.w3.org/TR/SOAP/>, 2000.

- [webMethods]                    *“The webMethods Integration Platform. Delivering the Enterprise Dial-tone”*, webMethods: The Business Integration Company,  
URL:[www.webmethods.com/PDF/EDT\\_Brochure\\_HIGH.pdf](http://www.webmethods.com/PDF/EDT_Brochure_HIGH.pdf),  
2002.
- [WebServices]                    URL: <http://www.webservices.org>.
- [Wooldridge, 2002]                M. Wooldridge, *An Introduction to Multiagent Systems*, John Wiley & Sons, 340 pp.
- [Wooldridge, 2003]                Wooldridge                    Mike,                    Multiagent                    Systems,  
URL:<http://www.cs.helsinki.fi/u/hhelin/opetus/oat/>, 2003.