Oleksandr Kononenko

ONTOLOGICAL SUPPORT FOR INDUSTRIAL MAINTENANCE OF SMART-DEVICES

Master's Thesis
Mobile Computing
31.10.2003

University of Jyväskylä

Department of Mathematical Information Technology

Author: Oleksandr Kononenko

Contact information:

Roninmäentie 1G 4B,
40500 Jyväskylä, FINLAND,
olkonone@cc.jyu.fi, tel. +358(40)5158866

Title: Ontology Support for Industrial Web Services

Work: Master's Thesis

Pages: 75

Study program: Mobile Computing

Keywords: Semantic Web, ontology, semantic web enabled web services, ontology management, ontology engineering, industrial maintenance

Abstract:

The need of having metadata for delivering new integration technologies has been recognized as one of the most important directions of ICT development. Nowadays tools for industrial maintenance need are being developed for high-technological distributed systems designated to condition monitoring, diagnostics and maintenance of complex equipments and maintenance-oriented environments spread all over the world. High automation within such systems and the autonomy of their components present great challenges for developers. Combining expert knowledge kept by experts with automated access to it by heterogeneous system components requires web services and a semantic technology involved for solving problems of interoperability, knowledge representation and standardization. The Semantic Web Activity led by W3C is aimed at designing a comprehensive framework for the development of future metadata-based systems, and the development of ontologies is seen as a nearly universal approach for solving many problems. In this thesis, specific needs of industrial web services are considered and a schema of appropriate ontology support is provided along with conceptual design of the *OntoServ.Net* environment, a semantic-based maintenance system for industrial devices. Results of this work will be used as a base of future research and development in the field of industrial applications of Semantic Web technology.

ACKNOWLEDGEMENTS

This part of the thesis was surely the easiest for me, because I do know whom and why I want to thank. These are people who have made an impact on my life and studies during last year and helped me on my way to obtain the degree of Master of Science, here in Jyväskylä.

I thank Vagan Terziyan and Helen Kaykova, MIT Department (University of Jyväskylä), for supervision and help in my educational activities. I appreciate their guidance and advice in the student exchange programme in which I participated.

I thank Ass. Prof. Vagan Terziyan and Dr. Jouni Pyötsiä (Metso Corporation Headquarter) for useful consultations and supervision during my practice training and thesis writing. Also I wish to mention my friends and colleagues from Industrial Ontologies group, Andriy Zharko and Oleksiy Khriyenko. The hottest discussions we had let us find the truth in many problematic topics, part of which is presented in this thesis.

Also, I want to thank Prof. Pekka Neittaanmäki and the staff of Agora Center for their important support and excellent environment provided for my research during year 2003 within the Innovations to Business and Communication Technology (InBCT) project.

Oleksandr Kononenko

University of Jyväskylä, 31.10.2003

TERMS AND ABBREVIATIONS

DAML

DARPA Agent Markup Language

A language aimed at representing semantic relations in a machine-readable form and associating information with ontologies.

EAI

Enterprise Application Integration

The term used for information systems that bind together many applications within an enterprise, typically dealing with the scheduling and control of information flow between them. EAI is often built on top of Middleware.

Messaging

Creating, storing, exchanging, and managing data messages across a communication network. The two main ways are publish-subscribe and point-to-point.

Metadata

Data that describes other data. Often deals with the format, search details or authorship of the underlying data.

Ontology

A conceptual representation of the entities, meanings, and relationships within a specific domain of knowledge.

RDF

Resource Description Framework. A broad W3C standard framework for description methods of any Internet resource in XML machine-processable statements.

RDFS

RDF Schema. A framework for the representation of vocabularies and ontologies for Resource Description Framework.

Semantic Web

A conceptual web built on top of the World Wide Web in which all identified resources have a machine-processable semantic description data attached for intelligent processing by software-agents.

UDDI

Universal Description, Discovery, and Integration, a de facto standard for service registries on the Web.

Web Service

Technology for the development of standalone service-software components for distributed computing, accessible on the Internet via system-independent XML-based protocols.

XML

Extensible Markup Language. A specification for computer-readable documents data formatting. Standardized by the W3C. XML defines data elements using a tree structure.

CONTENTS

1 INTRODUCTION

1.1 Preface

The thesis is made as a part of the research activities of the Industrial Ontologies Group, whose goal is to provide Semantic Web based IT solutions for industry.

Semantic Web is on the verge of becoming an industry-strong and mature technology for building information management systems. Ideas of having a comprehensive framework for semantic-aware information processing models have found support outside the research community and started to draw the attention of tool developers. A certain amount of tools already exists and presents a basis for Semantic Web based products.

There are still numerous problems to be solved. Design patterns for Semantic Web based products are only about to be outlined along with basic support of existing technologies. By making this we link existing technologies with novel approach.

Ontology support is an ontology-based information management infrastructure existing in a certain environment. Ontologies are the core elements of any semantic web system. Due to this, ontology support for such system defines its basic information-related features and design, and influences the internal structure of software.

1.2 Research Problem Statement and Related Work

Problems related to ontology support are from a more general problem domain – interoperability between heterogeneous information systems, semantic-based communication and co-operation in an open dynamic environment. In the context of Web Services technology development, new possibilities for joining it with the Semantic Web vision become very promising for future applications and they are observed in the thesis.

The importance of Web services has been recognized as a significant technology in the evolution of the Web and widely accepted by industry and academic research. The Web Services technology resides on the edge of limitation of the current web and desperately needs an advanced semantic-oriented approach found in Semantic Web, which will enable automatic discovery, selection, invocation, composition, monitoring of services (and more).

Web services, being self-described and self-contained modular active components, will become the key elements in assembling intelligent infrastructures for EAI and e-Business in the near future. The challenge is to make web services automatically usable by autonomous applications (artificial agents). This is going to be solved by creating effective frameworks, standards, appropriate ontology and software support for automatic web service discovery, execution, composition, interoperation and monitoring [1].

Current industrial systems use only initial and very partial solutions of the ultimate problem. Existing de-facto standards for web service description (WSDL [2]), publication, registration and discovery (UDDI [3]), binding, invocation, communication (SOAP [4]) provide merely syntactical capabilities and unfortunately do not really cope with service semantics. Known industrial implementations such as HP E-speak [5] base on these standards and do not completely solve the challenge of semantic service interoperability. It should be mentioned that major industrial players realize the necessity of further targeted joint research and development in the field [6].

Recent research and standardization activities within the DAML community resulted in offering a semantic service markup language, DAML-S [7], which is a substantial part of standardization support required for the development of semantic-enabled service-oriented systems.

This thesis is about applications of Semantic Web in industrial systems and, specifically, Semantic Web enabled Web Services in industrial maintenance domain. The goal of the thesis is to determine what is still to be done concerning ontology-related issues in order to develop automatically discoverable, composable and usable web services in a new kind of environment based on the next generation of the Web. Conceptual frames for this development are under intensive discussion and some proposals already appear (e.g., WSMF [8]).

In general terms, this thesis considers possible implementation of ontology-based service integration, standardization of information representation and information exchange on the first stage of new technology propagation. Main questions about Semantic Web technology application to be answered are: why we need Semantic Web enabled Web Services? what should be considered for the ontology support development of semantic-enabled web

services? and what can be the first steps in providing strong industrial ontology-based solutions?

An ontology-based approach is developed in this thesis using the case of industrial network of maintenance services for smart-devices. This work is an attempt of defining an ontology-supported environment, where semantic-enabled resources are built and integrated into a network of cooperative services. Requirements and strategy for the implementation of such environment are considered.

The thesis is closely related to the works made by my colleagues from the Industrial Ontologies Group, where Semantic Web based concept of service network is further detailed from the point of view of its architecture [9], service design and maintenance-related functionality [10].

1.3 Structure of the Thesis

The thesis contains the following parts. First, general information about the Semantic Web vision and the role of ontologies is given. Later, the thesis includes a description of OntoServ.Net, a concept developed for an industrial system, which is a reference system with ontology support to be discussed. Main reasons for using Semantic Web technology are provided here. In the following part of the thesis a general schema of information infrastructure for OntoServ.Net is presented along with solution descriptions of related problems and technical details concerning tools that can be used for the implementation of the provided schema and conclusions regarding possibilities of Semantic Web applications in industrial systems.

Some related work considering a general approach to the design of a communication infrastructure for service-based systems is presented in additional section at the end of this work.

2 SEMANTIC WEB, WEB SERVICES AND ONTOLOGIES

2.1 Semantic Web vision

The *Semantic Web* is the presentation of machine-processable semantics of *data* on the World Wide Web. It is a collaborative effort led by World Wide Web Consortium (W3C) with the participation of a large number of researchers and industrial partners. It is based on the Resource Description Framework (RDF) and new ontology languages such as Web Ontology Language (OWL) and DARPA Agent Markup Language (DAML), which integrate a variety of applications using XML for syntax and URIs for naming.

The Semantic Web is an initiative with the goal of extending the current Web and facilitating Web automation, universally accessible web resources, and the 'Web of Trust', providing a universally accessible platform that allows data to be shared and processed by automated tools as well as by people. As defined in the statement of Semantic Web Activity:

> *"The Semantic Web is a vision: the idea of having data on the Web defined and linked in a way that it can be used by machines not just for display purposes, but for automation, integration and reuse of data across various applications."*

This vision assumes annotating artifacts being involved in a semantic-enabled framework with machine-interpretable descriptions of their underlying semantics, and provides mechanisms for automated reasoning about them. To facilitate this, new web languages and technologies are being developed, ontology and schema integration techniques along with Web Services Integration Standards are being defined (e.g. UDDI, ebXML, e-Speak [11]), examined and refined. The success of the Semantic Web will depend on a widespread adoption of these technologies.

Despite of the novelty of the Semantic Web technology, it is on a way to developing a basis for the future of information technology which will revolutionize all spheres of Web [12], particularly e-commerce, e-business, Enterprise Application Integration and Web Services.

The set of standards used in Semantic Web technology has a layered structure, which is linked with the existing web-technology and extended with new layers that introduce support for description of semantics. The computing stack of SW standards is presented in Fig. 1.

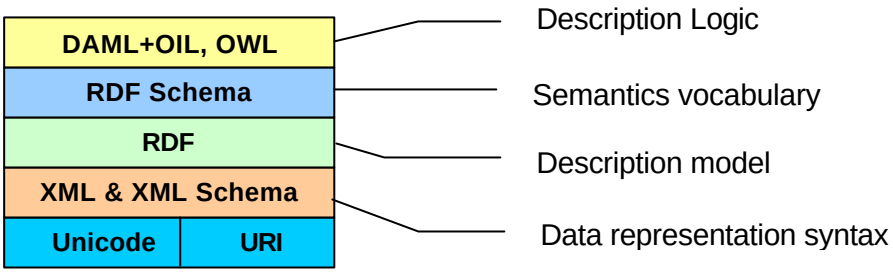


Fig. 1. Current Semantic Web standards stack

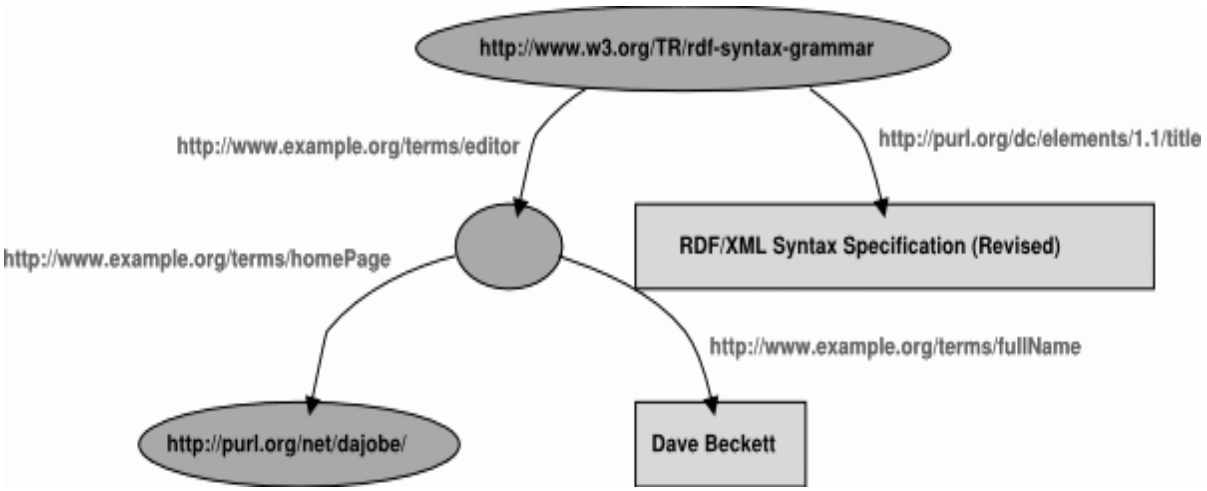
- **XML & XML Schema**

The tagging mechanism utilized in markup languages assigns labels to pieces of data. Internet-oriented markup languages use Unicode for internationalization purposes and URI as a referencing mechanism. Extended in XML with tag namespaces ([13], [14]) this markup has become a powerful syntax with many applications for data representation and exchange between information systems. Schema language for XML describes the vocabulary of an XML-document and restricts its structure; a document can be validated against schema to ensure its correct structure and correct processing by applications supporting this document schema.

- **RDF**

The Resource Description Framework (RDF) [15] provides a general-purpose language for representing information on the Web. RDF is based on a model, which expresses everything in statements of relations between entities: Object – Relation - Subject (e.g. X "is property of" Z; X "has value" Y, etc.). Resource is the central concept of RDF. Resources represent anything, from web pages to people. Properties express specific aspects, characteristics, attributes, or relations of a resource. Statements are special constructions that are composed of a specific resource together with a property and its value for that resource. Values of properties can be resources themselves in turn. A value can also be a literal, a primitive term that is not evaluated by an RDF processor.

The data model is very similar to a basic directed graph, a very well understood data structure in computer science. Parts of such graph are RDF statements. In Fig. 2 there is an example of an RDF-graph.



Resource with URI ***http://www.w3.org/TR/rdf-syntax-grammar*** *has title "****RDF/XML Syntax Specification (Revised)****" and its editor's name is "****Dave Beckett****", whose home page is resource with URI* ***http://purl.org/net/dajobe***

Fig. 2. Example of an RDF-graph

RDF uses XML as *serialization syntax*. Though, actually, there are also other syntaxes. For example, RDF statements can be written in form of so called triples that are sets of **<resource property value>** elements. Such triples denote Subject-Predicate-Object statements expressive enough to describe both information and real-world objects.

The Resource Description Framework is a foundation for representing and processing metadata; it provides interoperability between applications that exchange machine-understandable information on the Web and can be used in many application areas [16]:

- in *resource discovery* (to provide better search capabilities);
- in *cataloging* for describing the content and content relatio nships;
- in *content rating and* describing collections of pages that represent a single logical "document";
- for describing *intellectual property rights* of Web pages and for expressing the *privacy policies* of a Web site as well as the site user's *preferences.*
- *digital signatures* for building the "Web of Trust" – the ultimate aim of Semantic Web - for e-commerce, enterprise integration and other applications.

- **RDF Schema**

  Similarly to XML documents, an RDF document needs a schema to conform with. RDF Schema does the same thing for RDF that DTD and XML Schema do for XML. RDF Schema defines terms to be used in RDF statements (relation names, object classes) and schema restrictions (such as number of properties an object can have, for instance). RDF Schema also defines the hierarchy of object classes by making use of properties and restriction inheritance, which makes RDF Schema more than just a list of terms. RDF Schema fixes semantics of defined terms for referencing from RDF documents.

  RDF statements are used to write RDF Schema vocabulary descriptions. The additional descriptive power of RDF Schema comes from a collection of RDF resources described RDF Schema Specification. These resources are used to determine characteristics of other resources, such as the domains and ranges of properties.

- **Ontology languages (DAML+OIL, OWL)**

  Both RDF and RDF Schema provide basic features for information modeling and a simple knowledge representation mechanism for the description of resources. However, more modeling primitives than those that are used in RDF Schema are needed, such as data types and consistent facilities for expressing enumerations and description logic. RDF Schema extensions, which provide vocabulary based on ontology formalism, have richer descriptive features adopted from knowledge representation languages, are appropriate for ontological reasoning an cover this need. DAML+OIL and its successors OWL present ontology languages, which are put in the base of semantic technology at the moment.

  A DAML+OIL knowledge base is a collection of RDF triples. DAML+OIL prescribes a specific meaning for triples that use the DAML+OIL vocabulary. OWL is a DAML+OIL restricted standard from W3C. There are three levels of OWL defined (OWL Lite, OWL DL and OWL Full) with progressively more expressiveness and inferencing power. These levels were created to make it easier for tool vendors to support a specified level of OWL. DAML+OIL and OWL both depend on RDF/S semantics.

2.2 Ontologies: What, How and Why?

Communication between humans has no analogs by its successfulness and efficiency because we possess a large amount of knowledge that can be proactively used for communication: languages, grammar, rules for communication and, finally, specific domain knowledge of the subject of dialog. Whereas humans can normally distinguish between different interpretations of the information content, software will only be able to operate correctly if it carries sufficient explicit information to ensure this.

In order to achieve efficient communication between information systems, common understanding of the used vocabulary needs to be established, similarly to understanding of same concept expressed in different languages between humans.

There is a growing interest in the use of ontologies in agent systems as a means to facilitate interoperability among diverse software components, in particular, where interoperability is achieved through the explicit modeling of the intended meaning of the concepts used in the interaction between diverse information sources, software components and/or service-providing software [17]. Agent based systems (consisting of independent entities) via adoption of a common ontology will achieve the possibility of interoperation without misunderstanding, yet retaining a high degree of autonomy and flexibility.

Ontologies are key enabling technology for building the Semantic Web [12]. Ontology formalism was taken initially from philosophy to Artificial Intelligence domain and then to Semantic Web concepts. Ontologies are seen as universal way of expressing knowledge about the world and, practically, about more specific things like business, science, web resources, industrial processes, etc. Described as formal specifications of conceptualizations [18], ontologies provide a vocabulary of basic terms, understanding of objects sorts, their properties, relations between objects that are possible in a particular domain of knowledge, and their specification of the meaning. That explicitly defined vocabulary will be a basis for communication and interoperability across people and application systems.

The main use of an ontology is the development of a basis for communication between computer systems independently of the individual system technologies, information architectures and application domains.

Some clarity has to be established in the understanding of ontology. An ontology is not just a vocabulary of terms. It provides a set of primitives that can be used for building meaningful higher level knowledge. An ontology specifies terms, which correspond to the most basic concepts of the target domain and relationships between terms providing the semantic basis for the terminology. And an ontology is not just a taxonomy or classification of terms. Although building a taxonomy contributes much to the semantics of specified terms, ontologies can include richer term relationships. And with those relationships it is possible to express domain-specific knowledge, without inclusion of domain-specific terms.

In some sense, computers have been using rudimentary ontologies in the form of data dictionaries, enterprise data schemas, web architectures and taxonomies. As systems begin making fuller use of ontologies, computers can make sense of unstructured and semi-structured materials and take on a significantly more extensive role in processing transactions because they 'know' how a piece of information (document, fact, rule, etc.) relates to other pieces of information [19].

Ontological representation not only provides a structure of knowledge in an explicit and machine-readable form, but also enables integration. The ultimate goal is the development of reusable ontologies that can be applied across multiple disciplines.

Adequate data interpretation is an obligatory prerequisite of adequate behavior. In order to make computers process and store information in universal structures that allow intelligent processing, rather than just storing sequences of bytes in structured records, more advanced languages then XML and even RDF are required.

Different approaches to semantic technology are distinguished by the different ways knowledge representation languages express the connections between concepts:

- Taxonomies and Thesauri have very simple connections
- RDF has somewhat more complex connections
- DAML and OWL have very powerful logical connections

RDF and RDF Schema provide basic features for information modeling and a simple knowledge representation mechanism for Web resources. However, it is necessary to have more modeling primitives than used in RDF Schema, such as data types and consistent facilities for expressing enumerations and description logic. Such possible application of RDF

as a tool for practical AI systems development has come across the rather thin set of facilities of RDF.

DAML+OIL and OWL are ontology description languages manifested as RDF Schema extensions. DAML+OIL comes from DARPA Agent Markup Language (DAML), which was a simple language with additional RDF class definitions that permitted more complicated descriptions than RDFS, and the Ontology Inference Layer (OIL) effort providing a more sophisticated classification, using concepts from frame-based AI. As a result, DAML+OIL is a language for expressing far more sophisticated classifications and properties of resources than RDFS [20].

The most recent part of the growing stack of W3C recommendations related to the Semantic Web is Web Ontology Language (OWL). OWL has been designed to meet needs for a Web Ontology Language and it incorporates lessons learned from the design and application of DAML+OIL [21].

2.3 Web Services Technology Fundamentals

In a broad meaning, web services belong to a model in which tasks within e-business processes are distributed and accessible throughout a global network. From another point of view to web services as a programming technology, web services are a stack of emerging standards that describe a service-oriented, component-based application architecture.

The term "Web service" describes a specific business functionality exposed by a company, usually through an Internet connection, for the purpose of providing a way for another company or software program to use the service.

Web services are a new kind of applications available on the Web. Web Service performs special actions ("*service*") for an external application, on which it does not depend, providing a modular way of development of distributed applications. Service descriptions are advertised on the Web and can be located by its users. Descriptions provide enough information about service invocation rules.

Web Services connect computers and devices with each other using the Internet to exchange data and combine data in new ways. Web Services can be defined as software objects that can

be assembled over the Internet using standard protocols to perform functions or execute business processes.

The main aspects of Web Services are:

- Services are developed as software components with discrete functionality;
- Services are accessible from over the Web;
- Communication (programmatic interface) with services is performed through platform-independent protocols;
- Service advertisements are published on the Web via a mediation framework;

Current technology of web services is based on the set of common standards defining aspects of services description, discovery, binding, invocation, registration, etc. This "computing stack" consists of core standards, web service basic standards and additional specification of standards that introduce new features into web service technology (Fig. 3).

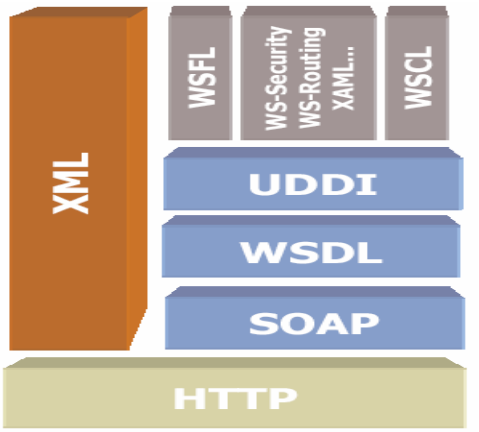


Fig. 3. Web Services computing stack (adapted from [22])

Core Layers of the Web Services Computing Stack

Common Internet Protocols

Web services rely on ubiquitous Internet connectivity and infrastructure to be nearly universally accessible. In particular, web services take advantage of HTTP and Secure HTTP, but also SMTP (Simple Mail Transfer Protocol) and FTP (File Transfer Protocol) are used.

XML (eXtensible Markup Language)

XML is a widely accepted format for data exchange and its corresponding semantics. It is a fundamental building block for almost every other layer in the web services stack. XML is a simple, very flexible text format that plays an increasingly important role in the exchange of a wide variety of data on the Web. Some XML[1] [13] benefits in brief:

- Internationalization;
- Reliability and openness;
- New possibilities for interoperability and information interchange;
- Universality in definition of platform-independent protocols;
- Human-readability

The simplicity of XML and new possibilities of the Web have made a significant impact on information interchange and development of new application architectures based on common Internet protocols. Some of the changes brought by XML are:

- Reduced dependence on proprietary data formats for applications;
- A new way to perform data exchange in e-commerce using XML;
- A movement away from tightly coupled systems such as RMI, CORBA, and DCOM to more loosely coupled frameworks;
- A new approach for service-oriented software development;
- A new basis for Web services as technology for discovering and accessing Internet-based services;
- A shift from monolithic applications to a component-based distributed software, which is a combination of components with well-defined interfaces.

XML Schema is an XML based language to express restrictions on the structure of other XML documents. Schemas provide means for defining the structure and content and allow the validation of XML documents.

SOAP (Simple Object Access Protocol)

SOAP [4] is an XML-based lightweight messaging protocol intended for exchanging structured information between applications in a decentralized, distributed environment. It is a message layout specification that defines a uniform way of XML-encoded data

[1] W3C Recommendation is available at http://www.w3.org/TR/2000/REC-xml-20001006

transmission. SOAP uses XML technologies to define an extensible messaging framework that provides a message construct, which can be exchanged over common Internet transport protocols. Unlike Microsoft's Distributed Component Object Model (DCOM) technology and Sun Microsystems's RMI for Java, which are programming model dependent, the SOAP framework has been designed to be independent of any particular programming model and other implementation specific semantics as a convention for accomplishment of Remote Procedure Calls (RPC) between heterogeneous systems.

Higher-Level Layers of the Web Services Computing Stack

WSDL (Web Services Description Language)

WSDL provides a description of connection and communication ways with a particular web service. Web Services Description Language (WSDL) [2] is an XML-based structured mechanism to describe:

- Abstract Operations that a Web Service can perform;
- Format of messages it can process;
- Protocols it can support;
- Physical bindings to communication languages and location of services.

WSDL is extensible to allow the description of endpoints and their messages independently of message formats or network protocols used for communication. WSDL defines services as collections of network endpoints or ports. In WSDL the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings. It allows the reuse of abstract definitions of messages, which are abstract descriptions of the exchange data, and port types, which are abstract collections of operations. The concrete protocol and data format specifications for a particular port type constitute a binding. A port is defined by associating a network address with a binding; a collection of ports defines a service.

A WSDL description forms a key element of the UDDI directory by means of abstraction of a service's various connection and messaging protocols.

UDDI

UDDI [3] stands for Universal Description, Discovery and Integration. Developed as a result of an industry initiative led by Microsoft, IBM and Ariba and more than 300

companies-participants, UDDI represents a set of protocols and was directed to providing a public directory (UDDI registry) for the registration and real-time lookup of web services and other business processes.

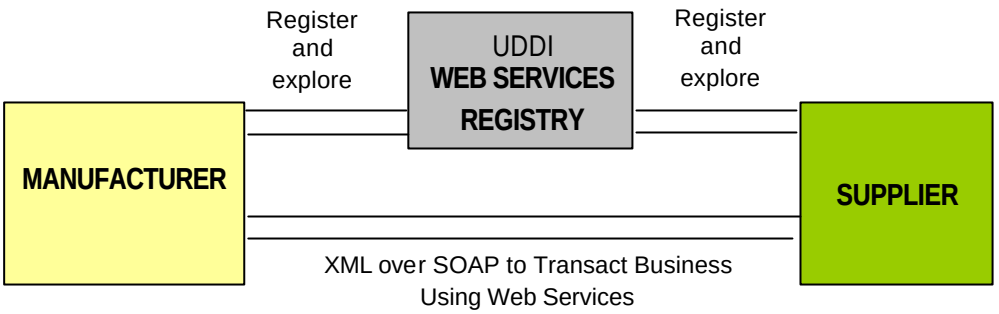


Fig. 4. How UDDI works

A UDDI registry has two kinds of clients: businesses that want to publish a service description (and its interfaces of usage), and clients who want to obtain service descriptions of a certain kind and bind programmatically to them (using SOAP messages over HTTP). UDDI itself is layered over SOAP and assumes that requests and responses are UDDI objects sent around as SOAP messages.

UDDI presents three different searchable views on registry. Searches on these views use API calls that are performed using SOAP/HTTP as it is currently implemented. Information presented about services is:

- White pages: description of the company offering the service, enables lookup for services by providers, contact details, etc.
- Yellow pages: categorization of services by industry type, enables keyword search based on standard taxonomies: NAICS (North American Industrial Classification System), SIC (Standard Industrial Classification);
- Green pages: descriptions of the interfaces to web services

A so-called “tModel” is used to describe the service *metadata* about a specification, including its name, publishing, organization, and URL pointers to the actual specifications. The company that exposes the tModel provides the reference to it for a service and this means that the company has implemented a service that is compatible with the tModel it references to. This is the way for companies to provide services compatible with the same specifications.

WSFL (Web Services Flow Language)

WSFL is the least developed layer of the current web services layers. WSFL is layered on top of WSDL to define a framework that is used to describe the business logic of web services required to assemble various services into an end-to-end business process.

Other Business Rules

Additional elements that support complex business rules must still be implemented before web services can automate truly critical business processes (security and authentication, contract management, quality of service):

WS-Security

WS-Security[1] describes enhancements to SOAP messaging to provide *quality of protection* through message integrity, message confidentiality, and single message authentication. These mechanisms can be used to accommodate a wide variety of security models and encryption technologies and also to allow the association of security tokens with messages. WS-Security is designed to be extensible and to support multiple security token formats.

WS-Routing

WS-Routing is a SOAP-based, stateless protocol for exchanging one-way SOAP messages from an initial sender to the ultimate receiver, potentially via a set of intermediaries. WS-Routing also provides an optional reverse message path enabling two-way message exchange patterns like request/response, peer-to-peer conversations, and the return of message acknowledgements and faults. WS-Routing is expressed as a SOAP header entry within a SOAP envelope making it relatively independent of the underlying protocol.

WSCL (Web Services Conversation Language)

WSCL[2] is a proposition of a simple conversation language standard that can be used for various Web-service protocols and frameworks. It focuses on modeling the sequencing of the interactions or operations of one interface. It fills the gap between mere interface

[1] IBM extension for SOAP, http://www-106.ibm.com/developerworks/webservices/library/ws-secure/

2 Specification at W3C site, http://www.w3.org/TR/wscl10/

definition languages that do not specify any choreography and more complex process or flow languages that describe complex global multi-party conversations and processes. WSCL allows the definition of abstract interfaces of Web services, e.g. the business level conversations or public processes supported by a Web service. WSCL specifies the XML exchange documents, and the allowed sequencing of these document exchanges. WSCL conversation definitions are themselves XML documents and can therefore be interpreted by Web services infrastructures and development tools. WSCL may be used in conjunction with other service description languages like WSDL.

2.4 Semantic Web enabled Web Services

The key to Web Services is dynamic service composition using independent, reusable software components. It has fundamental importance in both technical and business applications. Software products will be delivered and paid for as an easily configured set of services combined in the required software unlike packaged products. It will enable automatic, ad hoc interoperability between heterogeneous systems to accomplish complex organizational tasks [8][23].

Web Services technology nowadays is based on UDDI and WSDL which do not make any use of semantic information, hence, failing to meet the problem of matchmaking between provided capabilities of services and service requestors' needs [22][16]. This sought functionality can not be achieved just on a basis of keyword searches and vocabularies of service types.

Though the UDDI-WSDL-SOAP design only partially addresses the requirements sought by the Web Services vision, some lessons have been learnt from it. In [8] elements necessary to scalable web service discovery, mediation and composition were identified as:

- **Document types,** which describe the content of business documents.
- **Semantics**, which is introduced as semantic descriptions to be interpreted correctly by the service requestors and providers. Correct "understanding" of the descriptions requires having defined in some vocabulary a set of concepts that are common to requestors and providers (types of minimal pieces of information to be exchanged) described along with valid element values. The adoption of a common vocabulary gives a basis for communication and semantic sharing because of the same

interpretation of description elements by both sides. If vocabularies are available, then documents are described in terms from vocabularies; if ontologies are available, then document descriptions refer to the concepts declared in the ontology. Reasoning about service capabilities is possible by using tools which perform semantic check, whether the service corresponds to service requestor's needs. Generally, vocabularies have very limited support for this kind of reasoning, whereas ontologies are meant for this.

Finally, not only the semantics of message content can be annotated, but the intent of the message itself might be defined. The intentions are a very important aspect in communication between information entities of any kind (agents, humans, business processes and web services), so semantics behind a *speech act* have to be taken into account.

- **Transport binding**, which is an agreement between service requestor and service provider on the transport mechanism to be used for service requests. Several transport mechanisms are available, i.e. HTTP(S), SMTP and FTP. Each transport mechanism is associated with the representation of request and response messages and underlying communication technology.
- **Exchange sequence definition,** which is a transport-level communication protocol to follow in inherently unreliable data communication networks. The exchange sequence definition describes agreement on what kind of acknowledgment procedures and messages, time-outs and retry logic are used.
- **Communication process definition**, which is a service access protocol, a manifestation of business logic in terms of sequences of the business messages exchange.
- **Security**. Every data contained in the message from the service requestor to service provider, and vice versa, should be private and unmodified as well as non-reputable. Encryption and digital signing ensure the privacy whereby non-repudiation services ensure that neither service requestor nor service provider can claim not to have sent a message or to have sent a different one.
- **Syntax.** Documents can be represented in one of many syntaxes available. XML is the most popular syntax that fits the requirements to a general multipurpose structured data representation format.

In UDDI only Transport binding, Exchange Sequence Definition and Communication Process Definition elements' requirements are partially fulfilled via general UDDI architecture, SOAP and WSDL, and provide limited support in automated service recognition and comparison, configuration, combination and automated negotiation.

In addition to UDDI, WSDL and SOAP, there are standards such as WSFL, BPSS, XLANG, ebXML, BPML, WSCL and BPEL4WS, WS-Security and WS-Routing, which are intended to fill up other parts of the stack**.** But they are numerous, overlap each other in addressed problems, have heterogeneous data formats and have been developed by individual web-services industry players (like IBM, Microsoft, etc.) often for their own innovations. It is evident that a consistent solution cannot be achieved without the combined efforts of industrial leaders and research communities.

From the very beginning of the web services idea, many problems of the web have become apparent: human oriented technology does not provide an infrastructure for machine-readable data on the WWW. Even for a human the problem of search for required data on Internet sometimes becomes insuperable. For applications it is even worse: autonomous applications have to discover existing services and the general problem of service discovery can hardly be solved without support of the bundle of technologies that create semantic data based Web Services Infrastructure.

In order to bring Web Services on a top of performance and to make this technology flexible and adaptable for the whole variety of services that can be advertised on the web, many global problems have to be considered. Most of the important problems of Internet concerning Web Services are [2]:

1. Information exchange in the global network;
2. Web Services Infrastructure;
3. Trust, security and privacy in the distributed systems;
4. Services discovery and composition;
5. Transaction management.

The W3C's Metadata Activity[1] was tightly connected with Knowledge Management problems and has grown from the idea of having machine-understandable information in the Web.

[1] W3C Metadata Activity, http://www.w3.org/Metadata/

Metadata Activity has provided an approach for metadata labeling of web content. Further, the idea has developed into the Semantic Web vision of having data-oriented web with metadata and links between resources to provide effective discovery, integration, automation and interoperability across various semantic-aware applications.

The primary goal of Semantic Web Activity is the development of mature comprehensive standards and technologies for future Web, provision with building blocks that will assist in addressing critical issues concerning interoperability on the Web, and thus, Web Service technology. ***Ontology technology*** proposes ways to define such standards better and to map between them. It can bring communication on a higher, semantic-enabled level, provide basis for solutions of service description and integration problems. Especially efficient and significant contributions will be made for defining Web Services description framework [24].

The next-generation Web Services will transform the web from static content, human-oriented and dependent e-services to a distributed computational system, in which intelligent web services complemented by a scalable mediation infrastructure to promote substantially the performance of the Web (Fig. 5). To facilitate full the potential of Web Services, appropriate frameworks for the hottest problems of current Web are being developed [8].

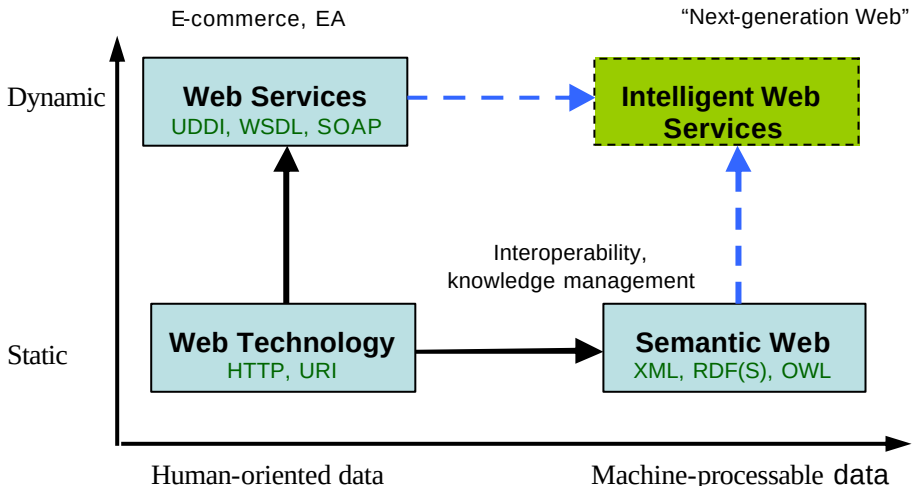


Fig. 5. Bringing Web on top of performance with Intelligent Web Services [23]

The main objectives of Semantic Web support in Web Services development are about building parts of a new stack of standards (Fig. 6):

- Provide a comprehensive Web Service ***description*** framework;
- Define a Web Service ***discovery*** framework;
- Provide a scalable solution for web services ***mediation***.

Description framework is based on an *ontological description* of services that enables an efficient *semantic-match* discovery framework within a *semantic-aware* mediation environment.

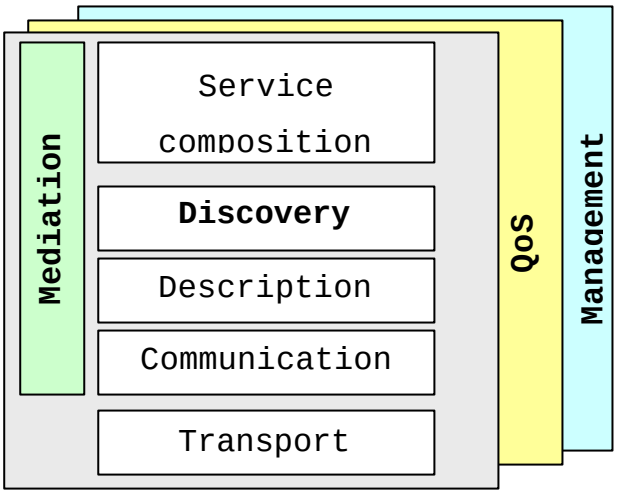


Fig. 6. The Conceptual Web Services Stack (adapted from [25])

The Semantic Web initiative provides Resource Description Framework for any kind or resources on the Web and in the real world. It is natural to use the results and approach of Semantic Web and apply them to the web technology of the future. Management of resources in Semantic Web is impossible without the use of ontologies, which can be considered as high-level metadata about semantics of Web resources and knowledge [26].

Generally, the Semantic Web will allow giving richer descriptions of Web services (e.g., semi-structured data, types, inheritance, and semantic constraints). Personalized machine agents will take over the role of a service requestor on behalf of a human user. And, they may also do the composition for the human users, other services and smart devices (see Table 1).

Table 1– Comparison between traditional and semantic-enabled web services

Dimension	**"Traditional"**	**Semantic Web**
Service	Simple	Composed
Requestor	Human	Machine
Provider	Registration	No registration
Broker	Key Player	Facilitator
Service description	Taxonomy	Ontology
Descriptive elements	Closed world	Open world
Data exchange	Syntactic-based	Semantics-based

DAML-S is a DAML+OIL upper ontology for describing properties and capabilities of Web Services. DAML-S became a point of junction of Semantic Web and Web Services. DAML-S provides a definition of a new class of resource on the Web: *Service*. Available properties and capabilities for describing Web services are introduced in DAML-S by providing an unambiguous, computer interpretable markup language, which enables automation of service use by agents and reasoning about service properties and capabilities [7].

DAML-S aims at enabling of automated Web service discovery, composition and interoperation, invocation and execution monitoring by using information provided in machine-interpretable descriptions of web services. A service profile written in DAML-S consists of three parts which are descriptions of the main aspects of the service:

1. **`ServiceProfile`**
   Contains properties of the service required for automatic discovery – about offered functionality, preconditions, inputs, outputs and effects of service invocation.
2. **`ServiceModel`**
   Description of the service's process model. Advertisement of process model enables automated integration and invocation of services.
3. **`ServiceGrounding`**
   Description of communication-level details of service - bindings to communication protocols, message descriptions etc. expressed in WSDL.

The current version of DAML-S is built on top of DAML+OIL. Next versions are likely to use OWL, since it has become a W3C standard.

2.5 Semantic Technology: New Possibilities for Industry

A semantic technology can be defined as a software technology that makes the meaningful description and associations between pieces of data known, presented in a proper syntax and processed at execution time. Some existing knowledge model used by several applications is required for a semantic technology to be applied and beneficial.

There are some differences between semantic technologies and more conventional applications:

- Semantic technologies denote conceptual description via links. The semantics of terms or concepts in the model is represented as the way they are interlinked. Conventional applications introduce no such links explicitly.
- Semantic models (ontologies) represent knowledge of the domain, in which the system operates. Several connected ontologies can be used for the representation of different viewpoints to the domain. Such models and their defined interconnections can establish links between applications that operate on them.
- Use of ontologies by applications is an essential part of their work. Access to a model represented in an ontology, sometimes called "inference on ontology" or "ontology reasoning", introduces the possibility to make the functionality of applications ontology-driven, i.e. dependent on a dynamic, developing model. Analogous use of "external model" in conventional applications has no common approach applicable to a variety of domains.

The essence of semantic technology – ontologies – can be used in within many areas of information system development and application areas, for instance, in database design, in object-oriented systems, in knowledge-based (expert) systems and data storage, knowledge representation, computer-aided collaboration and enterprise application integration.

Ontology-based solutions are central concerns for typical problems as:

- **Standardization of domain-specific models, processes, and knowledge architectures and semantic data integration.** Data from different sources is arranged according to different schemas. The need for a single point of access for all data and need for meaningful information share by applications, which are integrated into an intelligent whole, exists. Interoperable and extendable data formats for large number of parties are required.
- **Semantic-based information search and retrieval.** Search over thousands and millions (and billions) of documents can proceed using a search that matches concepts, rather than matching words.
- **Semantic "publish and subscribe".** Information is marked by content providers in a way that allows it to be searched by semantically meaningful tags.

Each of these applications makes essential use of ontologies, and each of them is properly a part of the vision of the Semantic Web.

2.5.1 Ontology-driven Applications

Attempts of defining industrial standards for information exchange have been made ever since first computer systems became available. Such standards exist on their own independently of applications and are meant for use by humans (software developers). It means that changes to standards are followed by changes of software, which are made via reimplementation with support of new aspects. In some sense, software is not *flexible enough* to follow the changing rules of the system.

If we can design software that uses a certain ontology as a configuration and functionality factor, than we can achieve the highest flexibility in complex long-term functioning systems that undergo constant changes and extensions. Generic software components that work with a wide variety of environment configurations providing context-dependent behavior are the distinctive feature of this approach.

By designing special software architecture and by providing an ontology as an essential part of software, we get an *ontology-driven software* concept. Ontologies, from this point of view, are means for higher-level programming of software systems.

This concept can be utilized for top-level modeling of heterogeneous systems integration. On that level architectural solutions are provided: system concepts are defined and bound to lower-level software solutions, relations between these system components (data flows, system process schemas, etc.) are declared in a certain meta-standard representation. For such architecture descriptions it is possible to develop a set of generic tools, which will *render desired system* as a whole unit applying those tools for mediation and configuration of system components. EAI can benefit from such approach and become less case-based, more generic technology.

Another utilization of this idea is found in *onto-adapter* software components for enabling service-like behavior of resources in the Semantic Web environment. The OntoAdapter concept is discussed later in the thesis.

2.5.2 Automated Intelligent Information Processing and Knowledge Representation

Modern challenges for information processing require involvement of knowledge-processing techniques in process control and knowledge management systems. Research activities in the Artificial Intelligence (AI) domain have created a basis for it (description logic, fuzzy logic, etc.), though having isolated approaches for knowledge representation. We can see a variety of languages designated for this. It is hard to combine them and develop intelligent applications, which can process and exchange the presented kind of data.

Semantic Web is based on application of Resource Description Framework. It does not propose any new techniques of data and knowledge processing, but supports software developers by (means of) defining a framework for dealing with it more easily. Upgraded with ontology formalism and supported by ontology languages (DAML+OIL, OWL), Semantic Web proposes a consistent, rather flexible and expressive, extendable approach for data/knowledge representation in a machine-processable form.

Industry can develop its own standards for data/knowledge representation to be used for exchange between systems. Each standard will be dependent on the application domain (in the context of maintenance, this is the standard for representation of accumulated experience how to diagnose a device state, what to do in a certain condition); in the same time there are software tools that will be reused for the development of knowledge management systems based on such representation formats.

Hence, semantic technology can be applied for:

- Unified information representation and semantic-based data search/retrieval;
- Reuse of tools for processing data formats based on meta-format (meta-standard).

2.5.3 Integration of Heterogeneous Systems

The Web Services technology has rapidly become popular in Enterprise Application Integration products, tailored distributed systems and e-Business solutions. Industrial standards such as UDDI, RosettaNet, ebXML bring solutions for service discovery, messaging and business process modeling in an open environment. These standards are supported by the largest players both in the ICT and the industrial sector.

At the same time the Web Service technology, in a certain sense, has many limitations. It has not advanced much further than enabling remote procedure calls via a commonly supported (XML, SOAP, WSDL) stack of standards. This was itself a really important step ahead, but it is not the "final destination" of web services yet. Mediation mechanisms for web services are not implemented more than registry-based keyword or identifier search (UDDI), and service composition on-the-fly [23] stays in the titles of research papers and within experimental development projects. Reasons for that are in the absence of semantic description features in the Web Service technology.

Semantics stored in service descriptions accordingly to it is pieces of text, which are meant for human use, not for software and automation. That is why service discovery can be done only if its name/identifier/code is hard-coded or entered manually. The same applies for service composition, since there is no adequate mechanism for presentation of how services require their use, which can be used for automated service composition. Software cannot assure the correctness of a found/composed service if its semantics is not available for processing.

An ontological vision of Semantic Web provides a basis for the representation of semantics. Efforts for the development of a common ontology for web services are made. DAML-S is the first strong attempt of substantial extending service description beyond the constructions provided by WSDL [2], coupled with UDDI and SOAP standards. Moreover, there is already a proposal of implementation of DAML-S extension in UDDI [27]. It can be the first step of a gradual involvement of market into semantic-enabled services, but further steps require more radical changes and building basics of Web Services with Semantic Web.

Although the adoption of a single common domain-specific standard for content representation and data exchange is highly desired, it is very difficult to achieve, because of simultaneous competitive initiatives developing their own industrial standards independently. These standards are hard to align without making them less usable by their creators. Moreover, it is often impossible to provide technical and procedural rules in advance.

One alternative is the development of common foundational ontologies, which make the basis for interoperability among information systems. This approach has the potential to significantly promote the integration, reducing the need for standardization at the technical level. It will enable service adaptation to the changing environment.

3 REFERENCE FRAMEWORK FOR ONTOLOGY-BASED INFORMATION MANAGEMENT SYSTEM

3.1 OntoServ.Net Concept

The OntoServ.*Net* concept was developed by the "Industrial Ontologies" research group[1] as an automated industrial environment for asset management, which integrates industrial services and expert knowledge and provides a conceptual and technological framework for information integration and automated process management in open distributed systems.

Industrial services are the "content" of a service network in the implementations of OntoServ.Net. People, machines and computers become interlinked in a way that all of them are available for "use" by other components of the network (Fig. 7).

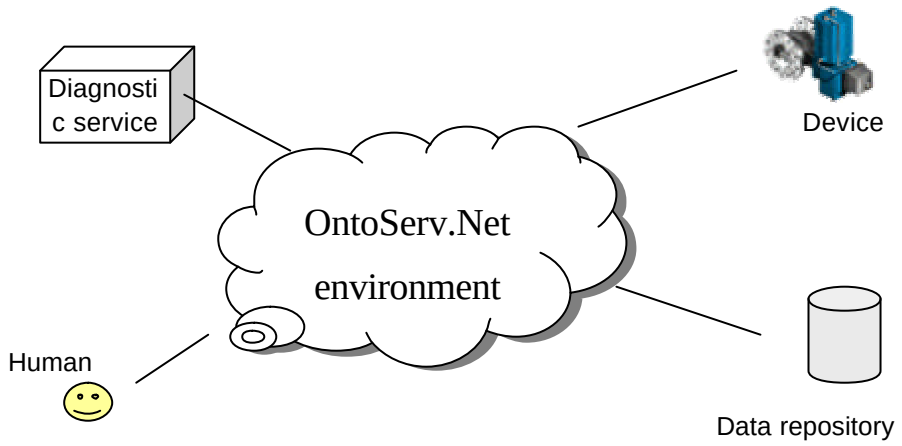


Fig. 7. Types of resources in OntoServ.Net

OntoServ.Net is an environment where federated groups of services exist as a dynamic distributed system. The most important concept within OntoServ.Net is that of service. Service represents a usable entity. Service is any kind of resources whether it is computation method, diagnostic algorithms, communication provider (transaction controller, service composer), data source (sensor data of devices, database, device configuration, etc.) or non-information object like a device or a human that controls the device.

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

OntoServ.Net's underlying technologies that enable its main desired features are Semantic Web and Web Services. Semantic Web uses ontologies to create a comprehensive environment, in which intelligent agents (software applications) can access annotated resources, communicate and perform collaborative activities. Semantic-aware web services in OntoServ.Net are main the constituent components.

OntoServ.*Net* Framework defines:

- Basic concepts and architecture vocabulary (Service Platform, OntoServ.Net Service, semantic adapter, etc.);
- Configuration rules (relations between objects, interconnections, their responsibilities and behavior);
- Information management principles and required ontology support;
- A technological infrastructure (software requirements, solutions, tools, design patterns for implementation, development process schema, etc.);
- Bindings to the domain-specific aspects of development (maintenance domain: device is a kind of service, services perform diagnostics, platforms are maintenance platforms and maintenance centers, etc.)

In a certain sense, ideas about a semantic-enabled resource presented as a service are somewhat wider than that of the semantic description of resources in Semantic Web. According to the creators of the Semantic Web idea, Semantic Web is built on top of Internet, where all resources have machine-readable metadata. Ontologies play the role of knowledge storages about specific domains and as they are referenced from resource metadata and other ontologies, they form the web of ontologies. Resource metadata is used by an intelligent software also called *intelligent agent* that is semantics-aware (supports descriptions made by using a certain ontology) and can "understand" resources based on that metadata. In this philosophy resources are passive and exposed for use by semantics-aware software (see Fig. 8a).

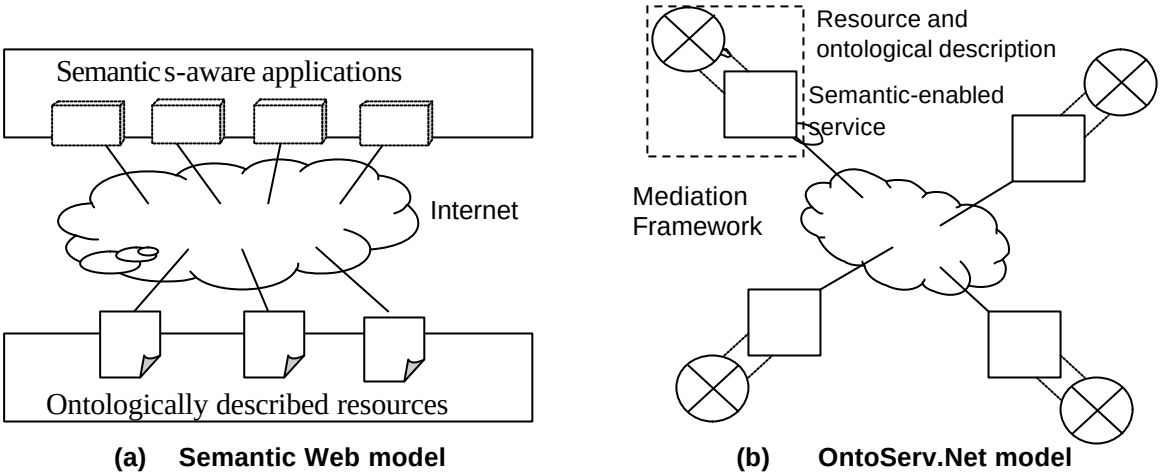


Fig. 8. New features in the OntoServ.Net model

In contrast to this schema, there are no passive resources in OntoServ.Net. Every resource is represented by a semantics-enabled service and not supposed to be semantic-enabled, i.e. initially support a specific communication and data representation model. The resource's "representative" hides from external world resource-specific features that are non-relevant for use (see Fig. 8b). On the other hand, it allows the resource to interact with other resources without knowing their transport, messaging and content representation specifics (remember that "*resource*" in this context is an entity of any kind, see above). We call this representative part *OntoAdapter*, emphasizing its semantic-oriented purpose, first of all. Adaptation is also performed on other information exchange levels. The OntoAdapter concept deserves more detailed consideration and it is described in section 4.4 of this thesis.

To summarize what was said above, resources are adapted to the OntoServ.Net environment via an onto-adapter. Together resource and onto-adapter form an OntoServ.Net service. Further in the thesis "service" is used for OntoServ.Net service unless stated otherwise.

A resource can be accessed using *semantic query* about the resource. "*Semantic*" means that it does not depend on the resource's nature and can be a presented in resource-specific form, which preserves the message semantics. "*Query*" is a communication act; in the context of service interactions, communication goes mostly in the "request-response" manner, so exchanged messages are queries and responses.

Adaptation performed by an onto-adapter can be shown in an example. If someone needs to get the value of a specific device parameter, the same semantic query will be composed using terms from a common ontology (e.g. "*parameter-type*", "*device-type*", "*query-target*", *etc.*)

whether this query is passed to a human, a database, an algorithm that calculates the value or the device itself. A service that represents a human, will ask the human about the desired value via some user interface that uses a textual description of the query, a database-representative service will make an SQL query that returns the required value, an algorithm-representative service will invoke query-specific algorithm methods, and, finally, a servicing system of the device will use device-specific communication means to get the answer. In all cases the result is returned in the same form from all queried resources.

If one application needs some data in OntoServ.Net, first it has to find its source and then make a query. Discovery is not performed by the application itself, but rather by its service-representative part, which is a part of the OntoServ.Net mediation framework. Once the appropriate resource is found, the application can make use of found resource avoiding direct interaction with it, which often is impossible due to heterogeneity of resources (Fig. 9).

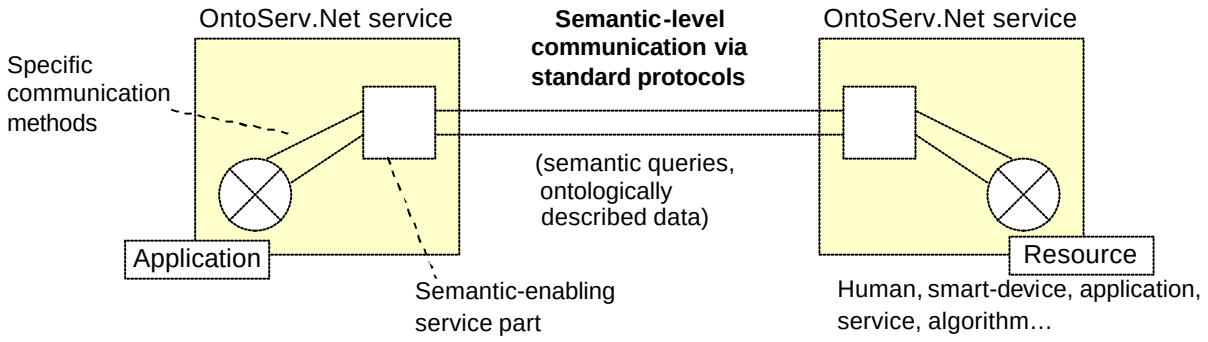


Fig. 9. Semantic-based communication between OntoServ.Net applications

3.2 Maintenance Service Network

Maintenance of complex industrial machines such as paper-machines, mills, turbines, etc. is a complicated and important task. Maintenance activities include condition monitoring, predictive maintenance, tuning, repair works. Unlike condition monitoring systems, predictive maintenance is directed to the analysis of the current device state with the objective to reveal some *possible* (not detected post facto) emerging problems and preventing failures via adjustment of parameters, change of parts, tuning, etc. beforehand. It leads to lower expenses for device maintenance (because failures can damage devices very hard sometimes). Advanced data mining and machine learning techniques are used for the prediction of faults. In order to recognize some dimensions of the device state and derive useful patterns from this information, which can be considered as “symptoms” of the device “health” (more strict terms

for that are “alarms” and “condition”), both batch learning and online learning techniques use historical data within predictive maintenance activities.

Usually a Web Service is expected to be accessed by human users or by software agents or applications on behalf of human users. However there already a new growing group of Web Service “users”, which is smart industrial devices, robots or any other objects created by an industry and equipped by an “embedded intelligence” or by a remote control system. There is a good reason to launch special Web Services for such industrial devices. Such services will provide necessary online information provisioning for the smart devices, allow the heterogeneous devices or their control systems to communicate and exchange data and knowledge with each other and even support co-operation between different devices.

*Maintenance services network (*industrial application of OntoServ.Net Framework*)* links consumers and providers of maintenance services in one network providing better condition monitoring support of industrial devices. Members of this service network can share their maintenance methods and information developed during the operation of machines (devices, equipment, installations). Improved locally, maintenance experience can be shared. OntoServ.Net is built as a dynamic network of *web services.* Industrial machines are *serviced* by the resources of OntoServ.Net, which are semantic-enabled services (Fig. 10).

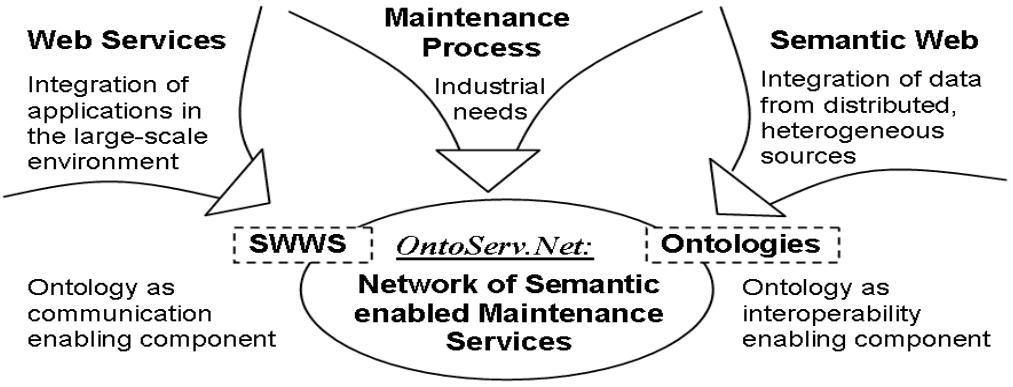


Fig. 10. Maintenance Services Network for Smart-Devices

In general, the OntoServ.Net framework is considered here for asset maintenance, but it is possible to apply its basics for process control, improvement of operating efficiency, field-performance diagnostics, plant-level management, etc., as well.

In addition, 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 devices

is extended to an automatic diagnostics and recovery within a network of maintenance centers. The role of a 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.

Since maintenance-related processes rely on relevant information, comprehensive and timely information delivery to the individuals involved in the maintenance can significantly benefit the process. This makes an automated maintenance system, which can integrate maintenance-related information from many sources, highly desired in order to give appropriate maintenance support. The typical lifecycle of maintenance activities is shown in Fig. 11.

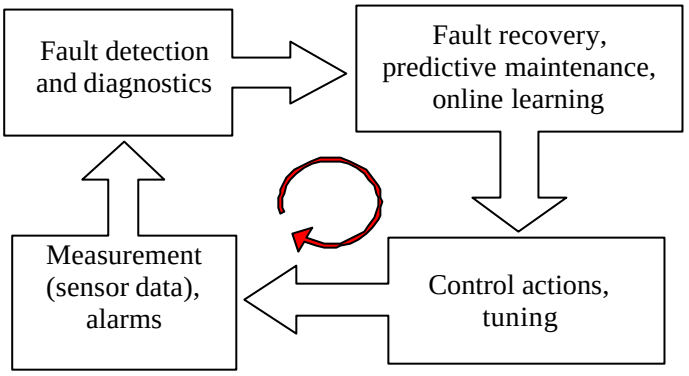


Fig. 11. Lifecycle of maintenance activities

Growing interest to machines with embedded intelligent maintenance capabilities leads to a special kind of industrial products – smart-devices (machines). The expectations from smart devices include advanced diagnostics and predictive maintenance capabilities. The concerns in this area are to develop a diagnostics system that automatically follows up the performance and maintenance needs of field devices offering also easy access to this information.

The Field Agent concept is used for a software component that automatically follows the condition of field devices. A field agent component is autonomous and communicates with its environment and other field agents; it is capable of learning new things and delivering new information to other field agents. It delivers reports and alarms to the user by means of existing and well-known technologies such as intranet and e-mail messages.

Easy on-line access to the knowledge describing field device performance and maintenance needs is crucial. There is also growing need to provide automatic access to this knowledge not

only to humans but also to other devices, applications, expert systems, agents etc., which can use this knowledge for different purposes of further device diagnostics and maintenance. Also the reuse of collected and shared knowledge is important for other field agents to manage maintenance in similar cases.

Appropriate field agents should communicate with each other (e.g. in a peer-to-peer manner) to share locally stored online and historical information, thus, improving the performance of the diagnostic algorithms, allowing even the co-operative use of heterogeneous field devices produced by different companies, which share common communication standards and ontologies. Maintenance centers supported by machine manufacturers or by some other parties will provide entry points to a maintenance network and play the role of mediator of the maintenance networking (see Fig. 12). Communication between nodes in the maintenance network is to be built as web services communication. Maintenance centers mediate such communication providing service discovery capabilities and provide their own services that can compose web services to deliver complex ones to an embedded maintenance platform of smart-devices.

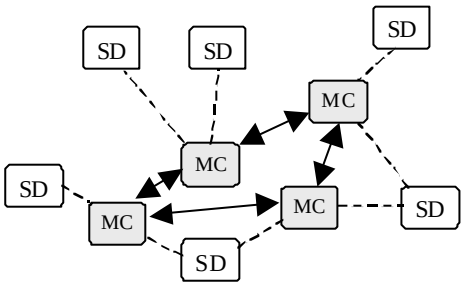


Fig. 12. Smart-devices and maintenance centers in OntoServ.Net

While monitoring a field device via one information channel, one can get useful information about some dimensions of the device state, and then derive online some useful patterns from this information. Observed patterns can be considered as “symptoms” of the changing device condition, and finally recognize these symptoms using "Ontology of Patterns" and get further into the classification of the device state using available diagnostic services.

In any case, history data, derived patterns and diagnoses can be stored and used locally. However, there should be a possibility to access this information easily and to share it with other agents for reuse purposes. There are at least two cases when such distributed infrastructure is reasonable. The first one is when we are monitoring a group of distributed devices, which are physically and logically disjoint, however they all are of the same type. In

this case any history of derived patterns and diagnoses from one device can be useful to better interpret the current state of any other device from the group. The second case relates to the monitoring of a group of distributed devices of a different type, which are considered as a system of physically or logically interacting components. In such case it would be extremely important for every field agent to use information from other field agents as additional data for the interpretation of the device state. Thus, agents should communicate with each other (e.g. in a peer-to-peer manner) in order to share locally stored online and historical information and, hence, to improve the performance of the diagnostic algorithms, allowing even the co-operative use of heterogeneous field devices produced by different companies, which share common communication standards and ontologies.

4 ONTOLOGY SUPPORT FOR *ONTOSERV.NET*

4.1 Scope for Ontology Support in OntoServ.Net

The goal of OntoServ.Net is to provide interoperability of heterogeneous services. It is natural to assume that the newest Semantic Web and Intelligent Web Services concepts can be applied to the problems of interoperability among field devices and will result to an essential improvement of field device performance.

The use of Semantic Web approach here provides basic means for dealing with:

- semantic heterogeneity of peers (initially incompatible components);
- semantic-based information retrieval;
- service description / search of services;
- need for service composition;

As it will be shown below, an ontological vision of Semantic Web can propose some solutions for these issues.

Ontological descriptions in OntoServ.*Net* play the role of enabling technology that will provide efficient service discovery and automated services use in such environment. DAML-S will be used for web services descriptions. RDF serialization of data is to be used. Most of interactions are done in form of *semantic queries*, so an appropriate communication ontology is required for exchanging such queries and other communication messages.

In order to provide interoperability in information exchange between nodes in OntoServ.*Net*, passed data has to be annotated using an ontology that is common to all nodes this data will be delivered to. Since virtually any part of the embedded maintenance platform can use network resources (access maintenance web services and provide its own services), it is required to have data annotated immediately after its creation and process it with semantic-aware applications in the embedded platform.

4.1.1 Standardized Data Exchange and Retrieval

In any type of communication, the information sharing is often problematic because the meaning of information is affected by the context for viewing it and interpretation. This is true for industrial systems because of the increasing complexity of information and growing need for information exchange among heterogeneous (not previously designed to be

compatible) software applications. Different applications may use a different representation for exactly the same concept or use the same term for quite different concepts. And very often descriptions given in a natural language, which are linked with the terms, are ambiguous, do not provide enough information and do not allow resolving the differences between similar, but not the same, concepts.

The ontological definition of maintenance concepts and terms along with their formal and unambiguous definitions is a basic solution for information exchange between maintenance-related systems, services, data storages and front-end (user interface) applications. Ontology engineering issues are the most central in ontology support for Maintenance Services Network.

In fact, a common ontology defines the basic vocabulary with which queries and assertions are exchanged among the services of a maintenance network. Ontological commitment is an agreement to use the shared vocabulary in a coherent and consistent manner [18]. If data is to be transferred, it has to be presented as an RDF Object – small manageable chunks of data with semantics attached accordingly to Resource Description Framework and used vocabulary [28].

While mostly all above-said concerns message content, both content language and message syntax need consideration in semantic-enabled communication. Standardized method for accessing Web Service technology is declared in the SOAP specification. A SOAP message consists of the SOAP envelope 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 and a convention that can be used to represent remote procedure calls and responses.

The SOAP standard matches perfectly the initial idea of exchange instances of application-defined data types in a heterogeneous distributed environment ("RPC over web"), but there are some limitations of SOAP to be a base standard of messaging framework for Semantic Web enabled Web Service technology:

- SOAP message formats are provided as a part of higher level standards, e.g. WSDL; communication requires an a-priori agreement between Web Services on message format and protocol;

- SOAP standard has no communicative speech acts: there is no way to determine the intention of the message sender or what the message is trying to achieve (the semantic of the message is not introduced explicitly).

From the point of view of the Semantic Web enabled Web Services approach, SOAP (as is) is not suitable as container mechanism for semantic-aware mediation since it, first, has no semantic and, second, scores low on possibility to be used in a situation where no a priori message format is defined.

It is possible to use RDF payload in SOAP (as a first step from SOAP to RDF messaging) or even a SOAP-less pure-RDF messaging system. Corresponding ontology support and mediation framework are required.

RDF can be chosen as a messaging language for Web Services because:

- it is not structure-oriented as SOAP, but semantic-oriented; there is a resource description model behind the RDF which binds assertions (RDF statements) in the message to ontology and there is XML Schema behind SOAP which only restricts XML structure of the message;
- it supports knowledge representation for service description and any other asserts (e.g. about preferences, security etc.), allowing inference on such information;
- it will be widely used for resources description and developed tools will be reused for web service if an appropriate web service ontology exists;
- RDF and ontologies in Semantic Web are going to be a universal semantic description framework and their adoption will be a crucial point in the future knowledge management technologies, so accepting it in advance is reasonable.

Summarizing this section: in order to achieve a higher level of interoperability in information exchange, communicating applications have to:

1. Have commitment to at least one common ontology on the subject of communication;
2. Have the ontology defined in one of ontology languages (DAML, OWL) or *at least* RDFS vocabulary;
3. Provide data (content) as RDF Objects in one of RDF serialization formats (usually RDF-in-XML, but also N3, or triplets, notation is acceptable);
4. Use extended SOAP (via defined extension) or pure-RDF representation of messages.

4.1.2 Service Description

The basic requirements to a service description language in [29] are formulated as:

Requirement 1: High degree of flexibility and expressiveness

Requirement 2: Ability to express semi-structured data

Requirement 3: Support for types and subsumption (categorization)

Requirement 4: Ability to express constraints

Considering these requirements and comparing ontological descriptions (written in DAML-S) proposed by Semantic Web, it is clear that they satisfy these requirements: the RDF representation layer conforms with **Req. 1** and **2**, the RDF Schema layer complies with **Req. 3** and refines **Req. 1**, whereas the DAML layer can meet **Req. 4**.

Service description has to be complex enough to correspond to its purpose. Main purposes of using service profiles (descriptions) in OntoServ.Net are:

1. to provide a general semantic description of a service: what type of service it is (e.g. «*ClimateReportingService*»[1], restriction: "*GeographicalLocation*" is "*Country*"= *Finland*);
2. to specify semantics and restrictions for accepted data as input parameters and semantics of provided results (e.g. input data is the string-name of "*City*" and the result is an instance of class "*Climate Parameter*", concrete class "*Air Temperature*", with parameter "*Scale*"= *Fahrenheit*);
3. to provide a messaging model of service: sequences of messages, which can occur during typical scenarios of service use, and also the failure/exception state behavior of the service (e.g. service supports simple request-response messaging: for query with required input data it provides a response message, also "*UnknownParameter*"=*City*, "*DataUnavailable*" and "*UnexpectedError*" messages can be returned in exceptional cases);
4. to specify the interface of the service on the networking level: bindings to specific transport protocols, ports, addresses, etc. (e.g. "connect via raw TCP/IP connection on **ws.iog.com:1024**" or "SOAP over HTTP, version 1.2 using WSDL descriptor from address… ", etc.[2])

[1] Concept names are italicized and given in these examples in quotes

[2] Description is given as plain text here only for simplicity. Actually this part of service profile is for use with extendable WSDL-based descriptions.

The division of the service profile into these parts is made absolutely for structuring it. None of them is enough alone for using the service.

The most important role of Semantic Web in the given list of service profile parts is in items 1 and 2, where *semantics* needs to be introduced for service discovery and service composition. Messaging model and protocol binding description structures seem to be well defined in WSDL, so reuse of WSDL is highly desired, because of support from leading software developers.

Whereas DAML-S can be used for general profile structures, basic and specific domain-oriented vocabularies (ontologies) are required in order to provide agreed semantics for terms used in the semantic description part of the service profile. For the maintenance service network an ontology regarding types of diagnostics, device descriptions, and maintenance actions will need to be developed. Thus, the development of a consistent service description ontology and domain-specific ontologies will be a substantial part of ontology support of semantic-enabled services.

4.1.3 Resource Discovery

Where, what and how to find in OntoServ.Net? It depends where. 'What' are, certainly, services. It is not trivial 'how' and depends on 'where'.

The problem of service discovery and the importance of corresponding "convenient" and expressive means for service advertisement have to be explained very clearly.

If a known business partner has a known maintenance service gateway to required resources (services), then there is nothing left to discover. But such assumption (that all of the information is already known) is not more then just desired thing in real world. If you want to find out which business partners have which services, the ability to discover the answers can quickly become difficult. It is even more difficult to choose among a variety of web services the one, which you really need.

The option to call every place on the phone and then try to find the right person to talk with is not appropriate. For an e-business that exposes Web services, most likely, there is no staff to satisfy random discovery demands. Moreover, the idea of having web services implies the absence of humans in service discovery. So the only possibility is to use an appropriate description framework and discovery mechanism that uses it efficiently and on its full

potential. Whether it is in a centralized, partially-centralized or in distributed mediation framework which exists in a particular web service solution, service advertisement are located in web service registry/registries or on service platforms of a P2P-like service network and *matchmaking* is performed, which is a process of discovering an advertisement that best matches a request for a particular service.

Advanced matchmaking services require rich and flexible metadata independently of the mediation architecture. The Semantic Web initiative evolves RDF and ontology languages as tools that might help fill the gap between the current "traditional" solutions and the requirement for advanced matchmaking.

Current service retrieval approaches have serious limitations with respect to meeting the challenges described above. They either perform relatively poorly or make unrealistic demands of those who wish to index or retrieve services. The information initially has focused on the retrieval of documents, not services per se, and has as a result emphasized keyword-based approaches. The software agents and distributed computing communities have developed simple 'frame-based' approaches for 'matchmaking' between tasks and on-line services (see Fig. 13).

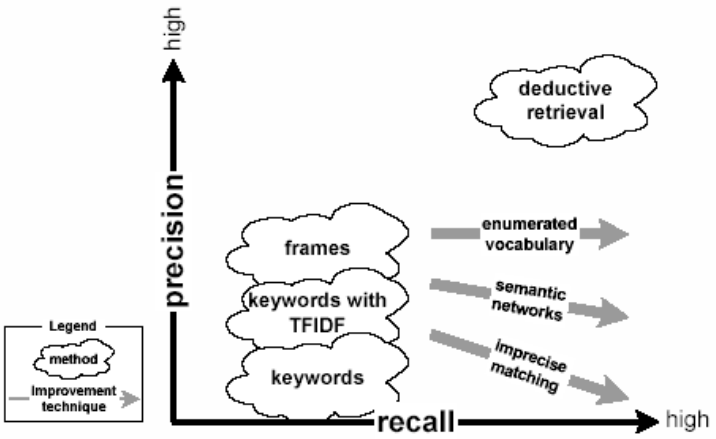


Fig. 13. State of the art service discovery (adapted from [30])

Recall is the extent to which a search engine retrieves *all* of the items that one is interested in (i.e. avoiding false negatives) while *precision* is the extent to which the tool retrieves *only* the items that one is interested in (i.e. avoiding false positives).

Semantic Web needs flexible and featured tools for querying a knowledge representation model contained in an ontology and doing reasoning on it as well as querying the RDF-based content of semantic annotations. Querying tools for RDF are being designed and the first

implementations are already available. These RDF Query Languages (RQL, RDQL, SeRQL) are supported by query engines that run as standalone servers over RDF repositories [31].

It is obvious that such query engines will be used for RDF data processing as part of any service component or network node in OntoServ.Net (which is a service platform). Service discovery on a local platform is made by querying its repository where all service descriptions are stored. An advanced matchmaking procedure has to be developed. It finds out which available services *match* the semantics of the requested service.

If required service cannot be found on local platform (e.g. the maintenance system has requested special service for detailed diagnostics of a rare device state) network search is performed. The search algorithm depends on network topology and existence of mediator-services. Search in a centralized network is not the same as search in a peer-to-peer network, etc. See more details about possible network architectures in section 5 – "Related work: web service system infrastructures"

The maintenance service network discussed in this thesis is build as a peer-to-peer network without a priori defined service-search services. This means that every service platform accepts external search queries, performs local search and supports further query distribution in the service network. Different strategies can be applied here. They are out of the scope of this thesis and discussed in the related work [9].

Ontology support for service discovery is an extension of service description ontology DAML-S with architecture description elements, which are specific to OntoServ.Net and required for the search algorithm.

4.1.4 Service Composition

The composition of web services that have been previously annotated with semantics and discovered by a mediation platform is another benefit proposed by Semantic Web for Web Services.

The composition of services can be a quite simple sequence of service calls passing outputs of one service to the next or it can be a much more complex, where *execution path* (service workflow) is not a sequence but a more sophisticated structure, or intermediate data transformation is required to join outputs of one service with inputs of another. Within the traditional approach such service composition can be created but with limitations: since the

semantics of inputs/outputs is not introduced explicitly, the only way to find a matching service is to follow data types of its inputs and/or know exactly what service is required. This approach works for simple composition problems but fails for problems required for the future Web Services for e-commerce.

In DAML-S the *ServiceGrounding* part of service description provides knowledge required to access a service (where, what data, in what sequence the communication goes) and the *ServiceProfile* part provides the *meaning* the service is used for. These two pieces of information are enough (supposed by the Semantic Web vision) to be used by an intelligent mediator (intelligent agent, mediation platform, transaction manager etc.) for using this service directly or as a part of a compound service.

As an example of composition, suppose there are a device producing a sequence of sensor data and two services, a monitoring service that calculates statistic data and a diagnostic service, where the first one calculates an estimation of the value from several observations and the second one accepts the estimation with a certain confidence level and returns the result of the classification of the device state into several possible classes. Considering a distributed service network it is unlikely that all services of the required type have exactly the same inputs, even though they perform the same diagnostic functions. If we need to use a diagnostic service, we have to supply it with all requested input parameters. It can be done by composing these two services. This is an example where parameter adaptation is performed.

Additional helper-services in this case perform the adaptation of existing data to target service. Helper-services can preprocess (convert, generalize, combine, etc.) output data from other services or from other data sources. The use of helper-services will make service requestors less dependent on the target service interface.

Another type of service composition is more promising for future service mediation. The principle is not to adapt an existing service with desired semantics, but to create a new service from several services that make partial operations. Complex services can be constructed in this way and presented as a "normal" monolithic unit for service consumers.

The construction of "on-the-fly services" requires appropriate model description framework (and an ontological description is obviously to be used for the same reasons as a service description). The knowledge base of a service-composing entity, which combines some other services and provides a new service as its own, contains service construction patterns written

using a service model description ontology. One of DAML-S's aims is to provide such description framework.

The composition algorithm itself can be a usual service or a part of service platform functional duties. The implementation of a service composer in [32] has shown how to use semantic descriptions to aid in the composition of web services - it directly combines the DAML-S semantic service descriptions with actual invocations of the WSDL descriptions allowing to execute the composed services on the Web. The prototype system can compose the actual web services deployed on the internet as well as providing filtering capabilities where a large number of similar services may be available.

4.2 Ontology Set

The OntoServ.Net Ontology provides a common language for the communication and cooperation between participants of the service network.

The ontology describes:

- semantics of the basic terms;
- relations between them;
- concepts for standardization of messaging (transport description, message structure and content representation);
- representation of service semantics;
- service invocation rules and specific information required for the inclusion of the service into the maintenance network.

The ontology ensures the compatibility of heterogeneous participants of the maintenance network, maintenance information sharing, service discovery and composition.

The concept of *OntoServ.Net Service* is central, since every participant of the maintenance network is positioned as a potential service and has an appropriate description. *Service* can be any entity, which has an appropriate (accordingly to the ontology) representation, can be accessed (invoked, queried, notified, signaled, etc.) and can access (invoke, query... etc.) other services via the provided messaging framework. A core upper-ontology for service descriptions and message structure exists as a standard in OntoServ.Net. Upper-ontologies are meant to remain unchanged at least for a separate implementation version.

The development of *Upper Maintenance Ontology* is a part of ontology support for the OntoServ.Net-based *maintenance service network*. The main purpose of this ontology is to manifest itself as an agreed standard for the representation of device descriptions: device configuration, capabilities and state. This ontology defines domain specific concepts of industrial maintenance and refines general concepts of *device*, *(device) state*, *diagnosis*, *maintenance activity*, etc. *Maintenance services* and *devices* are described with maintenance specific data: parameters, methods for dealing with it, requirements, constraints, etc.

A generalized view of the ontology infrastructure required for the maintenance service network is presented in Fig. 14. A detailed plan is shown in Appendix B, Fig. B-2.

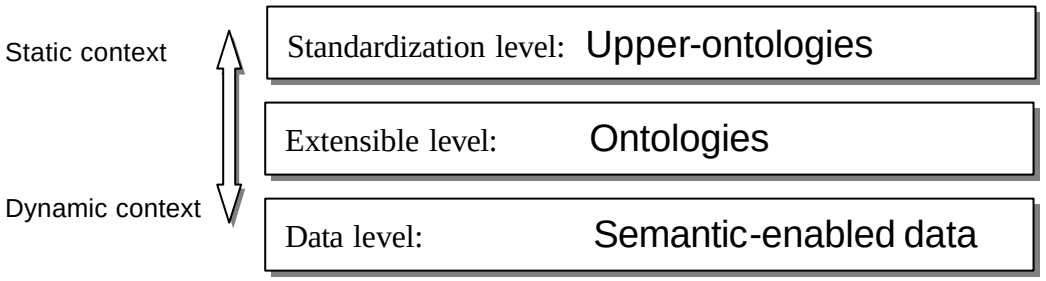


Fig. 14. Ontology-web infrastructure for OntoServ.Net

The taxonomy of maintenance services presented as a separate unit is a dynamic knowledge base of the maintenance service network. Diagnostics ontology provides the taxonomy of device states along with rules. Diagnostic terminology and diagnostic methods are a substantial part of the maintenance ontology. The ontology provides a common language for *device state* description, *device state classification* terms and the expert knowledge representation framework.

DAML-S upper-ontology for service description is the basic part of service description means required for OntoServ.Net. For the maintenance service network, only a taxonomy of services has to be built.

Services described using DAML-S present enough information to determine their semantics, parameter semantics, internal model (if any exposed) and messaging interface (via reuse of WSDL with semantic bindings to operations).

A specific ontology of maintenance is required for the support of data acquired from maintained devices, diagnostic knowledge representation and device interface description.

These sub-ontologies become interlinked in the main maintenance process cycle, where data is read from a device, analyzed as a part of condition monitoring, and then an appropriate mechanism performs maintenance for the arisen problem, when procedures of preliminary diagnostics determine a need for active interference to device exploitation.

Considering heterogeneous distributed services that may be involved in device maintenance, common device state representation is used for preliminary and in-depth diagnostics of a device. This Device Description Language (DDL) is standardized by "Device Description Ontology" (DD-ontology). Software, which "understands" DDL data (i.e. developed to support the terms of DD-ontology), processes device state data independently of device-specific data acquisition mechanisms.

Existing software that does not support DDL can be connected via an onto-adapter similarly to a device. Onto-adapter is configured for translating DDL data into the desired format. In order to make this possible for virtually any kind of existing services (or ones developed in the future), DD-ontology, surely, has to be "more general" in a sense that any other device description formats can be mapped as subsets of DDL.

DD-ontology undoubtedly will be extended with new constructs for describing new specific device features. Moreover device taxonomy as a part of the ontology will grow constantly. Ontology changes allow the software to work with newer-format data. The key here is that the ontology only grows and parts of it are never "cut away". In the world of Semantic Web data, a service will accept only known concepts and ignore others. If such ignorance is not acceptable by the data source, than the service is just out of use and it will not be used for this data. But it can be in use for other data sources.

The common semantic-enabled data-flow schema of a typical maintenance system is presented in Fig. 15. Following it, sensor data from a maintained resource (device) is transformed into a semantic annotation of its condition and further used in that form on all stages of processing, storing and analysis, by other (potentially distributed and heterogeneous) maintenance services and humans, who also are considered as services in the OntoServ.Net.

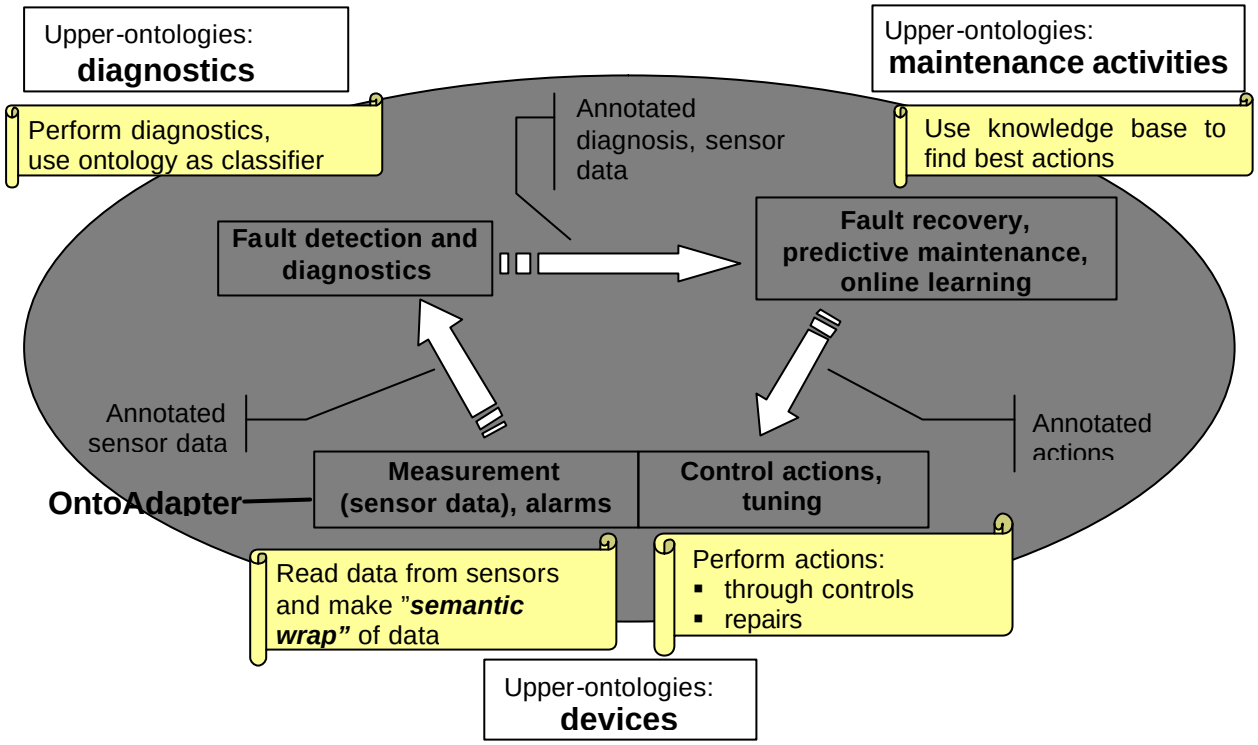


Fig. 15. Ontologically annotated data in maintenance process

4.3 Ontology Management

The use of ontologies is seen as the best solution not only to solve these particular problems, but also to provide a common knowledge infrastructure for other automation applications like process automation, computer aided engineering etc. Most of such applications will be knowledge-enabled and use ontologies to drive their services.

An extensive requirement gathering process has to be undertaken to compile requirements for ontology management solutions. Key requirements for ontology management are identified [33] as:

Scalability, Availability, Reliability and Performance

These requirements are considered the most essential for the industrial wotld, both during the ontology development, the maintenance phase and the deployment phase of the ontology. The ontology management solutions have to allow a distributed development of large-scale ontologies concurrently and collaboratively by multiple

users while maintaining a high level of reliability and performance. For the deployment, this requirement is considered to be even more important. Software accessing ontological data needs to be reliable and fast.

Distributed Multi-User Collaboration

Collaboration is seen as a key to knowledge sharing and building. Ontologists, domain experts, and business analysts need a tool that allows them to work collaboratively to create and maintain ontologies even if they work in different geographic locations.

Ease of Use

The ontology development and maintenance process must to be simple, and the tools must be usable by ontologists as well as domain experts.

Extensible and Flexible Knowledge Representation

The knowledge model should incorporate the best knowledge representation practices available in the industry and be flexible and extensible enough to easily incorporate new representational features and incorporate and interoperate with different knowledge models such as RDF(S) and DAML+OIL.

Standardized interfaces for application

For supporting interoperability and sharing information between applications, the ontology solution must provide standardized interfaces to enable interaction and interoperability with other applications.

Internationalization

Applications using ontological data have to serve users around the world. The ontology management solution should allow users to create ontologies in different languages and support the display or retrieval of ontologies using different locales based on the user's geographical location. We can assume, of course, that one "official" language will be chosen globally, but the internalization idea seems to be attractive to the end-users of the maintenance system.

These requirements are considered to be the most important for an industrial ontology management solution.

Over the years researchers and practitioners in the database, information systems and internet fields have made significant progress towards the building of solutions that involve such

systems for a wide range of application domains. In doing this, solutions necessarily concentrated more on syntax as the readily available unifying formalism for representation and structure, than on the broad variety of semantics involved. One of the recent unifying visions is that of Semantic Web, which proposed semantic annotation of data, so that programs can understand it, and help in making decisions. Researchers have subsequently seen the value of using semantics to understand information and decision-making needs of humans, so that data and humans' needs can be semantically intermediated. The scope of semantics-based solutions has also moved from data and information to services and processes.

Ontology management issues for the OntoServ.Net environment are concentrated around the problem: Who creates the system initially? Who controls ontology evolution when needs for it arise? What is the technical structure for this?

Obviously, only large industry companies can organize a service network for their internal use and create services for their own branches, plants, etc. This kind of company can develop an ontology accordingly to its own needs and control the evolvement of the service network. This is one of possible scenarios. Another one is the creation of an industrial companies association, the task of which is the centralized control of the service network evolution. Its role is similar to that of W3C (World Wide Web Committee) for Internet. Via this administrative organ developers can coordinate their efforts and perform versioning of base upper-ontologies.

4.4 Semantic Adapters for OntoServ.Net Resources

Along with new specially designed software for OntoServ.Net-compatible systems, a lot of already existing systems need to be included. Special attention is required for the issue of treating non-digital, non-standard or semantic-unaware entities, data sources or objects of the real world (human, expert knowledge, device, legacy system, semantic-unaware systems and software) as OntoServ.Net Services.

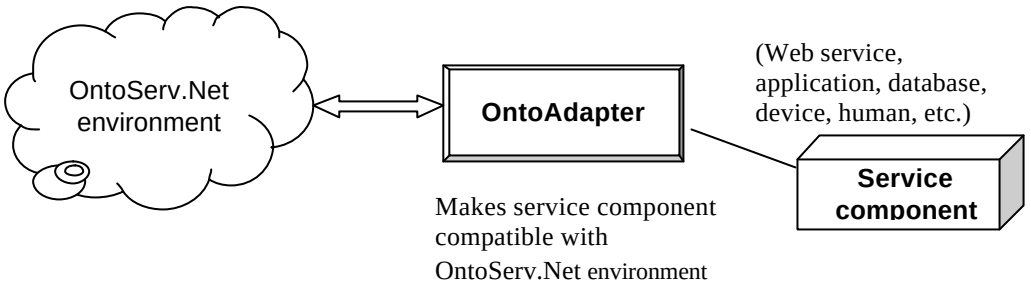


Fig. 16. Concept of OntoAdapter

Adapter technology that enables their participation in the maintenance network is provided as a part of OntoServ.Net. The concept of *OntoAdapter* is an integral part of OntoServ.Net. OntoAdapter interacts with the OntoServ.Net environment on behalf of *Object* (resource, service, device, etc.), performing mediation functions and integrating it into the environment on several levels:

- Semantics (represents the object's data semantically annotated, ensures its use respectively to its semantics);
- Messaging (uses standard message formats and message patterns regardless of the object's own communication manner, translates external messages according to the format assumed by the object);
- Networking (Transporting) (establishes a [physical] connection with the object, provides a base for communication).

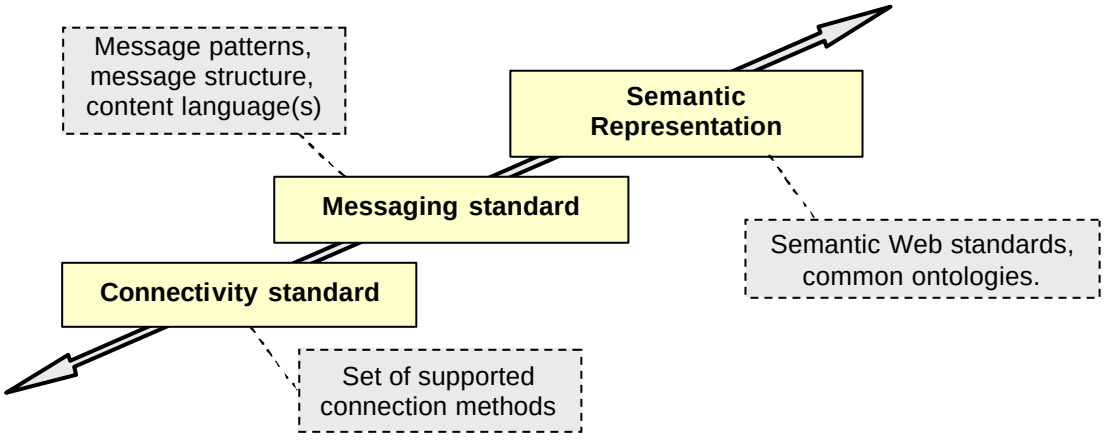


Fig. 17. Levels of ontology support for OntoAdapter software

A maintenance service network is device-centric. Its purpose is to monitor the state of the devices plugged into it. In order to make device data available to device-independent services (diagnostic components, user interfacing software, data storage and retrieval framework), certain efforts are required to construct an adapter to the device.

The ultimate goal is to have a generic adapter software that can be configured for any specific device. The configuration data schema is to be defined ontologically, where the device interface on the communication level is presented in *Ontology of Devices* and the service aspects of the device (interaction patterns, invocation rules) are presented using DAML-S ontology.

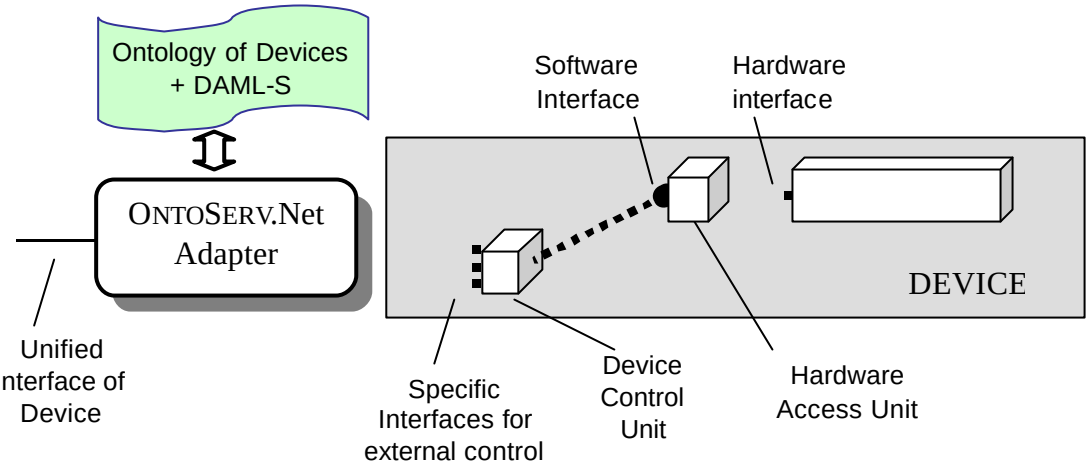


Fig. 18. Onto-adapter makes a device a semantic-enabled resource

Similar to the device adapter are adapters for other sorts of service components in the OntoServ.Net environment. User-agent services, data-access services and "real" services wrapped with onto-adapter are the same from the point of view of defining their capabilities, message patterns, invocation rules, etc., so the same onto-adapter software is used for each of them with a specific DAML-S description.

The layered structure of onto-adapter allows changing protocol and connectivity-enabling modules according to service the component description (Fig. 19).

Standard connectivity between onto-adapter components and the OntoServ.Net environment is implementation dependent. Various options are suited for this, any transport-level solution is acceptable (a TCP/IP connection is the most probable since Internet is assumed as a backbone for distributed service network).

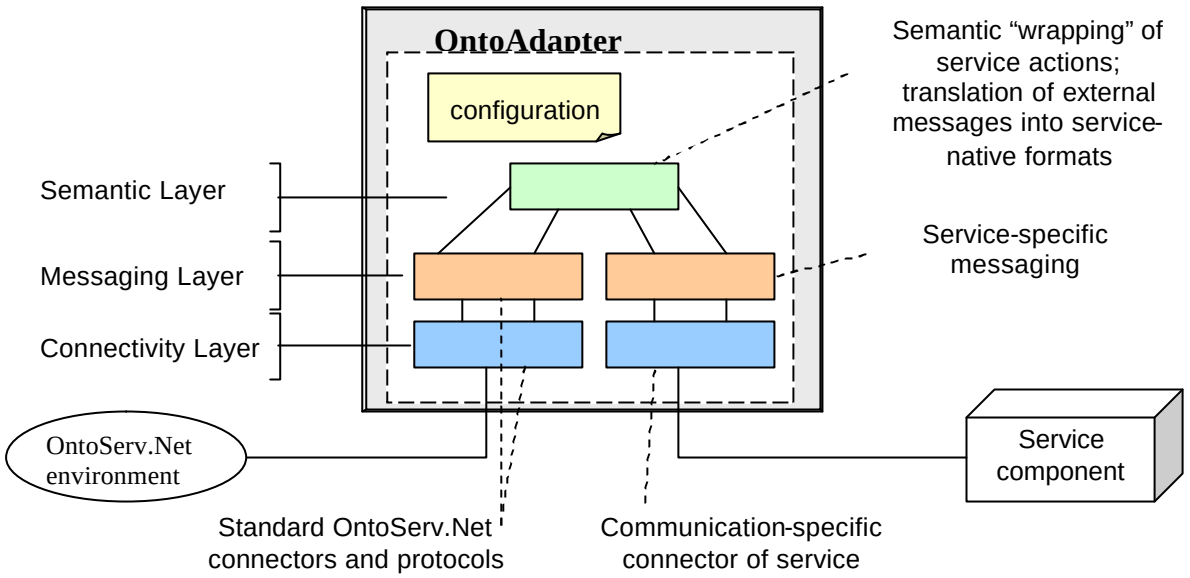


Fig. 19. Internal structure of onto-adapter

A standard messaging mechanism may also be built over already existing common internet protocols, such as Email, HTTP, FTP, etc. The message container (header, descriptor), as proposed, is presented as RDF data using vocabulary defined in the core OntoServ.Net ontology.

Finally, the content language used by the adapter components is totally OntoServ.Net-specific. It conforms to the vocabulary and schema defined in the core and domain ontologies. In the case of maintenance service network, the contents of exchanged messages refer their semantics to definitions in the Maintenance Ontology.

It is supposed that a set of service-connectors for base classes of services (SOAP-connector, RMI-connector, CORBA-connector) will be developed. Other service-specific messaging and communication parts of onto-adapter (e.g. COM port connector, HART, etc.) will have to be developed by third-parties using Software Development Kit (OntoAdapter-SDK).

Actually, the configuration of the service-specific part in onto-adapter is quite relative. The exact configuration is defined by the service component description and may both simplify and complicate it if appropriate.

The applications of semantic adapters (wrappers) are wide and very promising for many domains. The industrial maintenance domain provides a possibility for a demonstration of their features most sought: generic component-based structure and applicability for variety of resource types (Fig. 20).

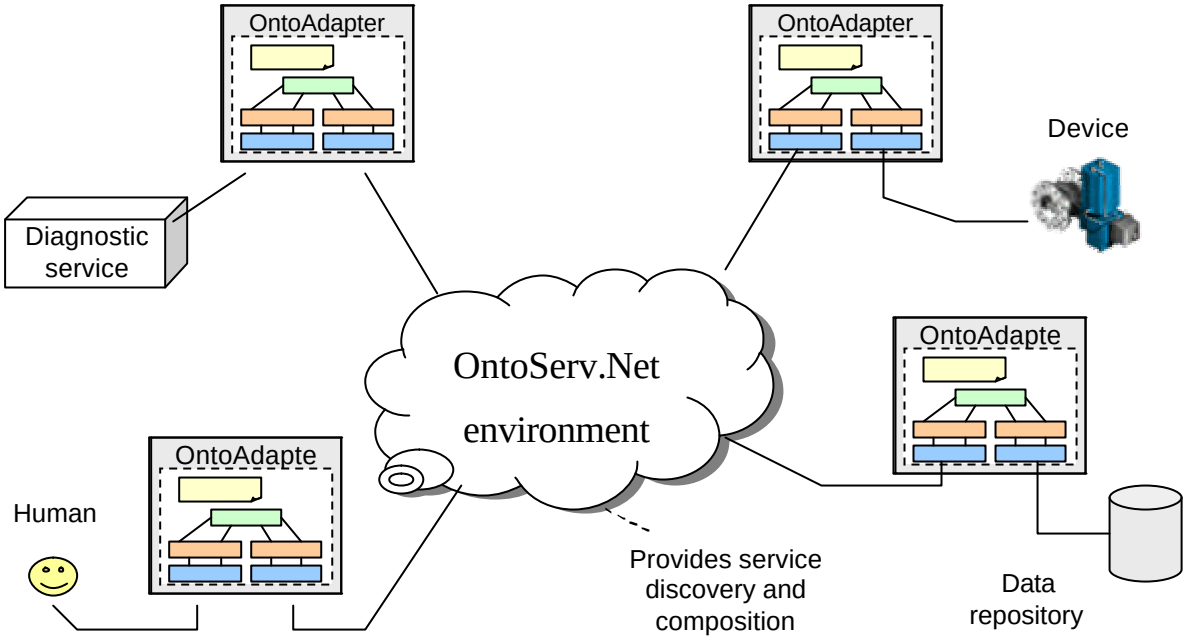


Fig. 20. Universal framework for resource adaptation in OntoServ.Net

4.5 Application Scenarios

4.5.1 Messaging

There are several reasons to present the messages exchanged in OntoServ.Net as RDF-Objects [28]. One of them would be using an ontology-based standardization approach for declaring message structure and linking message elements to a common ontology, to which any OntoServ.Net communication node has to conform.

Consider an example (Fig. 21) of an exchanged message. Terms (concepts) from several ontologies (denoted as element namespaces) are used:

- "**msg:**" for messaging ontology that defines basic message parts and attributes;
- "**rdf:**" refers to most basic RDF-defined constructs;
- "**dev:**" for device ontology, where everything concerning device description is specified (types of parameters, identifiers for devices, etc.)
- "**maint:**" here stands for maintenance ontology with domain specific terms, such as "Reason" (of request), "maintenance actor", "diagnostic result", "fault", etc – these are to be referred from, e.g. device descriptions or messages about maintenance-related actions.

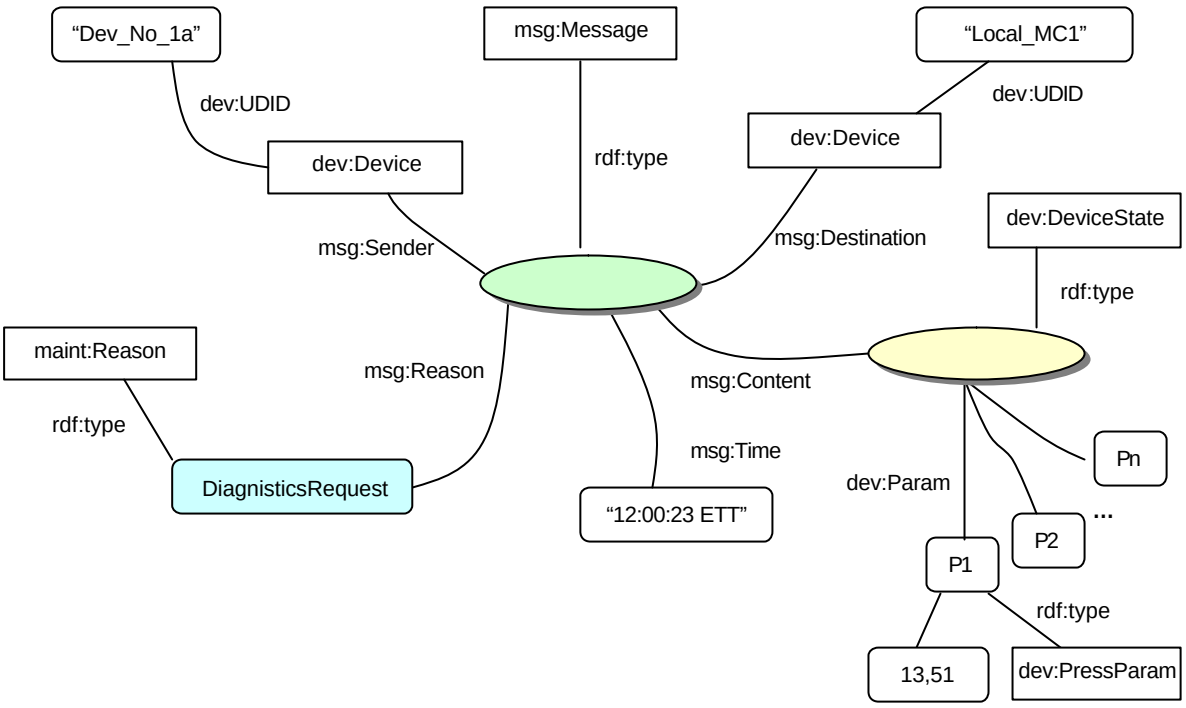


Fig. 21. Example of RDF-based message

Once received, the message is queried for data the receiving part is interested in. E.g. first of all the central element (resource in terms of RDF) of type "**msg:Message**" is found, then an analysis of its properties is performed. All additional data required for processing is taken from the ontology, e.g. that the value of **dev:PressParam** is positive float in pascal units and for device "**XXXX.34**" it is a critical maintenance parameter that has to be in range from A to B, etc… An ontology model can be as complex as it is required for domain knowledge description.

Whereas some of the properties are known to the processing agent (i.e. their use is hard-coded in program text or query strings), others can be new to it. Following "standard" XML-like processing, elements with unknown name (namespace plus tag name) are ignored or an exception is arisen. The use of an ontology and RDF data representation in such a situation proposes an additional option: get a description of the unknown concept (element type), which can be retrieved from the ontology, since its namespace is also its location, and find out how to process such data. Of course, special design of both data processor and the ontologies mentioned is required, but, in general, this approach has potential for ontology-driven software development.

4.5.2 Service Discovery

The location of the required services in OntoServ.Net is not a trivial task since there is no central registry, where each existing service is mentioned with its location. There are three main problems:

1. No centralized access to all services and only peer-to-peer communication is going on;
2. Keyword description of a service is not enough for complex search;
3. Service descriptions are (naturally) heterogeneous, no common standard initially exists.

Following the approach discussed in this work, the descriptions of the services have to be represented in a neutral form that is semantically enriched for automated processing. A domain ontology as a result of standardization process going on in the domain has to be developed. Ultimately, ontology engineering is the greatest challenge.

Secondly, semantic search procedures are required for processing descriptions in a new unified form. The simplest algorithms can consider just the types and property values of some parts of the description, whereas more advanced ones will take into account much more metadata from the ontology, for instance rules for matching individual nodes or even executable software-components for comparison. This is another challenge for service discovery brought with Semantic Web and Resource Description Framework to developers.

Finally, peer-to-peer based search-techniques will be combined with semantic matching methods. It is not a big challenge, though it does not seem to be a less important one.

Let us consider quite a scenario that is quite simple for Semantic Web Service, but rather complex for the already developed Web Services technology, a scenario of matching the *service-image* described (a description of service to be found) in the semantic query and semantic ontology-based description of the available service (Fig. 22). "Traditional" service discovery can fail because there is no "perfect match" between the searched description and the query provided for that. Even if services are described by keywords, not by labels with semantic concepts from a certain ontology, and "soft" matching rules are applied, matching a subset of key words cannot guarantee correct results, because another, not relevant description may be selected. On the other hand, labeled and bound to concepts from a common ontology, pieces of service description "features" are interlinked using relations also from the ontology-vocabulary. This allows defining more flexible matching rules that preserve the semantic correctness of the results.

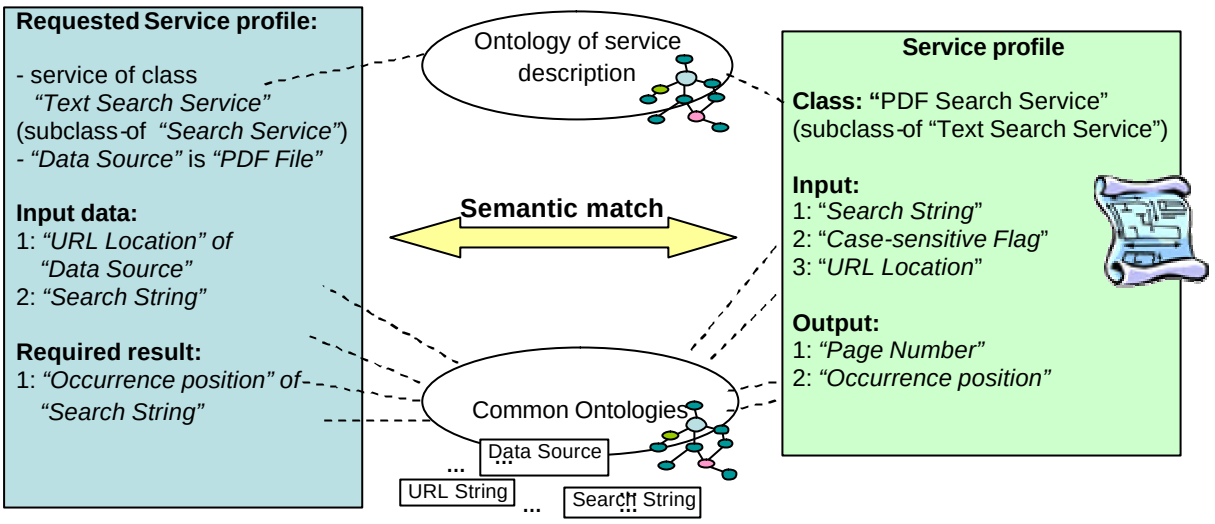


Fig. 22. Semantic match of services capabilities

4.5.3 Service Composition

Service composition in OntoServ.Net is applied in a narrowed sense, as a part of service discovery procedure, when the requested output and available inputs are specified and an appropriate service that uses some of inputs, provides all outputs and has semantics that satisfies the initiator of the service discovery, needs to be found. This simplified schema is not the same as the building-services-on-the-fly idea that proposes making complex workflows dynamically. However, if it is implemented, it will be a substantial basis for further development of semantic-based service composition methods.

5 RELATED WORK: WEB SERVICE SYSTEM INFRASTRUCTURES

This section contains an overview made during the conceptual design of the OntoServ.Net service network. Service platforms, communication architectures and integration strategies are observed and can be used as additional background for the topics discussed in the thesis.

5.1 Intelligent Integration Platforms for Web Services

The integration of web services can be done using ***mediation platforms*** within a ***mediation framework***. The key aspects of what mediation provides:

- Communication and interoperability
- Service composition
- Transaction management

The term "integration of services" is often seen as a statement that some services can use a platform- and program model independent protocol in some sort of a heterogeneous environment. In general, the integration of services means that there is a framework where services can be published and discovered, and where service composition and required mediation can be performed.

Communication in web-services community consists of three types of entities:

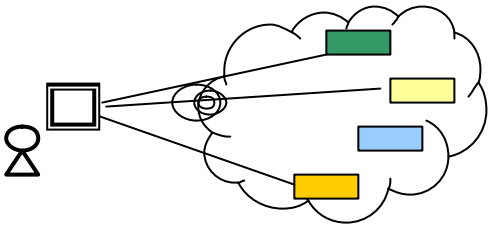
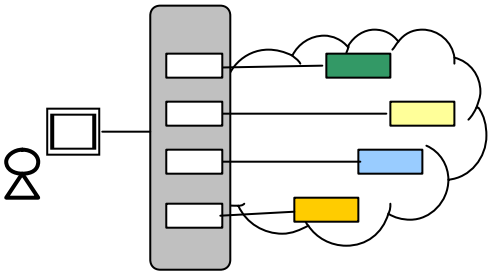
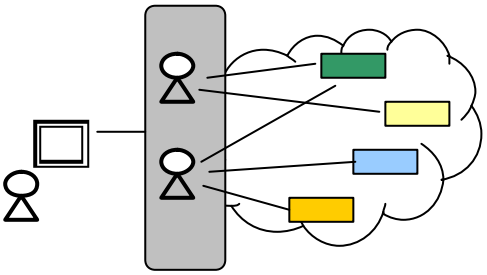
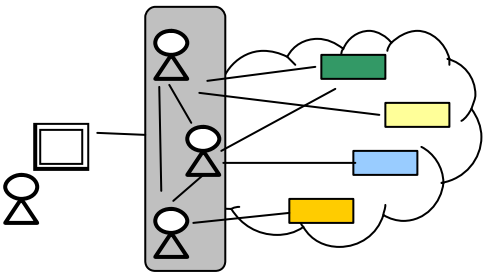
- Requestor: web service, agent or system who requests and consumes service
- Provider: web service, agent or system who provides web service interface
- Mediator: entity who retails, mediates and integrates services or service information

Implementing a thin SOAP/WSDL/UDDI layer on top of some application is not enough to build real Web Services. Trivial Web Services can be built that way, but a lot more infrastructure is needed for companies to create horizontal Web Services applications that span their enterprise.

It is very important to reflect two complementary principles in an appropriate modeling framework: the *loose coupling* and *scalable mediation* of web services. This requires mediators that map between different document structures and different business logics as well as the ability to express the difference between publicly visible workflows (public processes) and internal business logics of a complex web service (private processes) [23]:

- Strong de-coupling of the various components that realize an e-commerce application. This de-coupling includes information hiding based on the difference of internal business intelligence and public message exchange protocol interface descriptions.
- Strong mediation service enabling anybody to speak with everybody in a scalable manner. This mediation service includes the mediation of different terminologies as well as the mediation of different interaction styles.

Table 5.1 – Types of mediation on the service platform

	Picture	Description
Service provision		Provide one service using Web etc. Include web-enabling, wrapping: SOAP
Collection of services		Provide several individual services using Web etc. (UDDI) No integration between the individual services
Mediation of services		Provide several mediating services using Web etc. (E-speak) No integration between mediating services.
Integration of services		Dynamic integration of services by connecting and cooperating several mediating services *(Intelligent Web Services?)*

5.2 Communication Models

Specific communication patterns exist within a mediation framework. Considering them as point-to-point communication between *requestors*, *providers* and *mediator*, the following P2P models are distinguished [34]:

Broker mediated model

In a *mediated* P2P network central servers contain an index of all the content and where it can be found. When the server receives a request, peers hosting the content are identified from the index. The server acts as a broker and *"mediates"* a direct communications link between the requesting peer and the closest, most efficient host peer. The server instructs the requesting peer where it can obtain the file, but is not otherwise involved in the transfer. This greatly reduces bandwidth and storage costs over a central server architecture.

Direct P2P model

This model lets users register information with network neighbors. Searching across the network to find information is done by sending queries to neighbors, and if the neighbors do not know the answer they send the query to their (or a selection of their) neighbors. Techniques like a history profile of the query could prevent cyclic behavior and restrict the chain length.

The major advantage of this approach is the independence of a centralized server that could be a bottleneck in CPU or storage capacity and it also prevents the possibility of censorship.

A disadvantage is the difficulty of finding the peers you need, so efficient P2P search methods (more then just broadcasting) and semantic matchmaking methods (more then keyword search) are to be developed.

5.3 Integration Architectures

Integration architectures described in this section correspond to the basic metaphors used in the basis of service mediation framework. Three most distinctive architectures are presented.

1) Centralized architecture (UDDI, ebXML): registries and hubs.

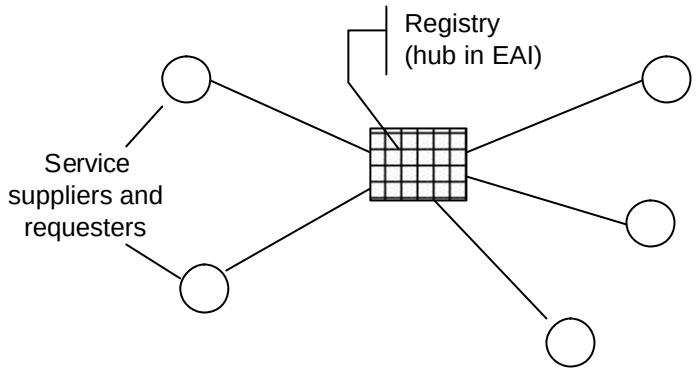


Fig. 23. Centralized mediation architecture

There is a central point which mediates service publishing/discovering in the framework. All services are registered on it (registries) or can be accessed via adapters on it (hubs). The discovering of a service is initiated by a services requester, and after finding the matching service, the central node provides information on how (where, in what manner) to access the discovered service.

The service requester can access the service via adapters proposed by the hub or directly using a platform-independent way of communication.

2) Semi-centralized architecture (E-speak): mediation cloud model

In this architecture there are more than one service registries that are not just replicas, i.e. contain the same information about services, but rather they form a distributed registry. Service publishing is required to only one registry, and after that it can be discovered by any mediation node.

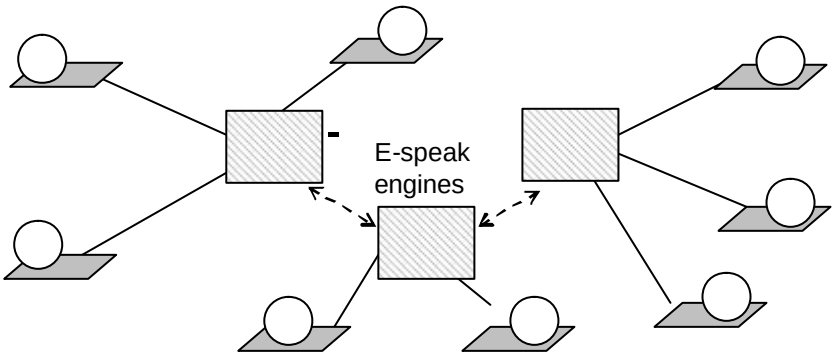


Fig. 24. E-speak engines and E-speak service platforms as an example of semi-centralized mediation architecture

3) Decentralized (P2P network of communicated nodes)

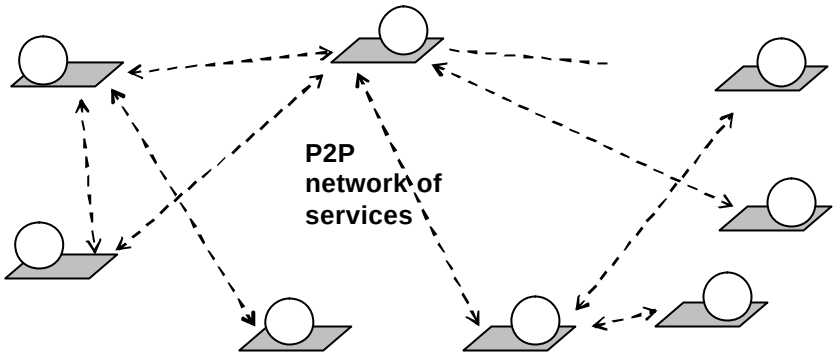


Fig. 25. P2P service network architecture

In the decentralized architecture there are no dedicated nodes (registries, hubs) that serve for mediation (service publishing, service discovery, service composition). Quite the contrary, every participant node has a kind of adapter to the service network which contains information about neighbors and can perform service discovery and routing of service requests.

Such architecture possesses a high degree of fault tolerance since there are no central elements. Though the implementation can be much more complex for such architecture and current solutions have certain restrictions [34], the peer-to-peer communication based on Semantic Web service description and discovery facilities will be compromising between efficiency, flexibility and faultiness.

Specific issues when combining P2P systems with the Semantic Web technology will be:

- Peer selection service

  In order to receive the right answers without flooding the peer network with queries one must ask the “right” peers. Ontology-based peer selection mechanisms need to exploit the similarity of ontologies for this purpose.

- Variation of ontologies and lack of ontological precision

  Different peers will use different, though overlapping ontologies. Alignment, mapping and visualization tools will have to cope with different ontologies, even though no alignments are explicitly specified. Some of the alignments and the mappings may be found by an analysis of peer knowledge by using the methods of the just emerging field of Emergent Semantics (e.g. same file categorized to different concepts indicates alignment.) Ontologies will be produced from various user interactions, like

classifications into folders or usage of meta-data. Ontology definitions will be imprecise and “sloppy” ontologies will be the norm rather than the exception. An inference engine for these ontologies must be able to ask and answer queries to peers in a robust, scalable and often locally contained manner.

- Ontological drift

  In a P2P environment, one cannot expect any maintenance to happen on the ontologies (in fact, users will often not know what is in the ontologies on their machine). As a result, we must design mechanisms that allow the ontologies to update them, in order to cope with ontological drift. Based on the queries and answers elsewhere in the P2P network, ontologies will have to adjust their own definitions accordingly.

CONCLUSION

This thesis is an attempt to observe needs for ontological support in the domain of industrial maintenance and to define a basic structure of the Semantic Web-based approach for solving such problems of information management as semantic-based information retrieval, information integration and intelligent information processing. Possible approach for ontology support for studied case is presented with the background for detailed design of prototype implementation.

Ontologies have emerged as a core technology and fundamental data model representation *metamodel* that will provide advanced functions of intelligent data processing systems and will be the foundation of the Semantic Web. Ontologies are the core elements of a semantic web system. Due to this, ontology support for such systems defines their basic information-related features and design, and influences the internal structure of software.

Ontologies are presented in the thesis as an enabling semantic technology, which can be used for defining the information management infrastructure existing in a certain environment. The design of industrial system with mentioned problem deals with necessity to rethink existing technology in the context of Semantic Web. It requires specific changes in the underlying principles in a base of such systems. Substantial part of new design decisions is around ontology-related issues: ontology engineering, interoperability basics, ontology management in new system, etc.

Ontology support for industrial maintenance domain includes a set of research and development aspects:

- Information infrastructure as ontology-based standards (Ontology engineering)

  Design of industry-wide standards can benefit from using ontologies and description models of the Semantic Web for representation that is used by humans and also can be used by the automated systems in the future;

  Support with ontology engineering tools is one of the most critical issues. Currently, industry-strong tools are only to appear, though standardization efforts of the Semantic Web Activity have created a background for that.

  During this thesis work Protégé-2000 ontology development and knowledge acquisition tool was used for the development of the prototype ontologies and

implementation of provided in the thesis infrastructure of the ontology-based data standardization for industrial maintenance of smart-devices;

- Development of semantic-enabled software

  Distributed information processing that involves heterogeneous systems requires using the widely accepted standards in order to be part of open industrial environments. An ontology-based solution requires new design of the software and development based on the semantic technology. Ontological support for this is a presentation of detailed schema, knowledge and expertise for software developers.

  Another option is the development semantic adapters for the existing software, resources, legacy systems, etc., including people who involved in to the maintenance of the industrial devices. Semantic Adapter Technology is one of innovations of Industrial Ontologies Group and presents very important direction of bringing the Semantic Web-based solutions to the industry;

- Implementation of new techniques of intelligent data processing

  The challenge of this aspect is to rethink existing methods of accessing and processing data that is extended with metadata, including:

  - new ways of information retrieval using the semantic query (which is not available in the used technology);
  - semantic search techniques in the decentralized (peer-to-peer) environment
  - self-annotation methods of the resources (for self-diagnosing of the devices);
  - integration of diagnostics results that are given from multiple sources;

- Design of distributed environment with heterogeneous components that uses ontologies for interoperability and information integration

  Using the example of OntoServ.Net this thesis presents simple schema for design and implementation of the main ontological support needs for development of complex industrial maintenance-oriented environment.

  Main challenge is the development of:

  - Infrastructure of the environment;
  - Principles of data sharing and appropriate data-exchange mechanism;
  - Interaction and mediation models;
  - Support for evolution (future development) of the maintenance environment.

The overall result of this work is a selection of the strongest arguments in favor of the Semantic Web technology coupled with ontologies for application in the industrial domain.

The success of the Web was based on easy information access before it became significantly difficult to search, to present and to maintain the variety of growing information in it. At present, industry is only to meet the creation of large-scale distributed systems, so correct choice of information management strategy can help to skip semantic-poor stages of the information system development.

One of the evolutional steps of Semantic Web before it becomes as well developed as Web nowadays is its applications in specific environments where target problems exist: distributed and heterogeneous resources and services, need for interoperability, great degree of system part independence, need for explicit semantics representation, complex and changing environmental conditions.

The Semantic Web technology has potential to solve some of emergent problems in the industrial system development. This thesis provides one more evidence for that.

REFERENCES

[1] McIlraith S., Son T. and Zeng H.: Semantic Web Services, IEEE Intelligent Systems. Sp. Issue on The Semantic Web 16, 46–53, 2002.

[2] Sankar K., Liu K., Booth D., eds. Web Services Description (WSDL) Version 1.2: Primer, World Wide Web Consortium, 3 March 2003.

[3] UDDI Technical White Paper on http://uddi.org/, Last accessed September 2, 2003.

[4] Simple Object Access Protocol (SOAP) 1.1. W3C Note 08 May 2000 http://www.w3.org/TR/2000/NOTE-SOAP-20000508/.

[5] Sliwa C., From e-Speak to Web Services, Computerworld, December 2, 2002.

[6] Layman A. Web Services Framework. In: Proc of W3C Workshop on Web Services 11-12 April 2001, San Jose, CA USA http://www.w3.org/2001/03/wsws-program/ Last accessed September 3, 2003.

[7] Ankolekar A., Burstein M., Hobbs J. R., Lassila O., Martin D. L., McDermott D., McIlraith S. A., Narayanan S., Paolucci M., Payne T. R. and Sycara K., DAML-S: Web Service Description for the Semantic Web.

[8] Fensel D. and Bussler C. The web service modeling framework WSMF. In White Paper and Internal Report Vrije Unversiteit Amsterdam, 2002, available at http://www.cs.vu.nl/swws/download/wsmf.paper.pdf, 2002.

[9] Zharko A. Peer-To-Peer Ontological Discovery Of Mobile Service Components in Semantic Web, University of Jyväskylä, MIT Department, Finland, 2003.

[10] Khriyenko O. Distributed Mobile Web Services Based On Semantic Web, Master's Thesis, University of Jyväskylä, MIT Department, Finland, 2003.

[11] Terziyan V., Kononenko O., Semantic Web Enabled Web Services: State-of-Art and Industrial Challenges, In: M. Jeckle and L.-J. Zhang (eds.): Web Services - ICWS-Europe 2003, LNCS 2853, Springer-Verlag, 2003, pp. 183-197.

[12] Fensel D. Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce, Springer-Verlag, Berlin, 2001.

[13] Bray T., Paoli J., Sperberg-McQueen C., Maler E. Extensible Markup Language (XML) 1.0 (Second Edition), 6 October 2000.

[14] Bray T., Hollander D., and Layman A., editors. Namespaces in XML. World Wide Web Consortium, 1999. (See http://www.w3.org/TR/REC-xml-names/.)

[15] Brickley D. and Guha R., eds. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Working Draft 23 January 2003. Latest version is available at http://www.w3.org/TR/rdf-schema/.

[16] Berners-Lee T., Hendler J., and Lassila O.: The Semantic Web, Scientific American, May 2001.

[17] Workshop on Ontologies in Agent Systems, 11 March 2003, http://oas.otago.ac.nz/OAS2003.

[18] Gruber T. A Translation Approach to Portable Ontology Specification. Knowledge Acquisition 5: 199-220, 1993.

[19] TopQuadrant white paper: Ontology Myth or Magic? Toward the Practical Applications of Ontology-enabled Knowledge Solutions. In proceedings of Semantic Technologies for E-Government, USA, September 2003.

[20] Connolly D., F. van Harmelen, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and Stein L. DAML+OIL (March 2001) Reference Description. W3C Note 18 December 2001.

[21] Dean M., Connolly D., F. van Harmelen, Hendler J., Horrocks I., McGuinness D., Patel-Schneider P., Stein L., eds. Web Ontology Language (OWL) Reference Version 1.0, 12 Nov 2002. This W3C Working Draft is available at http://www.w3.org/TR/2002/WD-owl-ref-20021112/.

[22] Sycara K. Autonomous Semantic Web Services. 15th Conference on Advanced Information Systems Engineering, Velden, Austria, 2003. See at http://www-2.cs.cmu.edu/~softagents/presentations/parisseminarcolor.pdf.

[23] Fensel D., Bussler C., and Maedche A. Semantic web enabled web services. In Proceedings of the International Semantic Web Conference 2002, LNCS, Springer, pages 1–2, 2002.

[24] Fensel D. et al. (eds.): Spinning the Semantic Web: Bringing the World Wide Web to its full potential, MIT Press, 2002.

[25] Kreger H., IBM Soft. Group, Web Services Conceptual Architecture, 2001.

[26] Fensel D., Bussler C., Ding Y. and others. Semantic Web Application Areas. In Proceedings of the 7th International Workshop on Applications of Natural Language to Information Systems, Stockholm - Sweden, June 27-28, 2002.

[27] Paolucci M., Kawamura T., Payne T., Sycara K., Importing the Semantic Web in UDDI. In Proceedings of Web Services, E-business and Semantic Web Workshop, 2001.

[28] Barnell A., RDF Objects, EuroWeb 2002 Conference - The Semantic Web and the Grid, UK, 17-18 December 2002.

[29] Trastour D., Bartolini C. and Gonzalez-Castillo J. A Semantic Web Approach to Service Description for Matchmaking of Services: In Proc. International Semantic Web Working Symposium (SWWS), Stanford, CA, USA, July 2001.

[30] Klein M. and Bernstein A., Searching for Services on the Semantic Web Using Process Ontologies. In the Emerging Semantic Web - Selected papers from the first Semantic-Web Working Symposium, Amsterdam: IOS press, pp. 159-172, 2002.

[31] Karvounarakis G., Alexaki S., Christophides V., Plexousakis D., Scholl M. RQL: A Declarative Query Language for RDF, The Eleventh International World Wide Web Conference (WWW'02), Honolulu, Hawaii, USA, May 7-11, 2002.

[32] Sirin E., Hendler J., Parsia B. Semi-automatic Composition of Web Services using Semantic Descriptions. *Accepted to "Web Services: Modeling, Architecture and Infrastructure" workshop in conjunction with ICEIS2003*, 2002.

[33] McGuinness D. Industrial Strength Ontology Management. Knowledge Systems Laboratory, July, 2001.

[34] Siebes R. Semantic Web and Peer-to-Peer (SWAP) Project Deliverable D1.1 “Peer to Peer solutions in the Semantic Web context: an overview”, 2002.

[35] Brickley D. and Guha R., eds. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Working Draft 23 January 2003. Benjamins V., Contreras J., Corcho O., Gómez-Pérez A. (UPM). Six Challenges for the Semantic Web. ISWC2002, June, 2002.

[36] Ermolayev V., Keberle N., Plaksin S. Towards Agent-Based Rational Service Composition - RACING Approach. Int. Conf. on Web Services Europe (ICWS'03 Europe), Sept., 23-25, 2003, LNCS Series.

APPENDIX A. SERVICE DESCRIPTION TECHNOLOGIES COMPARED

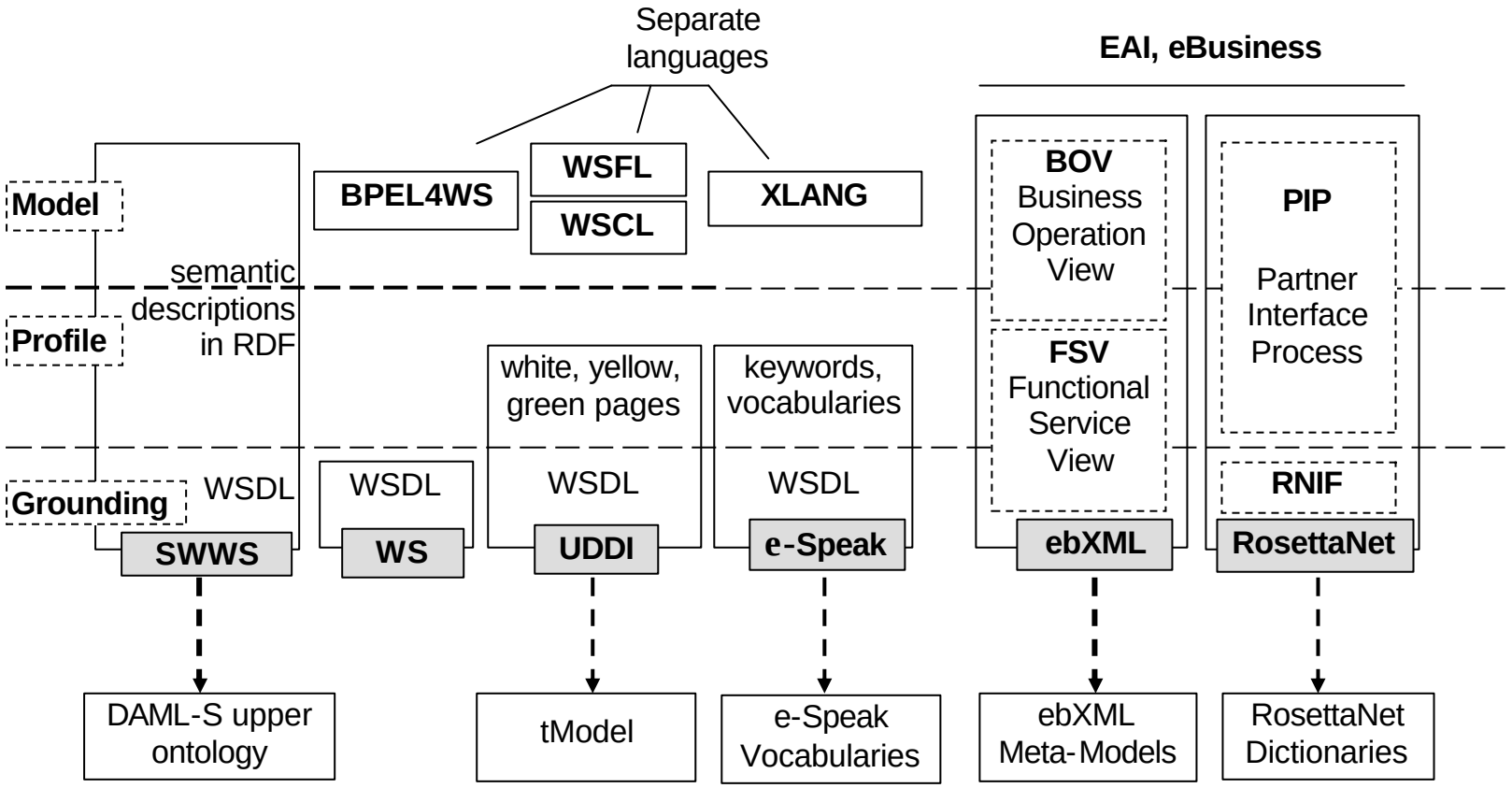


Figure A.1 - Technologies and description languages concerning Web Services. Correspondence to *ServiceModel*, *ServiceProfile* and *ServiceGrounding* parts of DAML-S service description

APPENDIX B. ONTOLOGICAL SUPPORT: STRUCTURE, PROCESS AND TOOLS

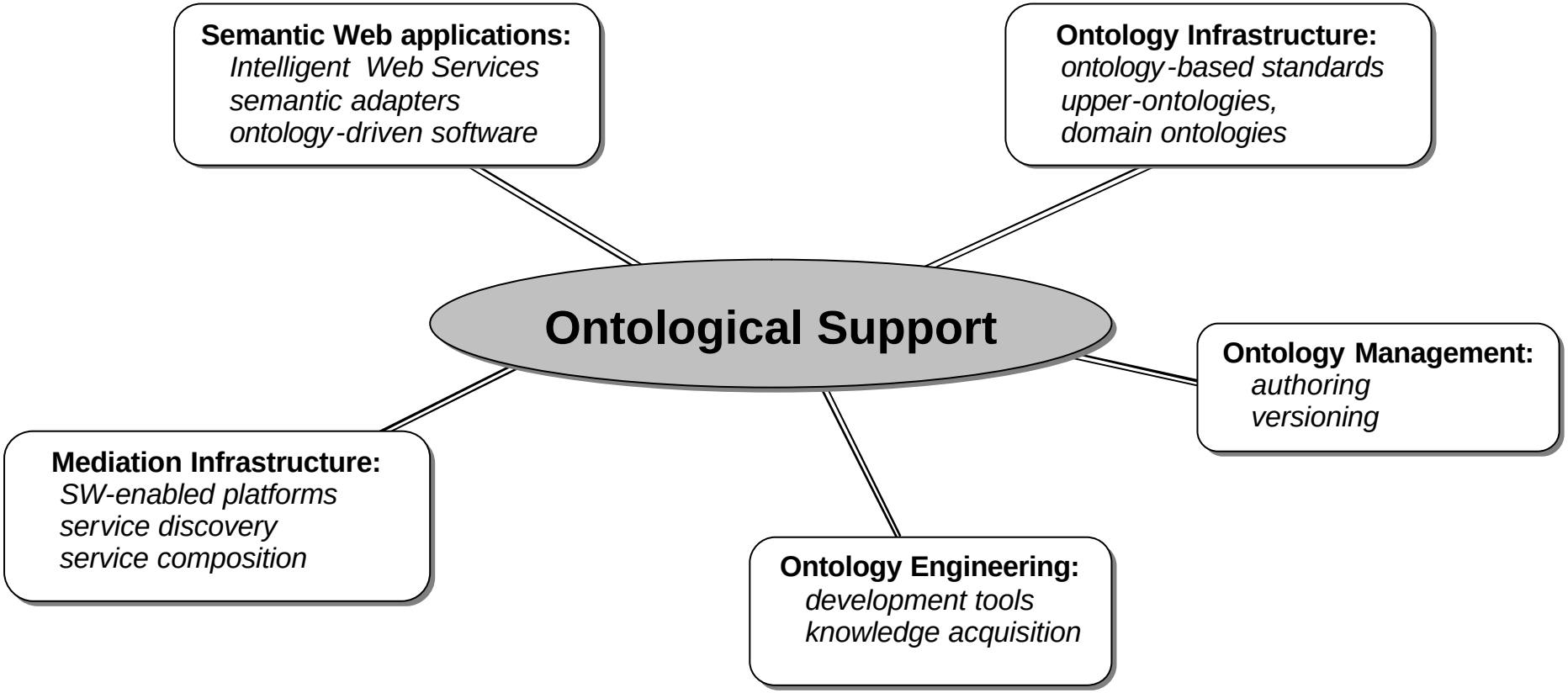


Fig. B-1. Ontological support: aspects of research and development.

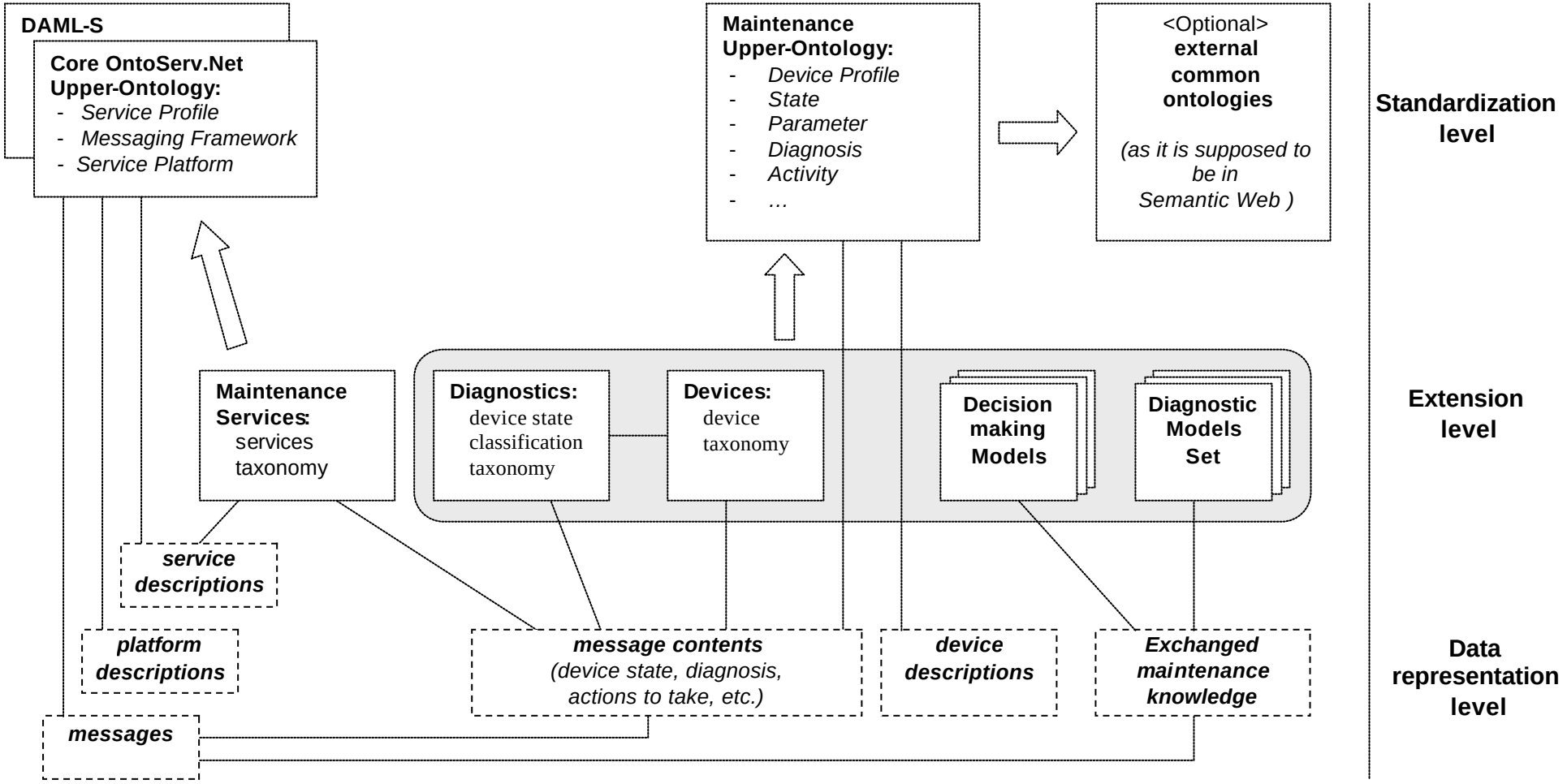


Fig. B-2. Upper-ontologies, ontology extensions and ontology-based data existing in the maintenance service network. Arrows show associations (“refines”, “extends”, “refers to”, “uses as a vocabulary”, etc.)

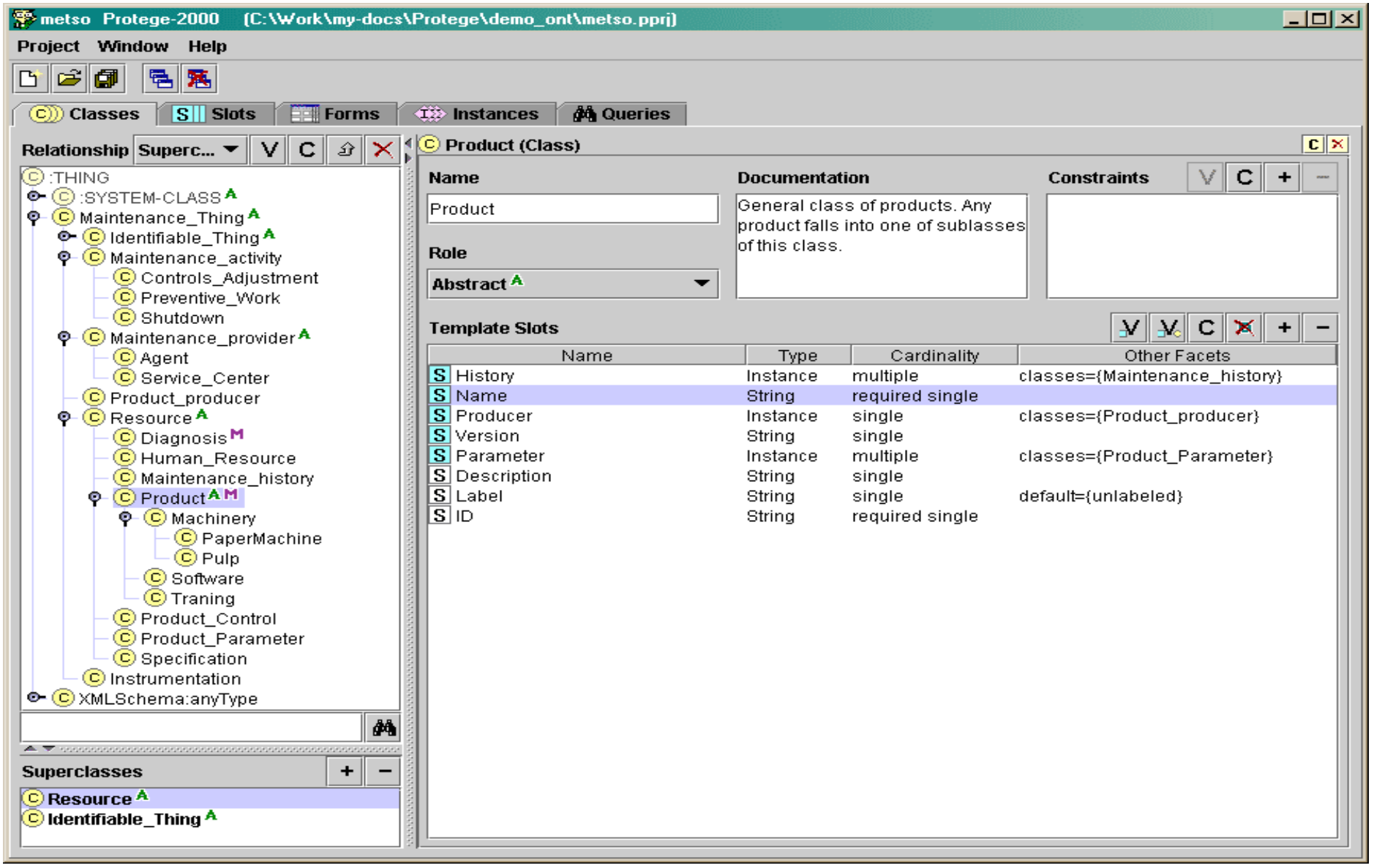


Fig. B-3. Ontology development: screenshot of Protégé-2000 ontology authoring tool; Maintenance ontology design.