

Die approbierte Originalversion dieser Dissertation ist an der Hauptbibliothek der Technischen Universität Wien aufgestellt (<http://www.ub.tuwien.ac.at>).

The approved original version of this thesis is available at the main library of the Vienna University of Technology (<http://www.ub.tuwien.ac.at/englweb/>).

Diese Dissertation haben begutachtet:



TECHNISCHE
UNIVERSITÄT
WIEN
Vienna University of Technology

Dissertation

A Semantic Web Approach to License Agreements

ausgeführt zum Zwecke der Erlangung des akademischen
Grades eines Doktors der technischen Wissenschaften unter
der Leitung von

O.Univ.-Prof. Dipl.-Ing. Dr.techn. A Min Tjoa

Institute of Software Technology and Interactive Systems,
Vienna University of Technology, Austria

und

A.Univ.-Prof. Dr. Josef Küng

Institute for Application Oriented Knowledge Processing (FAW),
Johannes Kepler University of Linz, Austria

eingereicht an der

Technische Universität Wien
Fakultät für Informatik

durch

Muhammad Asfand-e-yar

Matrikelnummer: 0627502

Josef Baumgasse 8A/202, A-1220

Wien, 20. Oktober, 2011.

Unterschrift

I have the honor to mention here that this thesis has been described today all because of my parents and sister, who always guided and inspired me to make hard effort in order to achieve this goal. I dedicate these efforts to my sweet Mother, Father and Sister, who encouraged me to do whatever I wanted to pursue in my career.

Kurzfassung

Mit dem Aufkommen digitaler Technologien wurde das Management von Information sowohl in ihrer herkömmlichen dinglichen Ausprägung als auch in ihrer elektronischen Ausprägung wesentlich revolutioniert. Lizenzvereinbarungen und allgemeine Geschäftsbedingungen stellen die eine besondere Form von Information dar, Die Administration von Lizenzvereinbarungen und allgemeinen Geschäftsbedingungen mit Zuhilfenahme der Techniken des Semantic Web stellen eine besondere technische und wissenschaftliche Herausforderung dar. Die Anwendung von Semantic Web Techniken sollte hier eine Kategorisierung der verschiedenen Charakteristika von Lizenzvereinbarungen für Endbenutzer und Konsumenten ermöglichen und damit eine Grundlage für Entscheidungsunterstützung auf Grund der Benutzeranforderungen bieten. Aufgrund der Komplexität von Lizenzverträgen und der großen Vielfalt der Formulierung vermeidet der Endanwender sehr oft Lizenzvereinbarungen genau zu studieren und für gewöhnlich unterzeichnen Endanwender Lizenzvereinbarungen ohne die Details genau zu kennen.

Dies führt in einigen Fällen dazu, dass Endbenutzer wegen Verletzung der Lizenzvereinbarungen bestraft werden, obwohl dies nur unwissentlich bzw. versehentlich verursacht worden ist. Die Motivation dieser Arbeit liegt in der Beseitigung dieser Mängel durch einen semantisch basierten Lösungsansatz.

In dieser Arbeit werden in einem Semantic-Web-Model die Benutzeranforderungen bezüglich Lizenzvereinbarungen konzeptionell spezifiziert. Diese Benutzeranforderungen werden mit Hilfe von Fallbeispielen zur Bildung eines ontologischen Modells herangezogen. Dieses Modell wird sodann zu einer weitestgehend allgemeinen Lizenzontologie ausgebaut, Jede Lizenzontologie bildet die Grundlage für ein Modell einer Lizenzvereinbarung. Auf Grund dieser Lizenzontologien wurde die Applikation "Digital License Agreement" entwickelt. Diese Applikation ermöglicht ein viel besseres und tiefer gehendes Verständnis von Lizenzvereinbarungen und darüber hinaus eine Überprüfung und Abgleichung der Benutzeranforderungen bezüglich einer Vereinbarung.

Ein vorrangiges Ziel dieser Dissertation ist die Schließung der (semantischen) Lücke,

die zwischen dem Endbenutzer und dem Text einer (meist dem Laien schwer zugänglichen) Lizenzvereinbarung. Durch die Einführung eines semantischen Modells. Dieses semantische Modell ist von generischer Natur und ist so geartet, dass geringfügige konzeptionelle Änderungen von Vereinbarungen leicht umzusetzen sind. Das hier vorgeschlagene Modell ist vielfältig anwendbar. Es ist geeignet für Abfragen bezüglich spezifischer Anforderungen an Lizenzvereinbarungen und für die Abgleichung von Benutzeranforderungen bezüglich einer bestimmten Lizenzvereinbarung. Zwei große Gebiete von Lizenzvereinbarungen werden in dieser Dissertation tiefergehend behandelt, nämlich Softwarelizenzen und On-line-shopping.

Abstract

Emerging digital technologies, especially in information techniques, have revolutionized information management. Information management includes both electronic and physical information and is regardless of source or format. License agreements are also a form of information, which describes product's usage and its terms and conditions. Management of license agreements using Semantic Web is a multi-disciplinary challenge, involving categorization of common features and structuring the required information in such semantics that could be easily extendable and fulfilling the requirements of end users. Consequently, the solution proposed in this thesis is to find and match user requirements in license agreements using Semantic Web techniques.

Generally, License Agreements are in documented form and are heterogeneously constructed. Documented license agreements are lengthy and each of the license agreement (of different organization), has very less resemblances, but the agreements that lay under same category in one organization still remain quiet similar. Due to the lengthiness of license agreements and its heterogeneous formation, end user very often avoids to read and study to understand it, and habitually end users sign the agreement without knowing and understanding, "what is written in it". Therefore, in some cases end user face penalty, if any section of license agreement is violated; whether violation is occurred willingly or mistakenly. To address the issues, a semantic based solution is proposed in this thesis.

User requirements are defined, which are based on the license agreements to construct a Semantic Web model. These requirements are structured by use cases, which help to construct an ontological model. The ontological model is then extended to construct "real world license ontologies". Each license ontology is a model for an agreement. Using the license ontologies, an application named *Digital License Agreement* is developed. The application facilitates better understanding of agreements and also specifies for the user required information according to an agreement.

The ultimate goal of this thesis is to bridge the gap between an end-user and a license agreement, by introducing a semantic license model. The introduced semantic model

is a general model for each license agreement. Furthermore it is designed to be adaptable with minor changes according to specificities of an agreement. The model can be used for multiple purposes such as querying appropriate licenses for specific requirements or checking the license terms and conditions with the help of user requirements. The general semantic model, of license agreements, is divided into two sub-sections i.e. software license agreements and online shopping agreements. In the thesis, the extended models of software license and online shopping agreements are investigated.

Acknowledgment

I would like to thank my adviser Prof. A Min Tjoa, for his scientific support and advices. Without his inspiration, advices, guidelines and continuous support this thesis would not have been possible. Special thanks to Amin Anjomshoaa, who guided me from the beginning of my research work. With his efforts and help it was possible for me to complete my work. Thanks is also due to my colleagues at Institute for Software Technology and Interactive Systems (Vienna University of Technology) and all the members, with whom I have the privilege to work with. Even though the list is very long, I would like to thanks all those who help me not only in my research work but also in my daily life, while living far away from my parents and sweet sister. Finally and foremost, I would like to thanks my parents who teaches me everything that one could face in life, and always they kept my moral high in facing new challenges and they pray for my success every time.

Contents

Dedication	i
Kurzfassung	ii
Abstract	iv
Acknowledgment	vi
1 Introduction	1
1.1 License Agreements	1
1.1.1 User Awareness	3
1.2 World Wide Web and Semantic Web	4
1.3 Research Questions and Research Strategy	5
1.4 Contribution and Thesis Outline	7
2 Review of Possible Relevant Building Blocks	11
2.1 Software and Online Shopping Licenses	12
2.1.1 Copyright and Copyleft Licenses	13
Copyright	13
Copyleft	13
2.1.2 Open Source License	14
Open Source Initiative (OSI)	15
Open Source Definition (OSD)	15

2.1.3	Creative Commons	16
2.1.4	Online Shopping Agreements	18
2.2	Semantic Web	18
2.2.1	Digital Rights Management	19
	Digital Rights Expression Languages	19
	Digital Rights applied to Copyright	20
	RDF Meta-data Diffusion	20
2.2.2	Policy Management and Semantic Web	20
	Privacy Policy Matching	21
	KAoS	21
	Semantic Mapping Framework	22
	Policy in Network Domains	22
3	Semantic Web solution and Use Case Scenarios	24
3.1	Semantic Web Cake Layered Approach	25
3.1.1	Hypertext Web Technologies	26
3.1.2	Standardized Semantic Web Technologies	26
3.1.3	Unrealized Semantic Web Technologies	27
3.2	Ontology Language	27
3.3	Rule Language and Inferences	28
3.4	Use Case Scenarios	28
3.4.1	Use Case	29
	Projects for Postgraduates Students	30
	Creative Commons Licensing	31
	Online Shopping	31
	Generating Automatic License Agreement	32

4	Constructing License Ontology	35
4.1	Ontology Definitions	36
4.1.1	Taxonomy and Thesaurus	36
4.1.2	Conceptual Model	37
4.1.3	Logical Theory	37
4.1.4	Ontology	38
4.2	Main Components of Ontology	38
4.3	Approaches for Ontology Construction	38
4.3.1	Top Down Approach	39
4.3.2	Bottom Up Approach	39
4.4	Creating Ontology	39
4.4.1	Overview of Ontology	40
4.4.2	Ontological Models	40
4.5	License Ontology	41
4.5.1	License Ontology Classes	41
	Agreement Items:	41
	Domain Concept:	44
4.5.2	OWL Properties used in License Ontology	46
5	Extending License Ontology	48
5.1	Online Shopping License Ontology	49
5.1.1	Amazon License Ontology	50
5.1.2	eBay License Ontology	51
5.2	Software License Ontology	53
5.2.1	Mozilla & Apache License Ontology	56
5.2.2	Adobe License Ontology	57
5.2.3	Creative Commons License Ontology	58
	eyeOS Web License Ontology	58

6 Reasoning Ontologies using Rules	65
6.1 Web Rule Language	66
6.2 Semantic Web Rule Language	66
6.3 Protege for License Ontology	67
6.4 SPARQL Protocol and RDF Query Language	68
6.5 Rules used for Constructing License Ontology	68
6.5.1 Disjoint Classes in Ontologies	69
6.5.2 Axiom in Ontology	69
6.5.3 Value Partition in Ontology	70
6.5.4 Restriction used for License Class	70
6.5.5 Restriction used for Product Class	71
6.5.6 Restrictions used for Modified Source Code Class	71
6.5.7 Restrictions used for Transfer and Download Class	71
6.5.8 Restrictions used for Sub License Agreement Class	72
6.6 Reasoning License Ontologies	72
6.6.1 Online Payment	73
6.6.2 Method of Receiving a Product	73
6.6.3 Replacing of Damaged Product	74
6.6.4 Source Code Modification	74
6.6.5 Software Product for specific Operating System platform	74
6.6.6 Backup Copy	75
6.6.7 Third party Trademark	75
6.6.8 Third party Distribution	75
6.6.9 Creative Commons (by_nc_nd)	76
6.6.10 Reimbursement of Payment	76

7 Knowledge Based Domain using Jena API	78
7.1 Jena Architecture	78
7.2 Semantic Web and Jena Architecture	79
7.3 Use Case	80
7.3.1 Study at Austria	80
7.4 Digital License Agreement (DLA)	81
7.4.1 DLA Searching Mechanism	81
7.4.2 Solution to the Use Case	82
7.4.3 Results of DLA	83
8 Conclusion & Future Work	85
8.1 Review and Answer to Research Questions	86
8.2 Future Research	88
A RDF License Ontology	91
B RDF Online Shopping License Ontology	99
C RDF Amazon License Ontology	101
D RDF Software License Ontology	105
E RDF Mozilla License Ontology	111
F RDF Creative Commons Licensing Ontology	117
G RDF eyeOS Web License Ontology	119
Bibliography	121
Curriculum Vitae	126

List of Figures

1.1	Traditional Web	5
1.2	A Model of Digital License Agreement	7
1.3	Thesis Outline.	9
1.4	Thesis Overview.	10
2.1	CC license types.	17
3.1	Semantic Web Cake Layered by W3C	26
3.2	Use Case of Digital License Agreement System.	29
3.3	User Scenario of Digital License Agreement System.	34
4.1	The Ontology spectrum: weak to strong semantics	37
4.2	General model to describe the work flow.	40
4.3	License Ontological Model.	42
5.1	Online Shopping License Ontology.	50
5.2	Amazon License Ontology.	51
5.3	eBay License Ontology.	52
5.4	Software License Ontology.	54
5.5	Mozilla License Ontology.	60
5.6	Apache License Ontology.	61

5.7	Adobe License Ontology	62
5.8	Creative Commons License Ontology.l	63
5.9	eyeOS Web License Ontology.	64
6.1	WRL positioned in Semantic Web Cake.	66
7.1	Jena Architecture.	79
7.2	DLA Overview.	82
7.3	Results obtained by applying previous on all License Ontologies.	84
7.4	Results Obtained by applying query on selected License Ontology using second method.	84

Chapter 1

Introduction

In general, installing software requires the user's agreement to the according terms and conditions. The terms and conditions describe issues such as functionality, restrictions of use, and other legal acts such as jurisdiction restriction and intellectual property rights. These terms and conditions cover various legal acts of an agreement. The legal acts contained in agreements are very often difficult for an average user to understand. Therefore, users do not read such lengthy agreements and sign or accept the agreement without understanding. In such situations user might unintentionally commit an illegal act. Therefore, an easy understandable solution is required in order to facilitate the user's decision making process and the fulfillment of license requirements and/or restrictions. For this reason a machine interpretable knowledge solution should be developed which provides an agreement repository that makes it easy for a user to select an appropriate product. Therefore, a semantic model for license agreements as a machine interpretable knowledge repository is developed and with its help user queries will be matched in conjunction with user's requirements. In this chapter we will introduce the notion of Semantic Web, license agreements and the motivation of this thesis.

1.1 License Agreements

License agreement is the base for a mutual agreement between the licensee and users to use a software and its services. Therefore, purchasing or selling any product requires a binding agreement between two parties. Every agreement is a legal binding between two parties and for any violation of an agreement consequences such as penalties are defined and applied on one of the parties. Similarly, installing software requires the user's agreement to the software's terms and conditions. Software

agreements describe issues such as its functionality, restrictions on its use, and the maximum number of copies that can be made. Software agreements may also specify the number of users that can work in a connected environment, state relevant laws if any, rules pertaining to the distribution of the software, modifications that can be made, accessing the software updates and the payment procedure.

When a piece of software is purchased, it does not mean that copyrights are “sold” (but actually copyrights are reserved for the authors and creators of software). Purchasing a software means that software should only be used under certain rights imposed by the copyright owner or software publisher. These rights are explained in the “terms and condition” of a license agreement. Terms and conditions explain the use of software according to the limits defined by the owner and authors, according to the production of backup copies, distribution and transfer of software rights, etc¹.

A similar legal situation occurs between a buyer and a seller while purchasing a product from online shopping centers. Before purchasing from online shopping centers, a buyer usually has to sign an agreement. After purchasing, the buyer transfers the payment of item through a proposed channel offered by seller. Finally, after completion of the transaction the seller ships the item to the buyer, otherwise the agreement fails its completion. Whenever a purchased item received by a buyer and it doesn't fit according to defined properties specified by a seller, then according to some lines of the license agreement the seller has to reimburse the amount to the buyer. These conditions and actions are usually defined in license agreements and very often the user bypasses it without reading and/or understanding. This kind of habit makes buyers, or in some cases sellers, vulnerable to violation of agreements.

The concepts contained in agreements are often difficult to understand for a user. Moreover, the terms and conditions are usually varying in every license agreement. Unfortunately, frequently users do not read through such lengthy agreements. User agrees by signing the agreement without reading the details, such as the restrictions and legal actions applied after violation of terms. Nonetheless, agreeing with an agreement, without any deeper understanding, might lead to a user's inadvertent implication in an illegal act. For example, an end-user might be violating the law by making “illegal” copies, installing software where it is restricted by jurisdiction or distributing copies without paying a share to the author or transferring an item not according to binding restrictions. Violation of an agreement is as much a problem as accepting an agreement without understanding it. In case of software products, several agreements allow free updates, while others only allow updates by payment. Some of the agreements state that whether or not, according to the agreement, a user receives a replacement in case of any damage occurred to the product before

¹<http://www.bsa.org/country/Anti-Piracy/Why a License Matters.aspx>

receiving. An unusual number of agreements stipulate the trial period; after which, either the software stops functioning or the user is required to make payment to re-execute the software. Approximately every agreement (i.e. commercial agreements or open source agreements) includes the specification of penalties.

1.1.1 User Awareness

For the awareness of understanding agreements, few efforts are made for example; GPL-violations², it is an awareness project about the violation of GNU licensing agreements. “GPL-violation” is a communication platform between all parties that are involved in licensing of open source software for authors and copyright holders, vendors and users. The main purpose of this site is to gather, maintain, distribute information about illegal use and distribution of open source software licenses. The goal of GPL-violation is to make awareness in vendors of GPL license, to understand that GPL is not public domain, and there are terms and conditions which are to be fulfilled.

Business Software Alliance (BSA)³ is an IT industry group, that support their member groups control and prevent the piracy of software and hardware tools. BSA provides awareness and aims at educating people on the ethical and digital security risk associated with unlicensed software use. Software Asset Management (SAM) includes a software audit tool to protect organizations from the risks associated with pirated software. Different tools are developed by BSA for the security of each group of users. BSA also preforms the auditing of companies, which use software⁴ for any type of development. In 2000, BSA conducts audit of a California based company. The company had installed some of legally purchased Microsoft products on more than one system illegally. The BSA trade group challenged the illegal act and the California based company was fined with over 1 Million dollars. This is an alert for other business groups to monitor their software licenses⁵.

A number of different forums and sites are available for user awareness in this area. However, this thesis provides a semantic model that is easily understandable for the user. The model is created on the basis of user requirements. The work performed by GPL-violations and BSA serves only for the sake of user awareness; while the semantic model approach of this thesis provides a solution, in selecting and matching a required license according to the needs of a user. It additionally makes the user aware of possible penalty issues.

²<http://www.gpl-violations.org/>

³[http://www.bsa.org/country/BSA and Members.aspx](http://www.bsa.org/country/BSA_and_Members.aspx)

⁴[http://www.bsa.org/country/Tools and Resources.aspx](http://www.bsa.org/country/Tools_and_Resources.aspx)

⁵http://news.cnet.com/2008-1082_3-5065859.html

1.2 World Wide Web and Semantic Web

Web technology is highly dependent on humans, in respect of searching any required information. Searching the information through links and keeping tracks of searched information is incomplete without software agents (i.e. search engines). These software agents are used by browsers. These browsers provide the information in a human readable format.

Web browsers use Web pages that provide information in the form of hypertext documents. The traditional Web based information is the collection of information from different resources. This information is represented through a Universal Resource Link (URL), as shown in figure 1.1. A theoretical idea of proto-hypertext was coined in 1945 by Vannever Bush' "Memex". The Memex was a electromechanical device that was designed to read a large library of links and notes created by an individual or recorded by researchers(Michael, 1992). In 1959, the concepts of Memex was more clarified in "Memex II", presented by Bush. Memex II explains the way of collecting and keeping record of research work that has been performed⁶.

The idea of Memex and Memex II was further purified in the form of the so called "traditional" Web, by Tim Berners Lee in 1980. His idea was to link the existing information at different systems and share this information with non common machines and software-presentation tools. In 1990 Tim Berners Lee and Robert Cailliau unsuccessfully presented this idea in a vendor community, who could support the project. In the same year they developed a working tool for the Web. The developed tool was based on the idea of Tim Berners Lee and communicated Web data through Hypertext Transport Protocol (Http)⁷.

Http does not completely fulfill the ultimate idea of Tim Berners Lee. Semantic Web describes different methods to allow machines to understand meanings of information on World Wide Web. Semantic Web defines a knowledge domain for World Wide Web, and this knowledge domain is build by using 1) Resource Description Framework (RDF), 2) Data interchange format for example XML, 3) Notations such as RDF Schema and 4) Developing a Web model using Web Ontology Language (OWL)⁸.

The Semantic Web model for license agreements presented in this thesis uses the concept of Semantic Web for realizing a knowledge based domain for license agreements. The knowledge based domain for license agreements is modeled according to an ontological model. This ontological model is based on a license ontology which

⁶<http://en.wikipedia.org/wiki/Memex>

⁷http://en.wikipedia.org/wiki/History_of_the_web_browser

⁸http://en.wikipedia.org/wiki/Semantic_Web

is supported by semantic rules. The rules are based on user requirements and these requirements are related to agreements. The semantic based knowledge domain of software and online shopping license agreements delivers a comprehensive view of a license agreement.

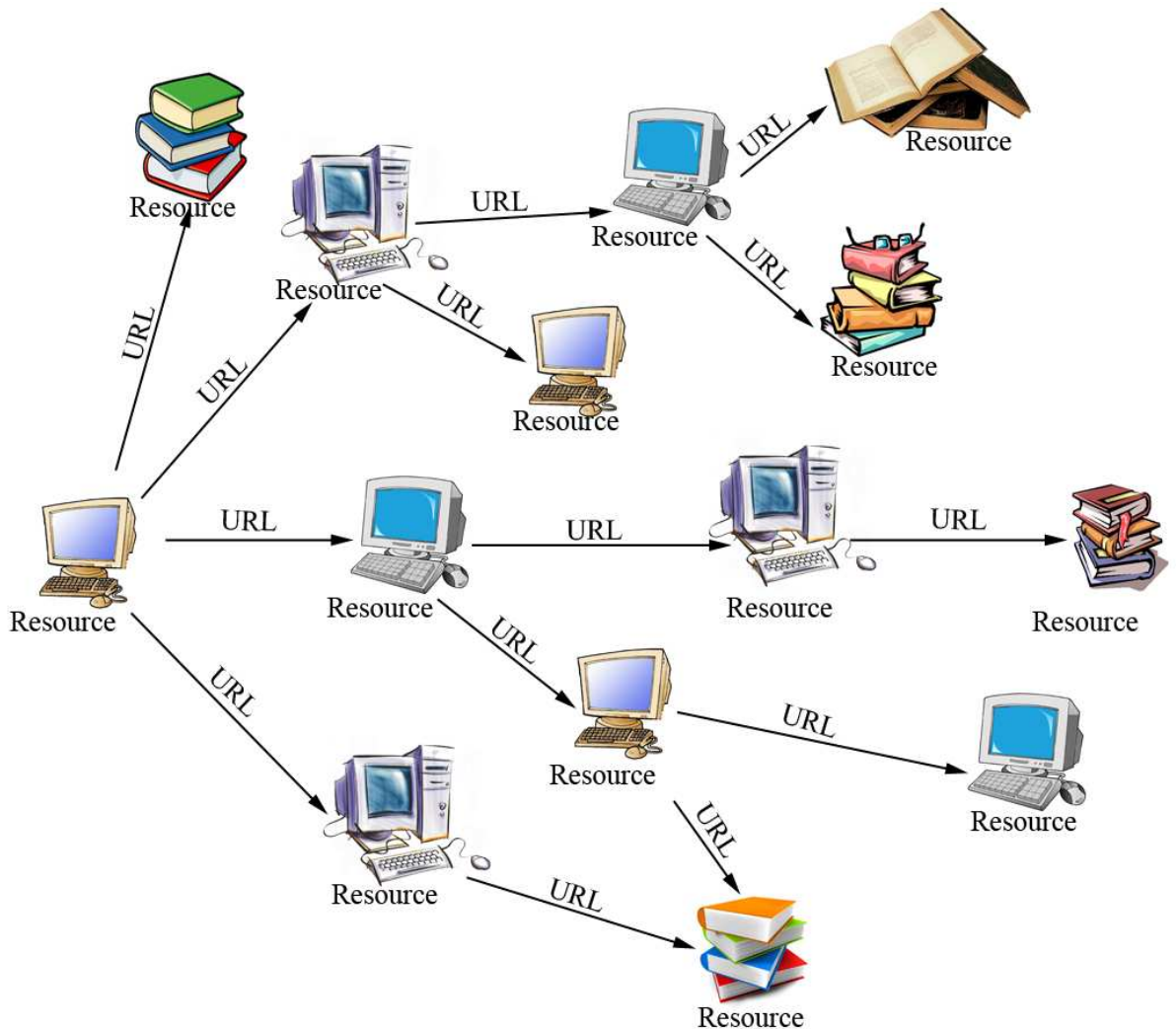


Figure 1.1: Traditional Web

1.3 Research Questions and Research Strategy

Undertaking an agreement without reading the complete license agreement, when purchasing a software product, is one of the main problems in information risk management. This problem can lead to the risk of penalties due to violation of a license agreement by an organization or individual. An automated solution can address this problem. There are several problems regarding social and theoretical aspects according to license agreements. These problems are handled in different ways such as by creating different discussion forums and some of organizations work

in the way BSA does. In relation to these problems, the questions which are prime focus of this thesis include:

To what extent can a machine simplify a license agreement for a user in understanding and identifying critical issues of a license agreement before undertaking any agreement? Which techniques are feasible to be implemented on a machine that can make it accessible to end user? To elaborate this research question and evaluate the results, we subdivide it into the following research questions;

- *RQ1: What are the main problems for a user in accepting an agreement without fully understanding or thoroughly reading the agreement? What are the problems that are generated after accepting an agreement? After identifying the problems of a user while accepting agreement, what will be a proper solution that makes the agreement easily understandable? How to bring all license agreements to one platform that could be easily accessible to all users?*
- *RQ2: To what extent Semantic Web technology is able to replace license agreements of service providers? Can Semantic Web technology provide a bridge between service provider and user? How to categorize different phases of license agreements using Semantic Web for modeling an ontology and defining a knowledge base domain of agreements?*
- *RQ3: Is Semantic Web able to provide facility to service providers in creating a license agreement for their new product? Another question arises in same context on how to update an existing ontology of license agreement for a product provider and what steps are required to present the agreement in a documented form?*
- *RQ4: How can a user and service provider, easily and effectively retrieve the required useful information from the agreement ontology? Additionally how can we overcome the problems of comprehensibility and semantic overload caused by using complex and detailed license agreements?*

In order to address the research questions, the interconnected points are to be identified for different categories of license agreements with the Semantic Web, to define a complete and comprehensive model. This model should be adoptable for distinct set of license agreements. Furthermore, the model should provide an adequate result to the end user and software developers on the basis of their requirements. Figure 1.2, gives a development overview of “Digital License Agreement” (DLA).

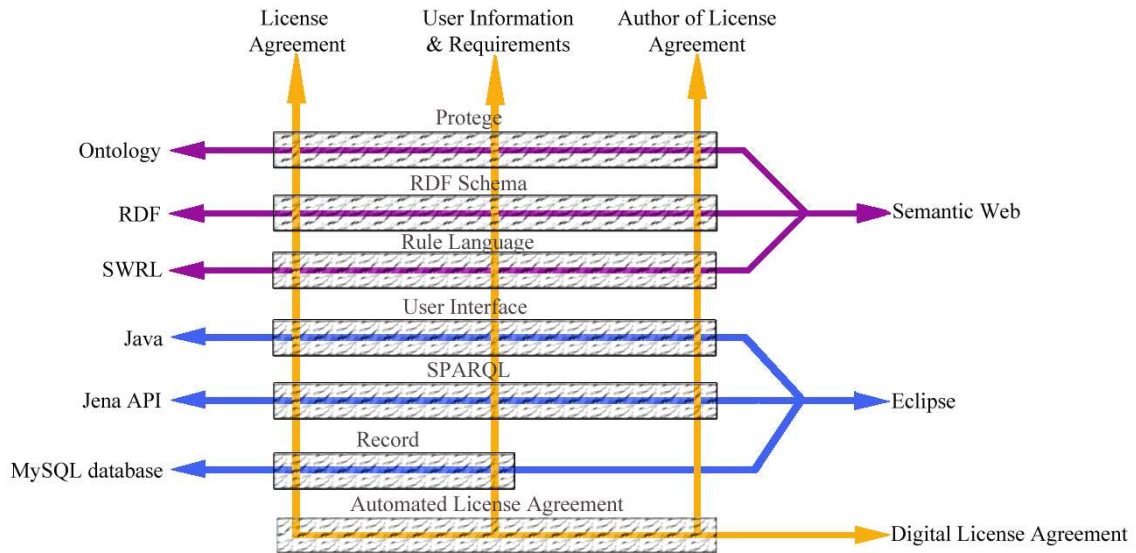


Figure 1.2: A Model of Digital License Agreement

The model depicted in figure 1.2; illustrates the interconnections of different tools, which helps a user and author of license agreement to communicate through DLA using license agreements. To develop an ontology of an agreement, we use the Protege tool. Protege helped to develop the according RDF Schema and defines rules while creating the ontology. DLA is an automated license agreement application, which provides a facility for users and authors of license agreement to explore and develop a license agreement for a specific product.

1.4 Contribution and Thesis Outline

The thesis aims at establishment of agreements as a formal knowledge model. The formal knowledge model depends on classification of different categories within an agreement and relevant concepts in subsequent goals. These subsequent goals are categorized as: 1) mapping the categories with each other in an ontological model, 2) creating triples that could exactly translate license agreement and 3) selecting those categories of license agreement which are very common in most of agreements. After designing a model, a user friendly interface is required that could provide a full picture of automated license agreements. The automated license agreement facilitates user and authors/providers in obtaining an easy to use and simple to understand platform.

Since the research results are applicable in real life scenarios, two major sections in Chapter 2, as shown in figure 1.3, discuss broadly the license agreements and Semantic Web concepts as dealt in the literature. Furthermore specific user related

problems in this domain are also discussed. Chapter 2 enlightens the digital rights management system that focuses on implementing rights and controlling possible illegal acts on software products. Chapter 3 describes the critical view of user requirements and explains the method confined to a proper selection of an agreement, which comply with the user's defined requirements. In Chapter 3, use cases and scenarios are discussed, to explain user requirements related to license agreements. Chapter 4 gives an introduction to an ontology domain and describes the development of an ontology that has been used for knowledge domain of license agreements. Furthermore, Chapter 4 also discusses different conflicting issues within license agreements and among organizations. It gives clues about the need of digitizing license agreements in this era. After constructing a License Ontology according to the common features of agreements as described in Chapter 4, it is extended to sub-license ontologies. The extension of License Ontology to sub-license ontologies are discussed and explained in Chapter 5. Each of these sub-license ontologies are created according to a specific license agreement. Selecting a proper license agreement of product, is based on rules of Semantic Web. The rules are discussed in Chapter 6. Chapter 7 illustrates each step in developing a user friendly interface using Jena API. The Chapter 7 explains the development of automated License Agreement (i.e. DLA) and describes the use of developed tool. Conclusion and future work is discussed in Chapter 8. In Chapter 8 we also discuss how our research could be broadened by using concepts of license agreements to different products other than software and online shopping centers agreements.

Figure 1.4 delivers an overview of chapters and sections used in this thesis. The main sections of each chapter, are shown in the figure 1.4. These main sections are the constituents of all the necessary development steps taken to finalized Digital License Agreement. Relations between these sections are also described in the figure to provide a complete overview of this thesis.

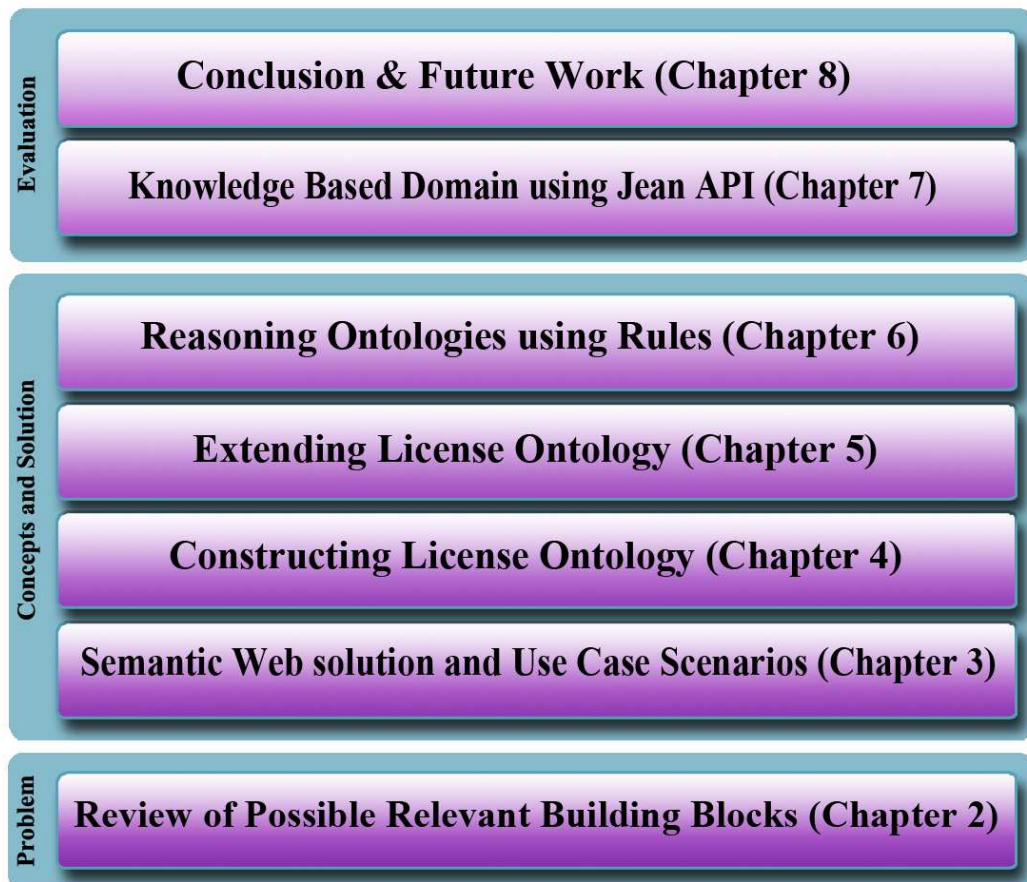


Figure 1.3: Thesis Outline.

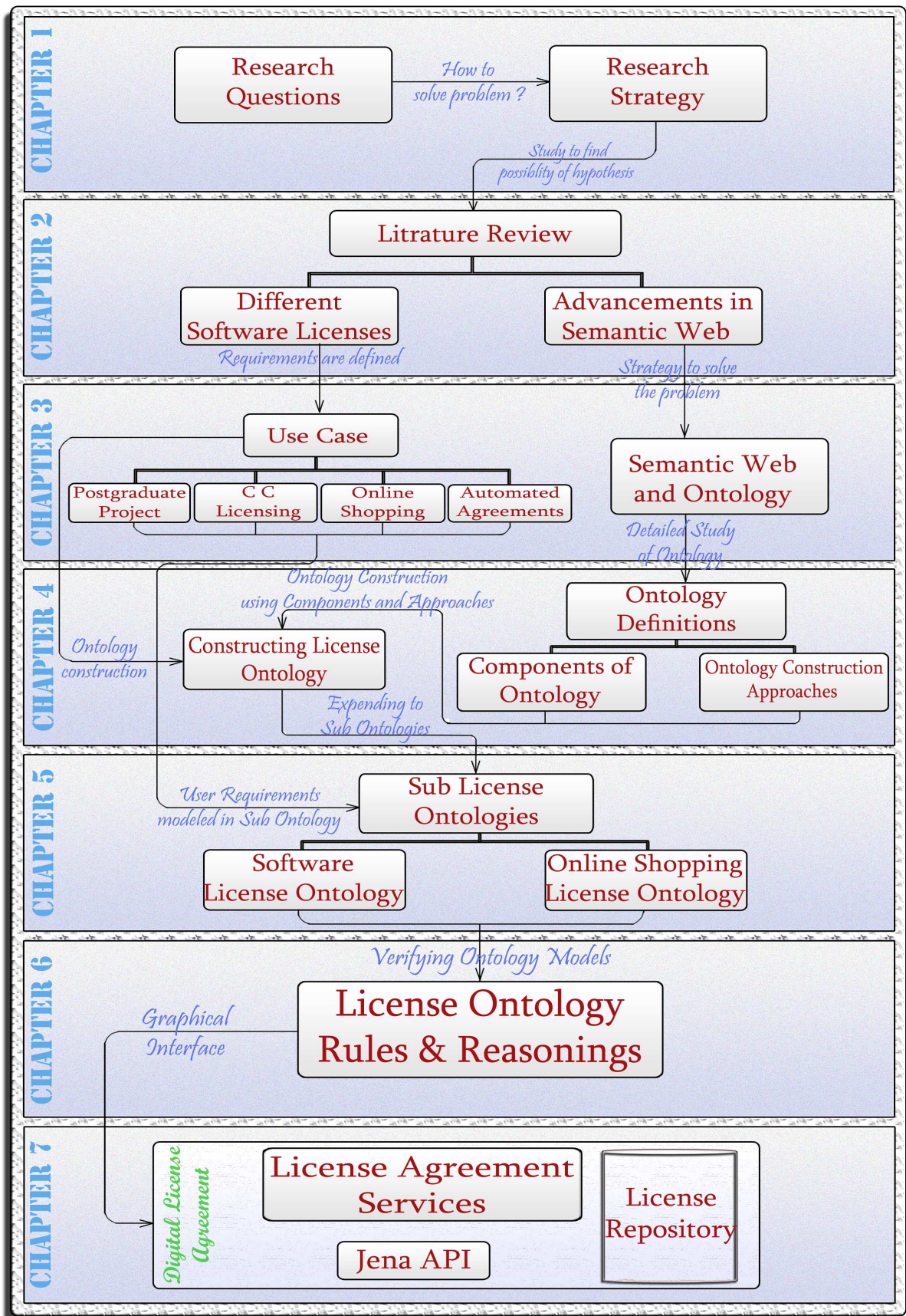


Figure 1.4: Thesis Overview.

Chapter 2

Review of Possible Relevant Building Blocks

In the literature, a plethora of studies has been conducted for developing a Semantic Web model to fulfill user requirements. The Semantic Web model is ontological model and provides a platform to collect data from different sources. The model creates an ontological repository which is important for a better performance and reliability of a system (Veijalaine et al., 2006). An ontological repository is a knowledge domain that gives meaning to requirement statements by using a semantic function. Using semantic functions an ontology could be a centralized ontology or a decentralized ontology. A centralized ontology provides an excellent performance from a functional point of view. But technically a decentralized ontology effectively supports distributed information and its maintenance. Therefore, ontology is an efficient and effective enabler to collect information from different sources and provides an easy access to all of the related information for a user.

This chapter has two main sections. First of all, it summaries fundamental concepts of license agreements, by considering different type of existing licenses, work done on licenses and its importance. Secondly, it highlights the use of ontology in information retrieval, management, automatic extracting of meta-data and different concepts related to Semantic Web. This chapter provides an overview on the relevant work done so far in the context of the thesis, and therefore provides a foundation for the thesis. According to the multifaceted nature of the thesis we have to describe the problem from more than one angle.

2.1 Software and Online Shopping Licenses

Organizations put lot of efforts to ensure that products are handled legally. Therefore, for the organizations the management of license agreements is an important issue. Different techniques are followed to control the misuse of license agreements, as discussed in section 1.1.1. Hence, penalties are defined to assure the legality of usage. The legal use of a license agreements is explained in the “terms and condition” of the license.

According to software license agreements, three type of software licenses are available i.e., 1) copyright license, 2) open source license and 3) proprietary license. The copyright license reserves rights of an owner, therefore a user faces lot of restrictions in using these software. Open Source License (OSL) is a free license provider, and provides free software which can be redistributed according to OSL. OSL is further extended to OSI and OSD as explained in section 2.1.2. Creative Commons is a type of open source license. Creative commons reserve some of the rights of owners and provides maximum usability to a user, as explained in section 2.1.3. Proprietary license are the owner’s defined licenses and they have nothing to do with any jurisdiction and international law¹(Nordquist et al., 2003).

Online shopping agreements are mostly similar to each other and have a lower degree of complexity as software licenses. In these types of agreements, rights of service providers and customers are clearly defined. In some agreements service providers are the sellers and provide a facility of purchasing from their Web sites. While some other agreements are totally different from service provides, for example in case of eBay. Only two types of online shopping agreements are explained and used as pars pro toto in this thesis, i.e. Amazon and eBay license agreements.

Various license verifying systems have been developed. These systems help to ensure that a licensed version of a product is properly and legally installed on each computer in an organization. SLMS is an example of such a license verifying system (Fujimura and Nagaishi, 2007). SLMS is implemented to verify legality of a software installation. Microsoft verifies the installing products online, before completing the installation of the software product on a system². This clarifies that the granter of license wants his product to be used legally and according to the terms and conditions defined in the license agreement. Normally these conditions are formulated with technical wordings which are difficult to understand, for an average user. Our goal of this thesis is to provide a facility, for a user to easily understand the main points of a product license which he is going to use.

¹http://simple.wikipedia.org/wiki/Proprietary_license

²<http://www.microsoft.com/genuine/downloads/faq.aspx>

2.1.1 Copyright and Copyleft Licenses

The noun “license” refers to permit another party to use one’s product, within the limits of written agreements between those parties. The granter of a license grant a license under intellectual property law to authorize a licensee in using product, but with limitation³. A license keeps the rights reserved with the owner (either all rights or some rights depending on the agreement). A license can be categorized in copyright license and copyleft license. Copyright license reserves all rights with the owner and copyleft license reserves some rights. Both are briefly discussed below;

Copyright

Copyright grants the rights to publish or sell written documents, art works, photographs or any from of intellectual property. According to copyright all rights are reserved with an author. Copyright is subjected to different state laws and regulations, written in the agreement, that govern the ownership, inheritance or transfer of intellectual property. Normally, copyrights are granted under the act of jurisdiction or international law. Transferring of copyright is usually performed under certain license agreements⁴. Each license agreement consists of terms and conditions. These terms and conditions are rights given to a user who agrees with the license. A copyright agreement secures the rights of an author and assures him that his product would not be used illegally.

An example of copyright license is the Adobe software license. It restricts the user from modification, unbinding and transferring⁵. Penalties are applied according to the agreement if any violation occurred. The Adobe user is restricted to solely use the software, without putting his efforts in modifying or transferring. A number of different license agreements applies some what similar conditions in agreements while distributing software to users.

Copyleft

Copyleft is the word used “against” copyright. In contrast to copyright, copyleft authors give rights to reproduce, adapt or distribute the product, but require that the resulting copies should be bounded by the same agreement. A copyleft license ensures that the pieces of work is freely available with verification that authors rights are reserved. General Public License (GNU) is the first dominating copyleft license.

³<http://www.en.wikipedia.org/wiki/License>

⁴<http://www.bitlaw.com/copyright/license.html>

⁵http://www.adobe.com/type/browser/legal/pdfs/EULA5seat_IntlEnglish02.03.04.html

A copyright license protects rights of an author by controlling distribution, transfer and modification, while a copyleft license allow users to redistribute, transfer and modify the work. Copyleft license or open source license are a new type of licensing agreements. They made it available to distribute modified work with acknowledging the author. With the help of copyleft licenses users are given the freedom to use, study, modify, copy and share their work with others. However, there are different free software licenses which cannot be considered as copyleft license because they don't allow their work to be modified. In creative commons licenses, discussed in section 2.1.3, authors allow users to modify and redistribute the work but with some restriction; as described in copyleft licenses⁶.

Apache and Mozilla license agreements are the examples of copyleft or open source licensing. Both license agreements allow free distribution of source code and also allow user to modify and redistribute the original source code. Apache license agreement grants royalty free patent license for users to sell, import and transfer the original source code⁷. Mozilla and Apache allow the distribution of user's modified source code and also allow users to provide an additional license agreement for the modified source code. According to the documentation of modified source code, Apache allows user either to distribute the same license agreement with some modification or develop a completely new license agreement. While Mozilla forces the user to develop a completely new license agreement with a new name of the modified product and requires that the new name should not be similar in any respect to Mozilla's products names⁸. A Mozilla license agreement will be automatically terminated when the user breaches the agreement and fails to fulfill terms and condition within 30 days. Apache and Mozilla do not take the responsibility of any damage while redistribute the product. These license agreements provide a clear picture of open source code agreements, in which the user is allowed to modify the with source code but with certain limitations.

2.1.2 Open Source License

An Open Source License (OSL) is a software copyright license. OSL is a free license which allows modification and redistribution of the software both commercially and/or privately. OSL provides the availability of source code with some required restrictions of the authors. OSL restricts a user to preserve the name of the author and the copyright statement within the code. A well known set of OSL is the Open Source Initiative (OSI) based on the Open Source Definition (OSD)⁹.

⁶<http://www.en.wikipedia.org/wiki/Copyleft>

⁷<http://www.apache.org/licenses/LICENSE-2.0>

⁸<http://www.mozilla.org/MPL/MPL-1.1.html>

⁹http://en.wikipedia.org/wiki/Open-source_license/

Open Source Initiative (OSI)

The OSI provides a platform for organizations to distribute their own work under a different method of licensing. OSI promotes the idea of general education by providing free software and protection of the identity of author while distributing. Known OSI licenses includes GNU, BSD license, MIT license, Mozilla Public License and few others¹⁰. The goal of OSI is to expand the OSL and encourage companies, projects and individuals, who distribute their work freely, to use OSI under the definition of OSD¹¹. The programmers are given feasibility for freely distributing their own work and protect the work from any misuse, and the programmers are granted the following rights;

- To make copies of program, and distribute it
- To access software source code before modification
- To make improvements in program.

OSI helps to keep the contribution at same level or relative to each other because of the given rights¹².

Open Source Definition (OSD)

OSD is the distribution terms of open source software. Open source software fulfill following criteria of OSD¹³;

1. Distribution should be free and shall not restrict others from selling.
2. Derived works and modification should be allowed for distribution under some terms and conditions of license.
3. Integrity of the author's source code should be provided while redistribution.
4. License must not restrict anyone, in using product in a business or for a research.
5. Some rights are applied to all redistribution and no more licensing is allowed.

¹⁰<http://www.open-source.org/history>

¹¹<http://open-source.usrbinruby.net/proliferation.html>

¹²<http://perens.com/OSD.html>

¹³<http://opensource.usrbinruby.net/docs/osd.html>

6. License must not insist or restrict other to distribute along with licensed software.
7. License must not restrict others to use or modify any specific technology or environment.

After completing the requirements of a license, a legal analysis report of the license that complies the terms and conditions of OSL is created¹⁴. OSL is a license authorizing organization which fulfills the stated requirements to make a balance in the open source software world. The terms and conditions are globally implemented under the specific jurisdiction. Only free software developers or distributors follow the OSL.

2.1.3 Creative Commons

Creative Commons (CC) is inspired by GNU and GPL. CC licensing provides a standardized way to copyright ones product. CC licensing help to increase the creativity in cultural, educational and in scientific content. CC licensing is available to every one for legal sharing, use and remixing others copyrighted work and supports the potential of Internet for learning and educational resources^{15,16}. It minimizes legal, technical and social barriers while making agreements (Adida and Birbeck, 2008) and serve to modify ones copyright terms to suit distributor's needs under some rights reserved by an owner¹⁷. CC license has no worries about copyright infringement, as long as the terms are obeyed and helps license to directly establish free and reserved rights of their product without requirement of intermediates (Piedra et al., 2009). CC licensing provides an individual to reserve some of the rights and distribute the work.

CC licenses can be characterized by four categories, these categories are used in combinations of symbols and keys¹⁸ and are used by a license provider according to his requirements. Each category defines a condition. Each condition or combination of different conditions makes "terms and condition" for a license. CC categories are shown in figure 2.1;

A CC license uses trade marks to identify public copyright. These trade marks are the combination of symbols and keys of categories used by a license provider. CC

¹⁴<http://opensource.usrbinary.net/approval.html>

¹⁵<http://creativecommons.org/about/licenses/>

¹⁶<http://wiki.creativecommons.org/RDFa/>

¹⁷http://wiki.creativecommons.org/images/6/62/Creativecommons-informational-flyer_eng.pdf

¹⁸<http://creativecommons.org/about/licenses/>























	Commercial User	New versions allowed or not	Keywords used
Attribution 			BY
Share Alike  		New version is allowed and new version must be licensed under share alike license	BY-SA
No Derivatives  			BY-ND
Non Commercial  		New version is allowed and new version must be non commercial, but can be under non commercial license	BY-NC
Non Comercial Share Alike   		New version is allowed and the new version licensed under a non commercial share alike license	BY-NC-SA
Non Commercial No Derivatives   			BY-NC-ND

Figure 2.1: CC license types.

licensing provides the facility for a user to select license according to his requirements to share his work. CC offers six different content licenses¹⁹. The contents are listed below;

1. Marking content
2. Marking Text
3. Marking Images
4. Marking Audio
5. Marking Video
6. Publishing work through file sharing or social networking

CC collaborated with different countries all over the world for global licensing²⁰. CC organizes open forums so called “salons”. Salons give the opportunities to license different events focused on building communities of artists, developers and creators

¹⁹http://wiki.creativecommons.org/images/6/61/Creativecommons-licensing-and-marking-your-content_eng.pdf

²⁰<http://creativecommons.org/international>

of all kinds²¹. Pootle server is used to manage translations of CC licensing in different languages around the world²². CC licensing is translated in more than 55 different languages. Each license is completed according to associated jurisdiction.

2.1.4 Online Shopping Agreements

Idea of online shopping starts from radio and television broadcasts. Radio and television were the attaching point for customers and sellers to make publicity through these inventions. In 1994 Pizza Hut was the initial online shopping store, available for customers²³, and successively in 1995 Amazon and in 1996 eBay appeared. In 2006 a study conducted in Canada about customers using online shopping centers revealed that 48% do research on Internet prior to purchase, 41% go directly to retail site and make a purchase, and the last 11% buyers are impulsive buyers, who make shopping without making any survey (Matthew, 2006). This identifies the importance of online shopping in current age.

Online shopping agreements are significantly simpler compared to software license agreements. Mostly, online shopping centers are service providers and selling their own products, while in some cases sellers are not the owners of online shopping centers (e.g., eBay shopping center or willhaben²⁴). In online shopping agreements, the main focus is laid on issues according to the customers, the billing system, the reimbursement in case of complaints and in some cases the seller's rights before launching any item for selling. Therefore, the approach used in this thesis is extended by incorporating online shopping agreements.

2.2 Semantic Web

Semantic Web Technology is an emerging development of the World Wide Web (WWW). Semantic Web technology provides a common platform for data to be shared and reused across applications. The word "Semantic" describes the study of "meaning", therefore it describes information on Web and make this information possible for a system to process. Semantic Web is composed of designed principles and various technologies such as Resource Description Framework (RDF), different data interchange formats (for example XML, N3, N-Triples), notations e.g. RDF Schema (RDFS) and Web Ontology Language (OWL).

²¹<http://wiki.creativecommons.org/Salon>

²²<http://wiki.creativecommons.org/Translate>

²³http://en.wikipedia.org/wiki/Online_shopping

²⁴<http://www.willhaben.at/>

World Wide Web is a radical way of thinking about sharing information. Quality of information can be achieved by enforcing Semantic Web techniques. Semantic Web techniques provide a resultant design of given information and strongly influence to produce a quality of application, which answers all the required queries of a user.

Diversified work has been done in Semantic Web for example in digital right management (section 2.2.1) and policy management (section 2.2.2). Semantic Web is in evolutionary stage and lot of different efforts are being made in this area. Some of the work is described below.

2.2.1 Digital Rights Management

Digital Rights Management (DRM) is an access control technology that is used to impose limitations on usage of digital contents and devices. It controls the illegal use of digital media by preventing access, copy or conversion to other formats by end users. DRM is commonly used in entertainment industry, online music stores, e-book publishers, etc ²⁵. Semantic Web technologies facilitates in creating ontology for copyright agreements. These copyright agreements are digital rights used to implement legal limitation on a user. Along with copyright ontology Semantic Web is also used in other respects such as Right Expressive Languages, discussed and others.

Digital Rights Expression Languages

Digital contents are distributed over the Internet by a regulated law. The rights of using digital contents are described in licenses and these licenses are adopted by Semantic Web. Rights Expression Languages (REL) reflects licenses legal requirements, by gathering terms and required relations of contents. REL creation is based on an ontology using existing standards for example ODRL, XrML and Creative Commons. A Digital Right Management system controls the authorization and use of contents for REL. Usage of contents depends on the rights of a license provider, granted to licensee. REL with DRM translates licenses in a generic way, therefore this translation of licenses can also be used in other legal systems. REL uses an ontology to deduce copyrights, context description and different type of expressions used in a license agreement . The REL ontology terms are mapped to a legal dictionary. This legal dictionary consists of legal classifications and legal rules used in the license law. Translations of legal terms definitions and interpretations by REL

²⁵http://en.wikipedia.org/wiki/Digital_rights_management

into other standards like ODRL and XrML is used to complete the dictionary of legal terms (Nadah et al., 2007).

Digital Rights applied to Copyright

In both Internet and Intranet, contents (i.e. reading materials, music, etc) are remixed and then distributed. Advancements in computing power makes it easy to manipulate content. Semantic Digital Right Management (DRM) is applied to protect copyrights misuse by remixing contents with Mashup and Syndication technology. Semantic DRM inserts tags into contents for tracking and managing contents. Social awareness is used to manage tracking of contents. Social awareness handles various social activities concerning Meta contents which is maintained by social networking. Semantic DRM track various tags used in those particular networks. Semantic DRM is helpful in easy transferring rights of contents to other individual or organization (Kang et al., 2009).

RDF Meta-data Diffusion

The work in Semantic Digital Rights Management for Controlled P2P RDF Meta-data Diffusion is carried out to develop a legally binding agreement between a user and a provider. A small ontology is proposed to implement this handshake mechanism. A four step exchange process is used in which, 1) user makes a request to a provider, 2) on the basis of this request a proposal is made, 3) the proposal is then sent to the user, 4) and finally the user signs the proposal and sends it back to the provider. A secure channel is created between user and provider. After creation of secure channel, product and related information are transferred between user and provider (García and Tummarello, 2006). The work defined above is similar to the work described in this thesis; however user requirements are considered as a canonical reference. A user lists his requirements and these requirements are fulfilled by selecting the appropriate license agreements using ontology description of these agreements.

2.2.2 Policy Management and Semantic Web

A number of different policy management methods provided among others a flexible management of Web security are being proposed. As an example Rein (Kagal et al., 2006) is an advanced approach in the field of policy management using Semantic Web. Rein defines decentralized policy domains and a reasoning engine, by using

ontology concepts. Semantic Web provides reasoning engines for policy management system to identify the security breach issues. Some of research studies related to policy management using Semantic Web are discussed below;

Privacy Policy Matching

Nyre discusses the issues related to “privacy policy matching”(Nyre et al., 2011). The solution suggested in the paper is to arrange policies of service providers according to customer requirements. Customer requirements are translated to policies using privacy preferences of the provider, for matching the customer requirements with the service provider policies. Privacy preference provider is introduced as an intermediate channel between service provider and customers; to solve, the issue of matching and arranging the policies for service providers.

According to this method, a customer defines his requirements at the “privacy preference provider” for purchasing an item. These requirements are henceforth handled by the “privacy preference provider” for the current purchases and also for future usage. For purchasing an item, the customer receives a message on the Web site of the service provider about the availability of the selected item, according to the defined requirements.

The word “policy” used in this method is similar as terms and conditions of license agreement. These policies are defined by the service provider, in the example case of online shopping license agreements. The main problem of this method is to maintain an intermediate channel as “privacy preference provider” for updates of customer requirements. The second problem is to translate customer’s requirements to policies, at this intermediate level, and then match these policies to service provider’s policies. Handling these problems, according to the above method, are worse in case to deal with complex policies of service providers.

To solve the above mentioned problem, we propose in this thesis a solution using Semantic Web technologies.

KAoS

KAoS is a service provider for clients communication. KAoS uses a hierarchy of ontology concepts to build policies. The service confirms a client’s defined policy. The client’s policy is automatically planned and executed in semantically described work-flows. The KAoS service, permits listing, managing, solving conflicts and enforcing policies in specific contexts (for example, a client’s context). KAoS manages

the policy in 4 steps, initially by creating a grid service for naming a client, his message transport and the directory management. After creating a grid service, a restoration method is followed to register clients in the KAoS domain for his resources. The third step expresses client's policies for authorizing actor/s, his actions and target/s. The method to express client's policies is carried out using KAoS policy ontology, the ontology helps to map various actor's classes and instances to execute an action on target/s. Finally the authorizations are checked; the KAoS Service has full control of access to provide resources based on permitted rights and KAoS serves as a policy enforcer in using resources (Uszok et al., 2004). KAoS policy mechanism is based on OWL.

Semantic Mapping Framework

Semantic Mapping Framework (SPDM) is used to map security policies of selected virtual organizations (VO) with the real organizations (RO), in a six step approach: 1) security policy ontology for VO and RO is created, 2) reasoning process is carried out (this reasoning process consists of checking consistency, classification and inferences), 3) checks on data type violations, 4) global security policy is generated, 5) merging similar instances and lastly 6) results are generated. The above steps are performed to update the global policy after mapping all VO security policies with RO. Semantic Web helps to maintain consistency in mapping global policies while updating security policies of VO and RO. A rule ontology and a rule parser are created to perform semantic mapping using JENA in aligning global policies and mapping security policies of VO and RO (Muthaiyah and Kerschberg, 2006).

Policy in Network Domains

Policy interactions in different network domains is achieved by using semantic reasoning. An architecture is defined (i.e. PRIMO) to deliver users context aware and QoS (Quality of Service) dependent services, while conforming with service providers and network operators. Semantic reasoning is based on those policies that interact between services of the system, for matching and transforming events. The reasoning process ensures that events are syntactically and semantically compatible to the system. Semantic Web technology makes the effort of network operator simple. Once a service is created by a network operator the user interacts with the system directly and his interaction with the network service generates customized services as required. As per requirement the network operator intervenes in system in the case of defining new policies or when the system does not has enough information to

process a decision. The PRIMO architecture provides an alternate solution towards autonomic systems adaptation (Davy et al., 2007).

In this chapter different aspects of license agreements and Semantic Web are explored. The literature suggests that the concepts of Semantic Web and ontology are applicable for license agreements. It is observed that there is a need to construct a significant license ontology which could interconnect all types of license agreements on one platform. Therefore, Semantic Web approach is used to construct license ontology which would be helpful in future to extend the license ontology for different types of license agreements. In chapter 3, use case scenarios are explained which elaborates user requirements. License agreements are selected on the basis of the user requirements defined in use case scenarios.

Chapter 3

Semantic Web solution and Use Case Scenarios

Semantic Web is an inspiration of a current Web, influenced by the earlier work of Vannervar Bush's¹, as discussed in section 1.2. In 2001, a Scientific American article described Semantic Web as the evolution of a Web that consists of documents. These documents include data and information for computers to manipulate. The Semantic Web is a Web of actionable information. The information is derived from data through semantic theory. The semantic theory provides “meanings” to information, which is a logical connection that establishes between systems. This semantic theory was proposed by Tim Berners Lee article, at World Wide Web Conference in 1994. Semantic Web is an evolutionary field of informatics. Most of the work being done in this field using ontologies (Schadbot et al., 2006). Ontologies provide a support to the Semantic Web in distinct communicates of scientists. Scientists, researchers, clinical drug trials and other group all needs to integrate their related components. Different organizations, for example environmental sciences² and e-government sectors in EU³ and United Kingdom^{4, 5} represent their efforts in generating projects using Semantic Web and its extensions.

Semantic Web organizes distributed information in a meaningful environment on the Web. Semantic Web provides an environment where similar information is gathered easily as compared to current Web's results. It gives a well defined meaning to information and thus enable computers and people to work in a co-operated environment. Information on Web is scattered in chunks on Internet and its a

¹<http://en.wikipedia.org/wiki/Memex>

²<http://marinemetadata.org/community/teams/ont>

³<http://www.ec-gis.org/document.cfm?id=486&db=document>

⁴<http://www.opsi.gov.uk>

⁵<http://doc.esd.org.uk/IPSV/2.00.html>

tiresome job to acquire the required information, from this scattered information. In comparison to current Web, Semantic Web performs its functionality as distributed as possible. Such distributed systems gather lot of information at each level, and present the distributed information in a collected environment (Berners et al., 2001). Tim Berners-Lee expresses his vision of Semantic Web as follows;

“I have a dream for the Web ‘in which computers’ become capable of analyzing all the data on the Web – the content, links, and transactions between people and computers. A ‘Semantic Web’, which should make this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The ‘intelligent agents’ people have touted for ages will finally materialize.” Tim Berners-Lee 1999⁶.

Semantic Web uses XML & RDF technologies to develop the ontological model. XML adds an arbitrary structure, without defining any meaningful structure. RDF uses triples to express an elementary sentence. The triple gives meaning to an XML structure. Therefore the Semantic Web is used to implement semantics of License Agreements, as explained in the section 4.4. Initially, use case scenarios are defined to identify requirements, that lead to construct an ontology structure and related rules. The ontology and rules of License Agreements make inferences on the basis of user requirement, which are already defined in use case scenarios. This chapter discusses Semantic Web, its supporting models, use case and user scenarios, which set the foundation to define requirements. These requirements are further refined to construct ontologies, as discussed in chapter 4. Rules explain the main structure of ontologies, which provides complete meanings to Semantic Web in the thesis.

3.1 Semantic Web Cake Layered Approach

Semantic Web Cake also known as Semantic Web Stack, is a hierarchical structure which shows an extension of classical Web technology. Semantic Web Cake layered architecture is proposed by Sir Tim Burners-Lee, who specify different layers of Semantic Web, as shown in figure 3.1. This architecture can be divided in three layers according to Web technologies and each of these layers have sub-layers that provides explanation of its development phases⁷. Each of the three layers are discussed below;

⁶http://en.wikipedia.org/wiki/History_of_the_World_Wide_Web

⁷http://en.wikipedia.org/wiki/Semantic_Web_Stack

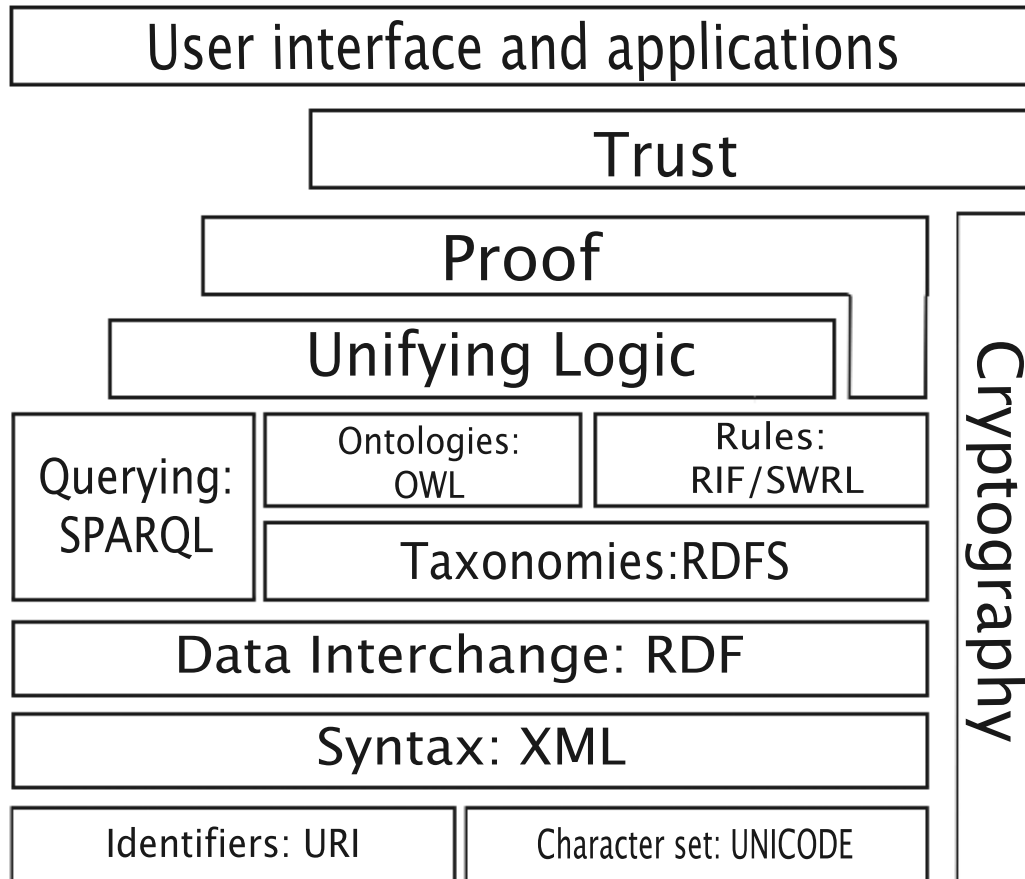


Figure 3.1: Semantic Web Cake Layered by W3C

3.1.1 Hypertext Web Technologies

The bottom layers contain Hypertext Web technologies, that provide a base for Semantic Web. In this part, Universal Resource Identifier (URI) and character set UNICODE provide a unique identification to the Semantic Web resource. URI, UNICODE and XML is used to avoid vocabulary conflicts.. XML name-spaces facilitate Semantic Web to connect data together and refer more sources in one document.

3.1.2 Standardized Semantic Web Technologies

The middle layer is standardized by W3C. RDF, RDFS, OWL and SPARQL are used to create a standardized application of Semantic Web. On top of the XML layer, triple based assertions of RDF enables to represent information about resources in the form of a graph (i.e. in hierarchical from). RDF is supported by the basic vocabulary of RDF Schema that creates hierarchies of classes and properties. The

Web Ontology Language (OWL) extends the RDF Schema by adding cardinality, restrictions of values, transitivity and brings reasoning power to Semantic Web. SPARQL is an RDF query language used to query any RDF based data. SPARQL is necessary to retrieve required information for Semantic Web application.

3.1.3 Unrealized Semantic Web Technologies

The top layer is not yet standardized and it contains ideas, whose implementation realizes the Semantic Web. Rule languages (i.e. RIF and SWRL) allow to infer knowledge and make decisions. Rules provide standard way to query and filter RDF. The trust level establishes trust between applications. Cryptography ensures and verifies the flow of statements from trusted sources in Semantic Web and User Interface Layer enable humans to use Semantic Web applications (Daconta et al., 2003).

3.2 Ontology Language

Different research groups have identified the need to design a powerful ontology modeling language. They proposed a DAML+OIL language, as the modeling language for ontology. DAML+OIL was the starting point for W3C to define the Web Ontology Language (OWL). OWL is then accepted as a standardized ontological language for Semantic Web. The ontology language provides a base to write explicitly conceptualized domain models. These models require well defined syntax and semantics, efficient reasoning support and sufficiently expressing required information.

OWL is further divided into three sub-languages, i.e. OWL Full, OWL DL and OWL Lite. OWL Full uses all OWL language primitives, which includes possibility to change meanings of pre-defined primitives, for example cardinality constraints. The advantage of OWL Full is to validate RDF/RDF Schema conclusion as a valid OWL conclusion. The second sub-language OWL DL, is specific to description logic. OWL DL restricts constructors from OWL and RDF to ensure that ontological model should corresponds to description logic. OWL DL permits reasoning support on ontological model. Finally, the third sub-language OWL Lite further limits OWL DL by excluding enumerated classes, disjoint statements and cardinality. OWL Lite is easy to understand and to implement (Antoniou and van Harmelen, 2003). Therefore, OWL Full is selected to implement License Ontology. OWL Full provides a full support for reasoning and draws conclusion. The License Ontology and its extension are discussed in chapter 4 and chapter 5 respectively.

3.3 Rule Language and Inferences

The development of a rule language for Web is recently explored. The tradition of rule based systems started in the days of expert systems. Rule based systems are used for constructing Semantic Web models, just like in modular and procedural languages in the software engineering domain. Many issues related to OWL are addressed by rules for example multiple properties, multiple inverse functionality and approaches to reasoning. OWL Full provides facility of inferences and reasoning of an ontological model. W3C standard groups are exploring the development of rule languages for Web and trying to define relation between OWL and rule language (Allemang and Hendler, 2008). In the proposed work rules are used to design License Ontologies on the basis of defined requirements. Rules of License Ontologies are discussed and explained in chapter 6. Inferences derive new information (i.e. results) from information. Querying is the form of inference, which enables to infer required results from a knowledge domain. Reasoning of ontology is possible with inferences and rules (Berners et al., 2001). The means to use rules and inferences in Semantic Web is to choose course of action that could answer questions.

3.4 Use Case Scenarios

Use case and user scenario are basically functional analyzing techniques to specify process in detailed steps. A use case is a generic scenario and describes a kind of interaction with an interface, while a user scenario is a concert description of specific interactions. User scenario elaborates use case into precisely defined system behavior, with a set of assumptions and its out comes. Many scenarios could be derived from a use case. A user scenario is explained in detail from the user's task perspective and also defines exact behavior of an application. From the software engineering point of view, user scenarios are defined in early stages of the development cycle. The early stage definition helps to design a prototype, which is an initial user interface of an application. User scenarios can be regarded as brainstorming of ideas to define the requirements, detailed understanding of the problem and out come of processed information.

The research explained in this thesis, is based on a use case in section 3.4.1 and four user scenarios in sub-sections of 3.4.1. The use case explains the need of an automated system, which translates license agreements. User scenarios defines requirements of end users and software developers. The use case and user scenarios are explained in following sections;

3.4.1 Use Case

Every organization has a license for each of their product. The license of a product is an agreement between the organization and an end user. End-users need to sign licenses to seal an agreement with the product providers. Habitually, users agree with the license (i.e. terms and conditions) without fully understanding the agreement. To address this issue, an automated system is required to translate a documented license agreement into a knowledge based domain.

The required automated system is able to categorize all licenses in the same frame of reference for end use understandability, as shown in figure 3.2. The system should provide a facility for an end user to search for his required category/categories in a license. Furthermore, the required automated system has to facilitate software developer to generate a text file. The text file has terms and conditions defined by a software developer. These terms and conditions are generated by selecting required options in the automated system by a software developer. The automated system is designed in such a way that every documented agreement can be easily adopted in Semantic Web. The Semantic Web model of agreements covers all common features of an agreement. The automated system facilitates users by providing a space to add extra features other than previously defined common features. Four user scenarios are discussed below, which explains the above defined use case.

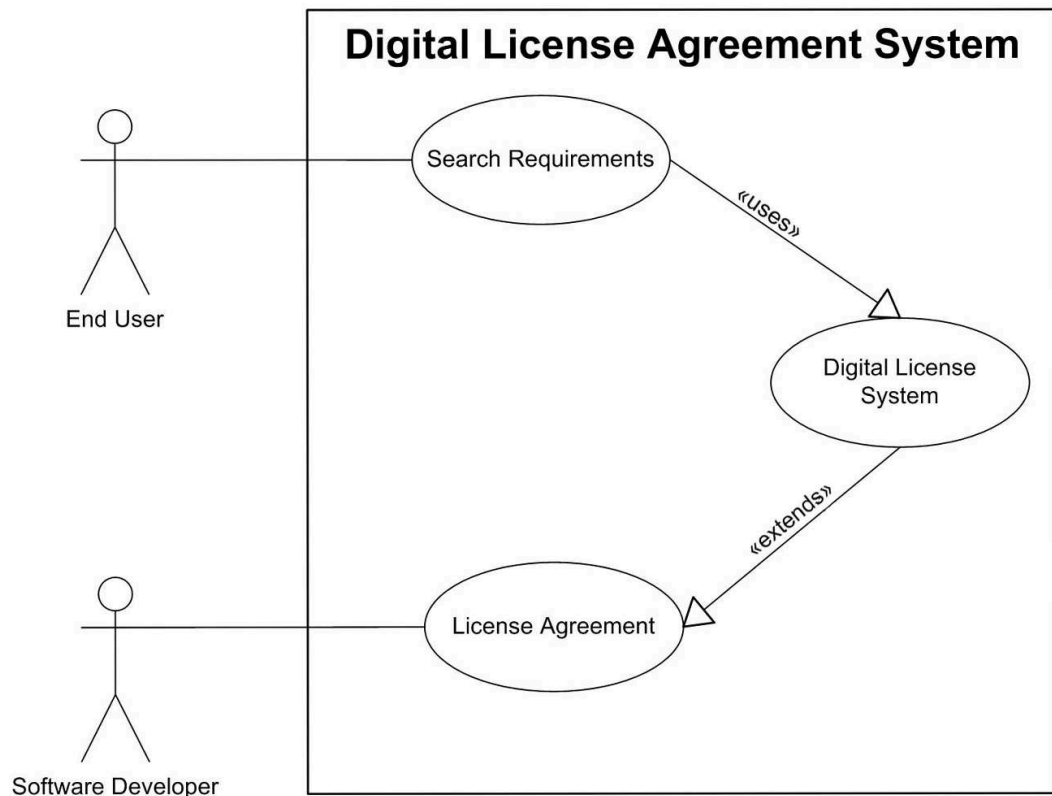


Figure 3.2: Use Case of Digital License Agreement System.

Projects for Postgraduates Students

A university introduces a new course outline for their postgraduate students. In the course outlines, a research project is made compulsory for every student by the university, as their final year project. Therefore, the university sets up a SOLARS computer laboratory for their students. Sun micro system and related applications are installed in the laboratory. Students of the university are allowed to order software products for their projects. Ordering for a software product should follow conditions defined by the university. The conditions are explained below;

A student could order those software products, whose payment process is on-line, whether payment is through bank or by third party. The product could be downloaded or received through post. The damaged product should be replicable and backup copy of original product should be allowed, according to an agreement. Only those software products are allowed to be purchased whose source code is available and modification should be allowed within the source code. The agreement of the software product should allow, to add a logo of the university in executable file after modification. The agreement should also give privileges to download the modified version of software, from university's Website.

The requirements defined above are formulated as follows:

1. On-line payment will be made.
2. Product can be downloaded or received by mail.
3. If the product is received damaged, a replacement is required.
4. Source code is required for the development of additional components or changes could be applied to the existing source code.
5. The product development will take place on SOLARS systems only.
6. Backup copies are necessary in case of error in development period.
7. The university logo will be placed on the modified version.
8. Modified versions will be available for everyone to download from the university Website.

These eight rules are used to develop a formal description of organization's requirements. Following three widely used licenses namely Mozilla, Apache and Adobe license agreements are selected, according to requirements of the use case scenario.

Creative Commons Licensing

An organization needs a source code for their specific purpose, with the following requirements:

1. Resource need to be downloaded to fulfill the requirements.
2. The resource is restricted by creative commons (by_nc_nd). According to creative commons, it can be downloaded but not distributed and not allowed to modify, and user should give credit to owner according to license agreement.

Normally the organization distributes the product after modification and crediting the owner, according to an agreement. In the given scenario the organization has a conflict in defining the policy using CC restrictions. The conflict occurs due to non commercialization and non distribution. Due to this conflict the resource is not allowed to download. Apache license agreement provides a user royalty free to copy, distribute, modify and irrevocable copyright to use the work. Therefore, Apache applications are appropriate for this use case scenario.

Online Shopping

A seller wants to sell his items on Internet. And the seller is not clear about the price. The used items are placed for the bid by the seller. Some buyer found the items relatively cheaper as compared to other online shops. They placed bids and one of the buyer bought an item.

A buyer received the item, but it is not in good condition and also has functionality problem. According to terms and condition of Online Shopping organization, a buyer can claim for reimbursement. Therefore, due to poor condition and functionality problem of item, the buyer claims for reimbursement. But unfortunately the seller was new and didn't define a proper procedure of reimbursement. The online shopping paid for the damages. And the online shopping ceased the seller's account and banded him from selling his item from the site and also from other online shopping sites. Therefore, the seller cannot do any type of business from any of online shops. If he knew the rules and obey them accordingly then he had never faced the problem. This normally happens when user bypass, by just clicking the check box of terms and condition, before creating any type of account. Above scenario is formulated as following;

1. Seller wants to sell items through Internet.

2. Seller wants to offer his items for bidding.
3. Buyers took part in bidding because the price is attractive. One of the buyer won the bid and bought the item.
4. The received item didn't work and expired with in the warranty time.
5. According to terms and condition of specific online shop, buyer has right to apply for reimbursement in such cases.
6. Buyer applies for reimbursement.
7. The seller didn't place information related to reimbursement because he didn't read the terms and condition of online shopping. Therefore, online shop paid for the damages.
8. Online shop stopped the seller from further selling items from their Website and also announced him as a law violator to other online shops.
9. Now seller is not allowed to offer his items for selling from any of online shop.

Generating Automatic License Agreement

Writing a document is a tiresome and time consuming job. Lot of efforts are required to compile a document for an agreement. The document includes legal terms and condition and other jurisdictional statements.

A software developing company needs to formalize a license agreement for trading purpose. They want to define their terms and conditions. Therefore, a system is required, which may generate such documents automatically.

The software developing company wants to compile a license agreement for an open source code software. They distribute their software products freely to public but with some restrictions defined below;

1. Software is freely distributable and it is not allowed for third party distributors to charge any type of fee while downloading the software.
2. Third party distributors can distribute the software from their personal Websites, but terms and condition of the license agreement should remain same.
3. End user can easily transfer the software and its license to other party. After transferring agreement and the software, terms and conditions of the license agreement are equally applicable.

4. Backup copies are allowed.
5. End user can modify the source code of software according to his needs.
6. After modification, end user can write a license agreement with the consideration that original software license terms and conditions are equally applicable.
7. The software is freely distributable in every part of the world.

These 7 defined rules are required by the company to automatically generate a documented agreement. Amendments will be allowed to change or modify the agreement in future. Such system allows end users to select their requirements, and after selecting requirements, system generates a text document. This text document would be in electronic form, which can be printable and also made available on Web.

Above described user scenarios provides a complete sketch of Digital License Agreement. The requirements defined in each of the user scenarios are the requirements to design Digital License Agreement application. These user scenarios are portrayed in diagrammatic form; as shown in figure 3.3.

In this chapter use case scenarios are described according to user requirements. The use case scenarios helps in selecting license agreements according to the user requirements. License Ontology and its sub-license ontologies are designed on the basis of selected license agreements. The License Ontology and sub-license ontologies are discussed in upcoming chapters, i.e. chapter 4 and chapter 5 respectively.

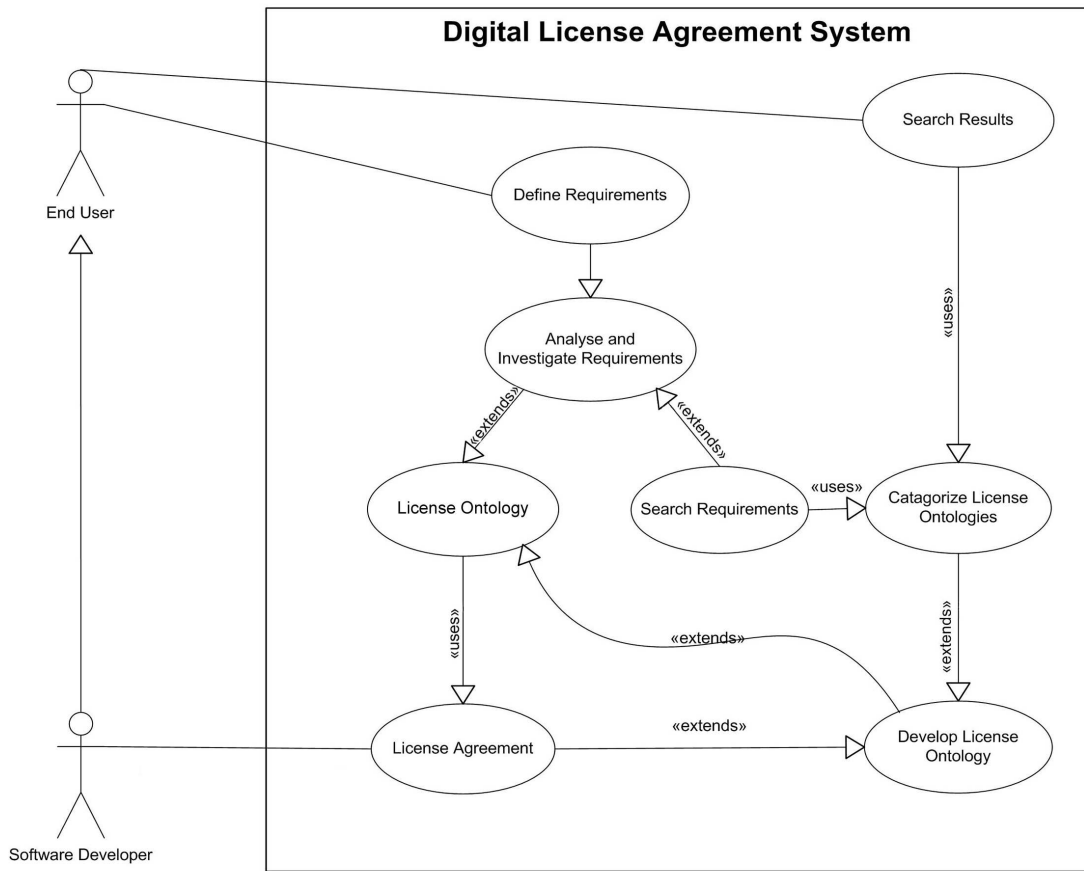


Figure 3.3: User Scenario of Digital License Agreement System.

Chapter 4

Constructing License Ontology

Automatic matchmaking in semantics is a challenge for the Semantic Web. Mainly ontologies are developed to implement software systems that focus on specific problems. The focusing of such ontologies is its usability but it is not the reason to extract theories from these ontologies. Ontology is the philosophical study of being in general, or of what applies naturally to every thing that is real. What are the things of ontology and how can they be classified to describe a full meaning? These questions are discussed by number of philosophers since Geek time. Among those, who discussed prominently the ontology as philosophical issues are Plato (427-347 BC), Aristotle (384-322) BC and Kant (1724-1804) ¹. In 1730 Christian Wolff (1679-1754), gave an idea about ontology in his book “First Philosophy or Ontology” in Latin, which in early 20th century adopted by Edmund Husserl (1859-1938). Edmund Husserl called Wolff’s philosophy as *Formal Ontology*, and provides a foundation for ontology research in computer science. Tom Gruber defines ontology as a set of representational primitives that helps in modeling a domain of knowledge and categories the representational primitives in classes, attributes and relationships among the classes ². Ontology is a philosophical concept, which is base line on the mathematical characteristics and cannot be described in formal definition. Introduction of ontology in computer science provides a new edge to scientists and they gave many definitions but ontology description and its algebra still belongs to the starting stage (Wang et al., 2009).

This chapter is organized as follows: in beginning of the chapter an ontology definition section describes ontology and its meaning, in the field of information technology. Then main components of ontology are explained with its classifications. License Ontology section describes the creation of License Ontology model and pro-

¹<http://www.britannica.com/EBchecked/topic/429409/ontology>

²<http://tomgruber.org/writing/ontology-definition-2007.htm>

vides explanation that how each license agreement's information used in the License Ontology model.

4.1 Ontology Definitions

Ontology is used in different fields of informatics to represent a domain of knowledge; for example, artificial intelligence, systems engineering, software engineering, biomedical informatics, library science, enterprise bookmarking and information architecture. Basically, ontologies are designed for knowledge sharing and reuse. Ontology is defined by Genesereth & Nilsson, 1987 as;

“A body of formally represented knowledge is based on a conceptualization: the objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them (Gruber, 1993).”

Semantic Web uses ontology as its basic component and uses taxonomy of information as an expressing knowledge to design an ontology. In representing a knowledge domain through ontology, following steps are taken to design a Semantic Web model. Figure 4.1 describes a complete spectrum of ontology (Daconta et al., 2003).

4.1.1 Taxonomy and Thesaurus

Taxonomy of ontology is the ill defined semantics to represent some information that is the minimal knowledge about hierarchical structure of parent and child (Obrst et al., 2001). It defines the parent and child relations between the classes and subclasses of an ontology. A semantically stronger taxonomy is the backbone of conceptual models and ontologies. Thesaurus is defined as a controlled, vocabulary arranged in a known order, so that equivalence, homographic hierarchical and associative relationships among different terms of knowledge domain are displayed clearly. Therefore, it can be said that thesaurus uses structured vocabularies in describing an object (Wielinga et al., 2001). Thesaurus ensures that concepts should be described in a consistent manner. This helps to refine search by experienced users, who could locate the required information and users need not to be familiar with technical or local terminology. Taxonomy and thesaurus of ontology spectrum are shown in figure 4.1.

4.1.2 Conceptual Model

To express the subclass of relation between a parent class and a child class is possible by conceptual model. Conceptual model expresses semantic of subclass relations between two given classes. Object model of a conceptual model construct domain; and system and meta model defines classes, relations and attributes, which are used at object model to define a complete conceptual model. Meta model level is the level where conceptual model is defined, as shown in figure 4.1. Enhanced Entity Relational (ER) model is used to define a conceptual schema where initial conception of the knowledge domain is modeled. Conceptual model defines a complete and understandable concept of a knowledge domain.

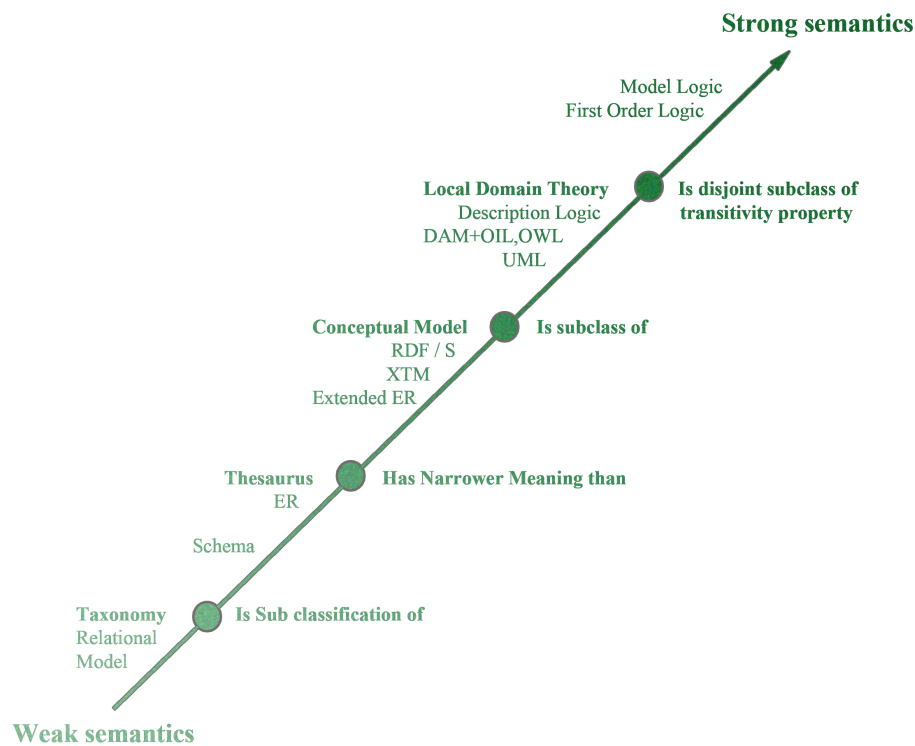


Figure 4.1: The Ontology spectrum: weak to strong semantics

4.1.3 Logical Theory

The upper right endpoint is a logical theory, as shown in figure 4.1. Most of current ontological engineering and knowledge representation are aimed to build ontology as logical theories. Logical theory expresses semantics of model to the highest degree possible, in which disjoint subclass provides a rich relation with the property of transitivity. Logical theories are built on axioms and inference rules, that are used to prove theorems about the domain representation.

4.1.4 Ontology

The ontology spectrum ranging from taxonomies to logical theories and defines the common words and concepts used to describe the representation of knowledge and standardizes the meanings. Ontologies are used to share domain of information therefore, ontologies include definition of basic concepts in domain and relationships among them, which helps to encode knowledge in a domain, so that knowledge is reusable (Daconta et al., 2003).

4.2 Main Components of Ontology

The previous definition and brief explanation is a solid base to understand constructing an ontology. In creation of an ontological model following components are used;

- Classes in the domain; represent concepts (e.g persons, items, etc).
- Instances; represent individuals in an ontology and these individuals are generated from existing classes.
- Relationships or Properties; represent the strong and meaning full relationship in between classes and/or instances.
- Functions; represent a special case of relation in which n^{th} element of the relation is unique for the preceding $n-1$ element.
- Constraints and rules; represent a logical and complete view of an ontology and its functionality.
- Axioms; represent model sentences.

4.3 Approaches for Ontology Construction

Two type of approaches are used to design an ontology, i.e. top down and bottom up. These are strategies of information processing and knowledge ordering, in order to create a hierarchy of knowledge domain. Top down designing method is used for knowledge decomposition, a class of knowledge is further decomposed in sub-classes, and bottom up is used for knowledge synthesis, two or more than two sub-classes are combined by a parent class.

4.3.1 Top Down Approach

Top down approach is a break down of information to gain insight into knowledge domain. For example a general concept of Food is sub-categorize (i.e. sub-classed) into fresh food items and dry food items. These sub-classes could be further defined to create a complete knowledge domain. In top down approach parent class is categorized into its child classes upto the leaves, to complete a hierarchical structure of a model.

4.3.2 Bottom Up Approach

Bottom up approach is the piecing together of knowledge to create a knowledge domain. Bottom up approach starts form the leaf level and then to the higher level. For example different types of pizza, i.e. Vegetable pizza and Margareta pizza lay under pizza category and pizza is a sub-class of fast food items. Integrating sub-classes from leaf level to higher class in a hierarchical structure, is a bottom up approach to design a model.

Another approach is also used in designing an ontology. It is the combination of both approaches i.e. top down and bottom up. In this approach a class is created and its possibility of extension is checked in both direction, to complete a hierarchical structure. For example, fast food items class could be a sub-class of food items class and fast food items class could be extended down ward to sub-classes such as pizza, burger, etc. This approach is a salient approach and used rarely in developing an ontological model(Natalya and Deborah, 2001).

4.4 Creating Ontology

Ontology is a representation of knowledge, in information sciences. The representation of knowledge is a set of concepts and relationships between concepts. An ontology is a “formal, explicit specification of a shared conceptualization” and provides a shared vocabulary, which is used to model a domain of knowledge (Thomas, 1993). Defining a domain of knowledge for license agreements, a root ontology (named as License Ontology) is created using bottom up approach. All required and common fields from documented license agreements are selected for designing License Ontology, using bottom up approach. These required and common fields became leaves of ontological structure and then the bottom up approach is followed to create the complete ontology. The ontology is categorized by arranging the leaf

classes, using a meaningful parental class. The definition of parent classes and categorization of leaves are according to selected agreements. Overview of ontological structure and detail explanation of ontology is covered as follows;

4.4.1 Overview of Ontology

The research work discussed in this thesis, is categorized in two parts, i.e. License Ontology and sub-license ontologies. The License Ontology is developed from common characteristics of different license agreements. The License Ontology is further extended to sub-license ontologies, in order to design the ontology as a meaningful license agreement. Each sub-license ontology describes a specific license agreement.

The resulting model is equipped with rules that depend on both user profile and license ontology. Functionality of entire process depends on search and comparison of user requirements using License Ontology. Query results obtained from sub-license ontologies are then sent back to a user for final approval. The whole scenario is depicted in Figure 4.2.

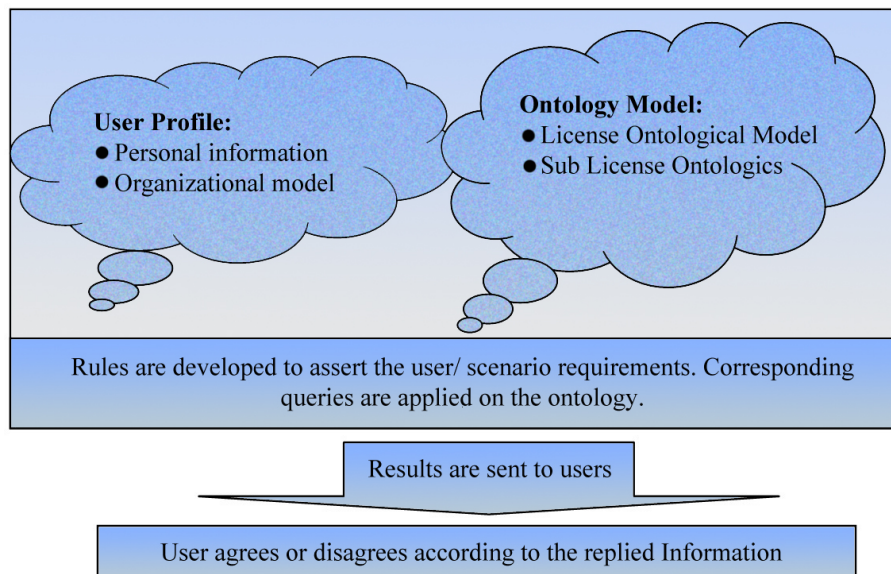


Figure 4.2: General model to describe the work flow.

4.4.2 Ontological Models

The research explained in this thesis, is based on an ontological structure, and its sub-ontological models. This ontological structure elaborates a document agreement. The construction of License Ontology is based on the common features, selected from the documented agreements. Detailed study of agreements is carried

out to analyze common features for creating the License Ontology. These common features helps in extending License Ontology to sub-license ontologies for specific agreements. The License Ontology is explained below:

4.5 License Ontology

As discussed in sections 4.4.1 and 4.4.2, common features are extracted from the selected agreements after studying these agreements. These common features help to construct a License Ontology. The License Ontology includes maximal common features of the agreements, which are feasible for, sub-license ontologies in describing a license agreement. The structure of License Ontology facilitates various companies to translate their documented agreement in License Ontology. Therefore, sub-license ontologies are created from selected agreements. These sub-license ontologies are explained in chapter 5. Figure 4.3 shows the License Ontology.

4.5.1 License Ontology Classes

License Ontology provides a base, for real world documented agreements and also facilitates information sharing between identical agreements of several organizations. License Ontology has two main classes i.e. “Agreement Items” and “Domain Concept”. “Agreement Items” consists of conceptual classes, and these classes are the core items of an agreement. The core items explains important issues related to an agreement, such as, conditions to apply penalty, copy rights, distribution, etc. “Domain Concept” consists of all those items, without which concepts of agreement are impossible to explain. Below, each class is explained in detail.

Agreement Items: In the context of License Ontology, license agreement is conceptualized by “Agreement Items” class. “Agreement Items” class is used to classify different terms and conditions of an agreement. Therefore, “Agreement Items” class is further categorized into three subclasses namely “Core Agreement”, “Non-Core Agreement” and “Breach”.

Core Agreement : “Core Agreement” class define terms, that are vital to be completed by a user. Whenever, user violates or fails to comply with any of the “Core Agreement” classes then the agreement can be revoked at any time by the provider. The “Core Agreement” class includes following sections of an agreement, that are explained below;

- **Copy Right Agreement** : “Copy Right Agreement” explains exception of rights that are tangible under certain circumstance, and these rights are provided to user by authors of products.

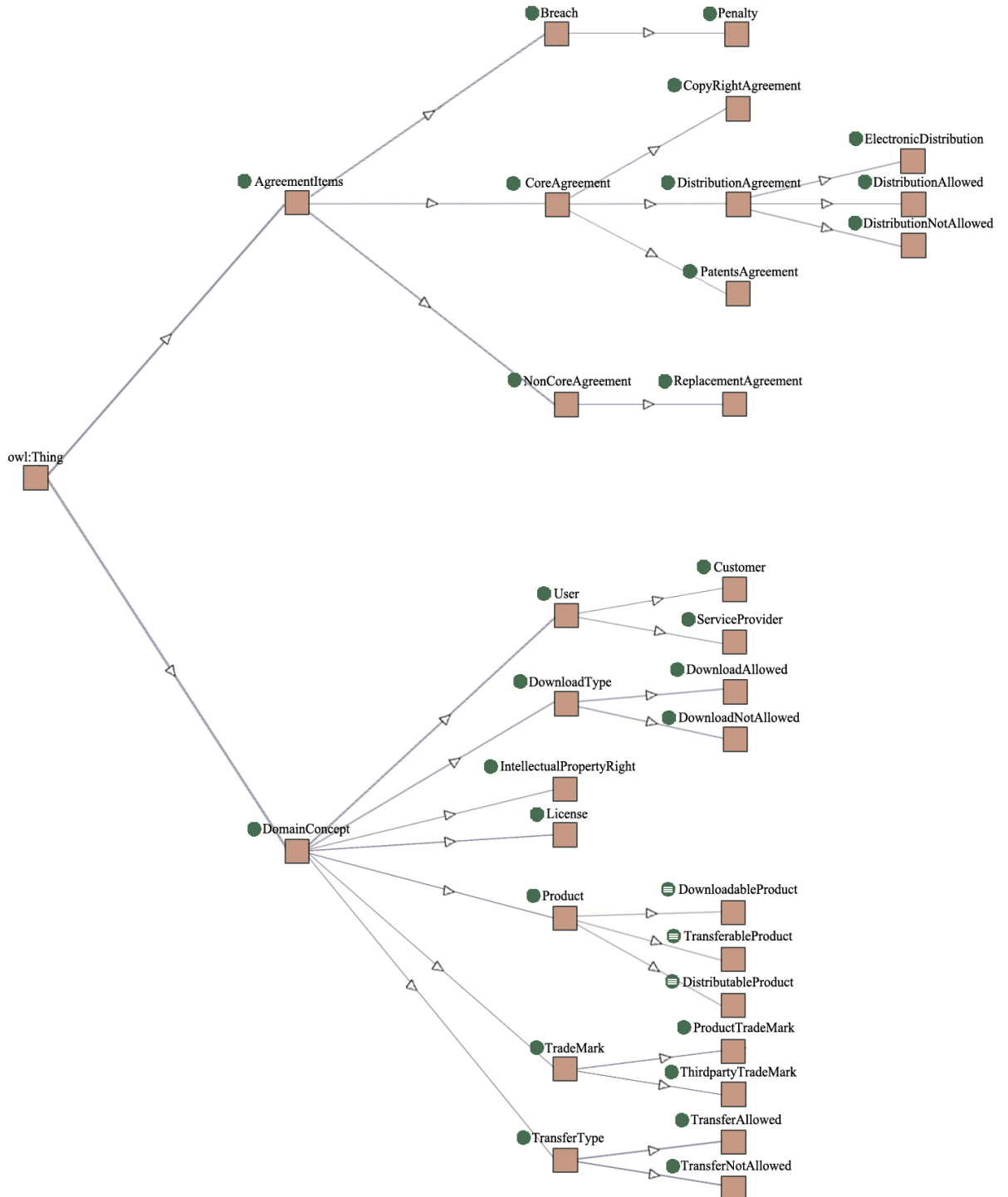


Figure 4.3: License Ontological Model.

- **Distribution Agreement Type**: “Distribution Agreement Type” class informs availability of product and its free distribution after purchasing. In-

stances defined at sub-classes of “Distribution Agreement Type” class, describes the availability of distributing products, according to terms and condition of agreement. “Distribution Agreement Type” class is further extended to “Electronic Distribution”, “Distribution Allowed” and “Distribution Not Allowed” classes as described below.

1. **Electronic Distribution:** “Electronic Distribution” class keeps information about the electronic distribution of product and its updates.
 2. **Distribution Allowed:** The information related to availability of distribution according to terms and condition of agreement, is kept in class “Distribution Allowed”.
 3. **Distribution Not Allowed:** “Distribution Not Allowed” class as the name describes, discusses the restriction of distribution according to terms and condition of an agreement.
- **Patents Agreement:** “Patents Agreement” class explains exclusive rights granted by a government to an author and/or user for manufacturing, use or sell a product within the defined rules. “Patents Agreement” could be further divided in sub-classes for explanation, which is not required according to requirements of this thesis.

Non Core Agreement: “Non Core Agreement” class defines those terms and conditions of an agreement, that are important for signing the agreement. The agreement continues without any penalties if any violation occurs in the sub-classes of “Non Core Agreement” class. In other words, “Non Core Agreement” class includes classes, which deal with the non-distinctive section of agreement. Only one sub-class exist so far, i.e. “Replacement Agreement” class, which is described below;

- **Replacement Agreement:** According to some agreements, the facility of replacing a product after damage is allowed while the others doesn’t replace. The related information about availability or unavailability of replacement of product according to an agreement is kept in this class.

Breach: “Breach” class defines different penalties according to an agreement. The class explains a procedure of providing compensation in case of violation, according to agreement. A sub-class “Penalty”, keeps such information of license agreement.

Domain Concept: A set of concepts used in License Ontology are kept in “Domain Concept” class. “Domain Concept” class describes those classes which are essential according to agreements. It is necessary to include following classes in “Domain Concept” class for designing and explaining the concepts of License Ontology.

Download Type: “Download Type” class informs a user about the availability of on-line downloading a product. “Download Type” class is further divided into two sub-classes described below:

- **Download Allowed:** “Download Allowed” class, as the name defines, informs about availability of download according to an agreement.
- **Download Not Allowed:** “Download Not Allowed” class informs about unavailability of download according to an agreement.

Intellectual Property Right (IPR): IPR explains granted privileges to a user according to an agreement.

License: “License” class identifies the specific product, who’s agreement is used to design the sub-license ontology, at leaf level. For example, in an ontology model, “License” class informs a user in providing validity of downloading of a product, according to an agreement. And “License” class also explains its importance at the place where user is given permission to distribute, download or transfer the product according to an agreement.

Product: “Product” class provides information about products, for example name of product. And according to each product’s agreement, the product is explained in License Ontology, at leaf level. It is further divided into “Distribution Product”, “Download Product” and “Transfer Product” class; as described below.

- **Distribution Product:** “Distribution Product” class describes the availability of distributing each product according to an agreement.
- **Download Product:** “Download Product” class describes the availability of downloading each product according to license agreement.
- **Transfer Product:** Same as above two classes, “Transfer Product” class provides information about transferring each product after purchasing.

Trade Mark: “Trade Mark” class describes information about trade mark according to an agreement of an organization or a team of developers. “Trade Mark” class has two subclasses namely product’s trade mark and third party trade mark.

- **Product Trade Mark:** “Product Trade Mark” is a trade mark of a product designed by a parent organization.
- **Third party Trade Mark:** “Third Party Trade Mark” class informs the availability of third party trade mark according to an agreement. A user could use his trade mark, according to an agreement.

Transfer Type: “Transfer Type” class provides information about the process of legal transfer of agreement after purchasing a product.

- **Transfer Allowed:** “Transfer Allowed” class informs the availability of transfer rights, product and modification (if it is a software product), according to license agreement.
- **Transfer Not Allowed:** “Transfer Not Allowed” class informs the unavailability of transferring any type of information, product or any thing related to product, according to an agreement.

User: “User” class is sub divided into “Service Provider” and “Customer” class. As the name describes, “User” class informs the responsibilities of a user, according to an agreement.

- **Service Provider:** “Service Provider” class provides information about a user, who maintains software application (i.e. Web site or any other public use software applications) and/or about a seller who sells his products. “Service Provider” class defines only those users, who are related with any type of managing or maintaining product according to an agreement. “Service Provider” class provides facility in ontology to deal with the developer’s or seller’s related issues in an agreement. For example, according to Mozilla license agreement, after modification in source code developer has to provide “Sub License Agreement” (i.e. this class is defined in Software License Ontology), which explains terms and conditions of a modified product. Therefore, the “Service Provider” class in the ontology of Mozilla license agreement, is connected with “Sub License Agreement” class using a defined property. This explains a complete concept that is used in the agreement.

- **Customer:** “Customer” class provides information about customer’s profile and also about issues related to purchasing or transferring an agreement. “Customer” class provides facility in ontology to deal with the issues in agreement related to customer.

4.5.2 OWL Properties used in License Ontology

OWL properties represent relationships between two or more individuals. Other than the two main types of OWL properties, object properties are used in constructing License Ontology. Object properties links an individual to another individual and is used in License Ontology to explain main concepts of an agreement. The defined properties in License Ontology describes all common features selected from various agreements. The object properties used in License Ontology are described below;

Copy right agreement and intellectual property rights *has rights for* a user. The property *has rights for* explains that copy right and intellectual property rights of each selected agreements have rights for service provider and customer. Another property i.e. *is replaced as* is used as; Product *is replaced as* replacement agreement, when replacement agreement allows to replace the product. The replacement is allowed with some limitations, as defined according to agreement. Therefore, product is replicable otherwise product after purchase is not replaceable. Same as another property i.e., product *is replaced by* service provider. List of other used OWL properties are given below;

1. Patents Agreement *has royalty free* Transfer Type
2. License *has validity of* Download Type, Transfer Type, Replacement Agreement, Trade Mark and Distributed Agreement
3. Product *has documented* License
4. License *has items* Agreement items
5. Agreement items *is unliable to* Penalty
6. Product trade mark *is unmodifiable by* User
7. Download Type and Transfer Type *is available as* Agreement items
8. User *is subject to* Penalty

9. Product and User *is allowed to* Download Type, Transfer Type and Distribution agreement
10. Service provider *is allowed for* Third party trade mark
11. Copy right agreement *is same as* Intellectual property right
12. Intellectual property right *is similar as* Copy right agreement
13. License *is for each* Product

The listed properties are self explanatory; individuals of classes are defined in next level of ontology models i.e. sub license ontologies. The sub-license ontology extends License Ontology for a specific category of agreements, for example software license agreements and online shopping license agreements as explained in chapter 5.

In this chapter a complete structure of the License Ontology is discussed. The License Ontology is a centralized ontological model, and is the foundation for all license agreements. The License Ontology is extended further to various categories of agreements, as discussed and explained in chapter 5.

Chapter 5

Extending License Ontology

Software developers prefer to reuse of existing objects as compare to develop something new from scratch, for developing or extending ontologies, schema or any software application. It is true that, existing objects though are not always a perfect fit for users requirements and also customizing the existing objects according to needs often depends on the designing of original object. It requires a considerable study about the existing object before using it in developing application. Other than these comments, existing object minimizes the time to create something equal to the object before using in developing application. Existing objects not only performs the required functionality but can be molded for other functionality, as per requirements of designed objects.

Ontology is defined as a formal specification in designing a model, using a vocabulary of concepts and axioms (Chandrasekaran et al., 1999), and allows to express the complex relationships within a domain area (Wouters, 2005). Ontology establishes relationship between concepts for understand the concepts, therefore, properties are used as relationships. Ontologies are designed to maintain expressiveness of the domain knowledge rather than for ease of use. Therefore, developers are looking for more efficient methods for reusing, extraction and extension of ontology (Flahive et al., 2007). Three main reasons for extending an ontology are, 1) fast access to most relevant information, 2) assist in finding new knowledge from within large domain ontologies and 3) assisting in ontology creation.

Ontology creation is a difficult task and it is relatively easy to create a small ontology to suit the purpose of domain knowledge. Consequently, sub-ontology are feasible to be extracted form a parent ontology to elaborate a section of domain knowledge (Uchibayashi et al., 2009). Extracting sub-ontology has to have following two features 1) consistency and 2) completeness. Extracting sub-ontologies involves selecting a number of elements form a large ontology (i.e. parent ontology) and

concise into a smaller one, for more suitable ontology according to requirements of application (Flahive et al., 2009). Ontology extends from time to time, and this extension depends on growth of domain or depends to have a complete ontology.

Sub-ontology concept is explained by proposing four methods i.e. extend, add, merge and replace. Extend method basically allows a user to create an ontology by using the existing ontology. The extend method depends on new features of sub-ontology engineering, which includes concepts, properties, relationships and mapping. Add method is completely different from extend method, in this method sub-ontology is added in parent ontology, as a new complete ontology. According to merge method, two or more sub-ontology are merged together to form a new ontology. Replace method, as the name defines, replaces some part of another ontology to form a new sub-ontology (Uchibayashi et al., 2009). The License Ontology is extended using extend method of sub-ontology concept. These sub-license ontologies are categorized into Online Shopping License Ontology and Software License Ontology. Each of these sub-ontologies are discussed in sections 5.1 and 5.2, in detail.

5.1 Online Shopping License Ontology

Constructing Online Shopping License Ontology is based on the study of two different agreements. These include amazon license agreement and eBay license agreement, both are famous for online shopping centers. These agreements have many things in common with some differences. Online Shopping License Ontology is an extension of License Ontology, as shown in figure 4.3. The Online Shopping License Ontology extends all the features of License Ontology. A class named “Reimbursement” is added to Online Shopping License Ontology. The class is required to provide a concept of reimbursement method for customer, after purchasing product. Extra properties are added to define the relationship of reimbursement class with customer and seller classes in the Online Shopping License Ontology. The properties are defined below;

1. User *is granted for* Reimbursement
2. Reimbursement *is illustrated by* Seller

The first property has a “User” class, as discussed in section 4.5.1, consists of “Customer” and “Service Provider” classes. These two sub-classes of “User” class are granted for reimbursement in case of damages. The second property explains that “Service Provider” should define reimbursement method before selling any product.

Online Shopping License Ontology, as shown in figure 5.1, has defined individuals. These defined individuals are connected with each other, according to an agreement, to define complete meaning of License Ontology. The sub-license ontology of Online Shopping License Ontology are discussed in sections 5.1.1 and 5.1.2;

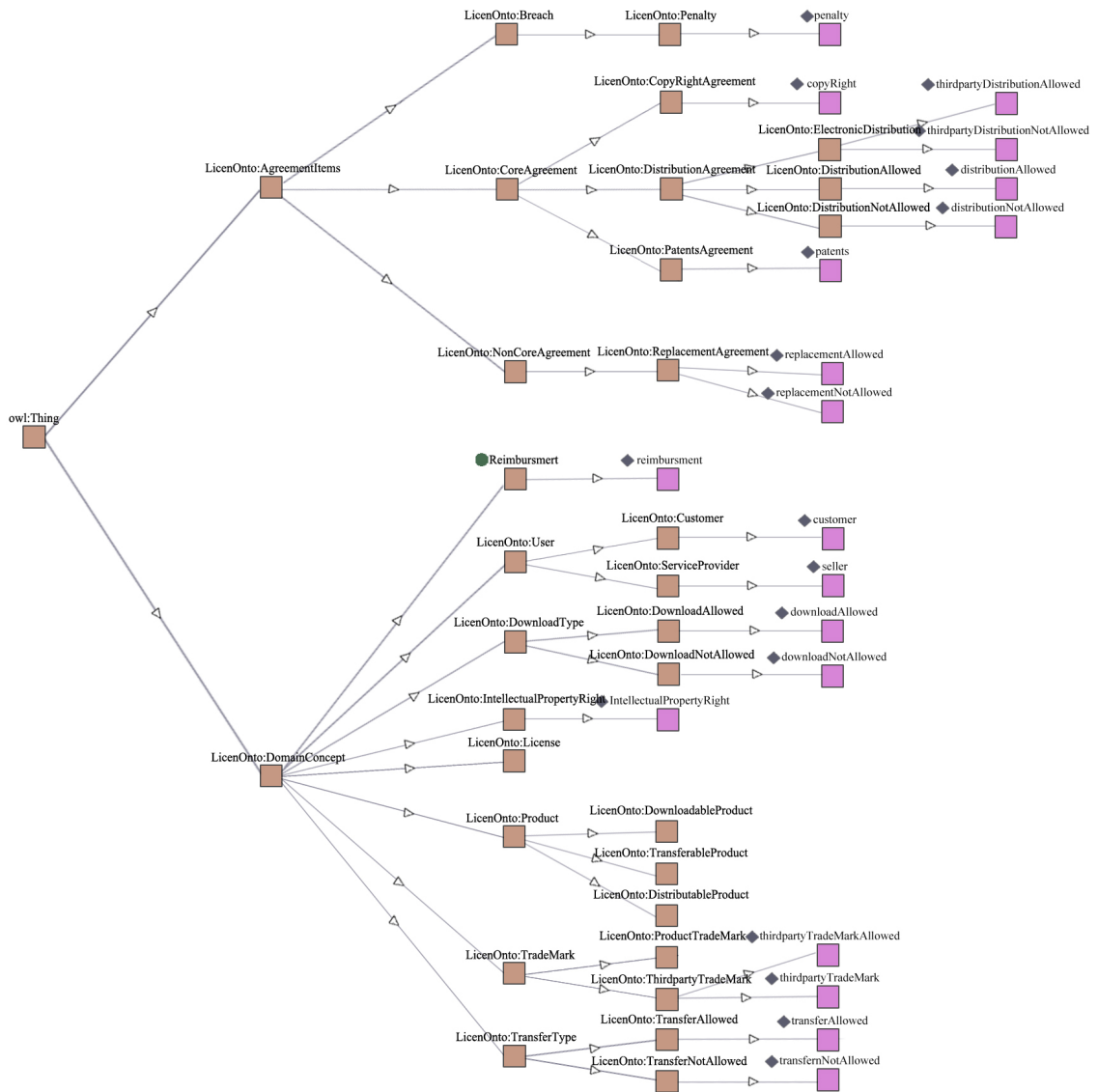


Figure 5.1: Online Shopping License Ontology.

5.1.1 Amazon License Ontology

Amazon License Ontology is a extension of Online Shopping License Ontology. Amazon License Ontology adds extra individuals to complete the meaning and explanation of amazon license agreement.

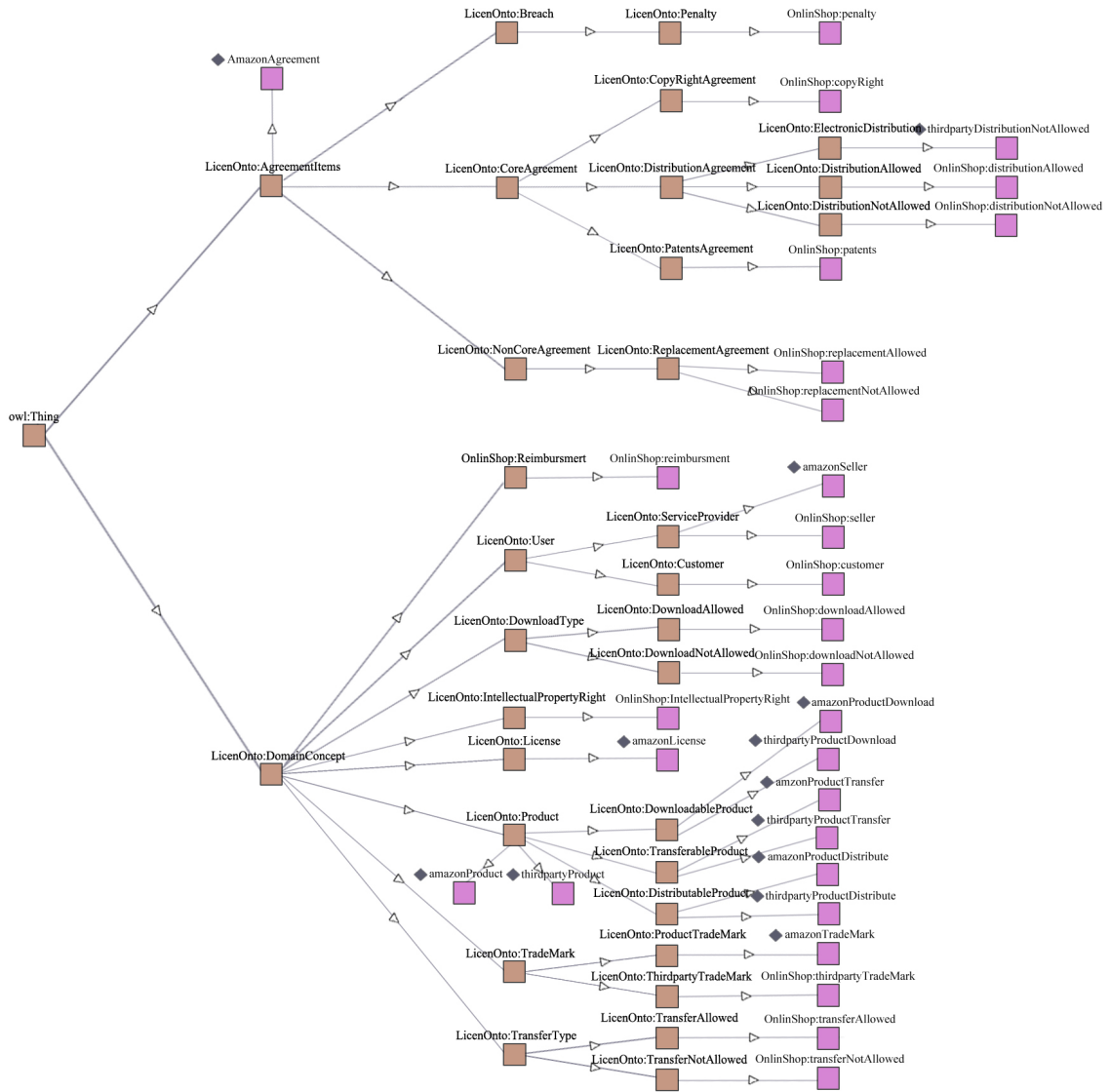


Figure 5.2: Amazon License Ontology.

These individuals include amazon License, amazon Agreement, amazon Seller, amazon Trade Mark, amazon Product, amazon Product Download, amazon Product Transfer, amazon Product Distribute, third-party Product, third-party Product Download, third-party Product Transfer and third-party Product Distribute, as shown figure 5.2. The given twelve individuals and the imported individuals from Online Shopping License Ontology are connected with each other, to completely define the ontology according to amazon license agreement.

5.1.2 eBay License Ontology

eBay License Ontology is also an extension of Online Shopping License Ontology, as like Amazon License Ontology. Some of extra individuals are added to the eBay License Ontology.

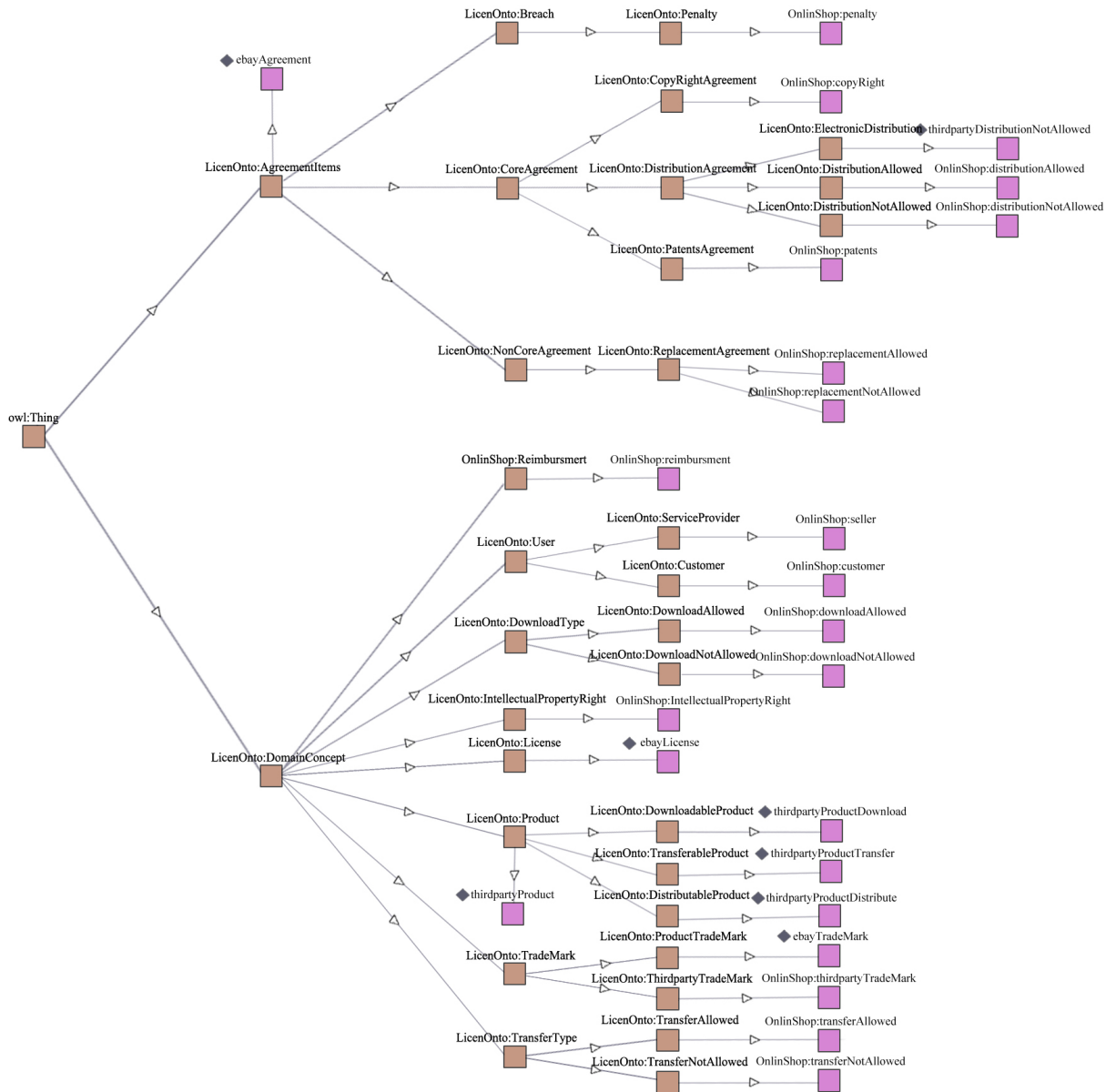


Figure 5.3: eBay License Ontology.

These individuals are shown in figure 5.3. The seven new individuals and the imported individuals from Online Shopping License Ontology are connected with each other, to complete the ontology according to eBay agreement.

As known to everyone amazon sell their own products, as per availability at stock, while eBay provides facility for sellers to sell their items. Therefore, both figures i.e. 5.2 and 5.3 shows a great difference in selling products. Amazon uses the amazon products and third party products for selling, while eBay has only third-party products. After defining the required individuals in each ontology, as discussed in sections 5.1.1 and 5.1.2, the individuals are connected with each other by relating the concepts of ontology. The relations are defined according to the license agreements and helps in explaining the meaning of concepts.

5.2 Software License Ontology

Construction of Software License Ontology is based on the study of four different license agreements. These agreements includes Mozilla¹, Apache², Adobe³ and Creative Commons⁴. Mozilla and Apache are open source code license agreement while Adobe is a commercial based license agreement. Creative Commons agreements provides a standard method to copyright user's Web sites or Web pages, as discussed in section 2.1.3. The different types of agreements are used to construct a Software License Ontology, therefore the Software License Ontology is capable of wrapping all software agreement.

Software License Ontology is an extension of License Ontology. Software License Ontology is defined with the addition of required concepts and properties according to license agreements, as shown in figure 5.4. Software License Ontology adds 6 new classes and 15 properties to complete the definition of ontology. These new classes and properties are discussed as follows;

- **Sub License Agreement:** “Sub License Agreement” class explains the modifier's rules after modification in original source code. Through “Sub License Agreement” class developer is allowed to include required information, which helps to use and/or to restrict modified source code. According to an agreement, developer can also provide information about original license agreement. “Sub License Agreement” class is a sub-class of “Core Agreement” class, i.e. imported from License Ontology.
- **Backup Copy:** “Backup Copy” informs a user about the availability of backup facility after purchasing, according to license agreement. This class is a sub-class of “Domain Concept” class, and the “Domain Concept” is imported from License Ontology.
- **Developing Platform:** “Developing Platform” class informs about tools or application, which are used in modification of source code and/or development of software product. This class is helpful for the developers who modify source code according to their needs. “Developing Platform” class lays under “Domain Concept” class.

¹<http://support.mozilla.com/en-US/kb/Terms+of+Service>

²<http://www.apache.org/licenses/>

³http://www.adobe.com/products/eulas/pdfs/Gen_WWCombined-combined-20080623_1026.pdf

⁴<http://creativecommons.org/about/licenses/>

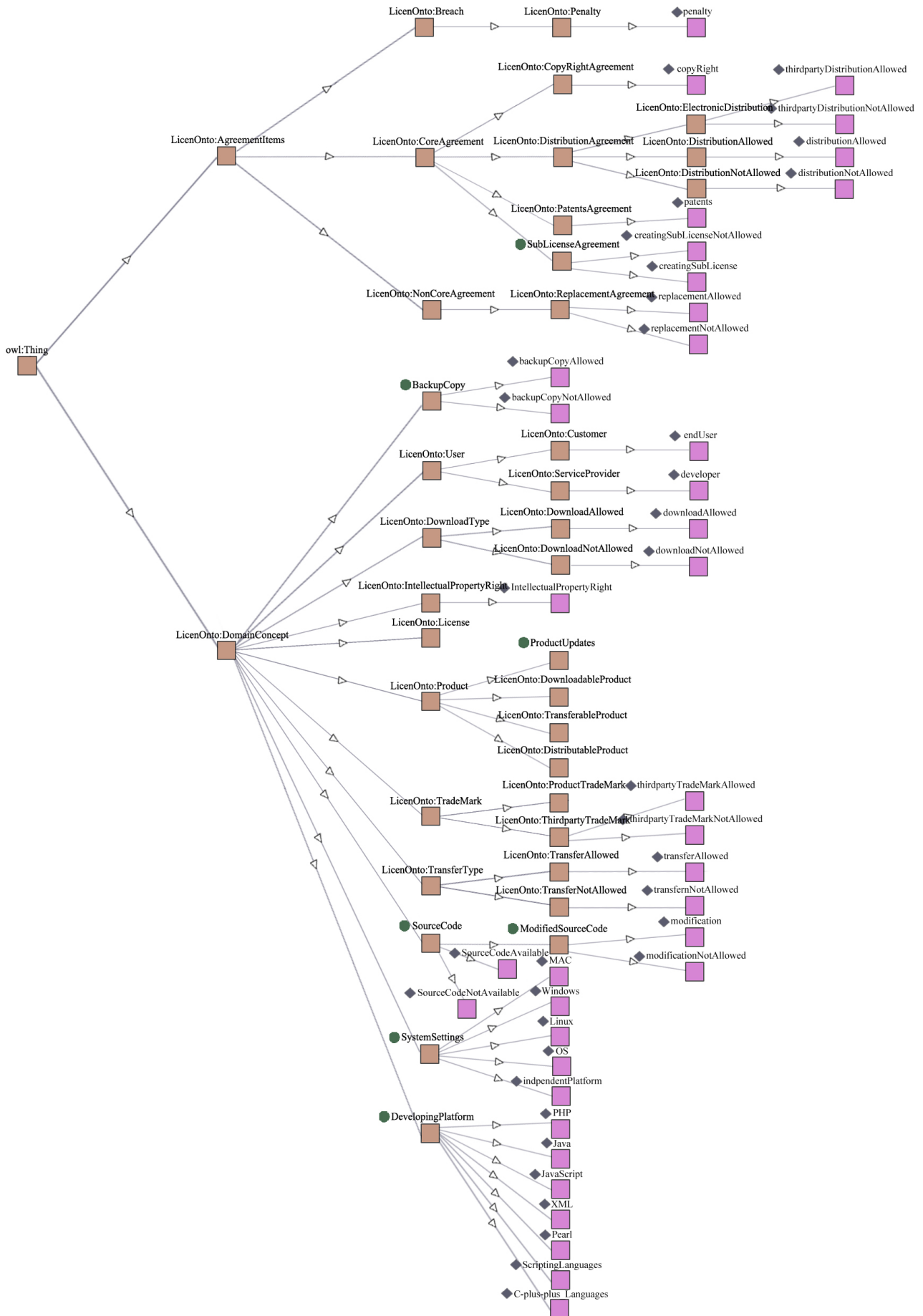


Figure 5.4: Software License Ontology.

- **Product Update:** “Product Update” class provides information about distribution, download and transfer of software updates according to an agreement. “Product Update” is a sub-class of “LicenOnto: Product” class (of License Ontology).
- **Source Code:** “Source Code” class informs about the availability of a source code according to an agreement. Some agreements for example Apache, allow developer to perform modification in provided source code according to requirements. But developer cannot distribute modified source code according to the agreement. “Source Code” class is a sub-class of “Domain Concept” class and has a child class named “Modified Source Code” class.
- **Modified Source Code:** Describes availability of modifying source code and also explains different actions that could be taken after modification of source code. For example an agreement allows an end user for modification in source code, then the end user might use his trade mark with modified source code and he may allow others to distribute his work according to agreement.
- **System Settings:** “System Settings” class provides information about operating system on which the specific product is executable . Certain software applications are platform independent, while others require defined platform for execution, for example executing an application using Window’s, Linux, MAC and OS.

The above defined classes (i.e. concepts) and inherited classes from License Ontology are related with each other through the define properties. Some of required properties are also added to the Software License Ontology. These newly added properties are listed as;

1. Modified Source Code *is allowed on* LicenOnto:Electronic Distribution
2. Modified Source Code *is distributed under terms of* LicenOnto:License
3. Sub License Agreement *has information of* LicenOnto:Product, LicenOnto:Patents Agreement, LicenOnto:Distribution Agreement and Modified Source Code
4. Sub License Agreement and Modified Source Code *is provided by* LicenOnto:Service Provider
5. LicenOnto:Product *is executed in* System Settings
6. LicenOnto:License *has permission of* Backup Copy, Sub License Agreement and Modified Source Code

7. LicenOnto:License *is unchangeable by* Sub License Agreement
8. Modified Source Code *is derived from* Source Code
9. LicenOnto:Service Provider *is unable to restrict* Modified Source Code
10. LicenOnto:Customer *is granted for* Backup Copy
11. LicenOnto:Service Provider *is allowed for* Third party Trade Mark
12. LicenOnto:Service Provider *is in possession of* Modified Source Code, LicenOnto:Patents Agreement, Sub License Agreement and Third party Trade Mark
13. LicenOnto:Service Provider *has terms described in* Sub License Agreement
14. LicenOnto:Product *is developed using* Developing Platform
15. Sub License Agreement and Modified Source Code *is liable to* LicenOnto:Penalty

In above defined properties the word “LicenOnto:” used with class name explains that the particular class is an extension from License Ontology. The License Ontology is extended to sub-license ontology i.e. Software License Ontology. The sub-ontologies of Software License Ontology contains instances and these instances are connected to each other according to a license agreement. Instances explains the existence of a value in a class, for example if source code is available to an end user for modification then class “Modified Source Code” will have an instance “modification” and “Modified Source Code” class is further connected with the class “LicenOnto: License”. This triple is defined through a property named as “*is distributed under terms of*”. The availability of property and instances explains that modification of source code is allowed according to an agreement. Availability of instances facilitates user, by providing SPARQL results after applying queries on an ontology.

Four of software license agreements are selected, to fulfill the requirements of a user, using License Ontology. In following each ontology is discussed briefly;

5.2.1 Mozilla & Apache License Ontology

Mozilla and Apache agreement support open source initiatives. Because of open source distribution, Mozilla and Apache freely distribute their source code for modification. They have many things in common, for example, both allow replacement of damaged products, on-line download of products, source code for modification and

provide environment independent executable applications. According to Mozilla license agreement, it states that a developer (i.e. who modifies source code of software product) should provide his own additional document (i.e. class “SoftOnto: Sub License Agreement” as shown in figure 5.5) for defining additional rights. The statement of Mozilla agreement is projected by using a class “SoftOnto: Sub License Agreement”, which connects with “SoftOnto: Developer” class using property “*SoftOnto: has terms described in*”. The triple (i.e. “SoftOnto: Developer *SoftOnto: has terms described in* SoftOnto: Sub License Agreement”) according to Mozilla license agreement, explains complete scenario in ontological model. While Apache license agreement allows a user for modification according to the user requirements, but the modified source code is neither allowed to distribute nor allows to create any type of text file (i.e using “SoftOnto: Sub License Agreement” class) for the modified source code.

Mozilla also allows for a third party trade mark, but Apache license agreement doesn't provide any information about third party trade mark. A developer can modify an Apache product according to requirements but Apache is seldom on behalf of re-modification of someone else modified source code. Apache license agreement does not has complexities as compared to Mozilla license agreement, ontology of both license agreements are shown in figures 5.5 and 5.6.

5.2.2 Adobe License Ontology

Adobe products are commercial products and are not allowed for free distribution. Adobe license agreement facilitates user to transfer Adobe products under the terms and conditions. Some of required features of the Adobe license agreement are selected and used to construct Adobe License Ontology. These required features benefits the License Ontology (i.e. root ontology) in explaining the capability to construct ontologies for open source code license agreements and commercial software license agreements. In this section the construction and usage of Adobe License Ontology is explained in detail.

According to the Adobe license agreement, modification in any of the Adobe product is not allowed and even distribution of the product through any third party distributor is stated illegal. These described statements of Adobe agreement are shown in Adobe License Ontology in the following form. A class “SoftOnto: Modified Source Code” has an instance “SoftOnto: modificationNotAllowed”, which is further connected with an instance “SoftOnto: third party Distribution Not Allowed”, of a class “LicenOnto: Electronic Distribution”. The connection between these two instances is made possible through a property “*SoftOnto: is allowed on*”. The word

“SoftOnto:” explains the inheritance of properties and classes from Software License Ontology.

It is easy to retrieve information by querying Adobe License Ontology for distribution of software products, as compared to search the documented Adobe agreement. Figure 5.7 shows a complete picture of Adobe License Ontology.

5.2.3 Creative Commons License Ontology

Creative Commons Agreement (CC) defines four keywords, and combination of these four keywords helps a user to construct an agreement, CC is discussed in section 2.1.3. Therefore, eye-OS Web Application⁵ is selected, which uses maximum combination of different categories defined by CC licensing agreement. In order to create a CC License Ontology, initially License Ontology is imported into CC License Ontology and then the required class, i.e. “Commercialization Agreement”, is added, as shown in figure 5.8. By the help of this class CC licensing agreement is translated into CC License Ontology. And by adding the class “Commercialization Agreement”, the CC License Ontology is being able to extend further and create a specific ontology for any organization using CC licensing agreement.

eyeOS Web License Ontology

eyeOS Web Application agreement uses CC licensing in defining its restriction. The eyeOS Web Application is using a combination of “by-nc-nd” keywords from CC licensing agreement. The combination of keywords explains that eyeOS Web Application allows user to share resources but with condition to pay credit to author of the application. The application is not for commercialization and even not for distribution, according to the agreement. The eyeOS Web License Ontology use classes i.e. “LicenOnto: Agreement Items”, “LicenOnto: Distribution Not Agreement” class, “SoftOnto: Modified Source Code” class and “CC: Commercialization Agreement” class, and uses instances i.e “eyeOS Web Agreement”, “SoftOnto: distribution Not Allowed”, “SoftOnto: modification Not Allowed” and “CC: commercialization Not Allowed” for the classes respectively. The meaning of “LicenOnto:” and “SoftOnto:” is discussed in previous sections i.e 5.2 and 5.2.2 respectively. In the eyeOS Web License Ontology “CC:” identifies that such class and instance is inherited from CC License Ontology. The instances defined according to CC licensing agreement, and these instances completes the meanings of triples used for CC License Ontology. eyeOS Web License Ontology, as shown in figure 5.9, illuminates an idea that

⁵<http://www.scribd.com/doc/3176583/eyeOS-User-Manual>

any type of agreement could be adopted, using License Ontology with minimum modifications.

In this chapter, the construction of Online Shopping License Ontology and Software License Ontology are discussed. With these extended ontologies, the sub-license ontologies of Online Shopping License Ontology and Software License Ontology are also elaborated. The sub-license ontologies not only explains the significance of License Ontology but also defines its extensibility. In chapter 6, different rules are discussed which helped to create the License Ontology and sub-license ontologies.

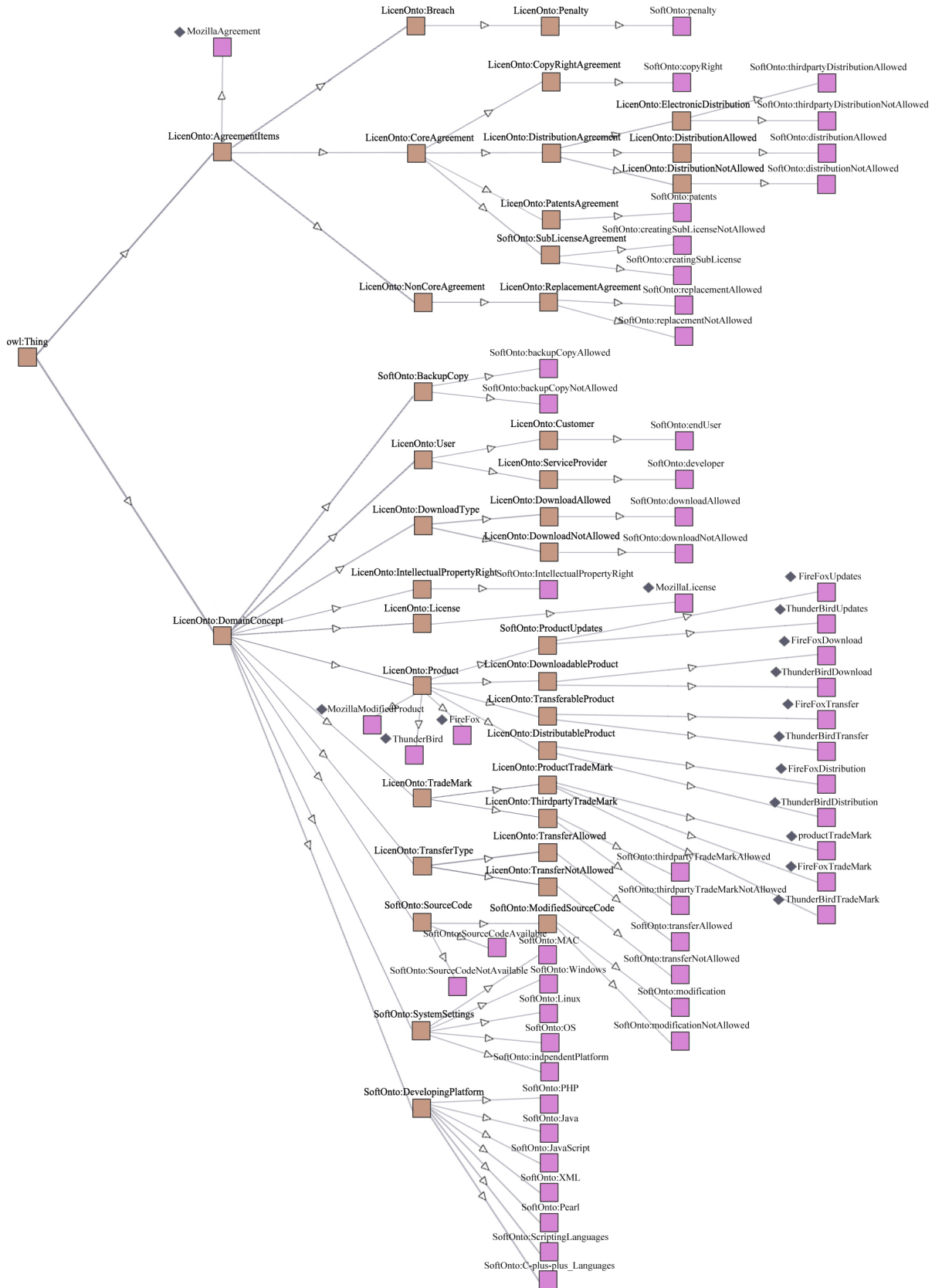


Figure 5.5: Mozilla License Ontology.

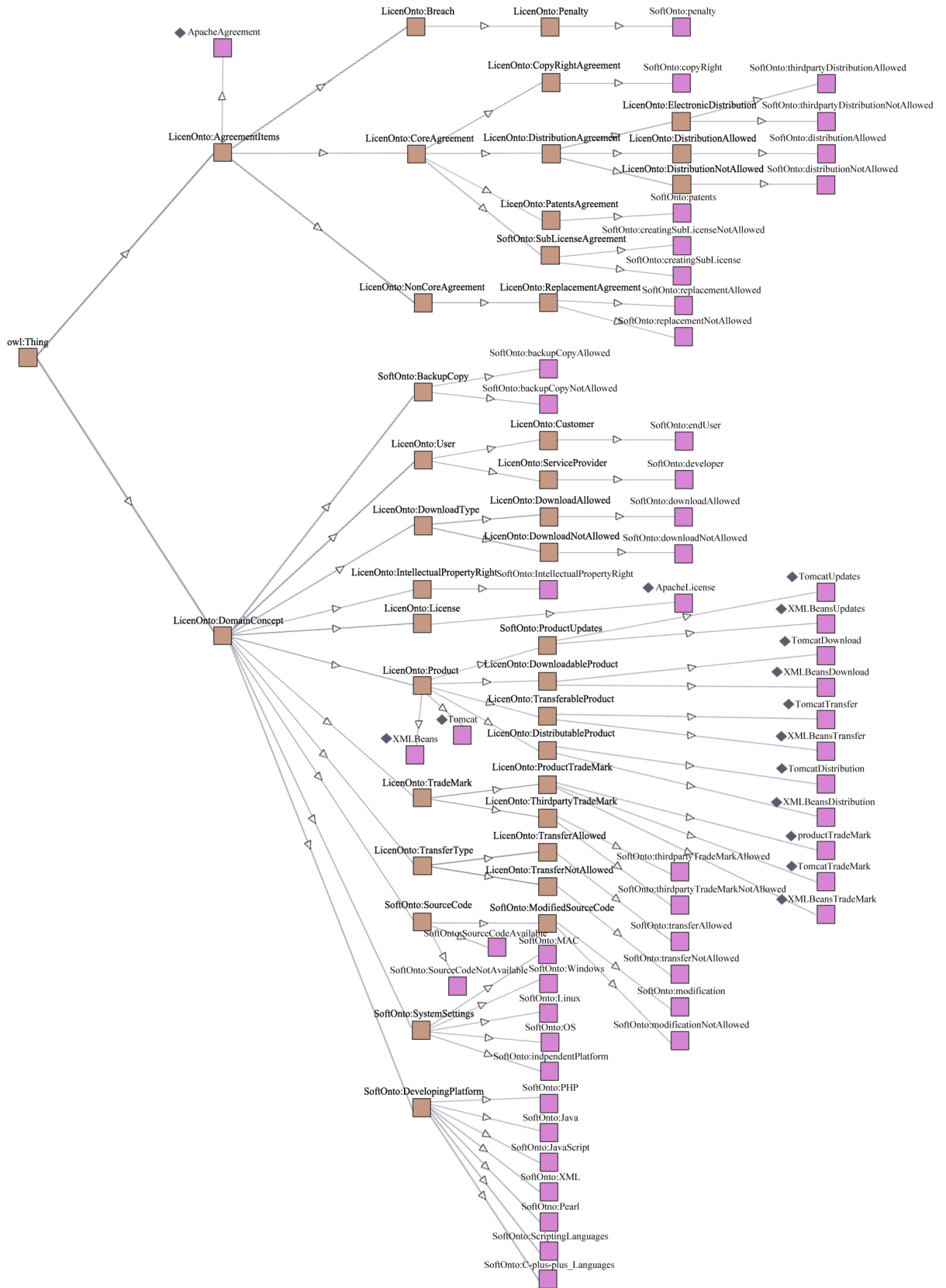


Figure 5.6: Apache License Ontology.

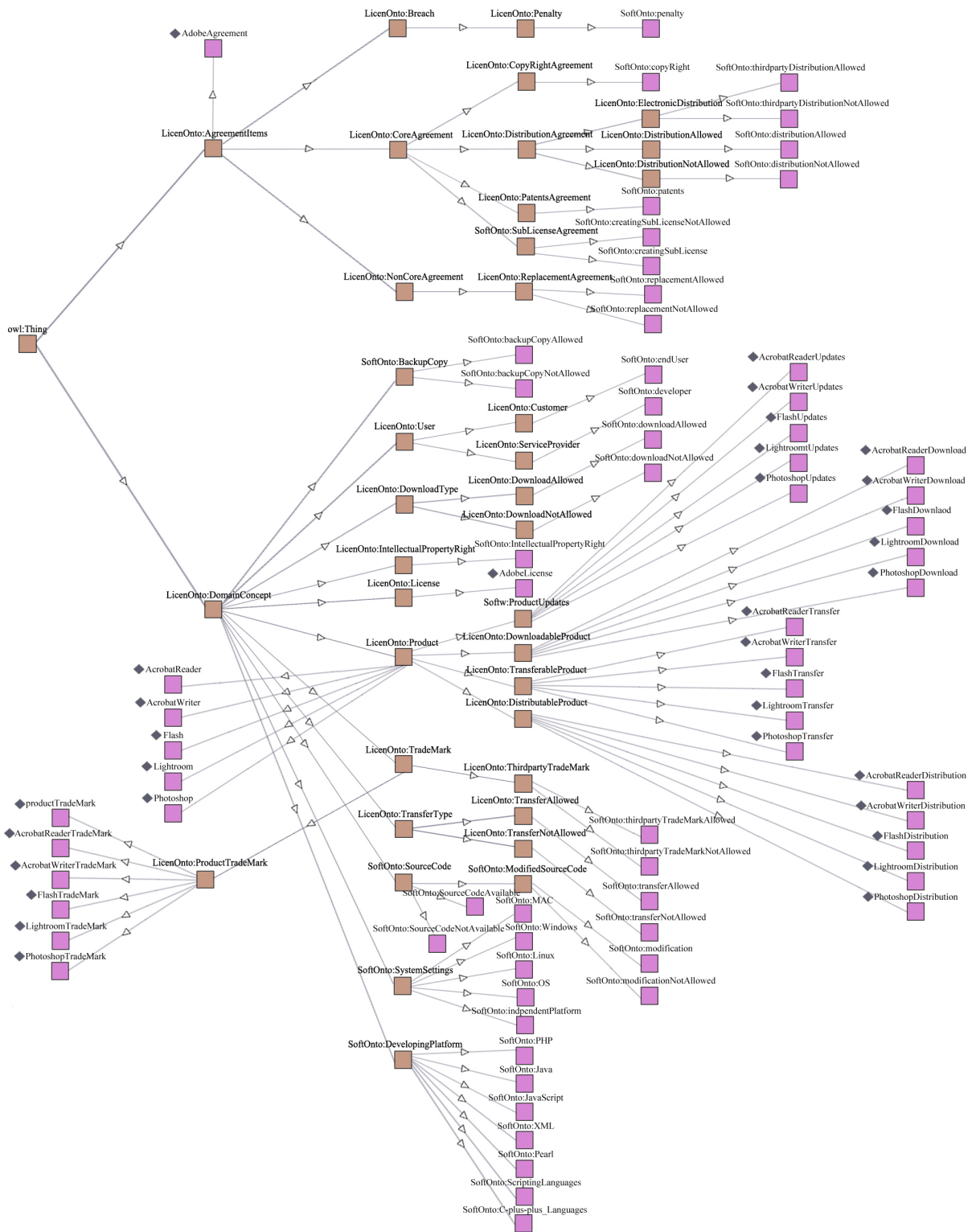


Figure 5.7: Adobe License Ontology

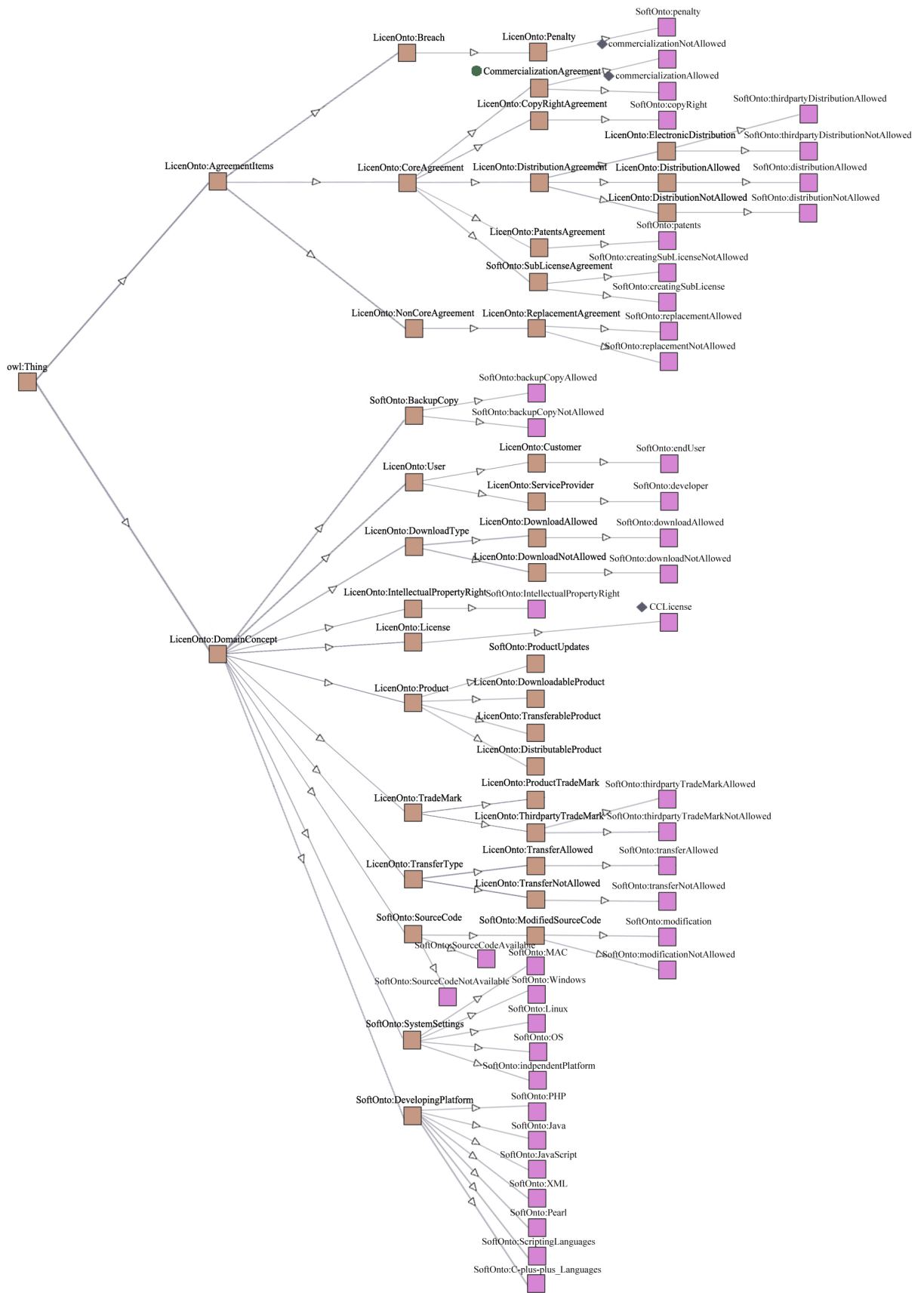


Figure 5.8: Creative Commons License Ontology.1

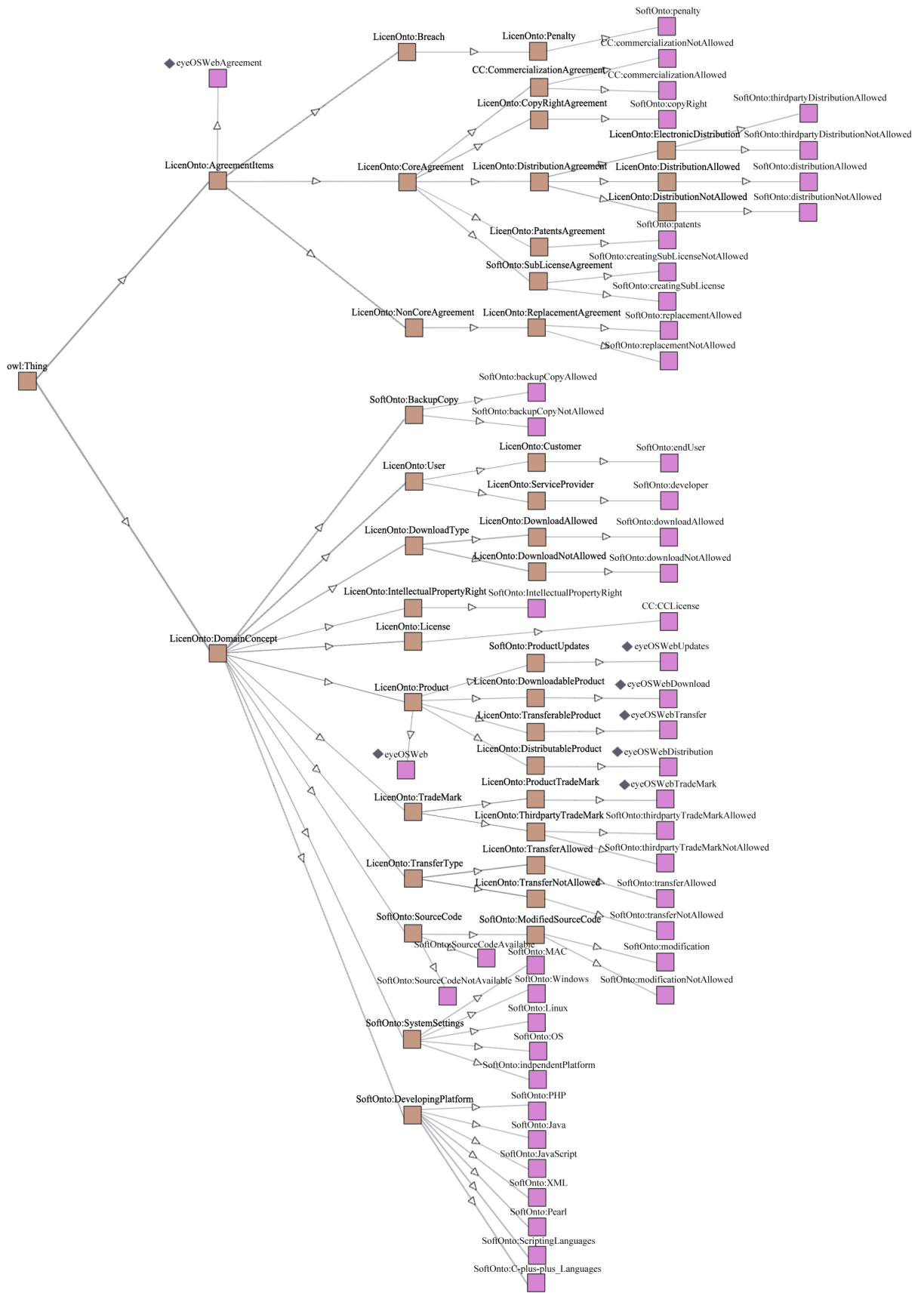


Figure 5.9: eyeOS Web License Ontology.

Chapter 6

Reasoning Ontologies using Rules

Intelligent agents have been a hot topics in computer science research for last 50 years. As time passed, intelligent agents have been integrated into other tools, such as search engines and content negotiation applications, which are used in Internet. These application agents are adaptable on different platforms, coordinating in transferring information and are mobile on the Internet. The application agents take input, make inferences on input and generate output. The application agents use set of rules i.e. logical expressions that filter information on the basis of user defined requirements. Filtering is a process of excluding un-required information from a domain of knowledge. Filters are based on a set of rules and set of facts, which defines a logical expression to acquire results (Johan, 2001). Network-based use case is a best example to explain rules and filters. In network communications, filters are placed to control and block un-required traffic. The network filters are based on set of rules, which make decision about required and un-required information (i.e. Internet). These set of rules are named as policies in network.

Rules elaborate sequence of RDF actions that explain behavior of a system (Johan, 2001). There are different ways to create a rule or set of rules, some of them are used by specific applications such as *Apple, Pellet, Racer and Jena Rule Engine*. Rule sets or rule language tools perform reasoning's on a domain of knowledge. And reasoning provides inferences of a data. Reasoning on Web data is possible by Web Rule Language.

This chapter describes Web Rule Language and its application. Further more, rules are discussed in co-ordinance with license agreements and license ontologies.

6.1 Web Rule Language

Web Rule Language (WRL) is a rule based ontology language, and is used for Semantic Web. WRL is derived from Web Service Modeling Language and is a component of ontology. Ontological meta model of WRL consists of concepts, relations between concepts, instances and axioms. Figure 6.1¹ shows the location of WRL in Semantic Web Cake.

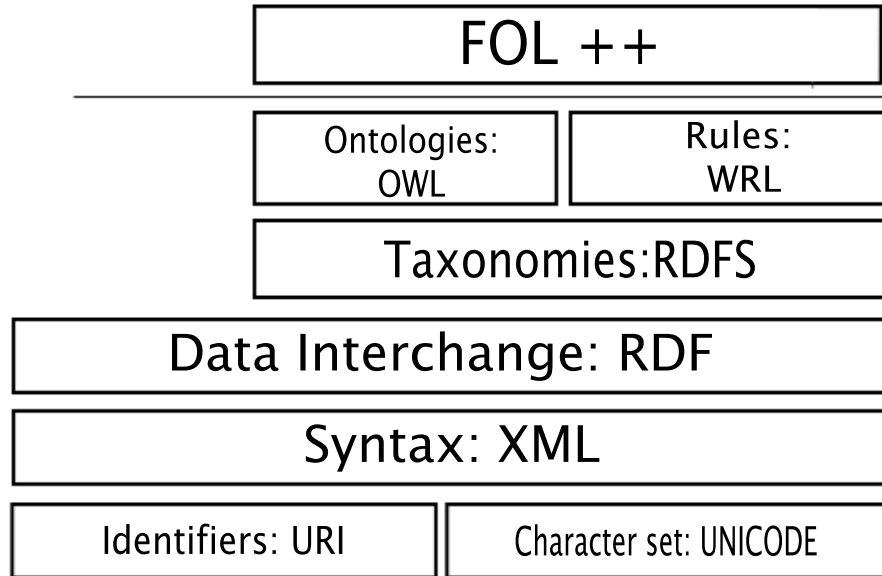


Figure 6.1: WRL positioned in Semantic Web Cake.

Ontology vocabulary layer refers to two sections of Semantic Web Cake, which includes OWL and WRL. In the figure 6.1, OWL is a Description Logic based ontology language and WRL is a rule based ontology language. Common super set of OWL and WRL is First Order Logic (FOL ++) and its extensions. FOL ++ is a non standardized expressive language for Rule based languages (Angele et al., 2005). WRL semantics provides mapping between conceptual syntax and logical expression of WRL. Mapping of WRL semantics provides complete meaning to rules in logical languages.

6.2 Semantic Web Rule Language

Semantic Web Rule Language (SWRL) is based on combination of OWL with Rule Markup languages. SWRL rules are divided into head (consequent) and body (antecedent). The division has a logical meaning. i.e. the condition satisfied in antecedent of a rule, must also be satisfied in consequent of the rule. Both antecedent

¹<http://www.w3.org/submission/WRL/>

and consequent consists of zero or more atoms. Empty atom antecedent conditions are always considered true, while multiple atoms are treated with conjunction of the consequent atoms. Multiple atoms antecedent could be translated in multiple rule statements. Atoms in rules are descriptions of OWL, i.e. properties, variables, individuals or data values of OWL. An OWL ontology consists of a sequence of axioms and facts. Axioms are of two main kinds e.g. sub-Class axioms and equivalent Class axioms, and both could be extended to rules axioms. A rule axiom is assigned a URI reference, which is an identity of a rule. Empty axioms in rules are provided an unconditional facts and are used without the rule construct. Facts contain conditions and satisfy a rule consequent to hold true condition of antecedent. The semantic rule satisfies an ontology if and only if the rule satisfies every axioms and facts in an ontology (Horrocks et al., 2004).

The idea of SWRL is to bind OWL variables to a domain of knowledge and rule satisfies interpretation of every binding. The interpretation of rules are satisfied whenever rules satisfies every axiom and fact in an ontology.

WRL and SWRL rules architecture are not used to develop the License Ontology and its sub-license ontologies. But some rules are defined, on the basis of user requirements, to acquire required results. Using Protege, classes and relationship between the classes are defined. The rules are designed to perform reasoning on License Ontology, these rules are applied through Protege. The designed rules, queries and its logical forms are discussed in sections 6.6.

6.3 Protege for License Ontology

Protege is an open source ontology design software (Hai et al., 2007). It provides an open hand to a user to design ontology with the flexibility of expanding, updating or modifying the ontology. Protege provides an extensible infrastructure that allows to construct a domain of ontologies, customize data and provides API, to extend the ontology by using Web services (Hogeboom et al., 2005). SPARQL (Protocol and RDF Query Language) is a query language applied on RDF graph, as a set of triples. The triples consists of subject, predicate and object. Protege provides a facility to apply SPARQL queries on RDF graph, and to accumulates results on basis of queried triple.

Protege software is used to construct License Ontology, as discussed in section 4.4.2. A knowledge based domain of License Ontology for license agreements is further expended by defining instances with in sub-ontologies of License Ontology. The License Ontology is developed in Protege, and is imported by sub-license ontology

to complete the explanation of selected agreements. The defined instances explain sub-license ontology as it is explained in respective agreement. SPARQL queries are applied on sub-license ontologies for reasoning the user requirements. These reasoning are enforced through Protege on each of designed license ontology, to acquire results.

6.4 SPARQL Protocol and RDF Query Language

SPARQL is a query language and is a general term applied for both SPARQL protocol and RDF query language. Usage of SPARQL is like syntactically SQL language, for querying RDF graphs using pattern matching (Melton, 2006). SPARQL uses patterns of conjunctive, value filters, optional and disjunction features. SPARQL simplifies interface, which is supported by HTTP or SOAP.

SPARQL queries RDF graph, and RDF graph consists of triples expressing binary relations between resources i.e. variables or URIs. SPARQL queries are simple and easy to understand having no complexity of joint queries or sub joint query concepts. RDF represents data as a collection of binary relations, therefore SPARQL queries dispartates data sources in a single query.

SPARQL query is relatively young, as compared to SQL or XPath. SPARQL does not support transitive queries, in RDF graph. But SPARQL is simple to understand and easy to grasp information from an RDF graph. SPARQL queries are used to have required information from implemented License Ontology and sub-license ontologies. In section 6.6, SPARQL queries applied on sub-license ontologies are elaborated in form of rules. Six of the different sub-license ontologies are implemented, and each of the sub-license ontology has different result using SPARQL queries.

6.5 Rules used for Constructing License Ontology

In layered architecture of Semantic Web, rules have a basic role in freely mixing of properties and class expressions. Rules are needed for inference about classes and properties, mapping ontologies, transforming data in different format, using complex queries, axioms and many more (Bak et al., 2010). The aim is to populate the domain of License Ontology by means of rules created form analyzing set of text documents. Approach to license agreements in thesis, mainly concerns with entities related to Agreement, License, Software, Modified Source Code, etc. These

entities are related with terminology used in License Ontology and sub-license ontologies; and are created on basis of thorough study of different documented terms and condition of agreements. Each rule applied on License Ontology and sub-license ontologies is described as follows;

6.5.1 Disjoint Classes in Ontologies

Disjointness of a set of classes guarantees that an individual which is a member of one class cannot simultaneously be a member of other class, in a set. Therefore, in License Ontology only those classes that have information related to terms and conditions of agreement are considered disjoint. In License Ontology, classes used under “Agreement Items” class, i.e. “Breach”, “Core Agreement” and “Non Core Agreement”, and also the sub-classes of these defined classes are disjoint to each other.

6.5.2 Axiom in Ontology

Axioms are of two types i.e closure axiom and covering axiom. A closure axiom is a restriction on properties and states that it can only be filled by a specified fillers. A covering axiom consists of two parts i.e. a class is being covered by another class and a class that covers other classes or sub-classes. For example; there are three classes A, B and C. A is a super class of B and C and A is a covering class of B and C. This means that an instance of type A could be either a member of B or C, if B and C classes are disjoint classes. An individual of A is a member of class A; in disjoint case, the member of class A, is also a member of class B and C.

Axiom is used and defined in sub-license ontology of License Ontology i.e. Software License Ontology using “LicenOnto: Product” class. The axiom states that union of “Source Code” class and “LicenOnto: Trade Mark” class is a “LicenOnto: Product” class. The axiom explains that the “LicenOnto: Product” class is the combination of two classes i.e. “Source Code” class and “LicenOnto: Trade Mark” class. A member of “LicenOnto: Product” class is either a member of “Source Code” class or a member of “LicenOnto: Trade Mark” class, because “Source Code” class and “LicenOnto: Trade Mark” class are not disjoint classes. The axiom is reflected as;

$$\text{LicenOnto:Product} \sqsubseteq \sqcup \text{SourceCode} \cdot \text{LicenOnto:TradeMark}$$

In above defined axiom, a keyword “LicenOnto:” is used with two of the class names. This explains that these classes are inherited from “License Ontology” (i.e. the root

ontology). The same keyword is used in some of below restrictions, and means the same.

6.5.3 Value Partition in Ontology

Value partition is used to refine the description of various classes. Value partition is used as design patterns in ontology language and is used to solve the modeling problems. Therefore in License Ontology the classes i.e. “Distributable Product” “Downloadable Product” and “Transferable Product” are included in the ontology. With the help of these three classes, License Ontology refines the description of distributable, Downloadable and transferable products, according to license agreement. These classes divide the list of products available under one license agreement, respectively.

Value partition will distinguish between the distributable products and non distributable products. Therefore, value partition enable us to classify the License Ontology (i.e. the root ontology) in distributable, downloadable and transferable products. Restrictions applied on respective classes are represented accordingly;

$$\text{DistributableProduct} \equiv \exists \text{ isAllowedTo} . \text{DistributionAllowed}$$

$$\text{DownloadableProduct} \equiv \exists \text{ isAllowedTo} . \text{DownloadAllowed}$$

$$\text{TransferableProduct} \equiv \exists \text{ isAllowedTo} . \text{TransferAllowed}$$

6.5.4 Restriction used for License Class

“LicenOnto: License” class provides details of terms and condition of an agreement, as described in section 4.5.1. According to some license agreements i.e. Mozilla agreement, it is stated that user can provide a text document (i.e. a sub-license agreement) to define terms and conditions for a modified source code. But this text document should not modify original license agreement. This condition is reflected as;

$$\text{LicenOnto:License} \sqsubseteq \forall \text{ isUnchangeableBy} . \text{SubLicenseAgreement}$$

It states that for all values of an agreement (i.e. “LicenOnto: License” class) is unchangeable, by a user defined agreement (i.e. “Sub License Agreement” class) of modified source code. This restriction is define at Software License Ontology.

6.5.5 Restriction used for Product Class

“Product” class is described in section 4.5.1. “Product” class use a restriction, which should be fulfilled by an individual, and this restriction is a necessary condition. The restriction is given as follows;

$$\text{Product} \sqsubseteq \forall \text{hasDocumented} . \text{License}$$

It states that for all values of product should have an agreement (i.e. “License”). Vice versa the restriction applied also on “License” class because both properties are inverse to each other, the restriction is as follows;

$$\text{hasDocumented} \equiv \text{isForEach}$$

6.5.6 Restrictions used for Modified Source Code Class

“Modified Source Code” class is defined in Software License Ontology to inform end user whether the modification in source code is allowed or not according to an agreement. A combinations of two restrictions are used to complete the meaning of this class. First restriction states that an end user (i.e. developer) wants to distribute his work and also allow others to distribute his modified source code on Web (i.e “LicenOnto:Electronic Distribution”). Second restriction states that modified source code should be distributed under terms and conditions of an agreement. The restriction is defined below;

$$\text{ModifiedSourceCode} \sqsubseteq (\exists \text{isAllowedOn} . \text{LicenOnto:ElectronicDistribution}) \sqcap (\forall \text{isDistributedUnderTermsOf} . \text{LicenOnto:License})$$

It states that modified source code is allowed to distribute electronically and should be distributed according to agreement.

6.5.7 Restrictions used for Transfer and Download Class

Purchasing and transferring a product has same restriction that lays under necessary condition. Both restrictions state that downloading (i.e. purchasing) or transferring of product is available only under terms and conditions of agreement. These restrictions are represented as;

$$\text{TransferType} \sqsubseteq \forall \text{ isAvailableAs} . \text{AgreementItems}$$
$$\text{DownloadType} \sqsubseteq \forall \text{ isAvailableAs} . \text{AgreementItems}$$

This means that user can download or transfer a purchased product, but it should be done according to terms and conditions of an agreement.

6.5.8 Restrictions used for Sub License Agreement Class

As discussed in section 5.2, “Sub License Agreement” class is a documented text file by a developer of modified source code. The class keeps information of modified source code, for example about its restrictions and other legal conditions. And the class informs a user about the availability to provide an agreement for a modified source code or not. Restrictions of this class explains that the class should have information of modified source code. It is represented as;

$$\text{SubLicenseAgreement} \sqsubseteq \exists \text{ hasInformationOf} . \text{ModifiedSourceCode}$$

The above rule explains that if a developer of modified source code provides a documented agreement then he should provide information about modified source code. The restriction of “Sub License Agreement” class is a necessary condition, which means that if the class has a member then it should fulfill the restriction.

6.6 Reasoning License Ontologies

Acquiring information from an ontology can be determined by rules. After having information from ontology by applying rules, then the next step is to represent the information. In this section, ten rules are introduced that are built according to the previously discussed user scenarios in section 3.4. These rules describe complete requirements of a user, to search for appropriate license using ontologies.

Nine different sub-license ontologies are extended from a root ontology (i.e. License Ontology). These nine ontologies are based on six license agreements. Six types of ontologies are constructed using these six agreements and the ontologies differ from each other because of defined instances, according to each agreement. The remaining three ontologies are the connections between parent ontology and child ontology. These three ontologies are Software License Ontology, Online Shopping

License Ontology and Creative Commons Licensing Ontology. The six ontologies have a similar structure because of License Ontology, but with different definitions of instances. Due to centralization of ontology, same rules are applied on each of sub-license ontologies and results depend on defined instances. In each of the following sub section same rules are applied on different sub-license ontologies and their results show the limitation and flexibility of an agreement.

6.6.1 Online Payment

Rule for online payment provides information, about procedure of payment according to an agreement. Antecedent part of the rule describes, that a product has an ontology. And according to the ontology, it is inquired that payment procedure supports online payment of the product or not. Then availability of user permissions are checked, for online purchasing of the product, according to the ontology. When antecedent holds argument, than ultimately consequent part of the rule is true. Thus, according to this rule, an end user is allowed to purchase the product according to an agreement as described follows;

$$\begin{aligned}
 & (?product LicenOnto:hasLicense ?license) (?license \\
 & LicenOnto:supportPaymentMethod LicenOnto:Online) (?user \\
 & profile:paymentMethod profile:Online) \rightarrow (?user profile:canPayOnline ?product)
 \end{aligned}$$

6.6.2 Method of Receiving a Product

The following rule is about downloading a product according to an ontology. Antecedent part describes, that a product having agreement and allows a user to download the product from Web. The antecedent examines whether a user can download the product from Web or not. According to ontology, if antecedent part holds answer then consequent part of the rule will also hold answer and allow user to download the product online.

$$\begin{aligned}
 & (?product LicenOnto:hasA ?license) (?license LicenOnto:receiveMethod \\
 & SoftOnto:Download) (?user profile:receiveMethod profile:Online) \rightarrow (?user \\
 & SoftOnto:canDownload ?product)
 \end{aligned}$$

6.6.3 Replacing of Damaged Product

“Replacing of Damaged Product” rule is about replacing a product, if the product is damage before receiving. Antecedent part describes the rule in reference to a user, who has purchased a product but the product is damage when he received. Therefore, according to ontology the availability of replacing damaged product is evaluated against respective agreement. If a method of replacing a damaged product is allowed according to an ontology then the product is sent back for replacement. Therefore, the antecedent part holds answer, and consequently the consequent part will also hold the answer. According to this rule, the possibility of replacing the damaged product is determined according to agreement.

$$\begin{aligned}
 & (?user \text{ profile:ownsProduct } ?product) (?product \text{ LicenOnto:hasLicense } ?license) \\
 & \quad (?license \text{ LicenOnto:replaceMethod } \text{SoftOnto:Return}) \rightarrow (?user \\
 & \quad \quad \text{SoftOnto:canReturn } ?product)
 \end{aligned}$$

6.6.4 Source Code Modification

“Source Code Modification” rule investigates for modification in a product’s source code according to agreement. Antecedent part of the rule examines, that a product has an ontology and the ontology allows to modify the source code. If ontology allows to modify the source code then antecedent part further examines ontology. And determine, that user has permission to modify the product. If antecedent hold the argument according to ontology then the consequent will also holds the argument and user is allowed to modify the source code of the product according to agreement.

$$\begin{aligned}
 & (?product \text{ LicenOnto:hasLicense } ?license) (?license \text{ SoftOnto:sourceModification} \\
 & \quad \text{SoftOnto: Allowed}) (?user \text{ profile:requiredModification } ?product) \rightarrow (?user \\
 & \quad \quad \text{SoftOnto:canModifySource } ?product)
 \end{aligned}$$

6.6.5 Software Product for specific Operating System platform

In this section execution of products on specified operating system is discussed. Antecedent part of the rule satisfies user about the product’s execution in specific or independent operating system. The rule provides information about an operating system on which the product is executable.

$$\begin{aligned}
 & (?user \text{ profile:hasOS } ?os) (?product \text{ LicenOnto:hasLicense } ?license) (?license \\
 & \quad \text{SoftOnto:supportPlatform } ?os) \rightarrow (?user \text{ SoftOnto:canRun } ?product)
 \end{aligned}$$

6.6.6 Backup Copy

Backup copy of product is discussed in this rule. Antecedent part of the rule satisfies that a backup copy of a product is allowed, or not, according to agreement. If antecedent part hold the statement true then user is allowed to make backup copies of product according to agreement.

$$\begin{aligned} & (?product LicenOnto:hasLicense ?license) (?license SoftOnto:backupCopy \\ & \quad SoftOnto:Allowed) (?user profile:ownsProduct ?product) \rightarrow (?user \\ & \quad \quad SoftOnto:canBackup ?product) \end{aligned}$$

6.6.7 Third party Trademark

To place a logo after modification in a product is discussed in this rule. Rule describes that a third party trademark is allowed for a developer. The antecedent part describes that personalized trademark is allowed or not according to agreement. Therefore, according to the rule ontology provides information to a user that he can place his trademark after modification or updating the product.

$$\begin{aligned} & (?product LicenOnto:hasLicense ?license) (?license SoftOnto:personalizedLogo \\ & \quad SoftOnto:Allowed) (?user profile:hasTrademark ?trademark) \rightarrow (?user \\ & \quad \quad SoftOnto:canPersonalizedLogo ?product) \end{aligned}$$

6.6.8 Third party Distribution

Rule for distributing a product is described as the rule, which determines the approval of redistributing of a product according to agreement. Antecedent part of the rule provides information of re-distribution of product through Web, after modification or without modification in a product. If the agreement authorizes user to distribute the product through Web then antecedent part of the rule is true and ultimately consequent part hold the argument.

$$\begin{aligned} & (?product LicenOnto:hasLicense ?license) (?license LicenOnto:redistribute \\ & \quad LicenOnto:Online) (?user profile:ownsProduct ?product) \rightarrow (?user \\ & \quad \quad LicenOnto:canRedistributeOnline ?product) \end{aligned}$$

6.6.9 Creative Commons (by_nc_nd)

Creative Commons (CC) as discussed in section 2.1.3, is an open licensing mechanism, that defines an agreement through symbols. These symbols are open to use and combination of symbols explain the type of an agreement, as discussed in section 5.2.3. CC license ontology extends the License Ontology and provides a concept that the License Ontology is capable in adopting variety of license agreements.

A rule is developed based on a user scenario, in section 3.4.1. According to this rule, antecedent part discusses that user want to use an organization's Web page as a resource for his own project. But the Web page is secured by CC, and the agreement's ontology discussed in section 5.2.3 does not allow user to use the resource. Therefore, in this case the antecedent part of the rule fails to allow user to use Web page resources.

$$\begin{aligned}
 & (?user \text{ profile:hasAffiliation } ?organization) (?user \text{ profile:stores } ?webpage) \\
 & (?webpage \text{ LicenOnto:contains } ?resource) (?resource \text{ CC:isSecuredBy} \\
 & \text{ CC:by_nc_nd}) \rightarrow (?user \text{ LicenOnto:notAllowedToCopy } ?resource)
 \end{aligned}$$

CC has defined four types of terms and conditions for their users. Combination of any three or less than out of four terms, defines a complete agreement of a product used by an author. Each of the four terms are used in form of queries to make reasoning using ontology. The eyeOS Web agreement is selected for this thesis to carry out the reasoning, using eyeOS Web ontology. These four terms of CC are used in constructing eyeOS Web ontology, according to eyeOS Web agreement.

6.6.10 Reimbursement of Payment

Rule of reimbursement is used in online shopping products. Some times user receives a product from online shopping centers, and the product is not according to the defined properties described on Web site. Customers are provided a defined reimbursement procedure, to follow after receiving a damaged product. Therefore, in ontology a rule is used to inform user about availability of reimbursement procedure according to an agreement. Whenever, such situation occurs then following rule is usable for customers to find out the availability of reimbursement method, according to agreement.

$$\begin{aligned}
 & (?user \text{ profile:hasPurchased } ?product) (?product \text{ LicenOnto:hasLicense } ?license) \\
 & (?license \text{ LicenOnto:replaceMethod OnliShopOnto:reimbursement}) \rightarrow (?user \\
 & \text{ OnliShopOnto:canReturn OnliShopOnto:reimbursement})
 \end{aligned}$$

In this chapter, the applied restrictions and rules in constructing the ontological model are discussed. These rules and restrictions simplifies the complex “terms and conditions” of license agreements in constructing the License Ontology and its sub-ontologies. In chapter 7, construction of GUI is explained, the GUI is developed for end-users to use the constructed ontologies and retrieve their required information from the ontologies.

Chapter 7

Knowledge Based Domain using Jena API

Subsequently introducing the License Ontology in an easily understandable way, using graphical user interface, this chapter provides description of knowledge based domain of license agreements using Jena API. Jena 1 was first released in 2000, and upgraded as per requirement in August 2003, as Jena 2, the latest version of Jena is Jena 2.2. The knowledge based domain implementation provides a graphical user interface, as described in this chapter.

7.1 Jena Architecture

Jena architecture is based on RDF API, and supports RDF graph in creation and manipulating ontology. RDF API helps developer to represent an RDF graph, by using inference layer, query functions and network APIs. RDF API has two distinct ways to represent an RDF graph, these are 1) Statement Centric Representation and 2) Frame Centric Representation. Jena API integrates both representation of RDF graph, to make a complete Jena architecture as shown in figure 7.1 (McBride, 2002). The Jena architecture provides APR parser that reads RDF/XML to represent the RDF graph. RDF/XML provides flexibility in writing RDF graphs in different ways, using Jena architecture. The multiple and flexible presentation of RDF graph allows data to manipulate through high level interfaces. The Jena API provides various tools, i.e. RDF/XML, N3, n-triples and query language, for manipulating RDF graph. Jena API helps user to store RDF graph in memory or persistent storage (McBride, 2002). Manipulated data of RDF graph in form of triples is particularly useful in RDFS and OWL reasoning. Jena architecture facilitates RDF

graph reasoning by supporting Semantic Web query language i.e RDQL (Miller et al., 2002).

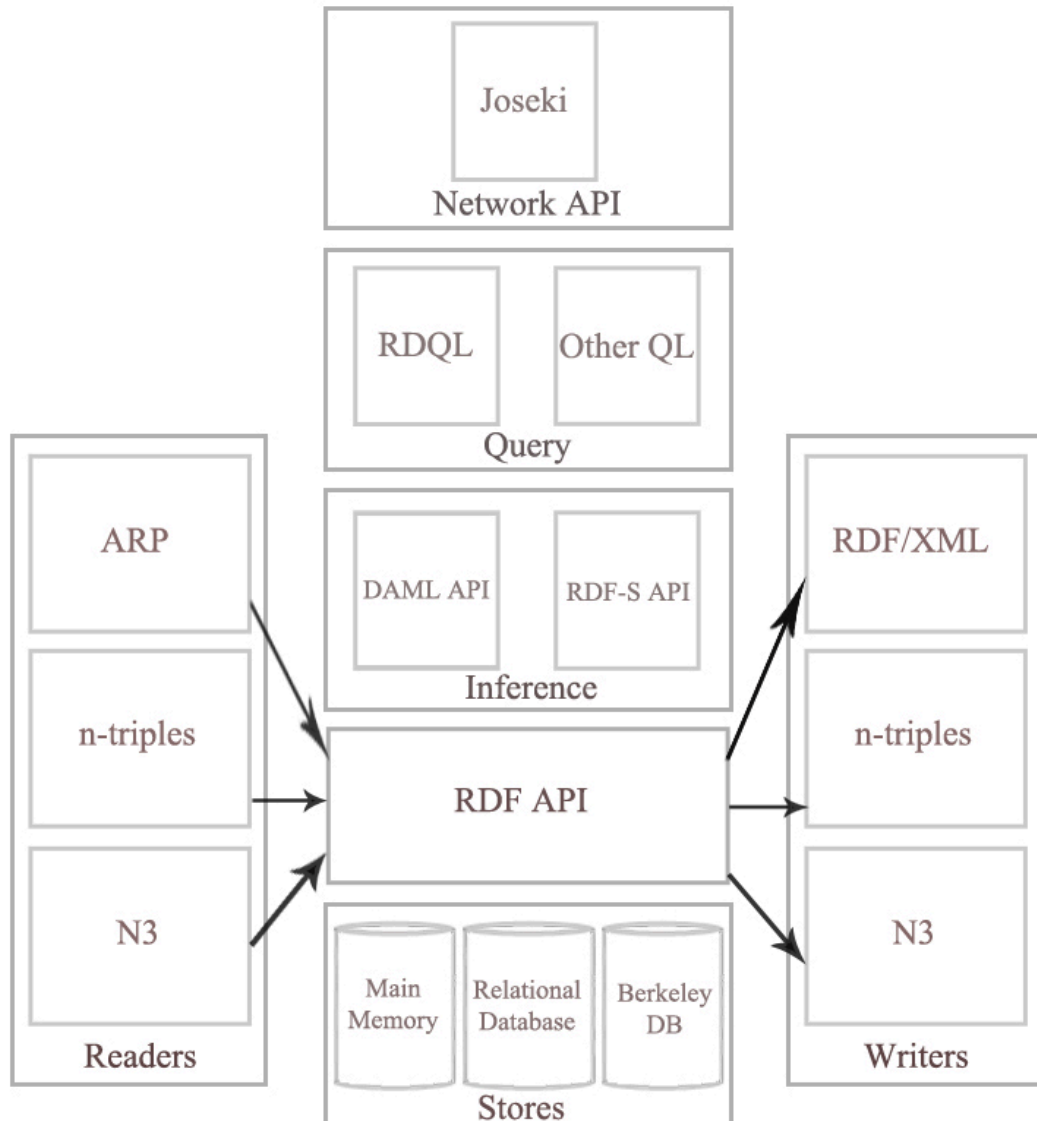


Figure 7.1: Jena Architecture.

The Jena architecture provides a base to Semantic Web query to access RDF graphs in Web clients (Seaborne, 2002). The query based access in Jena architecture act as an identical theme for the architecture. Jena provides a range of rich RDF/XML grammar, N3 support, storage of RDF graph, query language and Web API support (Carroll et al., 2004).

7.2 Semantic Web and Jena Architecture

Various API's are available to represent ontology information used in Semantic Web, one of them is Jena API. Jena provides a platform to represent ontology information

through programming interface, and also helps to designing ontology. Jena is based on RDF platform and Jena supports to built and to formalize ontologies on RDF but at limited extent. Therefore, Jena provides a brief introduction to RDFS and OWL. OWL allows a full detail to organize classes in ontologies as like the RDFS. OWL is built on top of RDF specifications and RDF provides triples as the core formalization for OWL. Jena support RDF and OWL to perform operations on ontologies using triples.

Jena has extended version of Model class, which allows access to collect RDF data. OntModel of Jena is used to access classes, objects and individuals of an ontology (i.e predicate), using RDF triples. Jena API reads and/or updates an ontology, but doesn't change the RDF representation of an ontology. Jena also provides facility to make reasoning about an ontological model. Jena API apply queries on an ontological model for reasoning and inferences as like an RDF uses SPARQL.

Digital License Agreement (DLA) is developed using Jena API with Java, for programming ontology interface. Through Jena API, License Ontology is accessed and analyzed by the queries. DLA is explained by discussing a user scenario in section 7.3.1, and DLA application in section 7.4 .

7.3 Use Case

Customers purchases various items from various locations, according to requirements, price and quality. Some of the purchased items require signatures of an end user before purchasing. Habitually, end user signs the agreements without understanding it and after signing such agreements, they are vulnerable to commit violation. Digital License Agreement (DLA) is designed to solve such situation and also updates the end user about his status of liabilities and restriction of using purchased products. To elaborate such problem a user scenario is discussed in detail:

7.3.1 Study at Austria

A student from a country, other than Austria, transferred his credits to Austria, according to cultural exchange program, to complete his education. During his stay he purchased software products from university. He also did some online purchases and bought a high definition camera with very cheap price. The camera requires a sensor cleaning pads, after some interval of time.

The software which he purchased for his education during the stay, are only allowed for students. Using his high definition camera in his county is forbidden for selling according to the agreement. Therefore, he could not purchase the sensor cleaning pads in his country.

After completing his education, he flew back to his home land. He initiated a software developing company. The company prospers very fast but after some years software agreement's violation authority personals visited his company, and found software which was purchased for education purpose is used for commercial purposes. Due to violation of agreement his company is subject to penalty.

For these above problems, DLA application is proposed. DLA keeps a record of a user and his purchased items. The application is also capable to warn the user about the restrictions of purchases and current status of liabilities. DLA application is based on License Ontology and provides complete information of different license agreements to DLA application. DLA application facilitates a user by searching his requirements, the searching is carried out by using License Ontology. DLA is explained below;

7.4 Digital License Agreement (DLA)

Digital License Agreement (DLA) is a graphical interface of the License Ontology. The graphical interface facilitates an end-user to select queries related to an agreement and post these queries to retrieve required results. An overview of steps followed in implementing DLA is shown in figure 7.2. DLA uses Jena, Java and My SQL for implementing the DLA application. Jena provides support for object oriented programming in handling RDF data model, to create a Semantic Web application (Oren et al., 2007). Jena also provides a general rule engine, which helps to apply SPARQL queries on inferred graph (Huang and Javed, 2008). Using the DLA application, Java methods access RDF triples using Jena API. The different methods used for implementing DLA are discussed below.

7.4.1 DLA Searching Mechanism

Ontology retrieval and searching mechanism, in DLA application, is based on triples. These queries are based only on predicate of the triple. Subject and object of a triple are retrieved with the help of predicate. Each successfully retrieved subject and object are stored in a buffer, with respected predicate.

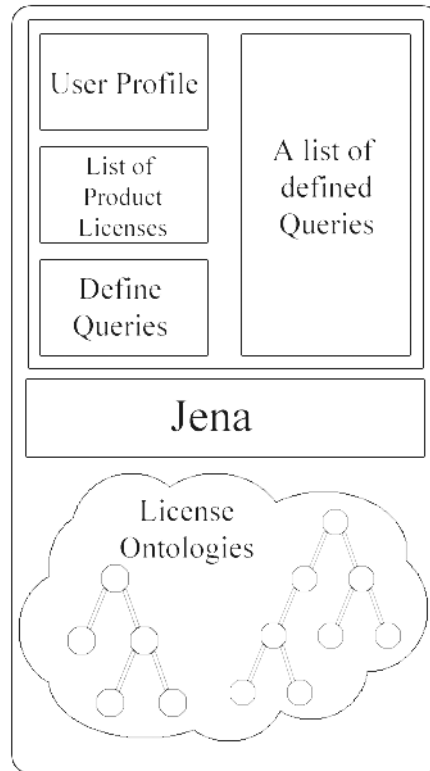


Figure 7.2: DLA Overview.

The retrieved information is compared with user requirements. User query helps to retrieve required information by applying search on Sub License Ontologies. A search is a successful search by successfully matching user requirements with predicate of a triple. Same procedure is repeated again and again to find successful hits of each triple using the iterator statement.

7.4.2 Solution to the Use Case

DLA is capable of keeping record of user and of purchased items. DLA facilitates user in describing status of a user, and also facilitates in describing the purchased item's liabilities and/or penalties. The liabilities and/or penalties are liable according to agreement, when violation occurs. Using the DLA application, user's profile helps user to check liabilities and penalties according to agreement as per user requirement. DLA application provides information of restrictions, copyright and user privileges according to an agreement. The restrictions explain different aspects of illegal usage of products, which includes the jurisdictional restriction, number of installation per client (in software) and distribution or transfer of products. DLA is a complete application that covers all aspects of a license agreement, and informs user about the aspects of violation.

7.4.3 Results of DLA

DLA application provides searching facility for end users; to search the required information. Therefore, concepts of license agreements are categorized into different classes (or keywords), as explained in chapter 4 and 5. In this section, results of DLA application, based on Software License Ontology, are discussed.

Results of DLA depends on two different types of methods used to search license agreements. According to first method, defined user queries are applied on all license ontologies. In this method user defines his requirements, by selecting different options to make a query. The query is applied on license ontologies to get required results, as shown in figure 7.3.

According to the result, as shown in figure 7.3, a query is applied on three license ontologies (i.e. Mozilla, Apache and Adobe). The result is based on a query, which states that a developer wants to download a product, source code of a product should be available and after modification, developer is allowed to use his logo or trade mark. The result describes that Mozilla License Ontology and Apache License Ontology legally allows to download their products, developer can modify their product's source code and after modification developer is allowed to place his trade mark with modified source code. While according to Adobe license agreement user is not allowed to make modification in source code of their software products. Therefore, Adobe license agreement is not selected for the final output.

According to second method, a user selects a product and after selection of product, then he selects different categories of a license agreement. The categories of license agreement are actually keywords (i.e. class names, as discussed in chapter 4 and 5) and each keyword is a user requirement. The selected keywords defines a query. The DLA application select license ontology, on the basis of user selected products. The defined query is applied on selected license ontology using DLA application and results are displayed on screen, as shown in figure 7.4.

The results shown in figure 7.4, are based on selection of XMLBean, a product of Apache License Agreement. After selecting software product, query is defined to exploit selected license agreement. The selected query in figure 7.4, is based on four user requirements i.e. Copy Right Agreement, Backup, Developer and Download. The DLA application, provides results on basis of Apache License Ontology, which describes the statements in form of triples used in ontology according to the agreement.

A GUI for license ontologies is developed in order to reduce cognitive load of end-users. The designed DLA application is a GUI application and it provides ac-

cessibility for less skilled end-users. Thus it reduces end-users cognitive load in understanding the “terms and conditions” of license agreements.

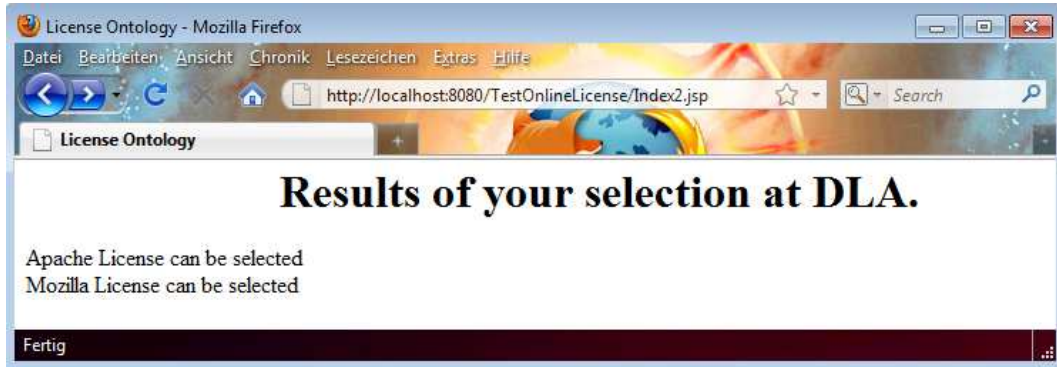


Figure 7.3: Results obtained by applying previous on all License Ontologies.

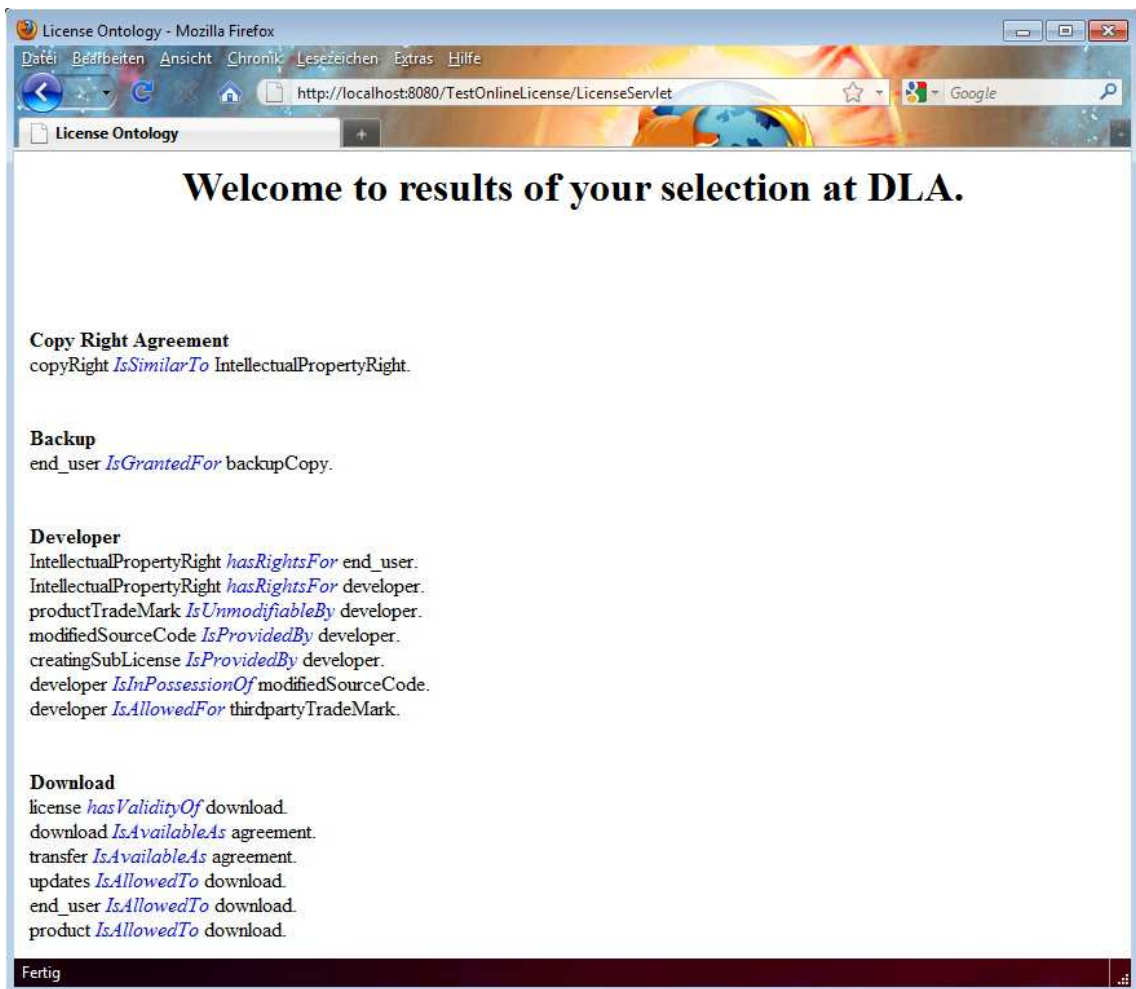


Figure 7.4: Results Obtained by applying query on selected License Ontology using second method.

Chapter 8

Conclusion & Future Work

Initially, undertaking an agreement without understanding is a major issue. Habitually customers agree with terms and conditions of an agreement, that they never read and understand. Incomplete knowledge and understanding about a license agreement in general and agree without an agreement with understanding specifically - is a main problem for ones own self. Customers faces penalties after violating any section to an agreement. The customer has to pay for penalties after violation and/or they cannot continue their running business (Asfandeyar et al., 2009), (Asfandeyar et al., 2010). To solve the issue, an application is developed, which is easy in understanding and reading a complicated software license agreement.

A convincing case study is a diary of a researcher to hand a wide range of information accumulated over a long period of time. For designing DLA application, a case study is explained in section 3.4, to define a set of requirements that visualize an end user queries. Set of requirements is an initial step to define goals, which will be achieved after designing a complete application. In designing the application, different steps are followed. For example, after requirements definition, a suitable platform is selected that could extend the research work. Semantic Web is selected as a platform for modeling the license agreement. The Semantic Web is selected because of its extensibility and adaptability in most of software language tools. Using Semantic Web, an ontological model is created from common features of different license agreements and this ontological model is extended to design sub-license ontologies for specific agreements. Construction and modeling of ontology is explained and discussed in section 4.4. After having a complete structure of ontology, a user interface using Jena API is developed . The user interface provides an open hand for end user to select their required queries, and submit the queries to the system. Defined queries are processed and results are sent to user on bases of selected license ontology. In this chapter, answers to research question described in section 1.3 are provided, and future work is also discussed.

8.1 Review and Answer to Research Questions

In this section research question are reviewed and answered in realizing the digital system, i.e. DLA, for license agreements and also in the light of user requirements. The overall research question of this thesis is stated as; *To what extent can a machine simplify a license agreement for a user in understanding and identifying critical issues of a license agreement before undertaking any agreement? Which techniques are feasible to be implemented on a machine that can make it accessible to end user?* To elaborate this research question it is subdivided into following research questions and answers to each research question is explained, according to the work done as described in this thesis.

- *RQ1: What are the main problems for a user in accepting an agreement without fully understanding or thoroughly reading the agreement? What are the problems that are generated after accepting an agreement? After identifying the problems of a user while accepting agreement, what will be a proper solution that makes the agreement easily understandable? How to bring all license agreements to one platform that could be easily accessible to all users?*

In addressing the RQ1 research question, some of the major problems occurred with end-user after violating an agreement are discussed, as described in chapter 1. In section 1.1, relating to these problems, a solution is proposed and then an application is developed. The application helps an end-user in having required information easily available, without reading a complete license agreement. The application is designed in such a way that different license agreements can be easily plotted on same platform with minor changes according to agreement. Use case and user scenarios are discussed in section 3.4, these use case and scenarios define different requirements, related to purchasing software product, and agreeing with the terms and condition of an agreement. Figure 3.3, provides a complete over view of user scenarios, which provide a base to construct the DLA application.

- *RQ2: To what extent Semantic Web technology is able to replace license agreements of service providers? Can Semantic Web technology provides a bridge between service provider and end user? How to categorize different phases of license agreements using Semantic Web for modeling an ontology and defining a knowledge base domain of license agreements?*

Based on Semantic Web technology, a License Ontology is constructed, according to license agreements. License Ontology, as explained in section 4.5, provides a plat-

form for different license agreements to construct an ontology according to the agreement's requirements. License Ontology is constructed using common features of different license agreements, an RDF of License Ontology is given in appendix A. After constructing License Ontology, the ontology is imported to construct sub-license ontologies, for a specified license agreements. Six different license agreements are used to create sub-license ontologies. The sub-license ontologies are constructed using License Ontology and defining instances according to license agreements. Sub-license ontologies are explained and discussed in Chapter 5. In appendix B to appendix G, an RDF Sub License Ontology of agreements are provided. Mozilla, Apache, Adobe, Amazon, eBay and eyeOS application license agreements are selected, and used in development of DLA application. Mozilla and Apache are open source code provider license agreements and under these two license agreements number of products are available. Adobe license agreement, a commercial software agreement is selected, which does not provide source code of its products. EyeOS Web application license agreement is using CC licensing agreement. CC licensing agreement is discussed and explained in section 2.1.3 and 5.2.3. CC licensing agreement provides a proof of the statement that DLA application is an adaptable application for different license agreements with minor changes according to a license agreement. Explanation of DLA application and its functionality is discussed in section 7.4. DLA application is supported with java script and a MySQL database, as well as Jena API and Semantic Web technology, using Java platform.

Java programming language, Jena API, Java Script and MySQL database are used to develop the DLA application. Jena API and MySQL helps to develop communication with Semantic Web. Jena API is used with Java programming, to apply SPARQL queries on Semantic Web and also used MySQL in maintaining end user's information in database. For applying SPARQL queries of Jena API and maintaining MySQL database, Java programming language and Java Script provides User Interface for end users and software developers.

Two research papers are published in year 2009 (Asfandeyar et al., 2009) and 2010 (Asfandeyar et al., 2010), explaining the concepts of License Ontology and DLA application. In paper (Asfandeyar et al., 2009), explains License Ontology (named as abstract ontology) and sub-license ontologies (named as license ontology). In this paper, results obtained from sub-license ontologies on basis of defined user requirements are discussed, as described in section 3.4. In paper (Asfandeyar et al., 2010), with respect of License Ontology and sub-license ontologies, CC licensing agreement and its ontology is also explained, as described in this thesis in section 5.2.3. Further eyeOS Web license agreement is added, as an example to explain the purpose of DLA application, i.e a unique platform for different types of license agreements.

- *RQ3: Is Semantic Web able to provide facility to service providers in creating a license agreement for their new product? Another question arises in same context on how to update an existing ontology of license agreement for a product provider and what steps are required to present the agreement in a documented form?*

Based on DLA application, instances are added to sub-license ontologies on the basis of user requirements. Jena API provides a support to add instances in an ontology. Using Jena API, *model.createResource()* and *instance.addProperty()* built-in functions help to add user required information into the ontology. Ontology uses triples to define a statement, therefore output of the file is always in form of triples. The documented file of a license agreement retrieved from License Ontology is also in form of triples, which is meaningful and understandable by a user.

- *RQ4: How can a user and service provider, easily and effectively retrieve the required useful information from the agreement ontology? Additionally how can we overcome the problems of comprehensibility and semantic overload caused by using complex and detailed license agreements?*

Based on ontologies, DLA application uses triples to define a meaningful sentence. Therefore, the results are based on the triples and these triples are designed carefully so that meaning of a statement in a license agreement should not be changed. The meaningful triples of a license ontology is understandable by a simple user. The DLA application is designed in such a way that user can retrieve his required information by just selecting requirements. After defining requirements user submit his query to DLA application and in return results are elaborated categorically and displayed to the user, as shown in figure 7.4.

Six of the license agreements are selected to perform the experiments and to proof the hypothesis, the hypothesis is discussed in section 1.3. These license agreements are selected on the basis of requirements defined in section 3.4. From the selected license agreement common features are gathered to design a comprehensible ontology (i.e. License Ontology), which can be easily adopted by all license agreements.

8.2 Future Research

DLA is currently elaborated according to user requirements. Although DLA focuses on visualization of search results, ontology and means of incorporating documented

license agreement with the system, to help out an end user. It is necessary to assist end user in refining queries in a manner to acquire required results from sub-license ontologies. DLA shows the feasibility, usability and efficiency in acquiring end user requirements using license agreements. This thesis addresses the stated research questions and further issues arises from the research results are;

Other than License Agreements of Software and Online Shopping: DLA is designed on basis of software product's and online shopping's license agreements, further extensions of DLA supports for the other License Agreements; for example agreements of hardware products, agreements of services providing companies, etc. Here the question arises that *is it possible to use same ontological structure for other license agreements?* DLA is designed for software product's and online shopping's license agreements, and only deals with the License Ontology (i.e. root ontology), which is based on the common features of the license agreements. Idea of developing such application for other than the software & online shopping license agreements, is same to model sub-license ontologies for different types of agreements. Same steps could be followed to design such application. The application will not only provide benefits to end user but also to the authors of license agreements. For authors, a unique template of a license agreement will provide a base to construct a license agreement for an individual. Extensions in License Ontology are discussed as follows.

Airline's Terms and Conditions: International airline's "terms and conditions" has mostly common features, which are addressed under common articles such as reservations, check-in, limitation on carriage, baggage, changing schedule of flights, refund, etc. On basis of these common features; an ontology could be created, that would facilitate an end-user to figure out required information about any airline. Ontology for airlines will also help end users in comparing different international airlines and find passenger benefits, which they can get while traveling.

Cross Boundary Agreements and its Conflicts: DLA is an initial step towards a platform to consolidate license agreements of same categories. DLA is an example for software and online shopping agreement license consolidation. And "Airline's Terms and Condition" explains the same process for other types of license agreement. Whenever, centralization of same categories of license agreements are achieved, then the ontology will be able to find out the conflicts between different laws (i.e. terms and conditions or license agreements). For example, airlines have to make agreements for using the airports and airspace of other territories. Normally, settling downing the conflicts in agreements, consumes maximum time. Such

system will not only explain the terms and conditions of each partner but will also explain the compromising statements that took place between them. The compromising statements will be helpful for other organization to settle down their conflicts with the same organization. The same approach of DLA, could be foresee in other conflicting agreements or laws; for example, in jurisdiction and in legislation laws.

Appendix A

RDF License Ontology

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>
<rdf:RDF xmlns="http://www.owl-ontologies.com/LicenseOntology.owl#"
  xml:base="http://www.owl-ontologies.com/LicenseOntology.owl"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="AgreementItems"/>
  <owl:Class rdf:ID="Breach">
    <rdfs:subClassOf rdf:resource="#AgreementItems"/>
  </owl:Class>
  <owl:Class rdf:ID="CopyRightAgreement">
    <rdfs:subClassOf rdf:resource="#CoreAgreement"/>
  </owl:Class>
  <owl:Class rdf:ID="CoreAgreement">
```

```

    <rdfs:subClassOf rdf:resource="#AgreementItems"/>
</owl:Class>
<owl:Class rdf:ID="Customer">
    <rdfs:subClassOf rdf:resource="#User"/>
</owl:Class>
<owl:Class rdf:ID="DistributableProduct">
    <owl:equivalentClass>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#isAllowedTo"/>
            <owl:someValuesFrom rdf:resource="#DistributionAllowed"/>
        </owl:Restriction>
    </owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="#Product"/>
</owl:Class>
<owl:Class rdf:ID="DistributionAgreement">
    <rdfs:subClassOf rdf:resource="#CoreAgreement"/>
</owl:Class>
<owl:Class rdf:ID="DistributionAllowed">
    <rdfs:subClassOf rdf:resource="#DistributionAgreement"/>
</owl:Class>
<owl:Class rdf:ID="DistributionNotAllowed">
    <rdfs:subClassOf rdf:resource="#DistributionAgreement"/>
</owl:Class>
<owl:Class rdf:ID="DomainConcept"/>
<owl:Class rdf:ID="DownloadableProduct">
    <owl:equivalentClass>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#isAllowedTo"/>
            <owl:someValuesFrom rdf:resource="#DownloadAllowed"/>
        </owl:Restriction>
    </owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="#Product"/>
</owl:Class>

```

```

<owl:Class rdf:ID="DownloadAllowed">
  <rdfs:subClassOf rdf:resource="#DownloadType"/>
</owl:Class>

<owl:Class rdf:ID="DownloadNotAllowed">
  <rdfs:subClassOf rdf:resource="#DownloadType"/>
</owl:Class>

<owl:Class rdf:ID="DownloadType">
  <rdfs:subClassOf rdf:resource="#DomainConcept"/>
</owl:Class>

<owl:Class rdf:ID="ElectronicDistribution">
  <rdfs:subClassOf rdf:resource="#DistributionAgreement"/>
</owl:Class>

<owl:ObjectProperty rdf:ID="hasA">
  <rdfs:domain rdf:resource="#Product"/>
  <rdfs:range rdf:resource="#License"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasItems">
  <rdfs:domain rdf:resource="#License"/>
  <rdfs:range rdf:resource="#AgreementItems"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasRightsFor">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#CopyRightAgreement"/>
        <owl:Class rdf:about="#IntellectualPropertyRight"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="#User"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasRoyaltyFree">
  <rdfs:domain rdf:resource="#PatentsAgreement"/>

```

```

    <rdfs:range rdf:resource="#TransferType"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="hasValidityOf">
    <rdfs:domain rdf:resource="#License"/>
    <rdfs:range>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#DistributionAgreement"/>
          <owl:Class rdf:about="#DownloadType"/>
          <owl:Class rdf:about="#ReplacementAgreement"/>
          <owl:Class rdf:about="#TradeMark"/>
          <owl:Class rdf:about="#TransferType"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:range>
  </owl:ObjectProperty>
  <owl:Class rdf:ID="IntellectualPropertyRight">
    <rdfs:subClassOf rdf:resource="#DomainConcept"/>
  </owl:Class>
  <owl:ObjectProperty rdf:ID="isAllowedFor">
    <rdfs:domain rdf:resource="#ServiceProvider"/>
    <rdfs:range rdf:resource="#ThirdpartyTradeMark"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="isAllowedTo">
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Product"/>
          <owl:Class rdf:about="#User"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
    <rdfs:range>

```

```

    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#DistributionAgreement"/>
        <owl:Class rdf:about="#DownloadType"/>
        <owl:Class rdf:about="#TransferType"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:range>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isAvailableAs">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#DownloadType"/>
        <owl:Class rdf:about="#TransferType"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="#AgreementItems"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isForA">
  <rdfs:domain rdf:resource="#License"/>
  <rdfs:range rdf:resource="#Product"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isReplacedAs">
  <rdfs:domain rdf:resource="#Product"/>
  <rdfs:range rdf:resource="#ReplacementAgreement"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isReplacedBy">
  <rdfs:domain rdf:resource="#Product"/>
  <rdfs:range rdf:resource="#ServiceProvider"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isSameAs">

```

```

    <rdfs:domain rdf:resource="#CopyRightAgreement"/>
    <rdfs:range rdf:resource="#IntellectualPropertyRight"/>
    <owl:inverseOf rdf:resource="#isSimilarAs"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isSimilarAs">
    <rdfs:domain rdf:resource="#IntellectualPropertyRight"/>
    <rdfs:range rdf:resource="#CopyRightAgreement"/>
    <owl:inverseOf rdf:resource="#isSameAs"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isSubjectTo">
    <rdfs:domain rdf:resource="#User"/>
    <rdfs:range rdf:resource="#Penalty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isUnliableTo">
    <rdfs:domain rdf:resource="#AgreementItems"/>
    <rdfs:range rdf:resource="#Penalty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isUnmodifiableBy">
    <rdfs:domain rdf:resource="#ProductTradeMark"/>
    <rdfs:range rdf:resource="#User"/>
</owl:ObjectProperty>
<owl:Class rdf:ID="License">
    <rdfs:subClassOf rdf:resource="#DomainConcept"/>
</owl:Class>
<owl:Class rdf:ID="NonCoreAgreement">
    <rdfs:subClassOf rdf:resource="#AgreementItems"/>
</owl:Class>
<owl:Class rdf:ID="PatentsAgreement">
    <rdfs:subClassOf rdf:resource="#CoreAgreement"/>
</owl:Class>
<owl:Class rdf:ID="Penalty">
    <rdfs:subClassOf rdf:resource="#Breach"/>
</owl:Class>

```

```

<owl:Class rdf:ID="Product">
  <rdfs:subClassOf rdf:resource="#DomainConcept"/>
</owl:Class>

<owl:Class rdf:ID="ProductTradeMark">
  <rdfs:subClassOf rdf:resource="#TradeMark"/>
</owl:Class>

<owl:Class rdf:ID="ReplacementAgreement">
  <rdfs:subClassOf rdf:resource="#NonCoreAgreement"/>
</owl:Class>

<owl:Class rdf:ID="ServiceProvider">
  <rdfs:subClassOf rdf:resource="#User"/>
</owl:Class>

<owl:Class rdf:ID="ThirdpartyTradeMark">
  <rdfs:subClassOf rdf:resource="#TradeMark"/>
</owl:Class>

<owl:Class rdf:ID="TradeMark">
  <rdfs:subClassOf rdf:resource="#DomainConcept"/>
</owl:Class>

<owl:Class rdf:ID="TransferableProduct">
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#isAllowedTo"/>
      <owl:someValuesFrom rdf:resource="#TransferAllowed"/>
    </owl:Restriction>
  </owl:equivalentClass>
  <rdfs:subClassOf rdf:resource="#Product"/>
</owl:Class>

<owl:Class rdf:ID="TransferAllowed">
  <rdfs:subClassOf rdf:resource="#TransferType"/>
</owl:Class>

<owl:Class rdf:ID="TransferNotAllowed">
  <rdfs:subClassOf rdf:resource="#TransferType"/>
</owl:Class>

<owl:Class rdf:ID="TransferType">

```

```
<rdfs:subClassOf rdf:resource="#DomainConcept"/>  
</owl:Class>  
<owl:Class rdf:ID="User">  
  <rdfs:subClassOf rdf:resource="#DomainConcept"/>  
</owl:Class>  
</rdf:RDF>
```


Appendix B

RDF Online Shopping License Ontology

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
  <!ENTITY LicenOnto "http://www.owl-ontologies.com/LicenseOntology.owl#" >
]>
<rdf:RDF xmlns="http://www.owl-ontologies.com/OnlineShopping.owl#"
  xml:base="http://www.owl-ontologies.com/OnlineShopping.owl"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:LicenOnto="http://www.owl-ontologies.com/LicenseOntology.owl#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://www.owl-ontologies.com/LicenseOntology.owl"/>
  </owl:Ontology>
  <LicenOnto:CopyRightAgreement rdf:ID="copyRight"/>
  <LicenOnto:Customer rdf:ID="customer"/>
  <LicenOnto:DistributionAllowed rdf:ID="distributionAllowed"/>
  <LicenOnto:DistributionNotAllowed rdf:ID="distributionNotAllowed"/>
```

```

<LicenOnto:DownloadAllowed rdf:ID="downloadAllowed"/>
<LicenOnto:DownloadNotAllowed rdf:ID="downloadNotAllowed"/>
<LicenOnto:IntellectualPropertyRight rdf:ID="intellectualPropertyRight"/>
<owl:ObjectProperty rdf:ID="isGrantedFor">
  <rdfs:domain rdf:resource="&LicenOnto;Customer"/>
  <rdfs:range rdf:resource="#Reimbursement"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isIllustratedBy">
  <rdfs:domain rdf:resource="#Reimbursement"/>
  <rdfs:range rdf:resource="&LicenOnto;ServiceProvider"/>
</owl:ObjectProperty>
<LicenOnto:PatentsAgreement rdf:ID="patents"/>
<LicenOnto:Penalty rdf:ID="penalty"/>
<owl:Class rdf:ID="Reimbursement">
  <rdfs:subClassOf rdf:resource="&LicenOnto;DomainConcept"/>
</owl:Class>
<Reimbursement rdf:ID="reimbursement"/>
<LicenOnto:ReplacementAgreement rdf:ID="replacementAllowed"/>
<LicenOnto:ReplacementAgreement rdf:ID="replacementNotAllowed"/>
<LicenOnto:ServiceProvider rdf:ID="seller"/>
<LicenOnto:ThirdpartyTradeMark rdf:ID="thirdpartyTradeMark"/>
<LicenOnto:TransferAllowed rdf:ID="transferAllowed"/>
<LicenOnto:TransferNotAllowed rdf:ID="transferNotAllowed"/>
</rdf:RDF>

```

Appendix C

RDF Amazon License Ontology

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
  <!ENTITY protege "http://protege.stanford.edu/plugins/owl/protege#" >
  <!ENTITY OnlinShop "http://www.owl-ontologies.com/OnlineShopping.owl#" >
  <!ENTITY LicenOnto "http://www.owl-ontologies.com/LicenseOntology.owl#" >
]>
<rdf:RDF xmlns="http://www.owl-ontologies.com/amazon.owl#"
  xml:base="http://www.owl-ontologies.com/amazon.owl"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:LicenOnto="http://www.owl-ontologies.com/LicenseOntology.owl#"
  xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
  xmlns:OnlinShop="http://www.owl-ontologies.com/OnlineShopping.owl#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://www.owl-ontologies.com/OnlineShopping.owl"/>
  </owl:Ontology>
```

```

<LicenOnto:AgreementItems rdf:ID="AmazonAgreement">
  <LicenOnto:isUnliableTo rdf:resource="&OnlinShop;penalty"/>
</LicenOnto:AgreementItems>
<LicenOnto:License rdf:ID="amazonLicense">
  <LicenOnto:hasValidityOf rdf:resource="#amazonTradeMark"/>
  <LicenOnto:hasValidityOf rdf:resource="&OnlinShop;distributionAllowed"/>
  <LicenOnto:hasValidityOf rdf:resource="&OnlinShop;transferAllowed"/>
  <LicenOnto:hasValidityOf rdf:resource="&OnlinShop;downloadNotAllowed"/>
  <LicenOnto:isForA rdf:resource="#amazonProduct"/>
  <LicenOnto:hasItems rdf:resource="#AmazonAgreement"/>
</LicenOnto:License>
<LicenOnto:Product rdf:ID="amazonProduct">
  <LicenOnto:isAllowedTo rdf:resource="&OnlinShop;transferAllowed"/>
  <LicenOnto:isAllowedTo rdf:resource="&OnlinShop;downloadNotAllowed"/>
  <LicenOnto:isAllowedTo rdf:resource="&OnlinShop;distributionAllowed"/>
  <LicenOnto:isReplacedBy rdf:resource="#amazonSeller"/>
  <LicenOnto:isReplacedAs rdf:resource="&OnlinShop;replacementAllowed"/>
  <LicenOnto:hasA rdf:resource="#amazonLicense"/>
</LicenOnto:Product>
<LicenOnto:DistributableProduct rdf:ID="amazonProductdistribute">
  <LicenOnto:isAllowedTo rdf:resource="&OnlinShop;distributionAllowed"/>
</LicenOnto:DistributableProduct>
<LicenOnto:DownloadableProduct rdf:ID="amazonProductDownload">
  <LicenOnto:isAllowedTo rdf:resource="&OnlinShop;downloadNotAllowed"/>
</LicenOnto:DownloadableProduct>
<LicenOnto:TransferableProduct rdf:ID="amazonProductTransfer">
  <LicenOnto:isAllowedTo rdf:resource="&OnlinShop;transferAllowed"/>
</LicenOnto:TransferableProduct>
<LicenOnto:ServiceProvider rdf:ID="amazonSeller"/>
<LicenOnto:ProductTradeMark rdf:ID="amazonTradeMark">
  <LicenOnto:isUnmodifiableBy rdf:resource="&OnlinShop;customer"/>
</LicenOnto:ProductTradeMark>
<rdf:Description rdf:about="&OnlinShop;copyRight">

```

```

    <LicenOnto:isSameAs rdf:resource="&OnlinShop;intellectualPropertyRight"/>
</rdf:Description>
<rdf:Description rdf:about="&OnlinShop;customer">
    <LicenOnto:isAllowedTo rdf:resource="&OnlinShop;downloadNotAllowed"/>
    <LicenOnto:isAllowedTo rdf:resource="&OnlinShop;transferAllowed"/>
    <LicenOnto:isAllowedTo rdf:resource="&OnlinShop;distributionAllowed"/>
    <LicenOnto:isSubjectTo rdf:resource="&OnlinShop;penalty"/>
</rdf:Description>
<rdf:Description rdf:about="&OnlinShop;downloadNotAllowed">
    <LicenOnto:isAvailableAs rdf:resource="#AmazonAgreement"/>
</rdf:Description>
<rdf:Description rdf:about="&OnlinShop;intellectualPropertyRight">
    <LicenOnto:isSimilarAs rdf:resource="&OnlinShop;copyRight"/>
    <LicenOnto:hasRightsFor rdf:resource="&OnlinShop;customer"/>
</rdf:Description>
<rdf:Description rdf:about="&OnlinShop;patents">
    <LicenOnto:hasRoyaltyFree rdf:resource="&OnlinShop;transferAllowed"/>
</rdf:Description>
<rdf:Description rdf:about="&OnlinShop;reimbursment">
    <OnlinShop:isIllustratedBy rdf:resource="#amazonSeller"/>
</rdf:Description>
<rdf:Description rdf:about="&OnlinShop;transferAllowed">
    <LicenOnto:isAvailableAs rdf:resource="#AmazonAgreement"/>
</rdf:Description>
<LicenOnto:Product rdf:ID="thirdpartyProduct">
    <LicenOnto:isAllowedTo rdf:resource="&OnlinShop;transferAllowed"/>
    <LicenOnto:isAllowedTo rdf:resource="&OnlinShop;downloadNotAllowed"/>
    <LicenOnto:isAllowedTo rdf:resource="&OnlinShop;distributionAllowed"/>
    <LicenOnto:isReplacedBy rdf:resource="#amazonSeller"/>
    <LicenOnto:isReplacedAs rdf:resource="&OnlinShop;replacementAllowed"/>
    <LicenOnto:hasA rdf:resource="#amazonLicense"/>
</LicenOnto:Product>

```

```
<LicenOnto:DistributableProduct rdf:ID="thirdpartyProductDistribute">
  <LicenOnto:isAllowedTo rdf:resource="&OnlinShop;distributionAllowed"/>
</LicenOnto:DistributableProduct>

<LicenOnto:DownloadableProduct rdf:ID="thirdpartyProductDownload">
  <LicenOnto:isAllowedTo rdf:resource="&OnlinShop;downloadNotAllowed"/>
</LicenOnto:DownloadableProduct>

<LicenOnto:TransferableProduct rdf:ID="thirdpartyProductTransfer">
  <LicenOnto:isAllowedTo rdf:resource="&OnlinShop;transferAllowed"/>
</LicenOnto:TransferableProduct>

</rdf:RDF>
```

Appendix D

RDF Software License Ontology

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
  <!ENTITY LicenOnto "http://www.owl-ontologies.com/LicenseOntology.owl#" >
]>
<rdf:RDF xmlns="http://www.owl-ontologies.com/Software.owl#"
  xml:base="http://www.owl-ontologies.com/Software.owl"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:LicenOnto="http://www.owl-ontologies.com/LicenseOntology.owl#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://www.owl-ontologies.com/LicenseOntology.owl"/>
  </owl:Ontology>
  <owl:Class rdf:ID="BackupCopy">
    <rdfs:subClassOf rdf:resource="&LicenOnto;DomainConcept"/>
  </owl:Class>
  <BackupCopy rdf:ID="backupCopyAllowed"/>
```

```

<BackupCopy rdf:ID="backupCopyNotAllowed"/>
<DevelopingPlatform rdf:ID="C-plus-plus-Language"/>
<LicenOnto:CopyRightAgreement rdf:ID="copyRight"/>
<SubLicenseAgreement rdf:ID="creatingSubLicense"/>
<SubLicenseAgreement rdf:ID="creatingSubLicenseNotAllowed"/>
<LicenOnto:ServiceProvider rdf:ID="developer"/>
<owl:Class rdf:ID="DevelopingPlatform">
  <rdfs:subClassOf rdf:resource="&LicenOnto;DomainConcept"/>
</owl:Class>
<LicenOnto:DistributionAllowed rdf:ID="distributionAllowed"/>
<LicenOnto:DistributionNotAllowed rdf:ID="distributionNotAllowed"/>
<LicenOnto:DownloadAllowed rdf:ID="downloadAllowed"/>
<LicenOnto:DownloadNotAllowed rdf:ID="downloadNotAllowed"/>
<LicenOnto:Customer rdf:ID="endUser"/>
<owl:ObjectProperty rdf:ID="hasInformationOf">
  <rdfs:domain rdf:resource="#SubLicenseAgreement"/>
  <rdfs:range>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="&LicenOnto;DistributionAgreement"/>
        <owl:Class rdf:about="&LicenOnto;PatentsAgreement"/>
        <owl:Class rdf:about="&LicenOnto;Product"/>
        <owl:Class rdf:about="#ModifiedSourceCode"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:range>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasPermissionOf">
  <rdfs:domain rdf:resource="&LicenOnto;License"/>
  <rdfs:range>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#BackupCopy"/>

```



```

    <owl:Class rdf:about="#ModifiedSourceCode"/>
    <owl:Class rdf:about="#SubLicenseAgreement"/>
  </owl:unionOf>
</owl:Class>
</rdfs:range>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasTermsDescribedIn">
  <rdfs:domain rdf:resource="&LicenOnto;ServiceProvider"/>
  <rdfs:range rdf:resource="#SubLicenseAgreement"/>
</owl:ObjectProperty>
<SystemSettings rdf:ID="independentPlatform"/>
<LicenOnto:IntellectualPropertyRight rdf:ID="intellectualPropertyRight"/>
<owl:ObjectProperty rdf:ID="isAllowedFor">
  <rdfs:domain rdf:resource="&LicenOnto;ServiceProvider"/>
  <rdfs:range rdf:resource="&LicenOnto;ThirdpartyTradeMark"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isAllowedOn">
  <rdfs:domain rdf:resource="#ModifiedSourceCode"/>
  <rdfs:range rdf:resource="&LicenOnto;ElectronicDistribution"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isDerivedFrom">
  <rdfs:domain rdf:resource="#ModifiedSourceCode"/>
  <rdfs:range rdf:resource="#SourceCode"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isDevelopedUsing">
  <rdfs:domain rdf:resource="&LicenOnto;Product"/>
  <rdfs:range rdf:resource="#DevelopingPlatform"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isDistributedUnderTermsOf">
  <rdfs:domain rdf:resource="#ModifiedSourceCode"/>
  <rdfs:range rdf:resource="&LicenOnto;License"/>
</owl:ObjectProperty>

```

```

<owl:ObjectProperty rdf:ID="isExecutedIn">
  <rdfs:domain rdf:resource="&LicenOnto;Product"/>
  <rdfs:range rdf:resource="#SystemSettings"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isGrantedFor">
  <rdfs:domain rdf:resource="&LicenOnto;Customer"/>
  <rdfs:range rdf:resource="#BackupCopy"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isInPossessionOf">
  <rdfs:domain rdf:resource="&LicenOnto;ServiceProvider"/>
  <rdfs:range>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="&LicenOnto;PatentsAgreement"/>
        <owl:Class rdf:about="&LicenOnto;ThirdpartyTradeMark"/>
        <owl:Class rdf:about="#ModifiedSourceCode"/>
        <owl:Class rdf:about="#SubLicenseAgreement"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:range>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isLiableTo">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#ModifiedSourceCode"/>
        <owl:Class rdf:about="#SubLicenseAgreement"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="&LicenOnto;Penalty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isProvidedBy">

```

```

<rdfs:domain>
  <owl:Class>
    <owl:unionOf rdf:parseType="Collection">
      <owl:Class rdf:about="#ModifiedSourceCode"/>
      <owl:Class rdf:about="#SubLicenseAgreement"/>
    </owl:unionOf>
  </owl:Class>
</rdfs:domain>
<rdfs:range rdf:resource="&LicenOnto;ServiceProvider"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isUnableToRestrict">
  <rdfs:domain rdf:resource="&LicenOnto;ServiceProvider"/>
  <rdfs:range rdf:resource="#ModifiedSourceCode"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isUnchangeableBy">
  <rdfs:domain rdf:resource="&LicenOnto;License"/>
  <rdfs:range rdf:resource="#SubLicenseAgreement"/>
</owl:ObjectProperty>
<DevelopingPlatform rdf:ID="Java"/>
<DevelopingPlatform rdf:ID="JavaScript"/>
<SystemSettings rdf:ID="Linux"/>
<SystemSettings rdf:ID="MAC"/>
<ModifiedSourceCode rdf:ID="modification"/>
<ModifiedSourceCode rdf:ID="modificationNotAllowed"/>
<owl:Class rdf:ID="ModifiedSourceCode">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#isDistributedUnderTermsOf"/>
      <owl:allValuesFrom rdf:resource="&LicenOnto;License"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#SourceCode"/>
</owl:Class>

```

```

<SystemSettings rdf:ID="OS"/>
<LicenOnto:PatentsAgreement rdf:ID="patents"/>
<DevelopingPlatform rdf:ID="Pearl"/>
<LicenOnto:Penalty rdf:ID="penalty"/>
<DevelopingPlatform rdf:ID="PHP"/>
<owl:Class rdf:ID="ProductUpdate">
  <rdfs:subClassOf rdf:resource="&LicenOnto;Product"/>
</owl:Class>
<LicenOnto:ReplacementAgreement rdf:ID="replacementAllowed"/>
<LicenOnto:ReplacementAgreement rdf:ID="replacementNotAllowed"/>
<DevelopingPlatform rdf:ID="ScriptingLanguages"/>
<owl:Class rdf:ID="SourceCode">
  <rdfs:subClassOf rdf:resource="&LicenOnto;DomainConcept"/>
</owl:Class>
<SourceCode rdf:ID="sourceCodeAvailable"/>
<SourceCode rdf:ID="sourceCodeNotAvailable"/>
<owl:Class rdf:ID="SubLicenseAgreement">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasInformationOf"/>
      <owl:someValuesFrom rdf:resource="#ModifiedSourceCode"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="&LicenOnto;CoreAgreement"/>
</owl:Class>
<owl:Class rdf:ID="SystemSettings">
  <rdfs:subClassOf rdf:resource="&LicenOnto;DomainConcept"/>
</owl:Class>
<LicenOnto:ElectronicDistribution rdf:ID="thirdpartyDistributionAllowed"/>
<LicenOnto:ElectronicDistribution rdf:ID="thirdpartyDistributionNotAllowed"/>
<LicenOnto:ThirdpartyTradeMark rdf:ID="thirdpartyTradeMarkAllowed"/>
<LicenOnto:ThirdpartyTradeMark rdf:ID="thirdpartyTradeMarkNotAllowed"/>
<LicenOnto:TransferAllowed rdf:ID="transferAllowed"/>
<LicenOnto:TransferNotAllowed rdf:ID="transferNotAllowed"/>
<SystemSettings rdf:ID="Windows"/>
<DevelopingPlatform rdf:ID="XML"/>

```

Appendix E

RDF Mozilla License Ontology

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
  <!ENTITY Softw "http://www.owl-ontologies.com/Software.owl#" >
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
  <!ENTITY protege "http://protege.stanford.edu/plugins/owl/protege#" >
  <!ENTITY LicenOnto "http://www.owl-ontologies.com/LicenseOntology.owl#" >
]>
<rdf:RDF xmlns="http://www.owl-ontologies.com/Mozilla.owl#"
  xml:base="http://www.owl-ontologies.com/Mozilla.owl"
  xmlns:Softw="http://www.owl-ontologies.com/Software.owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
  xmlns:LicenOnto="http://www.owl-ontologies.com/LicenseOntology.owl#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://www.owl-ontologies.com/Software.owl"/>
  </owl:Ontology>
```

```

<LicenOnto:Product rdf:ID="FireFox">
  <SoftOnto:isExecutedIn rdf:resource="&SoftOnto;independentPlatform"/>
  <SoftOnto:isDevelopedUsing rdf:resource="&SoftOnto;PHP"/>
  <SoftOnto:isDevelopedUsing rdf:resource="&SoftOnto;Perl"/>
  <LicenOnto:isAllowedTo rdf:resource="&SoftOnto;distributionAllowed"/>
  <LicenOnto:isAllowedTo rdf:resource="&SoftOnto;downloadAllowed"/>
  <LicenOnto:isAllowedTo rdf:resource="&SoftOnto;transferAllowed"/>
  <LicenOnto:isReplacedAs rdf:resource="&SoftOnto;replacementAllowed"/>
  <LicenOnto:hasA rdf:resource="#MozillaLicense"/>
</LicenOnto:Product>

<LicenOnto:DistributableProduct rdf:ID="FireFoxDistribution">
  <LicenOnto:isAllowedTo rdf:resource="&SoftOnto;distributionAllowed"/>
</LicenOnto:DistributableProduct>

<LicenOnto:DownloadableProduct rdf:ID="FireFoxDownload">
  <LicenOnto:isAllowedTo rdf:resource="&SoftOnto;downloadAllowed"/>
</LicenOnto:DownloadableProduct>

<LicenOnto:ProductTradeMark rdf:ID="FireFoxTradeMark">
  <LicenOnto:isUnmodifiableBy rdf:resource="&SoftOnto;developer"/>
</LicenOnto:ProductTradeMark>

<LicenOnto:TransferableProduct rdf:ID="FireFoxTransfer">
  <LicenOnto:isAllowedTo rdf:resource="&SoftOnto;transferAllowed"/>
</LicenOnto:TransferableProduct>

<SoftOnto:ProductUpdate rdf:ID="FireFoxUpdates">
  <SoftOnto:isExecutedIn rdf:resource="&SoftOnto;independentPlatform"/>
  <SoftOnto:isDevelopedUsing rdf:resource="&SoftOnto;PHP"/>
  <SoftOnto:isDevelopedUsing rdf:resource="&SoftOnto;XML"/>
  <LicenOnto:isAllowedTo rdf:resource="&SoftOnto;distributionAllowed"/>
  <LicenOnto:isAllowedTo rdf:resource="&SoftOnto;downloadAllowed"/>
  <LicenOnto:isAllowedTo rdf:resource="&SoftOnto;transferAllowed"/>
  <LicenOnto:isReplacedAs rdf:resource="&SoftOnto;replacementAllowed"/>
  <LicenOnto:hasA rdf:resource="#MozillaLicense"/>
</SoftOnto:ProductUpdate>

<LicenOnto:AgreementItems rdf:ID="MozillaAgreement">

```

```

    <LicenOnto:isUnliableTo rdf:resource="&SoftOnto;penalty"/>
  </LicenOnto:AgreementItems>
  <LicenOnto:License rdf:ID="MozillaLicense">
    <SoftOnto:hasPermissionOf rdf:resource="&SoftOnto;modification"/>
    <SoftOnto:hasPermissionOf rdf:resource="&SoftOnto;backupCopyAllowed"/>
    <SoftOnto:hasPermissionOf rdf:resource="&SoftOnto;creatingSubLicense"/>
    <SoftOnto:isUnchangeableBy rdf:resource="&SoftOnto;creatingSubLicense"/>
    <LicenOnto:hasValidityOf rdf:resource="#productTradeMark"/>
    <LicenOnto:hasValidityOf rdf:resource="&SoftOnto;downloadAllowed"/>
    <LicenOnto:hasValidityOf rdf:resource="&SoftOnto;distributionAllowed"/>
    <LicenOnto:hasValidityOf rdf:resource="&SoftOnto;transferAllowed"/>
    <LicenOnto:hasValidityOf rdf:resource="&SoftOnto;replacementAllowed"/>
    <LicenOnto:isForA rdf:resource="#ThunderBirdUpdates"/>
    <LicenOnto:isForA rdf:resource="#ThunderBird"/>
    <LicenOnto:isForA rdf:resource="#FireFox"/>
    <LicenOnto:isForA rdf:resource="#FireFoxUpdates"/>
    <LicenOnto:hasItems rdf:resource="#MozillaAgreement"/>
  </LicenOnto:License>
  <LicenOnto:Product rdf:ID="MozillaModifiedProduct"/>
  <LicenOnto:ProductTradeMark rdf:ID="productTradeMark"/>
  <rdf:Description rdf:about="&SoftOnto;copyRight">
    <LicenOnto:isSameAs rdf:resource="&SoftOnto;intellectualPropertyRight"/>
  </rdf:Description>
  <rdf:Description rdf:about="&SoftOnto;creatingSubLicense">
    <SoftOnto:hasInformationOf rdf:resource="&SoftOnto;distributionAllowed"/>
    <SoftOnto:hasInformationOf rdf:resource="&SoftOnto;modification"/>
    <SoftOnto:hasInformationOf rdf:resource="&SoftOnto;patents"/>
    <SoftOnto:hasInformationOf rdf:resource="#MozillaModifiedProduct"/>
    <SoftOnto:isProvidedBy rdf:resource="&SoftOnto;developer"/>
    <SoftOnto:is LIABLETo rdf:resource="&SoftOnto;penalty"/>
  </rdf:Description>
  <rdf:Description rdf:about="&SoftOnto;developer">

```

```

<SoftOnto:isUnableToRestrict rdf:resource="&SoftOnto;modification"/>
<SoftOnto:isAllowedFor rdf:resource="&SoftOnto;thirdpartyTradeMarkAllowed"/>
<SoftOnto:isInPossessionOf rdf:resource="&SoftOnto;modification"/>
<SoftOnto:isInPossessionOf rdf:resource="&SoftOnto;thirdpartyTradeMarkAllowed"/>
<SoftOnto:isInPossessionOf rdf:resource="&SoftOnto;creatingSubLicense"/>
<SoftOnto:isInPossessionOf rdf:resource="&SoftOnto;patents"/>
<SoftOnto:hasTermsDescribedIn rdf:resource="&SoftOnto;creatingSubLicense"/>
<LicenOnto:isAllowedTo rdf:resource="&SoftOnto;downloadAllowed"/>
<LicenOnto:isAllowedTo rdf:resource="&SoftOnto;transferAllowed"/>
<LicenOnto:isAllowedTo rdf:resource="&SoftOnto;distributionAllowed"/>
<LicenOnto:isAllowedFor rdf:resource="&SoftOnto;thirdpartyTradeMarkAllowed"/>
<LicenOnto:isSubjectTo rdf:resource="&SoftOnto;penalty"/>
</rdf:Description>
<rdf:Description rdf:about="&SoftOnto;downloadAllowed">
  <LicenOnto:isAvailableAs rdf:resource="#MozillaAgreement"/>
</rdf:Description>
<rdf:Description rdf:about="&SoftOnto;endUser">
  <SoftOnto:isGrantedFor rdf:resource="&SoftOnto;backupCopyAllowed"/>
  <LicenOnto:isAllowedTo rdf:resource="&SoftOnto;transferAllowed"/>
  <LicenOnto:isAllowedTo rdf:resource="&SoftOnto;thirdpartyDistributionAllowed"/>
  <LicenOnto:isAllowedTo rdf:resource="&SoftOnto;downloadAllowed"/>
  <LicenOnto:isAllowedTo rdf:resource="&SoftOnto;distributionAllowed"/>
  <LicenOnto:isSubjectTo rdf:resource="&SoftOnto;penalty"/>
</rdf:Description>
<rdf:Description rdf:about="&SoftOnto;intellectualPropertyRight">
  <LicenOnto:isSimilarAs rdf:resource="&SoftOnto;copyRight"/>
  <LicenOnto:hasRightsFor rdf:resource="&SoftOnto;developer"/>
  <LicenOnto:hasRightsFor rdf:resource="&SoftOnto;endUser"/>
</rdf:Description>
<rdf:Property rdf:about="&SoftOnto;isForA"/>
<rdf:Description rdf:about="&SoftOnto;modification">
  <SoftOnto:isDistributedUnderTermsOf rdf:resource="#MozillaLicense"/>

```



```

    <SoftOnto:isProvidedBy rdf:resource="&SoftOnto;developer"/>
    <SoftOnto:isDerivedFrom rdf:resource="&SoftOnto;sourceCodeAvailable"/>
    <SoftOnto:isAllowedOn rdf:resource="&SoftOnto;thirdpartyDistributionAllowed"/>
    <SoftOnto:isLiableTo rdf:resource="&SoftOnto;penalty"/>
</rdf:Description>
<rdf:Description rdf:about="&SoftOnto;patents">
    <LicenOnto:hasRoyaltyFree rdf:resource="&SoftOnto;transferAllowed"/>
</rdf:Description>
<rdf:Description rdf:about="&SoftOnto;transferAllowed">
    <LicenOnto:isAvailableAs rdf:resource="#MozillaAgreement"/>
</rdf:Description>
<LicenOnto:Product rdf:ID="ThunderBird">
    <SoftOnto:isExecutedIn rdf:resource="&SoftOnto;independentPlatform"/>
    <SoftOnto:isDevelopedUsing rdf:resource="&SoftOnto;Perl"/>
    <SoftOnto:isDevelopedUsing rdf:resource="&SoftOnto;PHP"/>
    <LicenOnto:isAllowedTo rdf:resource="&SoftOnto;distributionAllowed"/>
    <LicenOnto:isAllowedTo rdf:resource="&SoftOnto;downloadAllowed"/>
    <LicenOnto:isAllowedTo rdf:resource="&SoftOnto;transferAllowed"/>
    <LicenOnto:isReplacedAs rdf:resource="&SoftOnto;replacementAllowed"/>
    <LicenOnto:hasA rdf:resource="#MozillaLicense"/>
</LicenOnto:Product>
<LicenOnto:DistributableProduct rdf:ID="ThunderBirdDistribution">
    <LicenOnto:isAllowedTo rdf:resource="&SoftOnto;distributionAllowed"/>
</LicenOnto:DistributableProduct>
<LicenOnto:DownloadableProduct rdf:ID="ThunderBirdDownload">
    <LicenOnto:isAllowedTo rdf:resource="&SoftOnto;downloadAllowed"/>
</LicenOnto:DownloadableProduct>
<LicenOnto:ProductTradeMark rdf:ID="ThunderBirdTradeMark">
    <LicenOnto:isUnmodifiableBy rdf:resource="&SoftOnto;developer"/>
</LicenOnto:ProductTradeMark>
<LicenOnto:TransferableProduct rdf:ID="ThunderBirdTransfer">
    <LicenOnto:isAllowedTo rdf:resource="&SoftOnto;transferAllowed"/>
</LicenOnto:TransferableProduct>

```

```
<SoftOnto:ProductUpdate rdf:ID="ThunderBirdUpdates">
  <SoftOnto:isExecutedIn rdf:resource="&SoftOnto;independentPlatform"/>
  <SoftOnto:isDevelopedUsing rdf:resource="&SoftOnto;PHP"/>
  <SoftOnto:isDevelopedUsing rdf:resource="&SoftOnto;XML"/>
  <LicenOnto:isAllowedTo rdf:resource="&SoftOnto;distributionAllowed"/>
  <LicenOnto:isAllowedTo rdf:resource="&SoftOnto;downloadAllowed"/>
  <LicenOnto:isAllowedTo rdf:resource="&SoftOnto;transferAllowed"/>
  <LicenOnto:isReplacedAs rdf:resource="&SoftOnto;replacementAllowed"/>
  <LicenOnto:hasA rdf:resource="#MozillaLicense"/>
</SoftOnto:ProductUpdate>
</rdf:RDF>
```

Appendix F

RDF Creative Commons Licensing Ontology

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
  <!ENTITY SoftOnto "http://www.owl-ontologies.com/Software.owl#" >
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
  <!ENTITY LicenOnto "http://www.owl-ontologies.com/LicenseOntology.owl#" >
]>
<rdf:RDF xmlns="http://www.owl-ontologies.com/CreativeCommons.owl#"
  xml:base="http://www.owl-ontologies.com/CreativeCommons.owl"
  xmlns:SoftOnto="http://www.owl-ontologies.com/Software.owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:LicenOnto="http://www.owl-ontologies.com/LicenseOntology.owl#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://www.owl-ontologies.com/Software.owl"/>
  </owl:Ontology>
  <owl:ObjectProperty rdf:ID="by">
```

```

<rdfs:domain rdf:resource="&LicenOnto;License"/>
<rdfs:range rdf:resource="&LicenOnto;AgreementItems"/>
<rdfs:subPropertyOf rdf:resource="#cc"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="cc"/>
<LicenOnto:License rdf:ID="CCLicense"/>
<owl:Class rdf:ID="CommercializationAgreement">
  <rdfs:subClassOf rdf:resource="&LicenOnto;CoreAgreement"/>
</owl:Class>
<CommercializationAgreement rdf:ID="commercializationAllowed"/>
<CommercializationAgreement rdf:ID="commercializationNotAllowed"/>
<owl:ObjectProperty rdf:ID="nc">
  <rdfs:domain rdf:resource="&LicenOnto;License"/>
  <rdfs:range rdf:resource="#CommercializationAgreement"/>
  <rdfs:subPropertyOf rdf:resource="#cc"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="nd">
  <rdfs:domain rdf:resource="&LicenOnto;License"/>
  <rdfs:range rdf:resource="&LicenOnto;DistributionAgreement"/>
  <rdfs:subPropertyOf rdf:resource="#cc"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="sa">
  <rdfs:domain rdf:resource="&LicenOnto;License"/>
  <rdfs:range rdf:resource="&SoftOnto;ModifiedSourceCode"/>
  <rdfs:subPropertyOf rdf:resource="#cc"/>
</owl:ObjectProperty>
</rdf:RDF>

```

Appendix G

RDF eyeOS Web License Ontology

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
  <!ENTITY SoftOnto "http://www.owl-ontologies.com/Software.owl#" >
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
  <!ENTITY protege "http://protege.stanford.edu/plugins/owl/protege#" >
  <!ENTITY LicenOnto "http://www.owl-ontologies.com/LicenseOntology.owl#" >
  <!ENTITY CC "http://www.owl-ontologies.com/CreativeCommons.owl#" >
]>
<rdf:RDF xmlns="http://www.owl-ontologies.com/eyeOSApplication.owl#"
  xml:base="http://www.owl-ontologies.com/eyeOSApplication.owl"
  xmlns:CC="http://www.owl-ontologies.com/CreativeCommons.owl#"
  xmlns:SoftOnto="http://www.owl-ontologies.com/Software.owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:LicenOnto="http://www.owl-ontologies.com/LicenseOntology.owl#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://www.owl-ontologies.com/CreativeCommons.owl"/>
```

```
</owl:Ontology>
<rdf:Description rdf:about="&CC;CCLicense">
  <CC:by rdf:resource="#eyeOSWebAgreement"/>
  <CC:nd rdf:resource="&SoftOnto;distributionNotAllowed"/>
  <CC:sa rdf:resource="&SoftOnto;modificationNotAllowed"/>
  <CC:nc rdf:resource="&CC;commercializationNotAllowed"/>
</rdf:Description>
<LicenOnto:Product rdf:ID="eyeOSWeb"/>
<LicenOnto:AgreementItems rdf:ID="eyeOSWebAgreement"/>
<LicenOnto:DistributableProduct rdf:ID="eyeOSWebDistribution">
  <LicenOnto:isAllowedTo rdf:resource="&SoftOnto;distributionNotAllowed"/>
</LicenOnto:DistributableProduct>
<LicenOnto:DownloadableProduct rdf:ID="eyeOSWebDownload">
  <LicenOnto:isAllowedTo rdf:resource="&SoftOnto;downloadNotAllowed"/>
</LicenOnto:DownloadableProduct>
<LicenOnto:ProductTradeMark rdf:ID="eyeOSWebTradeMark"/>
<LicenOnto:TransferableProduct rdf:ID="eyeOSWebTransfer">
  <LicenOnto:isAllowedTo rdf:resource="&SoftOnto;transferNotAllowed"/>
</LicenOnto:TransferableProduct>
<SoftOnto:ProductUpdate rdf:ID="eyeOSWebUpdates"/>
</rdf:RDF>
```

Bibliography

- Adida, B. and Birbeck, M. (2008). Rdfa primer bridging the human and data webs.
- Allemang, D. and Hendler, J. (2008). *Semantic Web for the Working Ontologist, Effective Modeling in RDFS and OWL*. Morgan Kaufmann. ISBN-13: 978-0-12-373556-0.
- Angele, J., Boley, H., Bruijn, J., Fensel, D., Hitzler, P., Kifer, M., Krummenacher, R., Lausen, H., Polleres, A., and Studer, R. (2005). Web rule language (wrl).
- Antoniou, G. and van Harmelen, F. (2003). Web ontology language: Owl. In Staab, S. and Studer, R., editors, *Handbook on Ontologies in Information Systems*, pages 76–92. <http://www.few.vu.nl/frankh/abstracts/OntoHandbook03OWL.html>.
- Asfandeyar, M., Anjomshoaa, A., Edgar, R., and A, M. T. (2009). Blending the sketched use case scenario with license agreement using semantics. In *Knowledge Science, Engineering and Management*, volume 5914, pages 275–284. Springer.
- Asfandeyar, M., Anjomshoaa, A., Edgar, R., and A, M. T. (2010). Exploiting ontology for software license agreements. *International Journal of Software and Informatics (IJSI)*, 4:89–100.
- Bak, J., Jedrzejek, C., and Falkowski, M. (2010). Application of an ontology-based and rule-based model to selected economic crimes: fraudulent disbursement and money laundering. In *RuleML'10: Proceedings of the 2010 international conference on Semantic Web rules*, volume 6403 of *Lecture Notes in Computer Science*, pages 210–224, Washington, DC, USA. Springer.
- Berners, L., Hendler, J., and Lassila, O. (2001). The semantic web.
- Carroll, J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., and Wilkinson, K. (2004). Jena: Implementating the semantic web recommendations. In *Proceedings of the 13th International World Wide Web Conference on Alternate track Papers and Posters*, pages 74–83. HP Labs, Bristol, UK, ACM. ISBN: 4-58113-912-8.

- Chandrasekaran, B., Josephson, J., and Benjamins, V. (1999). What are ontologies, and why do we need them? *Intelligent Systems and their Applications*, 14(1):20. Digital Object Identifier: 10.1109/5254.747902.
- Daconta, M., Obrst, L., and Smith, K. (2003). *The Semantic Web; A Guide to the Future of XML, Web Services, and Knowledge Management*. Wiley Publishing. ISBN 0-471-43257-1.
- Davy, S., Barrett, K., Serrano, M., Strassner, J., Jennings, B., and Meer, S. (2007). Policy interactions and mangement of traffic engineering services based on ontologies. In *Latin American Network Operations and Management Symposium, LANOMS*, pages 95 – 105. IEEE. ISBN: 978-1-4244-1182-5.
- Flahive, A., Rahayu, W., Taniar, D., Apduhan, B. O., Wouters, C., and Dillon, T. (2007). A service oriented architecture for extracting and extending sub-ontologies in the semantic grid. In *International Conference on Advanced Information Networking and Applications, AINA*, pages 831–838. Digital Object Identifier: 10.1109/AINA.2007.21.
- Flahive, A., Taniar, D., Rahayu, W., and Apduhan, B. O. (2009). Ontology tailoring in the semantic grid. *Computer Standards & Interfaces, Elsevier*, 31:870 – 885. Journal Homepage: www.elsevire.com/locate/csi.
- Fujimura, N. and Nagaishi, K. (2007). Implementation of software license management support system. In *Proceedings of the 35th annual ACM SIGUCCS conference on User services*, pages 122–124. ISBN:978-1-59593-634-9.
- García, R. and Tummarello, G. (2006). Semantic digital rights management for controlled p2p rdf metadata diffusion. In *2nd Semantic Web Policy Workshop*.
- Gruber, T. (1993). International journal human computer studies. In *International Workshop on Formal Ontology*, pages 907–928, Padova, Italy. <http://tomgruber.org/writing/onto-design.pdf>.
- Hai, D., Hussain, F., and Chang, E. (2007). Application of protege and sparql in the field of project knowledge management. In *Second International Conference on Systems and Networks Communicaitons*, pages 74 – 79, Cap Esterel. IEEE, IEEE. ISBN: 0-7695-2938-0.
- Hogeboom, M., Fuhua, L., Esmahi, L., and Chunsheng, Y. (2005). Constructing knowledge bases for e-learning using protege 2000 and web services. In *International conference on Advanced Information Networking and Applicaitons*, volume 1, pages 215 – 220. IEEE, IEEE. ISBN: 0-7695-2249-1.

- Horrocks, I., Perter, F. P. S., Boley, H., Tabet, S., Grosz, B., and Dean, M. (2004). Swrl: A semantic web rule language combining owl and ruleml.
- Huang, V. and Javed, M. (2008). Semantic sensor information description and processing. In *Second International conference on Sensor Technologies and Applications*, pages 456–461. IEEE. ISBN: 978-0-7695-3330-8.
- Johan, H. (2001). *Creating the Semantic Web with RDF*. Wiley Publishing, United States of America. ISBN: 0-471-40259-1.
- Kagal, L., Berners, T., Connolly, and Weitzner, D. (2006). Using semantic web technologies for policy management on the web. In *21st National Conference on Artificial Intelligence (AAAI)*.
- Kang, J., Lee, K., Park, H., Lim, M., Kang, E., Im, Y., Oh, S., and Kang, M. (2009). A semantic drm contents in social awareness using ontology. In *International Conference on Hybrid Information Technology*, pages 508–511, Daejeon, Korea. ISBN:978-1-60558-662-5.
- Matthew, F. B. (2006). Online shopping for technology products: Designing web portals to address buyer behaviours, attitudes and individual differences. In *Proceedings of 2006 conference of the Center for Advanced Studies on Collaborative Research (CASCON '06)*, pages 1–4, New York, USA. IBM Canada, ACM.
- McBride, B. (2002). Jena: A semantic web toolkit. *IEEE Internet Computing*, 6:55–59. ISSN: 1089-7801.
- Melton, J. (2006). What’s wrong with this picture?
- Michael, K. B. (1992). Emanuel goldberg, electronic document retrieval, and vannevar bush’s memex. *Journal of the American society for Information Science*, 43(4):284–294. <http://people.ischool.berkeley.edu/buckland/goldbush.html>.
- Miller, L., Seaborne, A., and A., R. (2002). Three implementations of squish ql, a simple rdf query language. In Horrocks, I. and Handler, J., editors, *Lecture Notes in Computer Science*, volume 2342, pages 423–435, Berlin Heidelberg. HP Laboratories Bristol, Springer. ISBN: 3-540-43760-6.
- Muthaiyah, S. and Kerschberg, L. (2006). Dynamic integration and semantic security policy ontology mapping for semantic web services (sws). In *1st International Conference on Digital Information Management*. IEEE. ISBN: 1-4244-0682-X.
- Nadah, N., Rosnay, M., and Bachimont, B. (2007). Licensing digital content with a generic ontology escaping from the jungle of rights expression languages. In *International Conference on Hybrid Information Technology*, pages 65–69. ACM. ISBN:978-1-59593-680-6.

- Natalya, F. N. and Deborah, L. M. (2001). Ontology development 101: A guide to creating your first ontology.
- Nordquist, P., Petersen, A., and Todorova, A. (2003). License tracing in free open and proprietary software. *Journal of Computing Sciences in Colleges*, 19:101–112. ISSN:1937-4771.
- Nyre, A. A., Bernsmed, K., Solvar, B., and Pedersen, S. (2011). A server-side approach to privacy policy matching. In *First International Workshop on Privacy by design (PBD), ARES*.
- Obrst, L., Wray, E., and Liu, H. (2001). Ontological engineering for b2b ecommerce. In *International Conference on Formal Ontology in Information Systems*, pages 117–126. ISBN:1-58113-377-4.
- Oren, E., Delbru, R., Gerke, S., Haller, A., and Decker, S. (2007). Active rdf: Object oriented semantic web programming. In *The 16th International Conference on World Wide Web*, pages 817–824, New York, USA. ACM. ISBN: 978-1-59593-654-7.
- Piedra, N., Chicaiza, J., Lopez, J., Tovar, E., and Martinez, O. (2009). Open educational practices and resources based on social software utpl experience. In *Euro American Conference on Telematics and Information Systems: New Opportunities to increase Digital Citizenship*.
- Schadbot, N., Hall, W., and Berners, L. (2006). The semantic web revisited. *Intelligent Systems IEEE*, 21:96–101. ISSN: 154-1672.
- Seaborne, A. (2002). An rdf netapi. In *Lecture Notes in Computer Science*, volume 2342, pages 399–403, Berlin Heidelberg. HP Laboratory Bristol, Springer. ISBN:3-540-43760-6.
- Thomas, R. (1993). A translation approach to protable ontology specifications.
- Uchibayashi, T., Apduhan, B. O., Shiratori, N., and Taniar, D. (2009). A visual view of sub-ontology derivation as an end-user tool. In *International Conference on Ubiquitous Information Technologies & Applications, ICUT*, pages 1 – 6. IEEE. Digital Object identifier 10.1109/ICUT.2009.5405673.
- Uszok, A., Bradshaw, J., Johnson, M., Jeffers, R., Tate, A., Dalton, J., and Aitken, S. (2004). Kaos policy management for semantic web services. *Intelligent Systems IEEE*, 19(4):32–41. ISSN:1541-1672.
- Veijalaine, J., Nikitin, S., and Törmälä, V. (2006). Ontology based semantic web service platform in mobile environments. In *Proceedings of the 7th International conference on Mobile Data Management*. IEEE.

- Wang, J., Zuo, W., He, F., and Wang, Y. (2009). A new formal description of ontology definition and ontology algebra. *Second International Symposium on Knowledge Acquisition and Modeling*, 1:363–366.
- Wielinga, B., Schreiber, A., Wielemaker, J., and Sandberg, J. (2001). From thesaurus to ontology. In *International Conference On Knowledge Capture*, pages 194–201. ISBN:1-58113-380-4.
- Wouters, C. (2005). *A formalization and Application of Ontology Extraction*. PhD thesis, La Trobe University, Melbourne, Australia.

Curriculum Vitae

Personal information	
Name	Muhammad Asfand-e-yar
Address	Josef Baumann Gasse 8A/202, 1220, Vienna, Austria
E-Mails	asfand279@hotmail.com, asfandeyar@ifs.tuwien.ac.at
Gender	Male
Worked as an External Reviewer	External reviewer of research papers in “International Conference on Information Integration and Web-based Applications and Services” (IIWAS) 2009.
Work experience	2 Years
Dates	Aug 2004 - Aug 2006
Occupation	Lecturer in University
Main activities and responsibilities	<ol style="list-style-type: none">1. Delivering Lecture<ul style="list-style-type: none">• Network Management and Network Programming• Data Structure• Object Oriented Programming2. Helping out in Network Administration<ul style="list-style-type: none">• Installation & Troubleshooting• Creating & Managing groups• Permission to users and groups
Name of organization	NU-FAST University, Pakistan

Education and Training	
Dates	Sep 2001 - Feb 2004
Title of qualification	Master's
Principal subjects	Computer Science
Name of organization	International Islamic University, Islamabad, Pakistan
Level in international classification	Master's
Dates	Sep 1999 - Aug 2001
Title of qualification	Bachelor's
Principal subjects	Computer Science
Name of Organization	International Islamic University, Islamabad, Pakistan
Level in international classification	Bachelor's
Date	Sep 1995 - May 1997
Title of qualification	College
Principal subjects	Pre-Medical
Name of organization	Army Public College, Peshawar, Pakistan
Level in international classification	Higher Secondary School Certificate
Research publications	
	<ol style="list-style-type: none"> 1. Exploiting Ontology for Software License Agreements, Muhammad Asfandeyar, Amin Anjomshoaa, Edgar R. Weippl, A Min Tjoa, <i>International Journal of Software and Informatics (IJSI)</i>, 2010, Vol. 4(1). 2. Towards an Ontology Based Solution for Managing License Agreement Using Semantic Desktop, Mansoor Ahmed, Amin Anjomshoaa, Muhammad Asfandeyar, A. Min Tjoa, <i>International Conference on Availability Reliability and Security, ARES 2010</i>, ISBN: 978-0-7695-3965-2. 3. Blending the Sketched Use Case Scenario with License Agreements Using Semantics, M.Asfandeyar, Amin Anjomshoaa, Edgar R. Weippl, A Min Tjoa, <i>Proceedings of the 3rd International Conference on Knowledge Science, Engineering and Management, KSEM, Springer LNAI, 2009</i>, ISBN 978-3-642-10487-9. 4. Secure route path Formation in Ad hoc On-demand Distance vector Routing Protocol. Muhammad Asfandeyar, Muhammad Sher, <i>Information Technology Journal (ITJ)</i>, 2004, Vol. 3(2).