Using Ontology to bind Web Services to the Data Model of Automation Systems

Zaijun Hu

ABB Corporate Research Wallstadter Straße 59, Ladenburg, Germany
email: zaijun.hu@de.abb.com

Abstract. Supplying integrated data model with multiple structures for addressing diverse requirements from different production lifecycles and control levels is the natural development of automation systems. Data processing, presentation and manipulation require well-designed computation model that consists of interrelated groups of function components or applications that can be realized in the form of web services. Binding the available web services to the designed data model becomes a challenging task if a great amount of web services is confronted. In this paper we will use ontology technology to address the binding problem. We will present the required ontology model including the formal expression of ontology, object model, mapping to XML representation and the corresponding system architecture for binding web services.

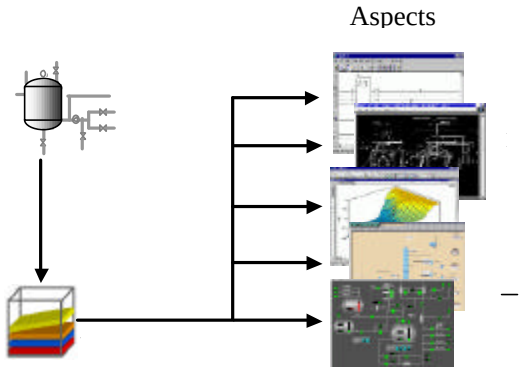
Keywords: ontology, web service, automation system

1 Introduction

Automation systems are becoming more and more complex, integrated and multiple-functional to deal with various needs from the production lifecycles including purchase, design, engineering, operation, etc and the different control levels ranging from the field device layer to ERP layer. To simply the development of such a system the design of data model is usually decoupled from the building of application logics. The separation of data modeling from the creation of the computational model that provides diverse applications or components for using, processing and viewing the data described in the data model facilitates, on the one hand, the development of automation systems in flexibility, extensibility and distributed development paradigm. But on the other hand it also, at the same time, creates problems such as heterogeneity in the implementation platform. Some applications use Java for the implementation, and some others use COM from Microsoft as implementation. Because of the platform- and language-neutrality web services are gradually showing the undoubted advantage in addressing the heterogeneity problem. With the development of more and more web services the binding or the association of the developed web services processing, viewing and using data to the corresponding data described in the data model is becoming one of new challenges. In this paper we are going to present an ontology method to address the binding problem.

2 Industrial IT - ABB Aspect Integrator Platform

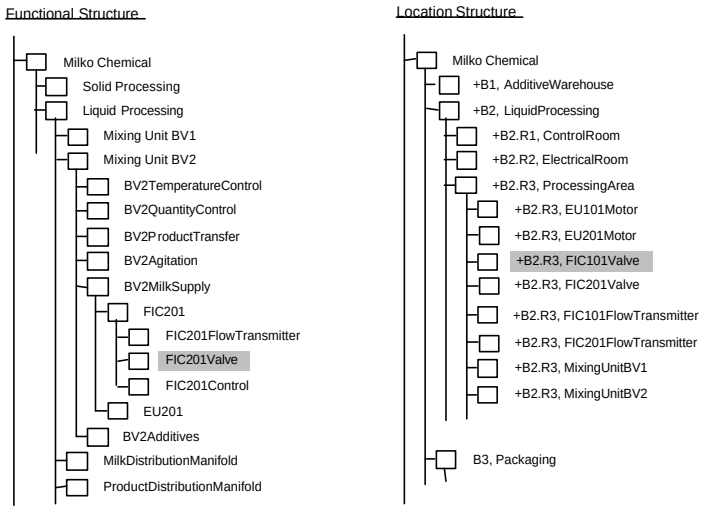
ABB Aspect Integrator Platform (AIP) is created for the integration of different information hierarchies – device layer, automation layer, manufacture execution and asset management layer, enterprise management layer and collaboration layer, and it also creates an integration platform for integrating diverse functional applications and components. The information model defined in AIP provides an unified modeling mechanism conform to IEC 61346 to organize or structure the information entities in an integrated automation system that contain plants, equipments, components, devices, instruments, sensors, actuators, controller etc. The basic elements in the model are Aspect Object and Aspect. An **Aspect Object** in the model is a container that holds different parts of an object in an automation system. Examples for an object can be a reactor, a pump or a node (computer). An **Aspect** is one 'piece' of data and operations that is associated with an object. Typical aspects of an (process) aspect object are its control program, operator faceplate, trend configuration, function specification etc. Figure 1 shows an example for presenting the AIP architecture concept



A structure defined in the AIP architecture is conform to the one in IEC 61346 and defines semantic relationships among the organized objects and also rules for creation of the semantic relationship. In IEC 61346 a structure is separated from the objects and represented through an additional aspect. In this way an object can be organized in different structures at the same time. IEC 61346 presents three examples of information structures that are important for the design, engineering, operation and maintenance – function-oriented structure, location-oriented structure and product-oriented structure. Each structure is represented through a defined hierarchy.

The function-oriented structure organizes objects based on their purposes or functions that are played in a system. The structural hierarchy is created through the subdivision of the functions or the purposes of objects. The function-oriented structure is the result of design and is interesting for engineering, operation and maintenance. The location-oriented structure defines the topologic relationship among

objects. The structural hierarchy of a location-oriented structure results from definition of spatial constitution relationship, e.g. ground area, building, floor, room etc. The location-oriented structure is useful for the engineering, operation, maintenance and so on. The product-oriented structure shows the breakdown of product-related information for a given product type. A product can consist of many components and each component can be made up of further subcomponents. This constituent relationship determines the product structure hierarchy. All structures are hierarchical, i.e., the parent-child relationships among objects determine the hierarchy. Fig. 2 shows two structures – function- and location-oriented structure.



3 Some definitions

In this paper we will use some definitions that build the discussion basis of the paper.

3.1 Ontology

"An ontology defines the terms used to describe and represent an area of knowledge"[4]. The purpose of ontologies is to share information in a certain context. Ontologies include computer-usable definitions of basic concepts in the context and the relationships among them. They encode knowledge in a context and also knowledge that spans the context. In this way, they make that knowledge reusable. In this paper ontologies are used to establish association between the data model of a domain on the one side and the operations in the form of web services on the other

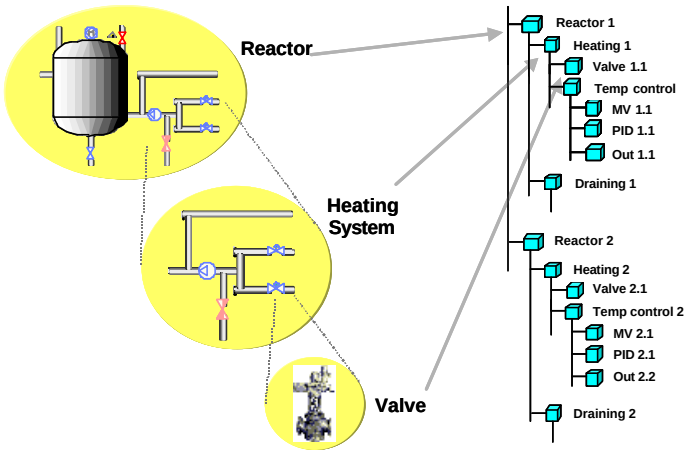
side. In other words they create a matching and binding mechanism between data and its operations.

3.2 Domain ontology and operation ontology

A domain ontology is an ontology for a certain domain. A domain is just a specific subject area or area of knowledge, like medicine, tool manufacturing, real estate, automobile repair, financial management, etc. In our context a domain refers directly to an industrial process such as energy generation and the related components such as power plant, gas turbine, boiler, steam turbine, generator etc. that take part in an process execution. An operation ontology is an ontology for data operations. It defines terms to describe and represent the knowledge in the data operation area, e.g. simulation, mechanical drawing, visualization, optimization. The differentiation between the domain ontology and the operation ontology is necessary to better deal with the association of data operation with data.

3.3 Data model

A data model in the context of this paper is used to describe and represent an industrial process with the related components and the control of the process with all necessary units. Fig. 3 shows an example of the data model



3.4 Computation model

A computation model defines structure, categorization and classification of data operations that deal with the operations of data such as drawing, simulation, cost

calculation, report, visualization etc. It contains a group of related functions for operating data. Data operations are built in form of software applications, components, and in this paper in the form of web services. A computation model can use the operation ontology to describe the structure, categorization and classification of operations.

3.5 Binding

After defining the data model and computation model we can begin with the binding definition. A binding is a process that associates a computation model containing a group of functions to the corresponding data model. The binding is to find the most suitable functions that can be used to the interested data and to provide a guideline to users for the association. What we are interested in this paper is the binding of web services that implement functions based on the web service protocol to the related data. Through the ontology technology we can convert the operation binding to the ontology binding, namely binding of the operation ontology to the domain ontology.

4 Ontology model

4.1 Expression

To give a mathematic expression of an ontology we use the atomic concept term and composite concept term defined in [6]. Let $V = \{v_0, v_1, \ldots, v_n\}$ be the universal set of atomic concept terms that can be used and $R = \{r_0, r_1, \ldots, r_m\}$ be the set of all relationships between the terms, V' be a subset of V ($V' \subseteq V$, $V' = \{v'_0, v'_1, \ldots, v'_k\}$) and R' be defined as subsets of R ($R' = \{R_i' \mid R_i' \subseteq R \wedge i=1,\ldots, k\}$), $I = \{I_i \mid i=1, \ldots, k\}$ be a piece of information, the ontology Ψ can be expressed by $\Psi = \{\Psi_i \mid i=1, \ldots, k\}$ where Ψ_i is an ontology element and can be expressed by a tuple $<v_i, R'_i, I_i, S>$ with $v_i \in V \wedge R'_i \in R' \wedge I_i \in I$. S in the tuple stands for the structure discussed in the section 2 that is interesting in the context of this paper. We associate a piece of information to each atomic concept term to give some explanatory information on it. For example we can take an example of DC motor, such information can be explanation of the principle, the deployment area, main features and so on. This would be a great help for users that are not familiar with the meanings of the term.

4.2 Relation

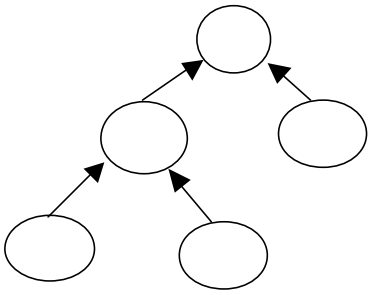
Is-A Relation

Is-A relation is generalization relation. It is a binary relation and thus needs two terms taking part in the relation. We use symbol $\phi_{Is\text{-}A}(v_i, v_j)$ where i, j =1,…, k to signify the generalization relationship between v_i and v_j, or v_i is a generalization of v_j. An example of generalization relation is DC motor and electric motor. Electric Motor is a generalization of DC motor, expressed by $\phi_{Is\text{-}A}$ (Electric Motor, DC motor). Is-A relation has properties of reflexivity and transitivity, namely

- $\phi_{Is\text{-}A}(v_i, v_j)$ is true
- $\phi_{Is\text{-}A}(v_i, v_j) \wedge \phi_{Is\text{-}A}(v_j, v_l) \rightarrow \phi_{Is\text{-}A}(v_i, v_l)$

Is-A relation can be used to organize the concept terms in a hierarchical structure. Imposing hierarchical structure on the concept terms can obtain precision increase substantially [13]. Another benefit of that is the information reuse, because the information associated to the generalized concept term can be used by its child. The Is-A relation also enhances possibility of the correct binding.

For the Is-A relation it is necessary to take into account the height of hierarchy levels. Fig. 4 illustrate the concept



For the Fig. 4 the relations $\phi_{Is\text{-}A}(A, B_1)$, $\phi_{Is\text{-}A}(A, B_2)$, $\phi_{Is\text{-}A}(B_1, C_1)$, $\phi_{Is\text{-}A}(B_1, C_2)$, $\phi_{Is\text{-}A}(A, C_1)$, $\phi_{Is\text{-}A}(A, C_2)$ hold. The difference between $\phi_{Is\text{-}A}(A, B_1)$ and $\phi_{Is\text{-}A}(A, C_1)$ is that B_1 has a direct Is-A relation to A while C_1 has an indirect Is-A relation to A via B_1. We call $\phi_{Is\text{-}A}(A, B_1)$ the Is-A relation of the first order and $\phi_{Is\text{-}A}(A, C_1)$ the Is-A relation of the second order and express them with $\phi_{Is\text{-}A}(A, B_1)^1$ and $\phi_{Is\text{-}A}(A, C_1)^2$. Is-A relation is very important in inferring some possible operations that can be bound to certain data.

Sibling relation

The sibling relation is used to describe the siblings in the Is-A relation hierarchy. It can be expressed by

$$\phi_{sibling}(v_{i,}\ v_j) \mid \exists k \phi_{Is\text{-}A}(v_{k,}\ v_i)^1 \wedge \phi_{Is\text{-}A}(v_{k,}\ v_j)^1$$

The sibling relation is reflexive, transitive and symmetric.

Specific Relation

As an inverse part of the Is-A relation we use the specific relation for expressing the specialization of a concept term. For example DC motor is a specialization of electric motor. The specific relation is also a binary relation requiring two terms. We use symbol $\phi_{spec}(v_{i,}\ v_j)$ where i, j =1,…, k to denote the specific relationship between v_i and v_j, or v_i is a specialization of v_j. Between the specific and Is-A relation the following implication holds

$$\phi_{Is\text{-}A}(v_{i,}\ v_j) \leftrightarrow \phi_{spec}(v_{j,}\ v_i)$$

Just as the Is-A relation the specific relation is also reflexive and transitive. It is to note that $\phi_{Is\text{-}A}(v_{i,}\ v_j) \neq \neg\phi_{spec}(v_{i,}\ v_j)$. The specific relation defined here is different from the definition in [6] that also includes the synonym relation. In our definition we don't consider the synonym relation because including synonym could result in complexity in the structure of the concept terms and redundancy. We can also use the specific relation to structure the concept terms just as the Is-A relation. The question is if it is necessary to include the specific relation in the ontology model.

Just like the Is-A relation we also introduce the height of hierarchy levels and express it with

$$\phi_{spec}(v_{i,}\ v_j)^n.$$

The expression can be read as the specific relation of the nth order

In the binding process the specific relation plays the same role in the identification of possible operations as the Is-A relation.

Part-Whole relation

The part-whole relation describes the composition or containment relation between two concept terms. For example the relation between room and walls is such one. The part-whole relation is also a binary one needing two terms. We introduce symbol $\phi_{part}(v_{i,}\ v_j)$ to indicate that v_i is a part of v_j. The part-whole relation is reflexive and transitive. Just as the Is-A relation the Part-Whole relation also has order and can be expressed by $\phi_{part}(v_{i,}\ v_j)^n$. Part-Whole relation can be used to infer some possible operations that can be associated to the corresponding data. The relation can be used to determine if it is possible to bind an operation with a data object or node when the data object has the part-whole relation to another one.

Buddy relation

The buddy relation is defined by $\phi_{buddy}\ (v_{i,}\ v_j)|\ \exists k \phi_{part}\ (v_{i,}\ v_k)^1 \wedge \phi_{part}\ (v_{j,}\ v_k)^1$. The buddy relation is used to describe the relation between terms that share the same container or are parts of same terms. For example floor and wall have the buddy relation because they both are parts of room. The buddy relation is reflexive, transitive and symmetric. Like the part-whole relation the relation can also be used for the binding.

Synonym relation

A term can have synonyms that have the same meanings as the term itself. In the binding process a data object in the data model represented by a term in the domain ontology can be associated with operations represented by a term in the operation model if both terms have the synonym relation. The synonym relation can be expressed by

$\phi_{Synonym}\ (v_{i,}\ v_j)$

The synonym relation is reflexive, transitive and symmetric.

Theorem: a data object or data node (D) represented by a term in the domain ontology (TD_1) can be bound with an operation (Op) represented by a term in the operation ontology (TO_2) if (TO_2) is bound to (TD_2) and $\phi_{Synonym}\ (TD_{1,}\ TD_2)$ holds.

Disjoint relation

A term has a disjoint relation with another term if both terms don't contain shareable semantic. The disjoint relation can be expressed by

$\phi_{Disjoint}\ (v_{i,}\ v_j)$

The Disjoint relation is symmetric.

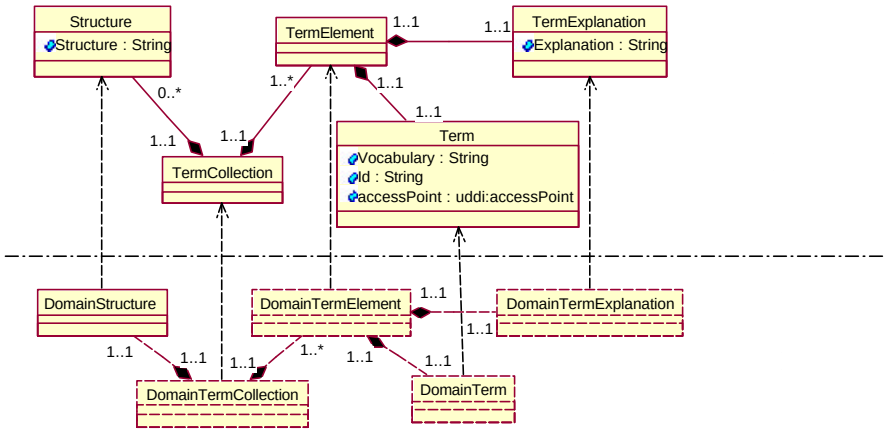
Theorem: a data object or data node (D) represented by a term in the domain ontology (TD_1) cannot be bound with an operation (Op) represented by a term in the operation ontology (TO_2) if (TO_2) is bound to (TD_2) and $\phi_{Disjoint}\ (TD_{1,}\ TD_2)$ holds.

Binding relation

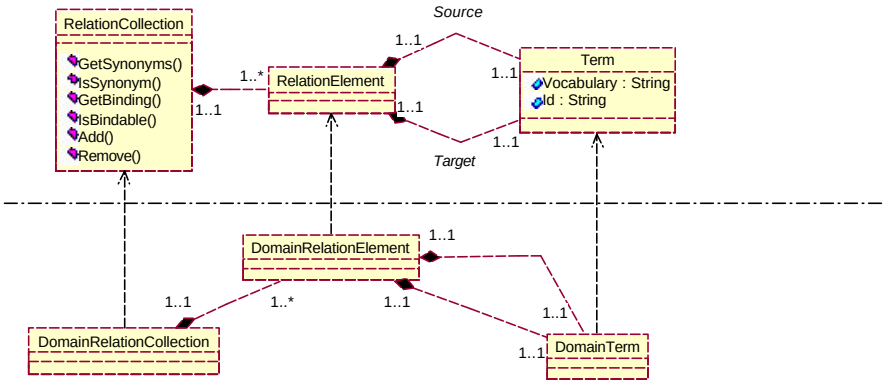
A binding relation describes that two terms are bound together in a certain way. It can be expressed by $\phi_{Disjoint}\ (v_{i,}\ v_j)$. The binding relation is reflexive, transitive and symmetric.

4.3 Object Model of an Ontology

The object model of an ontology describes a model that provides an way of mapping the ontology to memory for programmatic access, navigation and



manipulation. For the ontology modeling we use MOF (Meta Object Facility) [7] proposed by OMG because MOF is widely accepted in the industry and uses the layered concept (data, model, metamodel, metametamodel) that is very suitable in our addressed context. For the simplicity reason we just adopt three layers from the MOF: data, model and metamodel. The metamodel of an ontology consists of two parts: the term model and the relation model. The term model describes the meanings of terms. It is just like a dictionary that contains explanation of vocabularies. The term model defines four meta-classes: TermCollection, TermElement, Term, TermExplanation. TermCollection is a collection of TermElement that contains two elements: Term (Vocabulary, Id) and TermExplanation. The meta-class TermCollection also has an attribute called Structure that is introduced to deal with the issue of multiple structures in the data model discussed in Section 2 and 3.3. Instantiated from the metamodel the model layer is created for the modeling of the domain ontology and the operation

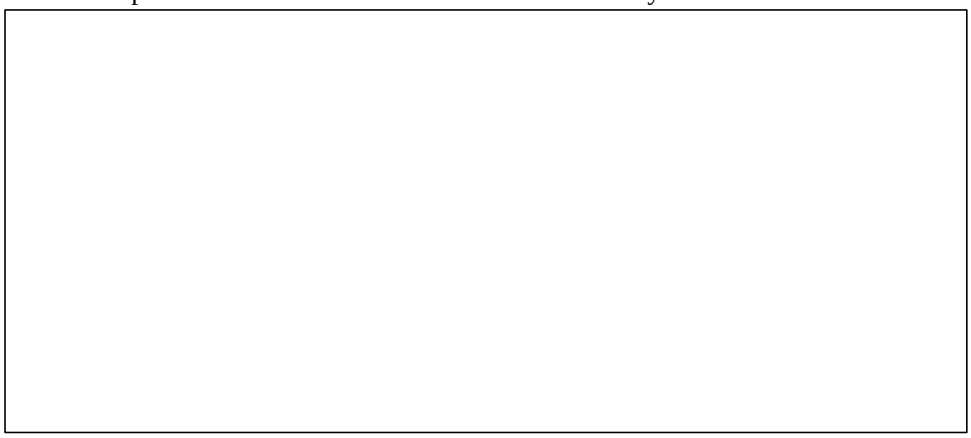


ontology. The created classes for the model layer and the metamodel for the domain ontology are displayed in Fig. 5. The same instantiation can also be applied for the operation ontology. In the figure accessPoint is used for reference to a data object or a URL address for the web service. For the simplicity reason the instantiated relation model for the operation ontology is omitted here.

The relation model describes the relations among terms that are defined in section 4.2. For the relation model two meta-classes are defined for the metamodel layer: RelationCollection and RelationElement. The RelationCollection is a collection of RelationElement that associates two terms. The RelationCollection also defines some operations that can be applied to the class. The operation can be classified into three categories: retrieval, check, and change. The retrieval operations are ones that can be used to retrieval terms that have specified relations to the specified term. The check operations check if both specified terms have the specified relation. The change operations are exploited to change the collection such as adding, removing, or changing relations. The metamodel for the relation model can be instantiated to the model layer of the relation model. In this paper the model layers for the domain ontology, the operation ontology and the binding relation between the domain ontology and the operation ontology are instantiated. Fig 6 shows the diagram illustrating the metamodel layer and the model layer of the relation model. The diagram only displays an instantiation example.

4.4 XML Representation

The object model discussed in 4.3 just creates a precondition for the memory mapping of an ontology. It describes the basic elements of the model, their relationships and semantics. But it doesn't define the way of how the elements are

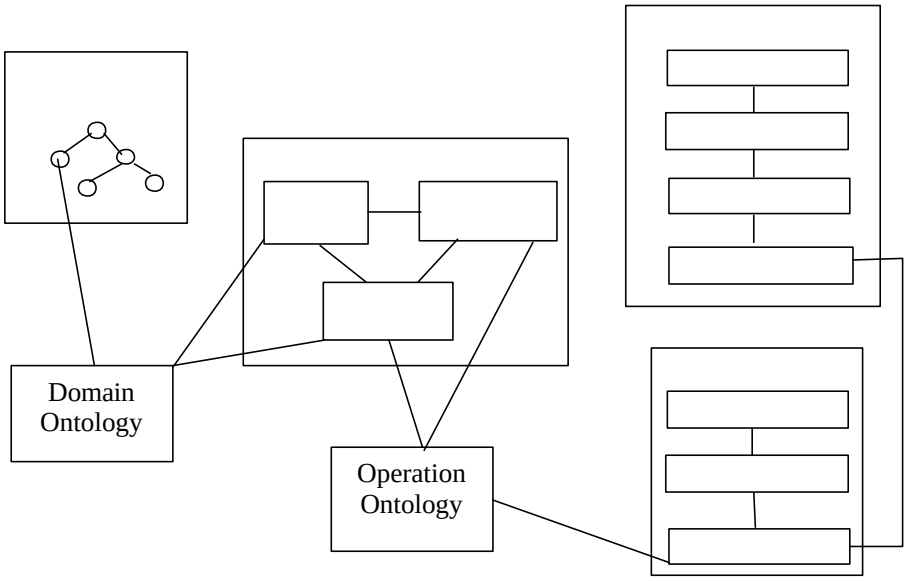


structured in the memory, how data is validated against the model layer, how the model layer is checked against the metamodel layer, how to access and to navigate the model. XML is already a widely accepted markup language that is neutral to different

platform and diverse programming language. What is more important of using XML is the semi-structural feature, i.e. XML provides not only a way of putting rule, structure and relationship to data but also flexibility in extension of data structure. Because of the convincing advantages of XML we will also use it for representing the ontology object model. Fig. 7 shows an XML schema file that maps the term model to the XML representation. The figure illustrates the concept on how we use XML to describe the metamodel layer and the model layer of the ontology object model. In XML schema there is no instantiation mechanism for the metamodel instantiation. Therefore we resort to another mechanism, i.e. substitution introduced in XML schema. First of all we defined XML elements: TermCollection, TermElement and Term, which are abstract. That means the elements must be replaced in the instance document with the defined substitution elements that are defined through "substitutionGroup". In this way we emulated the instantiation mechanism.

5 Web Service and System Architecture

In [9] Myerson presented different web service models with the so called architecture stack model from different organizations such as WebServices.Org, IBM, Oracle, Microsoft, Sun, W3C and so on. The architecture stack model describes the layers such as the transport layer, the message layer, the description layer, the

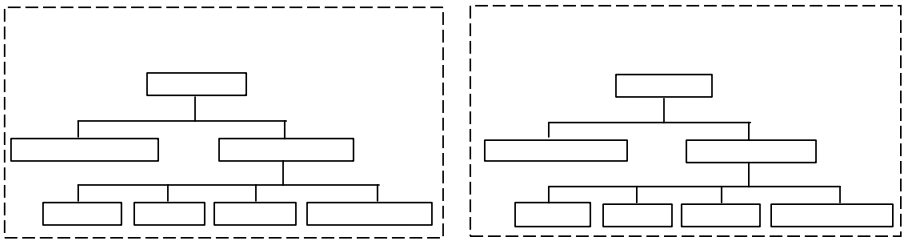


discovery layer, the web service flow layer, and the negotiation layer. What are relevant for the context of this paper are the discovery layer described by WSDL (Web Service Description Language) and the discovery layer that can be specified

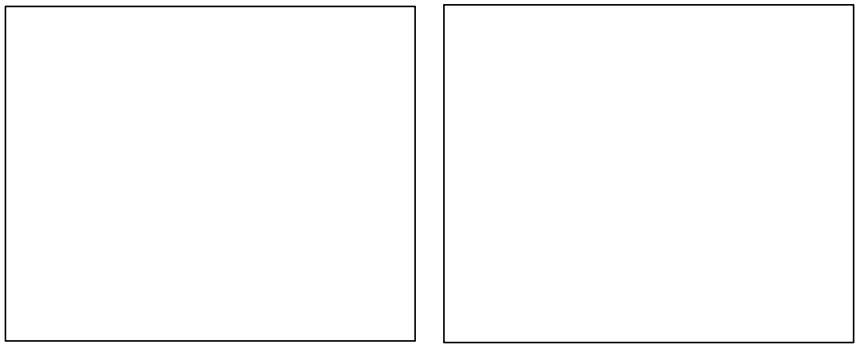
through the UDDI specification, a common initiative from IBM, Microsoft and so on. UDDI provides a way to publish and discover information about web services. But UDDI does not define mechanism to deal with ontology and how to bind web services representing the computation model to the target data. Therefore an addition concept or extension and the corresponding component are required. Fig. 8 shows the concept for the extension. There are a domain ontology component that describes the term structure of the domain side and is attached to the data model of interest and an operation ontology component that determines the term structure of the operation side and is connected to the web service component providing web services. Each web service is identified through an URL address. In UDDI the URL address is contained in the accessPoint XML element while in the WSDL it is embedded in the Port XML element. The accessPoint and Port element uses the same URL address to reference the same web service. Therefore they can be used to connect the UDDI and WSDL. The relation between the data model and the web service is established by the ontology service that is the central component in the architecture. The component checks the potential web services based on the binding relation between the domain ontology and the operation ontology. Once it finds a matched web service it will bind the web service to the target data.

6 Example

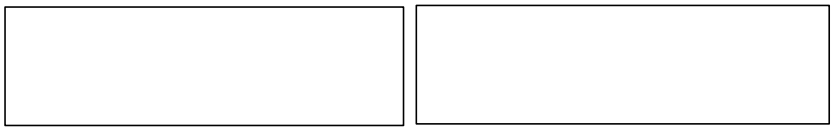
For illustration purpose we take power plant as an example for the domain side and simulation for the operation side or the web service side.



The domain ontology and the operation ontology can be expressed in XML (Fig. 10). For the space reason we don't list all elements in the example. The two



XML document will be stored in the memory as two separate document instance. These two documents can also be seen as two thesauruses that can be used for the binding engineering or binding process. The relations of the domain ontology and the operation ontology are contained in two separate XML documents: domain relation document and operation relation document (Fig. 11). The two documents can be used to get all related terms for the specified terms. For example in the XML document for the operation relation model you can find which element is child element of the “Continuous”. For the binding purpose the binding relation is used that can be expressed in the Fig. 12. In the XML document for the binding relation the term FossilbePowerPlant is associated with Simulation while the term “CombinedCycle” is attached to “Process”. This binding relation is created in the ontology engineering, i.e. through ontology expert. Users who want to use simulation program don’t need to know which term of the domain ontology is attached to the corresponding operation ontology. He only needs to tell the ontology service that he needs a



simulation service. The ontology service will find the most suitable simulation service, in our example the process simulation service.



7 Conclusion

In this paper we presented an ontology method that can be used to deal with the binding problem, especially for the case of a great among of web services. The main contribution in this paper is creation of the ontology model for the binding problem. Two kinds of ontology are identified to convert the binding problem to the connection of the domain ontology and the operation ontology. MOF model is used for definition of the object model. We have expressed the relations in formal form and then created the object model and the XML presentation. We also presented the system architecture and web service model. Further work is required for the ontology engineering.

8 Reference

[1] Ragaller, K., An Inside Look at Industrial IT Commitment, ABB Technology Day, 14 November 2001.

[2] Krantz L., Industrial IT - The Next Way of Thinking, ABB Review 1/2000

[3] Lars G. Bratthall, Robert van der Geest, Holger Hofmann, Edgar Jellum Zbigniew Korendo, Robert Martinez, Michal Orkisz, Christian Zeidler, Johan S Andersson: Integrating Hundred's of Products through one Architecture – The Industrial IT architecture. ICSE 2002

[4] International Electrical Commision (IEC). IEC 1346-1. Industrial Systems, installations and equipment and industrial products - Structuring principles and reference designations, First Edition, 1996

[5] Requirements for a Web Ontology Language, hppt://www.w3c.org

[6] Ling Feng Manfred A. Jeusfeld Jeroen Hoppenbrouwers. Beyond Information Searching and Browsing: Acquiring Knowledge from Digital Libraries. INFOLAB Technical Report ITRS-008, Tilburg University, The Netherands, February 2001. http://citeseer.nj.nec.com/421460.html

[7]OMG. OMG Unified Modeling Language Specification Version 1.3. Technical report, Object Management Group Inc., March 2000

[8]W3C. XML Schema Part 0: Primer, 2001. http://www.w3.org

[9]Judith M. Myerson, Web Service Architecture, Published by Tect, 29 South LaSalle St. Suite 520, Chicago, Illinois, 60603, USA, http://www.webservicesarchitect.com.

[10] David Ehnebuske, Dan Rogers, Claus von Riegen, UDDI Version 2.0 Data Structure Reference UDDI Open Draft Specification 8 June 2001, http://www.uddi.org/pubs/DataStructure-V2.00-Open-20010608.pdf.

[11] Thomas R. Gruber. A Translation Approach to Protable Ontology Speccifications. Knowl-edge Acquisition, 5(2):199-220, 1993..

[12] Nicola Guarino, Pierdaniele Giaretta. Ontologies and Knowledge Bases: Towards a Termi-nological Clarification. In N.J.I. Mars (ed.), Towards Very Large Knowledge Bases, IOS Press 1995

[13] Guarino N., Masolo C., and Vetere G., OntoSeek: Content-Based Access to the Web, IEEE Intelligent Systems 14(3), May/June 1999, pp. 70-80

[14]Nicola Guarino. The Ontological Level. Invited paper presented at IV Wittenstein Sympo-sium, Kirchberg, Austria, 1993. In R. Casati, B. Smith and G. Whte (eds.), Philosophy and the Cognitive Science, Vienna, Hölder-Pichler-Tempsky, 1994.