1-1-2003

The role of ontology in information management

Renato de Freitas Marteleto
University of Nevada, Las Vegas

THE ROLE OF ONTOLOGY IN INFORMATION MANAGEMENT

by

Renato de Freitas Marteleto

Bachelor of Science
Universidade Federal de Ouro Preto - Brazil
1999

A thesis submitted in partial fulfillment
of the requirements for the

Master of Science in Computer Science
Department of Computer Science
Howard R. Hughes College of Engineering

Graduate College
University of Nevada, Las Vegas
May 2003

UNLV
UNIVERSITY OF NEVADA LAS VEGAS

Thesis Approval

The Graduate College
University of Nevada, Las Vegas

APRIL 4TH , 20 03

The Thesis prepared by

RENATO MARTELETO

Entitled

THE ROLE OF ONTOLOGY IN INFORMATION MANAGEMENT

is approved in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

Examination Committee Chair

Dean of the Graduate College

Examination Committee Member

Examination Committee Member

Graduate College Faculty Representative

ABSTRACT

The Role of Ontology in Information Management

by

Renato de Freitas Marteleto

Dr. Kazem Taghva, Examination Committee Chair
Professor of Computer Science
University of Nevada, Las Vegas

The question posed in this thesis is how the use of ontologies by information systems affects their development and their performance. Several aspects about ontologies are presented, namely design and implementation issues, representational languages, and tools for ontology manipulation. The effects of the combination of ontologies and information systems are then investigated. An ontology-based tool to identify email message features is presented, and its implementation and execution details are discussed. The use of ontologies by information systems provides a better understanding about their requirements, reduces their development time, and supports knowledge management during execution time.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

I would like to thank Dr. Datta for the opportunity to be here. I really appreciate the assistance and directions you have given me. Some teachings are to be carried for life. Thank you, Dr. Taghva, for the opportunity to work at ISRI and for the insights that have made this research possible. Thank you for all your patience. I would also like to thank Dr. Nartker and Theo Lazarakis for all the time they have spent to help me improve my writing style and readability of this work.

Thank you, Dr. Batista, for all the help. Thank Zé, Dani, and Rosagela for the patience and great time together.

Very special thanks to my Mom and brothers for the support and love. Thank you Boris, Jiraya, Camila, Tatá, Dona, Ide, Brou, Carol, and all who were not cited here but believed that it would be possible.

Gratitude goes out to Steve Lumos and Claudio Henrique for all their LaTeX help.

CHAPTER 1

INTRODUCTION

The term *Ontology* was first presented by Aristotle in Metaphysics. His studies about the nature of existence and organization of beings drove him to this philosophical branch that attempts to answer questions like *what being is* and *what features are common to all beings* [62]. Indeed, the nature of existence of an entity and the identification of its common properties would explain the essence of that particular entity.

However, Aristotle did not consider language ambiguities caused by different senses of meanings, and since humans use their language to communicate, some limitations and misunderstandings wouldn't be avoided. Words with multiple meanings as well as different terms with the same meaning can be found in human languages. Hence, the relation between a word and an entity would be complete only after the interpreter processes the word concept and links it to *something* in the world.

There are some disagreements on a unique definition for ontology in Computer Science. It is reasonably defined in the information systems literature as explicitly formal specifications of the terms in a domain of knowledge and relations among them. An ontology is an abstract and simplified view of the world that it represents. It has been widely applied in natural language processing, in the representation of bioinformatics resources, intelligent information retrieval and classification, and in simulation and modeling. In general, it has a functional purpose that arises in the effort to computerize as much information as possible, inferring conclusions based

on the knowledge it represents [47].

In a world where more than 30 billion email messages are sent daily and web search engines claim to have more than 2 billion documents indexed (not considering documents generated inside enterprises and not publicly released) [82], the use of ontology to acquire, maintain, and to query information becomes essential.

In the area of knowledge representation, inference, and management, researchers have been developing standards, software, and policies to allow computers to interpret and understand data. The goal is to ease human access to relevant and trustworthy information. Indeed, computers are able to process and analyze information quickly when it is expressed in a precise, error free, and machine-interpretable format.

This thesis explores the elements needed to provide an efficient information management supported by ontologies. Chapter 2 describes ontology design, methods, and the different types of ontologies. This chapter also introduces the main ontology representation languages, their characteristics, and some tools used to facilitate ontology and knowledge manipulation tasks. Chapter 3 investigates the relationship between ontologies and information systems, pointing out how ontologies can improve system performance in handling information. This chapter also presents the role of ontology as a shared knowledge source for the Semantic Web. Chapter 4 presents an ontology-based specification and implementation of a tool used to identify email message features. Finally, chapter 5 addresses the future of the application of ontologies in existing technologies, and states the conclusion and future work.

CHAPTER 2

ONTOLOGY

The amount of information available for humans nowadays requires not only data to be easily accessible but also to be meaningful understandable by machines [88]. Ontologies are used to provide a common shared understanding of what data mean. Their utilization attempts to reduce semantic heterogeneity in a domain of knowledge [84], and to formally specify a common terminology used in a shared environment to describe a reality.

The use of a syntactically well-defined language is not enough for a machine to interpret information, especially when the language is complex, such as human languages. Hence, the relationship among the symbols or terms defined by the language must be provided to achieve a semantic understanding.

Researchers started developing ontologies once they realized that several existing ontologies could be integrated to describe larger domains of knowledge. Also, these ontologies would mitigate part of the problem that prevents reliable and maintainable software from being developed. Different systems to describe the same information were being developed using different terms, consequently sharing and reusing the knowledge across these systems were becoming more difficult [71].

The process of engineering an ontology is an interdisciplinary field that comprehends philosophy, metaphysics, knowledge representation formalisms, methodology for development, knowledge sharing and reuse, and modeling of real world concepts and their relations [16]. Knowledge acquired from people's perceptions must be analyzed and organized into a semiformal specification,

independent from the implementation language and environment. This is called conceptualization of knowledge, and it bridges the realistic view of the world with an ontological formal specification. An ontology is a product of knowledge abstracted from the conceptualization process.

The starting point defining the structure of an ontology comes from the semiotics, also called the theory of signs [62]. It is divided in three different but related parts: syntax, semantic, and pragmatic. The syntax is concerned with the relation among terms. The semantic links a term to a concept in the real world. The pragmatic analyzes how the terms are used in the real world to denote concepts. Based on the semiotics point of view, an ontology defines terms, which denote concepts in the real world, relations among terms, formalizing how things in the real world relate to each other, and how the terms are combined to express concepts in the real world. For example, the terms *student*, *person*, and *school*: if these terms are used in the English language, then they denote things in the real world and they relate to each other. For instance, every student is a person, and the expression *study at school* denotes students. Thus, these concepts are characterized in terms of axioms and constraints that are formally expressed.

Indeed, ontologies provide knowledge specifications that offer real world knowledge descriptions, used to organize the facts in a machine-understandable format, known as knowledge base [47]. The knowledge base is constructed from knowledge acquisition of the real world. Instances of objects in the real world represented in the particular domain and structured by the ontology will be used as facts by applications to infer new knowledge and to solve the problem that they had been developed for.

Knowledge Representation

knowledge is represented using five components: concepts, taxonomies,

relations and functions, and axioms as part of the ontology, and instances as part of the knowledge base.

Concepts, also referred as classes or categories, can represent anything on the real world. They can be abstract or concrete, real or fictitious, basic or complex. Indeed, they can be a task, action, strategy, table, or a car. Several issues must be taken under consideration when modeling knowledge concepts; for instance, whether they are separated in disjoint groups, what types of attributes can be defined to them, and what different values an attribute can assume.

One of the most important steps taken when designing and representing an ontology is to reasonably well-define the classes of objects in a specific domain of knowledge. For instance, not only *jaguar* can be modeled as an automotive brand in the automotive domain, but also it can be modeled as an animal in the fauna domain. Every instance of *jaguar* in the automotive domain will be related to the same concept, even though different instances may have different attribute values. These attributes, also known as slots, properties, or roles, describe the characteristics of a concept. Four types of attributes have been identified [38]:

- Instance attributes: these attributes may assume different values for each instance of the concept.

- Class attributes: all instances of a concept will assume the same attribute value.

- Local attributes: same-name attributes attached to different concepts (color is an attribute of many different concepts).

- Global attributes: these attributes are not attached to any specific concept, however they may be applied to any concrete concept in the domain of knowledge. The attribute color for example, can be used as a global attribute.

Local and global attributes are defined according to the application's needs. Instance and class attributes describe the concept when different values, types of values, and cardinality constraints are assigned to them. For example, the attribute color can be defined as numbers and RGB values that can be assigned to it. Also, if a unique instance of a concept is allowed to have different colors, the cardinality constraint can establish the number of colors, or the minimum and maximum numbers.

Taxonomies are used to organize the knowledge using generalization and specialization relations of concepts. The inheritance of classes of objects can be applied as following:

- Subclass of: if specializes general concepts in more specific ones.
- Disjoint decomposition: disjoint concepts relate to a more general concept in the same level of inheritance, but not necessarily all instances of the general concept will be an instance of a concept in the disjoint decomposition.
- Exhaustive subclass decomposition: the same as above, but the superclass is exhaustively decomposed such as any instance of it will be an instance of a subclass.
- Not subclass of: this is the denial of the subclass primitive. It is used to state that a concept is not a specialization of another concept.

As an example of a taxonomy application, consider the terms *chair*, *desk*, *furniture*, and *pen*. An instance of a chair as well as an instance of a desk are both instances of furniture, meaning that chair and desk are subclasses of furniture. Also, an instance of a chair cannot be an instance of a desk. In fact, their instance sets are disjoint. However, there are other instances of furniture that are not instances of chair and desk, then it is clear that furniture is not decomposed exhaustively. On the other hand, a pen is not a subclass of furniture.

Just as important as taxonomies are relations, which are interactions between concepts of the domain and their attributes. It may be essential to know the maximum number of arguments an attribute may assume, the type of these arguments, and the integrity constraints applied on them. For example, the concept *to study* may have attributes such as student name and school name. Functions are used to extract information of concept attribute values.

Axioms, or facts assumed to be always true in the domain, are used to constrain information, to verify inconsistencies and correct them, and essentially to deduct new knowledge. The usage of axioms is application dependent.

Lastly, instances of concepts represent elements in the domain, populating the ontology where relations, functions, and axioms are applied.

Nevertheless, the way different ontology and knowledge representation languages handle these features varies. Should the language chosen does not support a needed feature, it is required by the application to implement it, filling up the gap left by the language specification.

Ontology Engineering

The design, modification, application, and evaluation of ontologies are being studied in the area known as Ontology Engineering. Different methodologies have been proposed [10, 47, 61, 62, 73, 93] for different applications. However, literature agrees no perfect model of a domain exists. A viable solution depends on the application and the future extensions that may be added to it. Hence, there is no ideal methodology to ontology engineer. Moreover, there is a consensus that an ontology should be based on concepts in the real world, their properties, and how they relate to each other. A simple and iterative approach that is used to develop an ontology is described below [73]:

- Define the scope of the ontology: the questions that an ontology-based application should be able to answer often determine the scope. These questions are called competency questions.

- Reuse existing ontologies: this is an important issue due to two main reasons. First, if the application needs to interact with other applications, it is reasonable if they are based on the same ontology. Second, if the ontology requires an enormous amount of work to be developed, reusing existing ontologies is very helpful.

- List the terms of the domain used in the ontology: the terms used to describe concepts should be defined to the application and developers/users.

- Define the classes of objects and their hierarchy: most ontological designs use hierarchical modeling and they can be constructed using the top-down, bottom-up, or a combination of these approaches.

- Define the properties of the objects: the structural properties of the objects should be described to provide the necessary information to answer the competency questions.

- Define the values of each property: each object property should be assigned well-defined values, such as value types, cardinality, and the possible values.

Moreover, the way these steps are approached and executed is highly dependent on the characteristics of the application, on the existence of ontologies for arbitrary domains, and on the ontology engineers [47, 93]. For example, the high risk associated with a wrong advice in a medical application or how fast new insights may emerge for a specific domain of knowledge inspire the ontology development. The iterative feature of this approach based on human feedback also affects the ontology development process.

Nevertheless, identification and definition of concepts for large ontologies may be lengthy and costly. Literature presents semi-automated ontology engineering techniques, with human intervention. These techniques are based on natural language processing that may be used together with the regular concept identification by humans. They are known as ontology learning techniques. Their goal is to accelerate the ontology building process and maximize the development and usage of ontologies by knowledge-based systems. They attempt to extract relevant concepts, detect relations among concepts, and to arrange them hierarchically by identifying concept instances from unstructured, semi-structure, and fully structure data sources, including natural language text documents and database schemata [43, 63, 70, 76, 88, 94].

Other issues that must be addressed in the design of ontologies are the reusability of existing ontologies and their integration. The larger the information and knowledge an ontology represents, the better an application that is based on it would perform its tasks on a specific domain. For this reason, a consensual knowledge of a community of people must exist. Also, the terms to describe domain concepts and the structure of these concepts must be precisely defined [16, 65, 89]. Moreover, even if a perfect model and universal representation are non-existent, ontologies must provide stability and scalability in order to be integrated with other ontologies and applications.

Nevertheless, there is no correct way to evaluate an ontology or to compare it to a benchmark, nor there is a notion of experimenting or testing within ontologies [40]. Questions like *best design methodology* and *generic or specific ontology* are not answered and preferences vary among ontology designers. These characteristics are highly dependent on the application.

Ontology Integration

The process of ontology integration is not simple [41]: two systems describing the same vocabulary may not agree on the same information, unless they describe the same reality. For this reason, theories for sharing, manipulating, and composing ontologies regarded dependencies are needed. The Information Flow Framework (IFF) [55] purposes semantic interoperability among ontologies by a knowledge organization: registration, evaluation, and classification of thoughts, ideas, and concepts in order to adequately represent universal knowledge. Hence, the IFF operates in the structural level of ontologies.

The Information Flow Framework is an effort to set the logic of information flow in a shared environment. It develops a collection of mathematical theory applied to the information flow and gives the correspondence between theory and an IFF language [55]. It attacks problems that occur when no common language or format is used, when there are multiple point-to-point interactions, or when the information structure is inflexible.

It comprises two steps:

- Alignment: it is the connection of participant ontologies into a common agreement.

- Unification: it is the fusion of the participant ontologies.

The alignment process identifies the locus of integration, or in other words, the communicating parts of the ontologies and how these parts are related to the specific domain of knowledge. It is interested in the parts of the ontologies to be integrated. The alignment process also establishes the common semantics and meanings that will be expressed by the integrated ontology. Furthermore, this step must also define the boundaries of the universe that would be represented by the new ontology.

Once the process of alignment is completed, the participant ontologies can be merged during the unification process. This unification process represents the complete system of semantic integration with respect to the alignment process.

Ontology Classification

Different systems for classifying ontologies have been developed that are based on the amount and type of structures of the conceptualization [93], the subject of the conceptualization [41], and on the task accomplished during the information query process [76].

The former system includes three categories:

- Terminological ontologies: this type of ontology specifies the terms used to represent the knowledge in the domain.
- Information ontologies: specifies a framework for modeling the domain.
- Knowledge modeling ontologies: specifies the structure of the information - the objects, their properties, and their hierarchy.

The second system on which ontologies are categorized based on the subject of the conceptualization includes:

- Top-level ontologies: also called foundational ontologies, this category defines general concepts independent of any specific domain or application, for example space and time.
- Domain ontologies: defines the terms used in the generic domain, such as medicine or astronomy.
- Task ontologies: defines the terms used in generic tasks or activities, such as buying or driving.

- Application ontologies: this category of ontologies describes all the information that is needed for a specific application, which is often a specialization of the domain and task ontologies.

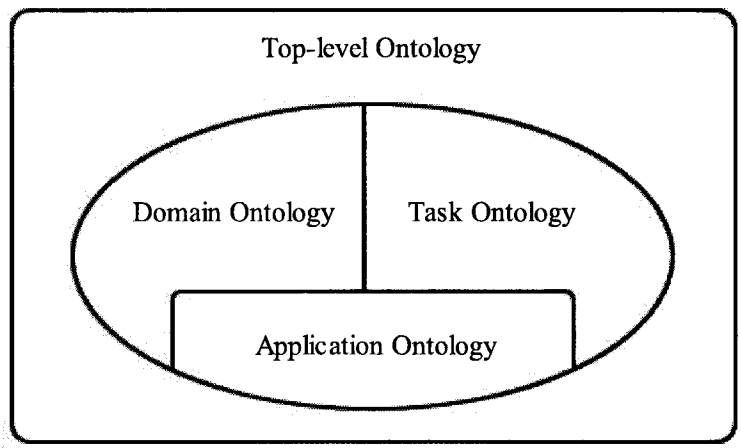

Figure 1: Relation Among Classes of Ontologies

The latter classification system is based on the ontological applications involved in information query process. This system is proposed for querying the Web [76], but it could also be extended for querying information from different sources. They are:

- Natural language ontologies: represent the knowledge of the system, containing the lexical relation between the language concepts. This level tries to describe all possible concepts with no detailed descriptions.

- Domain ontologies: describe detailed knowledge of concepts about a particular domain.

- Instance ontologies: represent the instances of the concepts described at the domain ontologies and they are subjected to frequent updates.

Ontology and Knowledge Representation Technologies

As new ontology-based applications are being developed and used, a common claim among ontology engineers is how the formal shared conceptualization could efficiently specify the common terminology of a domain of knowledge in a machine understandable format.

In recent years many ontology languages have been developed: the XML (eXtensible Markup Language) technology burst research for these ontology languages [5], leading many of them to be based on the XML syntax. The most common languages used to represent ontologies are the Simple HTML Ontology Extension (SHOE), Ontology Markup Language (OML), Conceptual Knowledge Markup Language (CKML), Ontology Exchange Language (XOL), Resource Description Framework (RDF) and RDF Schema, and the XML Declarative Description (XDD). Also, based on the RDF(S) - union of the RDF and RDF Schema - two additional languages were developed: Ontology Inference Layer (OIL) and DARPA Agent Markup Language + OIL (DAML+OIL). Following the ideas of the DAML+OIL, the Web Ontology Language (OWL) is also presented. Although a deep description of each language syntax specification and technological details are out of the scope of this work, some of their features, advantages, performance on ontology construction and representation, similarities, relations, and their roles in the ontology research area are analyzed and pointed out.

The SHOE language, which was developed at the University of Maryland, is considered to be not more than an extension of the HTML (Hyper Text Markup Language). It incorporates some machine-readable semantic knowledge to web pages. Unlike HTML tags that concern with information presentation, SHOE tags provide structure for knowledge acquisition and representation [45]. The main purpose of SHOE is to improve search mechanism in Web pages by gathering meaningful information. It allows intelligent agents to read and to understand data.

SHOE is a common language used to exchange Web data semantically.

Initial effort for the development the SHOE language started when the need to separate the word syntax to its semantic on the Web became evident. For example, a word index search for *cook* did not show any distinction whether it was about Cook County, cooking, or a person called Cook. Since natural language processing programs still need improvements to completely understand natural language content, SHOE was introduced to help bridge the gap on Web semantic understanding where single shared definitions about a domain did not exist. SHOE ontologies are publicly available on the Web, and ontology extensions and integration are promoted based on rules adhering to SHOE interoperability.

SHOE ontology is intended to declare categories for data entities and their relationship and to allow inference upon the data entities from existing rules. On the other hand, HTML pages with embedded SHOE data (SHOE pages) intend to declare the arbitrary data entities, categorize, and to describe the relationship between entities or between an entity and data. In fact, SHOE pages can be both queried and indexed, and some search engines have been adapted for this purpose. Furthermore, tools for improving searches on SHOE pages, integration of SHOE ontologies, and performing error checking for ensuring SHOE syntax and semantic correctness have been developed [45].

There are two categories of SHOE tags:

- Constructing ontology tags: used to define the set of rules that represent the domain, known as SHOE Ontology.

- Annotating Web document tags: used to declare data entities and to infer about those entities according to the rules specified in one or more SHOE ontologies.

XML is an application and platform independent language written in simple

text. Its grammar provides the basis for representing other languages in a standardized way. XML provides the data format for structured documents, but not the vocabulary [57]. For instance, Chemical Markup Language, Commerce XML, MathML, VoiceXML, Geography Markup Language, and the eXtensible Scientific Interchange Language are some of the XML applications that have been developed [34, 39].

XML handles managing information in the way that is required today. Rather than a language to create and to display contents on the Web, its tags are used to identify and to provide data availability to agents. Indeed, XML code contains not only data, but also the data structure information, called metadata, defined by Document Type Definitions (DTD) or XML schemas. Although they both are used to constraint the XML tags based on a set of rules, the XML schema addresses some DTD limitations such as the creation of complex object structure types, specification of numeric ranges in documents, and inclusion and derivation mechanisms. They define the elements, attributes, and other features for XML instances that are allowed or required in a complying document, specifying how tags are called, their meanings, how they can be used, and how they are structured and nested [9]. DTD and XML schemas are not only used to define a markup language, but also used to validate document consistencies according to the language syntax.

Nevertheless, the XML generality also produces a weakness. The XML language does not define the data use and semantics. In order to assure interoperability, flexibility, and functionality, the parties that use XML to exchange their data must agree beforehand on the vocabulary and its semantics, and also on the data structure. However, several initiatives, such as the ebXML [23], RosettaNet [78], and BizTalk [6], exist to provide XML schemas and vocabulary standards in many different areas. These repositories of schema specifications avoid similar schemas from being developed in their own divergent vocabularies.

The XML easy-to-use syntax for data structure made it useful as the basis for the development of many languages used for knowledge representation and ontology construction.

The Ontology Markup Language (OML) is considered to be a XML serialization of SHOE [55] with suitable changes and improvements to represent ontological and schematic structure. OML presents the ability to specify classes, relationship among classes, objects, and constraints.

The latest version of this language is highly RDF(S) compatible [60], providing its own solution to the namespace problem. This language is intended to provide a framework based on XML and predicate logic to the Web. For this reason, it is divided in four different layers, each of which having a specific purpose [38]:

- OML core: related to logical aspects (types, instances, and relations) of the language and it is included in the other levels.
- Simple OML: this level can be mapped directly to RDF(S).
- Abbreviated OML: promotes interoperability with the conceptual graphs standard, which can be defined as an abstract syntax independent of any notation. The formalism, however, can be represented in either graphical or character-based notations.
- Standard OML: it is the most expressive and natural version of OML.

The Conceptual Knowledge Markup Language (CKML) is based on the elements of the OML. It is extended to provide a conceptual knowledge framework for representation of distributed information: classification, theories, interpretation, local logics, and infomorphisms. This means that it provides not only knowledge representation and inference, but also principles and techniques from information flow and distributed systems. These principles and techniques are based on the fact

that classification must be able to grow organically, in the same pace of the growth of the knowledge [55]. Their purpose is to ease different knowledge structures, coded in ontologies, to be compared and merged [54].

The classification system divides the knowledge into dimensions of information, called facets, which provide a description of the information resource. This classification is approached by conceptual scales, which are divided in three types:

- Abstract Conceptual Scale: represented by attribute names. This type of conceptual scale introduces the terms and specifies attribute definitions by term-to-term relationship. This is represented as theories in the CKML, where terms or attributes are called types and the clauses are called sequent, and they define the vocabulary.

- Concrete Conceptual Scale: represented by queries bound to attribute names. This type is responsible for giving meanings to the terms in the previous type by attaching a single-variable query to each term. In the CKML, these are the theories interpretations.

- Realized Conceptual Scale: represented by object-attribute incidence constructed by attribute query evaluation. In the CKML, this is called infomorphisms. The infomorphisms generate the local logic, which are any binary relation on the vocabulary.

This language was developed based on knowledge management approach provided by the conceptual knowledge processing (CKP) [55]. CKP mathematically establishes the equivalence between the non-hierarchical structure of object relations and the hierarchical structure of concepts.

The Ontology Exchange Language (XOL) [53] was developed by the US bioinformatics community to share ontologies in their domain after studying the representational needs of experts in bioinformatics. Although it was developed with

a specific purpose, this language can be used for ontologies in any domain. The goal of XOL development was to provide the semantics of object-oriented knowledge representation based on the OKBC-lite (a simpler version of the Open Knowledge Base Connectivity) [11]. It also provides a simple to parse and well-defined syntax based on XML.

The Open Knowledge Base Connectivity is an application programming interface (API) based on conceptualizations of classes, individuals, slots, facets, and inheritance. It is a set of operations that provide a generic interface to the knowledge access. These operations are object-oriented represented by the OKBC knowledge model. Similar to any object-oriented language, classes in OKBC knowledge model are sets of entities, which are called instances. The classes define the types of their instances, called individuals. Any entity has a collection of slots associated with it describing its direct properties, which can be either inherited by subclasses, called template slots, or not inherited by subclasses, called own slots. The OKBC knowledge model also defines facets to describe the properties of slots associated to entities, such as cardinality and range [11].

XOL ontologies are syntactically defined using a single set of XML tags. This is a generic approach where a XML DTD or schema defines this set of tags that can be used to describe any and every ontology. These tags are generic to all ontologies, and the specific information about this ontology is in between the XML tags. However, the generic approach has one primary disadvantage: XML parsing engines have limited types of checking on XOL specifications. For example, an integer value for a slot cannot be identified as integer within the XML DTD for XOL. This task is left for applications, which load XOL files. Nevertheless, any software tool developed to manipulate XOL ontology in one domain will work for any XOL ontology in any domain. This is possible because XOL ability to specify a very restricted and well-defined set of XML documents.

The Resource Description Framework (RDF) [60] was developed by the W3C [95] to allow specification of concepts and abstract data syntax on the Web. It is a powerful language intended to provide mechanisms to represent data, services, processes, and business models based on the XML standards. Indeed, the RDF model defines neither the vocabulary nor the semantics of any application domain.

The RDF data model consists of three object types [60]:

- Resources: they are the concepts in the real world, described by RDF expressions. Resources are always named by Uniform Resource Identifiers (URI) and optional anchor ID.

- Properties: they define specific aspects, characteristics, attributes or relations describing resources.

- Statements: they consist of a specific resource with a property and its value for that resource. A value not only can be a literal value, free text, but also another resource.

The object-property-value triple basic foundation of RDF specifies the objects, their properties, and their values. It allows objects to be values of other object properties, hence building a basic data model for metadata. It is also defined in an object-oriented modeling system providing classes hierarchy and offering extensibility through subclasses refinement.

The semantics of the terms of a domain represented in RDF is defined by the RDF's vocabulary description language, also called RDF Schema. RDF schemas (RDFS) are related to RDF documents in the same way that XML schemas are related to XML documents. Their primary goal is to specify the classes of resources, their inheritance relationship, and their properties in terms of the classes of resources to which they may apply. RDF Schemas use modeling primitives such as *class*, *subClassOf*, *property*, *subPropertyOf*, *domain*, *range*, and *type*.

Although RDF(S) provides a suitable mechanism for ontology interoperability when describing information, it does not specify whether or how an application will use it. The task of capture meaningful generalizations about data represented by RDF(S) is left to be executed by a specific purpose application.

The XML Declarative Description (XDD) [96] is a single formalism capable of providing representations for axioms, conditions and constraints, as well as concepts and attributes. A XDD representation is a set of:

- XML element: denotes an instance of a concept in the domain.
- Extended XML element with variables: the variables represent implicit information of the instance.
- XML clauses: rules, conditional relationship, integrity constraint, and ontological axioms used to define XML element relations.

It adds to the XML and RDF(S) representation features such as *symmetry* and *inverse* of concepts. For example, in the email message domain, the sender plays an inverse role of the receiver, and this relation can be represented in XDD. Not only XML and RDF(S) can be mapped to XDD, but also the MathML (Mathematical Markup Language), XMI (XML Metadata Interchange Format), and WML (Wireless Markup Language) can. This mapping capability gives XDD the power of semantically define these languages, enabling interoperability of services and tools described in some languages previously developed.

The Ontology Inference Layer (OIL) was developed in the OntoKnowledge project [30] to permit semantic representation and to provide an inference layer for ontologies on the Web [31]. OIL incorporates three distinct areas:

- Frame-based system: OIL is based on notion of concepts, organized hierarchically into classes, and their properties.

- Description logics: OIL uses of the description logics formal semantics and reasoning support.

- Web standards: OIL is based on the XML syntax, considered to be an extension of the RDF(S) and XOL standards. OIL contains some language primitives that are not defined in the RDF(S) language specification [15, 49].

OIL is organized into layers, each of which adds functionality and complexity to lower layers. This allows at least a partial understanding of the top layers by agents limited to interpret part of the language description [31]. They are, from the lower to the upper layer:

- Core OIL: groups the OIL primitives and it can be mapped directly to the RDF schema. Therefore, RDF schema agents can also interpret this layer.

- Standard OIL: it is a complete OIL model that provides powerful expressiveness for both semantics specification and inference.

- Instance OIL: allows instantiation of concepts of the Standard OIL layer.

- Heavy OIL: it is the layer for future extensions of OIL for representation and inference capabilities.

Nevertheless, OIL also presents some weaknesses [49]:

- Property values inherited by subclasses can not be changed or overwritten,

- There is no automatic mechanism to rename, restructure, and to redefine imported ontologies, and

- There is no support for instances to be defined as classes of their own.

The DARPA Agent Markup Language (DAML) [66] is an initiative funded by the US Defense Advanced Research Projects Agency to develop tools,

infrastructure, and applications to convert current Web content into machine-understandable information.

The DAML+OIL consists of the DAML current markup language and it is a result from the combination of the DAML-ONT, initial version of the DAML ontology language specification, and the OIL. DAML+OIL relies on the power of XML syntax representation and on the formal semantics provided by RDF(S) to describe classes, subclasses, individuals, and their properties. It breaks XML and RDF(S) restrictions from supporting variables, general quantification, rules to describe constraints and resources relationships, such as cardinality and union, disjunction, inverse, or transitive relationship [14]. Furthermore, DAML+OIL specification also contains theorem provers and problem solver mechanisms for searching and detecting knowledge inconsistencies [14, 32].

The Web Ontology Language (OWL) [83] is a semantic markup language currently being developed by the W3C [95]. It is derived from the DAML+OIL ontology languages aiming to publish and to share ontologies and their related knowledge base on the Web. It is intended to provide classes descriptions and their relationship inherited in Web documents and applications. Therefore, it not only formalizes a domain of knowledge by defining classes and their properties, but also reasons about these classes and their individuals.

The Web Ontology Language can be divided in three parts [83]:

- OWL Lite: this part provides to the user hierarchy classification and simple constraint definitions. For example, it only permits cardinality values of 0 or 1.

- OWL DL: it is related to the description logics. This part includes the complete OWL vocabulary and supports simple constraints. It still presents type separation: class identifiers cannot represent individuals or properties.

- OWL Full: translates the constraint freedom of RDF to OWL, allowing classes

to be seen not only as a collection of individuals, but also as an individual to represent the class intention.

Nevertheless, the OWL describes mechanisms used to further define properties using the XML syntax, which enhance reasoning about them:

- Transitive: if a person is in a city, and this city is in a state, then the person is in that state.
- Symmetric: if John is Mary's neighbor, then Mary is John's neighbor.
- Functional: if a person has a specific date of birth, that person is always associated with the same date of birth.
- Inverse of: John receives an email from Mary if only if Mary sends an email to John.
- Inverse functional: this is the inverse of the functional property.

Furthermore, the OWL proposes a more powerful representation of property constraints, new operators to construct more complex classes, and also mechanisms to map and to compose ontologies. For instance, it allows cyclic subclasses and multiple property domain and range. It also eases the ontology reuse task during the ontology development process, where much of the effort is spent to relate classes and properties from different ontologies [83].

Ontology Tools

Tools to allow people to fully explore and apply emerging technologies in their activities are indispensable [5]. Several tools originally developed in the Artificial Intelligence field are being adapted and used as the foundation to the development of new tools for ontology and knowledge manipulation. These new tools present

some restrictions, especially on the ontology languages they support, however, in general they minimize the effort spent on handling ontologies and knowledge acquisition [25, 29, 37, 46].

The Parka-DB ontology management system [37, 87] is not only a tool to help the user to code an ontology, but also it provides a specific language supporting an efficient inference engine. It is capable to compute recognition, handle complex queries, and to infer on the knowledge base. Nevertheless, the Parka system offers RDF and DAML compatibility, thus allowing RDF instances to be loaded as Parka assertions into the system.

Approaching several aspects of ontology engineering, the OntoEdit [87] provides to the user a collaborative environment to ontology development based on requirement specifications, refinement, and evaluation. The requirements deal with the semiformal ontology description (domain, knowledge sources, design issues). Ontology formalization will evolve from the requirement specifications into an ontology language representation in the refinement process. Last, the ontology will be evaluated based on the application needs. This process will prove whether the ontology fulfills the requirements defined previously. OntoEdit is a client-server architecture tool that not only supports several tasks performed during those processes, but also provides a powerful inference engine.

Chimæra [67, 68] is a Web-based tool focused on the ontology creation and maintenance processes, especially on the evolving and merging multiple ontologies tasks. The support to these tasks plays an important role when team members need to integrate different ontologies that should work together (noncollaborative environment). It is also important when multiple ontologies are merged to produce a more consistent one. However, Chimæra requires that ontologies are represented according to the OKBC [11] representation system specification to be compatible.

The WebODE [2] is presented as a collaborative environment to develop and to

maintain ontologies, and to create other ontology development tools and ontology-based applications. This tool provides support to ontology building, translation, integration, merging, and browsing, and also provides an inference engine and an axiom generator. It provides extensibility and high usability by allowing ontology access through either a well-defined service-oriented API or export/import services into/from several ontology specification languages.

Specifically designed to build ontologies in the OIL and DAML+OIL languages, the OilEd [4] provides to the user the ability to design accurate and detailed ontologies by reasoning on the expressiveness of the OIL description logics. The reasoning support during the design and/or integration process, provided by the FaCT system [48], can be very helpful to expose logical inconsistencies and to find class relations, especially when the ontology is large or there are multiple authors, or even both. The OilEd offers a FaCT connection service to send the ontology to the reasoner for verification, and to receive the results back. It then highlights inconsistent classes and rearrange them hierarchically according to the changes proposed by the FaCT reasoner.

Protégé-2000 [74] is a highly customizable tool used to develop and to maintain ontologies. It not only provides to the user a friendly interface, but also presents as its main characteristics the ability to be adapted to different languages and technologies, such as reasoning mechanism and knowledge annotation, using back-end plug-ins. Its customization power goes beyond: standard class and slot definitions can be changed or extended, and the content and layout of the knowledge acquisition forms can be modified, depending on the particular domain being explored. These features make the Protégé-2000 a timesaving tool when customization is required by an application in a domain.

The Ontolingua Server [29] provides to the user a Web environment of tools and services to support a collaborative ontology development and maintenance by

storing a library of ontologies and enabling new ontologies to be created and existing ones to be modified. The collaborative work is possible as a result of the Ontolingua architecture that provides notification, comparisons, and shared logs. Despite of the usage of its own language for ontology specification, based on Knowledge Interchange Format [36] with some extensions to support reasoning and object-oriented knowledge representation, Ontolingua Server offers translation facilities from the ontology repository to other languages, as well as an application program interface (API) to enable ontology access by remote applications.

Ontosaurus [89] is also a Web-based tool that intends to facilitate the development and maintenance of large-scale ontologies. It is mainly an ontology server that interfaces with Web browsers as clients providing availability for browsing, editing, querying, and translating ontologies while maintaining theirs coherency and consistency. The basic distinction between Ontosaurus and Ontolingua relies on the knowledge representation system: Ontosaurus is based on Loom KR systems [51].

Following the same approach to enable browsing, creating, and editing ontologies over the Web, WebOnto [18] relies on a Java-based client to excel other tools' HTML solutions to interface problems: all data centralized in servers, one-shot connections, and Web browser limitations. It offers a collaborative client/server environment that displays ontologies using graphics. They can be moved or operated on, providing a direct manipulation interface and still guaranteeing data consistency.

The RDF Editor and RDF Instance Creator [37] are two tools used to markup information in the RDF language. The RDF Editor is a *WYSIWYG* (what you see is what you get) environment that provides the ability to semantically classify and to annotate data into RDF syntax on Web pages based on multiple preexisting ontologies on the Internet. Similarly, the RDF Instance Creator allows the user to

generate RDF markup simply by filling up object, property, and value information, especially from non-text source, into special forms constructed based on predefined ontologies.

The SMORE [52], Semantic Markup Ontology and RDF Editor, is proposed to assist users to semantically annotate, modify, or extend information on Web pages, emails, or any other online contents based on preexisting ontologies. It supports the specification of the text information into the triple model subject-predicate-object related to the ontologies being used. Also, it presents a module to annotate information from images or parts of images, generating semantic data from image sources. This tool also provides to the user an ontology search engine to help marking up documents from online ontologies.

CREAM [44] is a framework that supports the DAML+OIL language for ontology and knowledge base representation. It consists of several modules to provide the ability to develop ontologies and to annotate metadata while authoring Web pages, as well as a posteriori annotation. They keep the metadata consistent according to the ontology definitions. This framework also can be submitted to further enhancements such as information extraction and collaborative metadata creation as a result of its plug-in extension support architecture.

The RDF Screen Scraper [37] is a tool that helps the user extract RDF markup information form regular Web page markup, based on a tag-content mapping. Aiming the same goal, the ConvertToRDF tool extracts RDF information from delimited data source, such as electronic spreadsheets or databases, by mapping column headers and ontological terms.

ConsViSor tool [3] for ontology consistency checking plays an important role in the knowledge reasoning process with no human supervision. Inconsistent ontologies may lead to incorrect conclusions. This tool processes the ontology, the facts, and the rules in a logic-programming engine, checking the axioms and forcing errors,

even though the ontology is not inconsistent, presenting messages to the users. Inconsistent ontologies are common to occur when distinct smaller ontologies are merged, but similar concepts overlap one another and different assumptions are mad. ConsViSor helps to identify and correct these problems. Nevertheless, ConsViSor also has limitations: it does not check logical inconsistencies and it is not compatible to all existing ontology languages.

Despite not all the existing tools were cited previously, it is reasonable to identify some of their characteristics, which can be taken into consideration when selecting and using a tool to facilitate ontology manipulation and knowledge annotation processes. An evaluation framework has been proposed [19], which categorizes tool features into three distinct dimensions: general dimension, ontology dimension, and cooperation dimension.

The general dimension refers to aspects that deal with user interface. It includes not only what the user can visualize but also what actions the user can perform and the levels of customization provided to the user according to his needs. Important issues are considered at this dimension such as the ability to enter and navigate through complex knowledge structures [25], the clarity and consistency of the interface according to the ontology and available functions [44], the clarity of command meanings, the tool stability against crashes, its installation requisites, and its available help information and documentation [19]. Nevertheless, the ability to customize the tool regarding interface changes, ontology language available options, additional features such as inference engine, and ability to be a domain independent tool are relevant characteristics [25].

Similarly important, the ontology dimension refers to the level of support for ontology development and maintenance, and knowledge annotation [44]. When choosing a tool to help performing these tasks, one should consider the ability to correctly define classes, their attributes, and multiple-inheritance classes, the error

checking mechanisms to maintain data consistency according to the ontology, and the availability of examples and libraries of ontologies that can be reused for a specific domain [19].

Also, a collaborative environment may be desired when developing and maintaining large-scale ontologies, thus overcoming problems like data consistency and sharing [18, 19, 29, 89]. Some of the customization aspects overlap this cooperation dimension when considering the possibility to import and export ontologies and knowledge base from and into a specific language, as well as merging different parts of ontologies. Issues such as synchronous edition, locking mechanism, and data change recognition and feedback are relevant to consider in a shared ontology environment.

Nevertheless, there is no perfect tool to choose, and the user expertise, tool familiarity, and degree of support needed from the tool in the tasks to be performed are as important as the previously mentioned aspects [19].

CHAPTER 3

ONTOLOGY AND INFORMATION SYSTEMS

In an environment where people work with, distribute, or create new information and knowledge, information systems play an important role in information analysis and decision-making [79]. They offer a consistent and integrated architecture to allow easy access to one of the most important assets of any organization: information [81]. Information systems optimize the flow of knowledge by collecting, retrieving, processing, storing, and distributing information. They support business planning and conducting at operational, management, and strategic levels.

Before the use of ontologies in information systems emerged, there was a belief that knowledge could be inferred using simple reasoning and representation mechanisms [90]. However, the unstructured knowledge represented earlier in these systems did not aim the solution for problems such as term definitions, domain fact expressions, and inference and problem solving support. It led to insufficient understanding of the system and an expensive knowledge acquisition. Ontologies structure the knowledge base, promote knowledge sharing, and provide a solid foundation to build the system [90], hence lowering the system development and maintenance costs.

Indeed, the conceptual analysis and domain model provided by ontologies support information systems in both development and run-time processes. They support different tasks such as knowledge engineering, database design and integration, and information retrieval and extraction, resulting the so called

ontology-driven information system [41]. During the information system development process, ontologies provide vocabulary mappings, distinctions, and meanings according to a view of the world. They unify divergent vocabularies and increasing the quality of the natural language informal specification analysis [61]. This enables the developer to experience a higher level of knowledge use [41]. On the other hand, at run-time, ontologies are used explicitly to drive aspects and components of the system. For example, they support database queries, enable communication between agents, and support knowledge reasoning and natural language processing with their domain knowledge representation.

Special attention can be given to the natural language processing, ontology, and information system triad. Natural language systems support information systems both at the knowledge producer side, with the aforementioned ontology learning and knowledge annotation, and at the knowledge consumer side, supporting knowledge redundancy discovery [28], information retrieval, categorization, generation, and security [1, 13], question answering, and advice giving [72]. Ontologies form a central resource for natural language understanding, supporting the natural text meaning extractions, knowledge reasoning, and natural text generation. They reduce the natural language ambiguity by providing lexical meanings for a specific domain [72].

Diverse areas, such as chemistry [1, 61], computer science [14, 24], business management and e-commerce [33, 75], information classification and security [77], linguistics [64], education [42], medicine [35], engineering [8], tourism [21, 94], and bioinformatics [12, 27], confirm the advantages of developing ontology-driven information systems.

The Semantic Web

The most used information system nowadays, the World Wide Web, is

considered by some to be a nearly unlimited source of information [58]. It emerged from the idea of having a location independent information space with immediate access, where shared knowledge would support communication. Furthermore, the concept of communication is not only applied to people-to-people, but also extended to machine-to-people and machine-to-machine communication, where machines would devote their analytical power to describe, infer, and to reason about the vast human content available on the Web, providing the Semantic Web [5].

Ontologies play an important role in the Semantic Web: the well-defined format of information and the connection of related information give to machines much more computational power to process data and to build understanding about it [17]. Ontologies' ability to formally and precisely define terms for knowledge applications enables knowledge domains to be described and linked to each other to define a whole decentralized information system [5, 15, 87]. The idea of referencing information, not copying them, leads to a modeling approach. First, suitable reference libraries are selected, then customized, and finally the knowledge domain is created using the reference models [33]. Semantic mapping techniques between elements have been proposed [17, 55] to facilitate information reuse and reference. Indeed, the more successfully ontologies are used the more successfully Semantic Web results tend to be [56, 63].

Hence, the Semantic Web requires machines to interoperate accessing and using Web resources [15, 27]. These resources must be open and understandable, raising concerns about privacy, reliability, and security. Thus, the infrastructure for the Semantic Web must allow resources to be located, identified, and manipulated safely, and it must also provide language representations to define and to express knowledge efficaciously. These representations must explicitly identify objects and they must be manageable to eliminate inconsistencies, reduce differences, and to promote integration among models of the real world. Also, accurate knowledge

reasoning is essential. The quality of the reasoning results ought to be analyzed according to the application and reasoning method adopted. These issues must be addressed beforehand to assure a dependable Semantic Web, where data are timely, accurate, and precise. Previously unconnected services interoperate guaranteeing security and time constraints for real-time processing [92]. Moreover, the future of a Semantic Web depends on its degree of resources availability as well as knowledge reusability and quality. Thus, it is important to promote foundational ontology development and to well-define ontology libraries.

Despite of the simple theory and requirements above-mentioned, knowledge annotation for the Semantic Web deserves special attention no matter the information medium or representation language [26]. The difficult of producing markup information and the amount of information already available with no semantic annotation whatsoever are the most significant reasons against the Semantic Web [46]. A study [22] has revealed the lack of annotated information available on the web. It concluded that the amount of semantic applications is directly linked to the amount of annotated information. Also, the development of Semantic Web applications would motivate the community to explore the semantic annotation potential ideas and possibilities. However, the word semantics model [69] presents a feasible solution for existing technologies and resources. First, the identification of entities in a free text with no extra knowledge or markup information. Then, the connection of these entities to ontologies to find entities relations. Last, the addition of semantic tags to terms replacing the classic keyword-matching-based retrieval technique with matching between semantic forms. Nevertheless, steps are being taken to annotate information from relational database into a knowledge base representation [85]. They propose integrated and automated approachs to map logical database model and ontologies, generating knowledge base instances from data stored in the database. Also, natural language information

extraction techniques to generate annotation exist, linking nouns and properties to ontologies [59]. Although ontologies may evolve to accommodate different context, they set the boundaries for content annotation providing part of or the entire background knowledge about the domain [12]. They also describe how the information is annotated.

Nevertheless, the Semantic Web allows web portals to be created providing infrastructure and methods to acquire, structure, understand, process, and to share information. Web portals goals would support semantic browsing and querying with semantic similarity. Also, they would provide semantic information to agents aiming a higher communication quality with a wise information manipulation [14, 86]. In fact, one of the biggest impact of the Semantic Web is on the information retrieval area, especially on Web search engines [12]. The mechanisms to rank and retrieve relevant documents must incorporate semantic features. They will allow not only the best document delivery given a topic, but also they will analyze the fact into the document to help knowledge workers assess trustworthy information [80].

Although the entire World Wide Web will not become semantically navigated, the Semantic Web will affect some specific corners, such as e-commerce, and many companies have already turned themselves to ontology-based information systems development [33, 75, 88].

CHAPTER 4

EMAIL CLASSIFIER: A CASE STUDY

Information classification is an important issue in enterprises nowadays. The Information Science Research Institute (ISRI) at the University of Nevada, Las Vegas, has been studying and building tools to provide automatic filtering of information from documents intended for dissemination.

The Email Classifier project at ISRI intends to classify email messages, here considered documents, into categories by analyzing email message features. An email feature identifier reads the email messages, stored in XML files, identifies their features based on formal rules, and outputs these features to a Bayesian classifier. The Bayesian classifier applies probabilities to identified features and it categorizes the messages [91]. Figure 2 illustrates the email classifier mechanism.

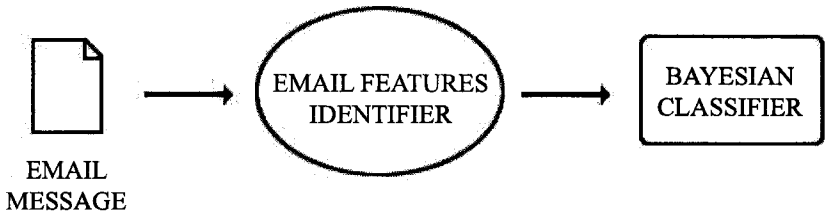

Figure 2: Email Classifier

The tool presented in this work was built in parallel with another tool used in the Email Classifier project at ISRI [91] comprising the email features identifier. Both tools are based on the same email ontology, however, the developed tools use different technologies. This work describes our developed tool, which is based on

OntoJava [20], and the actions taken for accomplishing its purpose of identifying email message features.

Email Ontology

The starting point for any ontology-driven information system development was the ontology description and implementation. The ontology set the domain boundaries where several concepts were defined according to the information system goal, in this case the identification of specific email characteristics.

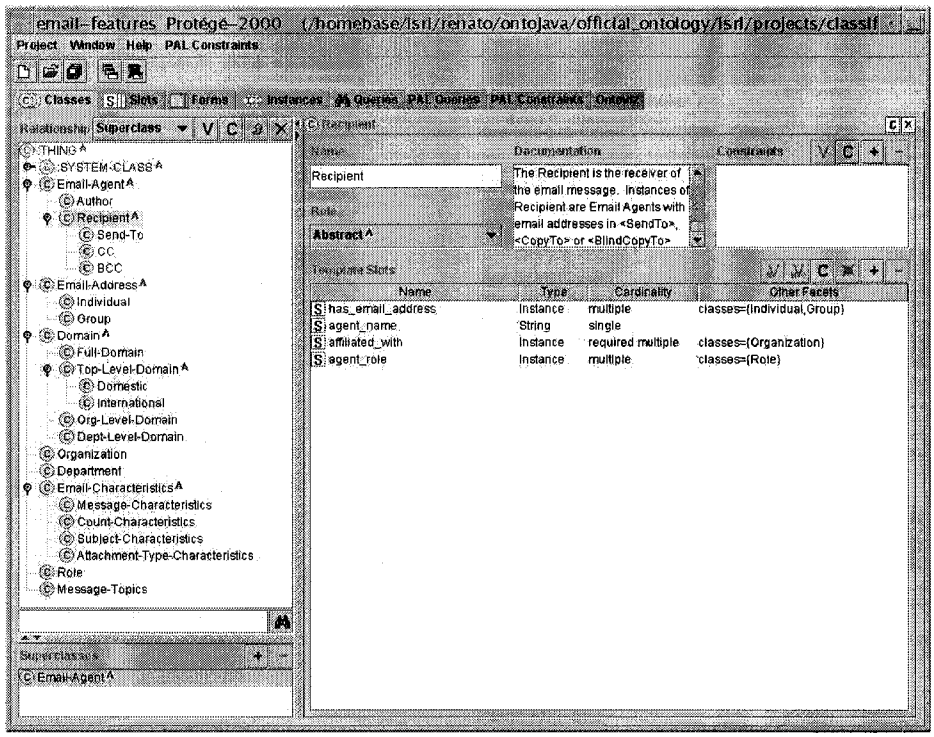

Figure 3: Email Ontology

The email ontology used by this tool, as mentioned before, was the same used by the Email Classifier project [91], implemented using Protégé-2000. This tool was chosen due to its scalability and stability, and also due to its explicit ontological representation that hierarchically displays classes and their properties. Another advantage of using Protégé-2000 [74], specifically in this case, is the RDFS and RDF

built-in ontology and instances representation that it provides, required by OntoJava. Figure 3 shows the Protégé-2000 interface and the classes defined.

This ontology was based on either aspects of the classification criteria or aspects observed in the training data. The concepts included in this ontology describe email metadata, such as author, recipient, subject, and email body and attachment type characteristics. Also, concept properties and possible values were defined, as well as the specific category for each value. Nevertheless, real world concepts and properties identified were translated into Protégé-2000 classes and slots in a flexible manner, assuring that new concepts could be easily added to it if needed.

RuleML

Another requirement established by OntoJava is the rule format: RuleML [7]. The Rule Markup Language is an initiative to define a rule system description suitable for the Web, based on XML syntax, that allows rule exchanges among systems. It was designed primarily to enhance content of web pages where four types of rules can be defined, each of which presenting its own syntax:

- Reaction rules: specify behaviors in response to events.
- Integrity constraint rules: signal inconsistency when specific conditions are fulfilled.
- Derivation rules: allow dynamic inclusion of derived facts.
- Facts: unit clauses where their premises are always true.

The rules used in the system described in this work were written according to the requirement analysis made in the ISRI Email Classifier project. For example, the presence of the term *lunch* in the subject line, the number of unique terms in

the email body, and the type of attachments are relevant characteristics to the classification system. Conditional rules were constructed according to these relevant characteristics and were then translated from the natural language to RuleML. It was facilitated due to the well-defined RuleML syntax and naming specification provided by the ontology.

During the program execution, rules are fired by matching values of properties of an email message to Protégé-2000 slot values, signalizing the presence of email features to the main program.

OntoJava

The combination of the RDFS, RDF, and RuleML results in a sophisticated software agent, defining respectively the taxonomies, factual knowledge, and the rules for knowledge manipulation [7]. Nevertheless, execution of the agent is dependent upon the knowledge, perception, and the action subsystems. These subsystems are responsible for knowledge inference, incoming message handling, and outgoing message handling.

The OntoJava cross compiler converts RDFS and RuleML into a set of Java classes that provides a main memory object database and a rule engine [20]. They fulfill the requirements of needed subsystems for the RDF(S) and RuleML based agent execution. OntoJava uses the hierarchical description of concepts provided by the RDFS representation. OntoJava directly translates every concept in the ontology to Java classes, where object inheritance is defined by the *subClassOf* property provided by the RDFS specification. However, the Java language does not support multiple-inheritance supported by the RDFS specification, and OntoJava will generate a Java interface instead of a Java class for each of the RDFS class that relies in this case.

Nevertheless, the properties defined for each concept at the RDFS are also mapped into Java object variables and methods, depending on the property range. Variables define literal properties while methods define relations to other Java objects. The property domain specified by the RDFS defines the object destination of these variables and methods.

Also, OntoJava provides a data loader mechanism that translates all RDF entries into Java commands to insert these RDF instances as Java instances into the object database. This mechanism not only handles the RDF data given as input to OntoJava, but also it interfaces the system and the user at run-time to allow extra instances to be added to the knowledge base during the system execution. Nonetheless, OntoJava offers a convenient data structure to allow an efficient access to objects stored in the main memory database. Figure 4 shows the OntoJava possible input and output.

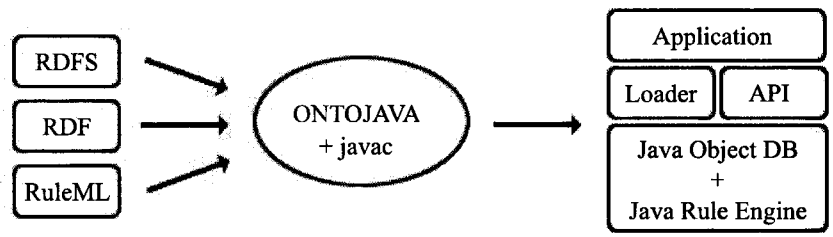

Figure 4: OntoJava

Each rule in RuleML is converted into a static method by OntoJava. A brutal force approach is applied by a Java rule engine using the forward strategy to check all possible bindings of free variables for each rule until no rule is fired. Although it is quite an inefficient inference engine, OntoJava implements some optimizations. Every time a specific property is updated, only rules that contain that property in the body are reevaluated. Furthermore, OntoJava restricts the syntax of RuleML to assure that one minimal model exists for describing the smallest possible fact base. These restrictions are specifically made for negation rules, allowed by RuleML

syntax. They would prevent an any-order rule evaluation. On the other hand, OntoJava extends the RuleML syntax to permit Java statements to be embedded in reaction rules' heads, rather than assertion of new facts, providing more flexibility to the rules.

The Working System and Results

The focus of this system is on the ontology sharing, reuse, integration with existing information source, and the deployment into an information technology solution.

The first step during the development process was to represent the ontology and the included instances using RDFS and RDF specification, respectively. A few adjustments were made on the RDF(S) representation generated by Protégé-2000 for compatibility with OntoJava. For example, instance annotation and label information on RDF, and overriding property tags and properties with no domain on RDFS were excluded. The excluded tags on the RDF representation were not related to instances information. Also, the overriding property tags and property tags where no domain was specified were identified as ontology representation overloading. They represented redundant information, hence not used for ontology class information. These exclusions had to be performed as a result of OntoJava compatibility issues, as mentioned, and had no effect on the final result, because they represented either no class/instance information or redundant information.

Also, for the ability to match ontological instances to run-time loaded email message instances as defined by the rules, the email ontology RDFS representation had to be duplicated. A specific prefix was added to class names to avoid redeclarations of classes.

Figure 5 shows how the naming difference were applied to the ontologies: the original ontology defines the class of email author as *Author* while the duplicated

```
<rdfs:Class rdf:about="&email;Author" rdfs:label="Author">
  <rdfs:subClassOf rdf:resource="&email;EmailAgent"/>
</rdfs:Class>
...
<rdfs:Class rdf:about="&email;InAuthor" rdfs:label="Author">
  <rdfs:subClassOf rdf:resource="&email;InEmailAgent"/>
</rdfs:Class>
```

Figure 5: Original and Duplicated Ontology

ontology defines the same class as *InAuthor*. The same approach was used to all other classes. By doing so, two identical ontology representations were defined, one containing the instances of relevant features identified beforehand, observed during the system requirement analysis, and another that is loaded with instances of the email messages. Both representations were available to the matching rules.

The next step was to run OntoJava with the RDFS, RDF, and RuleML files as input. The resulting Java code defines the rule engine and the main memory object database. They contain information about the concepts and properties specified by both original and duplicated ontologies, and also contain the RDF data given as input.

Afterward, an application was implemented and added at the top of the previous Java code. It handles the email messages one by one, originally managed within a Lotus Notes system and then converted to XML files using the Domino extended markup language (DXL) [50]. This application identifies email message features in the XML file, loads them into the duplicated ontology classes as instances, and then starts the rule engine. It outputs an array of recognized features according to the email instances and class instances matching rules. Figure 6 presents the entire email message feature identifier implementation process.

A set of email messages was used to compare the application output to the expected identified features of each of the messages. According to the knowledge

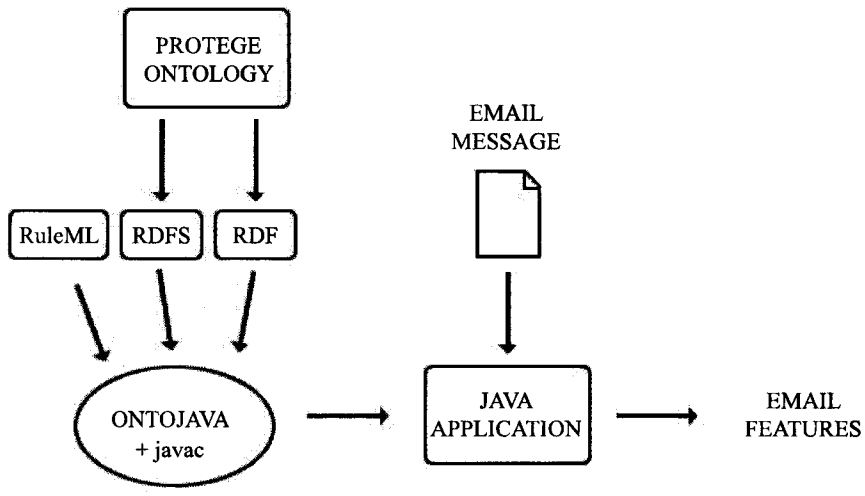

Figure 6: Email Features Identifier

base, represented in the RDF file, and to the rules provided, the system correctly identified the message features. This case study shows one example of how ontologies can be used to support information management applications both at development and run-time, providing a low cost development phase and an easy-to-maintain system structure.

CHAPTER 5

CONCLUSION AND FUTURE WORK

Ontological based hierarchical-organized concept models and term definitions and relations undoubtedly aid knowledge understanding in the complete life cycle of information systems. They implicitly define the rules that constrain the system complexities. They also explicitly establish the domain of knowledge to support machine reasoning, learning, and communication and interaction with humans or other machines in a location independent environment.

Although several methods for ontology engineering have been proposed, the ontology fundamental aspect is that they should be based on real world concepts and their properties. Indeed, ontology representation languages intend to model world reality in a machine and human understandable format, despite of the different syntax they may present. Furthermore, ontology tools and learning techniques exist to minimize the effort spent on handling ontologies. They support ontology activities such as creation, development, maintenance, editing, browsing, and integration, consistency and coherency checking, and knowledge extraction and annotation. Nevertheless, ontology-driven applications dictate evaluations for selecting the best-fit language and tool combination in accordance with their objectives and working environment, such as worker locations and available technologies and platforms.

The combination of ontologies and information systems aims a cognitive computing where the productivity and information value are more important than its raw computing speed. Ontologies not only reduce the system development cost

by promoting a better knowledge understanding, but also support a better communication between agents at run-time. They improve accuracy of information retrieval mechanism from a keyword matching approach to a meaningful analysis approach, and they facilitate information discovery and exchange due to knowledge structuring. Ontologies also provide the shared knowledge on the World Wide Web to support machine understanding of information, which was previously available only for human consumption. The idea is to convert machines into intelligent agents for complementing humans in areas of weak performance, such as fast processing of large volumes of information and analysis of large text for information recognition.

Although a semantically knowledge manipulation by intelligent agents is possible, problems such as lack of tool interoperability, lack of language translation methods or a standard language for information sharing, and, most important, lack of effort for information markup annotation must first be overcome. These problems prevent an ample practical view of an agent application and its advantages from being shown to enterprises and general public.

The feature identifier of email message example shows how ontologies may drive different aspects of the system from the development to execution time. It also shows how tools may be used to explore ontology knowledge and to minimize system development efforts. The knowledge model and its explicitly term definitions provided by ontologies give to the software engineering process a clear definition of the natural language specifications. This knowledge model also supports communication between agents, knowledge inference, and flow of information within the system.

Nevertheless, the absence of annotated markup information about each email message content restricted the semantic understanding, and a keyword matching approach was used to identify the features. As future work, specifically to the Email Classifier project, I would suggest the building of an ontology describing the domain

of knowledge based on the email classification categories. This ontology can be applied for natural language markup information extraction and semantic understanding for feature identification. Furthermore, ontologies can be applied to other ISRI projects, such as the medical record database. Ontologies can be used to translate natural language requests into a machine understandable format for use in information retrieval.

Clearly, ontologies provide a better semantic understanding of knowledge for both humans and machines in the information management area, which is essential in knowledge and information-based economies.

BIBLIOGRAPHY

[1] G. Aguado, A. Banon, J. Bateman, S. Bernardos, M. Fernandez, A. Gomez-Perez, E. Nieto, A. Olalla, R. Plaza, and A. Sanchez. OntoGeneration: reusing domain and linguistic ontologies for spanish text generation. In Proceedings of the 13th European Conference on Artificial Intelligence, Brighton, England, August 1998.

[2] J. Arpirez, O. Corcho, M. Fernadez-Lopez, and A. Gomez-Perez. WebODE: a scalable workbench for ontological engineering. In Proceedings of the International Conference on Knowledge Capture, October 2001.

[3] K. Baclawski, M. Kokar, R. Waldinger, and P. Kogut. Consistency checking of semantic web ontologies. In Proceedings of 1st International Semantic Web Conference ISWC 2002. Springer, June 2002.

[4] S. Bechhofer, I. Horrocks, C. Goble, and R. Stevens. OilEd: a reason-able ontology editor for the semantic web. In Proceedings of KI2001, Joint German/Austrian conference on Artificial Intelligence, number 2174 in Lecture Notes in Computer Science, pages 396–408, Vienna, September 2001. Springer-Verlag.

[5] T. Berners-Lee. Weaving the Web. Harper San Francisco, 1999.

[6] BizTalk. http://www.biztalk.org/.

[7] H. Boley, S. Tabet, and G. Wagner. Design rationale of ruleml: A markup language for semantic web rules. In Proceedings of the Semantic Web Working Symposium, July/August 2001.

[8] P. Borst, H. Akkermans, and J. Top. Engineering ontologies. International Journal of Human-Computer Studies, 46, February/March 1997.

[9] E. Castro. XML for the World Wide Web: Visual Quickstart Guide. Peachpit Press, 2001.

[10] B. Chandrasekaran, T. Johnson, and J. Smith. Task-structure analyses for knowledge modeling. Communications of the ACM, 35(9):124–137, September 1992.

[11] V. Chaudhri, A. Farquhar, R. Fikes, P. Karp, and J. Rice. Okbc: A programming foundation for knowledge base interoperability. In Proceedings of the American Association for Artificial Intelligence AAAI'98, July 1998.

[12] S. Cherry. Weaving a web of ideas. IEEE Spectrum, 39:65–69, September 2002.

[13] K. Church and L. Rau. Commercial applications of natural language processing. Communications of the ACM, 38, November 1995.

[14] R. Cost, T. Finin, A. Joshi, Y. Peng, C. Nicholas, I. Soboroff, H. Chen, L. Kagal, F. Perich, Y. Zou, and S. Tolia. Ittalks: a case study in the semantic web and daml+oil. IEEE Inteligent Systems, 17:40–47, January/February 2002.

[15] S. Decker, D. Fensel, F. van Harmelen, I. Horrocks, S. Melnik, M. Klein, and J. Broekstra. Knowledge representation on the web. http://www.ontoknowledge.org/oil/papers.shtml, 2000.

[16] V. Devedzic. Understanding ontological engineering. Communications of the ACM, 45(4):136–144, April 2002.

[17] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web. In Proceedings of the eleventh international conference on World Wide Web, pages 662–673. ACM Press, 2002.

[18] J. Domingue. Tadzebao and webonto: Discussing, browsing, and editing ontologies on the web. In Proceedings of Eleventh Knowledge Acquisition for Knowledge-Based Systems Workshop, April 1998.

[19] A. Duineveld, R. Stoter, M. Weiden, B. Kenepa, and V. Benjamins. Wondertools? a comparative study of ontological engineering tools. In Proceedings of Twelfth Workshop on Knowledge Acquisition, Modeling and Management, October 1999.

[20] A. Eberhart. Automatic generation of java/sql based inference engines from rdf schema and ruleml. In Proceedings of the International Semantic Web Conference, 2002.

[21] A. Eberhart. Smartguide: An intelligent information system basing on semantic web standards. In Proceedings of the International Conference on Artificial Intelligence, 2002.

[22] A. Eberhart. Survey of rdf data on the web. Technical report, International University in Germany, August 2002.

[23] ebXML. http://www.ebxml.org/.

[24] H. Eriksson, E. Berglund, and P. Nevalainen. Using knowledge engineering support for a java documentation viewer. In Proceedings of the 14th international conference on Software engineering and knowledge engineering, pages 57–64, July 2002.

[25] H. Eriksson and M. Musen. Metatools for knowledge acquisition. IEEE Software, 10:23–29, May 1993.

[26] J. Euzenat. Eight questions about semantic web annotations. IEEE Inteligent Systems, 17:55–62, March/April 2002.

[27] J. Euzenat. Research challenges and perspectives of the semantic web. IEEE Inteligent Systems, 17:86–88, September/October 2002.

[28] J. Everett, D. Bobrow, R. Stolle, R. Crouch, V. de Paiva, C. Condoravdi, M. van den Berg, and L. Polanyi. Ontology applications and design: Making ontologies work for resolving redundancies across documents. Communications of the ACM, 45, February 2002.

[29] A. Farquhar, R. Fikes, and J. Rice. The ontolingua server: a tool for collaborative ontology construction. http://www-ksl-svc.stanford.edu:5915/doc/project-papers.html, September 1996.

[30] D. Fensel and F. van Harmelen. Ontoknowledge. http://www.ontoknowledge.org/.

[31] D. Fensel, F. van Harmelen, and I. Horrocks. Oil: A standard proposal for the semantic web. http://www.ontoknowledge.org/oil/papers.shtml, November 1999.

[32] R. Fikes and D. McGuinness. An axiomatic semantics for rdf, rdf-s, and daml+oil. http://www.daml.org/2000/12/axiomatic-semantics.html, January 2001.

[33] C. Fillies, G. Wood-Albrecht, and F. Weichardt. A pragmatic application of the semantic web using semtalk. In Proceedings of the eleventh international conference on World Wide Web, pages 686–692. ACM Press, 2002.

[34] G. Fox. Xml and the importance of being an object. Computing in Science & Engineering, 4:96–98, May/June 2002.

[35] A. Gangemi, D. Pisanelli, and G. Steve. Ontology alignment: Experiences with medical terminologies. In Proceedings of Formal Ontology in Information Systems, pages 3–15, June 1998. Trento, Italy.

[36] M. Genesereth and R. Fikes. Knowledge Interchange Format, Version 3.0, Reference Manual. http://logic.stanford.edu/sharing/papers/kif.ps, June 1992.

[37] J. Golbeck, M. Grove, B. Parsia, A. Kalyanpur, and J. Hendler. New tools for the semantic web. In Proceedings of 13th International Conference on Knowledge Engineering and Knowledge Management, October 2002.

[38] A. Gomez-Perez and O. Corcho. Ontology languages for the semantic web. IEEE Intelligent Systems, 17:54–60, January/February 2002.

[39] M. Grado-Caffaro and M. Grado-Caffaro. The challenges that xml faces. Computer, 34:15–18, October 2001.

[40] M. Gruninger and J. Lee. Ontology applications and design. Communications of the ACM, 45(2):39–41, February 2002.

[41] N. Guarino. Formal ontology and information systems. In Proceedings of Formal Ontology in Information Systems, pages 3–15, June 1998. Trento, Italy.

[42] A. Gupta, B. Ludascher, and R. Moore. Ontology services for curriculum development in nsdl. In Proceeding of the second ACM/IEEE-CS joint conference on Digital libraries, Portland, Oregon, USA, 2002.

[43] U. Hahn and K. G. Marko. Joint knowledge capture for grammars and ontologies. In Proceedings of the international conference on Knowledge capture, pages 68–75. ACM Press, 2001.

[44] S. Handschuh and S. Staab. Languages & authoring for the semnatic web: Authoring and annotation of web pages in cream. In Proceedings of the eleventh international conference on World Wide Web, May 2002.

[45] J. Heflin and J. Hendler. Searching the web with shoe. In AAAI-2000 - Workshop on AI for Web Search, 2000.

[46] J. Heflin and J. Hendler. A portrait of the semantic web in action. IEEE Inteligent Systems, 116:54–59, March/April 2001.

[47] C. Holsapple and k. Joshi. A collaborative approach to ontology design. Communications of the ACM, 45(2):42–47, February 2002.

[48] I. Horrocks. The fact system. http://www.cs.man.ac.uk/ horrocks/FaCT/, June 1999.

[49] I. Horrocks, D. Fensel, J. Broekstra, M. Erdmann, C. Gobble, F. van Harmelen, M. Klein, S. Staab, R. Studer, and E. Motta. The ontology inference layer oil. http://www.ontoknowledge.org/oil/papers.shtml, August 2000.

[50] IBM. DXL Resources. http://www-10.lotus.com/ldd/dxl.

[51] Information Sciences Institute at University of Southern California. Loom project home page. http://www.isi.edu/isd/LOOM/LOOM-HOME.html.

[52] A. Kalyanpur, J. Hendler, B. Parsia, and J. Golbeck. Smore - semantic markup, ontology, and rdf editor. http://www.mindswap.org/papers/index.shtml.

[53] P. Karp, V. Chaudhri, and J. Thomere. Xol: An xml-based ontology exchange language. http://www.ai.sri.com/pkarp/xol/xol.html, 1999.

[54] R. Kent. Conceptual knowledge markup language: The central core. In Proceedings of the Twelfth Workshop on Knowledge Acquisition, Modeling and Management KAW'99, pages 16–21, October 1999.

[55] R. Kent. Ontologos. http://www.ontologos.org/, 2003.

[56] H. Kim. Predicting how ontologies for the semantic web will evolve. Communications of the ACM, 45(2):48–54, February 2002.

[57] M. Klein. Xml, rdf, and relatives. IEEE Inteligent Systems, 16:26–28, March/April 2001.

[58] I.-Y. Ko, ke Thia Yao, and R. Neches. Dynamic coordination of information management services for processing dynamic web content. In Proceedings of the Eleventh International Conference on World Wide Web, pages 355–365, May 2002.

[59] P. Kogut and W. Holmes. Aerodaml: Applying information extraction to generate daml annotations from web pages. In Proceedings of the First International Conference on Knowledge Capture K-CAP 2001 Workshop on Knowledge Markup and Semantic Annotation, October 2001.

[60] O. Lassila and R. Swick. Resource description framework RDF model and syntax specification. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/, February 1999.

[61] M. F. Lopez, A. Gomes-Perez, J. P. Sierra, and A. P. Sierra. Buildind a chemical ontology using methontology and the ontology design environment. IEEE Intelligent Systems, 14:35–46, Janary/February 1999.

[62] A. Maedche. Ontology Learning for the Semantic Web. Kluwer Academic Publishers, 2002.

[63] A. Maedche and S. Staab. Ontology learning for the semantic web. IEEE Intelligent Systems, 16:72–79, March/April 2001.

[64] K. Mahesh. Ontology development for machine translation: Ideology and methodology. Technical Report MCCS-96-292, Computing Research laboratory, New Mexico State University, 1996.

[65] D. McGuinness. Conceptual modeling for distributed ontology environments. In Proceedings of the Eighth International Conference on Conceptual Structures Logical, Linguistic, and Computational Issues ICCS 2000, August 2000.

[66] D. McGuinness, R. Fikes, J. Hendler, and L. A. Stein. Daml+oil: an ontology language for the semantic web. IEEE Inteligent Systems, 17:72–80, September/October 2002.

[67] D. McGuinness, R. Fikes, J. Rice, and S. Wilder. The chimæra ontology environment. In Proceedings of the Seventeenth National Conference on Artificial Intelligence AAAI 2000, July/August 2000.

[68] D. McGuinness, R. Fikes, J. Rice, and S. Wilder. An environment for merging and testing large ontologies. In Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning KR 2000, April 2000.

[69] R. Mihalcea and S. Mihalcea. Word semantics for information retrieval: Moving one step closer to the semantic web. In Proceedings of the 13th International Conference on Tools with Artificial Intelligence, pages 280–287, November 2001.

[70] M. Missikoff, R. Navigli, and P. Velardi. Integrated approach to web ontology learning and engineering. Computer, 35:60–63, November 2002.

[71] M. Musen. Ontology-oriented design and programming. Technical report, Stanford Medical Informatics, Stanford University, 2000.

[72] S. Nirenburg and V. Raskin. Ontological semantics, formal ontology, and ambiguity. In Proceedings of the international conference on Formal Ontology in Information Systems, Ogunquit, Maine, USA, October 2001.

[73] N. Noy and D. McGuinness. Ontology development 101: A guide to creating your first ontology. Technical Report SMI-2001-0880, Stanford Medical Informatics, Stanford University, 2001.

[74] N. Noy, M. Sintek, S. Decker, M. Crubezy, R. Fergerson, and M. Musen. Creating semantic web contents with protege-2000. IEEE Intelligent Systems, 16:60–71, March/April 2001.

[75] L. Obrst, R. Wray, and H. Liu. Ontological engineering for b2b e-commerce. In Proceedings of the International Conference on Formal Ontology in Information Systems, volume 2001, pages 117–126, 2001.

[76] B. Omelayenko. Learning of ontologies for the web: the analysis of existent approaches. In Proceedings of the International Workshop on Web Dynamics, January 2001. London, UK.

[77] V. Raskin, C. F. Hempelmann, K. E. Triezenberg, and S. Nirenburg. Ontology in information security: a useful theoretical foundation and methodological tool. In Proceedings of the 2001 workshop on New security paradigms, pages 53–59. ACM Press, September 2001.

[78] RosettaNet. http://www.rosettanet.org/.

[79] M. Sagheb-Tehrani. Information System's Roles and Responsabilities: Towards a Conceptual Model. Computer and Society Magazine, 32, September 2002.

[80] A. Sellen, R. Murphy, and K. Shaw. How knowledge workers use the web. In Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves, volume 4, pages 227–234, April 2002.

[81] P. Seltsikas. Managing global information strategy: Xerox, ltd. In Proceedings of the Twenty First International Conference on Information Systems, pages 791–800, December 2001.

[82] C. Sherman. Searchday. http://www.searchenginewatch.com/searchday/02/sd0617-update.html, June 2002.

[83] M. Smith, D. McGuinness, R. Volz, and C. Welty. Web ontology language OWL guide version 1.0. http://www.w3.org/TR/2002/WD-owl-guide-20021104/, November 2002.

[84] R. Stevens, C. Goble, I. Horrocks, and S. Bechhofer. Oiling the way to machine understandable bioinformatics resources. IEEE Transactions on Information Technology in Biomedicine, 6:129–134, June 2002.

[85] L. Stojanovic, N. Stojanovic, and R. Volz. Migrating data-intensive web sites into the semantic web. In Proceedings of the 17th symposium on applied computing, pages 1100–1107, March 2002.

[86] N. Stojanovic, A. Maedche, S. Staab, R. Studer, and Y. Sure. Seal: a framework for developing semantic portals. In Proceedings of the international conference on Knowledge capture, pages 155–162. ACM Press, 2001.

[87] Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, and D. Wenke. OntoEdit: Collaborative ontology development for the semantic web. In Proceedings of 1st International Semantic Web Conference ISWC 2002. Springer, June 2002.

[88] H. Suryanto and P. Compton. Discovery of ontologies from knowledge bases. In Proceedings of the international conference on Knowledge capture, pages 171–178. ACM Press, 2001.

[89] B. Swartout, R. Patil, K. Knight, and T. Russ. Toward distributed use of large-scale ontologies. In Proceedings of Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop, November 1996.

[90] W. Swartout. Future directions in knowledge based systems. ACM Computing Surveys, 28, December 1996.

[91] K. Taghva, J. Borsack, J. Coombs, A. Condit, S. Lumos, and T. Nartker. Ontology-based classification of email. Technical report, Information Science Research Institute, University of Nevada, Las Vegas, 2002.

[92] B. Thuraisingham, E. Hughes, and D. Allen. Dependable semantic web. In Proceedings of the Seventh International Workshop on Object-Oriented Real-Time Dependable Systems, 2002. WORDS 2002, pages 305–308, 2002.

[93] G. van Heijst. The Role of Ontologies in Knowledge Engineering. PhD thesis, Universiteit van Amsterdam, 1995.

[94] P. Velardi, P. Fabriani, and M. Missikoff. Using text processing techniques to automatically enrich a domain ontology. In Proceedings of the international conference on Formal Ontology in Information Systems, pages 270–284. ACM Press, 2001.

[95] World Wide Web Consortium. http://www.w3c.org/.

[96] V. Wuwongse, C. Anutariya, K. Akama, and E. Nantajeewarawat. Xml declarative description: A language for the semantic web. IEEE Intelligent Systems, 16:54–65, May/June 2001.

VITA

Graduate College
University of Nevada, Las Vegas

Renato de Freitas Marteleto

Local Address:
1321 Del Mar St #4
Las Vegas, Nevada 89119

Home Address:
Alameda das Tulipas 126
Granja das Hortências
36400-000 Conselheiro Lafaiete, MG
Brazil

Degrees:
Bachelor of Science, Computer Science, 1999
Universidade Federal de Ouro Preto, Minas Gerais - Brazil

Special Honors and Awards:
Best student in Computer Science, Class of 1999, UFOP

Publications:
Adilson P. Santos, Debora M. B. Paiva, Douglas R. Oliveira, Renato F. Marteleto. Building a System for Discipline Evaluations: A Dialogue of Education and Informatics. III Congress of Scientific Initiation and II National Congress of Researchers. São Carlos, SP. November 1998. Pages 219-231.

Renato de F. Marteleto (B) , Elton José da Silva (O) , Antônio Maria Claret de Gouveia (CO). A Multimedia Tutorial for the Divulgation of the Use of Steel in Civil Constructions. V Seminar of Scientific Initiation. Ouro Preto, MG. December 1998. Page 131.

Thesis Title:
The Role of Ontology in Information Management

Thesis Examination Committee:
Chairperson, Dr. Kazem Taghva, Ph. D.
Committee Member, Dr. Thomas Nartker, Ph. D.
Committee Member, Dr. Ajoy Datta, Ph. D.
Graduate Faculty Representitive, Dr. Jacimaria Batista, Ph. D.