A Pattern for Designing Distributed Heterogeneous Ontologies for Facilitating Application Interoperability

Moustafa Chenine[1] Vandana Kabilan[1] Marianela Garcia Lozano[2]

[1]Department of Computer and Systems Sciences,
Royal Institute of Technology and Stockholm University,
FORUM 100, Kista SE 164 40
[2]FOI, Swedish Defence Research Agency
Division of Systems Technology
Department of Systems Modelling
SE-164 90 Stockholm, Sweden
{Mchenine@kth.se, Vandana@dsv.su.se, Garcia@foi.se}

Abstract. The role of ontologies in knowledge base systems is gradually increasing. Along with the growth of Internet based applications and e-commerce, the need for easy interoperability between ontologies is paramount. Today methodologies and design guidelines for building and developing ontologies from scratch exist, and others have focused on evolving step-by-step growth of ontologies as shall be discussed in this paper. However, we find inadequate aid in the design of distributed, heterogeneous and multi-functioned, application ontology, primarily aimed to be the central hub for interoperability between a number of other applications which may or may not be ontology based. In this paper, we present a logical context based ontology design architecture in the form of Principle-Subject-Support(PSS) pattern. The PSS pattern has been used as a guide to analyze and model several perspectives involved in a practical case study carried out in a military network simulation project to build a distributed repository ontology (DRONT) for interoperability.

Keywords Ontologies, Interoperability, Ontology Architecture

1. Introduction

Interest in ontology has been on the rise in information technology, the concept has its origins (in respect to computer science) in the field of artificial intelligence. The most popular definition of what is an ontology is according to Gruber [5.]

"An ontology is an explicit specification of a conceptualization"

The use of ontologies is growing in a diverse range of applications and domains from enterprise systems, to patient health care, to e-governance, and now even in the realm of military modeling and simulations. But, in all the above cases, there persists a problem of interoperability between existing knowledge bases and newly developed

ontologies, existing ontologies and ontology to be designed, and finally between different applications using heterogeneous data sources and distributed in space and context. In this paper, we focus on this very issue of designing a distributed ontology based on heterogeneous and diverse domains and targeted for shared, integrated applications. We support reuse of existing methods and tools, as well as existing ontologies, as a way to establish interoperability. We propose a Pattern called the Principle Subject Support pattern (PSS) to analyze the heterogeneous domains into different contexts. We illustrate its usability through its application on a case study carried out at the Swedish Defence Research Agency (FOI)[1]. The rest of this paper is structured as follows:

In the following section we briefly summarize some of the state of the art ontology design and development methodologies and guidelines surveyed. In section 3 we present the proposed PSS Pattern for designing ontologies for interoperability as well as a discussion on its relation to existing software systems design architectures like the MDA and the Zachman Framework. In section 4 we discuss the case study and the results of applying the PSS to design the DRONT (Distributed Repository Ontology). However, due to space limitations we shall not go in to details of the DRONT conceptualizations per se, but restrict ourselves to the application of the PSS pattern in DRONT and illustrative examples from the DRONT conceptual models. Finally we conclude in section 5.

2. Ontology Design and Development Background

Ontology development methodologies suggest a process by which the identification and specification of ontology can be completed.

Four methodologies are discussed in this section, one that builds on the work on The Enterprise Ontology by Uschold and King [10.], The Gruninger Methodology that resulted out of the development of Toronto Virtual Enterprise (TOVE)[9.]. Uschold's[11.] also proposed another methodology seeking to unify the methodologies arising from Enterprise and TOVE projects and finally, we discuss the METHONTOLOGY methodology.

The Uschold and King methodology proposes 4 steps for the development of ontology [10.] .They propose, (1) Identifying the purpose which is important in order to set to establish the goal and aim of the intended ontology. (2) Build the ontology by capturing the knowledge, code this knowledge and integrate existing ontologies is available. Finally (3) evaluate this ontology and (4) document the ontology.

The Gruninger and Fox method [14.] (see figure 1 below) is based on building a logical model that will be specified into an ontology This is first done by (1) establishing motivating scenarios on ontology would serve, then, (2) informal competency question are formulated based on these scenarios that were established. (3) The specification of the terminology if the ontology in a formal language, this is based on the terms extracted from (2). The process then continues but this time with

[1] Swedish Defence Research Agency, www.foi.se

(4) formal competency question from which (5) the specification of axioms and the definition of the terms are formalized.

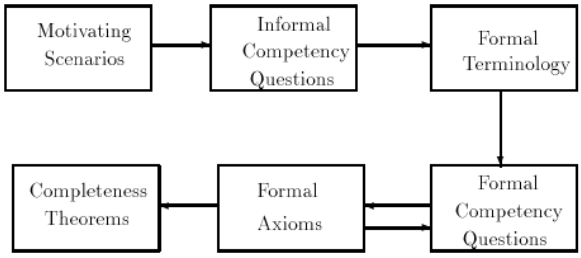

Fig. 1. The Gruninger and Fox Method

Uschold proposed a unified ontology based on the works of both TOVE and The Enterprise Ontology [10.] This methodology is based on Uschold and King Methodology but also integrates the Gruninger and Fox methodology [14.] in its steps. The main characteristics of this method are that it integrates the first two steps in Gruninger and Fox, (establish scenarios and informal competency questions) with the first step in Uschold and King. Steps (3), (4) and (5) from Gruninger and Fox [14.], t(Formal terminology, formal competency question and formal axioms) have been merged in to step (2) of the Uschold and King method [10.].

Finally, we discuss METHONTOGY framework, this method lays out a development process, a life cycle based on evolving prototypes and a techniques to carry out each activity in the process [3.] The development process is made up of five phases [4.]

- *Specification* is where the purpose, scope and stakeholders of the ontology are identified.
- *Conceptualization* is where the organization of acquired knowledge takes place. A conceptual model of the knowledge is represented in both tabular and graphical form.
- *Formalization*, transforms these models of the conceptualization phase in to semi-formal models, this is the intermediate stage, where the information can still be easily understood by domain experts.
- *Implementation*, based on the models produced in the formalization phases, the ontology is implemented in the desired knowledge representation language.
- *Maintenance*, the final phase where corrections are made to the ontology, if needed.

METHONTOLOGY caters for project management and support activities like planning, control, quality assurance knowledge acquisition, integration, evaluation. The methods life cycle identifies a set of stages that organizes the activities are to be performed [2.] .

As seen above the above surveyed methodologies and approaches each have their strengths and their weaknesses. While they concur on most of the design philosophies

they have some variations in approach. But with the exception of METHONTOLOGY, the others are predominantly useful in developing ontologies from scratch. These methodologies are also targeted at a single application and focused on single domain of discourse. We have adopted the design and development guidelines as proposed by METHONTOLOGY [13.] or Uschold and Gruninger [9.] as applicable to each of the different contexts for the distributed applications domain as shall be discussed later.

3. The Principle, Subject and Support Pattern

Application ontologies can be a specialization of domain or task ontologies or both as has been proposed by Guarino [6.] As seen in figure 2 below, Guarino has proposed the design of ontology architecture as top-level ontology (generic domain independent concepts), domain ontology (including the domain dependent concepts), a task ontology (which uses the top-ontology, but unlike the domain ontology focuses on activities or procedures) and finally Guarino proposes application ontology to be designed for the targeted usage by inheriting definitions from both the domain ontology and the task ontology. Thus the ontology design architecture as proposed by Guarino is based on the semantic conceptualization perspective alone.

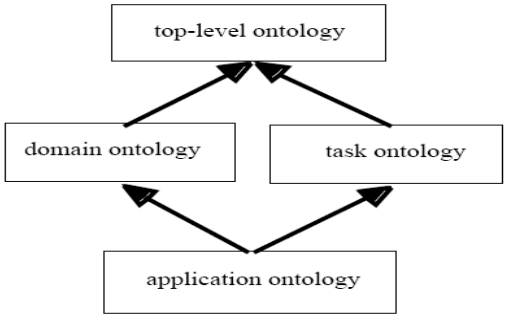

Fig. 2. Application Ontology Specialization as proposed by Guarino

The above design philosophy is suitable if we have a single domain of interest, or single homogeneous centralized application ontology to be developed. However, in reality we find that applications are usually distributed and their information and data not homogeneous. We visualize this as a fundamental obstacle in affecting interoperability at the application level. As the first step towards promoting interoperability amongst distributed ontologies, we need to have a structured architecture for designing and developing ontologies which takes in to consideration several perspectives other than the single domain, its associated tasks and thereby concentrating on only one application area. In this paper, we introduce the Principle, Subject and Support (PSS) Pattern as a design guideline aimed to help in the design of application ontologies specifically distributed application ontologies.

3.1 The Contexts Foundation

In order to bridge the gap between existing ontology design philosophies and the need for interoperability across heterogeneous domains, different technical platforms, various targeted user groups, we surveyed existing information systems design guidelines and architectures like the MDA [15] or the Zachman Framework. OMG's MDA proposes the three different viewpoints for facilitating portability and interoperability among information systems focusing only on the *structural organization of the software system* and not on the semantics of the system as:

- *Computation Independent Viewpoint* focuses only on the environment and requirements of the information system, structuring and processing is hidden.
- *Platform Independent Viewpoint* focuses on the operation of a system and focuses on a technology independent model.
- *Platform Specific Viewpoint* is the final platform and specific technology, application specific view.

On the other hand, the Zachman Framework [16] presents a *logical architecture* for classifying and describing enterprise information systems in to an intersection of six levels of perspectives based on the roles (users) to six levels of abstractions for the artifacts. The six contexts of abstractions from different users' (roles) perspectives are: Motivation (why), Function (How), Data (What), People (Who), Network (Where), and Time (When). The orthogonal axis has 6 levels of artifacts, namely – contextual, conceptual, logical, physical, as-built and functioning.

The usefulness of MDA and the Zachman Framework has been proved in the realm of designing information systems. Combining the concepts of *structural* and *logical perspectives* from both MDA and Zachman Framework and the *semantics perspective* from Guarino, we propose the PSS Pattern as a design guideline for developing heterogeneous ontologies.

We propose an identification of the different concepts from the distributed heterogeneous ontology domain by (a) its logical context type level and (b) by its conceptual structural type level. The PSS Pattern identifies seven basic logical areas and levels of knowledge that have to be captured.

The PSS pattern contexts, similar to the Zachman framework, analyze the domain of discourse into a logical classification of contexts as listed in table 1 below. Each of the contexts may be implemented as a conceptual model or an individual ontology which could interoperate with other contextual ontologies independently.

Context	Description
Generic View Context	Represents a common perspective on the core of the subject domain.
Standards Context	Captures different standards that exist and that might regulate the subject domain
Domain Context	Representation of concepts that exist in the subject from the view of the topic.
User Context	Captures concepts about the persons, access rights , organizations, roles etc.
Software Context	Represents concepts about types of software applications, and making distinctions between them. Also captures minimal

	information on hardware platform needed to specify software hardware requirements.
Process Context	Represents generic concepts needed to generally capture information on process and activities
Resource Context	Defines the resources that ontology implemented according to the Context pattern deals with.

Table 1. Knowledge Areas in the PSS Pattern

The contexts are further classified in to domain types called the Principle, the Subject and the Support.

3.2 Principle, Subject and Support Domains

The following table illustrates these domain types and the corresponding context in the PSS Pattern. Structurally, though these are similar to the MDA's CIM, PIM and PSM viewpoints, semantically these are dissimilar.

Domains Type	Definition	Corresponding Contexts
Principle	Core topic area or domain of the ontology.	Generic
		Standard
Subject	Topic area in which the core topic is applied on	Domain
		Software
Support	Topic area that covers domains that could support the Principle and/or the Subject	User
		Process
		Resource

Table 2. Domain Types and Contexts

We propose this classification for analyzing the conceptual domain of the proposed ontology. In fact the Principle domain is similar to the application ontology layer as proposed by Guarino and the Subject the domain ontology and task ontology layer, and finally the Support type is similar to the general ontology layer. The Principle domain is the core nucleus and can be further divided into two sub categories,

Dependent category captures the different perspectives and standards that describe the domain of interest, For example, national accounting standards like the German, American, British, have different standards that describe the domain of interest, accounting and finance.

Independent category captures the core concepts or the core knowledge of the domain of interest independent of perspectives and standard. For example, concepts such as balance sheet, income statements, cash flows, etc, would be captured independent of procedure, rules and regulations specified in national accounting standards.

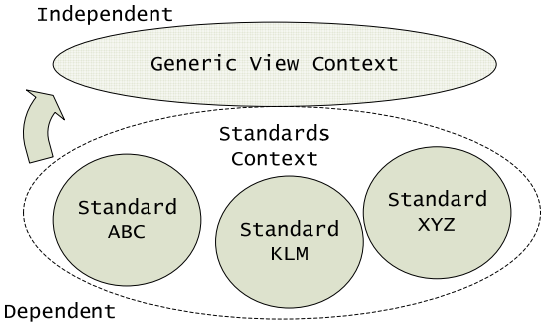

Fig. 3. Principle domain divided into Independent and Dependent Sub Categories

The Independent category contains the common essence of the perspectives and standards that exist in the Dependent category. Furthermore, concepts from the Dependent category map to the Independent category and *not* vice versa.

The Subject domain is that, on which the domain of interest is applied upon, i.e. it is in concepts of the Subject domain that concepts from the Principle domain relate to. For example, in military simulations where we are interested in simulations and its application in the military domain, then the military domain is the Subject and the simulation domain is the Principle. Finally, the Support domain type is not directly related to the Principle or Subject domains per se. The Principle and Subject and can exist independent of any Support domains, but the Support domain gives an extended view which links the Principle and Subject domains to the broader abstract domain or 'the real world'. For example the Process Context would in general, be a part of the process/ task domain. likewise, the User Context is also part of a wider domain of organization and users, both resembling established ontologies like the Enterprise Ontology [11.]

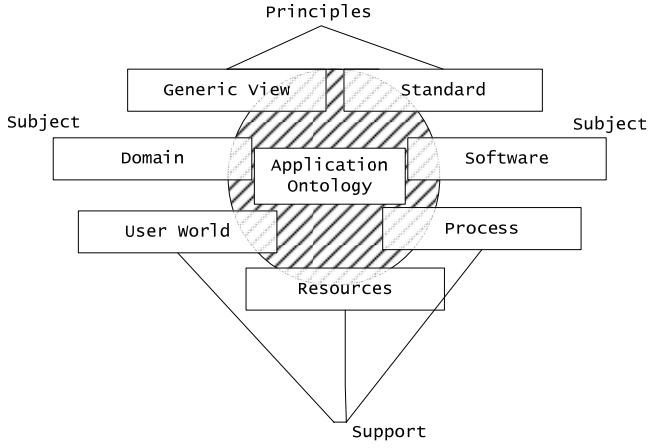

Fig. 4. PSS Pattern and domain divisions

The Resource Context is an exception. It is geared towards repositories or for systems that will actually share resources described by other contexts. The Resource Context should make minimal claim on domains, and in fact, resemble a raw hierarchy with minimal relations within the hierarchy. Figure 4 depicts the PSS Pattern's domain divisions, where Principle, Subject and Support are the three different axis and the seven context types are oriented in between these axis. As seen, we visualize the application ontology to be a specific combination of these seven context types.

4. The NETSIM Case Study

A simple definition of simulation as given by Banks [1.] states that

> "Simulation is the imitation of the operation of the real-world process or system over time".

Simulations duplicate real life phenomenon through the use of mathematical models. Simulations are therefore, models of a real life phenomenon that can be executed and observed on a computer system. Miller et al have used ontologies for simulation modeling [8.] Network Based Modeling and Simulation is a system being developed at the Swedish Defense Research Agency (FOI). The NetSim project aims to streamline the modeling and simulation process in its entire lifecycle, from conception to execution.

NetSim would cater for distributed modeling, collaboration and simulation composability over varied platforms and architectures. Users of NetSim can be located at diverse geographic locations, and can continue to collaborate as they design new simulation models. In addition, executed simulation would be composed of different smaller simulation models designed in different simulation standards, running on different platforms on geographically dispersed computers. Here thus not only the simulation domains are different ranging from military, geographical, biological sciences to war games, Also the visualized use of the proposed ontology is not only for simulations but also for a variety of other uses like C2I (Command and Control Interoperability).

One of the goals of this project has been to model ontology for interoperability between different standards, applications and contexts. As the first step, subject matter experts and domain experts from different fields were interviewed. Existing knowledge bases and targeted applications that are to interoperate were studied. Based on the above, a topic map for the visualized Distributed Repository ONTology (DRONT) was drawn up. Simultaneously literature study of current state of the art ontology design principles, methodologies and guidelines were studied as mentioned in section 2 above. Thereafter, using the proposed PSS pattern as a guideline an architectural map for DRONT was drawn up. Depending upon the specific requirements of the analyzed individual contexts appropriate design methodologies were chosen (from the surveyed ontology design methodologies). Thus, each context of the PSS pattern was implemented as independent set of conceptual models and formalized using OWL (using protégé tool).

4.2 Applying PSS for Distributed Repository Ontology (DRONT)

DRONT can be generally classified as an application or domain ontology, i.e. it is a specialized ontology dealing with a specific domain or application field. This domain can be generally labeled as the 'simulation and modeling domain'..DRONT captures concepts in the military simulation domain and those that are applied in the field of developing, sharing and maintaining these simulations. The following table illustrates these domain types and the corresponding context in the PSS Pattern.

Domains Type	PSS Pattern Contexts	DRONT Contexts
Principle	Generic	Simulation
	Standard	Simulation Standard
Subject	Domain	Military
	Software	Software
Support	User	User
	Process	Process
	Resource	Resource

Table 3. Domain Types and Contexts

In DRONT the Independent category is represented by the Simulation Context and the dependent category is represented by the Simulation Standard Context.

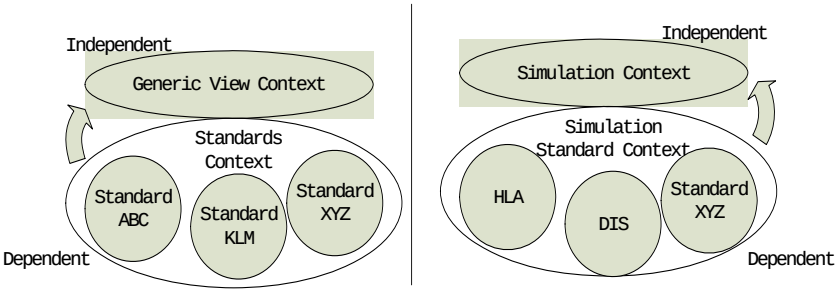

Fig. 5. Applying the PSS in the case of the NETSIM project for DRONT

Another example of the application of the Principle would be for example in accounting and finance situations. Assume a multinational cooperation with divisions in multiple countries. In essence, accounting and finance is the same in all countries. Neverthelss, every country may have a different standard on accounting and finance principles and procedures. In this case, the organization would model the core concepts shared among all divisions of the cooperation and place these in the Independent principle. Then the organization would model every division's definition and place them in the Dependent principle.

DRONT has two Subject domains, Military and Software. The simulations are related to Military domain concepts on one hand while at the same time they need to be mapped to the Software domain as well.

Using the multinational cooperation example again, the Subject domain would be the field of business the cooperation is involved in like, retailing or manufacturing.

Finally, we have the Support domains that are not directly related to the Principle or Subject domains, and there is little dependence on them. The support domains are drawn from other wider domains (like existing generic ontologies, knowledge bases, taxonomies, operating standards) and can be added according to the information the designers/ users want to expand on. For example the Process Context in DRONT is, in general, a part of the process/ task domain. The User Context is also part of a wider domain on organization and users, both resembling established ontologies like the Enterprise Ontology [11.]

The iterative sequence of tasks followed within the ontology development methodology for the application of the PSS Pattern is as follows. (1) identify the central topic that would be the Principle domain. (2) Identify the Independent and Dependent Principle and map the concepts of the Dependent onto the latter. (3) Identify the Subject domains and map and relate the concepts of the Principle domain to the Subject domains. (4) Set the scope of extended knowledge and capture it in the Support domains and map and relate the concepts of the Principle and Subject unto the Support. (5) Identify critical and exchangeable data and information and classify them as resources.

4.3. Illustrative Example from DRONT

As an illustrative example, we present an extract of one of the contexts in fig 6 below, from the proposed DRONT designed by applying the PSS pattern discussed above. Detailed discussion of the ontology itself is the subject of another paper and is out of scope for this paper due to space limitations.

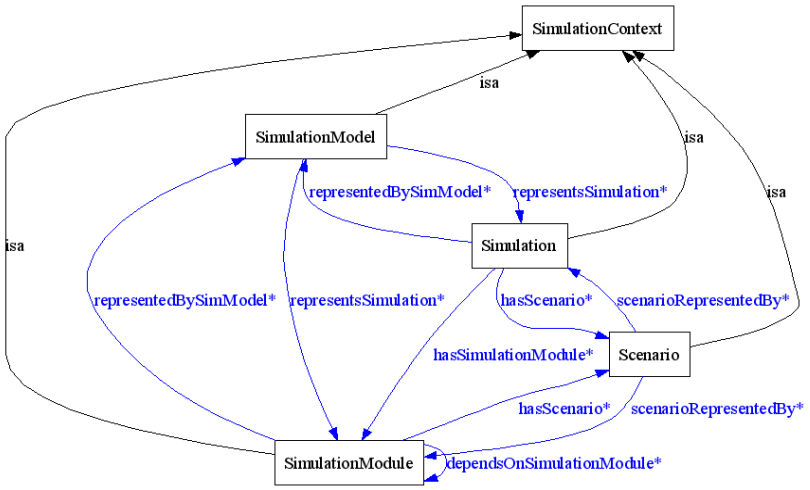

Fig. 6. Extract from the Simulation Context in DRONT

As discussed in table 3 above, the Generic Context of the PSS pattern for the military simulations domain is the Simulations Context. The simulation context takes

on the perception of simulations from a modular view independent of standards of implementation. To illustrate, consider a simulation that represents two aircrafts an F-16 and a MIG. The MIG has a missile of type X and a radar system of type Y, on the other hand, the F-16 has a missile of type B and a sensor system of type C. applying this to our ontology we can say that each module the fighter planes (the F-16 and the MIG) represent a simulation model each, and the each missile and sensor system are also simulation models.

An object modeled as a semantic concept in one context may have different semantics in another context. By interconnecting the different perspectives we get a composite model of each object being modeled. In other words, by describing the diverse definitions of the same object in different related contexts we can achieve semantic interoperability. Consider as an illustration a simulation of an aircraft object, an aircraft would exist in the Simulation Context as a *simulation* and in the Simulation Standard as a *federation* which is a subclass of *HLA.* In the Military Context the aircraft would be classified as a subclass of *MilitaryObject.* In the Software Context the aircraft would be represented as a *ComposedSimulation* which in turn is a subclass of *SoftwareApplication.* In the Process Context, depending on its importance and of focus, the aircraft would either be an *Artifact* of a *Project* or the *Product* of it, likewise, in the User Context where it would be associated with a *project*, *user* or *organization* or all. Further details and all the context models are available in [17].

5. Conclusion and Future Work

In this paper we have illustrated through the military simulations case study that in order to design an ontology for interoperability, it is not enough to consider the domain, tasks and application perspectives alone. One needs to consider a separation of contexts which are based on the functional requirements, technological considerations, knowledge representation formalisms, user requirements as well as design aspects. We have proposed a pattern based approach PSS to aid in the analysis of all applicable perspectives. As discussed in the previous section, the PSS pattern can be applied in other knowledge domains like accounting, enterprise systems etc. One of the strong points of our pattern is that it makes use of existing design methodologies as suitable for each ‘context’. Another feature is that by the dissociation in to different contextual layers, we segregate the semantics in to smaller packets of readily reusable, extendable knowledge. And finally, but not the last, the PSS pattern is a handy guideline for ontology engineers in their endeavor to design ontologies in a heterogeneous knowledge domain.

One future work would be the actual application of the proposed pattern in other case studies and other domains of discourse. Ongoing work is focused on the development of the DRONT and its integration in to the military simulations applications that already exist.

References

[1.] Banks, Jerry (ed.) (1998) Handbook of Simulation: Principles, Methodology, Advances, Applications and Practice, New York, John Wiley and Sons.

[2.] Blazquez, M., Fernandez M., Garca-Pinar J., and Gomez-Perez A., Building Ontologies at the Knowledge Level Using the Ontology Design Environment. Proceedings of the Knowledge Acquisition Workshop, KAW98, 1998**.**

[3.] Fernandez, M., Gomez-Prez, A., Pazos-Sierra, A., Pazos-Sierra, J. "Building a Chemical Ontology Using Methontology and the Ontology Design Environment". IEEE Intelligent Systems & their applications. January/February 1999. PP. 37-46.

[4.] Gomez-Perez, A., Corcho-Garcia, O., and Fernandez-Lopez, M. 2003 Ontological Engineering. Springer-Verlag New York, Inc

[5.] Gruber T. "Toward principles for the design of ontologies used for knowledge sharing", Report KSL 93-04, Stanford University, 1993.

[6.] Guarino N., "Semantic Matching: Formal Ontological Distinctions for Information Organization, Extraction, and Integration" Summer School on Information Extraction, Frascati, July 1997.

[7.] López, M.F., "Overview of the methodologies for building ontologies". In V. R. Benjamins, B. Chandrasekaran, A. Gómez-Prez, N. Guarino, and M. Uschold (Eds): Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods (KRR5), Stockholm, Sweden, August 1999.

[8.] Miller J., Baramidze G., Sheth A., Fishwick P., "Investigating Ontologies for Simulation Modeling," Proceedings of the 37th Annual Simulation Symposium (ANSS'04), Arlington, VA, April 2004, pp. 55-63.

[9.] Uschold M. and Gruninger M., "Ontologies: principles, methods, and applications", Knowledge Engineering Review, 11(2), 93--155, (1996).

[10.] Uschold M. and King M., *Towards a methodology for building ontologies*. Technical Report AIAI-TR-183, University of Edinburgh, July 1995.

[11.] Uschold, M., King, M., Moralee S., and Zorgios, Y., "The Enterprise Ontology", at http://www.aiai.ed.ac.uk/~entprise/enterprise/ontology.html, 1995.

[12.] Joint Command Control and Consultation Information Exchange Data Model Specification Available on Homepage of MIP. *http://www.mip-site.org/MIP_Specifications/Baseline_3.0/JC3IEDM-Joint_C3_Information_-Exchange_Data_Model/* Accessed: 2006-03-03

[13.] Fernandez, M., A. Gomez-Perez and N. Juristo (1997). METHONTOLOGY: From Ontological Art Towards Ontological Engineering Workshop on Ontological Engineering. Symposium on Ontological Engineering of AAAI, Stanford, California.

[14.] Gruninger, M. and M. S. Fox (1995). Methodology for the Design and Evaluation of Ontologies. IJCAI Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal, Quebec, Canada.

[15.] Model Driven Architecture, Design Guideline 1.01, OMG 2003. Available online http://www.omg.org/docs/omg/03-06-01.pdf

[16.] Zachman J. Zachman Framework . www.zifa.com (last accessed on 14th March 2006)

[17.] Chenine, Moustafa, 'Distributed Repository Ontology and a Design Pattern for Application Ontologies' , Master thesis, Submitted to Royal Institute of Technology, Stockholm, Sweden, 2006.