Conceptual study on Ontology based Application and its Specification in Information Science

Chauhan Vipul[1], Chauhan Falguni[2]

[1]Narmada College of Science and Commerce, Zadeshwar, Bharuch, Gujarat
Affiliated with VNSGUniversity, Surat, India
njoy_vipul@yahoo.com

[2]Shri Manilal Kadakia College of Management and Computer Studies, Ankleshwar, Bharuch, Gujarat
Affiliated with VNSGUniversity, Surat, India
falguni.chauhan7@yahoo.com

Abstract: ***Throughout the past decade, the notion of ontology has influenced research in many application areas including databases, information retrieval,* Biomedical, Bioinformatics, *electronic commerce, natural language processing, knowledge management, enterprise systems, systems analysis and design, the Web, and more. Ontology-Based Applications for Information Science and Knowledge Management provides an opportunity for readers to clearly understand the notion of ontology engineering, Knowledge Management Systems and knowledge-representation in Information Science. A perfect reference for researchers, scholars, postgraduate students, and practitioners, this study aims to gather the recent advances and research findings of various topics in ontology use for different application areas.***

Keywords: Ontology, Ontology-Based Application, Axiom, knowledge-representation

1. Introduction

This paper presents an Ontology based Application's Approach for Semantic Integration, knowledge representa tion in different information systems, which uses ontology as conceptual models for defining mappings between applications in three layers: data, service and process. In the ontology community very little research have been made in producing such patterns where the solution shows how architecture of ontology based software might look like. We describe in this paper about the framework, application patterns and Ontology-Based Application development model and he results of examining how upper ontology describes very general concepts that are the same across all knowledge domains.

With the large amount of applications developed that are based on ontology, there is a need to capture the essence of the solutions used to create good applications so that other developers can use this information to help them find and understand good solutions in this field. This can facilitate the creation of new software that is based on ontology. We are here tried to realize that ontology representing application domain knowledge are used for the development of modern information systems (IS). A number of authors believe that the use of such ontology, transformed and/or translated to IS components, help to 1) reduce the costs of a conceptual modeling [1] and 2) assure the ontological adequacy of the IS [1],[2],[3]; and allow to 3) share and reuse a domain knowledge across heterogeneous software platforms [2],[4], and 4) cognize of an application domain. If the IS is a traditional one, application domain knowledge will be just embedded in the standard components of the IS. If it is going to be an ontology-driven (or ontology-based) IS, then a separate component – application domain ontology – will be developed and included in the IS [1].

In the step of an IS conceptual modeling, researchers are challenged to transform application domain ontology to a conceptual data model, since their conceptualization of a real world is similar. Both see an application domain in terms of concepts, presenting entities of an application domain, relationships between concepts, properties of concepts and rules (in ontology axioms), presenting constrains of an application domain. While a number of approaches and methods for the transformation of application domain ontology to a conceptual data model have been proposed, like [3], [5]-[8] etc, there is lack of a formal theory and methods of ontology components transformation to application domain rules.

2. Purpose and Scope

The purpose of this paper is to find a definition of Ontology and examine how these Ontology Based Applications relate to other types of software Applications. These could help the development of software by letting developers understand solutions to general problems in this field. Since these are high-level patterns, the solutions are usually architectures of applications and by understanding them, designing architectures can be facilitated. Another purpose of this paper is to examine what the benefits of type of system.

The scope of this Paper is to investigate what Ontology Based Applications really are and how they relate to other types of softwares.

3. Theoretical background

Recently, ontology is expected to contribute to knowledge sharing and reuse. It is, however, difficult to develop a well-organized ontology because the principles of ontology design are not clear enough. Therefore, a methodology for ontology design and a computer system supporting for ontology design are needed.

[4] Ontology

The word ontology was first used in philosophy and means "The metaphysical study of the nature of being and existence". In more plain words: The study of how do thoughts, words and things really relate to each other. The meaning that the word has in computer science is derived from the philosophical meaning [9].

Probably the most common definition of ontology in the field of computer science is "a formal, explicit specification of a shared conceptualization" [10]. Conceptualization is an abstract model of how people think and when this model is given an explicit specification, we give names and meanings to the concepts and relations present in this abstract model. When the specification is formal, the meaning is that a language is used that have well understood formal properties so no ambiguities that natural language tend to give exist. This usually means some kind of logic based language is used.

In computer science and information science, ontology formally represents knowledge as a set of concepts within a domain, and the relationships between pairs of concepts. It can be used to model a domain and support reasoning about entities.

In theory, an ontology is a "formal, explicit specification of a shared conceptualization"[10] an ontology renders shared vocabulary and taxonomy which models a domain with the definition of objects and/or concepts and their properties and relations.

Ontology is the structural frameworks for organizing information and are used in artificial intelligence, the Semantic Web, systems engineering, software engineering, biomedical informatics, library science, enterprise bookmarking, and information architecture as a form of knowledge representation about the world or some part of it. The creation of domain ontology is also fundamental to the definition and use of an enterprise architecture framework.

Why is ontology important in the first place? What the ontology provides is a specification of the concepts in the domain without the ambiguities that natural language gives. Since the meaning of the concepts and relations in ontology are specified, these ambiguities are removed. This gives the users of the ontology, humans and computers, a shared vocabulary both syntactically and semantically [11].

In ontology concepts and relations between concepts are defined. Rules for how these concepts may be related are also defined. Concepts are anything that it is possible to say something about. It can be something physical and real but it can also be something abstract and fictitious. When a concept is defined, what is really done is to create a meaning that this particular concept stands for in the domain that is being described. To make it possible to refer to a concept, it is necessary to put a label on it, i.e. to connect a term to the concept. A domain is the area that are of interest to the parties involved in developing the ontology. This can be information concerning an organization or what wines that is produced in France [9].

The concepts usually have attributes that describes them. For example the concept "person" usually has an attribute "name" and perhaps "phone number" and "address". Attributes are sometimes called slots or properties [9].

Between concepts in the ontology exists relations. Perhaps the most usual relation is "is a" which is used to describe a hierarchy of concepts. In a hierarchy there are subclasses which inherit the attributes that the parent concept has. This is generally called taxonomy and it is very common that ontology is structured this way although it is not a necessity [11].

3.2 Ontology components

As mentioned above, most ontology describes individuals (instances), classes (concepts), attributes, and relations. In this section each of these components is discussed in turn.

- Common components of ontology include:
- Individuals: instances or objects (the basic or "ground level" objects)
- Classes: sets, collections, concepts, classes in programming, types of objects, or kinds of things
- Attributes: aspects, properties, features, characteristics, or parameters that objects (and classes) can have
- Relations: ways in which classes and individuals can be related to one another
- Function terms: complex structures formed from certain relations that can be used in place of an individual term in a statement
- Restrictions: formally stated descriptions of what must be true in order for some assertion to be accepted as input
- Rules: statements in the form of an if-then (antecedent-consequent) sentence that describe the logical inferences that can be drawn from an assertion in a particular form
- Axioms: assertions (including rules) in a logical form that together comprise the overall theory that the ontology describes in its domain of application. This definition differs from that of "axioms" in generative grammar and formal logic. In those disciplines, axioms include only statements asserted as a priori knowledge. As used here, "axioms" also include the theory derived from axiomatic statements
- Events: the changing of attributes or relations

3.3 Domain ontology and upper ontology

A domain ontology (or domain-specific ontology) models a specific domain, which represents part of the world. Particular meanings of terms applied to that domain are provided by domain ontology. For example the word card has many different meanings. An ontology about the domain of poker would model the "playing card" meaning of the word, while an ontology about the domain of computer hardware would model the "punched card" and "video card" meanings.

As mentioned before, only concepts that are relevant to the domain of interest are defined in the ontology. Such ontology is called domain ontology. Some concepts are more general and might be present in all ontology. One idea how to avoid that these concept have to be defined over and over is to introduce top-level ontology or upper level ontology. In this ontology general concepts that can be reused are defined. When new ontology is constructed the general concepts from this ontology are included. At this moment some ontology that could function as top-level ontology exists but none of them are used as some sort of standard. An example of a suggested top-level ontology is the Suggested Upper Merged Ontology developed by Standard Upper Ontology Working Group [12]. There are also ideas concerning domain dependent top-level ontology that have concepts that are general in a specific domain. This would also limit the work when new ontology is constructed since a top-level ontology constitutes the foundation for the ontology being built [11]. If the operational data is included in the ontology, i.e. there are individual instances of concepts in the ontology; a knowledge base has been created. So in the ontology about wines the concepts of wine with the subclasses red and white wine are defined. If an extra level where there are instances of a lot of individual red and white wines exist, this ontology constitutes a knowledge base [11].

When ontology is constructed within an organization it also makes the information concerning the domain more unambiguous since the task of structuring the information demands strict definition of the concepts. This means that the information can be reused and assumptions that have been made about the domain will be explicit, i.e. the process of creating ontology not only structures the information but also clears up misunderstandings and identifies gaps in the information [13].

Although ontology has been proven to be very useful there are some problems and issues that have to be taken into concern when using them. The process of classifying concepts is inherently problematic since not everyone has the same idea of what a certain concept refers to. When ontology is developed this is one of the big problems to overcome. The conceptualization must be shared among its users and this is problematic if the users are a very heterogeneous group. This “common idea” of what a concept stands for is called the ontological commitment. There is also the issue of maintaining the ontology. The stored information must continually be changed to mirror the real world or the ontology will be useless [13].

An upper ontology (or foundation ontology) is a model of the common objects that are generally applicable across a wide range of domain ontology. It employs a core glossary that contains the terms and associated object descriptions as they are used in various relevant domain sets. There are several standardized upper ontology available for use, including Dublin Core, GFO, OpenCyc/ResearchCyc, SUMO, and DOLCE. WordNet, while considered an upper ontology by some, is not strictly ontology. However, it has been employed as a linguistic tool for learning domain ontology. The Gellish ontology is an example of a combination of an upper and domain ontology.

Since domain ontology represents concepts in very specific and often eclectic ways, they are often incompatible. As systems that rely on domain ontology expand, they often need to merge domain ontology into a more general representation. This presents a challenge to the ontology designer. Different ontology in the same domain can also arise due to different perceptions of the domain based on cultural background, education, ideology, or because a different representation language was chosen.

At present, merging ontology that are not developed from common foundation ontology is a largely manual process and therefore time-consuming and expensive. Domain ontology that use the same foundation ontology to provide a set of basic elements with which to specify the meanings of the domain ontology elements can be merged automatically. There are studies on generalized techniques for merging ontology, but this area of research is still largely theoretical.

3.4 Ontology languages

An ontology language is a formal language used to encode the ontology. There are a number of such languages for ontology, both proprietary and standards-based:

Common Algebraic Specification Language is a general logic-based specification language developed within the IFIP working group 1.3 "Foundations of System Specifications" and functions as a de facto standard in the area of software specifications. It is now being applied to ontology specifications in order to provide modularity and structuring mechanisms.

Common logic is ISO standard 24707, a specification for a family of ontology languages that can be accurately translated into each other.

The Cyc project has its own ontology language called CycL, based on first-order predicate calculus with some higher-order extensions.

- DOGMA (Developing Ontology-Grounded Methods and Applications) adopts the fact-oriented modeling approach to provide a higher level of semantic stability.
- The Gellish language includes rules for its own extension and thus integrates ontology with an ontology language.
- IDEF5 is a software engineering method to develop and maintain usable, accurate, domain ontology.
- KIF is syntax for first-order logic that is based on S-expressions.
- Rule Interchange Format (RIF) and F-Logic combine ontology and rules.
- OWL is a language for making ontological statements, developed as a follow-on from RDF and RDFS, as well as earlier ontology language projects including OIL, DAML, and DAML+OIL. OWL is intended to be used over the World Wide Web, and all its elements (classes,

properties and individuals) are defined as RDF resources, and identified by URIs.
- Semantic Application Design Language (SADL) captures a subset of the expressiveness of OWL, using an English-like language entered via an Eclipse Plug-in.
- SBVR (Semantics of Business Vocabularies and Rules) is an OMG standard adopted in industry to build ontology.
- OBO, a language used for biological and biomedical ontology.
- (E)MOF and UML are standards of the OMG

3.5 Ontology dimensions

To be able to describe a whole ontology there is a need for some sort of characteristics of the ontology. These characteristics can also be called the dimensions of the ontology. A try to formulate such characteristics has been done in [18] and the dimensions described there are:

Level of Authoritativeness: This is a measure of how authoritative the ontology is of the area it describes. If the author of the ontology is the organization that is responsible for specifying the conceptualization, then the ontology might define the knowledge in the area, then this is clearly a highly authoritative ontology [18].

Source of Structure: If the ontology is developed externally from the application that will use it and changes are made systematically are called transcendent, while ontology where the structure comes directly from the application that will use the ontology are called immanent. This means that the structure might change radically depending on what happens in the use of the application [18].

Degree of Formality: Degree of formality refers to the level of formality of the specification of the conceptualization; this could be from highly informal or taxonomic ontology, to semantic networks that may include complex subclass/superclasses relations but no formal axiom expressions, to highly formal ontology that include axioms that explicitly define concepts [18].
Model Dynamics: This concern the rate of how changes in the ontology are made. From the extreme where the ontology is stable and rarely or never changes, to very volatile ontology that changes very often [18].

Instance Dynamics: This dimension is closely related to Model Dynamics but concerns the instances of the ontology [18].

Control / Degree of Manageability: This dimension considers who decides when and how much change to make to ontology. One extreme is that the author of the ontology has the sole decision on changes, and the other extreme of course is that the ontology must change based on outside parties. This is called internal and external focus [18].

Application Changeability: The applications that use the ontology might be on one extreme developed only once and on the other dynamically during run time [18].

Coupling: This dimension describes how closely coupled applications committed to shared ontology are to each other. The applications in an e-commerce exchange are tightly coupled, since they must interoperate at run time. At the other extreme, applications using the Periodic Table may have nothing in common at run time. They are loosely coupled, solely because they share a component [18].

Integration Focus: This dimension describes the focus of the ontology concerning integration. One extreme is the ontology that specifies the structure of interoperation but not the content. This is called application integration. The Other extreme is the information integration where the structure of the information is described [18].

Lifecycle Usage: In some cases the ontology is only used in the specification or design of an application but is never used during run time. The other extreme is for example that every message sent in an application is verified so it conforms to ontology.

4. Ontology Application

There are a lot of applications that uses ontology in some way. In [20] a classification of four ontology application scenarios is made:

Neutral authoring: In this scenario the ontology is authored in a single language and is converted into an appropriate form for each system that uses is. The benefits of this approach are knowledge reuse, improved maintainability and long term knowledge retention [20].

Ontology as Specification: Ontology of a given domain is created and used as a basis for specification and development of some software. Benefits of this approach include documentation, maintenance, reliability and knowledge (re)use [20]. Common Access to Information: Information is required by one or more persons or computer applications, but is expressed using unfamiliar vocabulary, or in an inaccessible format. The ontology helps render the information intelligible by providing a shared understanding of the terms, or by mapping between sets of terms. Benefits of this approach include inter-operability, and more effective use and reuse of knowledge resources [20].

Ontology-Based Search: Ontology is used for searching an information repository for desired resources (e.g. documents, web pages, names of experts). The chief benefit of this approach is faster access to important information resources, which leads to more effective use and reuse of knowledge resources. [20] In [21] four scenarios for ontology based applications are described. These are made from a business perspective, i.e. the scenarios are based on a business area, not on what technology that is used within the applications.

These scenarios are:

Corporate Intranet and Knowledge Management: This scenario concerns knowledge management. In the past, knowledge management has focused on management of

knowledge stored within text documents. In the future, the possibility to use ontology to specify a shared conceptualization of an application domain and in this way provide a foundation to define metadata that have a precisely defined semantics and are machine-process able give the possibility to create solutions that are based on semantically grouped information [21].

E-Commerce: Electronic Commerce is based on the exchange of information between stakeholders using some sort of communication infrastructure. [21] define two scenarios within this scenario, Business-to-Customer (B2C) and Business to Business (B2B). The B2C applications enable customers to find offers that meets their demands and service providers the possibility to reach their customers. The B2B applications make it possible for companies to exchange information concerning services already agreed upon or perhaps investigating new business proposals between the companies. In the past the information exchange has been more or less strictly of the first type but adding ontology to the systems may give the possibility to make the information exchange more powerful and less static which could give for example the possibility to have the B2B-applications find appropriate partners. [21]

Information retrieval: These types of applications are designed to find relevant information that meets the information demand of a user. The ontology is here used to guide the search so that the application returns more relevant results. [21]

Portals and Web communities: Portals are websites that provide information on a semantic basis. These portals usually have a backbone consisting of a knowledge warehouse, i.e. the ontology and the knowledge base and an inference mechanism. The system also has a front end that the users can interact with. These users may be other applications such as software agents. The human users can both be general users that only can access the data or community users which can contribute data. [21]

Examples of applications from the group Common access to data are KRAFT, InfoSleuth, DOME, Toronto Virtual Enterprise (TOVE) and MOMIS. KRAFT is an agent based system that enables integration of heterogeneous data sources. In the system there are three types of agents; wrappers, mediators and facilitators. Wrappers are proxies for the data sources and user agents. The wrapper is essentially a translator between the communication language used in the KRAFT system and the language used in the data source. The mediators are internal knowledge processing agents in the system. Typical tasks for these agents are filtering, sorting, and fusing knowledge obtained from other agents. The mediators are the matchmakers that enable the communication between the other agents in the system.

Each data source has a local ontology that describes the information stored in the source. The concepts and relations in this ontology are mapped to concepts and relations in a shared ontology .When a message is sent, the content of the message are terms that are defined in the shared ontology [16].

The InfoSleuth system is pretty similar to the KRAFT system. Each resource has a Resource agent that connects the source to the system. The resource agent maps the information in the source to an ontology. There is usually several ontology defined in the system. The resource agent then can translate queries that are written in the format described by the ontology to the internal format used in the data source. The users interact with the system using a user agent that translates the queries made by the user to a format described by the appropriate ontology. The system also has other agents that are necessary to find the appropriate resources and make advanced queries possible.

In the DOME approach three types of ontology are used; resource ontology, shared ontology and application ontology. The resource ontology specifies the structure of the content in a data source. The shared ontology is general ontology over the domain and these are specialized to application ontology which describes the domain for a certain application or group of users. The resource ontology is mapped to the application ontology and this makes interoperability within the system possible.

The MOMIS approach does not rely on mapping a local ontology to a shared one. Instead the general idea is to integrate all data sources into a single ontology that are called the common thesaurus which can be used to search and collect the information. The shared ontology is built from the schemas over the information that is present. MOMIS is intended to integrate structured or semi structured information, not unstructured like web pages. The integration is semi automatic. Each data source has a wrapper that connects the source to the system and acts like a translator. A mediator connects the wrappers to the shared ontology and the users interact with the ontology to collect information.

The approach of having a single shared ontology to integrate the data sources is also used in TOVE, where one ontology has been developed to provide a shared vocabulary for applications to use to understand each other. In this system there are also agents that act as a translator between the applications and the ontology.

Almost all ontology based application uses some sort of query based information retrieval. Using ontology in information retrieval systems gives the obvious benefit of higher precision, using context based searching. However, it is also possible to deal with other common issues in information retrieval systems; information quality and user adaptation.

Information quality determines the quality by answering a few questions. Is the information up to date? Are there any different versions of the information? Do it exist any conflicting information?

Most users or groups of users use different vocabularies. Also, they usually want a different level of specialization of the information retrieved. This can be called user adaptation.

With user ontology this can be set with different templates, and the ontology can also adapt with user interaction. The information retrieval component in InfoSleuth uses user agents to adapt to different users vocabulary.

5. Ontology and Information Science

5.1 Ontology and Information System

Two main directions of this branch may be defined. One is about developing of application domain ontology and other is about using ontology for the development of IS. The first one is analysed in ontology engineering field and is not going to be discussed in this paper.

According to [1], every IS has its own ontology, since it ascribes meaning to the symbols used according to a particular view of the world. N. Guarino [1] distinguishes two orthogonal dimensions in IS: a temporal dimension, concerning whether an ontology is used at development time or at run time, and a structural dimension, concerning the particular way an ontology can affect the main IS components, like application programs, information resources like databases and/or knowledge bases, and user interfaces.

In this paper, the main attention is placed on the usage of ontology at IS development. One of the major trends in this context is using ontology for conceptual data modelling, since a conceptual data model and an ontology both include concepts, relationships between them and rules (in ontology – axioms).

However, it is typically the case that in ontology-based conceptual data modelling approaches the process of developing domain rules is not defined in a formal manner.

Ontology defines the basic concepts, their definitions and their relationships comprising the vocabulary of an application domain and the axioms for constraining relationships and interpretation of concepts [31]. Some authors, like [32], also distinguish properties from concepts. In the simplest case [1], application domain ontology describes a hierarchy of concepts related by particular relationships (e.g., is-a, part-of, etc). In more sophisticated cases, constraints are added to restrict the values of concepts and relationships, like cardinality constraints, possible length, etc. In the most sophisticated cases, suitable axioms are added in order to express and restrict complex relationships between concepts and to constrain their intended interpretation.

In mathematics [33], an axiom is any starting assumption from which other statements are logically derived. It can be a sentence, a proposition, a statement or a rule that enables the construction of a formal system. Axioms cannot be derived by principles of deduction, because they are starting assumptions.

Following the terminology used in [32] and [34], axioms in ontology can be classified as epistemological, consolidation, and derivation axioms. Epistemological axioms are defined to show constraints imposed by the way concepts are structured. These include all axioms which can be directly included by the use of modelling primitives and relations that are used in a structural specification of ontology (e.g., is-a relation, part-of relations, cardinality constraints). An example of epistemological axioms imposed by the most basic form of a part-whole relation is: if exists x and y and x is a part of y, then y is not a part of x ($\forall x,y\ partOf(x,y) \rightarrow \neg partOf(y,x)$). Consolidation axioms impose constraints that exclude unintended interpretations over the structure of the ontology specification. An example of the consolidation axiom from a software quality ontology presented in [35] is: if a product quality characteristic (qc) is decomposed in sub characteristics (qc1), then these sub characteristics should also be a product quality characteristic ($(\forall qc,qc1)(subqc(qc1,qc) \wedge prodqc(qc) \rightarrow prodqc(qc1))(C1)$). Finally, derivation axioms allow new knowledge to be derived from the previously existing knowledge represented in the ontology. Typically, derivation axioms are created in order to derive information which can be used to answer the ontology competence questions. An example of a derivation axiom from [35] states that "if there is not a paradigm to which a quality characteristic qc is applicable, than qc is paradigm-independent"

$$(\forall qc) \neg (\exists p)\ (applicability\ (qc, p) \rightarrow pdgInd\ (qc))$$

If it is necessary, the fourth type of axioms can be defined in addition. They are definitional axioms that define the meaning of concepts in ontology.

According to [36], implementation of axioms in ontology modelling environments is:

• restricted in a framework of a description logics [37] or in some kind of logic language, like Knowledge Interchange Format (KIF) [38] in Protégé ontology [39] and SUMO [40], or

• axiom modelling is completely neglected in WordNet [41], which can be used as a lexical ontology, Protégé ontology (not all), ontology presented by [42] and [43], DBpedia [44].

This situation is detrimental to the modelling of large-scale ontology, because it aggravates engineering and maintenance of large sets of axioms.

N. Guarino, S. Staab and A. Maedche propose using of objects and categories to represent axioms. They state that categorisation of axioms allows representing the semantics of axioms, and specifying axioms like objects provides a compact, intuitively accessible representation.

C. S. J. Hou, N. F. Noy, M. A. Musen attempt to reduce the difficulty of writing axioms by identifying groups of axioms that manifest common patterns creating templates that allows users to compose axioms by "filling-in-the-blanks". The method for collecting the templates is also presented in. This method is implemented in Protégé ontology development and management tool.

E. Sirin and J. Tao inspired of growing usage of OWL analyse the possibilities of defining integrity constraint semantics for OWL axioms. Authors implement the proposal in the prototype using Pellet. Authors show that integrity constraints validation can be reduced to SPARQL (Query Language for RDF) query answering using off-theshelf reasoning. They state that the obtained results show that the goal of using OWL both as a knowledge representation and constraint language for data validation can be achieved without too much effort.

The analysis of ontology development tools, like Protégé, and ontology, like SUMO, from the implementation perspective shows that epistemological axioms are implemented by structuring concepts in an ontology; consolidation and derivation axioms are not distinguished and they are implemented using some languages suitable for this purpose, like Protégé Axiom Language (PAL) or Ontology Web Language (OWL). Some consolidation and definitional axioms are implemented by restricting definition of concepts in a particular ontology.

5.2 Ontology Axioms in Comparison with Application Domain Rules

Here we present differences between ontology axioms, application domain rules, information processing rules, and executable rules, expressed in the form of event condition action (ECA) rules. This comparison is necessary to define a correct mapping of ontology axioms to application domain rules.

The IS level rules are statements that define information processing rules using a rule-based language, like OCL, etc. They are taken from the business system level and implement application domain rules. Information processing rules should be precise and expressed as ECA rules to be implemented by executable rules. Therefore, it is necessary to develop ECA rules, which define when the rule should be applied, what should be checked and what to do after checking.

Application domain ontology axioms belong to a particular application domain. They define admissible states of a domain. In particular cases axioms can have conditions under which defined states should be taken.

According to this comparison, the following conclusions could be done. Since axioms can be formalised together with a domain ontology using a particular language, it is reasonable to use this formalisation to automatically transform the ontology axioms to information processing rules or even to executable rules.

Protégé axioms and axioms from [35] and [46] were analysed and it was determined that consolidation and derivation axioms have structure state or condition-state.

A state axiom clearly defines a state in which a domain should be and which can be transformed to the condition of an ECA rule. An action can be understood in two ways:

if the condition is satisfied, then the transition from one state of the system to another is admissible;
if the condition is not satisfied, then the transition is forbidden.

An example of a state axiom, defined by PAL, is presented as follows. It constrains that the number of pages in a newspaper should not exceed 30. This axiom defines a possible state of a newspaper in a domain, i. e. it defines that for all instances of a class newspaper an attribute number_of_pages should not exceed 30.

defrange ?Newspaper :FRAME Newspaper forall ?Newspaper
(> (number_of_pages ?Newspaper) 30))
A condition-state axiom defines an admissible state of a domain under the defined condition. In the sense of the ECA structure, a condition-state axiom can be transformed into an ECA rule in two ways:

the condition of an axiom is transformed to the condition of an ECA rule, the state of an axiom is transformed to the action of an ECA rule;
the condition-state axiom is transformed to an ECA rule as in the case of a state axiom.

An example of a condition-state axiom, defined by PAL, is presented as follows. It constrains that only finished Content (an article or an advertisement) can be included in a Newspaper. This axiom defines a possible state of a newspaper under the defined condition, i. e. it defines that content (an article or an advertisement) can be included in a newspaper. However, it should satisfy a condition – it should be finished.

defrange ?Content :FRAME Content
defrange ?Content-SlotVal :FRAME Content 'published_in'
forall ?Content (forall ?Content-SlotVal
(=> (not('isFinished' ?Content \"must contain\"))
(instance-of? Content-SlotVal Newspaper)))

Axioms hold in a domain in all cases. However, computer systems should have information when they apply rules. Therefore, according to the structure of an ECA rule, it is necessary to define important events and link them with corresponding rules during the transformation of ontology axioms to information processing rules or executable rules.

6. Developing Ontology-based Applications using Hozo

An environment for building/using ontology, named Hozo, based on both of a fundamental consideration of an ontological theory and a methodology of building an ontology. Since Hozo is based on an ontological theory of a role-concept, it can distinguish concepts dependent on particular contexts from\ so-called basic concepts and contribute to building reusable ontology.

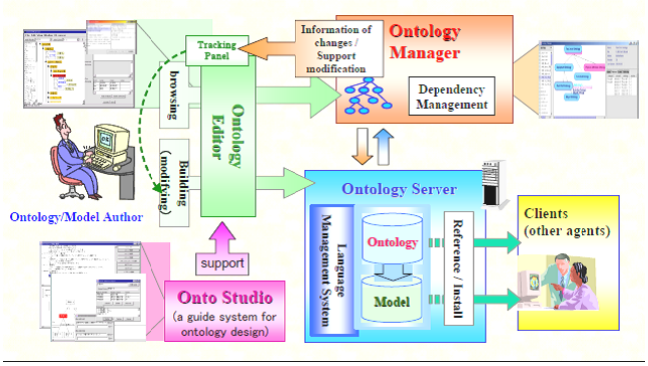

Figure 1. Hozo: an Environment for Building/Using Ontology

6.1 Outline of Hozo

Hozo is composed of four modules: Ontology Editor, Ontology Manager, Ontology Server and Onto-Studio (Figure.1).

Ontology Editor provides users with a graphical interface (Fig.2), through which they can browse and modify ontology by simple mouse operations. Its feature is the functionality to treat "role concept" and "relation" on the basis of fundamental consideration.

Ontology Manager helps users for distributed ontology development. It manages ontology as components of a target ontology-based on their dependencies. And when a change of some ontology influences others, it supports modification of influenced ontology for keeping its consistency.

Onto-Studio is based on a method of building ontology, named AFM (Activity-First Method). It helps users design ontology from technical documents. It consists of 4 phases and 12 steps.

Ontology Server manages the storage and use of ontology and models. Models are built by choosing and instantiating concepts in the ontology and by connecting the instances by defining specific relation among them. Hozo also checks the consistency of the model using the axioms defined in the ontology. The latest version of this ontology editor is published at the URL: http://www.hozo.jp.

6.2 Ontological theories of Hozo

Several ontology development environments have been already developed. Most of the tools are based on a frame-based knowledge representation language with an additional functionality for writing axioms. Hozo is similar to them in that sense, but is different from them in some respects:

1. Clear discrimination among a role-concept (husband role), a role-holder (husband) and a basic concept (man) is done to treat "Role" properly.
2. Management of the correspondence between a wholeness concept and a relational concept.

In this section, we outline these ontological theories.

1) Basic concept, role concept and role holder

John Sowa introduces the firstness and the secondness of concepts. The former is roughly defined as a concept which can be defined without mentioning other concepts. Examples include ion, a man, a tree, etc. The latter is roughly defined as a concept which cannot be defined without mentioning other concepts. Examples include wife, husband, student, child, etc. We call concepts of the secondness type except artifacts role-concepts in this paper. Based on his theory, we identified three categories for a concept. That is, a basic concept, a role-concept, and a role holder.

A role-concept represents a role which a thing plays in a specific context and it is defined with other concepts. On the other hand, a basic concept does not need other concepts for being defined. An entity of the basic concept that plays a role such as husband role or wife role is called a role holder. For example in "a bicycle", its wheel plays the role as a front wheel ("a front wheel role") or a role that steers its body ("a steering role"), which is defined as a role-concept. A wheel that plays these roles is called "a front wheel" and "a steering wheel", respectively, which are role holders.

In ontological theory the role concept is very important and discussed by many researchers. We discuss how to organize role concepts and suggest a framework for organizing role-concepts in their hierarchy according to their context dependencies.

2) Relational concept and wholeness concept

There are two ways of conceptualizing a thing. Consider a "brothers" and a "brotherhood". "The Smith brothers" is a conceptualization as a concept, on the other hand "brotherhood between Bob and Tom" is conceptualized as a relation. On the basis of the observations that most of the things are composed of parts and that those parts are connected by a specific relation to form the whole, we introduced "wholeness concept" and "relational concept".

The former is a conceptualization of the whole and the latter is that of the relation. In the above example, the "brothers" is a wholeness concept and the "brotherhood" is a relational concept. Because a wholeness concept and a relational concept are different conceptualizations derived from the same thing, they correspond to each other. Theoretically, every thing that is a composite of parts can be conceptualized in both perspectives as a wholeness concept and a relational concept.

6.3 Creating Ontology-based Applications

Hozo provides several functions to develop applications based on ontology and instance models which are built by its Ontology Editor. It helps the users to develop ontology-based applications by the following two ways.

1) Export of ontology and models into different formats for another application

Hozo can translate the ontology and models into different formats/languages (hierarchical text, XML/DTD, DAML+OIL, RDF(S), and OWL) that make them portable and reusable. Users develop their own applications to import these ontology and models utilizing existing tools which speak these languages. Hozo is used as a program to build and manage data of ontology and models for the applications consistently. The use process is supported by its functionality, such as building ontology and models, dependency management of them, and checking the consistency of the model using the axioms defined in the ontology.

2) Access using HozoAPI

The operational functions which Hozo provides are open to the public as API implemented in Java, it is named HozoAPI. Using the API other systems can use some functionalities of Hozo. It has about 30 functions, necessary for applications to use ontology and instance models. The users can implement their systems easily to use these basic functions for operation of ontology and models.

6.4 Implementations using Hozo

We developed the system using Hozo with the following steps:

1) Experts of nanotechnology built a preliminary ontology of nanotechnology based on keywords which are extracted from textbooks, papers and patents. We call the ontology "General Index". It is constructed from the name of concepts in an is-a hierarchy. The number of the concepts is about 2,300.

2) We inputted the data of General Index into Hozo and made it to be accessible through internet. Experts accessed it and add "link information" between concepts in General Index and these in their resources in the platform using Ontology Editor of Hozo. The "link information" consists of its name, the kind of link and the hyper link to the resources. They are managed by Ontology Server of Hozo and edited by different experts in a distributed environment.

3) Hozo exports General Index with "link information" in a simple hierarchical XML format.

4) The system to show General Index read the exported XML data and shows the hierarchy of General Index in web-browser. Figure 4 shows a snapshot of the system. General Index is presented in a tree-format on the left side, and the user can brows its hierarchy by mouse operation. When he/she selects a concept in it, then the system shows "link information" edited by experts and related resources which the system found in the platform. This prototype system is implemented by JavaScript.

This system is used as an index page of the Structured Knowledge Platform for Nano-materials and Products. We plan to improve it and develop a knowledge portal for the platform.

7. Conclusion

Final conclusion of this study is that, first to examine if it is possible to find a definition of high-level implementation of ontology based applications and examine how these application patterns relate to other types of software patterns. And second to investigate the strengths and weaknesses of this type application and how they could be used.

An examination of the relationships between ontology application patterns and software architecture patterns have showed that these are related with one difference, the ontology application patterns include at least one ontology. This means that there are demands on the pattern description that are not present in standard architecture patterns. The properties of the ontology have to be described and the connections to the ontology also have to be given properties.

8. Future Scope

The following is the summary of our future plan:

- Ontological organization of various role-concepts.
- Management of ontology and instance models by version control, updating and reusing.
- Improvement of ontology development method based on ontological theory of role-concept.
- Augmentation of the axiom definition and the language.
- Import from different formats (RDF(S), OWL .etc.)
- Gradable support functions according to a user's level of skill.

References

[1] N. Guarino. Formal Ontology and Information Systems. In: Proc. Of FOIS'98. Amsterdam: IOS Press, 1998, pp. 3–15.

[2] M. Jarrar, J. Demey, R. Meersman. On Using Conceptual Data Modeling for Ontology Engineering. In: S. Spaccapietra et al. (eds.), Journal on Data Semantics. LNCS, Vol. 2800. Berlin/Heidelberg: Springer, 2003, pp. 185–207.

[3] Y. Wand, V. C. Storey, R. Weber. An ontological analysis of the relationship construct in conceptual modelling. ACM Transactions on Database Systems (TODS), Vol. 24(4), 1999, pp. 494–528.

[4] T. R. Gruber. Toward Principles for the Design of Ontology for Knowledge Sharing. International Journal of Human and Computer Studies, Vol. 43(4–5), 1995, pp. 907–928.

[5] J. Trinkunas, O. Vasilecas. Ontology Transformation: from Requirements to a Conceptual Model. Acta Universitatis Latviensis [Latvijas Universitates Raksti], Computer Science and Information Technologies, Vol. 751. University of Latvia, 2009, pp. 54–68.

[6] E. Bozsak et al. KAON – Towards a Large Scale Semantic Web. In: K. Bauknecht et al. (eds.), Proc. of the Third International Conference on E-Commerce and

Web Technologies (EC-Web 2002). LNCS, Vol. 2455. London: Springer-Verlag, 2002, pp. 304–313.
[7] M. A. Goncalves, L. T. Watson, E. A. Fox. Towards a Digital Library Theory: A Formal Digital Library Ontology. International Journal on Digital Libraries, Vol. 8(2), 2008, pp. 91–114.
[8] OMG: OntologyDefinition Metamodel, 2005. Available: http://www.omg.org/docs/ad/05-08-01.pdf.Accessed September, 2011.
[9] Guarino, N. and Poli, R. (eds.) 1995. "Formal Ontology in Conceptual Analysis and Knowledge Representation," Special issue of the International Journal of Human and Computer Studies, vol. 43 n. 5/6, Academic Press.
[10] Gruber, T. . "A translation approach to portable ontology specification," Knowledge Acquisition 5:199-220; 1993
[11] M. Uschold and M. Gruninger, "Ontology: principles, methods, and applications", Knowledge Engineering Review, 11(2), 93--155, (1996).
[12] Pease, A., Niles, I., and Li, J. 2002. The Suggested Upper Merged Ontology: A Large Ontology for the Semantic Web and its Applications. In Working Notes of the AAAI-2002 Workshop on Ontology and the Semantic Web, Edmonton, Canada, July 28-August 1, 2002.
[13] Mike Uschold, Martin King, Stuart Moralee and Yannis Zorgios , "The Enterprise Ontology", The Knowledge Engineering Review , Vol. 13, Special Issue on Putting Ontology to Use (eds. Mike Uschold and Austin Tate). Also available from AIAI as AIAI-TR-195:
[14] V. Zacharias. Technical Report: Development and Verification of Rule Based Systems – a Survey of Developers. Technical Report, 2008. Available: http://vzach.de/papers/2008_SurveyTechReport.pdf. Accessed May, 2011
[15] A. D. Preece, K.-Y. Hui, W. A. Gray, P. Marti, T. J. M. Bench-Capon, D. M.Jones, and Z. Cui, "The KRAFT Architecture for Knowledge Fusion andTransformation", in 19th SGES Int. Conf. on Knowledge-based Systems and Applied Articial Intelligence (ES'99). SpringerVerlag, 1999
[16] C. Golbreich, A. Imai. Combining SWRL rules and OWL ontology with Protégé OWL plugin, Jess and Racer. In: Proc. of the 7th International Protégé Conference,2004. Available: http://galimed.med.univrennes1.fr/lim/doc_92.pdf. Accessed May, 2011.
[17] H . Herbst et al. The Specification of Business Rules: A Comparison of Selected Methodologies. In: A. A. Verrijn-Stuart, T. W. Olle (eds.), Methods and Associated Tools for the Information System Life Cycle. New York: Elsevier, 1994, pp. 29–46.
[18] "Ontology Definition Metamodel, " http://www.omg.org/docs/ad/06-05-01.pdf(20060810)
[19] Sowa, J. F. (1995). "Top-level ontological categories". International Journal of Human-Computer Studies 43 (5-6 (November/December)): 669–85. doi:10.1006/ijhc.1995.1068
[20] Jasper, R. and Uschold, M. 1999. "A Framework for Understanding and Classifying Ontology Applications", In Twelfth Workshop on Knowledge Acquisition Modeling and Management KAW'99
[21] OntoWeb, D21, Successful Scenarios for Ontology-based Applications.
[22] Gruber, T. (1995). "Toward Principles for the Design of Ontology Used for Knowledge Sharing". International Journal of Human-Computer Studies 43 (5-6): 907–928.
[23] Gruber, T. (2001). "What is an Ontology?". Stanford University. http://www-ksl.stanford.edu/kst/what-is-an-ontology.html. Retrieved 2009-11-09.
[24] Oberle, D., Guarino, N., & Staab, S. (2009) What is an ontology?. In: "Handbook on Ontology". Springer, 2nd edition, 2009.
[25] Fensel, D., van Harmelen, F., Horrocks, I., McGuinness, D. L., & Patel-Schneider, P. F. (2001). "OIL: an ontology infrastructure for the Semantic Web". In: Intelligent Systems. IEEE, 16(2): 38–45.
[26] Maria Golemati, Akrivi Katifori, Costas Vassilakis, George Lepouras, Constantin Halatsis (2007). "Creating an Ontology for the User Profile: Method and Applications". In: Proceedings of the First IEEE International Conference on Research Challenges in Information Science (RCIS), Morocco 2007.
[27] Mizoguchi, R. (2004). "Tutorial on ontological engineering: part 3: Advanced course of ontological engineering". In: New Generation Computing. Ohmsha & Springer-Verlag, 22(2):198-220.
[28] Prabath Chaminda Abeysiriwardana, Saluka R Kodituwakku, "Ontology Based Information Extraction for Disease Intelligence". International Journal of Research in Computer Science, 2 (6): pp. 7-19, November 2012. doi:10.7815/ijorcs.26.2012.051
[29] Razmerita, L., Angehrn, A., & Maedche, A. 2003. "Ontology-Based User Modeling for Knowledge Management Systems". In: Lecture Notes in Computer Science: 213–17.
[30] Soylu, A., De Causmaecker, Patrick. 2009.Merging model driven and ontology driven system development approaches pervasive computing perspective. in Proc 24th Intl Symposium on Computer and Information Sciences. pp 730–735.
[31] Smith, B. Ontology (Science), in C. Eschenbach and M. Gruninger (eds.), Formal Ontology in Information Systems. Proceedings of FOIS 2008, Amsterdam/New York: ISO Press, 21–35.
[32] W. Pidcock, What are the differences between a vocabulary, a taxonomy, a thesaurus, an ontology, and a meta-model?
[33] Guarino, N.: Some Ontological Principles for Designing Upper Level Lexical Resources. Proc. of the First International Conference on Lexical Resources and Evaluation, Granada, Spain, 28-30, May 1998.
[34] Farquhar, A., Fikes, R. and Rice, J.: The Ontolingua Server: a Tool for Collaborative Ontology Construction, Proceedings of the 10th Banff Knowledge Acquisition Workshop, 1996
[35] Kozaki, K., et al: Development of an Environment for Building Ontology which is based on a Fundamental Consideration of "Relationship" and "Role":PKAW2000, pp.205-221, Sydney, Australia, December, 2000
[36] Kozaki K.. et al.: Hozo: An Environment for Building/Using Ontology Based on a Fundamental Consideration of "Role" and "Relationship", Proc. of the 13th International Conference Knowledge

Engineering and Knowledge Management (EKAW2002), pp.213-218, Siguenza, Spain, October 1-4, 2002
[37] Kozaki K.. et al.: Systematization of Nanotechnology Knowledge Through Ontology Engineering - A Trial Development of Idea Creation Support System for Materials Design based on Functional Ontology, The Second International Semantic Web Conference (ISWC2003), Sanibel Island, FL, USA, Oct. 2003
[38] Nanomaterial Center, Institute of Engineering Innovation, School of Engineering, The University of Tokyo, http://mandala.t.u-tokyo.ac.jp/
[39] Mizoguchi, R., Kozaki, K., Sano, T., and Kitamura, Y.: Construction and Deployment of a Plant Ontology, 12th International Conference on Knowledge Engineering and Knowledge Management, Juan-les-Pins, French Riviera, October, 2000.
[40] N. F. Noy, M. Sintek, S. Decker, M. Crubezy, R. W. Fergerson, and M. A. Musen: Creating Semantic Web Contents with Protege-2000, IEEE Intelligent Systems 16(2), 60-71, 2001
[41] Sunagawa, E., Kozaki, K., Kitamura, Y.,and Mizoguchi, R.: An Environment for Distributed Ontology Development Based on Dependency Management, in Proc. of the 2nd International Semantic Web Conference (ISWC2003), pp. 453–468, 2003
[42] Sunagawa, E., Kozaki, K., Kitamura, Y.,and Mizoguchi, R.: Organizing Role-concepts inOntology Development Environment: Hozo, AI Technical Report (Artificial Intelligence Research Group, I. S. I. R., Osaka Univ.), AI-TR-04-1, 2004
[43] D. Lenat and R. Guha, Building large knowledge-based systems: representation and inference in the cyc project, 1990. Addison-Wesley, Reading, Massachusetts.
[44] A. Farquhar, R. Fikes, and J. Rice, “The ontolingua server: A tool for collaborative ontology construction”, In Proceeding of the 10th Knowledge Acquisition for Knowledge-based Systems workshop, 1996.
[45] M. Kifer, G. Lausen, and J. Wu, „Logical foundations of object oriented and frame-based languages”, Journal of the ACM, 1995.

Author Profile

Chauhan Vipul received the B.Sc. degrees in Computer Science from VNSGU, Surat 2009 and M.Sc. degrees in Computer Science from Ganpat University, Mesana 2011. He was participated in National Seminar on Natural language Processing and Data Mining (NLPDM – 2012) organize by VNSGU, Surat, India on the topic Speech Synthesis. He is currently working in NCSC ,Bharuch as Lecturer since 2011. He was also attended one short term training program organize by NCCA, Bharuch.

Chauhan Falguni received the B.Sc. degrees in Electronics from VNSGU, Surat 2005 and M.C.A degrees in Computer Science from VNSGU, Surat 2009. She was participated in National Seminar on Natural language Processing and Data Mining (NLPDM – 2012) organize by VNSGU, Surat, India on the topic Speech Synthesis. She is currently working in Shri Manilal Kadakia College,Ankleshwar, Bharuch as Lecturer since 2011.