ONTOLOGY & ITS TOOLS

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
IN
Computer Science & Engineering

Thapar University, Patiala

By:
Amrit Kaur
(80632001)

Under the supervision of:
Ms. Shalini Batra & **Ms. Rinkle Aggarwal**

Lecturer (SS) **Lecturer (SS)**

MAY 2008

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

Certificate

I hereby certify that the work which is being presented in the thesis entitled, "**Ontology and Its Tools**", in partial fulfillment of the requirements for the award of degree of Master of Engineering in Computer Science & Engineering submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Ms Shalini Batra* and *Ms. Rinkle Aggarwal.*

The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

(Amrit Kaur)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(**Ms. Shalini Batra** & **Ms.Rinkle Aggarwal)**
Lecturer (SS),
Computer Science and Engineering Department,
Thapar University,
Patiala.

Countersigned by

(Dr. SEEMA BAWA)
Professor & Head,
Computer Science & Engineering Department,
Thapar University,
Patiala.

(Dr. R.K.SHARMA)
Dean (Academic Affairs),
Thapar University,
Patiala.

Acknowledgement

It is a great pleasure for me to acknowledge the guidance, assistance and help I have received from **Ms**. **Shalini Batra** Lecturer (SS) & **Ms**. **Rinkle Aggarwal**, Lecturer (SS), Computer Science and Engineering Department. I am thankful for their continual support, encouragement, and invaluable suggestions. They not only provided me help whenever needed, but also the resources required to complete this thesis report on time.

I am also thankful to **Dr. Seema Bawa**, Head, Computer Science and Engineering Department for her kind help and cooperation.

I would also like to thank all the staff members of Computer Science and Engineering Department for providing me all the facilities required for the completion of my thesis work.

I would like to say thanks for support of my Classmates. I want to express my appreciation to every person who contributed with either inspirational or actual work to this thesis.

I am highly grateful to my parents and brother for the inspiration and ever encouraging moral support, which enabled me to pursue my studies.

Amrit Kaur

Abstract

Ontology is a shared and common understanding of some domain that can be communicated between people and application systems. Ontological commitments are agreements to use the shared vocabulary in a coherent and consistent manner. It defines the vocabulary with which queries and assertions are exchanged among domains. Ontologies have been widely accepted as the primary method of representing knowledge in the Semantic Web. A number of ontology development tools like Protégé-2000, OiLed, Apollo, Ontosaurus, OntoStudio etc. are available for knowledge elicitation in the fulfillment of dream of new web i.e. Semantic Web.

In this thesis the intent is to analyze the features provided by various tools by implementing the common ontology among all the ontology development tools. Different parameters used to measure the effectiveness of a tool include hierarchical view, GUI support, plug-ins interoperability with other tool, user friendliness, consistency check, etc. The final analysis has been depicted via a bar graph.

Table of Contents

Certificate---------- i

Acknowledgement---------- ii

Abstract---------- iii

Table of Contents---------- iv

List of Figures & Tables---------- vii

Chapter 1 Introduction---------- 1

1.1 Ontology---------- 1

1.2 Ontology Life Cycle---------- 1

1.3 Types of Ontology---------- 2

1.3.1 Level of Formality---------- 2

1.3.2 Extent of Explication 3

1.4 Ontology-based Architectures 4

1.4.1 The Ontology Layer 4

1.4.2 The Middleware Layer 4

1.4.3 The Application Layer 6

1.5 Advantages of Developing Ontology 6

1.6 Beneficial Applications of Ontology 7

1.6.1 Semantic Web 7

1.6.2 Knowledge Management 7

1.6.3 Interoperability 8

1.6.4 Information Retrieval 8

1.6.5 Service Retrieval 9

1.7 Components of Ontology 9

1.8 Types Of Ontology---------- 10

1.9 Ontology and Semantic Web---------- 11

1.9.1 Structure Of Semantic Web---------- 11

Chapter 2 Literature Review ---------- 13

2.1 Evaluation Framework Used To Compare The Ontological ---------- 13
Engineering Tool

2.2 Ontology Development Tool---------- 14

2.2.1 Protégé-2000---------- 14

2.2.2 OiLed---------- 16

2.2.2.1 Components of OiLed---------- 16

2.2.2.1.1 DL---------- 16

2.2.2.1.2. DAML+OIL---------- 17

2.2.2.1.3 FaCT---------- 17

2.2.3 Apollo---------- 17
2.2.4 Ontosaurus---------- 18
2.2.5 OntoStudio---------- 18

Chapter 3 Problem Statement---------- 20

3.1 Problem Definition ---------- 20

3.2 Methodology---------- 20

Chapter 4 Implementation---------- 21

4.1 Implementation in Protégé -2000---------- 22

4.2 Implementation in OilLed---------- 25

4.3 Implementation in Apollo---------- 29

4.4 Implementation in Ontosaurus---------- 31

4.5 Implementation in OntoStudio---------- 34

Chapter 5 Conclusion and Future---------- 42

5.1 Discussion---------- 42

5.1.1 Protégé-2000---------- 42

5.1.2 OiLed---------- 43

5.1.3 Apollo---------- 43

5.1.4 Ontosaurus---------- 44

5.1.5 OntoStudio---------- 45

5.2 Conclusion---------- 45

5.2.1 Merged Bar Diagram---------- 45

5.3 Future Scope---------- 47

References---------- 48

List of Publications---------- 52

List of Figures and Tables

---

1 Figure 1.1 Ontology life cycle---------- 2

---

2 Figure 1.2 Classification of types of ontologies, based on the level of ----formality 3

3 Figure 1.3 A generic architecture of ontology-based applications---------- 5

4 Figure 1.4 The Semantic Web layers---------- 11

5 Figure 4.1 Ontology of Inventory---------- 21

6 Figure 4.2 Protégé-2000---------- 23

7 Figure 4.3 Queries in Protégé-2000---------- 24

8 Figure 4.4 OilEd---------- 26

9 Figure 4.5 Consistency checking in OiLed---------- 27

10 Figure 4.6 Consistent Ontology---------- 28

11 Figure 4.7 Apollo---------- 30

12 Figure 4.8 Ontosaurus---------- 32

13 Figure 4.9 PowerLoom---------- 33

14 Figure 4.10 Ontology Navigator---------- 35

15 Figure 4.11 Entity property view of concept---------- 36

16 Figure 4.12 Entity property view of property---------- 37

17 Figure 4.13 Entity property view of instance---------- 38
---

18 Figure 4.14 Entity property view of rules---------- 39
---

19 Figure 4.15 Graph visualize of concept “Information” ---------- 40
---

20 Figure 4.16 Queries in Ontostudio---------- 41
----

21 Figure 5.1 Bar graph of Protégé – 2000---------- 42
----

22 Figure 5.2 Bar graph of OiLed---------- 43
---

23 Figure 5.3 Bar graph of Apollo---------- 44
---

24 Figure 5.4 Bar graph of Ontosaurus---------- 44
---

25 Figure 5.2 Bar graph of OntoStudio---------- 45
---

26 Figure 5.6 Comparison of ontological tools. ---------- 46
---

27 Table 1.1 Comparison of ontological tools---------- 47
--

Chapter- 1
INTRODUCTION

1.1 Ontology

This chapter introduces the basic concepts of ontology engineering. Its main goal is to provide basic understanding of ontologies, which are the basis of this work. This chapter is partly based on previously published work [30].

Ontology is formal and explicit specification of a shared conceptualization. 'Conceptualization' refers to an abstract model of phenomena in the world by having identified the relevant concepts of those phenomena. 'Explicit' means that the type of concepts used, and the constraints on their use are explicitly defined. 'Formal' refers to the fact that the ontology should be machine readable. 'Shared' reflects that ontology should capture consensual knowledge accepted by the communities. Ontology is a key factor for enabling interoperability in the Semantic Web [17].

The main problem with the use of a shared vocabulary according to a specific conceptualization of the world is that much of the information remains implicit. Ontologies have been set out to overcome the problem of implicit and hidden knowledge by making the conceptualization of a domain explicit.

1.2 Ontology Life Cycle

The design of an ontology is an iterative maturing process. This means the ontology will come to full development i.e. become mature by evolving through intermediate states to reach a desired or final condition. As soon as the ontology becomes important, the ontology engineering process has to be considered as a project, and therefore project management methods must be applied. [22] recognized that planning

and specification are important activities. The authors list the activities that need to be performed during the ontology development process. As seen in figure 1.1 the life of an ontology moves on through the following states: specification, conceptualization, formalization, integration, implementation, and maintenance. Knowledge acquisition, documentation and evaluation are support activities that are carried out during the majority of these states.

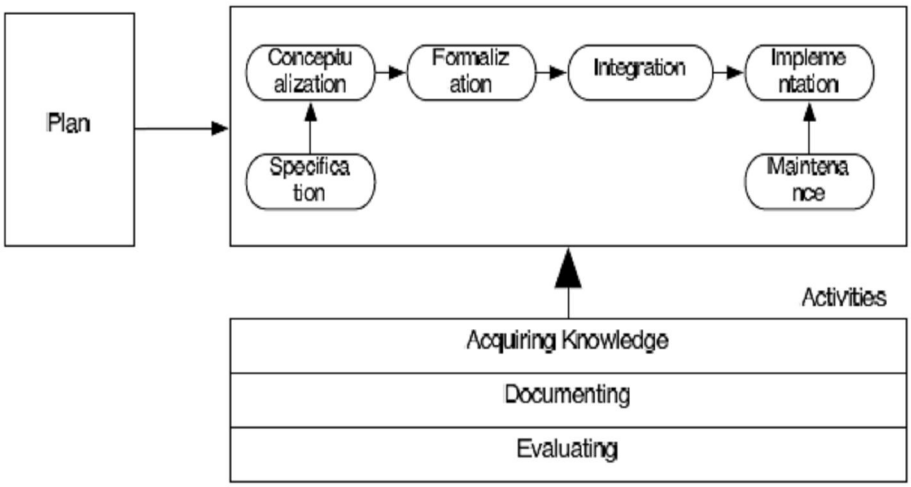


Figure 1.1 Ontology Life Cycle

1.3 Types of Ontologies

There are different ways in which an ontology may explicate a conceptualization and the corresponding context knowledge. This may range from a purely informal natural language description of a term corresponding to a glossary up to strictly formal approaches with the expressive power of full first order predicate logic or even beyond. There exist several ways to categorize types of ontologies. Jasper and Uschold distinguish two ways in which the mechanisms for the conceptualization of domain knowledge by an ontology can be compared [29].

1.3.1 Level of Formality

One of the well-known divisions to categorize types of ontologies is by their level of formality: ranging from a list of terms to concepts having relations and axioms. Figure 1.2 summarizes these distinctions. It also includes other terminologies including lightweight and heavyweight ontologies.

1.3.2 Extent of Explication

The other comparison criterion is the extent of explication that is reached by the ontology. This criterion is very much connected with the expressive power of the specification language used. The least expressive specification of an ontology consists of an organization of terms in a network using two-placed relations. More expressive ontology languages like RDF schema contain class definitions with associated properties that can be restricted by so called constraint properties. However, default values and value range descriptions are not expressive enough to cover all possible conceptualizations. A greater expressive power can be provided by allowing classes to be specified by logical formulas. These formulas can be restricted to a decidable subset of first order logic. This is the approach of description logic [21]. Nevertheless there are also approaches allowing for more expressive description. In Ontolingua, for example, classes can be defined by arbitrary KIF-expressions. Beyond the expressiveness of first order predicate logic there are also special purpose languages that have an extended expressiveness.

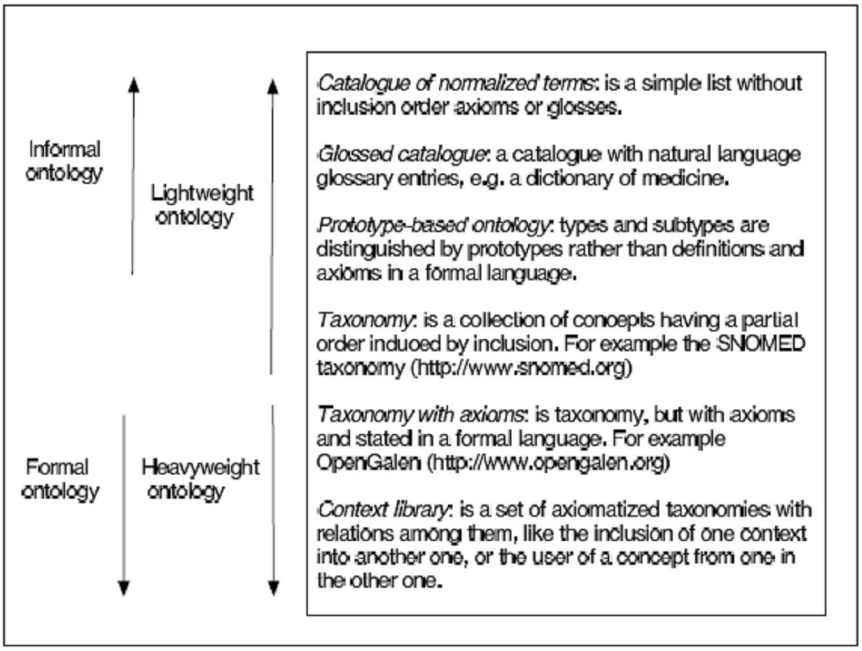


Figure 1.2 Classification of types of ontologies, based on the level of formality (Adopted from [25]).

1.4 Ontology-based Architectures

Effective and efficient work with the Semantic Web in general and ontology in particular, must be supported by advanced tools enabling the full powers of the technology. [28] suggests to review the different tools in an ontology-based architecture instead of focusing on individual tools separately. In fact, many of the current ontology engineering environments provide a broad range of services rather than only one service. Figure 1.3 sketches a decomposed design of ontology-based applications, highlighting the different elements that will contribute to the success of the applications.

1.4.1 The Ontology Layer

The components in this layer serve the common goal of the acquisition of ontologies. In particular, it requires the following elements:

- Ontology extraction applies Natural Language Processing (NLP) techniques in domain documents to determine the most relevant concepts and their relationships in a domain.

- Ontology learning is a more generic term applied to all bottom-up approaches of ontology acquisition that start from a given set of data that reflects the human communication and interaction process.
- Ontology annotation tool is used to create an instance set on the basis of an existing ontology.
- Ontology editor is an application intended for creating or editing ontologies manually by a knowledge engineer.
- Ontology evaluation tools aim at improving the quality of ontologies.
- Ontology mapping, aligning and merge tools provide support for users to find similarities and differences between sources ontologies.

1.4.2 The Middleware Layer

Ontology middleware plays the role of hiding the ontology layer in systems and providing advanced services to applications such as ontology management, storage, query and inference.

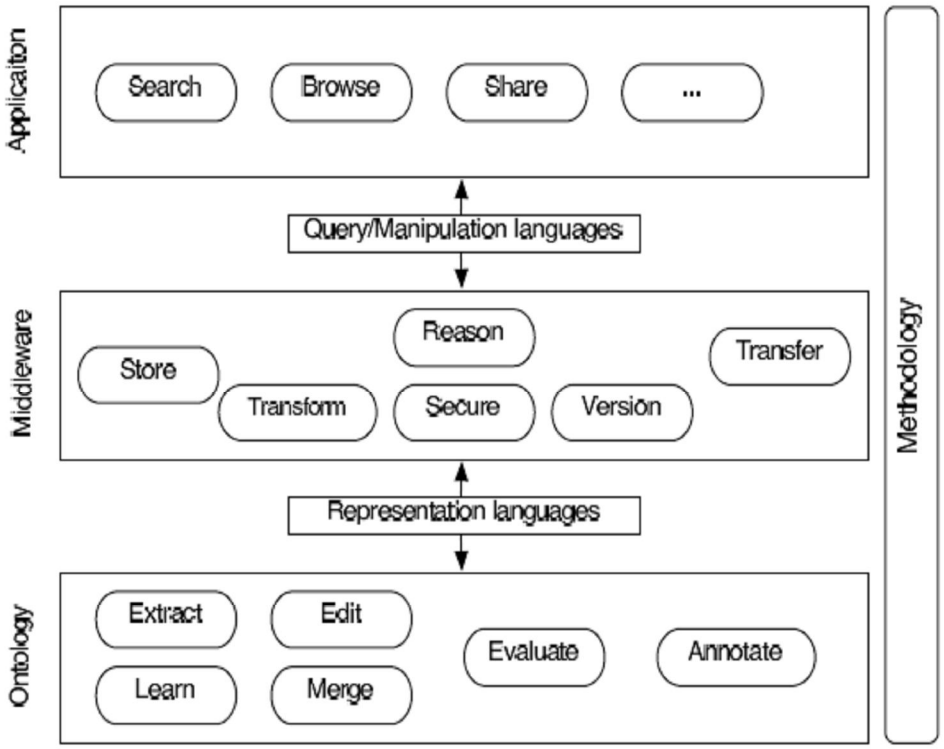


Figure 1.3 A generic architecture of ontology-based applications, (Adopted from [26])

- Ontology storage facilities (also called ontology server) provide database like functionality for the persistent storage and selective retrieval of ontologies.
- The goal of query is to provide high level access to the ontology through questions formulated in a query language that is easy both for people to write and for machines to evaluate.
- Inference engines process the knowledge structures captured in ontologies to reason implicit knowledge in the ontologies.
- Ontology management is the set of techniques that are necessary to efficiently use multiple variants of ontologies, and it includes issues like version control, security, access right and trust management etc.
- Ontology transfer refers to the ability of middleware to connect ontology servers over the network.

1.4.3 The Application Layer

The application layer is the home of ontology-based applications, and software which supports users to access, organize, exchange and aggregate information through the use of ontologies [29]. Example applications are:

- Ontology-based search and browsing support different information seeking modes for accessing large collection of instance sets or data items referred by the ontology.
- Ontology-based sharing provides interoperability between different systems through the use of referring to common ontology.

1.5 Advantages of Developing Ontology

Major benefits of developing ontology are: -

1 **Sharing common understanding of the structure of information among people or software agents**: It is one of the most common goals in developing ontologies. For example several different web sites contain medical information or provide e-commerce services. These web sites share and publish the same underlying ontology of the terms they use, and then

computer agents can extract and aggregate information from different sites. The agents can use this aggregated information to answer user queries or as input data to other applications.

2 **Enabling reuse of domain knowledge:** It is one of the driving forces behind recent surge in ontology research. For example, models for many different areas need to represent the opinion of time. This representation includes the planning of time intervals, points in time, relative measures of time, etc. If one group of researchers develops such ontology in detail, others can simply reuse it for their area. To build a large ontology need to integrate several existing ontologies describing portions of the large domain. General ontology can be reused and extend it to describe desired ontology.

3 **Making explicit domain assumptions:** Explicit specifications of domain knowledge are useful for new users who want to learn the specific area.

4 **Separating the domain knowledge from the operational knowledge**: It is another common use of ontologies. It describes a task of configuring a product from its components according to a required specification and implements.

5 **Analyzing domain knowledge:** Analyzing is possible once a declarative specification of the terms is available. Formal analysis of terms is valuable when attempting to reuse existing ontologies and extending them for solving problem purposes [2].

1.6 Beneficial Applications of Ontology

In [20], it is stated that ontologies are used in e-commerce to enable machine based communication between buyers and sellers, vertical integration of markets and description reuse between different marketplaces. Search engines also use ontologies to find pages with words that are syntactically different but semantically similar. In particular, the following area will benefit from the use of ontologies:

Semantic Web

1.6.1 Semantic Web

The Semantic Web aims at tackling the growing problems of traversing the expanding web space, where currently most web resources can only be found by syntactical matches. The Semantic Web relies heavily on formal ontologies that structure underlying data for the purpose of comprehensive and transportable machine understanding. They properly define the meaning of data and metadata. In general, one may consider the Semantic Web more as a vision than a concrete application.

1.6.2 Knowledge Management

Knowledge management deals with acquiring, maintaining and accessing knowledge of an organization. The technologies of the Semantic Web build the foundation to move from a document oriented view of knowledge management to a knowledge pieces oriented view where knowledge pieces are connected in a flexible way. Intelligent push service, the integration of knowledge management and business process as well as concepts and methods for supporting the vision of ubiquitous knowledge are urgently needed. Ontologies are the key means to achieve this functionality. They are used to annotate unstructured information with semantic information, to integrate information and to generate user specific views that make knowledge access easier.

1.6.3 Interoperability

An important application area for ontology is the integration of existing systems. In order to enable machines to understand each other we need to explicate the context of each system in a formal way. Ontologies are then used as inter-lingua for providing interoperability since they serve as a common format for data interchange [25].

1.6.4 Information Retrieval

Common information retrieval techniques either rely on a specific encoding of available information or simple full-text analysis. Both approaches suffer from problems like the query entered by the user may not be completely consistent with the vocabulary of the documents and the recall of a query will be reduced since related information with slightly different encoding is not matched. Using ontology to

explicate the vocabulary may overcome some of the problems. When used for the description of available documents as well as for query formulation, an ontology serves as a common basis for matching queries against potential results on a semantic level. In some cases, the ontology can also be directly used as a user interface to navigate through available document. Information retrieval benefits from the use of ontologies, because ontologies help to decouple description and query vocabulary and increase retrieval performance [24].

1.6.5 Service Retrieval

The ability to rapidly locate useful online service (e.g. software applications, software components, process models, or service organizations), as opposed to simple useful documents, is becoming increasingly critical in many domains [23].

1.7 Components of Ontology

Major components of ontology are:

1. Classes - Classes represent concepts, which are taken in a broad sense. For instance, in the traveling domain, concepts are: locations (cities, villages, etc.), lodgings (hotels, camping, ec.) and means of transport (planes, trains, cars, ferries, motorbikes and ships). Classes in the ontology are usually organized in taxonomies through which inheritance mechanisms can be applied. We can represent a taxonomy of entertainment places (theater, cinema, concert, etc.) or travel packages (economy travel, business travel, etc.).

2. Relations – Relations represent a type of association between concepts of the domain. They are formally defined as any subset of a product of n sets, that is: R C1 x C2 x ... x Cn. Ontologies usually contain binary relations. The first argument is known as the domain of the relation, and the second argument is the range. For

instance, the binary relation arrivalPlace has the concept Travel as its domain and the concept Location as its range.

3. Functions - It is a special case of relations in which the n-the element of the relationship is unique for the n-1 preceding elements. Example of function is Price-of-a-used-car can define the calculation of the price of the second-hand car on the car-model, manufacturing data and kilometers

4. Axioms – According to Gruber (1993), formal axioms serve to model sentences that are always true. They are normally used to represent knowledge that cannot be formally defined by the other components. In addition, formal axioms are used to verify the consistency of the ontology itself or the consistency of the knowledge stored in a knowledge base. Formal axioms are very useful to infer new knowledge. An axiom in the traveling domain would be that it is not possible to travel from the America to Europe by train.

5. Instances – Instances are used to represent elements or individuals in an ontology. An example of instance of the concept AA7462 is the flight AA7462 that arrives at Seattle on February 8, 2006 and costs 300 (US Dollars, Euros, or any other currency).

1.8 Types of Ontologies

1. Knowledge Representation ontologies
    - This ontology captures the representation primitives used to formalize knowledge in a standard model, e.g. include Frame-Ontology
2. General ontologies
    - This ontology is related to common things, events, time, space, etc. eg. meter and inch exchange table.
3. Meta-ontologies
    - This ontology is reusable among domains. ‘Go’ ontology is commonly used by all persons in medical fields for disease diagnoses.
4. Domain ontologies

- This ontology describes the vocabularies about the concepts in a domain. Theory or elementary principles governing the domain is a good example of this.

5. Task ontologies
   - This ontology describes a systematic vocabulary of the terms used to solve problems associated with tasks that may or may not from the same domain. For example scheduling task ontology.
6. Application ontology
   - This ontology describes necessary knowledge for modeling a particular domain.

1.9 Ontology and Semantic web

1.9.1 Structure of the Semantic Web

Currently, the World Wide Web consists of documents written in HTML. This makes the web readable for humans, but since HTML has limited ability to classify the blocks of text apart from the roles they play, the Web in its current form is very hard to understand. The purpose of the Semantic web is to add a layer of descriptive technologies to web pages so they become readable [6].

The Semantic Web is implemented in the layers of Web technologies and standards. The layers are presented in Figure 1.4

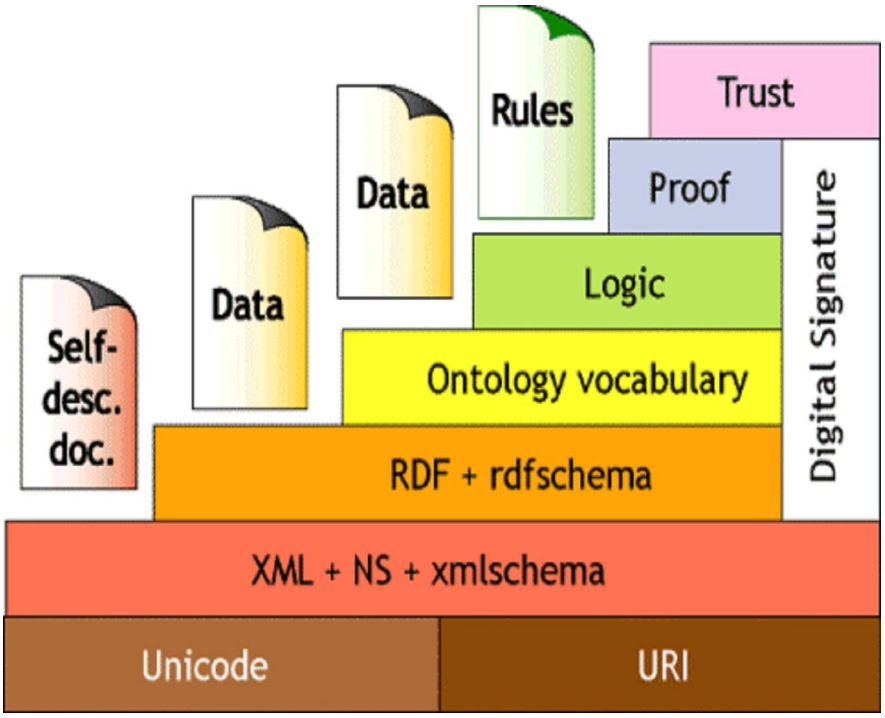


Figure 1.4: **The Semantic Web layers [16]**

The **Unicode and Uniform Resource Identifier** (URI) layers make sure that international character sets are used and provide means for identifying the objects in the Semantic Web. The most popular URI's on the World Wide Web are Uniform Resource Locaters (urls).

The **Extensible Markup Language** (XML) layer with namespace and schema definitions make sure the Semantic Web definitions can integrate with the other XML based standards [3]. XML provides a surface syntax for structured documents, but imposes no semantic constraints on the meaning of these documents. XML Schema is a language for restricting the structure of XML documents [7].

The **Resource Description Framework (**RDF) Schema is a simple type modeling language for describing classes of resources and properties between them in the basic RDF model. It provides a simple reasoning framework for inferring types of resources.

The **ontology layer** supports the evolution of vocabularies as it can define relations between the different concepts. In this layer knowledge is expressed as descriptive statements, stating some relationship exists between one thing and another.

A **digital signature** is an electronic signature that can be used to authenticate the identity of the sender of a message or the signer of a document. The Digital Signature layer ensures that the original content of the message or document is unaltered.

The top layers **Logic, Proof and Trust** are currently being researched and simple application demonstrations are being constructed. The Logic layer enables the writing of rules while the Proof layer executes the rules and evaluates together with the Trust layer mechanism for applications whether to Trust the given proof or not.

Chapter- 2

LITERATURE REVIEW

Present day research on semantic web is concentrated on two main things: RDF and ontology. Ontology is a theory about the nature of existence, of what types of things exist. It is a study of such research. Ontology is a document or file that formally defines the relations among terms. The most typical kind of ontology for the web has taxonomy and a set of inference rules. The taxonomy defines classes of objects and relations among them. There are various type of Ontology engineering tools. Each tool have its own advantage and disadvantage.

2.1 Evaluation Framework Used to Compare the Ontological Engineering Tools

Set of criteria used for comparing ontology development tools can be divided into the following groups:

1. General description of the tools: This includes information about developers, releases and availability.

2. Software architecture and tool evolution: Software architecture includes information about the tool design e.g. standalone, client/server, n-tier application etc. It also explains how the tool can be extended with other functionalities and modules. It describe how ontologies are stored and if there is any backup management system.

3. Interoperability with other ontology development tools and languages: This information describes the interoperability capabilities of the tool. Tool's interoperability with other ontology tools can be recognized by functionalities like merging, annotation, storage, inference, etc.

4. Knowledge representation: This feature is relevant in order to know what and how knowledge can be modeled in the tool.

5. Inference services attached to the tool: This feature analyze whether the tool has a built-in inference engine or it can use other external inference engines. It also analyze if the tool performs consistency checking.

6. Usability: It analyzes the existence of graphical editors for the creation of concept, taxonomies and relations, the ability to prune these graphs and the possibility to perform zooms of parts of it.

2.2 Ontology Development Tools

Ontology development tools describe the environments that are used for building a new ontology from scratch or reusing existing ontologies. Apart from the common edition and browsing functionality, this tool usually include ontology documentation, ontology exportation and importation from different formats, graphical views of the ontologies built, ontology libraries, attached inference engines, etc. Various ontological tools available are Protégé-2000, Apollo, Oiled, Ontosaurus, OntoStudio, etc.

2.2.1 Protégé – 2000

Protégé is one of the most widely used ontology development tool. It is free and open source. It provides a graphical, interactive and knowledge-base ontology development environment. In addition to the concept of class, relations, slots, instance and facets it also includes extensibility and scalability of ontologies. A class is represented in a hierarchical manner and also supports multiple inheritances. New functionality can be added by creating appropriate plug-ins. Protégé-2000 is the latest tool in an established line of tools developed at Stanford University for Knowledge acquisition.
It contains the following ontology components [9]:

1. **Classes** represent entities of the domain. They can be concrete or abstract, that is, classes can have direct instances or not. Slot constraints can be attached to the class, and determine constraints on the slot values of the class instances. The taxonomic relation subclass-of can be defined between classes, allowing multiple classifications. The relation subslot-of can be defined between slots.

2. **Slots** represent interactions between domain objects or characteristics of class instances. Slots have value types minimum and maximum. Cardinalities have minimum and maximum values in case of numeric slots. It is defined independently of the classes to which they are attached. There are two ways for slots to attach.

    - **Template slot**. It describe properties of the instances of the class to which it is attached or interactions between instances of the class and

instances of other classes. It is inherited by the subclasses and instances of the class.

- **Own slot**. It describes properties of the class itself or interactions between the class and other classes taking some values in the class. These values are not inherited by its subclasses or by its instances. The template slots of a class become own slots in its instances.

3. **Facets** define slot constraints. The features of slots defined before value types, minimum and maximum cardinalities, etc. are built-in facets in the Protégé-2000 knowledge model. However, new facets can be defined for them.

4. **Instances** of classes define individuals in the domain. They contain own slots, which are the template slots of the classes from which they are instances with their corresponding values. In Protégé-2000 instances belongs to only one class.

5. **Constraints** are first order logic sentences used to check constraints in the ontology. They are created with Protégé Axiom Language.

The query can be formulated to the knowledgebase. The Queries tab allows querying project and locating all instances that match the criteria that have been specified. To run queries add the class, slot and string in it [10].

2.2.2 OilLed

OILEd is a graphical ontology editor developed by the University of Manchester. The knowledge model of OiLed is based on that of DAML+OIL, although this is extended by the use of a frame-like presentation for modeling. Thus OiLed offers a familiar frame-like paradigm for modeling while still supporting the rich expressiveness of DAML+OIL where required. Classes are defined in terms of their super classes and property restrictions, with additional axioms capturing further relationships such as disjoint ness. The expressive knowledge model allows the use of complex composite descriptions as role fillers.

The main task that OILEd is targeting is editing ontologies or schemas. Additional functionality is provided that allows the definition of individuals, primarily intended for the definition of nominal. A key aspect of OiLed behavior is the use the FaCT reasoner to classify ontologies and check consistency.

This allows describing ontology classes and having the reasoner determine the appropriate place in the hierarchy for the definition. This tool reads and writes concept hierarchies in pure RDF and renders ontology definitions as HTML for browsing [8].

2.2.2.1 Components of OiLed

Following are the componenets of OiLed:

2.2.2.1.1 DL

Description logics (DL) are a family of knowledge representation languages which can be used to represent the terminological knowledge of an application domain in a structured way. The name description logic refers to concept descriptions used to describe a domain and the logic-based semantics. Description logic is designed as an extension to frames and semantic networks, which were not equipped with formal logic-based semantics.

Description Logics is an important powerful class of logic-based knowledge representation languages. Description logic has become a keystone of the Semantic Web for its use in the design of ontologies. Ontology Web Language (OWL) is based on description logic.

2.2.2.1.2. DAML+OIL

DAML is DARPA Agent Markup Language and DARPA is Defense Advanced Research Projects Agency for the semantic web. DAML+OIL are syntax layered on RDF and XML that can be used to describe sets of facts for making up ontology.

OIL (Ontology Inference Layer or Ontology Interchange Language) can be regarded as an Ontology infrastructure for the Semantic Web. OIL is based on concepts developed in DL. DAML+OIL use RDF namespaces to organize and assist with integration of arbitrarily much different and incompatible ontology. The World Wide Web has developed the Extensible Markup Language (XML) which allows information to be more accurately described using tags. However, XML has a limited capability to describe the relationships schemas or ontologies with respect to objects. The use of ontologies provides a very powerful way to describe objects and their relationships to other objects. The DAML language is being developed as an extension to XML and the Resource Description Framework (RDF) [11].

2.2.2.1.3 FaCT

The FaCT (Fast Classification of Terminologies) is used with OiLed for consistency checking. FaCT is written in Common Lisp and has been run successfully with several commercial programs [12].

2.2.3 Apollo

Apollo is a user friendly ontology development application. Apollo supports all the basic primitives of knowledge modelling: ontologies, classes, instances, functions and relations. Full consistency checking is done while editing, for example, detecting the use of undefined classes. Apollo has its own internal language for storing the ontologies, but can also export the ontology into different representation languages, as required by the user. Apollo is implemented in Java [13].

2.2.4 OntoSaurus

OntoSaurus is a Web Browser for Loom knowledge bases. Loom is a language and environment for constructing intelligent applications. It provides a graphical hyperlinked interface to Loom knowledge bases. OntoSaurus also provides limited editing facilities for Loom knowledge bases. However, its main function is browsing ontologies.

The heart of Loom is a knowledge representation system that is used to provide deductive support for the declarative portion of the Loom language. Declarative knowledge in Loom consists of definitions, rules, facts, and default rules. A deductive engine called a classifier utilizes forward-chaining, semantic unification and object-oriented truth maintenance technologies in order to compile the declarative knowledge into a network designed to efficiently support on-line deductive query processing [14].

2.2.5 Ontostudio

OntoStudio is the professional developing environment for ontology-based solutions. It combines in a unique modeling tool for ontologies and rules with components for the integration of heterogeneous data sources. Through its modular design OntoStudio can be enriched with self-developed modules and be customized to your personal needs. As ontology-language, OntoStudio supports OWL, RDF and F-Logic for the logic-based processing of rules. Ontostudio is multiple window editors [].

OntoStudio comes along with many connectors to databases, documents, file-systems, applications and web-services. A fine access to single elements of a database is enabled through an enhanced database schema import. Every single table it can be defined, which columns. With the support of the integrated mapping-tool, heterogeneous data sources can be connected. By using rules and filters it is possible to resolve conflicts of names, values and structures among different sources and therefore achieve a true integration of all contents. Thus it provides a single view through the ontology to all connected databases and shows real time information.

OntoStudio supports the creation of rules by various functionalities. Rules can easily be created by users with the help of a graphical rule modeling editor. They can be easily handled, extended and also tested. An explanation component shows how results were derived and a rule debugger shows the progression of an inquiry graphically. Both support the modeler to find possible exceptions and increase the transparency while modeling complex domains. Regression tests allow the building of defined and repeatable test cases [15].

Chapter- 3

PROBLEM STATEMENT

3.1 Problem Definition

Ontology editors are important tools to support the elaboration of ontologies. They facilitate development and management of ontologies, the definition and modification of concepts, properties, axioms and restrictions, even some of them enable inspection and browsing of ontologies.

Tool support is important both for the ontology development process and for the ontology usage in applications, such as electronic commerce, knowledge management, the Semantic Web, etc.

Various ontology tools are available in different language. Depending on the knowledge base of the user, problem domains and outputs desired, different ontology development tools can be used. The main aim of the thesis is to analyze the features provided by the available tools like Protégé-2000, OiLed, Apollo, Ontosaurus, etc. by taking one common ontology and designing it on various ontology development tools. Different parameters are used to measure the effectiveness of a tool include hierarchical view, graph view, plug-ins interoperability with other tool, user friendly, consistency check etc and they all have been depicted graphically via bar graph.

3.2 Methodology

- Study and download all the available ontology development tools.
- Design Ontology in the available Ontology editors.
- Develop the classes, properties and relation among them to reflect the appropriate knowledgebase.
- Analyze the feature and benefits of using various tools.

Chapter- 4

IMPLEMENTATION

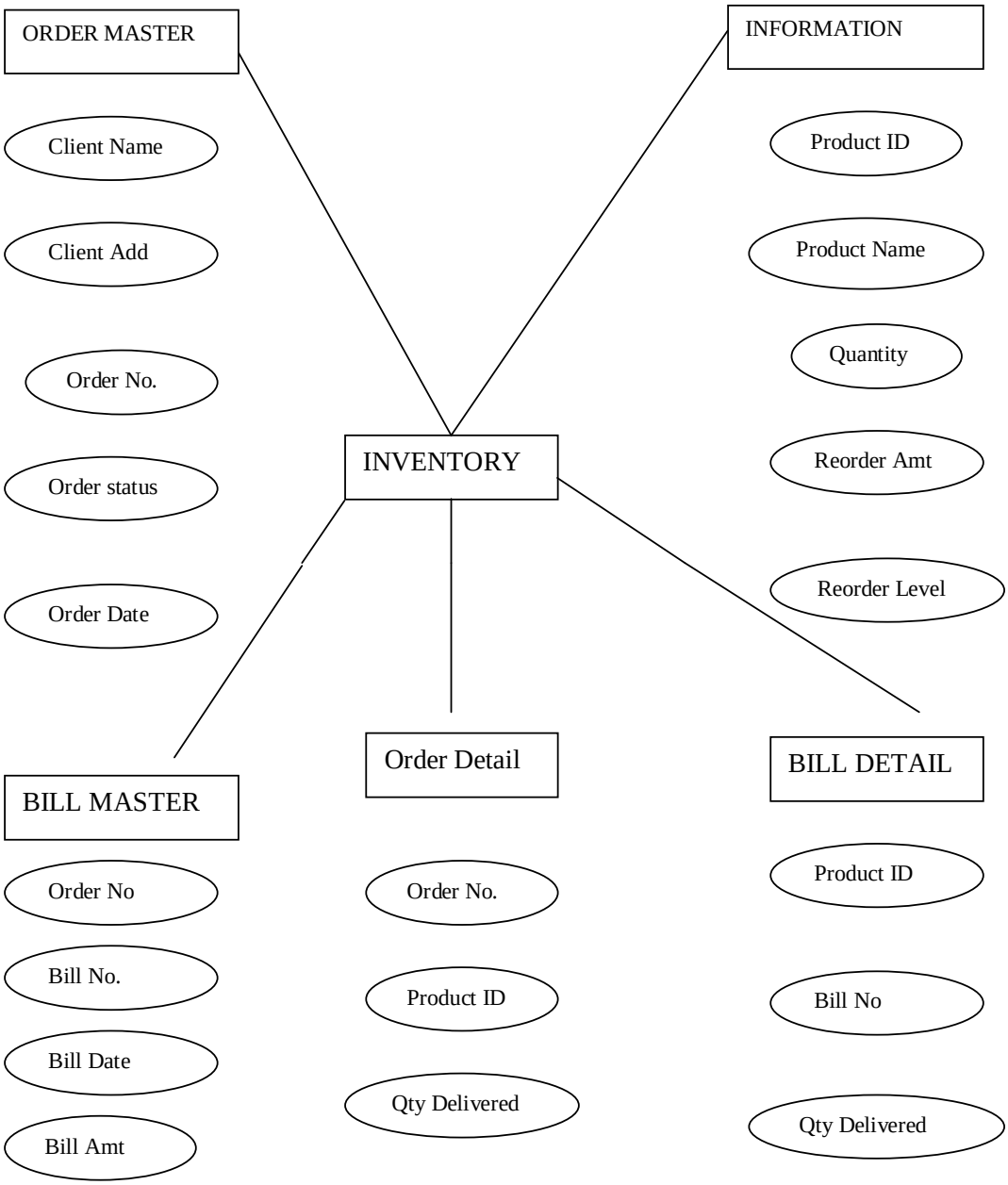


Figure 4.1 Ontology of Inventory

Figure 4.1 represents the Inventory Ontology. The box shows the classes and oval shows the slots. This ontology has been implemented among all the tools like as Protégé-2000, OiLed, Apollo and Ontosaurus and OntoStudio.

4.1 Implementation in Protégé -2000

Protégé-2000 implements a rich set of knowledge-modeling structures and actions that support the creation, visualization, and manipulation of ontologies in various representation formats. Protégé can be customized to provide domain-friendly support for creating knowledge models and entering data. Protégé-2000 can be extended by way of a plug-in architecture and a Java-based Application Programming Interface (API) for building knowledge-based tools and applications. Plug-ins like Prompt & Jambalaya is also available for Protégé-2000 editor. Protégé-2000 is comes in three flavors Protégé 3.3.1, Protégé 3.4 beta, Protégé 4.0 alpha along with plug-ins and demo ontologies.

The figure 4.2 represents the main windows of Protégé-2000.The classes are shown in right pane. Slots can be created by clicking on to the slot window. The slots listed in the left pane and the properties shown in the right pane in the slot editor. The Slot can be assigned to one or more classes. Slots Tab has a layout similar to the Classes Tab. Slot has the default value type String.

Instances are created by clicking on to the instance tab. The instance tab has three panes. The first shows at the far left displays the class hierarchy. The middle pane shows the list of instances created for a class. The third pane shows the Instance Editor which displays the form for the currently highlighted instance where slot values can be added.

The query can be formulated to the knowledgebase. The Queries tab allows querying project and locating all instances that match the criteria that have been specified. To run queries add the class, slot and string in it. To run the query clicks the find button. To see the results enlarge the right window and double click on the instance which is used in the query. Figure 4.3 shows the query window.

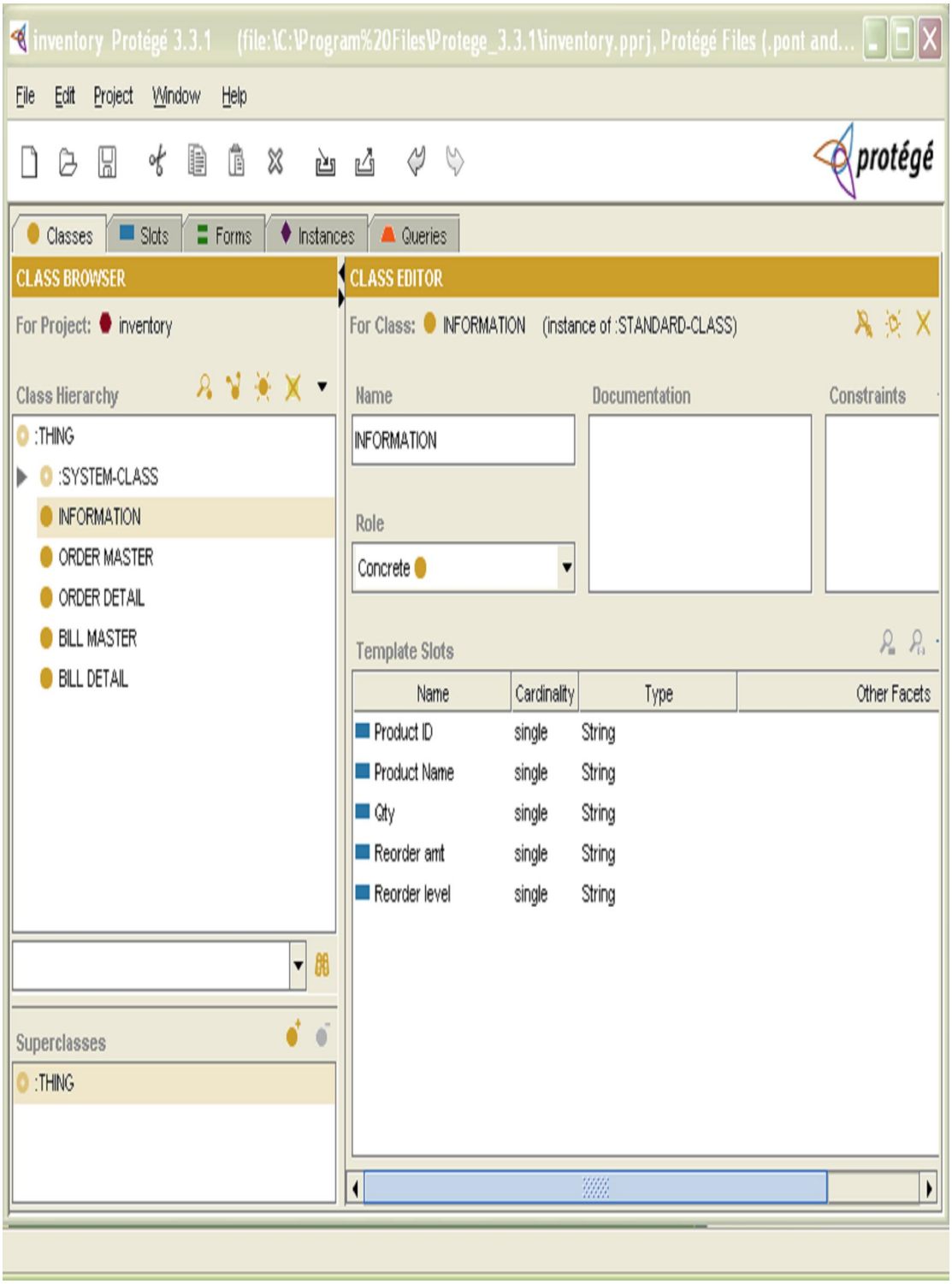


Figure4.2 Protégé-2000

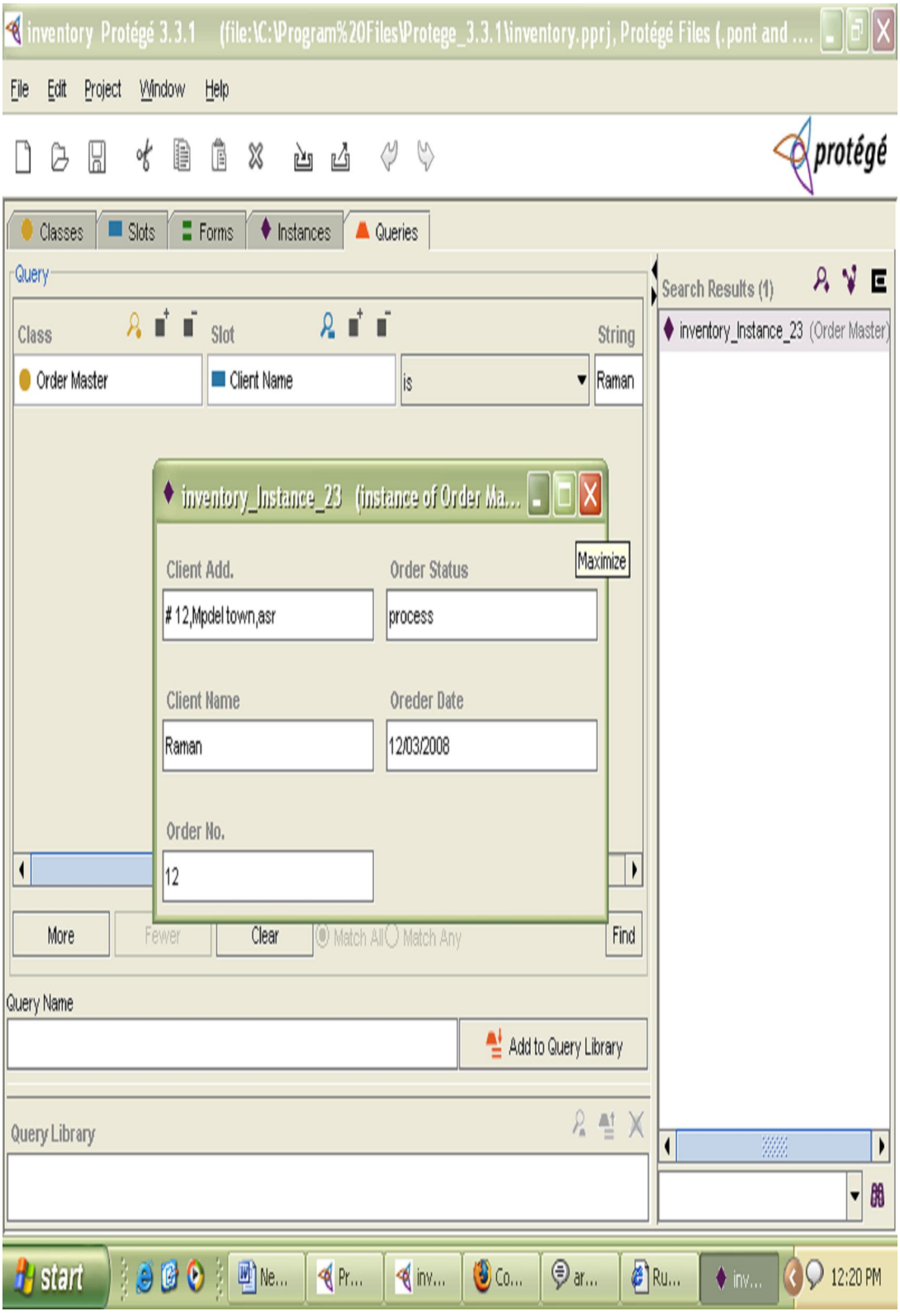


Figure4.3 Queries in Protégé-2000

4.2 Implementation in OilLed

The class pane displays a list of all classes on the left hand side in alphabetical order. Selecting a class will display its definition in the right hand pane. The definition consists of a description editor along with some documentation and a pair of radio buttons. Radio buttons indicating whether the class is a subclass or same class

The slots can be created by clicking to the slot tab. A slot definition gives the name of the slot and allows additional properties of the slot to be asserted. Domain and range restrictions on a slot can be specified. As with class descriptions, the domain and range restrictions can be arbitrary class expressions e.g anonymous frames or Boolean combinations of classes or frames. Domain and range restrictions are global, and apply to every occurrence of the slot, whether explicit or implicit. A slot can also be asserted to be transitive, functional or symmetric.

Properties are added in the property tab. Instances are added in the individual tab. The reasoner is user to check the consistency in the in the ontology. Reasoning is performed on a single-shot basis, i.e., at some suitable point the user connects to the reasoner and requests verification of the ontology. OiLed does not support large scale ontologies. Figure 4.4 shows the main window of OiLed.

OiLed provides the consistency checking. If the reasoner is connected button 

will be displayed along with commit result button  . If the reasoner is not connected OiLed displays 

button. If reasoner find any inconsistency in classes, properties, individuals and axioms it will mark inconsistent class as red icon as figure 4.5 represents the inconsistent ontology in OiLed editor. Classes with yellow icons in figure 4.6 represent consistent ontology with respect to relations, properties, individuals and axioms.

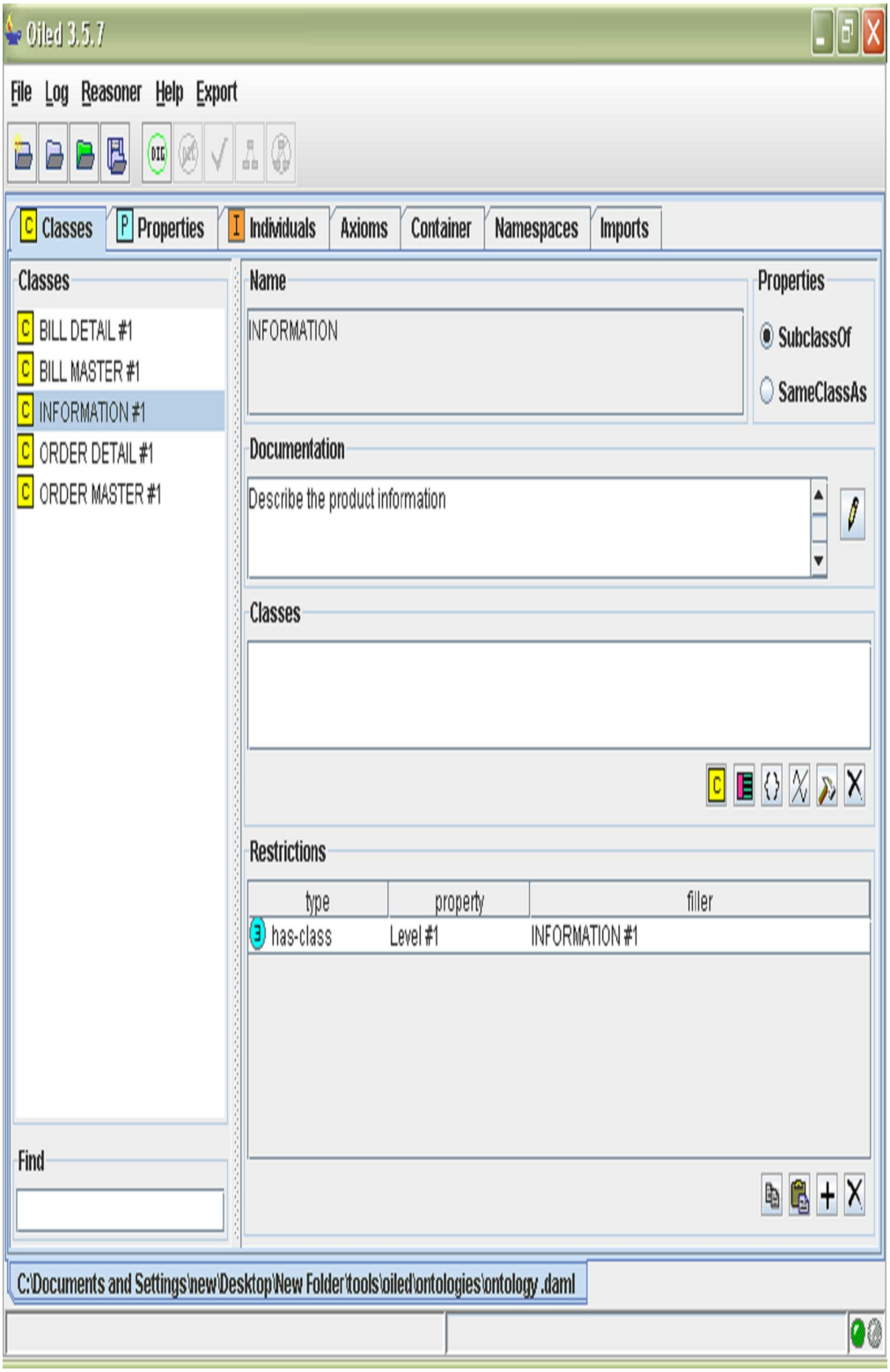


Figure4.4 OilEd

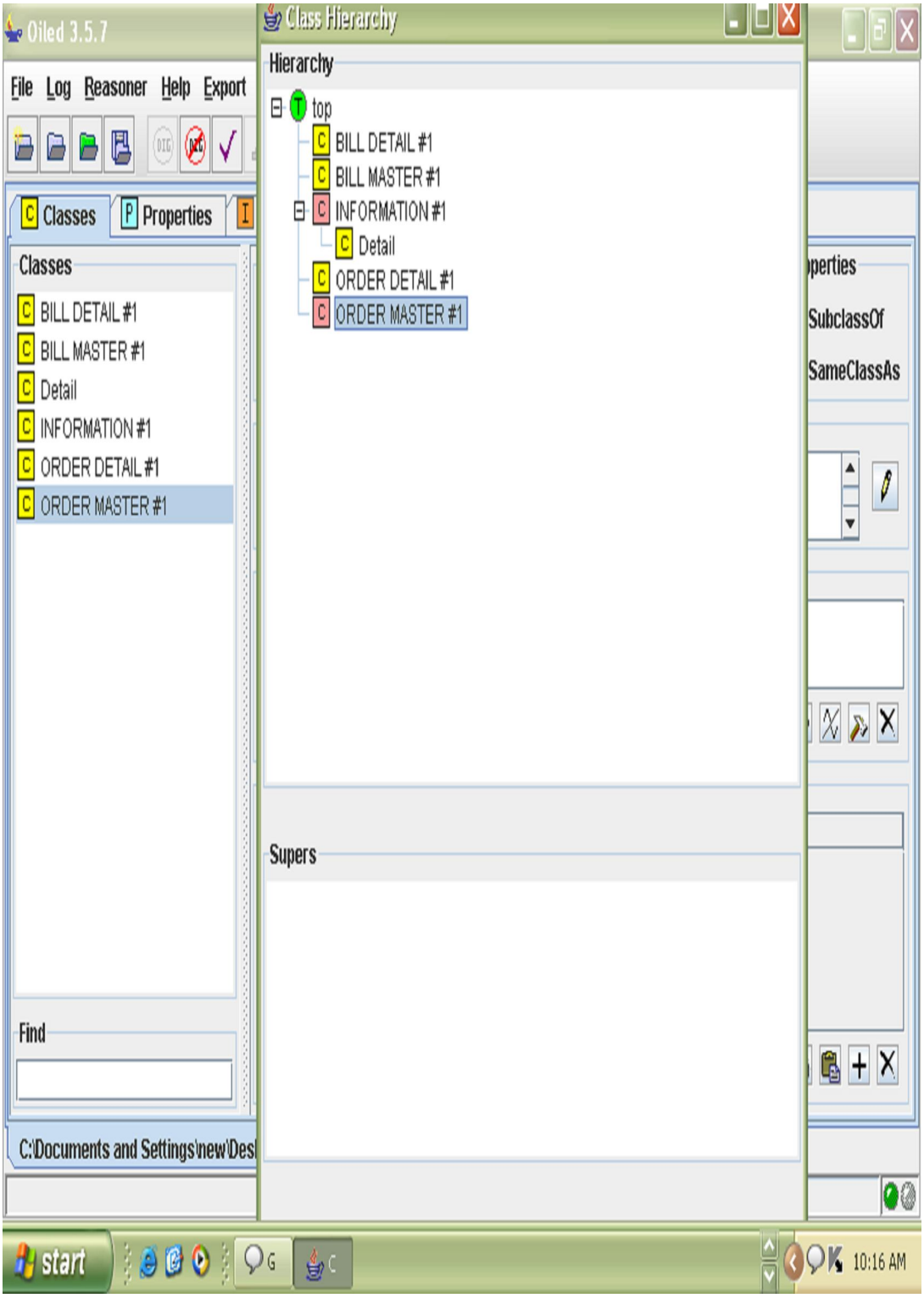


Figure 4.5 Consistency checking in OiLed

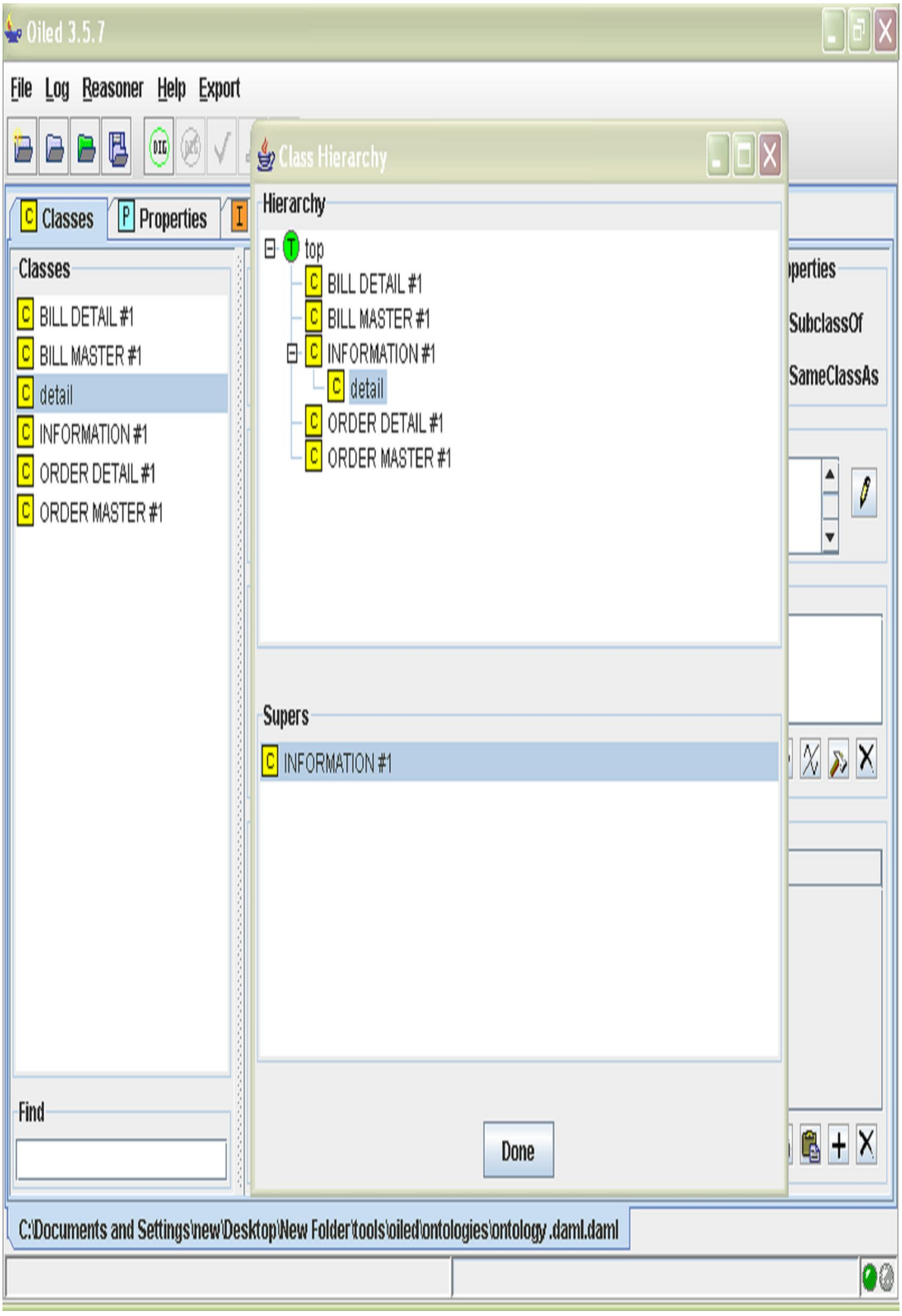


Figure 4.6 Consistent Ontology

4.3 Implementation in Apollo

Apollo is a user-friendly knowledge modeling application. The modeling is based on the basic primitives, such as classes, instances, functions, relations etc. In Apollo classes are represented in hierarchal fashion. Clicking on to a class or instance displayed on left pane explores the detailed view in the right bottom pane. Apollo performs a full consistency check while editing. Ontologies are represented in top left pane in a hierarchical manner.

The new project is open by clicking the file option, option new Ontology comes under it. Ontologies which have been made will be shown in top right pane. Classes are added by clicking to the class option below the file option.

Instance is added by clicking on to instance tab. Instances add against one or more class. The value of instance can get any of the optional values. Default value of instance is sting. Apollo does not accept space among the character e.g.” Order Master” have single space which Apollo cannot accept.

Relations, properties, axioms and slots are added by clicking on to there respective tabs. Apollo does not provide the plug-ins facility. Figure 4.7 represents the main Apollo window.

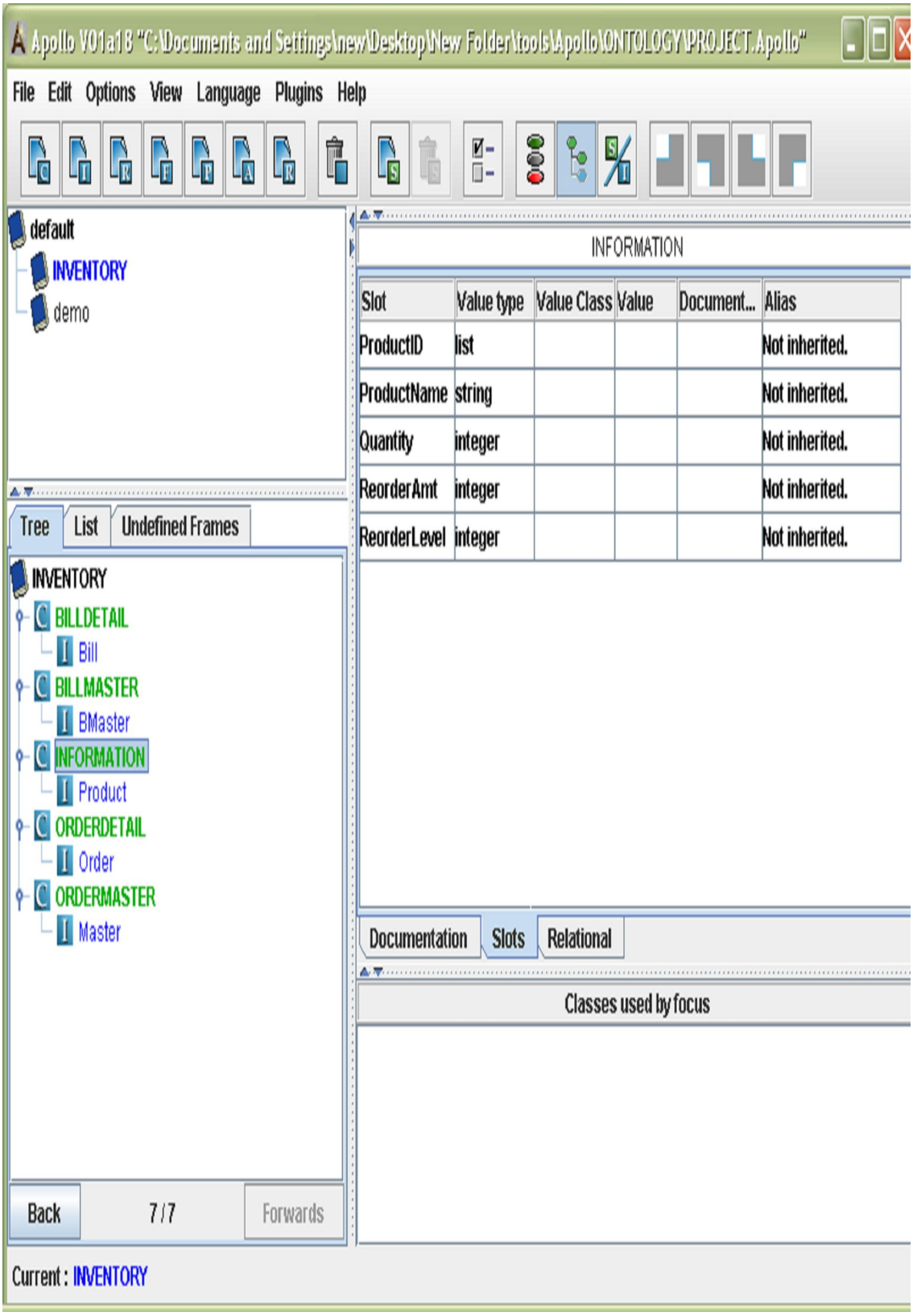


Figure4.7 Apollo

4.4 Implementation in Ontosaurus

OntoSaurus it is very easy to browse through ontologies, even if they are very complex. The browser has four frames. In one frame special bookmarks of the ontology can be marked, which can make browsing easier and faster. In the top frame choose which ontologies want to see. The right frame shows the main concepts of ontology. Figure 4.8 represents the Ontosaurus window.

Figure 4.9 shows the console of Power Loom. Ontosaurus is based on it. Power Loom starts with the message “Welcome to Power Loom”. Some of the basic commands of Power Loom are:-

HELP+ – This command tells the following commands are available.

ALL-FACTS-OF: Return a list of all definite true or false propositions that reference the instance.

ASK: Perform inference to determine whether the proposition specified in options is true.

ASSERT: Return the asserted proposition object.

ASSERT-FROM-QUERY: Evaluate query, instantiate the query proposition for each generated solution and assert the resulting propositions.

CALL-ALL-FACTS-OF: Return a list of all definite true or false propositions that reference the instance.

CLASSIFY-INSTANCES: Classify instances visible in module.

CLASSIFY-RELATIONS: Classify named relations visible in module.

CREATE: Create a logic object with name ' and return it.

CURRENT-INFERENCE-LEVEL: Return the current inference level that is active in the current query.

DEFCONCEPT: Define a concept.

DEFFUNCTION: Define a logic function.

DEFMODULE: Define module.

DEFOBJECT: Define (or redefine) a logic instance.

DEFPROPOSITION: Define a named proposition.

DEFRELATION: Define) a logic relation.

DEFRULE: Define a named rule

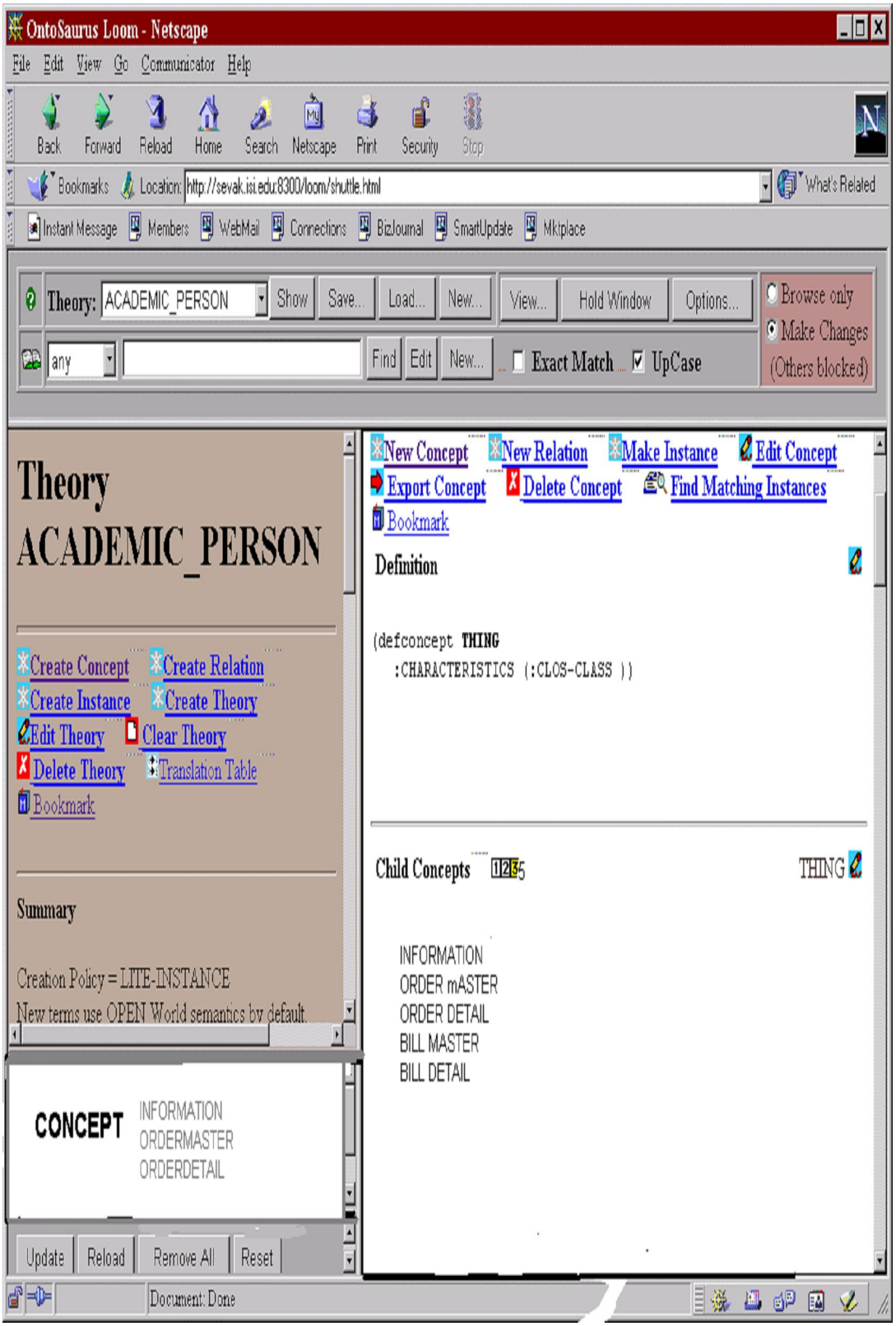


Figure4.8 Ontosaurus

DELETE-RULES: Delete the list of rules associated with relation.

FIND-INSTANCE: Return the nearest instance

FIND-RULE: Search a rule.

GET-RULES: Return the list of rules associated with relation.

PROCESS-DEFINITIONS: Finish processing all definitions and assertions that have been evaluated.

Figure 4.9 PowerLoom

4.5 Implementation in OntoStudio

Ontostudio provides the 4 basic schemas.

1. Ontology Navigator display: In this schema different ontology projects and their corresponding ontologies are displayed. This view shows the concepts, attributes, relations instance, rules and queries folder. Figure 4.10 shows the Ontology navigator window.

2. Entity Properties View: This schema describes the separate window for concepts, attributes, relations, rules and queries.

    - In the entity properties view of concepts all properties of a concept can edit. Figure 4.11 represents the entity property view of concept.
    - The relations of ontology can be editing within the relation area of the Entity Properties View. Figure 4.12 represents the entity property view of relation.
    - All existing instances for a concept will be shown in figure 4.13. New instances can be created, remove and deleted as well.
    - The Graphical Rule Editor is the main feature for creating and existing rules. Validating, testing, regression test and an explanation component is supported by a rule debugger, figure 4.14 represents the rules corresponding to the rule of queries.

3. Graph View: In this schema graph of concepts and attributes can be viewed. Figure 4.15 shows the graph of concept “Information”.

4. In OntoStudio queries can directly post in the query window, in order to test the modeled ontologies for their logical dependencies and validity. All changes to the ontology can be tested directly on the running system. Figure 4.16 represents the query window.

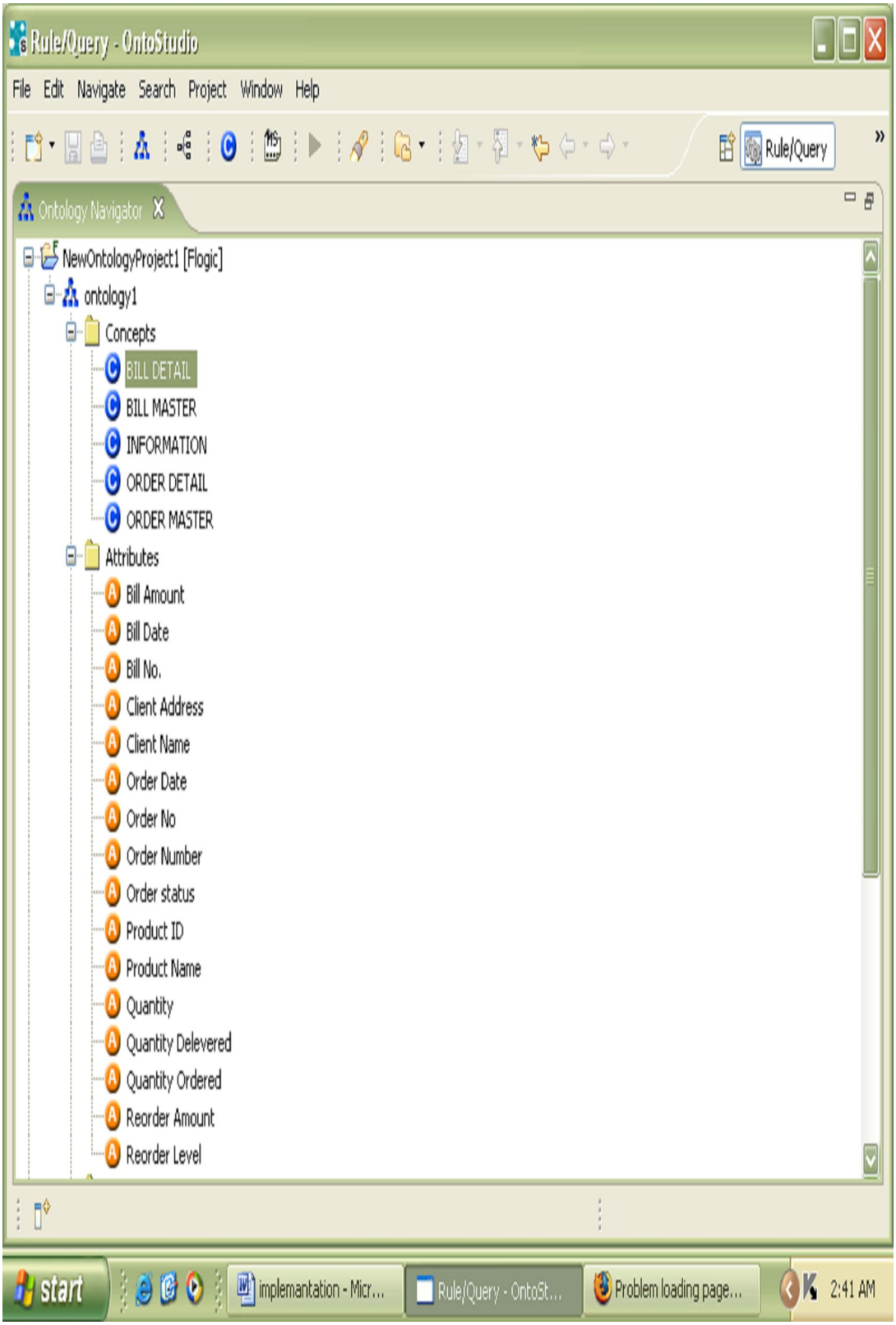


Figure 4.10 Ontology Navigator

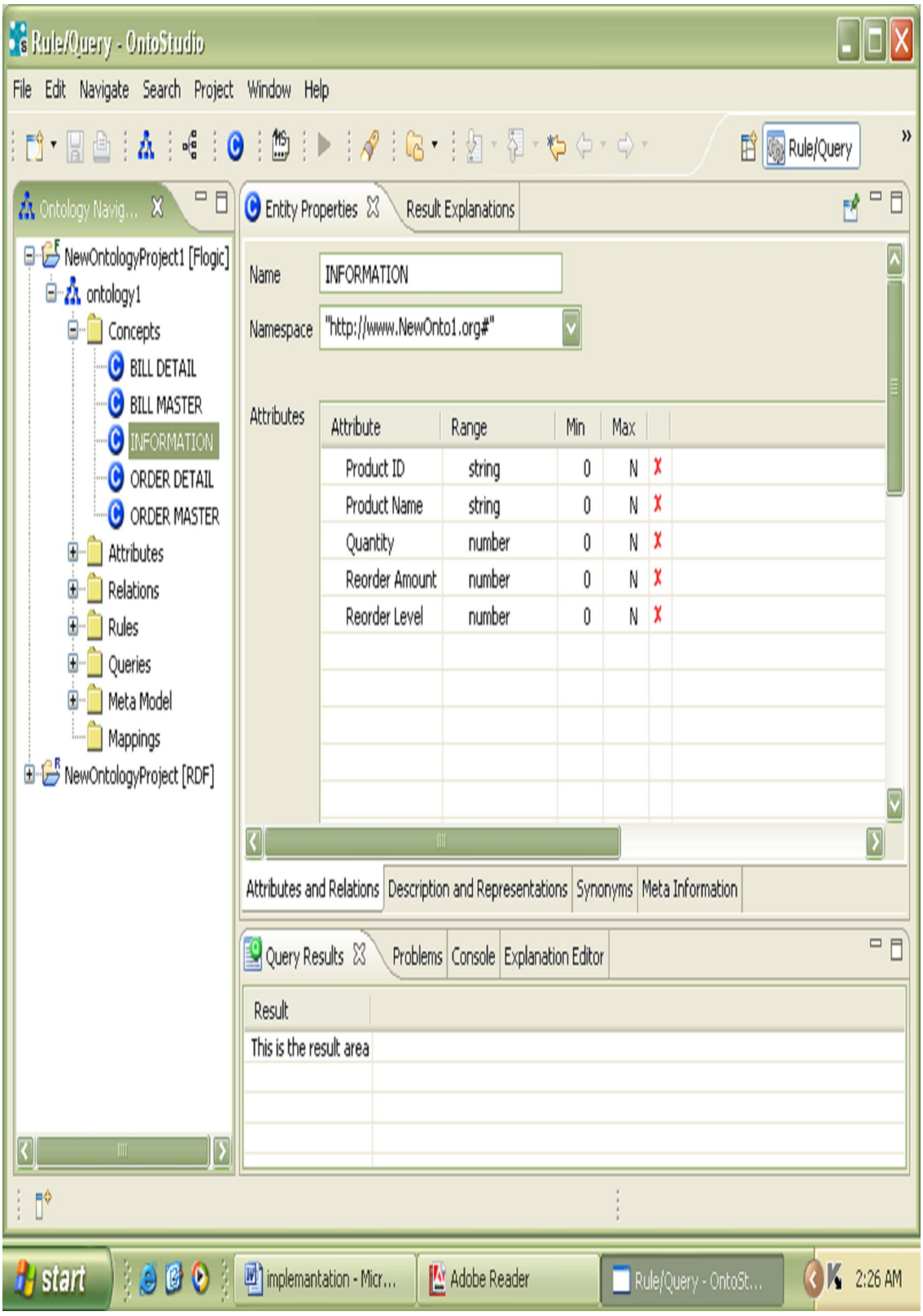


Figure4.11 Entity property view of concept

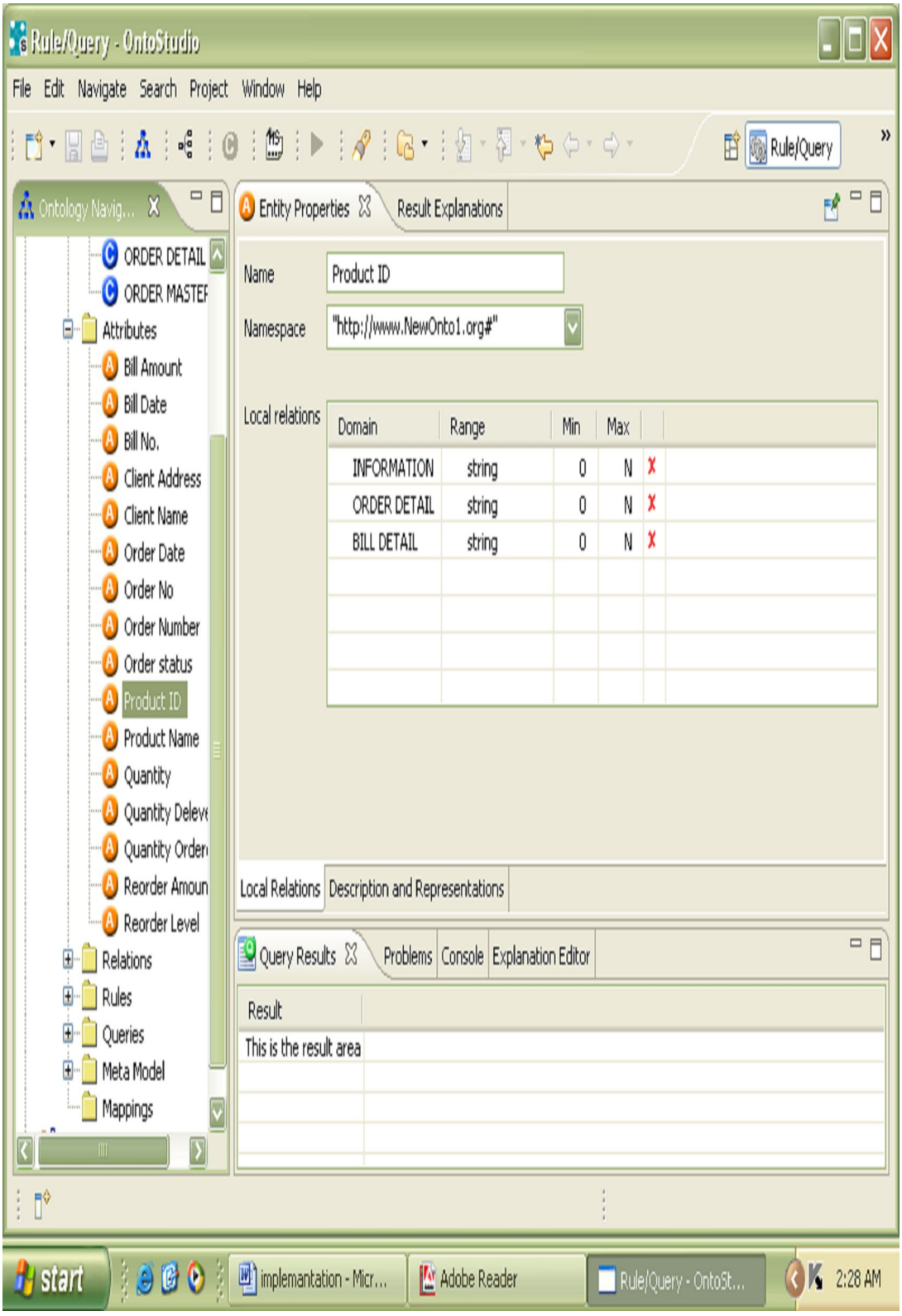


Figure 4.12 Entity property view of property

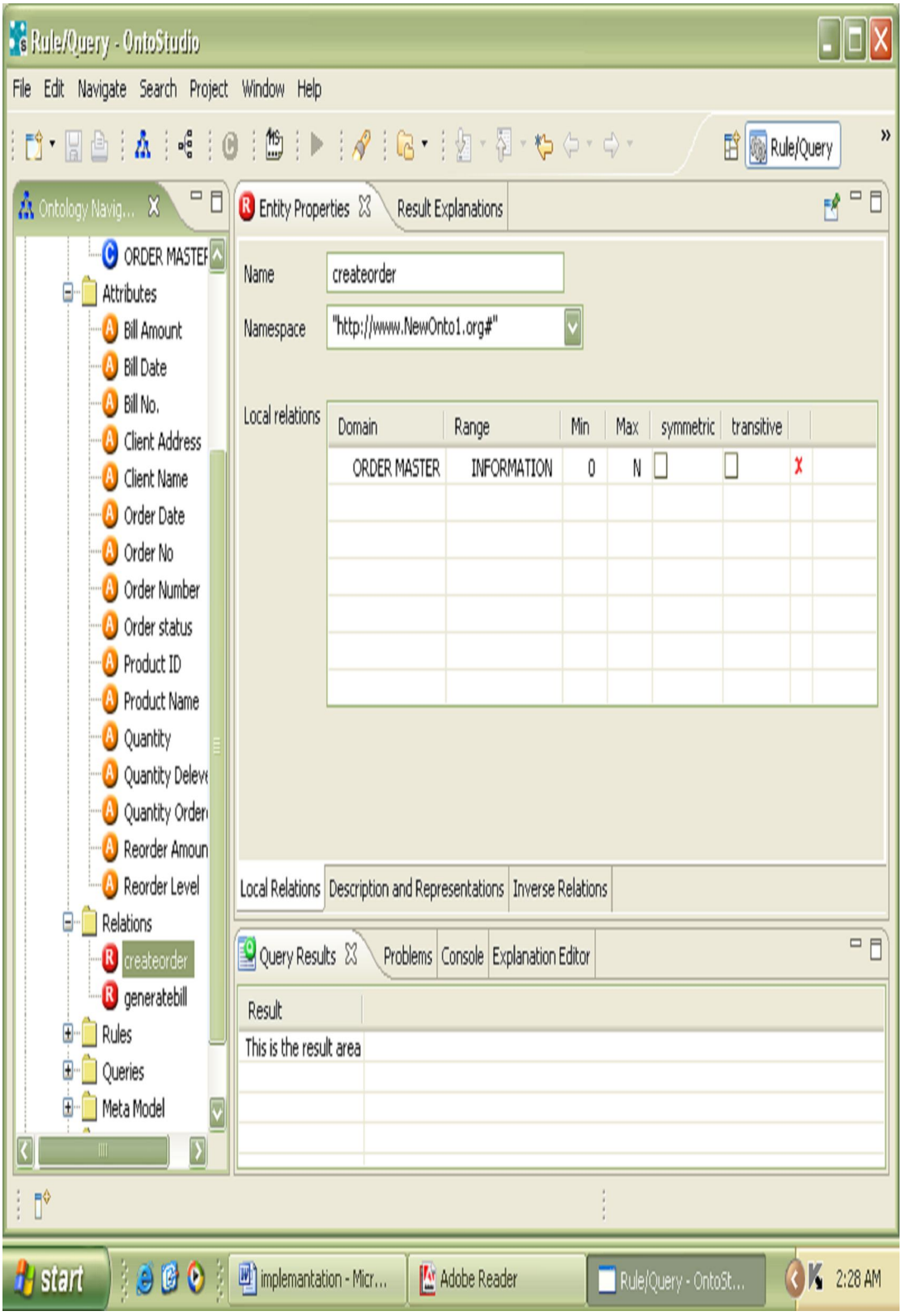


Figure 4.13 Entity property view of instance

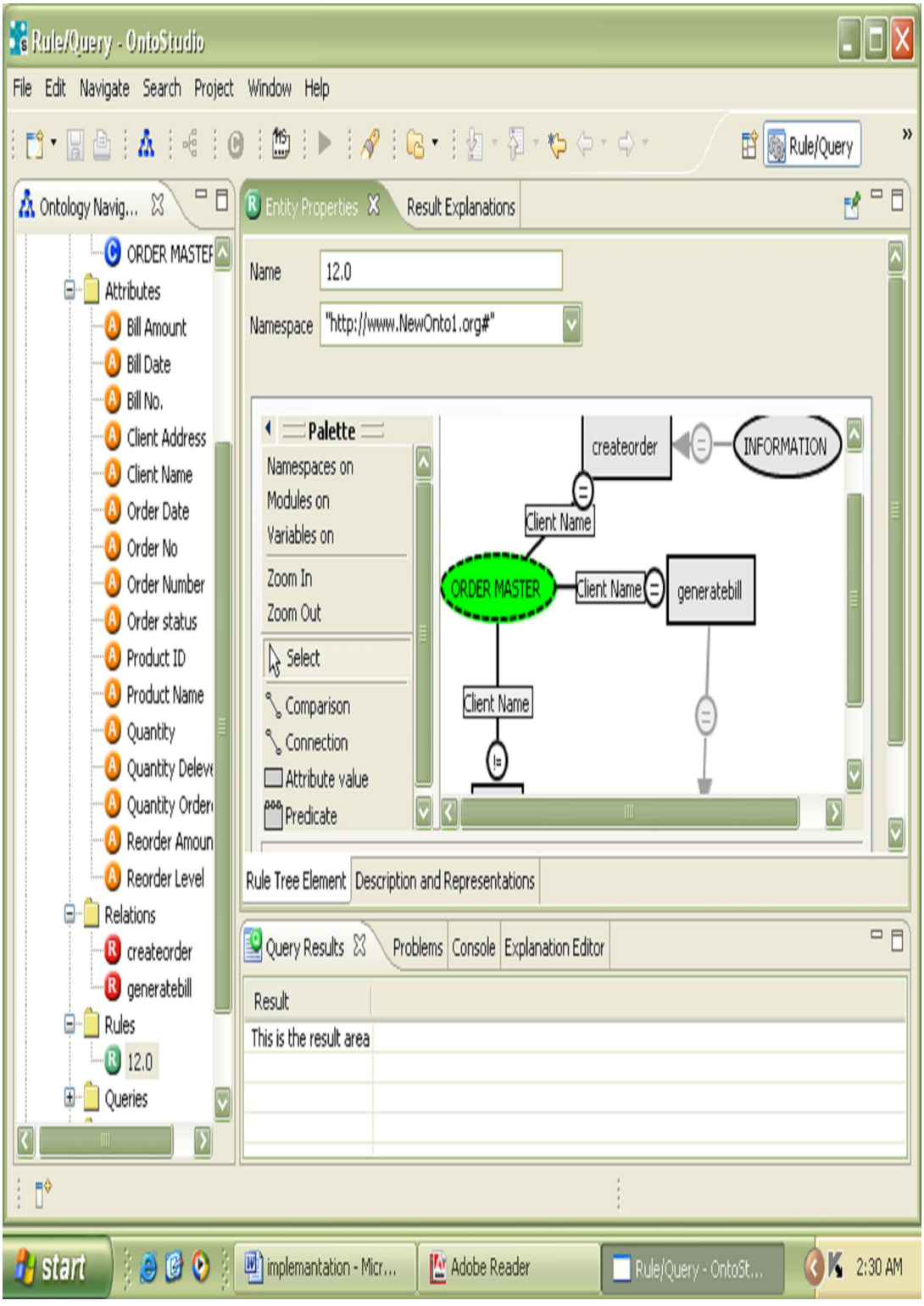


Figure 4.14 Entity property view of rules

Figure 4.15 represents the graph visualize of concept ”Information”. The yellow lines represent the attributes and blue line represents the instance.

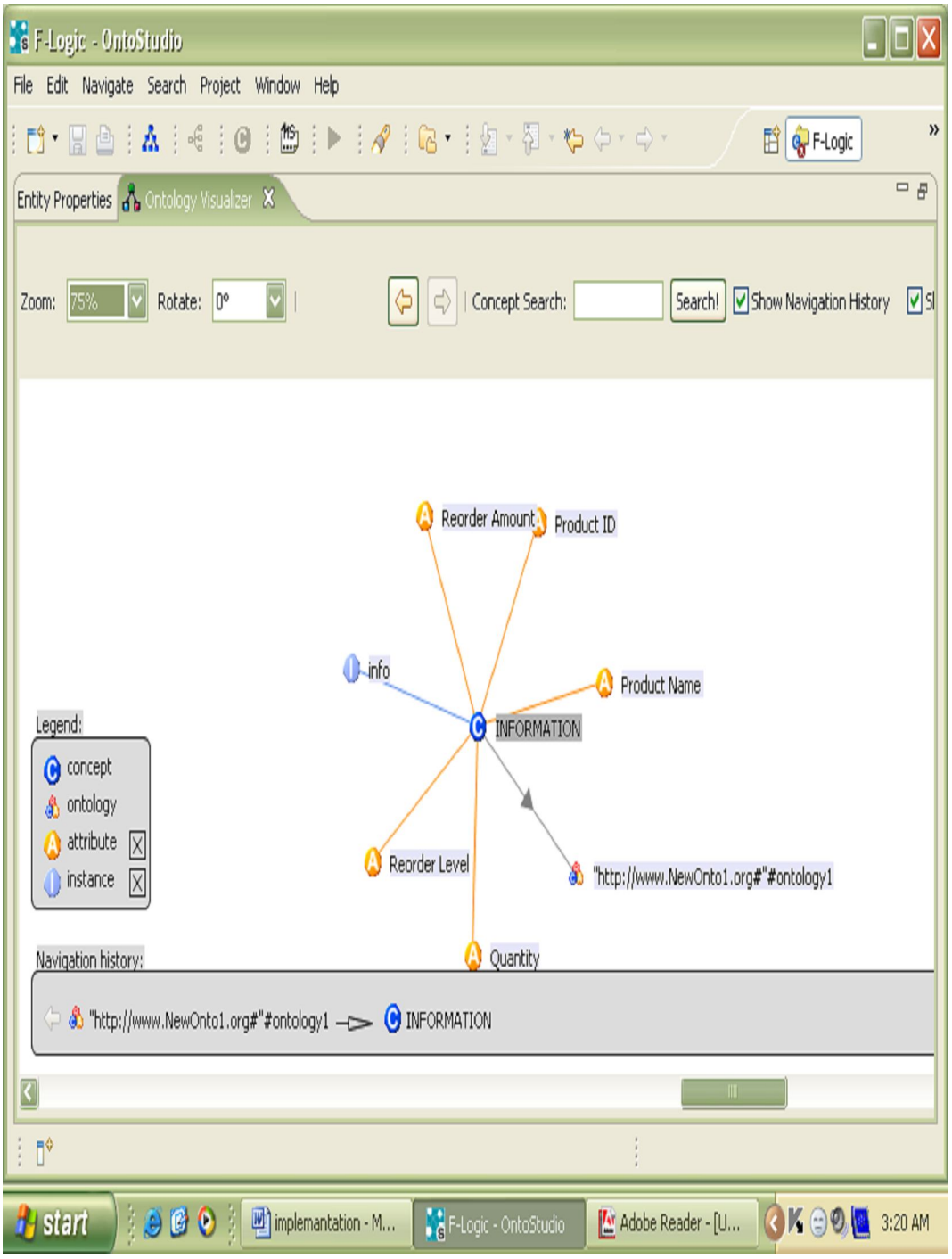


Figure 4.15 Graph visualize of concept “Information”

OntoStudio can give results to queries regarding the ontology. Figure 4.16 represents the query window. On querying which customer bought keyboard of product ID A101 results the “name of customer”.

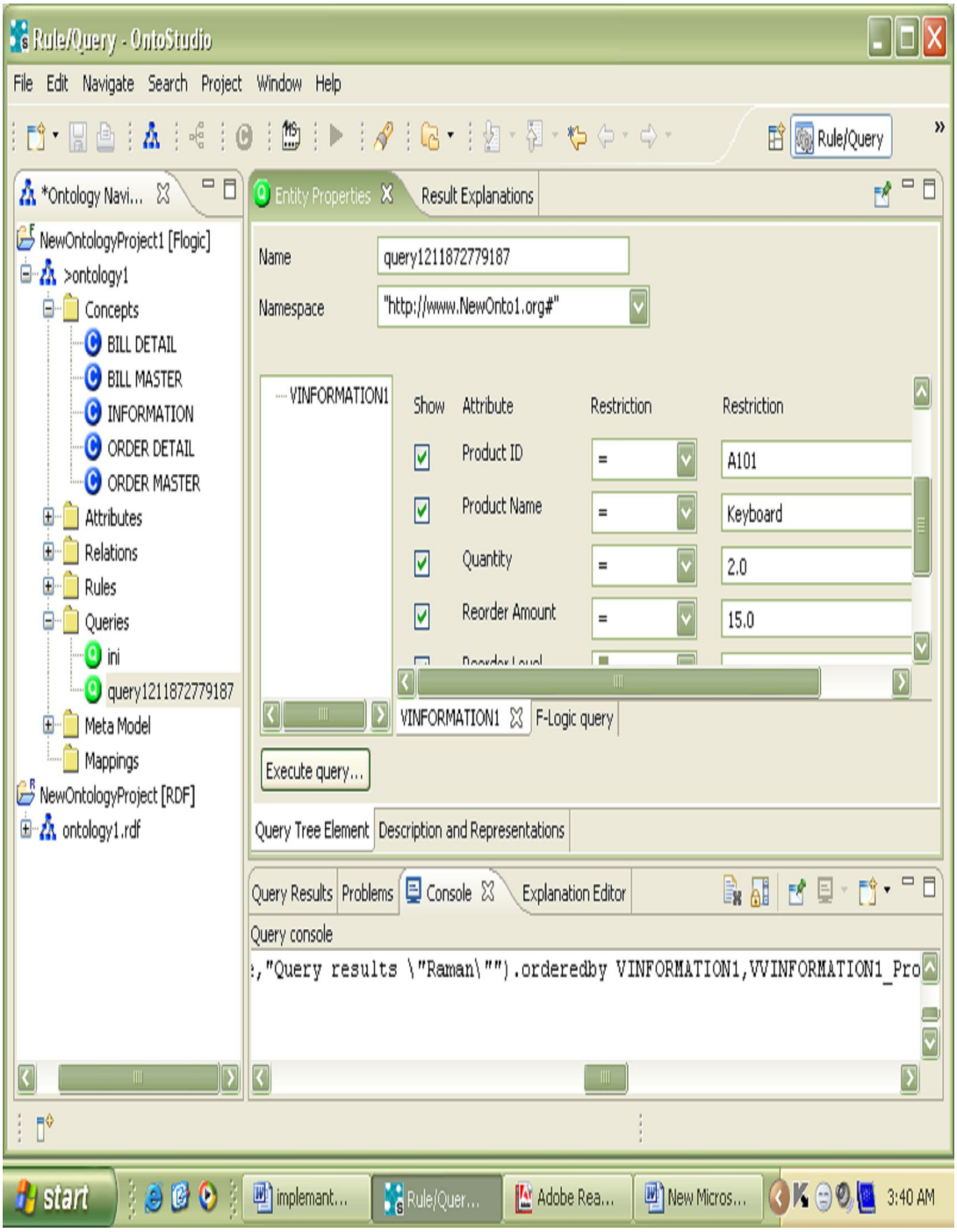


Figure 4.16 Queries in Ontostudio

Chapter- 5

CONCLUSION AND FUTURE SCOPE

5.1 Discussion

The goal of surveying various ontology tools is to get acquainted with the tools and to assess the features provided in it. Amount of pre-knowledge is required to use those tools. The results achieved are depicted below in a bar graph.

5.1.1 Protégé – 2000

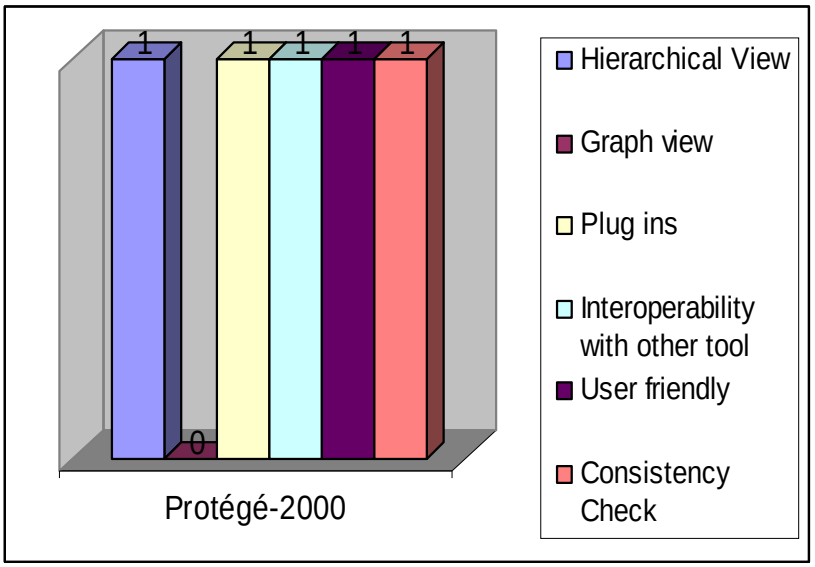


Figure 5.1 Bar graph of Protégé – 2000

Figure 5.1 shows the bar graph of Protégé-2000. High bar represented by value 1 shows the presence of Protégé-2000 feature and value 0 shows the absence of feature. In Protégé-2000 classes with slots, function, relations and hierarchical view is displayed. There is also an option of multiple inheritances in it. Protégé does not require java for installation. Plug-ins like Prompt & Jambalaya is also available for

Protégé-2000 editor. Protégé-2000 is user friendly editor. Ontology which can be build in other tool can be used in Protégé-2000 with little changes. It also provides the facility of consistency checking.

5.1.2 OiLed

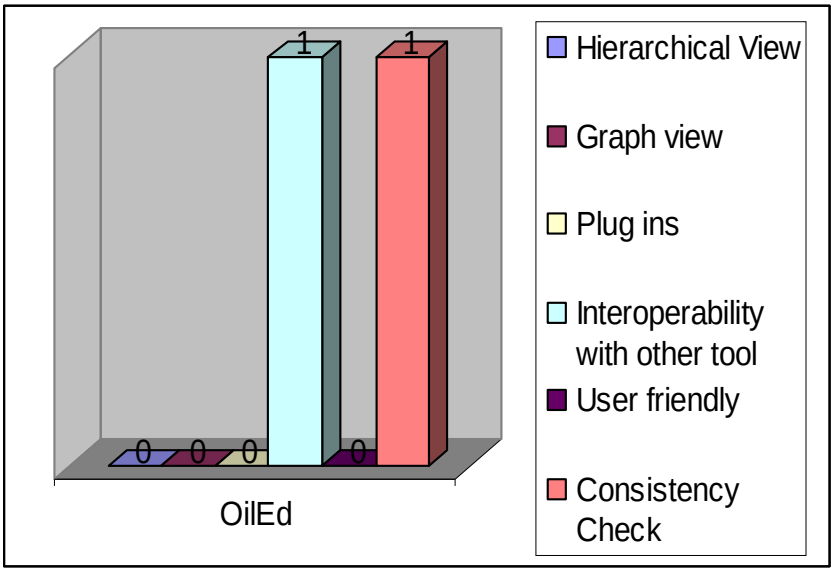


Figure 5.2 Bar graph of OiLed

Figure 5.2 represents the bar graph of OiLed editor for developing ontology. The value 1 shows the presence of feature. OiLed require java for installation. Ontology which can be built in other tool can be used in OiLed. Feature consistency checking is also provided in it so the bars for them are high. Value 0 shows the absence of feature as OiLed does not provide hierarchical and graphical view and is complex to understand hence it is not user friendly. OiLed does have plug-ins features.

5.1.3 Apollo

In Apollo classes with slots, function, relations can be represented in hierarchical fashion; hence it is easy to understand. Apollo is user friendly editor. It requires java for installation. The bars for them are high and are shown with value 1. Apollo has some disadvantage too. It does not provide graphical view of ontology. In addition to it does not have plug-ins. Ontology which are build in other tool cannot be import in Apollo. It does not provide the facility of consistency checking. Figure 5.3 shows the bar graph of Apollo.

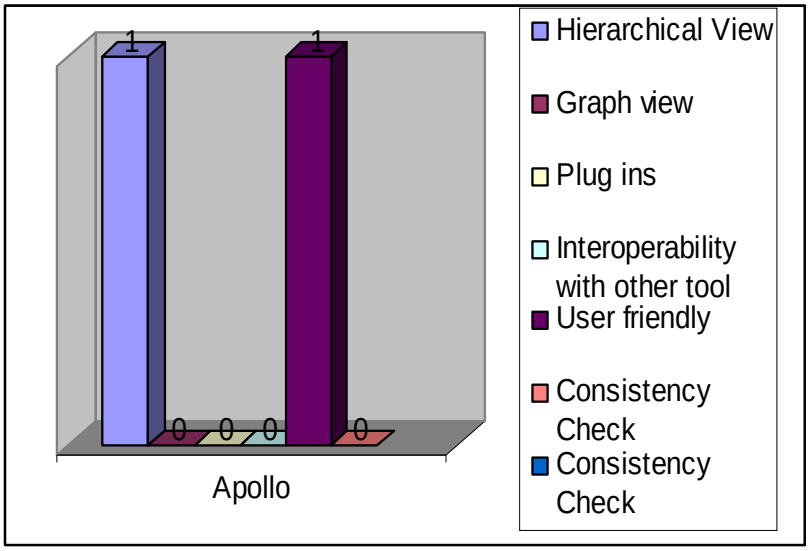


Figure 5.3 Bar graph of Apollo

5.1.4 Ontosaurus

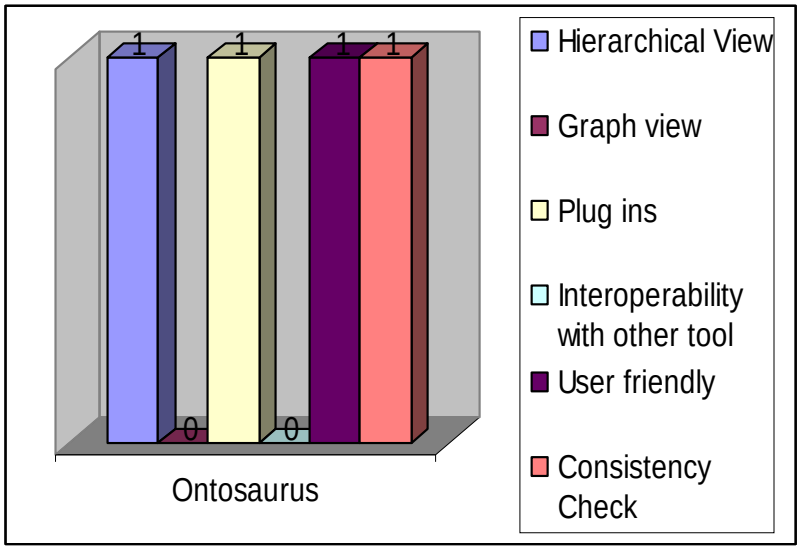


Figure5.4 Bar graph of Ontosaurus

Figure 5.4 shows the bar graph of Ontosaurus editor. In Ontosaurus classes with slots, functions, and relations can be represented in hierarchical manner. Plug-ins is also available for it. Ontosaurus is user friendly editor. It also provides the facility of consistency checking. Hence the bars for these features are high and represented the

value 1. Feature interoperability with other tools and graphical view are missing. Hence value 0 in bar graph.

5.1.5 Ontostudio

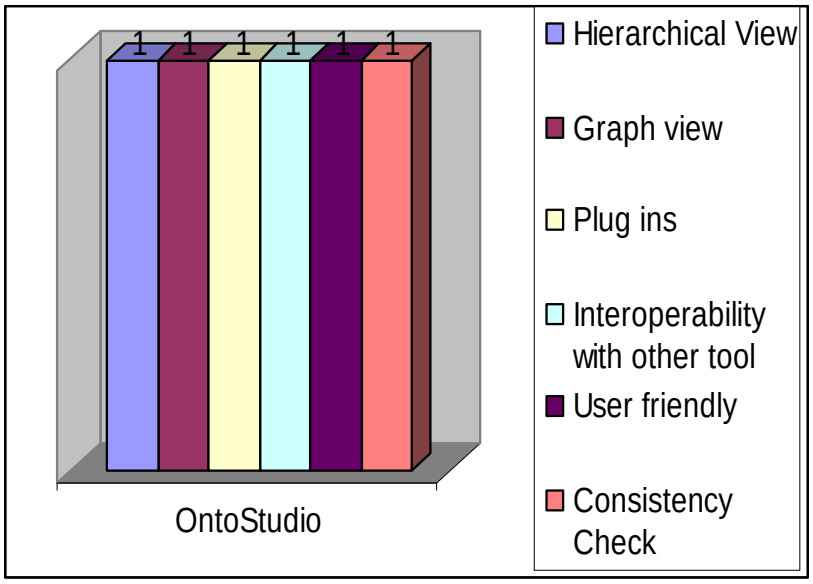


Figure 5.5 Bar graph of Ontostudio

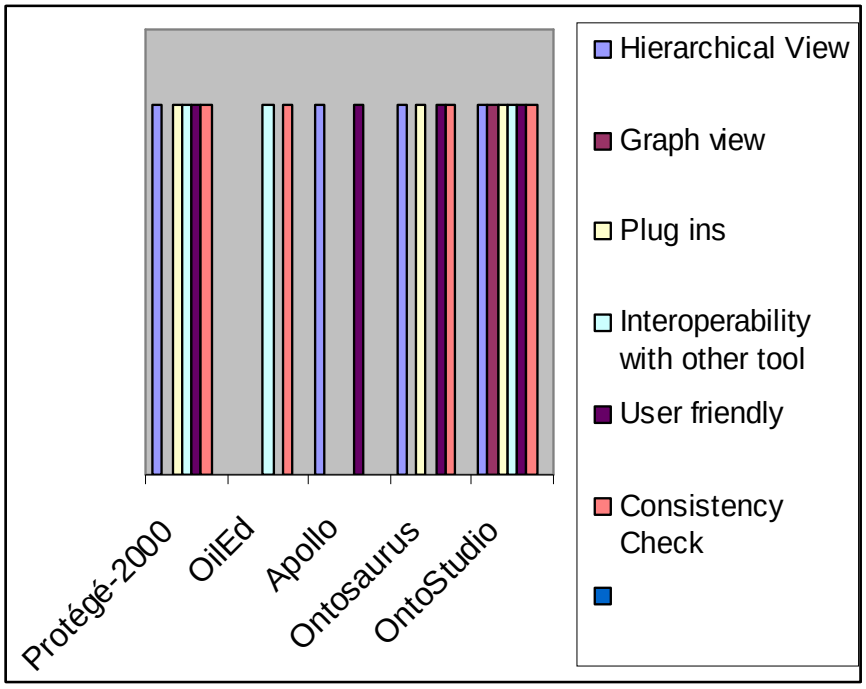
Figure 5.5 shows the bar graph of Ontostudio. Ontostudio have an advantage that classes with slots, function, relations are displayed in hierarchical view. Plug-ins is also available for Ontostudio editor. It is user friendly editor. Ontology which can be built in other tool can be used in Ontostudio with little changes. Hence the bar for it is also high. Graphical view of classes can be viewed in it. It also provides the facility of consistency checking

5.2 Conclusion

5.2.1 Merged Bar Diagram

As seen in the above graphs it is clears that in Apollo, Protégé-2000, Ontosaurus and OntoStudio support classes with slots, function, relations in hierarchical view. There is also an option of multiple inheritances in Protégé-2000, Ontosausus and OntoStudio. No such view is displayed in Oiled. Plug-ins is also available for Protégé-2000, Ontosaurus and OntoStudio. No such facility is available for Apollo,

Oiled and Ontosaurus. Apollo, Protégé-2000, Ontosaurus and OntoStudio are more user friendly tool compared to OiLed especially for novice .Consistency checking is provided in all tools except Apollo. Figure 5.6 shows the merged bar graph comparing all the tools.



.

Figure 5.6 Comparison of Ontological tool

All three tools have there own advantages and disadvantages. The developed ontology is stored in files in the above three tools. In Protégé-2000 and Ontosaurus ontology is stored in DBMS too. Web support for them is limited and there is no backup management for all the tools. Software architecture is client-server for Ontosaurus and standalone software architecture for all other tools. Multiple inheritances are available in Protégé-2000 and Ontosaurus whereas Apollo and OiLed do not have multiple inheritances facility. Consistency checking is done in all the above tools. Multiple inheritances are available in Protégé-2000 and Ontosaurus whereas Apollo and OiLed do not have multiple inheritances facility. Consistency checking is done in all the above tools. Table 1.1 represents the comparison among the tools.

Feature	**Apollo**	**OilEd**	**Protégé-2000**	**Ontosaurus**

Tool				
Storage	Files	Files	Files & DBMS	Files & DBMS
Web Support	Limited	Limited XML schema	Limited	HTTP browser
Backup	No	No	No	No
Software architecture	Standalone	Standalone	Standalone	Client Server

Table 1.1 Comparison of Ontological tool

Multiple inheritances are available in Protégé-2000 and Ontosaurus whereas Apollo and OiLed do not have multiple inheritances facility. Consistency checking is done in all the above tools.

5.3 Future Scope

Integrate all the ontology development tools in such a way that a single tool provides the mutual interface among all the tools.

APIs should be developed for providing plug-ins to facilitate more and more integration and interaction.

REFERENCES

[1] Yiling "Roadmap for Tool Support for Collaborative Ontology Engineering" Thesis submitted for the Degree of Master of Science in Computer Science Department of University of Victoria.

[2] Natalya F. Noy and Deborah L. McGuinness "Ontology Development: A Guide to Creating Your First Ontology", http://protege.stanford.edu/publications /ontology_development/ontology101-noy-mcguinness.html

[3] Deborah L. McGuinness, Dieter Fensel, J im Hendler, Henry Lieberman, and Wolfgang Wahlster "Ontologies Come of Age". http://www.ksl.stanford.edu/ people/dlm/papers/ontologies-come-of-age-mit-press-(with-citation).htm.

[4] URL: http://en.wikipedia.org.

[5] URL: http://www.ida.liu.se.

[6] I. Horrocks and J. Hendler, "The Semantic Web", in proceedings of First International Semantic Web Conference, Sardinia, Italy, June 9-12, 2002.

[7] S. Decker, S. Melnik, F. van Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann and I. Horrocks, "The Semantic Web: The Roles of XML and RDF", IEEE Internet Computing, Vol. 4(5), pp. 63-74, Sept/Oct 2000

[8] Oscar Corcho, Mariano Fernández-López, Asunción Gómez-Pérez, Arthur Stutt "OntoWeb Ontology-based information exchange for knowledge management and electronic commerce ",http://www.aifb.uni-karlsruhe.de/WBS/ publications/OntoWeb_Del_-3.pdf.

[9] Oscar Corcho and Asunción Gómez-Pérez “Ontology translation approaches for interoperability: a case study with Protégé-2000 and WebODE”, http://www.cs.man.ac.uk/~ocorcho/documents/EKAW2004_CorchoGomezPerez.pdf.

[10] URL: http://protege.stanford.ed

[11] Sean Bechhofer, Ian Horrocks, Carole Goble and Robert Stevens “OilEd: a Reason-able Ontology Editor for the Semantic Web”, http://www.cs.man.ac.uk/~horrocks/Publications/download/2001/oiled-dl.pdf

[12] Daniel Oberle1, Raphael Volz1,2, Ralf M¨oller3 and Peter Crowther, “ FaCT and OilEdClients”, http://wonderweb.semanticweb.org/deliverables/documents/D10.pdf

[13] Kamil Matoušek, Luboš Král, Martin Falc “Apollo CH Manual”, http://cipherweb.open.ac.uk/news/index.pl

[14] URL: http://www.isi.edu/isd/ontosaurus.html

[15] URL: http://www.ontoprise

[16] From Data Federation Pyramid to the Semantic Web ‘Birthday Cake’, http://www.mkbergman.com.

[17] T. Berners-Lee. The semantic web. Scientific american, 284(5):35–43, 2001.

[18] A. Bernstein and M. Klein. Towards high-precision service retrieval. In Proceedings of the First International Semantic Web Conference 2002, pages 84–101, 2002.

[19] O. Corcho, M. Fern´andez-L´opez, and A. G´omez-P´erez. Methodologies,

tools and languages for building ontologies: where is their meeting point? Data and Knowledge Engineering, 46(1):41–64, 2003.

[20] S. Decker, S. Melnik, F. van Harmelen, D. Fensel, M. C. A. Klein, J. Broekstra, M. Erdmann, and I. Horrocks. The semantic web: The roles of xml and rdf. IEEE Internet Computing, 15(3):63–74, 2000.

[21] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in description logic. In Principles of Knowledge Representation. CSLI Publications, 1996.

[22] M. Fern´and´ez, A. G´omez-P´erez, and N. Juristo. Methontology: From ontological arts towards ontological engineering. In Proceedings of the AAAI'97 Spring Symposium Series on Ontological Engineering, Stanford, USA, March 1997.

[23] N. Guarino. Formal ontology in information systems. In Proceedings of FOIS'98 (Formal Ontology in Information Systems), pages 3–15. IOS Press, 1998.

[24] N. Guarino, C. Masolo, and G. Vetere. Ontoseek: content-based acess to the web. IEEE Intelligent Systems, 3(14):70–80, 1999.

[25] C. M. Keet. Aspects of ontology integration. Technical report, School of Computing, Napier University, 2004.

[26] P. Mika and H. Akkermans. Analysis of the state-of-the-art in ontology-based knowledge management. Technical report, Vrije Universiteit, Amsterdam, 2003.

[27] N. F. Noy and C. Hafner. The state of art in ontology design. AI Magazine, 3(18):53–74, 1997.

[28] Y. Sure and R. Studer. On-to-knowledge methodology final version. Technical report, On-To-Knowledge Deliverable 18, 2002.

[29] M. Uschold and R. Jasper. A framework for understanding and classifying ontology applications. In Proceedings of the IJCAI99. Workshop on Ontologies and Problem-Solving Methods(KRR5), Sweden, 1999.

[30] Xiaomeng Su, " Semantic Enrichment for Ontology Mapping".

LIST OF PUBLICATIONS

- Amrit Kaur, Ms Shalini Batra and Ms. Rinkle Aggrawal "*Ontology and its Tools*" at International Conference ICCDIT-2008, held at PCTE, Ludhiana (Punjab) **[Presented & Published]**.