Automation of legal reasoning and decision based on ontologies

Mirna El Ghosh

Normandie Université

THESE

Pour obtenir le diplôme de doctorat

Spécialité INFORMATIQUE

Préparée au sein de « INSA de ROUEN »

AUTOMATISATION DU RAISONNEMENT ET DECISION JURIDIQUES BASÉS SUR LES ONTOLOGIES

Présentée et soutenue par
Mirna El Ghosh

Thèse soutenue publiquement le (24 septembre 2018) devant le jury composé de		
M. Enrico FRANCESCONI	Prof / Chercheur / TTIG-CNR, the Legal Information Institute of the National Research Council of Italy	Rapporteur
Mme Zahia GUESSOUM	MCF / HDR / Laboratoire d'Informatique de Paris 6 (LIP6)	Rapporteur
Mme Maroua BOUZID	Prof / Professeur / Université de Caen-Basse Normandie, Membre du GREYC UMR6072	Examinateur
Mme Cecilia ZANNI-MERK	Prof / Professeur / INSA de ROUEN, LITIS	Examinateur
M. Habib ABDULRAB	Prof / Professeur / INSA de ROUEN, LITIS	Directeur de thèse
M. Mohamad KHALIL	Prof / Professeur / Université Libanaise, Faculté de Génie	Codirecteur de thèse
Mme Hala NAJA	Prof / Professeur / Université Libanaise, Faculté des Sciences	Co-encadrante de thèse

Thèse dirigée par Habib ABDULRAB, INSA du Rouen, laboratoire LITIS et Mohamad KHALIL, Université Libanaise, Faculté de génie.

INSA ROUEN NORMANDIE

MI ΙΣ

litis

To Julie

Acknowledgements

First, I gratefully acknowledge INSA de Rouen and LITIS laboratory for making it possible for me to carryout my research.

My deepest gratitude goes to my thesis supervisor prof. Habib ABDULRAB for his invaluable advice, constant support and trust in me and without whom this work would never have been done. He provided me with many useful comments and suggestions for the preparation of this thesis. He also deserves my gratitude for providing me the opportunities to attend several conferences, workshops and meetings that have widen my research network.

I extend my gratitude to my thesis co-supervisor prof. Hala NAJA for her insightful guidance, continuous effort and for her repeated reading of the work and offering comments that have led to many improvements over the years. I am grateful too for my thesis second co-supervisor prof. Mohamad KHALIL for his continuous support and encouragement during my research work.

Besides my supervisors, I would like to thank the rest of my thesis committee: prof. Enrico FRANCESCONI, prof. Zahia GUESSOUM, prof. Maroua BOUZID and prof. Cecilia ZANNI-MERK, for their insightful comments and encouragement, but also for the hard question which incented me to widen my research from various perspectives. In particular, I am grateful for the reporters of this PhD for accepting the extra amount of work, for examining my PhD thesis and providing many helpful suggestions.

Finally, i would like to thank my family: my husband, my parents, my sisters and brothers for supporting me spiritually throughout my thesis and my life.

This work has been supported by the project CLASSE2 (Corridors Logistiques, Application à la Vallée de la Seine et à Son Environnement), CNRS Lebanon, Lebanese university and LaseR.

Computer Science is no more about computers than astronomy is about telescopes. - EW Dijkstra, 1970

Abstract

This thesis analyses the problem of building well-founded domain ontologies for reasoning and decision support purposes. Specifically, it discusses the building of legal ontologies for rule-based reasoning.

In fact, building well-founded legal domain ontologies is considered as a difficult and complex process due to the complexity of the legal domain and the lack of methodologies. For this purpose, a novel middle-out approach called MIROCL is proposed. MIROCL tends to enhance the building process of well-founded domain ontologies by incorporating several support processes such as reuse, modularization, integration and learning.

MIROCL is a novel modular middle-out approach for building well-founded domain ontologies. By applying the modularization process, a multi-layered modular architecture of the ontology is outlined. Thus, the intended ontology will be composed of four modules located at different abstraction levels. These modules are, from the most abstract to the most specific, UOM(Upper Ontology Module), COM(Core Ontology Module), DOM(Domain Ontology Module) and DSOM(Domain-Specific Ontology Module).

The middle-out strategy is composed of two complementary strategies: top-down and bottom-up. The top-down tends to apply ODCM (Ontology-Driven Conceptual Modeling) and ontology reuse starting from the most abstract categories for building UOM and COM. Meanwhile, the bottom-up starts from textual resources, by applying ontology learning process, in order to extract the most specific categories for building DOM and DSOM. After building the different modules, an integration process is performed for composing the whole ontology.

The MIROCL approach is applied in the criminal domain for modeling legal norms. A well-founded legal domain ontology called CriMOnto (Criminal Modular Ontology) is obtained. Therefore, CriMOnto has been used for modeling the procedural aspect of the legal norms by the integration with a logic rule language (SWRL). Finally, an hybrid approach is applied for building a rule-based system called CORBS. This system is grounded on CriMOnto and the set of formalized rules.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

1.1 Overview

This thesis is the result of a research work into legal knowledge conceptualization, representation and reasoning. More precisely, developing well-founded domain ontologies for building (rule-based) legal reasoning models.

A lesson learned over decades of research designing rule-based legal reasoning systems is the need for an ontology to organize the concepts and manage their interactions (Ashley, 2011). Accordingly, this thesis attempts to answer the question: "For building rule-based legal reasoning systems, what kinds of ontologies should be used and what kinds of methodologies are available for building them?".

For answering the question, the thesis discusses mainly the legal ontologies as a type of knowledge representation and formalization of legal knowledge and explores the issues related to the ontology engineering domain such as knowledge extraction, modeling methodologies and tools for the construction of well-founded ontologies. Additionally, the thesis addresses the role of well-founded legal ontologies for building rule-based legal reasoning models in order to perform decision support purposes.

Thus the work arises from two strands of research: (1) the research into legal ontologies and (2) the research into legal reasoning models. Therefore, the intersection of these two strands permits the development of legal knowledge-based systems (LKBS) that perform legal reasoning grounded on legal domain ontologies.

This chapter provides the general research lines and technical aspects of the thesis.

1.2 Thesis Context

1.2.1 Interest of AI in Law

The interest from the AI community for the formalization of legal information and knowledge for computer processing is not thus recent (Casellas, 2008a). Since the 1970s there is a great amount of interest in the development of legal expert systems around the world (Greenleaf, 1989). Legal expert systems, also called "Legal Knowledge Based Systems (LKBS)" (Susskind, 1986), are considered as attempts to develop programs aiming to solve legal problems. This because the law is considered as an attractive domain for AI research since much legal knowledge is readily accessible and relatively well structured, codified and indexed (Gardner, 1987). Moreover, Law is a system of readily understood rules with procedures for interpreting these rules. These skills are of interest to computer science researchers wishing to examine the way in which humans think, in order to replicate some features of human reasoning (Zeleznikow et al., 1992).

Although, the International Association for Artificial Intelligence and Law (IAAIL) was founded on 1987 and the first meeting of the ICAIL conference took place in Boston on that same year and the Foundation for Legal Knowledge Based Systems (JURIX) has held annual international conferences on legal knowledge and information systems since 1988 (Casellas, 2008a). This, in turn, led to the foundation of the Artificial Intelligence and Law Journal, first published in 1992. Moreover, since 2007 the JURISIN workshops have been held in Japan under the auspices of the Japanese Society for Artificial Intelligence.

Since the 1990s, the development of formalizations of domain conceptualizations, (so-called ontologies), became popular in AI following the work of Gruber (Gruber, 1993) (Gruber, 1995). Early examples in AI and Law include Valente's functional ontology (Valente et al., 1994b) and the frame-based ontology of Visser and van Kralingen (Van Kralingen, 1995), (Visser, 1995). Legal ontologies have since become the subject of regular workshops at AI and Law conferences.

1.2.2 Artificial Intelligence and Law

AI & Law is a subfield of AI research that focuses on designing computer programs, or computational models, that perform or simulate legal reasoning. In other words, AI & Law is the field of modeling computationally the legal reasoning for the purpose of building tools to assist in legal practice (Ashley et al., 2001). Therefore, the development of such computational models of legal reasoning can actually be used for problem-solving systems called legal expert systems or legal knowledge-based systems (LKBS).

The field of AI & Law has always had two distinct motivations: practical and theoretical. The practical side is for building intelligent legal information systems that can assist users in their interactions with legal rules. Meanwhile, the theoretical side is for understanding the process of legal reasoning and legal argumentation using computational models and techniques (McCarthy, 1980).

Thus, this domain is called *Computational Law* (Love et al., 2005) which is an interdisciplinary research field that addresses both the use of strategies for the representation of legal knowledge, as the possibilities for creating automated reasoning systems.

According to Ashley's studies (Ashley et al., 2001), computational models of legal reasoning can be described in terms of the inputs to the program, its outputs and the intervening steps that transform the former into the latter. Thus, they are comprised of a knowledge representation and an inference mechanism (see Figure 1.1).

The knowledge representation aims to capture some important aspects of legal knowledge, as inputs, and to represent it formally in a conceptual hierarchy in

order to be used by computer systems. These techniques has become a primary goal of expert systems research (Gruner, 1986). Such *"Knowledge engineering"* has been successfully undertaken in a number of fields to produce expert systems that perform a variety of important tasks. Knowledge engineering was also the source for the growing interest in the development and use of legal ontologies (Valente, 1995) aiming to improve information retrieval of legal sources (Benjamins et al., 2005a) and to bridge the gap between text and knowledge (Buitelaar et al., 2007).

Meanwhile, the inference mechanisms are algorithms that enable a program to use those input elements of legal knowledge that are represented in order to solve problems (Ashley et al., 2001). For instance, the inference mechanism may take a problem situation and compare it to other cases, draw inferences about how that problem should be decided, and generate arguments, as outputs, using the information in the computational model.

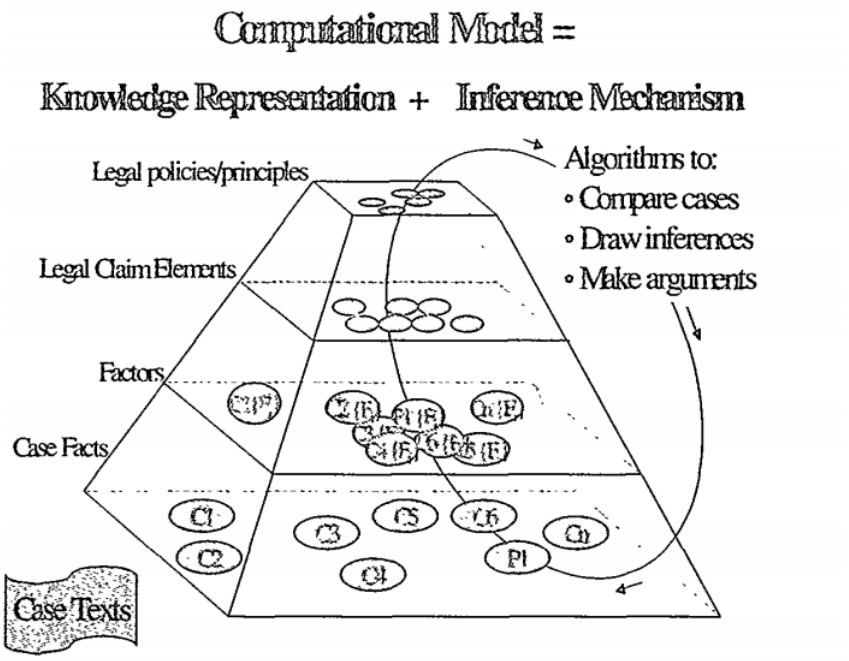


FIGURE 1.1: Ashley's illustration of computational model of legal reasoning (Ashley et al., 2001).

1.2.3 Legal Knowledge Based Systems (LKBS)

Legal professionals, such as judges and lawyers, dedicate a significant amount of their time to finding, reading, analyzing and synthesizing information in order to take decisions, and prepare advice and trials, among other tasks (Benjamins et al., 2005b). For this purpose, developing legal AI systems can support legal professionals in fulfilling their tasks.

Generally, legal AI systems are categorized in legal retrieval and legal analysis systems (Popple, 1990). The legal analysis systems can be judgment machines or legal expert systems, known as Legal Knowledge Based System (LKBS), which are decision support systems.

Rather than aiming at the impossible dream (or nightmare) of building judgment machines, or automatic judges, AI research has aimed at developing LKBS which are practical tools to support judicial activities as well as new analytical tools for understanding and modeling judicial decision-making (Sartor et al., 1998).

Any LKBS must be capable of legal reasoning (Popple, 1990). Thus, the system must be based upon a model of legal reasoning by describing the norms that operate within the legal system (Popple, 1996). Legal reasoning, applied earlier in various approaches for decision making purposes, describes how legal expert system takes legal decisions with the help of rules (Valente et al., 1991). Accordingly, legal reasoning is considered as a rule-guided activity, where most part of it consists of applying legal rules to interpretations of cases (Gardner, 1987), (Breuker, 1990). This kind of reasoning is called rule-based reasoning performed by rule-based expert systems where the reasoning process is based on a set of `if-then` rule statements (Dove, 1996a), used to describe certain patterns of the giving domain such as legal norms, and a process called inference. Inference is the process of deriving conclusions from premises (Zeleznikow et al., 1992).

1.2.4 Role of Ontologies in the Automation of Legal Reasoning

Ontologies provide reusable pieces of declarative knowledge, which can be together with problem-solving methods and reasoning functionality assembled into high-quality technology and application systems in an economical fashion (Neches et al., 1991a) (Guarino, 1998).

In the 1990s, research interest in large scale formalizations and reuse of legal knowledge shifted to ontologies (Bench-Capon et al., 2012). The importance of an ontological approach in the legal domain is introduced first by the work of Valente and Breuker, and later stressed by Bench-Capon and Visser (Bench-Capon et al., 1997), (Bench-Capon et al., 1987) with regard to information management in general and for the legal domain in particular.

In later years, ontologies introduced to AI & Law in works such as (Ven et al., 2008) and (Bench-Capon et al., 1997) have been widely used for modeling, comparing and harmonizing legal knowledge. For instance, the work of (Ven et al., 2008), where ontologies are used for building legal expert systems and for reasoning with description logics.

In this regard, the work of Sartor (Sartor, 2006) is considered as an essential reference for legal knowledge modeling for his contribution to the characterization of legal concepts by providing a formal model of their structure and a logical framework able to deal with the specificity of legal reasoning.

In this context, it is noticeable that the Semantic Web technologies are used earlier for modeling legal knowledge and resources with the work of Boer, van Engers and Winkels's (Boer et al., 2003) paving the way to succeeding works on legal knowledge modeling and acquisition within the same legal Semantic Web framework.

1.2.5 The Semantic Web and Law

The Semantic Web is one of the most important application areas of ontologies (Simperl et al., 2013). In this context, the knowledge components, that is, the ontologies, are formalized using Web-suitable, semantically unambiguous representation languages such as Resource Description Framework (RDF) Schema and Web Ontology Language (OWL) and are pervasively accessible and shared across the Web.

The Semantic Web intends to enable machines to understand -to some extent- what is in the Web; not only to improve human communication, but in particular to delegate more and more "intelligent" tasks to machines. The Semantic Web technology had an important role in the legal domain (Benjamins et al., 2005a). It enables accessing of Web resources by semantic content rather than by keywords, involving the automation of service discovery, acquisition, composition and monitoring (Davies et al., 2004).

The legal domain has several characteristics that make it an interesting application for Semantic Web technology (Benjamins et al., 2005a). For instance, the notion of information retrieval as document retrieval is not always sufficient in the legal domain. Often a particular question requires some deduction or inference before an appropriate answer can be given. In order words, 'question-answering' seems more relevant than 'information retrieval', as regulations may contain many different articles about the same topic and one can only assess whether something is permitted or not by understanding the full documentation. A rather detailed understanding is required, in particular, because regulations generally contain complex structures of exceptions (Benjamins et al., 2005a).

Legal professionals, be they judges or lawyers, handle information in order to take decisions. As such they are vulnerable to the Information Overload phenomenon. Moreover, increasingly more non-legal professionals have to deal with the Law due to increasing regulations in for example environmental protection and public security in buildings (Benjamins et al., 2005a).

Legal Semantic Web technologies typically involve two types of applications:

- Applications in internal corporate settings: CLIME (Boer et al., 2001), e-COURT (information management of criminal court cases (Breuker et al., n.d.) and E-POWER (drafting tax law (Boer et al., 2002b)).
- Applications with public information on the Internet such as JurWordNet (Sagri et al., 2004).

1.3 Problem Statement

This thesis addresses two main problems: (1) difficulty and complexity of building well-founded legal domain ontologies and (2) lack of definition of rule-based legal reasoning models based on domain ontologies for decision support purposes.

1.3.1 Difficulty and Complexity of Well-founded Ontology Building Process

In fact the problem of the difficulty and complexity of building well-founded ontologies is divided into two main sub-problems: (1) the complexity of the legal domain and (2) the lack of methodologies for building well-founded domain ontologies.

1. The complexity of the legal domain: any ontology building process should include the complexity of the relationship between ontology and corpus since the complexity attached to ontologies is mirroring the complexity of the application field (Mazzega et al., 2011). In the legal domain, legal conceptual knowledge is closely related to the use of language (Francesconi et al., 2008). The written text is the most widely used way of communicating legal matters (Lame, 2005). Meanwhile, the implicit knowledge of the natural language is one of the main obstacles to progress in the field of artificial intelligence and law (McCarty, 2007).

   The complexity of legal knowledge is seen from two different perspectives. First, because the language used in legal documents is too complex. Legal rules and standards are written, for the most part, in ordinary language. The language implemented in the writing of the rules and standards contain ambiguities, either by accident or intent (Dove, 1996a). Specifically the problem of open texture property, or in other words, incomplete definition of many legal concepts of the law that is originally addressed by Gardner (Gardner, 1987). For instance, 'reasonable' and 'intentional' are examples of vague concepts that are intentionally or unintentionally arguable in meaning and cannot be modeled in a way analogous to human thinking. What constitutes reasonable behavior will vary from time to time, place to place and person to person (Dove, 1996a). For the other perspective, the legal complexity is defined in

terms of the amount of information that must be collected and processed in order for lawyers or judges to evaluate a case and litigation to proceed (White, 1992).

2. The lack of methodology for building well-founded domain ontologies: well-founded domain ontologies must be represented with the support of a foundational theory not just building them from scratch. Such an approach has still not been broadly adopted (Zamborlini et al., 2008). One of the key problems the ontology engineering community has to face is that most ontologies are built from scratch-rather than reusing existing ones-leading to high engineering efforts and costs (Hartmann et al., 2009). Generally, Building ontologies from scratch is not an easy task. It is considered as a resource-intensive, time consuming and costly task. This is due to the difficulty and complexity of capturing knowledge from legal sources which are mainly unstructured textual documents such as legislations and codes. Additionally, this strategy will not lead to well-founded domain ontologies.

1.3.2 Lack of Definition of Rule-based Legal Reasoning Models based on Domain Ontologies

For decision support purposes, there is a need for building legal reasoning models aiming to resolve giving legal problems. The second problem addressed in this thesis is the lack in defining reasoning models, specifically rule-based, grounded on legal domain ontologies. In other words, there is a need for a methodology that defines the process of building rule-based reasoning models by integrating domain ontologies.

1.4 Limitations of Existent Approaches

Two main limitations are addressed in the existent approaches: (1) limitation in knowledge acquisition and modeling and (2) limitation in knowledge sharing and reasoning.

1. Limitation in knowledge acquisition and modeling: the fundamental assumption for building any expert or knowledge-based system is "Knowledge is Power" (Buchanan et al., 1982). The point is the richer the knowledge-base of a system the higher performance of problem-solving capabilities. Thus, it is now widely recognized that building ontologies is an essential step in the development of knowledge-based systems. Meanwhile, there is lacking in a clear understanding of how to build ontologies. However, there exists a growing number of methodologies that specifically address the issue of the development of ontologies, either from scratch or by reusing existent ones,

such as (Uschold et al., 1995), (Swartout et al., 1997), and (Gruninger et al., 1995) . These methodologies have focused on core ontology development activities such as: requirements analysis, conceptualization, implementation, evaluation and maintenance.

A second generation of methodologies, such as Methontology (Corcho et al., 2005) and OnToKnowledge (Sure et al., 2003) shifted this focus towards a more iterative engineering process in which application-specific requirements are seen as an integral part of the requirements analysis activity.

Actually, by studying these methodologies, we found that most of the approaches focus on building ontologies from scratch due to several reasons (Hartmann et al., 2009). First, ontologies are usually tailored to work for specific applications, restricting its potential reusability. Second, developers usually follow a monolithic approach when developing ontologies, usually covering different domains, restricting the reusability of relevant parts for other applications. Therefore, most of the existing developed domain ontologies are not well-founded. Additionally, there is no focus on the participatory approach in the ontology engineering process. Meanwhile, such approach can effectively simplify and enrich the ontology building process specifically for complex domains such as the legal domain. In fact, there is a need for a truly collaborative effort carried out by different contributors, domain experts and ontology engineers, handling heterogeneous data sources for building well-founded and rich domain ontologies (Simperl et al., 2013).

2. Limitation in knowledge sharing and reasoning: in the literature, most of the legal ontologies are not built for reasoning purposes such as the works of (Hoekstra et al., 2007), (Breuker et al., 2002), and (Casellas, 2008b). In the other hand, some other ontologies are built for legal reasoning and specifically case-based reasoning such as (Henderson et al., 2001), (Zeng et al., 2005), (Wyner, 2008) and (Ashley, 2011). However, developing ontologies for rule-based reasoning is missing.

1.5 Thesis Objectives

The overall objective of the thesis is to build a well-founded legal domain ontology for a rule-based reasoning model.

The concrete objectives of the research are categorized into two main categories: (1)Knowledge acquisition and modeling; (2)Knowledge sharing and reasoning.

1. Knowledge acquisition and modeling: To capture knowledge from several data sources, considered as heterogeneous sources, starting from scratch

(documents of the domain application), existent validated foundational ontologies, and domain experts and turn it into machine processable form. In this category, this thesis aims to:

- propose a novel approach in order to simplify the complexity of building well-founded domain ontologies.
- apply the proposed approach in order to develop a well-founded legal domain ontology for modeling the legal norms of the legal domain. This includes a deep analysis for the legal norms with the support of legal experts.
- enable the procedural aspect of the legal norms to be expressed through the legal domain ontology for building rule-based reasoning model. This requires the application of a modeling process based on a rule language and the legal domain ontology for building a set of logical formal rules.

2. Knowledge sharing and reasoning: To combine semantically enriched information with context to provide actionable meaning, applying inferencing and reasoning for decision support. In this category, this thesis aims to:
   - enable the execution of rules combined with the instances of the legal domain by using inferential mechanisms for reaching new conclusions.
   - develop rule-based reasoning model grounded on the integration of the logical rules and the legal domain ontology.

1.6 Thesis Contributions

As aforementioned, this thesis aimed at fulfilling the main requirements for building a well-founded legal domain ontology for legal decision support purposes. The primary contribution of this research is the exploration of a novel solution to the complex problem of building well-founded domain ontologies for reasoning purposes. Although, this research has produced several other contributions. The main contributions are the following:

- The design of MIROCL: a modular middle-out collaborative approach that supports the following features:
  - Collaborative ontology building process handling heterogeneous data sources as a foundation for decentralized and iterative approach for developing the target ontology. Thus ontology building is performed in a collaborative workspace involving at least a computer scientist and an expert of the (legal) domain. The expert has the theoretical and practical

knowledge of the field and the computer scientist brings methods and tools to represent it.

  - Ontology modularization that divides the target ontology into independent ontology modules serving in simplifying the ontology building process.
  - Ontology-driven conceptual modeling process, as a top-down strategy, that supports the development of well-founded ontologies by reusing existent validated foundational ontologies and using an ontologically well-founded conceptual modeling language (OntoUML).
  - Ontology learning process, as a bottom-up strategy, that extracts semi-automatically semantic objects from textual resources with the support of NLP techniques.
  - Ontology integration that merges the independent developed ontologies into one global ontology.

- The development of a well-founded modular legal domain ontology, named CriMOnto, by applying the MIROCL approach. The purpose of CriMOnto is to model the legal norms of the legal domain, specifically the criminal domain, in order to support rule-based legal decision support purposes. CriMOnto covers the criminal domain by connecting multiple complementary levels of granularity.
- The procedural aspect of the legal norms is expressed through the integration of the legal domain ontology and a logic rule language by applying an ontology-based modeling approach. For the current work, the ontology-based approach is monotonic where the SWRL rule language is selected for formalizing the legal rules. This approach suffers from some limitations specially in modeling complex rules and negation as failure. For future works, it is recommended to apply a non-monotonic approach where logic programming formalisms are applied.
- The design and implementation of a rule-based reasoning model for a decision support system named CORBS grounded on the integration of the legal domain ontology and set of logic rules. For the implementation, the Java-based frameworks such as Jena Semantic Web or libraries such as OWL API can be used.

The reader is invited to note that although the above contributions are made in the context of the legal domain, they can be applied to general ontology applications.

1.7 Structure of the Thesis

The remainder of the thesis is structured as follows:

Chapter 2 provides a general overview of the background of the thesis mainly ontology and knowledge engineering domains. We discuss the ontologies, their definitions, classifications, criteria and components. Furthermore, a series of related studies including the legal ontologies, their role and uses, the ontology engineering methodologies, tools, environments, languages and formalisms are discussed. Then, we discuss the existent approaches in the literature concerning the knowledge engineering in general and in the legal domain as well.

Chapter 3 introduces a novel modular middle-out approach for building well-founded domain ontologies named MIROCL. The Phases of the approach are defined where top-down and bottom-up strategies are applied in a complementary fashion using a diversity of ontology engineering support processes including ontology modularization, reuse, learning and integration.

Chapter 4 discusses the application of MIROCL in the legal domain for developing well-founded legal domain ontology named CriMOnto. The aim of CriMOnto is modeling the content of the legal norms. The Lebanese criminal system is selected as a case study for this work, specifically the Lebanese criminal code.

Chapter 5 works on modeling and formalizing the procedural aspect of the legal norms based on the developed legal domain ontology. A modeling approach as well as a rule language are defined for this purpose.

Chapter 6 describes CORBS, a rule-based legal knowledge based system which is grounded on the integration of the developed legal domain ontology (CriMOnto) and the formalized legal rules. This chapter outlines as well the implementation of this system using Java-based Semantic Web libraries.

Chapter 7 summarizes the thesis, draws conclusions and discusses the future work.

Chapter 2

Background

Contents

2.1 Overview

This chapter introduces the background research of the thesis needed for the development of formal legal ontologies and their use for reasoning support purposes in legal knowledge-based systems. Thus, the most important related fields are Ontology Engineering (OE) for developing legal ontologies and legal Knowledge

Engineering (KE) for building legal knowledge-based systems (LKBS). We start by introducing the ontology engineering domain (section 2.2) and discussing the ontologies (section 2.2.1), their definitions (section 2.2.1.3), classifications (section 2.2.1.4), design criteria (section 2.2.1.5) and components (section 2.2.1.6). Therefore, an overview about foundational ontologies (section 2.2.2) and legal ontologies (section 2.2.3) and their roles are outlined (section 2.2.4). Moreover, we discuss the available methodologies (section 2.2.5), tools (section 2.2.6) and formalisms (section 2.2.7) for developing ontologies as well as some basic ontology engineering support processes (section 2.2.8). Furthermore, the knowledge engineering field is explored (section 2.3) and the available approaches in the legal domain for building knowledge based systems are presented (section 2.3.3). In (section 2.4), the rule-based systems are outlined, their evaluation, methods of reasoning and the existents rule interchange languages. Finally, the conclusion (section 2.5) concludes the chapter.

2.2 Ontology Engineering

The ultimate purpose of *Ontology Engineering* as declared by (Mizoguchi et al., 1997) is: "*To provide a basis of building models of all things in which computer science is interested*". Ontology Engineering refers to the set of activities that concern the ontology development process, the ontology life cycle and methodologies for building ontologies, and the tools and languages that support them (Gomez-Perez et al., 2004). More generally, the ontology engineering is considered as the task of designing, implementing and maintaining ontology-based applications (Euzenat et al., 2007).

In this section, we will overview the term ontology starting from philosophy to the domains of AI and semantic web, ontology definitions, classifications, design criteria and components. Additionally, we will outline the foundational ontologies and the ontologies in the legal domain. The ontology engineering methodologies, tools, languages are discussed as well.

2.2.1 Ontologies

2.2.1.1 From Philosophy to AI

The term ontology is borrowed from philosophy where an ontology is a systematic account of Existence (Gruber, 1993). *Ontology* or *Ontologia* is composed of two greek words: `ontos`(be, exist) and `logos`(study, science), which means the study of what exists, *The science of what is*. *Ontology* is introduced by the german philosopher *Jacob Lorhard* in 1606 in his *Ogdoas Scholastica*, and then used by *Johannes Clauberg* in 1646, the disciple of Descartes, in his *Elementa philosophiae sive Ontosophia*. In this work,

Clauberg considered the ontology as a branch of metaphysics that studies the kinds and structures of objects, properties, events, processes and relations in every area of reality (Savini, 2011).

In the Artificial Intelligence domain, it was John McCarthy who first recognized the overlap between work done in philosophical ontology and the activity of building the logical theories of AI systems. McCarthy affirmed already in 1980 that builders of logic-based intelligent systems must first "list everything that exists, building an ontology of our world" (McCarthy, 1980).

Most AI efforts focused on capturing information about the world that is compatible with the perspective of human common sense, and these efforts were closely allied with research on the topic of common-sense reasoning (Ernest, 1990). A similar perspective, but with broader ambitions and with an even more explicit recognition of an overlap with philosophy, was proposed by John Sowa, who refers to *an ontology for a possible world a catalog of everything that makes up that world, how it's put together, and how it works* (Sowa, 1984).

Finally, the lessons drawn from information systems are resumed in that ontology can support the efforts of those philosophers who have concerned themselves not only with the development of ontological theories, but also in a field sometimes called *Applied ontology* (Koepsell, 1999) - with the application of such theories in domains such as law (Koepsell, 2003), or commerce (Grassl, 1999), or medicine (Schubert, 2001).

2.2.1.2 Conceptualization and Ontologies

The notion of conceptualization has been defined in (Genesereth et al., 1987) as: *an abstract, simplified view of the world that we wish to represent for some purpose. Every knowledge base, knowledge-based system, or knowledge-level agent is committed to some conceptualization, explicitly or implicitly* .

Uschold (Uschold, 1996), considered that a conceptualization may be implicit or explicit. The implicit conceptualization exists only in someone's head or embodied in a piece of software.

Meanwhile, the explicit conceptualization is usually called an *ontology*. Furthermore, Guarino (Guarino, 1998) considered the conceptualization as the result of a complex abstraction process from multiple presentation experiences. Guarino defined formally the conceptualization as an ontological commitment (see Figure 2.1):

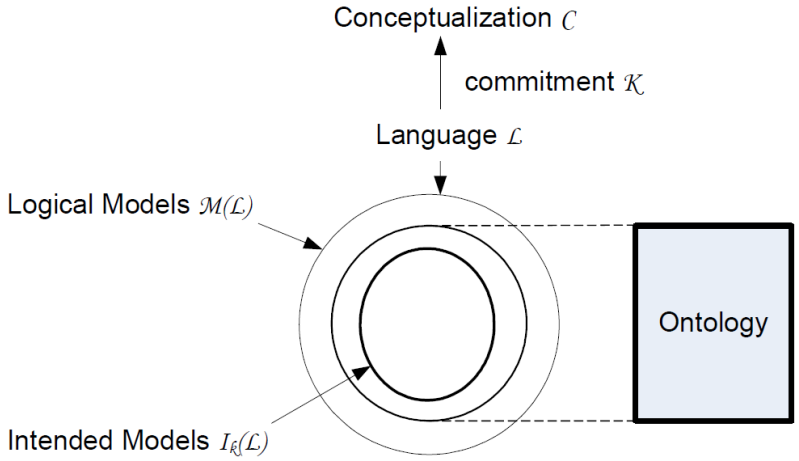


FIGURE 2.1: Conceptualization (Guarino, 1998).

According to Guizzardi, conceptualization of reality is represented by a conceptual model (Guizzardi et al., 2008b). Conceptualizations and Models are abstract entities that only exist in the mind of the user or a community of users of a language. In order to be documented, communicated and analyzed they must be captured, i.e. represented in terms of some concrete artifact. This implies that a language is necessary for representing them in a concise, complete and unambiguous way (Guizzardi, 2005).

The relation between a model and its representation, and their relationship with the conceptualization and representation language is depicted in figure 2.2. The representation of a model in terms of a representation language L is called a model specification and the language L used for its creation is called a modeling (or specification) language.

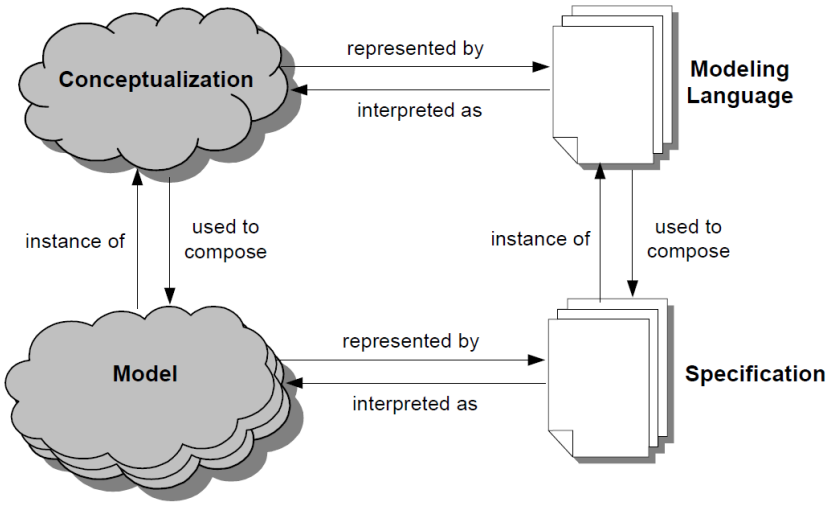


FIGURE 2.2: Relations between conceptualization, Model, Modeling Language and Specification (Guizzardi, 2005).

In this context, a domain ontology is considered as a special type of conceptual model specification (Guizzardi et al., 2008b) and an ontology representation language as a special type of general conceptual modeling language (see Figure 2.3).

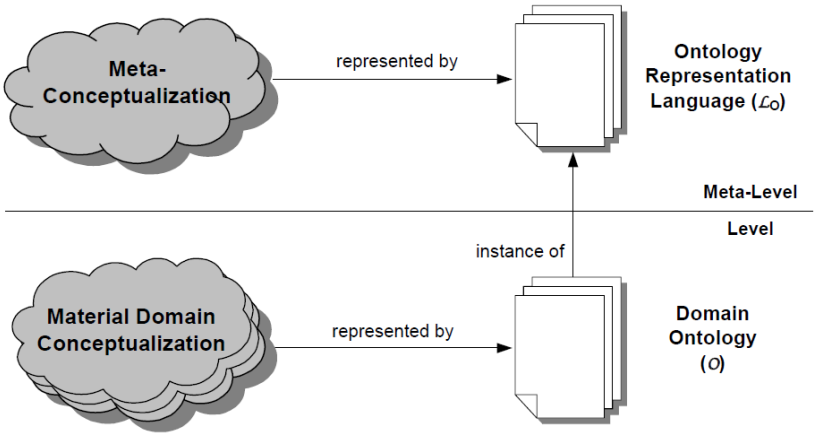


FIGURE 2.3: Relations between a material domain conceptualization, domain ontology, general meta-conceptualization, and ontology representation language (Guizzardi, 2005).

This idea is elaborated from the well known *Ullmann's triangle* depicted in figure 2.4 that represents the relation between a language, a conceptualization and the portion of reality that this conceptualization abstracts.

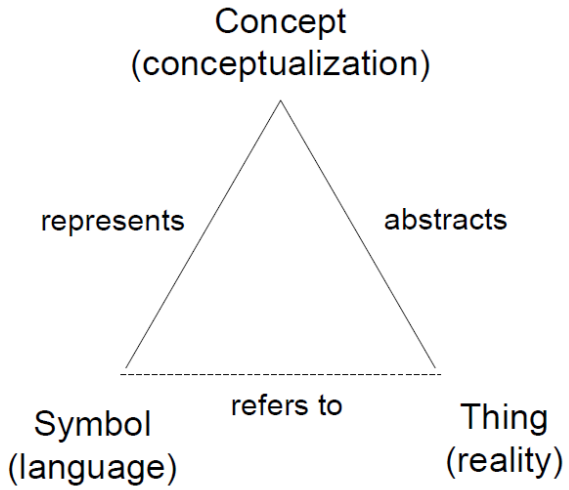


FIGURE 2.4: Ullmann's Triangle (Guizzardi, 2005).

2.2.1.3 Definitions of Ontologies

In the Artificial Intelligence field, Neches and his colleagues (Neches et al., 1991b) were the first who defined the term ontology as follows: *An ontology defines the basic terms and relations comprising the vocabulary of a topic area as well as the rules for combining terms and relations to define extensions to the vocabulary*. This definition is considered as an attempt to explain how to proceed in building an ontology by giving general vague guidelines (Gomez-Perez, 1999). By 1993, Tom Gruber defined the ontology as *An explicit specification of a conceptualization* (Gruber, 1993) (Gruber, 1995). This definition become the most widely famous and cited one. This definition is based on two main concepts: conceptualization and specification.

- Conceptualization: an abstract, simplified view of the world aimed to be represented for some purpose. Every knowledge base, knowledge-based system, or knowledge-level agent is committed to some conceptualization, explicitly or implicitly (Genesereth et al., 1987).
- Specification: the formalization or non-ambiguous formal description of the conceptualization.

The term *explicit* used in the definition means that the ontology concepts and axioms should be represented clearly and precisely.

Furthermore, various authors defined ontologies based on the Gruber's definition, such as Borst and Studder. In 1997, Borst added two new concepts to the Gruber's definition (formal and shared) and defined the ontology as *a formal specification of shared conceptualization* (Borst, 1997). In 1998, Studer combined both definitions in one: *an ontology is a formal, explicit specification of a shared conceptualization* (Studer et al., 1998). According to (Guarino et al., 2009), adding *shared* to the definition of ontologies is essential in order to make them supporting large-scale interoperability and to be well-founded in the sense that the basic primitives they are built on are sufficiently well-chosen and axiomatized to be generally understood. Guarino defined the ontology in the simplest way as *an ontology describes a hierarchy of concepts related by subsumption relationships; in more sophisticated cases, suitable axioms are added in order to express other relationships between concepts and to constrain their intended interpretation* (Guarino, 1998).

For Swartout and his colleagues, *an ontology is a hierarchically structured set of terms for describing a domain that can be used as a skeletal foundation for a knowledge base* (Swartout et al., 1997).

Author	Year	Definition
Gruber	1993	An explicit specification of a conceptualization
Borst	1997	a formal specification of shared conceptualization
Swartout	1997	hierarchically structured set of terms for describing a domain that can be used as a skeletal foundation for a knowledge base
Studer	1998	a formal, explicit specification of a shared conceptualization
Guarino	1998	describes a hierarchy of concepts related by subsumption relationships; in more sophisticated cases, suitable axioms are added in order to express other relationships between concepts and to constrain their intended interpretation

TABLE 2.1: A summary of Ontology definitions.

We conclude that the literature provides a diversity of definitions of the term ontology giving different and complementary points of view of the same reality. Meanwhile, the Gruber's definition is the most widely used and commonly known in the studies of the field.

2.2.1.4 Classifications of Ontologies

In the literature, there is a diversity of strategies for ontology classifications presented in different studies such as: (Schreiber et al., 1993), (Heijst, 1995), (Heijst et al., 1997), (Mizoguchi et al., 1995), (Uschold et al., 1995), (Borst, 1997), (Guarino, 1997) and (Studer et al., 1998).

These classifications of ontologies have been established based on different dimensions: the purpose, the level of formality, the level of generality, the complexity, the subject-matter, etc. The most known classification is the one proposed by Guarino (see Figure 2.5). Meanwhile, most of the researchers agree that it is useful to distinguish between different generality levels of ontologies. Based on this, ontologies can be classified in four main categories: Domain, Generic (upper or core ontologies), Application and Representation.

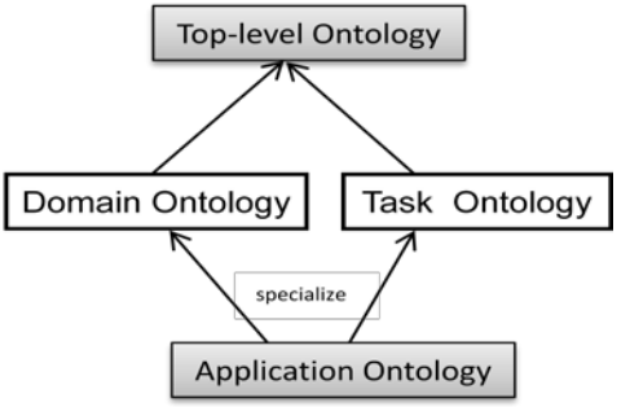


FIGURE 2.5: Example of ontology classification according to the level of generality (Guarino, 1998).

The ontologies of such classification tend to capture static knowledge in a problem-solving independent way. However, Knowledge Engineering is also concerned with problem-solving knowledge, therefore another useful types of ontologies are so-called method and task ontologies are needed (Studer et al., 1998).

In order to resume and to present the most recent analysis of ontology classification, we will refer to a study provided by Gomez-Perez and Fernandez-Lopez (Gomez-Perez, 1999), (Gomez-Perez et al., 2003a). According to them, the most commonly types of ontologies according to their subject of conceptualization are:

- Knowledge representation ontologies: capture the representation primitives used to formalize knowledge in Knowledge representation paradigms.
- General or common ontologies: represent common sense knowledge that is reusable across domains.
- Top-level or upper-level ontologies: describe very general concepts and provide general notions under which all root terms in existing ontologies should be linked.
- Linguistic ontologies: describe semantic constructs rather than model specific domains.
- Domain ontologies: provide vocabularies about concepts and their relationships within a domain and the activities that take place in it.
- Task ontologies: describe the vocabulary related to a generic task or activity by specializing the terms in the top ontologies, providing a vocabulary to solve problems associated with tasks that may or may not belong to the same domain.

- Domain-task ontologies: task ontologies reusable in a given domain, but not across domains (application-independent).
- Method ontologies: give definitions of the relevant concepts and relations applied to specify a reasoning process so as to achieve a particular task.
- Application ontologies: application-dependent and contain all the definitions needed to model the knowledge required for a particular application (specialize the vocabulary of task and domain ontologies for the envisioned application).

2.2.1.5 Criteria of Ontologies

In this section, we summarize some design criteria of ontologies suggested in the literature. In 1993, Gruber (Gruber, 1993) proposed a set of criteria for the design of ontologies based on a claim that "in order to guide and evaluate our designs, we need objective criteria that are founded on the purpose of the resulting artifact". The proposed criteria are: clarity, coherence, extendibility, minimal encoding bias and minimal ontological commitment. Later in 1996, Uschold and his colleagues (Uschold et al., 1996) defined the same general criteria based on Gruber in order to help building ontologies: clarity, consistency and coherence, and extensibility and reusability. Furthermore, Gomez-Perez provided list of criteria "that have been proved useful for ontology development" (Gomez-Perez, 1999). This list is established based on the previous authors and other contributions.

- Clarity and objectivity (Gruber, 1993): "the ontology should provide the meaning of defined terms by providing objective definitions and also natural language documentation";
- Completeness (Gruber, 1993): "a definition expressed by a necessary and sufficient conditions is preferred over a partial definition (defined only by a necessary or sufficient condition)";
- Coherence (Gruber, 1993): "to permit inferences that are consistent with the definitions";
- Maximum monotonic extendibility (Gruber, 1993): "new general or specialized terms should be included in the ontology in such a way that is does not require the revision of existing definitions";
- Minimal ontological commitments (Gruber, 1993): "making as few claims as possible about the world being modeled, which means that the ontology should specify as little as possible about the meaning of its terms, giving the parties committed to the ontology freedom to specialize and instantiate the ontology as required";

- Ontological Distinction Principle (Borgo et al., 1996): "classes in an ontology must be disjoint";
- Diversification of hierarchies to increase the power provided by multiple inheritance mechanisms (Arpirez et al., 1998): "if enough knowledge is represented in the ontology and as many different classification criteria as possible are used, it is easier to enter new concepts and to inherit properties from different points of view";
- Modularity (Bernaras et al., 1996): "to minimize coupling between modules";
- Minimization of the semantic distance between sibling concepts (Arpirez et al., 1998): "Similar concepts are grouped and represented as subclasses of one class and should be defined using the same primitives, whereas concepts which are less similar are represented further apart in the hierarchy";
- Standardization of names whenever is possible (Arpirez et al., 1998).

2.2.1.6 Components of Ontologies

According to authors such as Gruber (Gruber, 1993), knowledge in ontologies are formalized using five main kinds of components: concepts, relations, functions, axioms and instances. For (Gomez-Perez, 1999), classes in ontologies are usually organized in taxonomies. Sometimes, taxonomies are considered to be full ontologies (Studer et al., 1998). Gomez-Perez established the list of components as defined by Gruber (Gomez-Perez, 1999).

- Concepts: "can be abstract or concrete, elementary or composite, real or fictitious. A concept can be anything about anything about which something is said and therefore, could also could also be the description of a task, function, action, strategy, reasoning process, etc";
- Relations: "represent a type of interaction between concepts of the domain. They are formally defined as any subset of a product of n sets, that is: *R : C1xC2x...xCn. Examples of binary relations subclass-of and connected-to* ";
- Functions: "special case of relations in which the n-th element of the relationship is unique for the n-1 preceding elements. Formally, functions are defined as : *F: C1xC2x...xCn-1* $\Longrightarrow$ *Cn*. Examples of functions are Mother-of and square";
- Axioms: "are used to model sentences that are always true";
- Instances: "are used to represent elements".

Recent studies such as (Mizoguchi, 2004) and (Guarino et al., 2009) consider that in order to build expressive ontologies, hierarchically organized concepts, axiomatic

definition of concepts and relations are necessary. From this perspective, the main components of an expressive ontology **O** are identified. A formal ontology is represented mainly by 6-tuple (Guarino et al., 2009): **O = (C, P, H, R, I, A)**, Where:

C = $C^C \cup C^I$, where: C is the set of entities or concepts of the ontology. C^C is the classes set, i.e. concepts that represent entities.

Example: Person $\in C^C$.

C^I is a class instances set.

Example: Erik Brown $\in C^I$

P = $prop^C$ (c_k, datatype) | $c_k \in C^C$ P is the set of properties of ontology entities, where the relationship $prop^C$ defines the basic datatype of a class property.

Example: subject (Case, String).

H = kind_of (c_1, c_2) | $c_1 \in C^C$, $c2_2 \in C^C$, where $c1_1$ is a subclass of c_2. H is the set of taxonomic relationships between concepts. These relationships define a concept hierarchy and are denoted by "kind_of (c_1, c_2) ".

Example: kind_of(Lawyer, Person).

R = rel_k (c_1, c_2,..., c_n) | $\forall$ i, $c_i \in C^C$ R is the set of non-taxonomic ontology relationships.

Example: represents (Lawyer, Client).

I = is_a (c_1,c_2) | $c_1 \in C^I$, $c_2 \in C^C$ $\cup$ $prop^I$ (c_k,value) | $c_k \in C^I$ $\cup$ rel_k(c_1,c_2,..., c_n) | $\forall$ i, $c_i \in C^I$ I is the set of instance relationships related to the C^C , P and R sets.

Example: C^C: is_a (Anne Smith, Client) R: represents (Erik Brown, Anne Smith) P: subject (Case12, "adoption")

A= $condition_x \implies conclusion_y$ (c_1,c_2,..., c_n) | $\forall$ j, $c_j \in C^C$

$condition_x$ = ($cond_1$,$cond_2$,...,$cond_n$) | $\forall$ z, $cond_z \in H \cup I \cup R$.

A is a set of axioms, rules that allow checking the consistency of an ontology and infer new knowledge through some inference mechanism.

Example: "$\forall$ Defense_Argument, OldCase, NewCase, applied_to (Defense_Argument, OldCase), similar_to (OldCase, NewCase) $\implies$ applied_to (Defense_Argument, New-Case)

(If two legal cases are similar then, the defense argument used in one case could be applied to the other one).

2.2.2 Foundational Ontologies

Foundational ontologies are the most general and formal ontologies (Borgo et al., 2004). They define a range of top-level domain-independent ontological categories, which form a general foundation for more elaborated domain-specific ontologies (Guizzardi et al., 2005b). Theoretically, they are well-founded domain independent systems of categories that have been successfully used to improve the quality of conceptual models and semantic interoperability (Guizzardi et al., 2010b) (Guizzardi et al., 2008a). Foundational or top ontologies have four main roles:

- Give a starting position for more detailed modeling by providing a structure of classes where a domain ontology can hang its main concepts as subclasses.
- Help in consistency checking by applying inheritance (get already defined properties).
- by reusing them, they tend to facilitate and speed up the ontology development process by preventing to reinvent known modeling solutions (Keet, 2011).
- Help to establish a theoretical basis to achieve consistency in the negotiations of meaning during the collaboration process (Rosa et al., 2012).

Various foundational ontologies exist in the literature such as DOLCE (Masolo et al., 2003) and UFO (Guizzardi et al., 2005b). In the following, the Unified Foundational Ontology (UFO) is discussed.

2.2.2.1 The Unified Foundational Ontology UFO

The unified foundational ontology (UFO) is an example of a descriptive foundational ontology that has been constructed for more than a decade employing results from formal ontology, cognitive psychology, linguistics, philosophical logics, but also significant accumulated empirical and theoretical results from the area of conceptual modeling in computer science (Griffo et al., 2015). UFO has been used to evaluate, redesign and integrate (meta) models of conceptual modeling languages, as well as to evaluate, re-design and give real-world semantics to domain ontologies.

UFO is initially proposed by Guizzardi and Wagner (Guizzardi et al., 2005b) and developed to support the activities of both conceptual and organizational modeling. UFO essentially distinguishes between *Universals* and *Particulars* (see Figure 2.6). Particulars are entities that exist in reality possessing a unique identity, while universals are patterns of features, which can be realized in a number of different particulars.

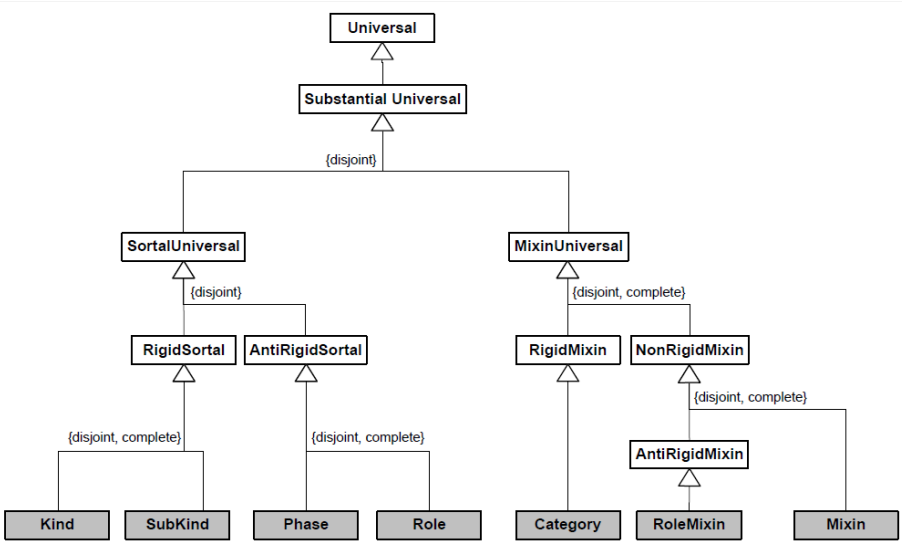


FIGURE 2.6: Ontology of *Universals* (Guizzardi, 2005).

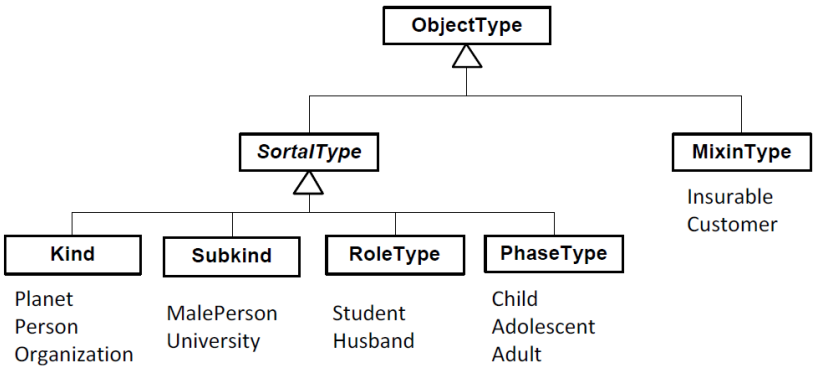


FIGURE 2.7: Ontology of *Universals* (Guizzardi et al., 2012).

Therefore, UFO permits the building of an ontology by reusing some generic concepts as modeling primitives such as `category`, `kind`, `subkind`, `relator`, `role` and `role-mixin` where the ontologist does not need to rebuild these concepts (see Figure 2.8).

- The concept `Kind` provides a principle of application and a principle of identity for its instances (Guizzardi, 2005). It represents a rigid concept, i.e., a class that applies necessarily to its instances. In other words, instances of these types will continue to be so as long as they exist in the model (Guizzardi, 2005).
- A `Kind` can be described in a taxonomic structure where its subtypes are also rigid types known as `Subkinds` (e.g., Man and Woman) (Guerson et al., 2014).
- The concept `Role`, in turn, is an anti-rigid concept, applying contingently to its instances (e.g., Offender, Instigator).

- A `Phase` is an anti-rigid concept that it is defined by a partition of a kind and whose contingent instantiation condition is related to intrinsic changes of an instance of that kind.
- A `Relator` (e.g. entities with the power of connecting other entities) is a rigid concept and existentially depends on the instances it connects through mediation relations.
- A `Mixin` represents properties which are essential to some of its instances and accidental to others (semi-rigidity). An example is the mixin Seatable, which represents a property that can be considered essential to the kinds Chair and Stool, but accidental to Crate, Paper Box or Rock.
- A `RoleMixin` represents an anti-rigid and externally dependent nonsortal, i.e., a dispersive universal that aggregates properties which are common to different roles. It includes formal roles such as whole and part, and initiatior and responder.
- A `Category` represents a rigid and relationally independent `mixin`, i.e., a dispersive universal that aggregates essential properties which are common to different substance sortals. For example, the category RationalEntity as a generalization of Person and IntelligentAgent.

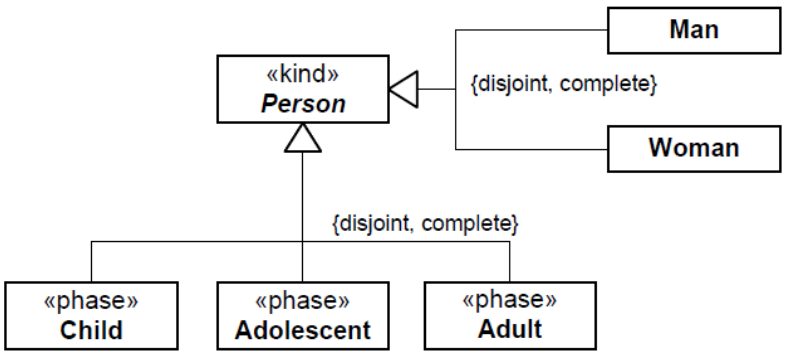


FIGURE 2.8: Two partitions of the same Kind Person: a subkind partition (Man, Woman) and a phase partition (Child, Adolescent, Adult) (Guizzardi, 2005).

UFO is divided into three layered sets: UFO-A, UFO-B and UFO-C.

- UFO-A (see Figure 2.9): the core of UFO (Guizzardi et al., 2004a), ontology of objects, defines terms related to *endurants* such as `Universal`, `Relator`, `Role`, `Intrinsic_Moment` among many others.

  A fundamental distinction in this ontology is between the categories of Particular (Individual) and Universal (Type). Particulars are entities that exist

in reality possessing a unique identity. Universals, conversely, are pattern of features, which can be realized in a number of different particulars.

Substances are existentially independent individuals such as person, house, car, etc..

Moment, in contrast, denotes an individualized (objectified) property or property in particular. The word bears no relation to the notion of time instant in colloquial language. A moment is an individual that can only exist in other individuals. Typical examples of moments are a color, a connection, an electric charge, a symptom, a covalent bond. Moments have in common that they are all dependent of other individuals (their bearers).

Intrinsic moments are dependent of one single individual (e.g., color, a headache, a temperature).

Relators depend on a plurality of individuals (e.g., an employment, a medical treatment, a marriage).

Relations are entities that glue together other entities. Two main categories of relations are typically considered, namely, **material** and **formal** relations.

Situations are special types of endurants. These are complex entities that are constituted by possibly many endurants (including other situations). Situations are taken here to be synonymous to what is named state of affairs in the literature, i.e., a portion of reality that can be comprehended as a whole. Examples of situations include "*John being with fever and influenza*".

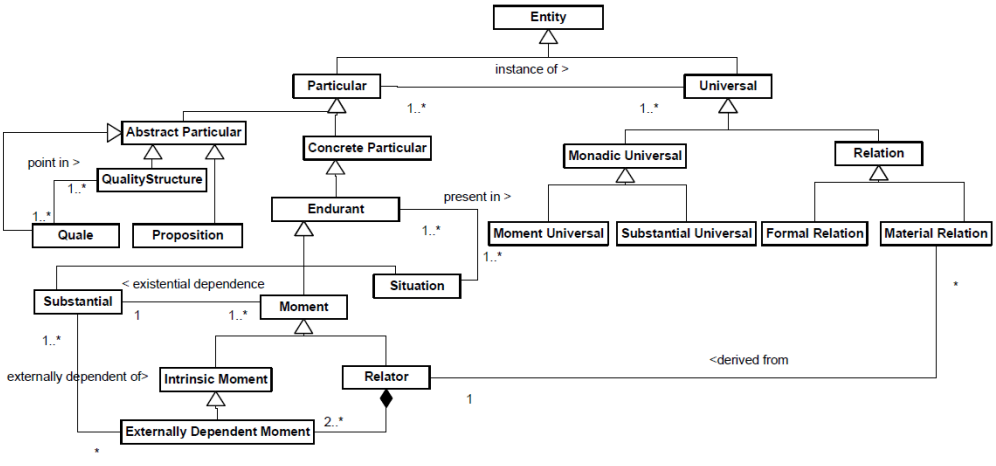


FIGURE 2.9: A Fragment of a Foundational Ontology of Endurants (UFO-A) (Guizzardi et al., 2008a).

- UFO-B (see Figure 2.10): ontology of events, defines terms related to *perdurants* such as `Event`, `State`, `Atomic_Event`, `Complex_Event` among many others (Guizzardi et al., 2013a).

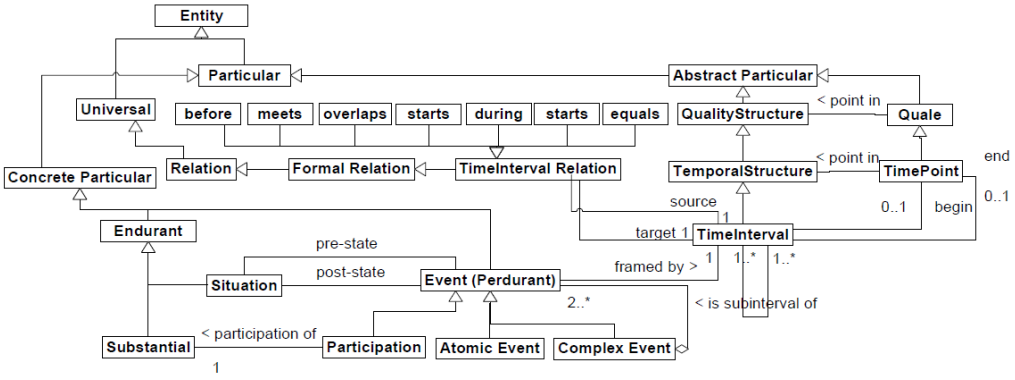


FIGURE 2.10: A Fragment of a Foundational Ontology of Perdurants (UFO-B) (Guizzardi et al., 2008a).

UFO-B makes a distinction between *endurants* and *perdurants*. *Endurants* are individuals that are wholly present whenever they are present, i.e., they don't have temporal parts (Guizzardi et al., 2004a) (Guizzardi et al., 2008a) . Examples of endurants are a house, a person, the moon.

"*Perdurants* are individuals composed of temporal parts, they happen in time in the sense that they extend in time accumulating temporal parts. Examples of perdurants are a conversation, a football game, a symphony execution, a birthday party, the Second World War and a business process" (Guizzardi et al., 2008a).

As shown in figure 2.10, the main category on this ontology is `Event` (Perdurant, Occurrent). Events can be `Atomic` or `Complex`, depending on their mereological structure. Atomic events have no improper parts. Meanwhile, complex events are aggregations of at least two events (that can themselves be atomic or complex).

Events are ontologically dependent entities in the sense that they existentially depend on their participants in order to exist (see Figure 2.11). As shown in figure 2.11, the relations of exclusively depends on, participation of and the notion of participation itself are all derived notions (derived from the relations of parthood and existential dependence) (Guizzardi et al., 2013a).

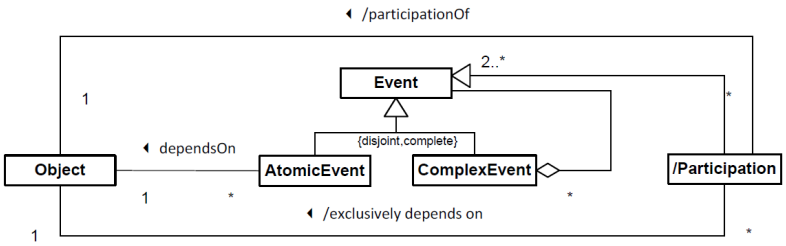


FIGURE 2.11: Complex Events as Sums of Object's Participations (Guizzardi et al., 2013a).

- UFO-C: ontology of social_entities (both endurants and perdurants) built on top of UFO-A and UFO-B. UFO-C defines terms related to intentional and social entities including linguistic aspects such as `Social_Agent`, `Social_Object`, `Social_Role` and `Normative_Description`. A fragment of this ontology is depicted in Figure 2.12.

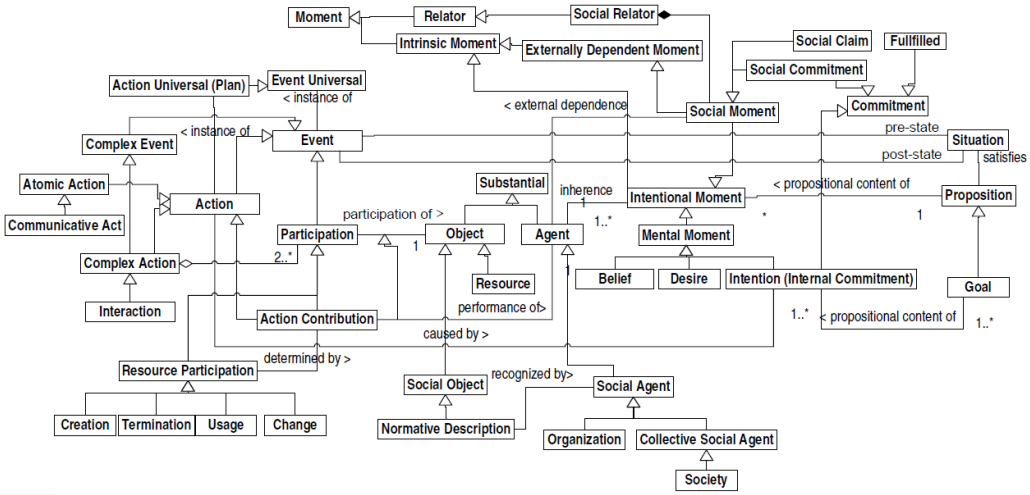


FIGURE 2.12: A Fragment of a Foundational Ontology of Social Entities (UFO-C) (Guizzardi et al., 2008a).

In UFO-C, there is a distinction between Agents and Objects. Agents can be physical (e.g., a person) or social (e.g., an organization, a society). Objects are particulars that possess (direct) spatial-temporal qualities and that are founded on matter (Guizzardi et al., 2010a). They can also be further categorized in physical and social objects. Physical objects include a book, a tree, a car; Social objects include money, language and Normative Descriptions.

Figure 2.13 presents a fragment of UFO-C that focuses on the distinction between agents and objects, and on the definition of normative description.

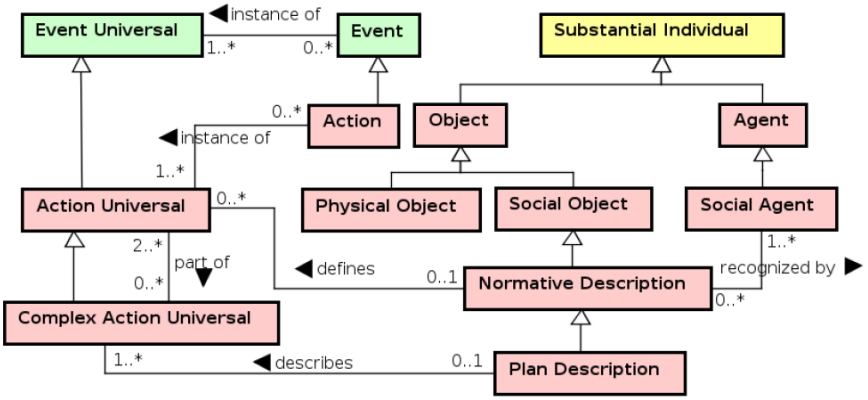


FIGURE 2.13: Agents, objects, and normative descriptions (Nardi et al., 2016).

A `normative description` defines one or more rules/norms recognized by at least one social agent.

`Agents` are substantials that can bear special kinds of moments named Intentional Moments.

Every `intentional moment` has a type (e.g., Belief, Desire, Intention) and a propositional content. *Intending something* is a specific type of intentionality termed Intention. The propositional content of an Intention is a Goal.

`Intentions` cause the agent to perform Actions.

`Actions` are intentional events, i.e., events which instantiate a Plan.

A Communicative Act (a speech act such inform, ask or promise) is an example of an atomic action.

As events, actions can be atomic or complex. A `complex action` is composed of two or more `participations`. These participations can themselves be `intentional` (i.e., be themselves actions) or `unintentional` events.

Figure 2.14 presents a fragment of UFO-C that focuses on types of intentional moments.

It is not the case that any participation of an agent is considered an action, but only those intentional participations - termed here Action Contributions.

Only agents (entities capable of bearing intentional moments) can perform actions. An object participating in an action is termed a Resource.

A complex action composed of action contributions of different agents is termed an Interaction.

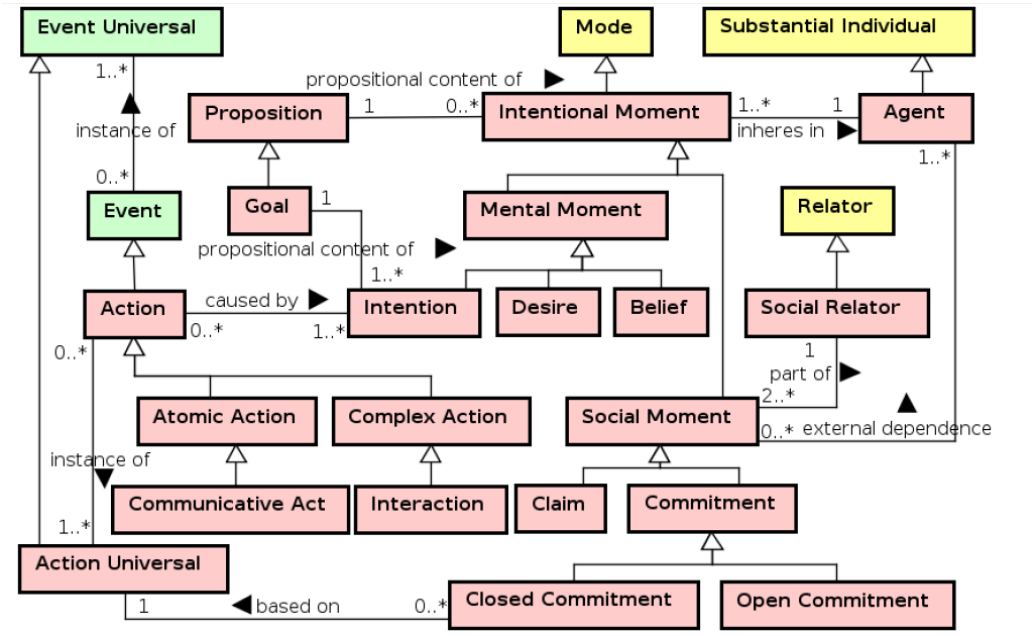


FIGURE 2.14: Actions, mental moments, and social moments (Nardi et al., 2016).

2.2.3 Ontologies in the legal Domain

Legal ontologies are generalized conceptual models of specific parts of the legal domain. They provide stable foundations for knowledge representation (Mommers, 2003). They have been developed already far before the idea of the Semantic Web was put into W3C actions and standards where they have been used for legal knowledge management and as knowledge bases in legal knowledge systems (Benjamins et al., 2005a). Moreover, these ontologies have been useful in a number of applications to support information retrieval, extraction, integration and reasoning as demonstrated by (Wyner, 2009), (Ashley, 2009) and (Wyner et al., 2010).

It is important to notify that legal ontologies differ from ontologies in other fields of practice, like medicine or engineering in that they have to cover a wide range of common-sense concepts that are part of physical, abstract, mental, and social worlds. Legal domains share complex and varied notions of norm and responsibility, but besides this legal *core*, a legal domain refers to some world of social activities (Benjamins et al., 2005a). Concerning the legal core notions, they include: norm, case, contract, institution, person, agent, role, status, normative position (duties, rights, etc.), responsibility, property, crime, provision, interpretation, sanction, delegation, legal document.

We revised the literature to offer a brief review about some of the most relevant legal ontologies: FOLaw(Valente et al., 1994b) (Valente et al., 1996) (Valente et al., 1999a), LRI-Core (Breuker, 2004), OCL.NL (Ontology of Dutch Criminal Law) (Breuker et al., 2002), LKIF-Core (Hoekstra et al., 2007), DALOS domain ontology (Agnoloni

et al., 2007) (Francesconi et al., 2008), OPJK (Casellas, 2008a) and UFO-L (Griffo et al., 2015).

2.2.3.1 The Functional Ontology of Law (FOLaw)

The Functional Ontology of Law (FOLaw) is a legal core ontology developed by (Valente, 1995). FOLaw captured dependencies between various types of knowledge involved in legal reasoning (see Figure 2.15). The main purpose of FOLaw is to enable knowledge reuse for building new applications, in particular, enabling the reuse of a library of ontologies. However, authors also included the idea that it could be used for the study of legal knowledge (Valente et al., 1996).

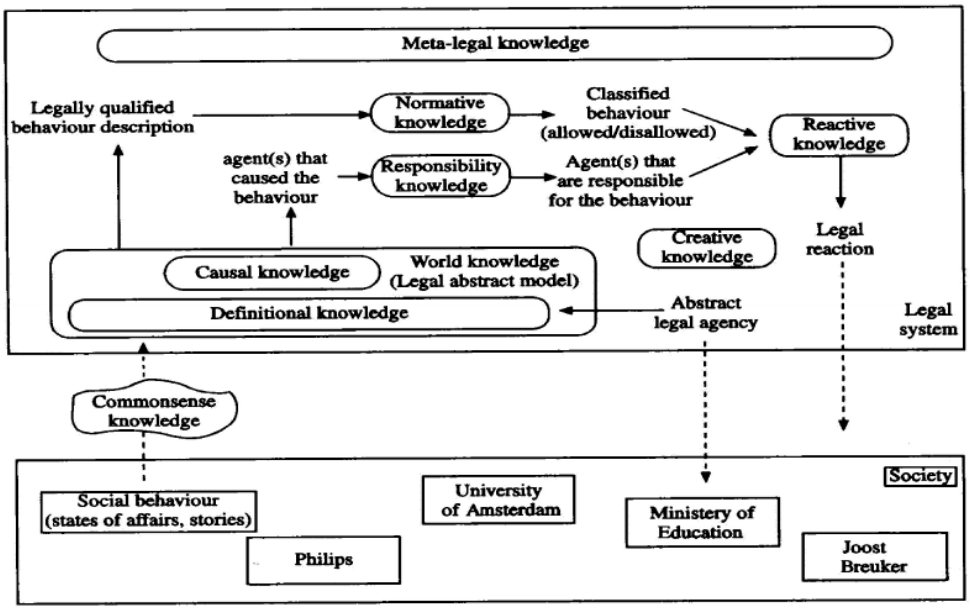


FIGURE 2.15: The Functional Ontology of Law FOLaw (Valente, 1995).

FOLaw was used for designing the architecture the project ON-LINE (Valente et al., 1999a) for legal problem solving specifically to distinguish the various types of knowledge in legal reasoning, to explain the dependencies between types of legal knowledge, to organize and index libraries of domain ontologies and to support knowledge acquisition towards the construction of new ontologies.

List of primitive categories are defined in FOLaw: normative knowledge, world knowledge, responsibility knowledge, reactive knowledge, creative knowledge, and meta-level knowledge.

- Normative knowledge: this is probably the most typical legal knowledge: it refers to norms(adopted by the authors from Kelsen (Kelsen, 1991)) as

indicated by deontic operators such as 'permitted', 'forbidden and 'obliged' but also to concepts like rights and duties.

- world knowledge: composed of two related types of knowledge: definitional knowledge (used by normative knowledge) and causal knowledge (used by responsibility knowledge).
- responsibility knowledge: had the function of assigning or limiting the responsibility of an agent over certain state of affairs. Acts as an intermediary concept between normative and reactive knowledge.
- reactive knowledge: specified which action ought to be taken and how. It concerns the kinds of punishments or rewards that the law has in stock.
- creative knowledge: it was defined as the creation of entities that did not exist before in the world. The law may create (virtual or real) agents or institutions with a legal status.
- meta-level knowledge: the knowledge required to solve conflicts between norms.

Finally, Breuker and Hoekstra (Breuker et al., 2004a) described FOLaw as an epistemological framework, rather than a core ontology as it "lacked the abstract, core concepts that make up law", although the authors recognized that it had provided insight into legal reasoning.

2.2.3.2 LRI-Core

LRI-Core is a more generic core ontology developed in order to overcome the epistemological promiscuity of the FOLaw ontology (Breuker et al., 2004a). The purpose of developing our 'LRI-Core' is not to propose yet-another-upper-ontology, but to provide a broad, rather than 'deep' conceptual structure for the typical legal, or legally relevant, 'upper notions (Breuker et al., 2002). LRI-Core aims at supporting knowledge acquisition for legal domain ontologies (Breuker, 2004).

LRI-Core ontology was not grounded in any existing foundational or top ontology, but is constructed including FOLaw. The legal core ontology consists at the top level of five portions or worlds: physical concepts (object and process), mental concepts, abstract concepts, roles and occurrences (see Figure 2.16).

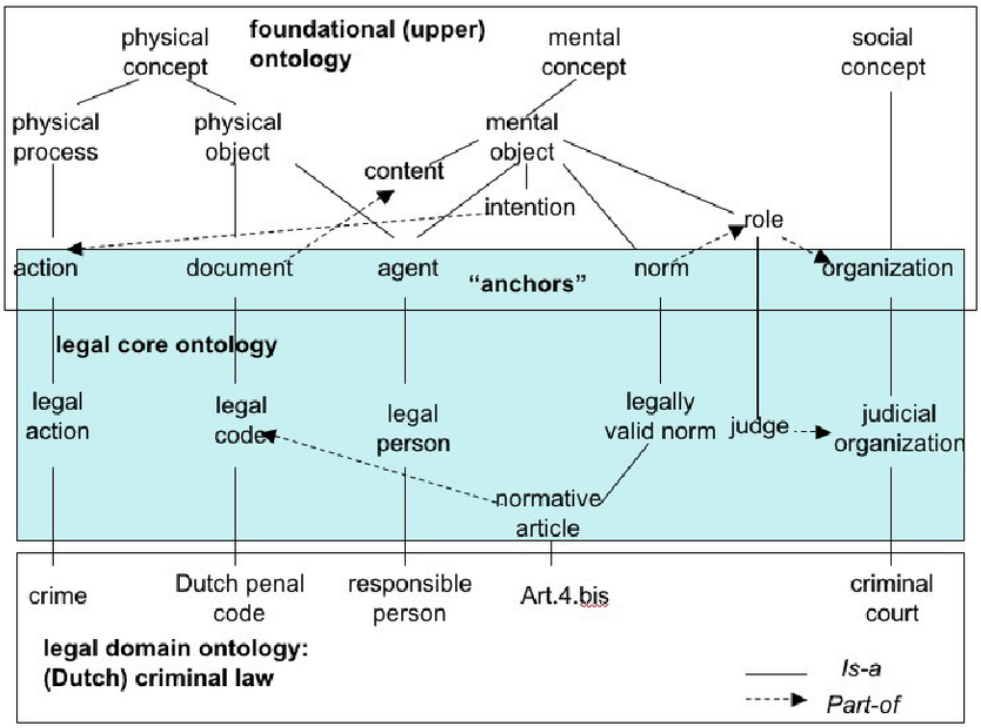


FIGURE 2.16: LRI-Core (Breuker et al., 2004c).

The upper or foundational part of LRI-Core had 5 major worlds: physical world (Energy, Matter, Phycical Object, Substance, Physical Change, Time, Space), mental world (Emotion, Mental Process, Mental Object), roles (Case Role, Social role, Communication Role, Function, Social Organisation), abstractions (Set, Formula, Geometrical Entity, Number) and occurrences (Situation, History, Spatial Temporal Reference, Event, Causation, State).

The ontology was formalized in OWL-DL using Protégé and in (Breuker et al., 2003) LRICore was formalized in RDF(S) with Protégé-2000). The size of the ontology was about 100 concepts (Breuker, 2004).

LRI-Core was used to aid the development of ontologies in the e-Court project (Breuker et al., n.d.), specifically, used by the Dutch criminal law ontology of the e-Court project to support knowledge acquisition.

Nevertheless, the authors concluded that "the number of legal concepts in LRI-Core was rather small; it was rather a top ontology covering abstract concepts of common-sense rather than the field of law" (Breuker et al., 2007).

2.2.3.3 Ontology of Criminal Law (OCL.NL)

The OCL.NL ontology was developed by Leibniz Center for Law within the e-Court project[1] that aimed at semi-automated information management of transcriptions of criminal trial hearings (Breuker et al., 2002).

The core of OCL.NL is conformed by actions (offences) and punishments. The role of ontologies in indexing the e-Court hearing documents was: 1) they provided a structured vocabulary for meta-data descriptions and maintained consistent use and semantic distinctions, 2) the ontology browser supported the hand tagging of the hearings, and 3) the concepts contained in the ontology were used to index documents (Breuker et al., 2004c).

OCL.NL concepts were 'anchored' in LRI-Core concepts. Figure 2.17 describes the anchoring of the OCL.NL concepts into *agent* LRI-Core concepts (boldface terms).

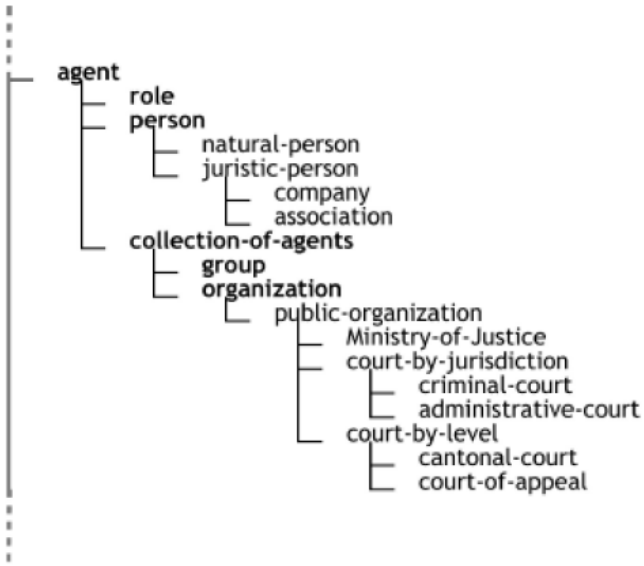


FIGURE 2.17: Concept anchoring in OCL.NL/LRI-Core (Breuker et al., 2002).

2.2.3.4 LKIF-Core

LKIF is a Legal Knowledge Interchange Format developed at the Estrella Project[2] in order to "enable the translation between legal knowledge bases written in different representation formats and formalisms" and to act "as a knowledge representation formalism that is part of a larger architecture for developing legal knowledge systems" (Hoekstra et al., 2007).

[1] e-COURT European Project IST-2000-28199, project duration 2000-2003

[2] Estrella is a 6th European Framework project (IST-2004-027665). See also: http://www.estrellaproject.org

The LKIF Core Ontology [3] is a legal core ontology that contains 'basic concepts of law' and is part of a generic architecture to enable the interchange of knowledge (LKBS). Thus, LKIF-Core is directed at supporting legal inference, knowledge acquisition and knowledge exchange.

For building LKIF-Core, a combination of methodologies for ontology engineering are used such as (Uschold et al., 1996) within others. The main steps of the ontology building process are: identification of purpose and scope, ontology capture, ontology coding, integration with existing ontologies, and evaluation.

The ontology is composed of three main layers: Top level(most classes are borrowed from LRI-Core), Intentional Level(includes concepts and relations which describe behaviour "of rational agents that can be effectively influenced by the law") and Legal level (legal agents, actions, rights, roles and concepts definitions).

The LKIF legal core ontology is composed of 15 modules (see Figure 2.18), categorized in 3 main categories, each of which describes a set of closely related concepts from both legal and commonsense domains (Hoekstra et al., 2009).

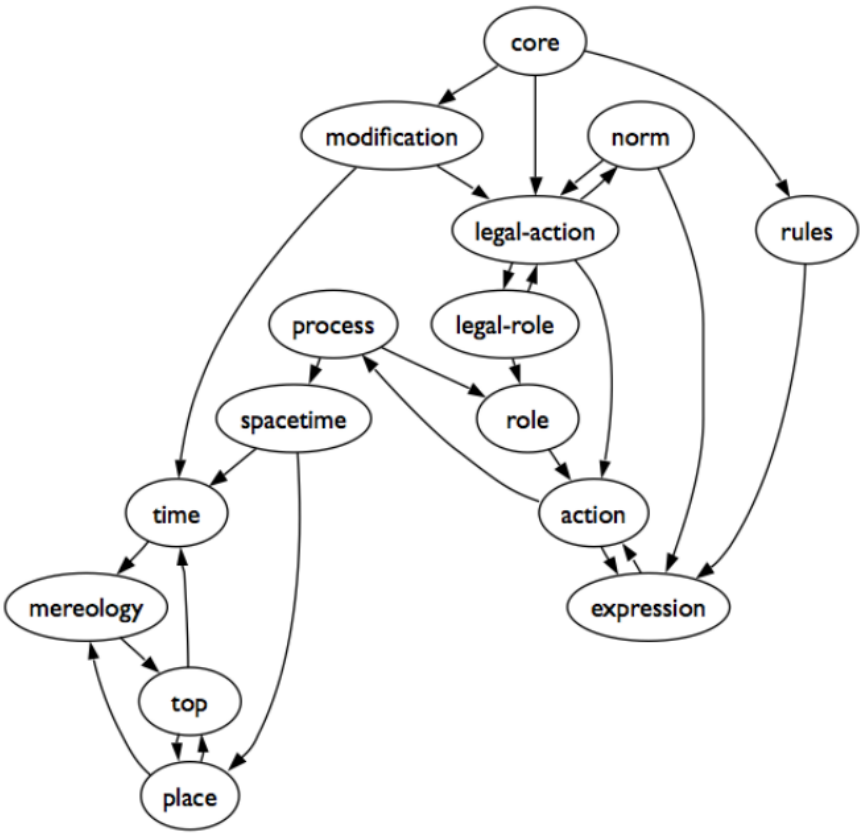


FIGURE 2.18: The modules of LKIF-Core.

- Abstract Concepts: The most abstract concepts are defined in five closely related modules: top, place, mereology, time and spacetime.

[3] http://www.leibnizcenter.org/general/lkif-core-ontology, retrieved september 10 2017

  - Top: The LKIF top ontology is largely based on the top-level of LRI-Core but has less ontological commitment in the sense that it imposes less restrictions on subclasses of the top categories.
  - Place: The place module partially implements the theory of relative places in OWL DL.
  - Mereology: The mereology module defines mereological concepts such as parts and wholes, and typical mereological relations such as part of, component of, containment, membership etc.
  - Time: The time module provides an OWL DL implementation of the theory of time.

- Basic Concepts: Basic-level concepts are distributed across four modules: process, role, action and expression.
  - Process: The process module extends the LKIF top ontology module with a definition of changes, processes (being causal changes) and physical objects. It introduces a limited set of properties for describing participant roles of processes.
  - Role: The role module defines a typology of roles (epistemic roles, functions, person roles, organisation roles) and the plays-property for relating a role filler to a role.
  - Action: The action module describes the vocabulary for representing actions in general. Actions are processes which are performed by some agent (the actor of the action). This module does not commit itself to a particular theory on thematic roles.
  - Expression: The expression module describes a vocabulary for describing, propositions and propositional attitudes (belief, intention), qualifications, statements and media. It furthermore extends the role module with a number or epistemic roles, and is the basis for the definition of norms.

- Legal Concepts: These basic clusters are extended by three modules that form the legal ontology: legal action, legal role and norm.
  - Legal-action: The legal action module extends the action module with a number of legal concepts related to action and agent, such as public acts, public bodies, legal person, natural person etc.
  - Legal-role: The legal role module extends the role module with a small number of legal concepts related to roles, legal professions etc.
  - Norm: The norm module is an extension primarily on the expression module where norms are defined as qualifications. It furthermore defines

a number of legal sources, e.g. legal documents, customary law etc., and a typology of rights and powers.

- Framework Modules: two modules(modification and rules) are provided that cover the basic vocabulary of two frameworks.
  - Modification: The modification module is both an extension of the time module and the legal action module. The time module is extended with numerous intervals and moments describing the efficacy and being in force of legal documents. The action module is extended with a typology of modifications.
  - Rules: The rules & argumentation module defines roles central to argumentation, and describes the vocabulary for LKIF rules.
- Core and Extended Ontology: Finally, the twelve modules of the abstract, basic and legal level are integrated in the LKIF Core ontology module. This module does not provide any additional definitions, but functions as an entry-point for users of the ontology library. The two framework modules are accessible through the LKIF Extended ontology module. This module imports the LKIF Core module.

The ontology is formalized in OWL-DL. The current version of LKIF-Core (1.1) contains 155 classes and 97 object properties [4].

2.2.3.5 DALOS Domain Ontology

The DALOS domain ontology is the consumer protection ontology developed within the DALOS project by Francesconi et al., 2007 (Agnoloni et al., 2007) (Francesconi et al., 2008) (Agnoloni et al., 2009) (Biasiotti et al., 2011). The DALOS project aims at ensuring that legal drafters and decision-makers have control over legal language at national and European level, by providing law-makers with the linguistic and knowledge management tools to be used in the legislative processes, in particular within the phase of legislative drafting. This project uses the results of the LOIS project, composed by about 35.000 concepts, but focuses on the consumer protection domain.

The DALOS resource is organized in two layers (see Figure 2.19):

- Ontological layer: contains the conceptual representations of the domain at a language-independent level (Agnoloni et al., 2007). This layer acts as a knowledge layer where to align concepts at European level independently from the language and the legal systems.

[4] https://github.com/RinkeHoekstra/lkif-core, retrieved september 10, 2017

- Lexical layer: contains lexical manifestations in different languages (Italian, English, Spanish and Dutch) of the concepts at the ontological layer.

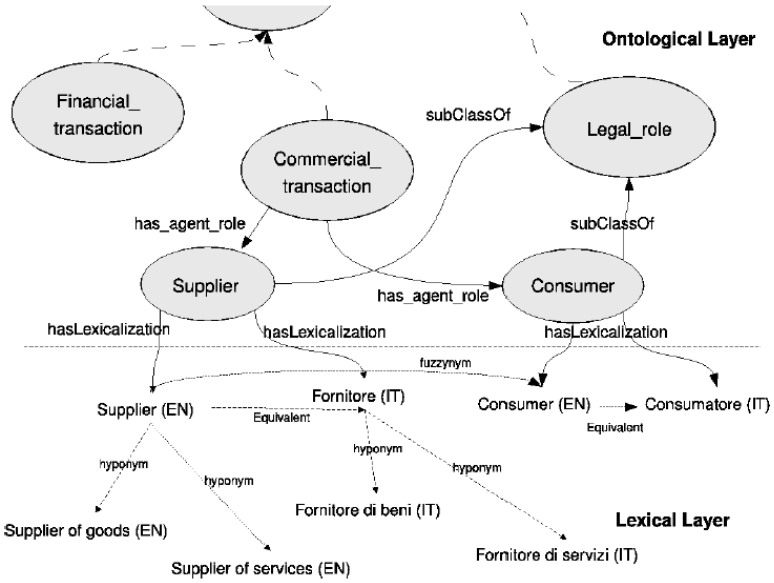


FIGURE 2.19: DALOS knowledge organization (Francesconi et al., 2008).

At the ontological layer, concepts are linked by *subsumption* (`subclass-of`) as well as by domain-dependent object property relationships. However, at the lexical layer, terms are linked by linguistic relationships as those ones used for the LOIS database (*hyperonymy*, *hyponymy*, *meronymy*, etc.). In particular, to implement the lexical layer, the subset of the LOIS database pertaining to the *consumer protection* lexicon is used.

The connection between these two layers is aimed at representing the relationship between concepts and their lexical manifestations. In order to link the two layers, the `hasLexicalization` and `hasConceptualization` relationships are used.

Therefore, the DALOS ontological-linguistic resource is implemented through three main activities (Agnoloni et al., 2009) (Francesconi et al., 2007):

- Semi-automatic term extraction on the domain of `consumer protection` law from a set of selected texts by using NLP tools such as GATE and ontology learning systems such as T2K (Lexical layer implementation)

  T2k (Text to knowledge) (Dell'Orletta et al., 2014) is a hybrid ontology learning system combining linguistic technologies and statistical techniques and is used for implementing the Italian version of the DALOS Lexical layer.

  GATE is an architecture, a framework and a development environment for Language Engineering (LE) applications. GATE uses NLP based techniques to assist the knowledge acquisition process for ontological domain modeling,

applying automated linguistic analyses to create ontological knowledge from textual resources or to assist ontology engineers and domain experts by means of semi-automatic techniques.

- Manual construction of a Domain Ontology on the `consumer protection` domain (Ontological layer implementation);
- Connection between the Lexical layer (the LOIS database) and the Ontological layer by the `hasLexicalization` property implementation.

The DALOS domain ontology (see Figures 2.20 and 2.21) imports some top-level basic notions, such as `legal_role` and `legal_situation` (see Figure 2.22), from so called CLO (Core Legal Ontology) which specializes the DOLCE foundational ontology.

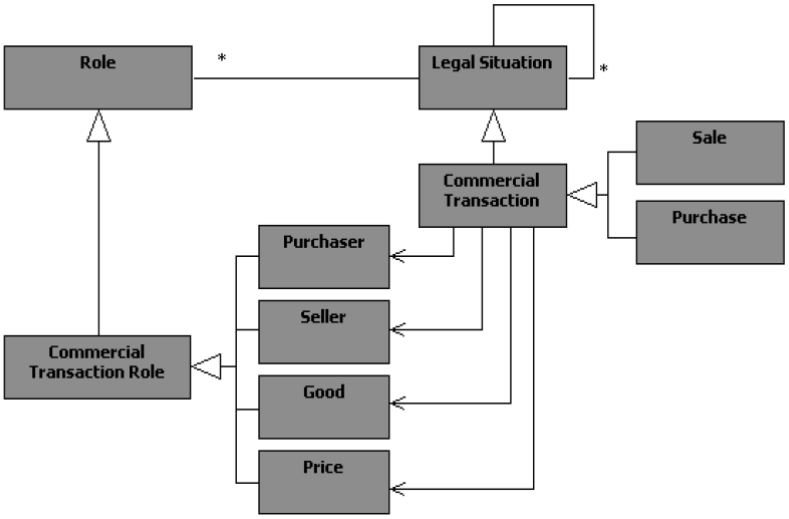


FIGURE 2.20: Excerpt of the DALOS Consumer Protection Ontology (Francesconi et al., 2008).

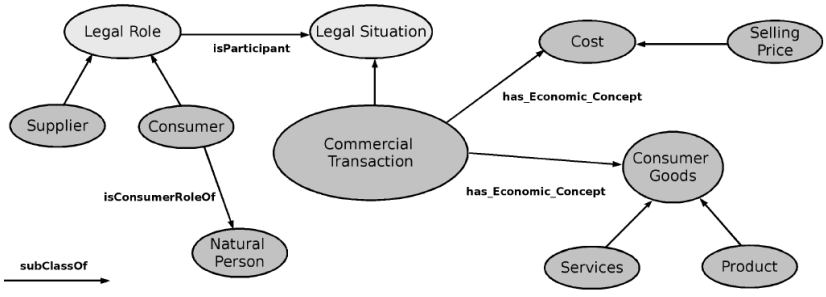


FIGURE 2.21: Excerpt of the DALOS Consumer Protection Ontology (Agnoloni et al., 2011).

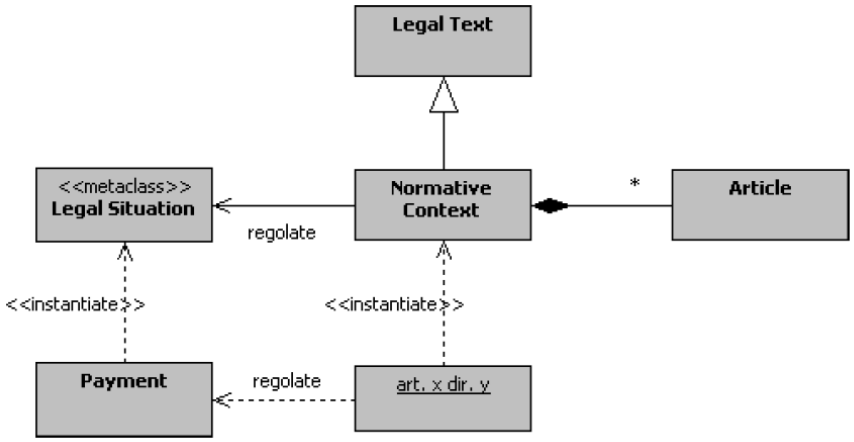


FIGURE 2.22: Contexts and regulated situations (Francesconi et al., 2008).

2.2.3.6 Ontology of Professional Judicial Knowledge (OPJK)

The Ontology of Professional Judicial Knowledge (OPJK) (Casellas, 2008a) is a legal ontology developed to map questions of junior judges to a set of stored frequently asked questions. The purpose of OPJK is searching and indexing for the JURISERVICE web-based application. OPJK ought to represent the relevant concepts related to the problems that take place during the on-call period in Spanish first instance courts. The ontology contains domain specific knowledge, professional knowledge, gathered by experience from the practice during on-call periods with semi-structured interviews.

The methodological approach applied for building OPJK comprise list of steps that have been established from the analysis of the most detailed and complete methodologies: 1) preparatory phase (specification of ontology requirements), 2) development phase (knowledge acquisition-experts, documents, reuse-, conceptualization-classes, relations, properties, instances-, expert validation and formalization), and 3) evaluation phase (internal consistency, requirements, competency questions and expert evaluation).

The construction process of OPJK and its formalization has been based on the terminology, information and knowledge contained in the corpus of questions. The judicial knowledge contained in these questions expresses practical problems faced by judges in their fist appointment during their on call court period. In order to conceptualize and formalize the terms extracted from the corpus, the modeling decisions are supported by course syllabus, legislation, and doctrine. Moreover, several upper and core ontologies, such as SUMO, PROTON, DOLCE/CLO or LKIF-Core, have been revised in order to obtain a set of top concepts for grouping and supporting middle-out conceptualization.

Thus, the main top classes of OPJK are: Role, Agent, Document, Process and Act (see Figure 2.23).

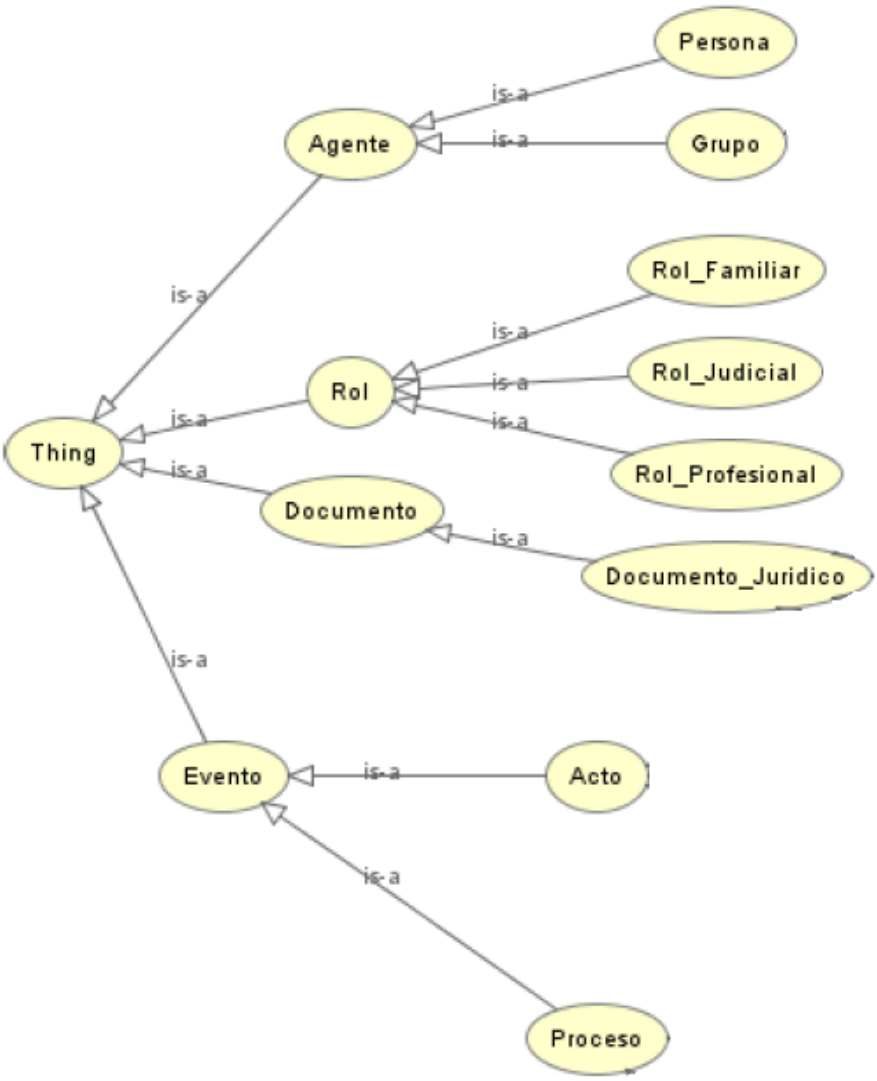


FIGURE 2.23: Excerpt of OPJK v 1.0 in Protégé (Casellas, 2008a).

OPJK version 1.0 includes 56 classes, 55 rdfs:subClassOf relations and 913 instances, together with a total of 25 owl:ObjectProperty axioms (10 owl:subPropertyOf and 12 owl:inverseOf), 1 transitive and 1 functional owl:ObjectProperty.

Furthermore, OPJK is integrated into PROTON (Proto Ontology)[5] offering some constraints towards the engineering process. This integration implies that the Ontology for Professional Judicial Knowledge should include the System Module and Top Module from PROTON.

[5]http://proton.semanticweb.org/, retrieved 7 September 2017

2.2.3.7 UFO-L

UFO-L (Ontology of Legal Concepts) is a legal core ontology (LCO), which represents essential concepts of the Law based on Alexy's Theory of Fundamental Rights (Griffo et al., 2015). This ontology is still under development, but we will overview its general aspects and methodology applied.

UFO-L is built as a layer on top of the Unified Foundational Ontology (UFO). UFO is an example of a descriptive foundational ontology that has been constructed for more than a decade employing results from formal ontology, cognitive psychology, linguistics, philosophical logics, but also significant accumulated empirical and theoretical results from the area of conceptual modeling in computer science. (Guizzardi, 2005).

UFO-L uses domain-independent concepts of domain provided by UFO. Extending these concepts, a conceptualization for legal domain is built, which can be used in other particular domain ontologies and legal knowledge bases. The main idea is to built ontological patterns in order to be used for supporting the modeling of legal concepts in conceptual models of the legal domain (see Figure 2.24).

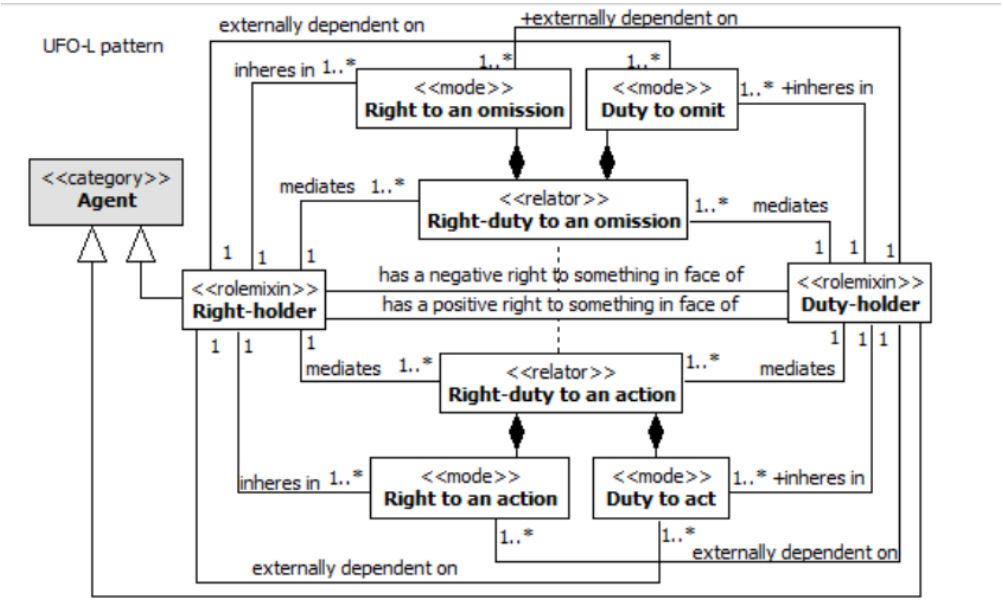


FIGURE 2.24: UFO-L pattern for right-duty relations (Griffo et al., 2016).

Ontology	Aim	Architecture
FOLaw	designing the architecture of ON-LINE for legal problem solving	normative knowledge, world knowledge, responsibility knowledge, reactive knowledge, creative knowledge, and meta-level knowledge.
LRI-Core	supporting knowledge acquisition for legal domain ontologies	physical, mental, roles, abstractions, occurrences
OCL.NL	e-Court	concepts anchored in LRI-Core
LKIF-Core	Estrella	Abstract concepts, basic, legal, framework, core and extended
DALOS Consumer Protection	Consumer protection Law with the DALOS European project	Two knowledge level: Ontological (reuse of CLO) and Lexical (ontology learning from texts).
OPJK	mapping questions of junior judges to FAQs	Main classes: Role, Agent, document, Process, Act
UFO-L	LCO based on Robert Alexy's Theory of Constitutional Rights, grounded on UFO	Ontological patterns

TABLE 2.2: A summary of Legal ontologies

2.2.4 Roles and Uses of Legal Ontologies

The need for the development of legal ontology is increasing together with the diversity of applications of legal knowledge management (Casanovas et al., 2007), ranging from document retrieving, data and text mining, modeling legal reasoning, development of decision support systems, etc (Mazzega et al., 2011) (Mommers, 2010). Five main roles of legal ontologies are identified in the literature (Valente, 2005), (Breuker et al., 2004b) and (Mommers, 2010):

- Organize and structure information: ontologies are used to define legal vocabularies which are typically used to define the terms used in regulations. They are considered as representations of the law domain, e.g. taxes, crime, traffic, immigration, etc. Examples of such ontologies: Jur-Wordnet ontology (Gangemi et al., 2003) and Italian Crime Ontology (Asaro et al., 2003).
- Reasoning and problem solving: ontologies aim to represent the knowledge of the domain so that an automated reasoner can represent problems and generate solutions for these problems. This use is found in the many expert systems and decision making systems developed in AI & Law. Examples of

such ontologies: CLIME Ontology (Boer et al., 2001), that is used as a basis for a legal advice system for maritime law, and Zeleznikow and Stranieri's ArgumentDeveloper (Zeleznikow et al., 2001), which was used in connection with several legal knowledge-based systems. By using these ontologies, there is a need for inference engine that is used to conclude specific goals. Valente et. al. (Valente et al., 1999b) argue that ontological choices are strongly influenced by the purpose of the ontology. That is, the same knowledge will be structured or formalized differently depending of how it will be used by the reasoner in reaching the desired conclusions in a specific context. This indicates that reusability is a good idea, but it can never be accomplished completely.

- Semantic indexing and search: ontologies can be used to represent and search semantically the content of documents to go beyond word or keywords. Two main works,(Benjamins et al., 2003) and (Saias et al., 2003), are examples of such ontologies.
- Semantics integration and inter-operation: The basic role of ontologies in this case is to support applications to exchange information electronically. This use is less common in the legal domain.
- Understanding the domain: ontologies in this case is to provide a view of what a domain is about. They work as a map that specifies what kinds of knowledge can be identified in the domain. These types of ontologies have been called core ontologies (Valente et al., 1996) such as FOLaw (Valente et al., 1994b).

There is a growing body of research and practice in constructing legal ontologies and applying them to the law domain (Valente, 2005). To build and maintain legal ontologies, proper techniques and methods from ontology engineering have been used: conceptual analysis, knowledge representation, ontology modularization and layering, ontology alignment and merging, evolution and dynamics, multilingual and terminological aspects, etc. (Benjamins et al., 2005a). We will outline the most common known and used methodologies.

2.2.5 Ontology Engineering Methodologies

When a new ontology is going to be built, several basic questions arise related to the methodologies, tools and languages to be used in its development process: (Corcho et al., 2003):

- Which methodologies have to be used for building ontologies, either from scratch, or reusing other existent ontologies?
- Which tool(s) give/s support to the ontology development process?
- Which language(s) should be used to implement the ontology?

In this section, the available methodologies for building ontologies will be outlined, and the tools and languages will be discussed in the following sections. Starting with a survey about the methodologies used to develop ontologies (see Figure 2.25) where 60% of the participants did not use any methodology to build their ontologies (Cardoso, 2007), and the methodologies with greatest adoption among ontologists are On-to-knowledge (Sure et al., 2003) and Methontology (Corcho et al., 2005). In the literature, several methodologies have been proposed to build ontologies. Some of them are designed for building ontologies from scratch or reusing other ontologies without modifying them. Generally, the ontology engineering methodologies are divided into two main generations. The first generation compose the methodologies that focused on core development process such as: (Uschold et al., 1995), (Swartout et al., 1997), and (Gruninger et al., 1995). The second generation of methodologies are the most complete ones, according to (Roussey et al., 2011), such as: Methontology (Corcho et al., 2005) and On-to-knowledge (Sure et al., 2003). These methodologies shifted the focus towards a more iterative process where the development process is not linear but a refinement one where each activity can be repeated several times.

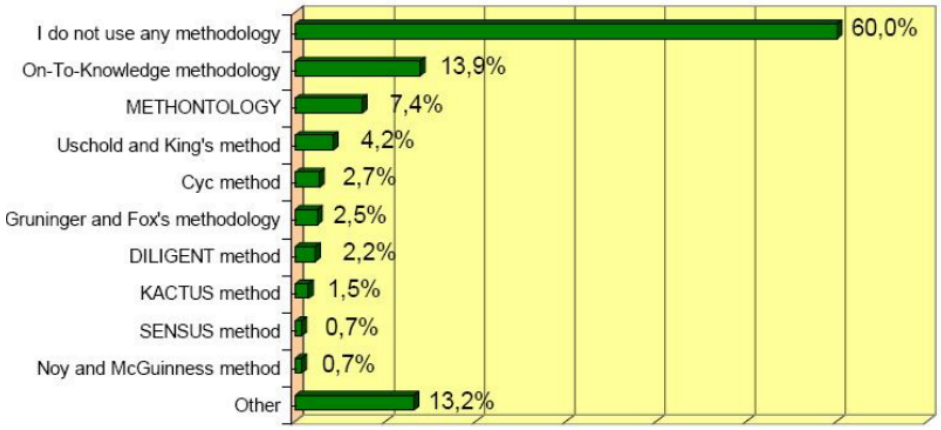


FIGURE 2.25: Survey about methodologies used to develop ontologies (Cardoso, 2007).

2.2.5.1 Uschold and colleagues

Uschold and King's (Uschold et al., 1995), Uschold and Gruninger (Uschold et al., 1996) offered a set of guidelines towards ontology construction and merging. According to Uschold and king, there are no standard methodologies for building ontologies. In an attempt to begin filling this gap, they proposed a comprehensive methodology for building ontologies four main stages:

- Identify purposes: identify why the ontology is being built and what are its intended uses and users.

- Building the ontology:
  - Ontology capture: defined in four main steps.
    1. identification of the key concepts and relationships in the domain of interest.
    2. production of precise unambiguous text definitions for such concepts and relationships.
    3. identification of terms to refer to such concepts and relationships.
    4. agreeing on all of the above.
  - Ontology coding: an explicit representation of the conceptualization captured in the ontology capture stage in some formal language. This will involve committing to some meta-ontology choosing a representation language and creating the code.
  - Integrating existing ontologies: during the capture and/or the coding stages, there is a possibility to use ontologies that already exist. The question is how and whether to use these ontologies.
- Evaluation: generally evaluation of ontologies could contribute to a comprehensive methodology for building ontologies. The current approach is to look

  first at what has been done in the field of KBS and to adapt it for ontologies.
- Documentation: establish guidelines for documenting ontologies possibly differing according to type and purpose of the ontology. Inadequate documentation of existing knowledge bases and ontologies is one of the main barriers to effective knowledge sharing (Skuce, 1995).

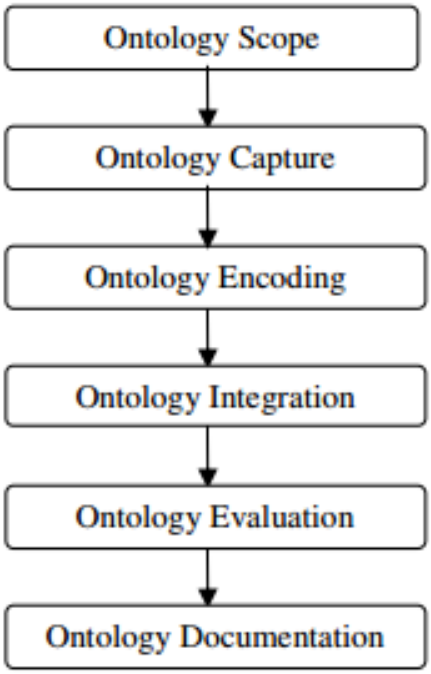


FIGURE 2.26: Uschold and King's methodology (Uschold et al., 1995).

2.2.5.2 CommonKADS

CommonKADS (Schreiber et al., 2000) is a methodology for building knowledge-based systems, but also offers useful features also for ontology engineering (Sure et al., 2004). The CommonKADS methodology for the construction of a knowledge model includes activities which could be relevant towards ontology construction.

- Knowledge Identification:
  - Domain familiarization (information sources, glossary, scenarios);
  - List of potential model components for reuse (task- and domain related components).
- Knowledge Specification
  - Choose task template (provides initial task decomposition, e.g., diagnosis, assessment, etc.);
  - Construct initial domain conceptualization (main domain information types): Domain-specific conceptualizations and Method-specific conceptualizations.
  - Complete knowledge-model specification (knowledge model with partial knowledge bases).

- Knowledge Refinement:
  - Validate knowledge model (paper simulation, prototype of reasoning system);
  - Knowledge-base refinement (complete the knowledge bases).

2.2.5.3 Methontology

Methontology is an ontology building methodology from scratch described in (Fernandez-Lopez et al., 1997), (Fernandez-Lopez et al., 2002), (Fernandez-Lopez, 1999) (Gomez-Perez et al., 2003a) and (Corcho et al., 2005). This methodology describes the different steps to be taken not only in the conceptualization process of an ontology but also during the ontology development life cycle.

The methodology takes into account development activities as well as management and support activities (see Figure 2.27) (control and quality assurance together with knowledge acquisition (from experts or semi-automatic ontology extraction), integration, evaluation (ontology verification, validation, and assessment), documentation and configuration management).

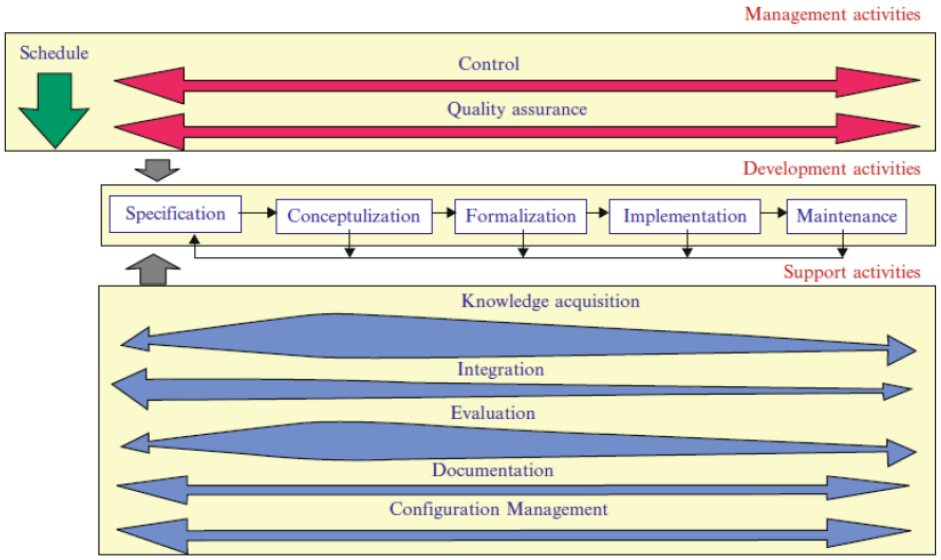


FIGURE 2.27: The Methontology methodology (Corcho et al., 2005).

Concerning the development, the main activities are:

- Specification: establishes informally or formally (competency questions) the purpose and scope of the ontology (why, what use, who are the end users).
- Conceptualization: organize the knowledge acquired (see Figure 2.28).

  - Build a glossary of terms (with definitions, synonyms and acronyms) following a middle-out strategy;
  - Classify terms into one or more taxonomies of concepts (understood as abstractions of one or more terms);
  - Define binary relations between the concepts;
  - Built the concept dictionary (class attributes);
  - Detail the concept dictionary (cardinality, inverse relations, properties, etc.)
  - Define axioms and rules.
- Formalization.
- Implementation.
- Maintenance.

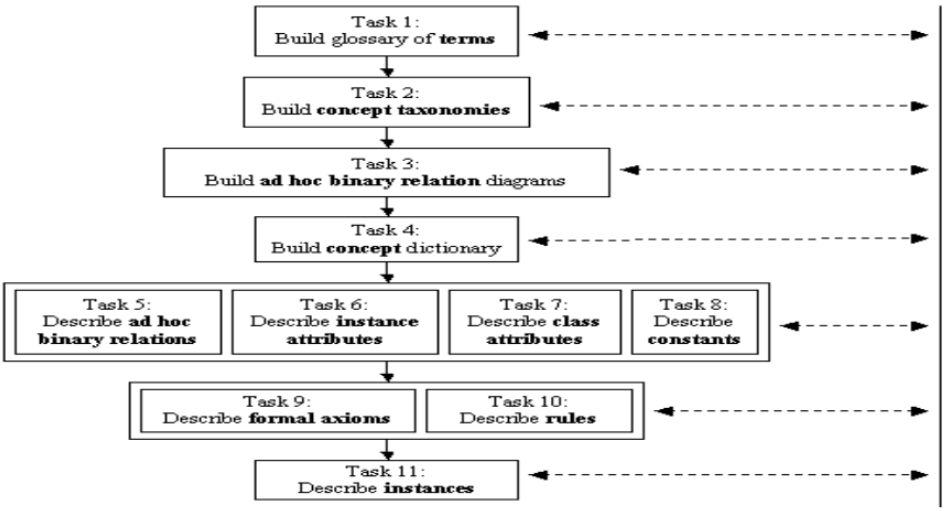


FIGURE 2.28: Tasks of the conceptualization activity according to METHONTOLOGY (Corcho et al., 2005).

Finally, several tools such as ODE and WebODE were built to give technological support to this methodology (Corcho et al., 2005), although several other existing tools may also be used such as Protégé, OntoEdit and KAON, etc.).

2.2.5.4 Ontology Development 101

Noy and McGuinness (Noy et al., 2001) offered a simple knowledge-engineering methodology in 7 steps *Ontology Development 101*. These steps are resumed in follows:

- Determine the domain and the scope of the ontology:
  - What is the domain that the ontology will cover?
  - For what we are going to use the ontology?
  - For what types of questions (competency questions) the information in the ontology should provide answers?
  - Who will use and maintain the ontology?
- Consider reusing existing ontologies
- Enumerate important terms in the ontology.
- Define the classes and the class hierarchy. Several approaches:
  - Top-down
  - Bottom-up
  - Combination
- Define the properties of classes-slots, such as:
  - *intrinsic* properties
  - *extrinsic* properties
  - parts, if structured
  - relationships to other individuals
- Define the facets of the slots, such as:
  - Slot value-type: string, number, boolean slots, enumerated slots, instance-type slots.
  - Domain and range.
- Create individual instances of classes in the hierarchy.

2.2.5.5 ON-TO-KNOWLEDGE Methodology (OTKM)

The OTKM is described in (Sure et al., 2002a) and (Sure et al., 2003). This methodology was influenced by the methodologies of Uschold and his colleagues, CommonKADS and METHONTOLOGY.

The ontology building process in OTKM is divided into five steps:

1. Feasibility study: identify stakeholders (users and supporters of the system), identify uses cases describing usage scenarios and their supporting uses cases.
2. Ontology Kickoff: initiates the development of the ontology,

- Description of an Ontology Requirements Specification Document (ORSD), which includes:
  - Goal, domain and scope of the ontology;
  - Design guidelines;
  - Knowledge sources;
  - (Potential) users and usage scenarios;
  - Competency questions;
  - Supported applications.
- Analysis of knowledge sources (build initial lexicon);
- Create a semi-formal description of ontology (draft).

3. Refinement: Knowledge is acquired and formalized in a cyclic approach.
   - Knowledge elicitation process with domain experts (based on input from kickoff phase), modification or extension of draft ontology;
   - Formalization of target ontology.
4. Evaluation:
   - Technology-focused evaluation: mainly consistency and language conformity checking.
   - User-focused evaluation: assessment of the requirements specified and the competency questions established in the resulting ontology, and testing results from prototype application.
   - Ontology-focused evaluation: formal analysis of ontologies (e.g., OntoClean evaluation methodology).
5. Application.

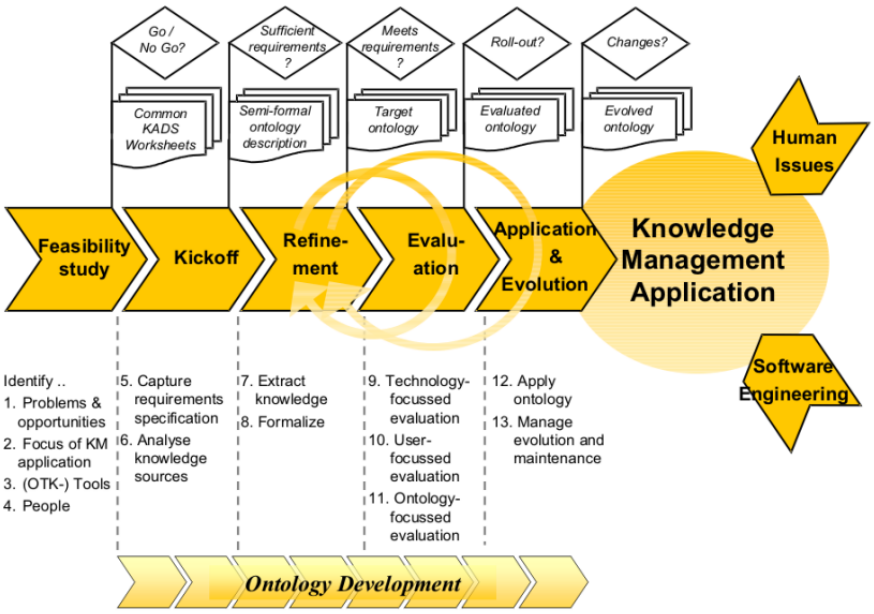


FIGURE 2.29: The methodological process of OTKM (Sure et al., 2003).

OTKM is supported by the OntoEdit ontology development tool.

2.2.5.6 SABiO: Systematic Approach for Building Ontologies

The Systematic Approach for Building Ontologies (SABiO) (Falbo, 2014) is an ontology engineering approach that focus on the development of domain ontologies, and also propose support processes. SABiO distinguishes between reference and operational ontologies, providing activities that apply to the development of both types of domain ontologies.

Domain reference ontology means a domain ontology that is built with the goal of making the best possible description of the domain. It is a solution-independent specification (conceptual model) with the aim of making a clear and precise description of domain entities for the purposes of communication, learning and problem-solving. Meanwhile, domain operational ontologies are machine-readable implementation version of the ontology, designed with the focus on guaranteeing desirable computational properties (Fielding et al., 2004). Thus, before implementing an operational ontology, a design phase should be accomplished taking technological non-functional requirements and the ontology implementation environment into account.

SABiO has been used for building several domain ontologies, such as ontologies for the software process and cardiology domains. The development process of SABiO comprises five main phases (see Figure 2.30): (1) Purpose identification and requirements elicitation; (2) ontology capture and formalization; (3) design; (4)

implementation; and (5) test. Support processes are performed in parallel to the development process.

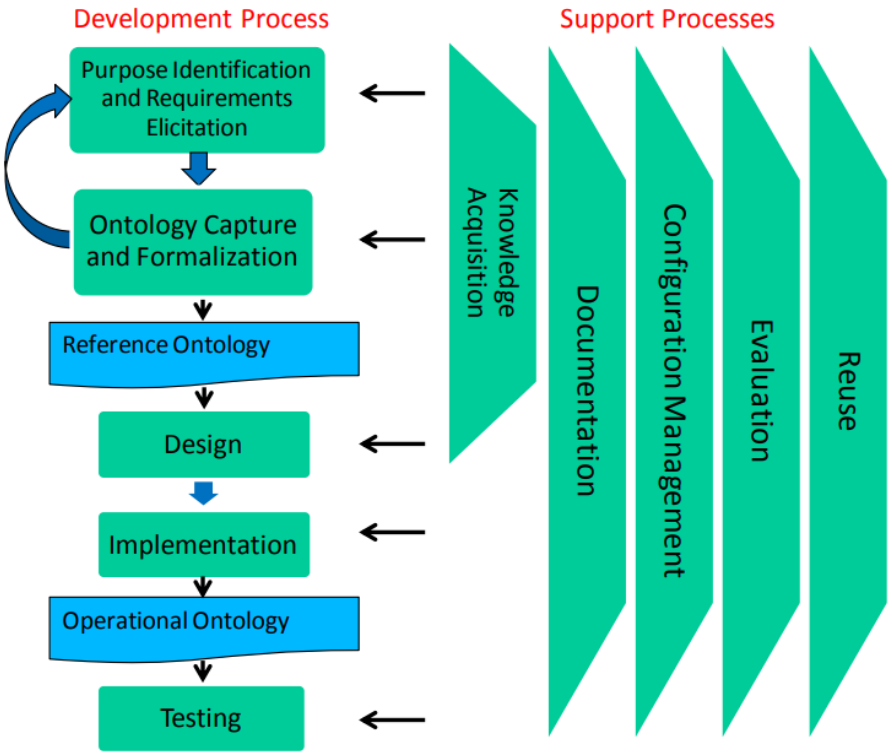


FIGURE 2.30: Ontology engineering process of SABiO (Falbo, 2014).

the first three activities of the development process are accomplished if someone is interested in building a domain reference ontology. However, for building an operational ontology, then the entire development process should be performed.

In Table 2.3, the ontology engineering methodologies are summarized by mentioning the building process.

Methodology	Building process
Uschold and colleagues	Purpose, building, evaluation and documentation
CommonKADS	Identification, ipecification, refinement
Methontology	Specification, conceptualization, formalization, implementation, maintenance
Ontology Development 101	Scope, reuse, hierarchy, properties, domain, range, instaces
OTKM	Feasibility, requirement, analysis, draft, refinement, formalization
SABiO	Purpose identification and requirements elicitation, ontology capture and formalization, design, implementation, testing

TABLE 2.3: A summary of Ontology development methodologies.

2.2.6 Ontology Engineering Tools and Environments

As aforementioned, the ontology engineering methodologies are needed for ontology development. However, a methodology itself is not sufficient. Developers need an integrated environment or tool in order to help them while building the intended ontology in every phase of the building process (Mizoguchi et al., 2009). In the literature, there is a diversity of ontology engineering tools and environments that support the ontology building process, specifically for ontology learning or acquisition and editing. In fact, Ontology acquisition is considered as an essential step in building ontologies. Knowledge can be acquired using ontology learning or editing tools. In the following section, we will outline briefly the most relevant tools and environments for ontology development based on some extensive revisions for (Gomez-Perez, 1999), (Gomez-Perez et al., 2003a).

Ontolingua Server Ontolingua server [6] was the first ontology tool created at the beginning of 1990s at Stanford university. Initially, the main module inside the Ontolingua Server was the ontology editor, then other modules were included in the environment, such as a Webster, an equation solver, and Chimaera (an ontology merging tool). The ontology editor also provides translators to languages, such as Prolog, and CLIPS.

[6] http://ksi.cpsc.ucalgary.ca/KAW/KAW96/farquhar/

OntoEdit OntoEdit[7] (Sure et al., 2002b) has been developed by AIFB in Karlsruhe University. It is similar to the previous tools: it is an extensible and flexible environment, based on a plug-in architecture, which provides functionality to browse and edit ontologies.

Protégé Protégé[8] is an extensible, platform-independent environment for creating and editing ontologies and knowledge bases developed by the Stanford Medical Informatics (SMI) at Stanford University. It is an open source, standalone application with an extensible architecture.

The core of this environment is the ontology editor, and it holds a library of modules that can be plugged, called plug-ins, to add more functions to the environment. The main Protégé functions are to: load and save OWL and RDF ontologies; edit and visualize classes, properties, and SWRL rules; define logical class characteristics as OWL expressions; execute reasoners such as description logic classifiers; and edit OWL individuals for Semantic Web markup. Protégé is available in different versions, each including different plug-ins, whose main difference is the ontology language that they support:

- Protégé version 3 supports OWL 1.0, RDF(S) and Frames.
- Protégé versions 4 and 5 supports OWL 2.0.

Neon Toolkit The Neon Toolkit [9] is an ontology engineering environment that supports the complete life cycle of large-scale ontology networks (Haase et al., 2008). In order to support such a broad ontology modeling functionality, it has an open and modular architecture, which the NeOn Toolkit inherits from its underlying platform, Eclipse.

Eclipse is a very rich development environment, which is widely adopted in the programming world and which perfectly fits to the modeling paradigm for ontologies. It provides developers with a framework to easily create, publish and integrate new features into the NeOn Toolkit. A substantial number of so-called plugins has been developed within and outside the NeOn consortium and are available at NeOn Toolkit homepage. The NeOn Toolkit is available as an installable core version with the basic ontology functionality such as editing, browsing, ontology and project management. Currently, the following versions are available:

- The basic NeOn Toolkit provides the core functionality for handling OWL 2 ontologies.

[7] http://ontoserver.aifb.unikarlsruhe.de/ontoedit/
[8] http://protege.stanford.edu/
[9] http://neon-toolkit.org/

- The NeOn Toolkit extended configuration includes advanced functionality for managing rule based models and ontology mapping facilities based on commercial extensions.

WebODE WebODE[10], a successor of ODE (Blazquez et al., 1998), is an ontological engineering workbench developed by the Ontological Engineering group at Universidad Politécnica de Madrid (UPM) in order to give technological support to most of the activities involved in the ontology development process proposed by METHONTOLOGY (Arpirez et al., 2003). Although this does not prevent it from being used with other methodologies or without following any methodology (Corcho et al., 2003).

The editor is a Web application built on top of the ontology access service (ODE API), which integrates several ontology building services from the workbench: ontology edition, navigation, documentation, merge, reasoning, etc.

OntoUML Lightweight Editor (OLED) The OntoUML lightweight editor (OLED) [11] is an environment for the development, evaluation and implementation of domain ontologies using the UFO-based ontologically well-founded modeling language OntoUML (Guerson et al., 2015). OLED is developed by the Nemo research group [12]. UFO, the Unified Foundational Ontology, is a foundational ontology that provides a sound ontological basis to evaluate and give real-world semantics to conceptual modeling language's constructs such as UML (Guerson et al., 2015). OntoUML (Guizzardi, 2005) is an ontologically well-founded profile of the class diagram fragment of UML 2.0. OntoUML's categories are put forth by UFO.

Modelers specify their domain ontologies in OntoUML, constraining them using the Object Constraint Language (OCL) [13]. They can import models designed with EA into OLED. The tool provides a set of built-in design patterns to speed up the modeling activity through re-use. To improve the quality of the models built using OLED, it provides an automatic syntax verification alongside two complementary validation features, visual simulation (Benevides et al., 2011) and anti-patterns. To apply the knowledge formalized in the OntoUML in semantic web applications, OLED features a number of predefined automatic transformations to the Web Ontology Language (OWL) (possibly enhanced with SWRL rules) (Albuquerque, 2013) (Barcelos et al., 2013).

[10] http://webode.dia.fi.upm.es/webODE/

[11] https://nemo.inf.ufes.br/projects/oled/, https://code.google.com/p/ontouml-lightweight-editor/, retrieved September 7 2017

[12] https://nemo.inf.ufes.br/, retrieved 7 September 2017

[13] http://www.omg.org/spec/OCL/2.4/

Tool	Functionalities	Pricing policy	Standards
Ontolingua	Ontology editor, support building of shared ontologies geographically seperated	Free web access	Ontolingua
OntoEdit	support OTKM	Freeware and licenses	XML, RDF(S), FLogic, DAML+OIL
Protégé	A graphical ontology editor and knowledge base framework for ontology manipulation and query	Open source	RDF, RDFS, OWL, OWL2
Neon Toolkit	supports the complete life cycle of large-scale ontology networks	Open source	RDF, RDFS, OWL, OWL2
WebODE	support METHONTOLOGY	Free Web access Licenses	XML, RDF(S)
OLED	model-based environment to support Ontology Engineering in OntoUML	Open access	OntoUML, OWL, SWRL

TABLE 2.4: A comparison of Ontology engineering systems.

2.2.7 Ontology Languages and Formalisms

In the ontology engineering process, there is a need for ontology languages in order to implement expressively the ontology and make it processable and understandable by the machine. Different ontology languages having different expressiveness and inference mechanisms are found in the literature. A major decision to take is to select the appropriate language. AI-based Ontology implementation languages started to be created at the beginning of the 1990s (Corcho et al., 2006) such as KIF (Genesereth et al., 1992) (based on first-order logic as a knowledge representation (KR) formalism), FLogic (Kifer et al., 1995) (based on frames combined with first-order logic), Loom (MacGregor, 1991) (based on description logics), etc.

Furthermore, the web-based ontology languages (see Figures 2.31 and 2.32), or ontology markup languages, appeared such as SHOE (Luke et al., 2000), XOL (Karp et al., 1999), OIL (Horrocks et al., 2000), DAML+OIL (Horrocks et al., 2001), RDF (Lassila et al., 1999), RDF Schema (Brickley et al., 2004), OWL (Dean et al., 2004)

and OWL2 (Motik et al., 2009). From these ontology languages, the ones that are based on description logics (DL) and supported now and that we will overview in the following are: RDF, RDFS, OWL and OWL2.

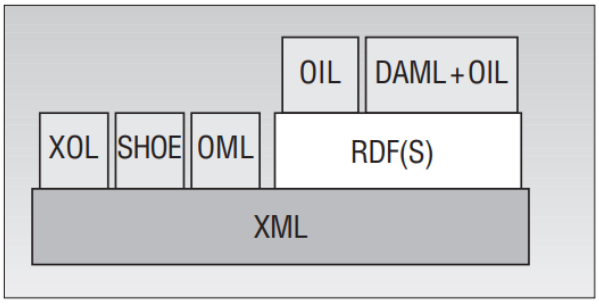


FIGURE 2.31: The languages stack in the Semantic Web (Corcho et al., 2003).

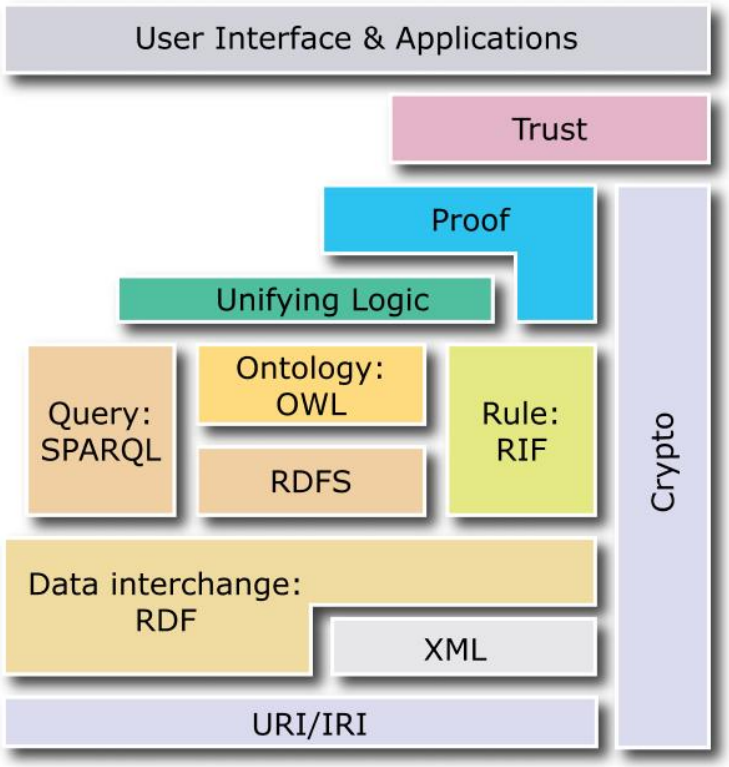


FIGURE 2.32: W3C Semantic Web stack.

2.2.7.1 RDF

RDF stands for Resource Description Framework. It was developed by the W3C (World Wide Web Consortium) to create metadata for describing web resources and its data model is equivalent to the semantic networks formalism, consisting of three object types: resources, properties and statements. RDF is just a data model; it does not have any significant semantics (Smith et al., 2004). In RDF

models, the classes used to type resources and the properties of these classes can be defined using a vocabulary such as `rdfs:Class`, `rdf:Property`, `rdfs:subClassOf`, `rdfs:subPropertyOf`, `rdfs:domain` and `rds:range`.

2.2.7.2 RDF Schema

The RDF data model does not have mechanisms for defining the relationships between properties and resources. This is the role of the RDF Vocabulary Description language also known as RDF Schema. RDF(S) is the term commonly used to refer to the combination of RDF and RDFS. Thus, RDF(S) combines semantic networks with frames but it does not provide all the primitives that are usually found in frame-based knowledge representation systems. Summary of basic features of RDF Schema:

- Classes and their instances.
- Binary properties between objects.
- Organization of classes and properties in hierarchies.
- Types for properties: domain and range restrictions.

2.2.7.3 OWL

OWL, W3C Web Ontology Language, is a semantic web language designed for encoding and exchanging ontologies. It is designed for use by applications that need to process formally the content of information instead of just creating standard terms for concepts as is done in XML. Every OWL document is RDF document. Semantically, OWL is based on description logics (Baader et al., 2002) and is used to formalize a domain by defining terminology that can be used in RDF documents such as classes and properties. Classes are declared explicitly in OWL as `rdf:type owl:Class`. OWL can also define two types of properties: object properties and datatype properties. Object properties specify relations between pairs of classes. Datatype properties specify a relation between a class and a data type value.

- Define instances called individuals and assert properties about them.
- Reason about these classes and individuals to the degree permitted by the formal semantics of the OWL language.

In OWL, taxonomies can be specified for both classes and properties. The OWL language consists of three increasingly expressive sub-languages: OWL Lite, OWL DL and OWL Full. OWL Lite is intended for users with simple modeling needs. OWL DL has the closest correspondence to an expressive description logic. OWL Full is meant for users who want maximum expressiveness.

OWL is grounded on Description Logics (Baader et al., 2002) and has several inference engines that can be used for constraint checking of concepts, properties and instances, and for automatic classification of concepts into hierarchies. Its semantics are described in two different ways: as an extension of the RDF(S) model theory and as a direct model-theoretic semantics of OWL. Both of them have the same semantic consequences on OWL ontologies.

2.2.7.4 OWL 2

OWL 2, a Semantic Web KR language based on description logics (DLs), is the successor of OWL and has a very similar overall structure to OWL. OWL 2 adds new features such as keys, property chains, richer datatypes, asymmetric, reflexive and disjoint properties, enhanced annotation capabilities, etc. [14]. The language family of OWL 2 is composed of OWL 2 Full and OWL 2 DL. OWL 2 Full interprets any RDF graph under OWL-RDF entailment regime (undecidable). OWL 2 DL, fragment of first-order predicate logic (FOL), interprets OWL 2 ontologies by means of decidable SROIQ description logic semantics.

OWL 2 also defines three new profiles or sub-languages that trade some expressive power for the efficiency of reasoning [15]: OWL 2 EL, OWL 2 QL and OWL 2 RL. The three profiles are subsets of OWL 2 DL.

- OWL 2 EL: is particularly suitable for applications where very large ontologies are needed, and where expressive power can be traded for performance guarantees.
- OWL 2 QL: is particularly suitable for applications where relatively lightweight ontologies are used to organize large numbers of individuals and where it is useful or necessary to access the data directly via relational queries (e.g., SQL).
- OWL 2 RL: is particularly suitable for applications where relatively lightweight ontologies are used to organize large numbers of individuals and where it is useful or necessary to operate directly on data in the form of RDF triples. Reasoning systems for ontologies in the OWL 2 RL (Rule Language) profile can be implemented using rule-based reasoning engines.

2.2.7.5 Description Logics (DL)

In this section, We briefly introduce Description Logics (DLs), which are the logic-based knowledge representation (KR) formalisms behind the OWL family of web ontology languages and designed to represent and reason about knowledge in

[14] https://www.w3.org/TR/owl2-overview/, retrieved 7 September 2017
[15] https://www.w3.org/TR/owl2-profiles/, retrieved 7 September 2017

a structured and well-understood way (Baader et al., 2003a). DLs are decidable fragments of first-order logic (FOL) and widely used in ontological modeling (Krotzsch et al., 2014). Notably, DLs are essentials in the design of OWL (description logic SHOIN, 2004) and OWL2 (description logic SROIQ, 2009), the current standard language to represent ontologies. DLs provide the techniques to model relationships between individuals of a given domain (Krotzsch et al., 2014). DLs are based on a common family of languages called description languages, which provide a set of constructors to build concept (`class`) and role (`property`) descriptions. Such descriptions can be used in axioms and assertions of DL knowledge bases, or ontologies, and can be reasoned about DL knowledge bases by DL systems (Baader et al., 2005). Reasoning allows one to infer implicitly represented knowledge from the knowledge that is explicitly contained in the knowledge base (Baader et al., 2003a).

Description Logics support inference patterns that occur in many applications of intelligent information processing systems, and which are also used by humans to structure and understand the world: classification of concepts and individuals.

Classification of concepts determines subconcept/superconcept relationships (called `subsumption` relationships in DL) between the concepts of a given terminology. This hierarchy provides useful information on the connection between different concepts, and it can be used to speed-up other inference services.

Classification of individuals (or objects) determines whether a given individual is always an instance of a certain concept (i.e., whether this instance relationship is implied by the description of the individual and the definition of the concept). It thus provides useful information on the properties of an individual. Moreover, instance relationships may trigger the application of rules that insert additional facts into the knowledge base (Baader et al., 2003b).

Typically, a DL ontology consists of a set of statements called axioms that can be categorized into three groups: TBox, ABox, and RBox (Krisnadhi et al., 2014).

TBox and RBox axioms describe intensional knowledge about concepts and roles respectively. TBox axioms describe relationships between concepts. The basic TBox axioms are concept inclusion as in

$$\texttt{Mother} \sqsubseteq \texttt{Parent},$$

this assertion is used to states that the concept `Mother` is subsumed by the concept `Parent`.

Concept equivalence asserts that two concepts have the same instances such as

$$\texttt{Person} \equiv \texttt{Human}.$$

To describe more complicated situations, DLs allow building new complex concepts expressions using various concept constructors in order to describe relationships such as disjointness which asserts that two concepts do not share any instances.

RBox axioms refer to properties of roles. Role inclusion and role equivalence are role assertions. Role inclusion such as

$$\texttt{parentOf} \sqsubseteq \texttt{ancestorOf},$$

which asserts that `parentOf` is a subrole of `ancestorOf`: every pair of individuals related by `parentOf` is also related by `ancestorOf`.

The two roles `parentOf` and `ancestorOf` are disjoint. This axiom is written as follows:

$$\texttt{disjoint(parentOf, ancestorOf).}$$

Concerning the complex roles, DLs provide inverse role, universal role and empty role. Other RBox axioms are provided such as role transitivity, symmetry, asymmetry, reflexivity and irreflexivity.

ABox axioms describe specific knowledge in the form of membership of an individual in a concept and relationships between individuals through a role. The most common ABox axioms are concepts assertions such as

$$\texttt{Mother(Julia)},$$

which asserts that the individual named `Julia` is an instance of the concept `Mother`.

Role assertions describe relations between named individuals such as

$$\texttt{parentOf(Julia, John)},$$

which asserts that the individual named `Julia` is in relationship represented by `parentOf` to the individual named `John`.

Individual inequality assertions such as

$$\texttt{Julia} \neq \texttt{John},$$

are used to assert that `Julia` and `John` are different individuals. In contrast, Individual equality assertions such as

$$\texttt{John} \approx \texttt{Johnny},$$

are used to states that two different names refer to the same individuals.

2.2.8 Ontology Engineering Support Processes

All the methodologies discussed in section 2.2.5 were proposed for building ontologies. However, many others have been proposed for supporting the ontology

building process, such as ontology learning, ontology modularization, ontology reuse, ontology re-engineering , ontology evaluation, ontology merging , etc.

In this section, the most relevant ontology support processes that may an ontology engineer needs during the development process of his intended ontology are discussed.

2.2.8.1 Ontology Learning

The term *Ontology Learning* (OL) was introduced by Madche and Staab (Maedche et al., 2001) and is considered as an important task in Artificial Intelligence, Semantic Web and Knowledge Management. It can be described as the acquisition of a domain model from data (Cimiano, 2006). More specifically, OL is considered as a subtask of Information Extraction (IE), which is a type of Information Retrieval (IR) (Rogger et al., 2010). The main purpose of OL process is to apply methods from various fields such as linguistic analysis, machine learning, knowledge acquisition, statistics and information retrieval in order to extract knowledge from texts and support the construction of ontologies.

Knowledge acquisition techniques, usually supported by machine learning and natural language processing, can be used for implementing taxonomies or suggesting concepts for upper level ontologies, mainly hand-crafted by domain experts, as well as for identifying and representing legal rules. Ontology learning needs input data from which to learn the concepts relevant for a given domain, their definitions as well as the relations holding between them (Cimiano, 2006). OL is considered as a dynamic process of building ontologies. This dynamic process, depicted in the Figure 2.33, takes as input implicit and unstructured knowledge and produces as output explicit structured knowledge (Cimiano et al., 2005a). Purely automatic learning approaches will fail to generate ontologies which are good enough for a particular, e.g. reasoning-based, application (Lehmann et al., 2014). Generally, OL is a semi-automatic process where the ontology engineer and the domain expert can be involved to achieve better results. Thus, the techniques used in the ontology development process will be under their supervision. Their expertise and background knowledge helps in verifying the obtained information and decide the valuable information.

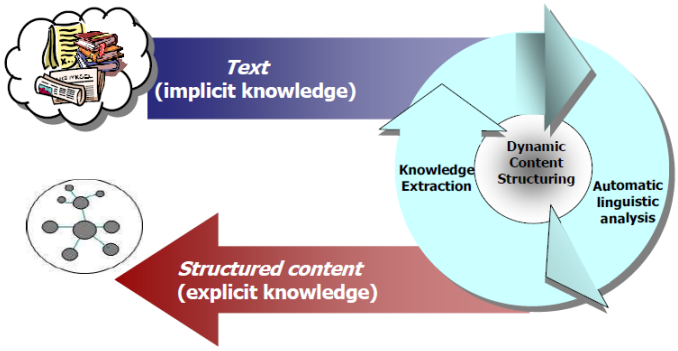


FIGURE 2.33: Ontology Learning process.

- Input: Ontologies can be learn, by applying the OL process, from various sources of data types: structured (e.g. databases, existing ontologies and knowledge bases), semi-structured (e.g. XML and WordNet) and unstructured natural language textual documents such as Word, PDF and Web pages. The unstructured type is the most available format as input for ontology learning processes. They reflect mostly the domain knowledge for which the user is building the ontology. In addition, they describe the terminology, concepts and conceptual structures of the given domain. However, some authors, such as (Rogger et al., 2010), consider that processing unstructured data is the most complicated problem because most of the knowledge is implicit and allows conceptualizing it by different people in different manner.
- Output: Ontology learning from text is the process of deriving concepts, relations and axioms from textual resources to build ontologies. The main output of the OL process is a structured content represented in an explicit formal way. For (Cimiano et al., 2004), the tasks in ontology learning from text are organized in a set of layers (see Figure 2.34) known as *ontology learning layer cake* (Cimiano, 2006). These tasks aim at returning six main outputs: terms, synonyms, concepts, taxonomic relations, non-taxonomic relations and axioms. These outputs represent the main elements of ontology.

For illustration purposes, Figure 2.34 includes some concrete examples from the domain of medicine on the left of each layer.

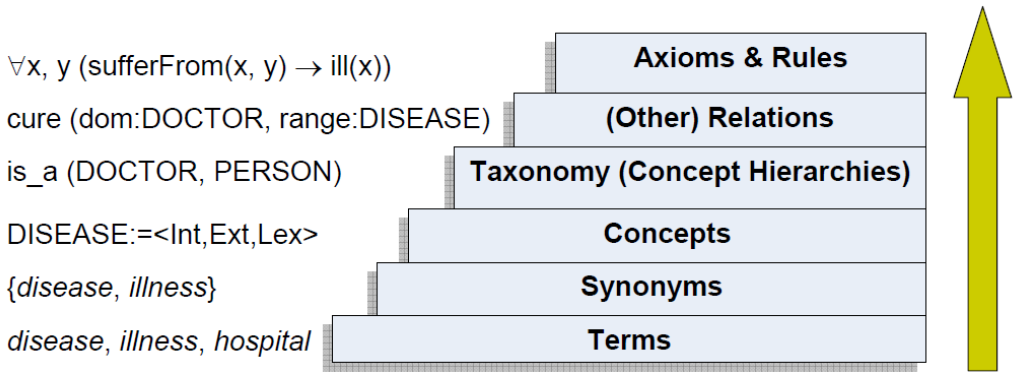


FIGURE 2.34: Ontology learning from text, layer cake (Buitelaar et al., 2005a).

Ontology Learning Tasks and Methods In this section, the ontology learning tasks will be described along with the lines of ontology learning layer cake (Buitelaar et al., 2005b) (Cimiano, 2006).

1. Terms: The term extraction task is a prerequisite for all aspects of ontology learning from text (Cimiano, 2006). Terms are linguistic realizations of domain-specific concepts and are therefore central to further more complex tasks. From a linguistic point of view, terms are either single or multi-word compounds with a very specific meaning in a given domain. For Cimiano, term is defined as *any single or multi-word compound relevant for the domain in question as a term*. The input of this task is a collection of documents representing the domain of interest, while output is a set of strings S_C and S_R representing terms which will be used as signs for concepts and relations respectively (Cimiano, 2006).

   Concerning the term extraction methods, the literature provide many examples that could be used as a first step in ontology learning from texts. Most of these methods are are based on information retrieval methods for term indexing (Salton et al., 1988), but many also are inspired by terminology and NLP research such as (Bourigault et al., 2001) (Pantel et al., 2001).

2. Synonyms: Generally, synonyms discovery consists of finding words which denote the same concept. The synonym level addresses the acquisition of semantic term variants in and between languages, where the latter in fact concerns the acquisition of *term translations* (Buitelaar et al., 2005a). For Cimiano (Cimiano, 2006), two words are considered as synonyms if they share a common meaning which can be used as a basis to form a concept relevant to the domain in question.

   For synonyms extraction, most of the works have focused on the integration of WordNet[16] for the acquisition of English synonyms, and EuroWordNet[17]

[16] http://wordnet.princeton.edu
[17] http://www.elda.fr

for bilingual and multilingual synonyms and term translations (Buitelaar et al., 2005a).

Meanwhile, other works have focused on algorithms for dynamic acquisition of synonyms using different techniques and methods such as clustering and related techniques, particularly Harri's hypothesis that consider words are semantically similar to the extent to which they share linguistic context (Harris, 1968). Other important techniques are cited in the literature LSI (Latent Semantic Indexing) (Landauer et al., 1997) and PLSI (Probabilistic Latent Semantic Indexing) (Hofmann, 1999). Finally, synonyms are detected on the web using statistical information measures (Baroni et al., 2004).

3. Concepts: According to Buitelaar and his colleagues (Buitelaar et al., 2006), *concept formation* should ideally provide:

   - An intensional definition of concepts
   - Set of concept instances, i.e. its extension
   - A set of linguistic realizations, i.e. (multilingual) terms for this concept.

   Thus, a concept is defined as a triple < i(c), [c], Ref_C(c) > (Cimiano, 2006).

   The task of concept extraction from text is considered as difficult and controversial since it is not clear what a concept extraction is supposed to be. Some works considered clustered of related terms as concepts such as (Lin et al., 2002). Other works, such as (Evans, 2003) have addressed concept formation from an extensional point of view. Finally, some systems learn concepts intensionally such as OntoLearn (Velardi et al., 2005).

4. Concept Hierarchies: The concept hierarchy is defined by Cimiano as *the tasks related to inducing, extending and refining the ontology's backbone*. Three main paradigms are exploited to induce concept hierarchies from texts:

   - Lexico-syntactic patterns.
   - Harris' distributional hypothesis using hierarchical clustering algorithms (Cimiano et al., 2005a).
   - Analysis of co-occurrence of terms in the same sentence, paragraph or document (Sanderson et al., 1999).

5. Relations: Four main tasks can be distinguished in the *relation learning* task (Cimiano, 2006):

   - Finding concepts in *C* standing in non-taxonomic ontological relation,
   - Specifying *R*, i.e. finding appropriate labels and relation identifiers on the basis of te given corpus,

- Given a certain relation r $\in R$, determining the right level of abstraction with respect to the concept hierarchy for the domain and range of the relation,
- Learning a hierarchical order $\leq_R$ between the relations in R.

In the literature, few approaches have addressed the issue of learning ontological relations from texts such as the use of association rules extraction algorithm based on sentence-based term co-occurrence (Madche et al., 2000), the use of syntactic dependencies (Gamallo et al., 2002).

6. Axioms and Rules: The task of learning axioms can be understood as consisting in deriving more complex relationships and connections between concepts and relations (Cimiano, 2006). These axioms can be represented using the Horn-fragment of first-order logic. Initial blueprints of this task can be found in (Lin et al., 2001) (Haase et al., 2005) (Shamsfard et al., 2004).

Ontology Learning Approaches There are many approaches in the literature that deal with ontology learning from textual resources (Gomez-Perez et al., 2003b). The most relevant approaches are, among others, the ones proposed by Aussenac-Gilles and her colleagues (Aussenac-Gilles et al., 2000), Bachimont (Bachimont et al., 2002), Sabou and her colleagues (Sabou et al., 2005)and Maedche and Staab (Maedche et al., 2001).

Aussenac-Gilles and colleagues's Approach Aussenac-Gilles and her colleagues proposed an ontology learning approach based on knowledge elicitation from technical documents (Aussenac-Gilles et al., 2000). The approach allows creating a domain model by analyzing of a given corpus using natural language processing (NLP) tools and linguistics techniques. Thus, the central role in this method is given to the textual resources.

Therefore, the approach combines knowledge acquisition tools based on linguistic with modeling techniques that allows keeping links between models and texts.

The ontology learning process in this approach is composed of four main activities:

- Corpus constitution: Texts are selected among the available technical documentation from the ontology requirements. The authors recommend that the selection of texts be made by an expert in texts of the domain. Also according to the authors, the corpus has to cover the entire domain specified by the application. To perform this activity is very useful to have a glossary of terms of the domain. Thus, the expert selects texts containing the terms of the glossary.
- Linguistic study: This activity consists in selecting adequate linguistic tools and techniques and applying them to the texts. The main difficulty is to select

the tools to be used, which strongly depend on the language to be processed. As a result of this activity, domain terms, lexical relations, and groups of synonyms will be obtained.

- Normalization: The result of this activity is a conceptual model expressed by means of a semantic network. This conceptual model is rather informal, however, it can be easily understood by the ontology designer. Normalization includes a linguistic step and a conceptual modeling step.
- Formalization: It includes ontology validation and implementation.

Bachimont's Approach The ontology learning approach proposed by Bachimont (Bachimont et al., 2002) is based on a claim that *an ontology has to introduce knowledge primitives which will be the building blocks for programming a Knowledge-Based System (KBS)*. From this perspective, the approach consists of three main steps (see Figure 2.35):

- Semantic normalization: The goal of the first step is to reach a semantic agreement about the meaning of the labels used for naming the concepts. The ontologist has to choose the relevant terms of a domain and specify their meaning, expressing the similarities and differences of each notion with respect to its neighbors: its parent-notion and its siblings-notions. The result is a taxonomy of notions where the meaning of a node is given by the gathering of all similarities and differences attached to the notions found on the way from the root notion (the more generic) to this node.
- Formalization: The ontological tree obtained in the first step allows to disambiguate the notions and to clarify their meanings for a domain-specific

  application. Hence, the user can constrain the domains of a relation, define new concepts, add properties to these concepts or add general axioms.
- Operationalization: this step transcribes the ontology into a specific knowledge representation language.

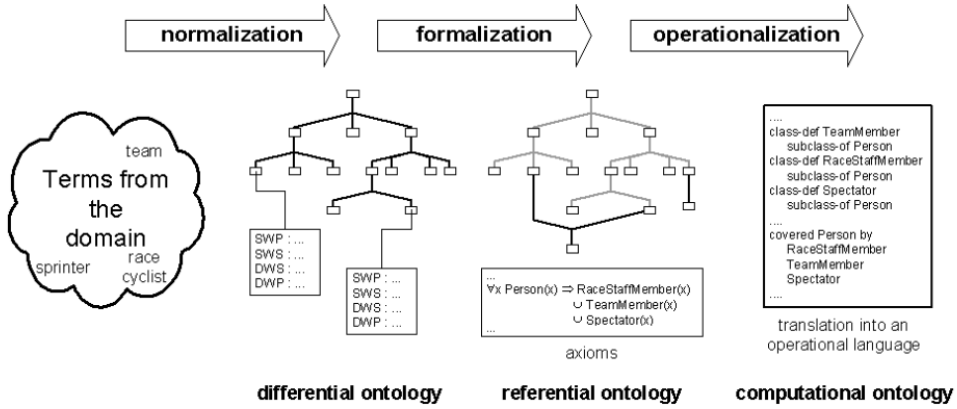


FIGURE 2.35: Bachimont's Ontology learning approach (Bachimont et al., 2002).

Sabou's Approach The ontology learning approach proposed by Sabou and her colleagues (Sabou et al., 2005) represent a natural language processing approach. The ontology extraction method uses a set of syntactic patterns to discover the dependency relations between words. Their extraction method exploits the syntactic regularities which are inherent from the sublanguage nature of web service documentations, which is a specialized form of natural language. The ontology extraction process consists of four main steps, as depicted in Figure 2.36.

- Dependency parsing: consists of annotating the corpus with linguistic information that helps in deciding the possible role of each word in the future ontology to be built.
- Syntactic patters: a set of syntactic patterns is used to identify and extract information from the annotated corpus.
- Ontology building: transforms the extracted relevant information into ontological constructs.
- Ontology pruning: excludes potentially uninteresting concepts from the ontology.

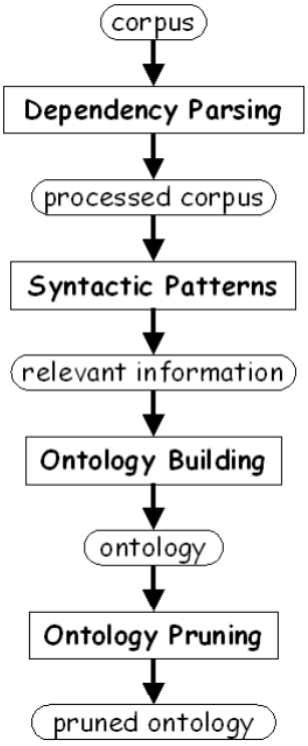


FIGURE 2.36: Ontology learning process steps (Sabou et al., 2005).

Madche and Staab's Approach Maedche and Staab (Maedche et al., 2001) have proposed a semi-automatic ontology learning approach with human intervention adopting the paradigm of balanced cooperative modeling (Morik, 1993) for the construction of ontologies for the Semantic Web. The ontology engineering cycle is composed of five main steps (see Figure 2.37):

1. Import/Reuse: existing ontologies are imported and reused by merging existing structures or defining mapping rules between existing structures and the ontology to be established.
2. Extract: in this phase, major parts of the target ontology are modeled with learning support feeding from web documents.
3. Prune: the rough outline of the target ontology needs to be pruned in order to better adjust the ontology to its prime purpose.
4. Refine: ontology refinement profits from the given domain ontology, but completes the ontology at a fine granularity.
5. Validate: application serves as a measure for validating the resulting ontology.

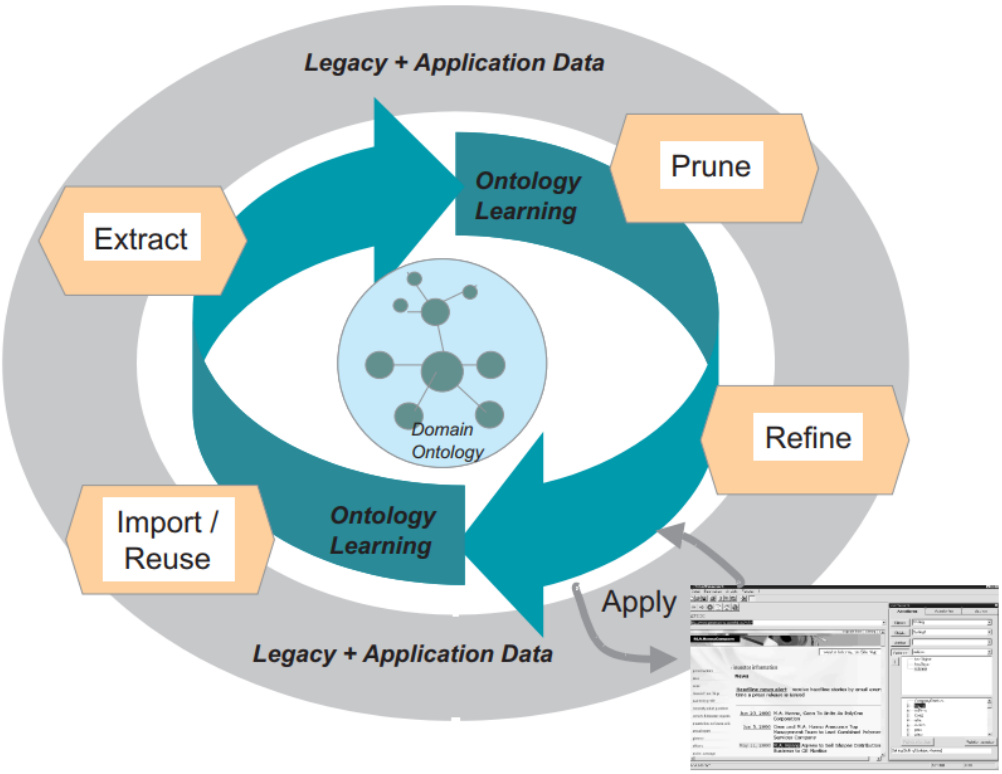


FIGURE 2.37: Ontology learning process steps (Maedche et al., 2001).

Ontology Learning Tools The main goal of using ontology learning tools is to reduce the time and cost of ontology development process. In the literature, a long list of ontology learning tools has been proposed. The existent tools differ according to input data types, output formats and mainly the methods and algorithms used in order to extract the ontological structures. In this section, the most relevant tools are outlined: Terminae, Text2Onto, OntoGen and T2K.

Terminae Terminae (Biebow et al., 1999) is a tool based on a methodology elaborated from practical experiments of ontology building in the domain of telecommunications. Linguistic and knowledge engineering tools are integrated in Terminae to guide the knowledge acquisition from texts and to build terminological and ontological models. The linguistic engineering part allows the definition of terminological forms from the study of term occurrences in a corpus using several NLP techniques (such as term extractor and relation extractor with lexico-semantic patterns). The knowledge engineering part involves knowledge-base management with an editor and browser for the ontology.

The Terminae methodology is composed of two main steps (see Figure 2.38):

- Terminological: established list of terms. This requires the constitution of a relevant corpus of texts on the domain. Then LEXTER (Bourigault, 1994), a

term extractor, proposes to the knowledge engineer a set of candidate terms from which the effective terms have to be selected with the help of an expert.

- Modeling: conceptualizes each term. The knowledge engineer analyzes the uses of the term in the corpus to define all the notions (meanings) of the term. He/she gives a definition in natural language for each notion and then translates the definition into a formalism.

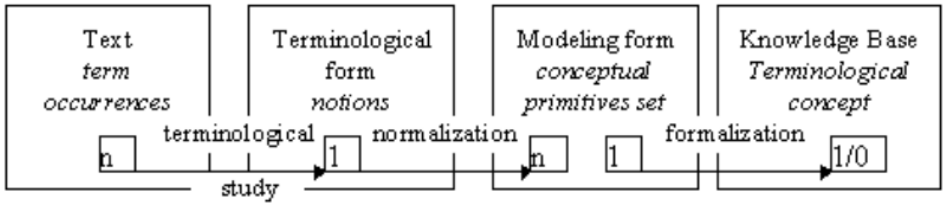


FIGURE 2.38: Architecture of Text2Onto (Biebow et al., 1999).

Text2Onto Text2Onto is a data-driven ontology learning tool that supports semi-automatic development of ontologies from textual documents (Cimiano et al., 2005b). Text2Onto is built upon the GATE[18] framework. Accordingly, Text2Onto implements linguistic processing and machine learning statistical techniques to extract domain concepts and relations. This tool features also algorithms for generating concepts, taxonomic and non-taxonomic relations.

In Text2Onto, the learned knowledge is represented into a meta level model called probabilistic ontology model (POM). POM is a collection of modeling primitives independent from any ontology representation language. Such primitives are defined in the Modeling Primitives Library (MPL) (see Figure 2.39).

In Text2Onto, seven main modeling primitives are distinguished:

- Concepts
- Concept inheritance (Taxonomic relationships)
- Concept instantiation (Instances)
- Properties/relations (Non taxonomic relationships)
- Domain and range restrictions (Axioms)
- Mereological relations (Part-of relations)
- Equivalence.

Text2Onto uses data driven change discovery for algorithms for supporting automatic and semi-automatic adaptation of a given ontology according to changes in a

[18]https://gate.ac.uk/

data set and provides several algorithms for instantiating each modeling primitive from POM.

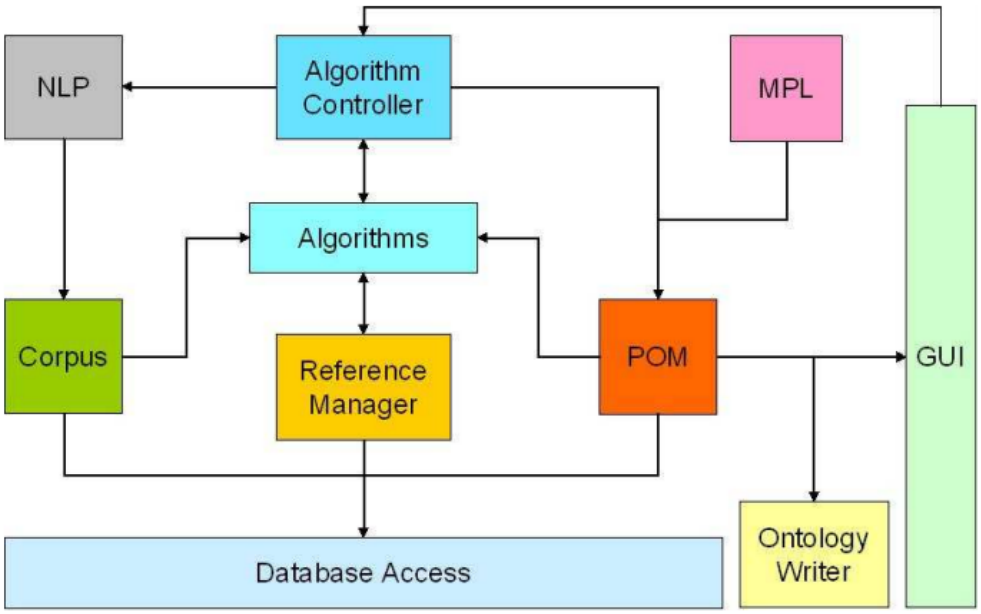


FIGURE 2.39: Architecture of Text2Onto (Cimiano et al., 2005b).

OntoGen OntoGen is a semi-automatic and data-driven ontology editor focusing on editing of topic ontologies (a set of topics connected with different types of relations (Fortuna et al., 2007). OntoGen tends to help the users to build ontologies by suggesting concepts and relations. This system integrates machine learning and text mining algorithms. OntoGen offers two main features: concept suggestion and naming and ontology and concept visualization.

T2K T2K, Text to Knowledge, extracts domain-specific information from texts using natural language processing techniques in three main phases (see Figure 2.40) (Dell'Orletta et al., 2014):

- Preprocess text and extract terms using NLP tools
- Form concepts using POS patterns
- Relations or knowledge organization.

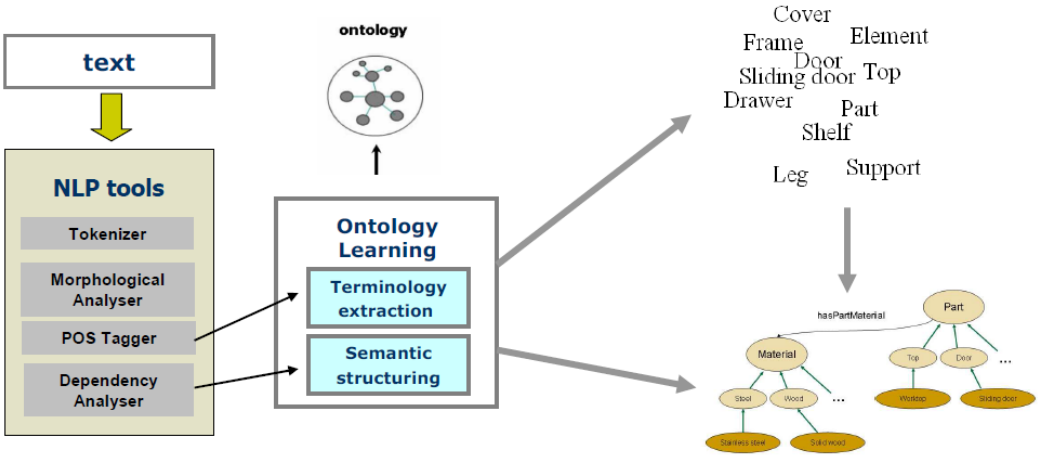


FIGURE 2.40: T2K.

Tool	Extracted Elements	Learning Techniques
Terminae (2005)	Terms, synonyms, concepts, taxonomies, non-taxonomic relations	linguistic and knowledge engineering (Conceptual Clustering)
Text2Onto (2005)	Terms, synonyms, concepts, taxonomies, non-taxonomic relations, instances	linguistic processing, statistical text analysis, machine learning and association rules
OntoGen (2006)	Terms, concepts, taxonomies	Machine learning and text mining
T2K (2008)	Terms, concepts, taxonomies	statistical text analysis and machine learning

TABLE 2.5: A summary of Ontology learning tools.

2.2.8.2 Ontology Reuse

Ontology reuse can be defined as the process in which existing ontological knowledge is used as input to generate new ontologies. Being reusable is an intrinsic property of ontologies, originally defined as means for "knowledge sharing and reuse" .

Ontology reuse is still rarely encountered today. This is partly due to the problem of finding suitable ontologies to reuse, and the way most ontologies are created, namely without reusability in mind. Also, most of the established ontologies containing domain knowledge are simply too big to be easily reused, and no quality information is available on web ontologies (Hartmann et al., 2009).

Ontology Reuse Case Studies In this section we give an overview of the most prominent case studies in ontology reuse, which have been published in the knowledge/ontology engineering literature.

Gómez-Pérez and Rojas-Amaya's Case Study Gomez-Perez and Rojas-Amaya describe a case study, in which an ontology for standard units and a chemical ontology are reused for the purpose of developing an ontology for environmental pollutants (Gomez-Perez et al., 1999). The reuse process clearly focuses on a method for ontology re-engineering, which attempts to capture the conceptual model of the implemented source ontologies in order to transform them into a new, more correct and more complete ontology. The re-engineering methodology proposed by the authors consists of three steps:

- Reverse engineering: on the basis of the code of the source ontology (i.e. its implementation in a particular representation language) one derives a possible conceptual model. This step is performed iteratively, by extracting models with an increasing complexity: the taxonomic structure, followed by relations between concepts and instances and finally more expressive constructs such as axioms or functions.
- Re-structuring: the objective of this step is to evaluate the correctness of the extracted model, correct the detected errors and refine it in conformity with the requirements of the new application setting.
- Forward engineering: the ontology is re-implemented on the basis of the revised conceptual model.

The reuse process was performed along the following stages:

1. Select reuse candidates: ontologies stored on the Ontolingua and the Cyc servers were manually selected and evaluated with respect to their relevance to the target domain and with respect to a series of general-purpose modeling guidelines.
2. Re-engineering: relevant ontologies were re-engineered as described above
3. Merging: the ontologies were merged to a final product.

The focus of the work is to demonstrate the applicability of the re-engineering approach with the help of a case study.

The authors admit the limitations of their approach with respect to the complexity of the ontological sources employed, and the need for automatic means. The experiment is restricted to taxonomic ontologies containing a manageable number of at most several hundreds of concepts.

Capellades' Case Study Capellades aimed at building an application ontology by reusing ontologies available at the Ontolingua Server (Capellades, 1999).

The reuse process covered two main stages:

- Select candidate ontologies: the selection step does not have to cope with the issue of discovering potential reuse candidates, as the set of reusable ontologies was limited to the Ontolingua repository.
- Customize and integrate relevant ontologies: due to the poor application relevance results obtained in the previous step, the integration was restricted to extracting particular fragments of the selected ontology, which were subsequently embedded to the application system.

In this case study, two main problems was associated with ontology reuse were: (1) the analysis and evaluation of existing ontologies is too costly in terms of development time, even when using current on-line browsing and editing tools; and (2) the lack of methods for quickly assessing the reusability and the consistency of existing ontologies make the decision process associated with reuse difficult.

The authors suggested that it is necessary to experiment more with ontology reuse and to objectively measure its cost-effectiveness, so that convergence to a conclusion regarding the feasibility of ontology sharing can be reached.

Arpirez et al. According to Arpirez and his colleague, knowledge reuse by means of ontologies faces three important problems at present (Arpirez et al., 2000): (1) there are no standardized identifying features that characterize ontologies from the user point of view; (2) there are no web sites using the same logical organization, presenting relevant information about ontologies; and (3) the search for appropriate ontologies is hard, time-consuming and usually fruitless.

In order to solve these problems, they give an account for a case study in which the $(ONTO)^2$ ontology was reused in order to build a living domain ontology about ontologies called *Reference ontology*, a meta-ontology intended to capture information about ontologies and ontology engineering projects.

The activities performed in the case study are not representative for a complete reuse life cycle, covering three phases:

- Choosing candidate ontologies: in this step the $(ONTO)^2$ ontology was evaluated with respect to its relevance and usability for the desired purpose. The reuse candidate fulfilled many of the evaluation criteria, ranging from domain to representation formalism.
- Analysis of the candidate ontologies: the ontology was analyzed as regards the quality of its modeling decisions and its validity.

- Integration: the $(ONTO)^2$ ontology was extended and revised in order to adapt it to the requirements of the new Reference ontology.

Reusing the $(ONTO)^2$ ontology was perceived as beneficial by the case study authors, who mention cost and interoperability as two of the major advantages of this engineering strategy. However, they also identify the circumstances which contribute to the efficient operation of the reuse process: the availability of the reused ontology in an appropriate representation form (including its conceptual structure) and the extensive knowledge of the ontology engineers with respect to the domain of the ontology.

2.2.8.3 Ontology Modularization

The main idea of modularization originates from the general notion of modular software in the area of software engineering. In software engineering domain, the modularity is a well established notion where it refers to a way of designing software in a clear, well structured way that supports maintenance and reusability (Grau et al., 2007b). However, in the ontology engineering domain, the notion of modularization and the problem of formally characterizing a modular representation for ontologies are not as well understood (Grau et al., 2007a), which causes suffer in the existing work and prevents further development (d'Aquin et al., 2007).

Despite this vagueness, ontology modularization is considered as a major topic in the field of formal ontology developments and a way to facilitate and simplify the ontology engineering process (Hois et al., 2009).

Generally, modularization denotes the possibility to perceive a large knowledge repository as a set of modules, i.e. smaller repositories that, in some way, are parts of and compose the whole knowledge (Stuckenschmidt et al., 2009). Therefore, an ontological modularization process is seen as a call for organizing ontologies into modules which could then be reused and combined in novel ways (Hois et al., 2009). In other words, ontology modularization is considered as a way to structure ontologies, meaning that the construction of a large ontology should be based on the combination of self-contained, independent and reusable knowledge components (d'Aquin et al., 2007).

In the literature, several different approaches for ontology modularization appeared. These approaches are classified into two main categories (Abbes et al., 2012) (d'Aquin et al., 2007): (1) ontology composition and (2) ontology partitioning and module extraction.

The first main category comprises approaches that focus on the composition of existing ontologies by means of integrating and mapping ontologies. Ontology composition aims to develop independently a set of ontology modules and assemble

them coherently and uniformly, by means of integrating and mapping, to form a wider ontology. Examples of ontology composition approaches are, among others (Steve et al., 1997) and (Bezerra et al., 2009).

The second main category comprises approaches for modularizing ontologies in terms of ontology partitioning and ontology module extraction.

- Ontology partitioning aims at splitting up an existing ontology into a set of ontology modules. Approaches for partitioning ontologies are proposed by (Schlicht et al., 2007), (MacCartney et al., 2003).
- Ontology module extraction, or segmentation (Doran, 2006), aims at reducing an ontology to its relevant sub-parts. Examples of approaches for ontology module extraction are (Grau et al., 2007c) (Sattler et al., 2009).

Ontology Modules Generally speaking, a module is a part of a complex system that functions independently from this system (Konev et al., 2009). In contrast to the software engineering domain, the notion of ontology module is not clear or understood in the domain of ontological engineering (Doran, 2006).

There is a need to formalize and define an ontology module, particularly in terms of its requirements (Bezerra et al., 2008). For (Grau et al., 2006), ontology module is considered as extractable part that can be reused outside the context of the general ontology.

More clearly, an ontology module is defined by (Doran, 2006) as "An ontology module is a reusable component of a larger or more complex ontology, which is self-contained but bears a definite relationship to other ontology modules including the original ontology". This definition implies that ontology modules can be reused either as they are, or by extending them with new concepts, and relationships. Each ontology module is considered as ontology itself since it can be extended with new concepts and relationships. Thereby, ontology modules are themselves ontologies (Abbes et al., 2012).

Based on their content, ontologies, or ontology modules, are classified into five main categories (Guarino et al., 1994), (Guarino, 1997) and (Heijst et al., 1997):

- Generic, or top-level, ontologies: describe generic concepts independently of a particular domain or problem.
- Core ontologies: in contrast to generic ontologies that span across many fields, core ontologies describe the basic categories within a domain such as law.
- Domain ontologies: specialize a subset of generic ontologies in a domain or sub-domain, e.g., criminal law.

- Application or domain-specific ontologies: developed for a specific application.

Ontology Modules Criteria The criteria of ontology modules generally aim at characterizing modular ontologies in order to evaluate the quality of modules (Gangemi et al., 2004). Generally, inspired by the soft-ware engineering domain, three main criteria a module should fulfill: self-contained, loose coupling and high cohesion (Stuckenschmidt et al., 2007) (Stuckenschmidt et al., 2003). Therefore, in the ontological engineering domain, some studies, such as (d'Aquin et al., 2009), believe that modularization criteria should be defined in terms of the applications for which the modules are created. They defined some ontology module criteria such as:

- Encapsulation: a module can be easily exchanged for another, or internally modified, without side-effects on the application.
- Independence: self-containment and reusability in order to improve the scalability of reasoning mechanisms.
- Domain coverage: generate significant module according to the different domains or topics covered by the original ontology.

2.2.8.4 Ontology Evaluation

Ontologies are a fundamental data structure for conceptualizing knowledge,but many different ontologies are built for conceptualizing the same body of knowledge and it should be possible to define them concerning some predefined criterion. Constructing an ontology need a way to evaluate the resulting ontology and possibly to guide the construction process and any refinement steps. Automated or semi-automated ontology learning techniques also require effective evaluation measures, which can be used to select the "best" ontology out of many candidates, to select values of tunable parameters of the learning algorithm, or to direct the learning process itself (Brank et al., 2005).

Ontology evaluation is an emerging field that has a number of frameworks and methodologies in existence (Vrandecic, 2009). Several studies in the literature, such as (Vrandecic, 2009), (Brank et al., 2005), (Gomez-Perez, 1995) and (Gomez-Perez et al., 2004), have defined ontology evaluation. Generally ontology evaluation is described as the process of deciding on the quality of an ontology in respect to a particular criteria (Brank et al., 2005).

An ontology can be evaluated against many criteria: its coverage of a particular domain and the richness, complexity and granularity of that coverage; the specific use cases, scenarios, requirements, applications, and data sources it was developed

to address; and formal properties such as the consistency and completeness of the ontology and the representation language in which it is modeled (Obrst et al., 2007).

In the literature, various approaches to the evaluation of ontologies have been considered depending on what kind of ontologies are being evaluated and for what purpose (Brank et al., 2005). These approaches are mainly classified into four main categories:

- based on comparing the ontology to a "golden standard" (Maedche et al., 2002).
- based on using the ontology in an application and evaluating the results (Porzel et al., 2004).
- based on involving comparisons with a source of data (e.g. a collection of documents) about the domain to be covered by the ontology (Brewster et al., 2004).
- based on human evaluation by assessing how well the ontology meets a set of predefined criteria, standards, requirements, etc. (Gomez-Perez, 2004).

Moreover, the ontology evaluation approaches can be grouped according to level of evaluation based on some quality metrics (Gangemi et al., 2006; Brank et al., 2005): lexical and concept/data (Maedche et al., 2002), taxonomic and semantic relations (Brewster et al., 2004), context-level (Ding et al., 2004), application-based (Porzel et al., 2004) and data-driven (Patel et al., 2003).

- Lexical, vocabulary, or data layer: Evaluation on this level tends to involve comparisons with various sources of data concerning the problem domain (e.g. domain-specific text corpora), as well as techniques such as string similarity measures (e.g. edit distance). An example of an approach that can be used for this evaluation level is the one proposed by Maedche and Staab (Maedche et al., 2002).
- Hierarchy or taxonomy and other semantic relations: An ontology typically includes a hierarchical is-a relation between concepts. Although various other relations between concepts may be also defined, the is-a relationship is often particularly important and may be the focus of specific evaluation efforts.

  Several approaches exist in the literature concerning this level such as (Brewster et al., 2004) that suggested a data-driven approach to evaluate the degree of structural fit between an ontology and a corpus of documents, (Guarino et al., 2002b) that presented a different aspect of ontology evaluation based on several philosophical notions that can be used to better understand the nature of various kinds of semantic relationships that commonly appear in ontologies, and to discover possible problematic decisions in the structure of

an ontology and (Maedche et al., 2002) that proposed several measures for comparing the relational aspects of two ontologies.

- Application level: Typically, the ontology will be used in some kind of application or task. The performance of the application depends partly on the ontology used in it. Thus, a good ontology will helps the application in question to produce good results on the given task. Therefore, ontologies may be evaluated by plugging them into an application and evaluating the results of the application.

  In this context, task-based evaluations offer a useful framework for measuring practical aspects of ontology deployment, such as the human ability to formulate queries using the query language provided by the ontology, the accuracy of responses provided by the system's inferential component, the degree of explanation capability offered by the system, the coverage of the ontology in terms of the degree of reuse across domains, the scalability of the knowledge base, and the ease of use of the query component (Obrst et al., 2007).

- Data-driven level: Evaluation of the ontology by comparing it to existing data (usually a collection of textual documents) about the problem domain to which the ontology refers. Examples of approaches in this level are (Patel et al., 2003) and (Brewster et al., 2004).

- Syntactic level: Evaluation on this level may be of particular interest for ontologies that have been mostly constructed manually. The ontology is usually described in a particular formal language and must match the syntactic requirements of that language.

- Structure, architecture, design: This is primarily of interest in manually constructed ontologies. The ontology should meet certain pre-defined design principles or criteria.

Additionally, other studies, such as (Gangemi et al., 2006), identify three main types of validation measures: (1)structural measures, that are typical of ontologies represented as graphs; (2)functional measures, that are related to the intended use of an ontology and of its components, i.e. their function; (3)usability-profiling measures, that depend on the level of annotation of the considered ontology.

Finally, we present the ontology validation concept in the work of Gomez-Perez (Gomez-Perez, 1995) who claimed that the development team must perform a global technical evaluation that ensures well-defined properties in two main levels: definitions of the ontology and documentation.

- Definitions of the ontology: technical evaluation that must be performed during the whole ontology life-cycle in order to detect the absence of some well-defined properties in the definitions. The evaluation steps include:
  - Check the structure or architecture of the ontology: to figure out if the definitions are built following the design criteria of the environment in which they are included.
  - Check the syntax of the definitions: to detect syntactically incorrect structure and/or wrong keywords in definitions without looking into their meaning.
  - Check the content in the definitions: to identify lack of knowledge and mistakes in the definitions. It deals with the problem of the three Cs: Consistency, Completeness and Conciseness.
    - Consistency: refers to the incapability of getting contradictory conclusions simultaneously from valid input data. An ontology is semantically consistent if and only if its definitions are semantically consistent.
    - Completeness: refers to the extension, degree, amount or coverage to which the information in a user-independent ontology covers the information of the real world.
    - Conciseness: refers to if all the information gathered in the ontology is useful and precise.
- Documentation: to guarantee that certain documents are developed and that they evolve in step with the definitions and software. Documentation includes: the natural language string in each definition, general information about the ontology, its basic ontological commitments, a summary of its definitions, studied cases in its evaluation and definitions taken from other ontologies.

2.3 Knowledge Engineering

After surveying the domain of ontology engineering, we will overview the knowledge engineering domain that consists of the process of building intelligent systems (Negnevitsky, 2005) which are considered as expert systems, or commonly known as knowledge-based systems (KBSs). Early, the Knowledge engineering (KE) is considered as a transfer process that turns the process of constructing KBSs from art into an engineering discipline (Studer et al., 1998). In other words, to transform the human knowledge into an implemented knowledge base based on knowledge acquisition. This requires the analysis of the building and maintenance process itself

and the development of appropriate methods, languages, and tools specialized for developing KBSs.

During the last decade, comprehensive knowledge-engineering methodologies have emerged which provide support for organizing the development process of knowledge-based systems (KBS) (Heijst et al., 1997). So far, the process of knowledge engineering or building a KBS may be seen as a *modeling activity* (Studer et al., 1998) (Valente et al., 1992). Building a KBS means building a computer model with the aim of realizing problem-solving capabilities comparable to a domain expert.

This knowledge is not directly accessible, but has to be built up and structured during the knowledge acquisition phase. Therefore this knowledge acquisition process is no longer seen as a transfer of knowledge into an appropriate computer representation, but as a model construction process (Clancey, 1989) (Fensel et al., 1996). This modeling view of the building process of a KBS has the following consequences (Studer et al., 1998):

- The modeling process is only an approximation of the reality.
- The modeling process is a cyclic process. New observations may lead to a refinement, modification, or completion of the already built-up model. On the other side, the model may guide the further acquisition of knowledge.
- The modeling process is dependent on the subjective interpretations of the knowledge engineer. Therefore this process is typically faulty and an evaluation of the model with respect to reality is indispensable for the creation of an adequate model.

2.3.1 Modeling Principles in Knowledge Engineering

In an overview of the field of knowledge engineering, three modeling principles are identified which lie at the heart of all recent knowledge engineering approaches (Musen et al., 1995): (1) role-limiting, (2) knowledge typing and (3) reusability. An additional fourth general knowledge engineering principle, which is skeletal models, is identified by (Heijst et al., 1997).

1. Role-limiting: is a mechanism for organizing knowledge by putting constraints on the ways knowledge elements of particular types can be used in reasoning (Heijst et al., 1997).
2. Knowledge typing: according to role-limiting, knowledge elements must be typed according to their role in problem solving. According to (Heijst et al., 1997), five different types of knowledge are distinguished in the literature (see Figure 2.41):

- Tasks: goals that must be achieved during problem solving.
- Problem-solving methods: ways to achieve the goals described in tasks.
- Inferences: reasoning steps in the problem-solving process, called also mechanisms. The inferences form a functional model which is sometimes called the inference model or inference structure.
- Ontologies: describe the structure and vocabulary of the static domain knowledge.
- Domain knowledge: refers to a collection of statements about the domain.

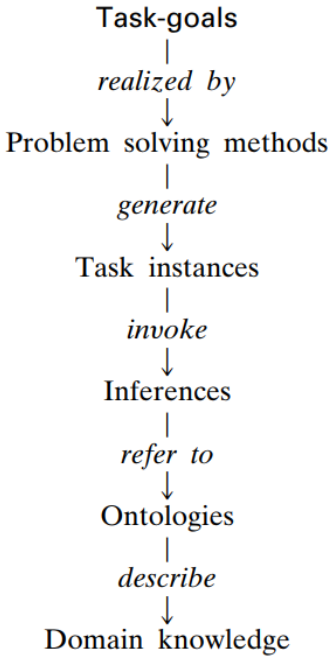


FIGURE 2.41: The different components of knowledge models (Heijst et al., 1997).

3. Reusability: reuse of knowledge components across domains and tasks is essential for knowledge engineering approaches. The availability of libraries of validated and well-documented knowledge components not only speeds up the KBS development process but it also facilitates maintenance and upgrading. However, there are differences between the approaches with respect to the nature and the grain size of the components that they consider potentially reusable.
4. Skeletal models: for this principle, the knowledge model components are often reused in the form of skeletal models (see Figure 2.42). Such models specify one part of a knowledge model (e.g. the problem solving method). The knowledge engineer then has to fill in the other parts to complete the

knowledge model. As a result of knowledge typing, the already specified parts in the skeletal model constrain how the other parts can be modeled. This way, skeletal models structure the knowledge modeling process.

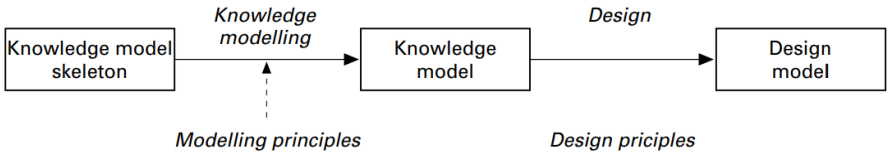


FIGURE 2.42: A schematic overview of how modern knowledge engineering approaches view the knowledge engineering process. (Heijst et al., 1997).

The knowledge model is an implementation-independent description of the knowledge and methods needed to perform a task. The design model describes how the knowledge model can be operationalized in a knowledge based system. The design model specifies both the general architecture of the KBS and the representations and algorithms that are used by the KBS to perform its task.

2.3.2 Knowledge Engineering Approaches

During the last decade, a number of approaches to knowledge engineering were proposed that are similar in spirit, although they differ in their details and terminology (Heijst et al., 1997) such as, among others, CommonKADS (Schreiber et al., 1993), MIKE (Angele et al., 1998), Generic Tasks (Chandrasekaran et al., 1992), Components of Expertise (Steels, 1990), Role-limiting Methods (McDermott, 1993) and Protégé (Puerta et al., 1992). We will review briefly the main features of the most known and used approaches.

2.3.2.1 CommonKADS

CommonKADS is the leading approach to support KBS engineering (Schreiber et al., 1993). It is considered as the best established and the most comprehensive knowledge engineering approach. CommonKADS provides the methods to perform a detailed analysis of knowledge tasks and processes (Schreiber et al., 2000). This approach originated from *KADS* (Schreiber et al., 1993). A basic characteristic of KADS is the construction of a collection of models, where each model captures specific aspects of the KBS to be developed as well as of its environment (Studer et al., 1998). An important aspect of CommonKADS is its reliance on multiple models to address the complexity of a knowledge management or knowledge engineering

project. In CommonKADS, six main models are distinguished (see Figure 2.43): *Organization*, *Task*, *Agent*, *Knowledge*,*Communication* and *Design*.

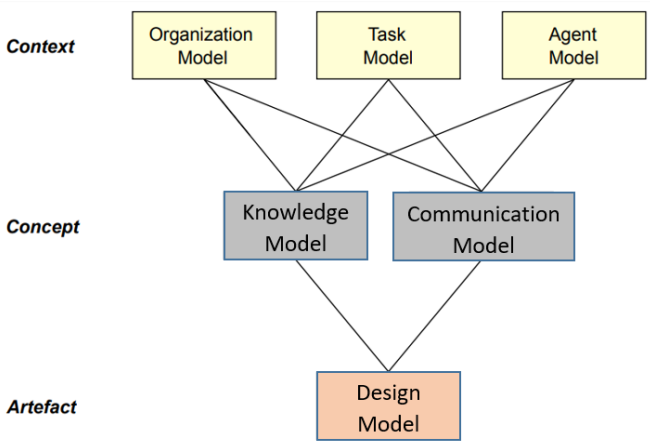


FIGURE 2.43: CommonKADS models (Schreiber et al., 2000).

We will discuss the *knowledge model* which distinguishes three main types of knowledge required to solve a particular task (Studer et al., 1998): domain layer, task layer and inference layer.

- Domain layer: in order to solve a given task, there is a need to model all the domain specific knowledge. This includes a conceptualization of the domain in a domain ontology, and a declarative theory of the required domain knowledge. One objective for structuring the domain layer is to model it as reusable as possible for solving different tasks.
- Inference layer: at this layer the reasoning process of the KBS is specified by exploiting the notion of a PSM (Problem-Solving Method). The inference layer describes the inference actions the generic PSM is composed of as well as the roles, which are played by the domain knowledge within the PSM. The dependencies between inference actions and roles are specified in what is called an inference structure.
- Task layer: the task layer provides a decomposition of tasks into subtasks and inference actions including a goal specification for each task, and a specification of how these goals are achieved. The task layer also provides means for specifying the control over the subtasks and inference actions, which are defined at the inference layer.

The clear separation of the domain specific knowledge from the generic description of the PSM at the inference and task layer enables in principle two kinds of reuse: on the one hand, a domain layer description may be reused for solving different tasks by different PSMs, on the other hand, a given PSM may be reused in a different domain by defining a new view to another domain layer (Studer et al., 1998).

Furthermore, CommonKADS methodology is tailored to the legal domain by adding domain-specific elements (Van Kralingen et al., 1999).

2.3.2.2 MIKE

The MIKE approach (Angele et al., 1998) (Model-based and Incremental Knowledge Engineering) provides a development method for KBSs covering all steps from the initial elicitation through specification to design and implementation. In MIKE, the entire development process is divided into a number of sub-activities (see Figure 2.44): Elicitation, Interpretation, Formalization/Operationalization, Design, and Implementation. Each of these activities deals with different aspects of the system development.

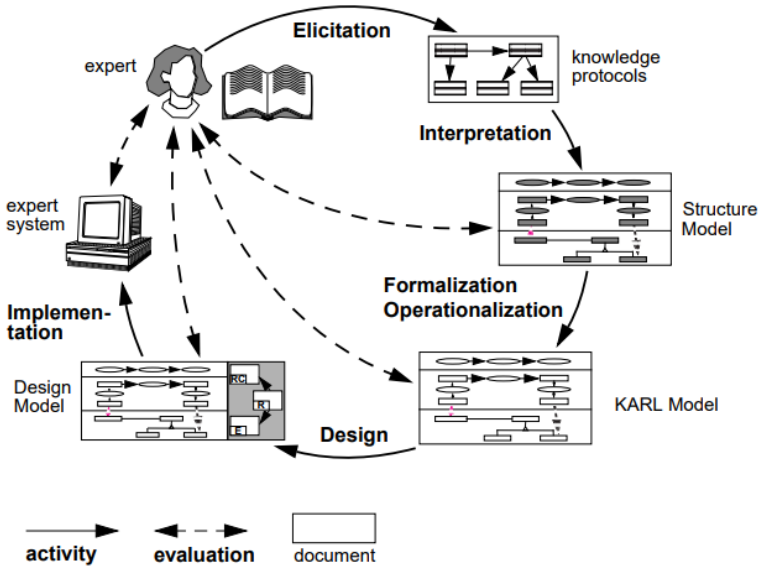


FIGURE 2.44: MIKE development process (Angele et al., 1998)

.

2.3.2.3 PROTÉGÉ-II

The PROTÉGÉ-II approach (Eriksson et al., 1995), (Puerta et al., 1992) aims at supporting the development of KBSs by the reuse of PSMs and ontologies. In addition, PROTÉGÉ-II puts emphasis on the generation of custom-tailored knowledge-acquisition tools from ontologies (Eriksson et al., 1994). PROTÉGÉ-II relies on the task-method-decomposition structure. By applying a PSM, a task is decomposed into corresponding subtasks. This decomposition structure is refined down to a level at which primitive methods, so-called mechanisms, are available for solving the subtasks (Studer et al., 1998). Three main types of ontologies are used in PROTÉGÉ-II: method, domain and application. Method ontologies define the

concepts and relationships that are used by the PSM for providing its functionality. Domain ontologies define a shared conceptualization of a domain. Both PSMs and domain ontologies are reusable components for building up a KBS. Application ontologies extend domain ontologies with PSM specific concepts and relationships.

Approach	Core Concepts	Concepts Instances
CommonKADS	Models	Organization, Task, Agent, Communication, Knowledge, Design
MIKE	Ativities	Elicitation, Interpretation, Formalization/Operationalization, Design, and Implementation
PROTÉGÉ-II	Task-method decomposition structure	Subtasks

TABLE 2.6: A summary of Knowledge engineering approaches

2.3.3 Legal Knowledge Engineering Approaches

The development of a knowledge-based system is seen as the construction of a set of models of problem solving behavior, such that the system is a computational realization of these models. All approaches to legal knowledge engineering have an implicit and general goal which is the development of (better) legal knowledge-based systems (LKBS) (Valente et al., 1992).

In the legal engineering domain, three main approaches are distinguished for building legal knowledge-based systems: case-based, rule-based and model-based. In this section, the rule-based and model-based approaches will be discussed.

2.3.3.1 Rule-Based Approach

The aim of the rule-based approach is to represent the knowledge of the domain in the form of productive rules in order to solve given problems by applying rule-based reasoning techniques. Rules represent general knowledge of the domain and exhibit a number of attractive features such as, naturalness, modularity and ease of explanation. They constitute a popular knowledge representation scheme used in the development of KBSs. However, one of their major drawbacks is the difficulty to acquire them (Gonzalez et al., 1993).

Rule based reasoning (RBR) is one of the most popular reasoning paradigms used in artificial intelligence (AI) (Buchanan et al., 1984). The reasoning architecture of rule-based systems has two major components: the knowledge base (usually consisting of a set of "IF ... THEN ..." rules representing domain knowledge) and the inference engine (usually containing some domain independent inference

mechanisms, such as forward/backward chaining). (Buchanan et al., 1984). In the legal domain, a legal norm is represented by an obligation rule that denotes that the conclusion of the rule will be treated as an obligation (Governatori et al., 2006). The representation of legal norms is obviously crucial for representing legal documents, regulations and other sources of law (Gordon et al., 2009). Representing legal contents though obligation rules comports with the widespread idea that legal norms typically have the conditional form:

IF *condition* (operative facts) THEN *conclusion* (legal effect).

`If ... Then` is a normative conditional. This view highlights an immediate link between the concepts of the norm and the rules (Gordon et al., 2009). This link relies on ontologies since they are used for filling the gap between document representation, expressed in natural language, and rules modelling using logical formalisms (Palmirani et al., 2009). Thus, the legal rules are considered as legal interpretation and modelling of the meaning of texts by transforming the legal norms to logical rules for permitting rule-based reasoning (Palmirani et al., 2012).

The most popular expert systems are rule-based systems. A great number have been built and successfully applied in such areas as business and engineering, medicine and geology, power systems and mining (Negnevitsky, 2005). Examples of legal expert systems that applied rule-based approach:

1. LDS: a rule-based legal decision-making system for the field of product liability law (Waterman et al., 1980). The system is being used to study the effect of changes in legal doctrine on settlement strategies and practices.
2. Gardner's System: a rule-based system to identify legal issues in the analysis of law school examination fact patterns involving the contracts law of offer and acceptance. The program primarily used if-then rules to represent its legal knowledge of contract law(Gardner, 1987).
3. TAXMAN: McCarty's rule-based system concerned with the development of a computational theory of legal reasoning, using corporate tax law as an experimental problem domain.
4. JUDITH: a rule-based system for the German Civil Code (Popp et al., 1975)
5. LEGOL: a rule-based system for the analysis of the administrative systems in terms of rules and regulations.

2.3.3.2 Model-Based Approach

The case-based and rule-based approaches are interested mainly in capturing the inferential aspects of legal knowledge more than in expressing the conceptual components and dependencies among kinds of knowledge (Biasiotti et al., 2011).

The model-based approach, proposed by (Valente et al., 1992), implied two major changes to the domain of knowledge engineering focusing on the modeling perspectives (see Figure 2.45):

- The process of developing knowledge-based systems is divided into two phases: modeling and realization.
  - The focus of the modeling phase is on the conceptual level where there is much more emphasis on the analysis of the problem and the domain.
  - Meanwhile, the focus of the realization phase is on the design and the implementation of the computational system.
- The modeling paradigm proposes the separation of the domain knowledge and problem-solving knowledge. This separation implies two main advantages. The first advantage is that domain models can be developed independently of specific tasks and problems, and can therefore have a high degree of reusability. The second advantage is that general problem solving models can also be used and adapted to domain-specific tasks, generating relatively domain- independent tasks.

The development of domain-specific models should, whenever possible, be supported by theories. Following the separation of domain and problem-solving knowledge, there can be two theories: a theory of domain reasoning and a theory of domain knowledge. The first specifies the specific characteristics of reasoning in the domain. This is made by specifying common reasoning modes and structures for typical tasks in the domain. The second explains how the domain is organized by containing an ontology of the domain (Valente et al., 1992).

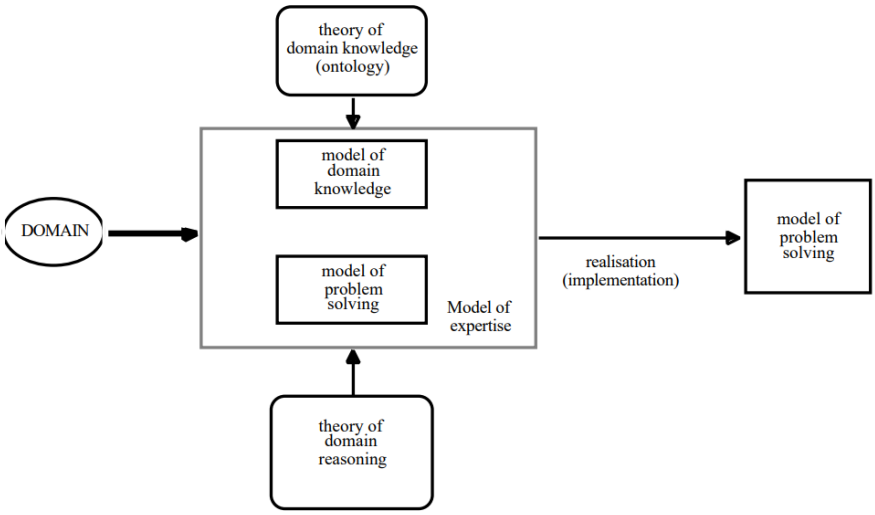


FIGURE 2.45: Model-based approach (Valente et al., 1992).

The model-based approach established some principles and guidelines summarized in follows (Valente et al., 1992):

- The use of models of legal argument and argumentation: Legal arguments and argumentation play a significant role in legal reasoning (Gardner, 1987) , (McCarty, 1980).The representation of legal arguments(arguments in conceptual, structured form, not only in natural language) can be used as a basis for representing legal reasoning.
- Generality in legal systems and disciplines: Within some legal systems there are sub-systems that are referred to as different disciplines (e.g. Civil and Penal Law). However, legal systems and legal disciplines could have more in common than what is usually assumed. Thus, a principle is taken that, whenever possible, the theories and models developed within the model-based approach should be general - i.e. valid for all major legal systems. As a result, there should be theories that explain these differences in terms of the knowledge categories each system or discipline contains and/or how they are used in specific legal problem solving.
- Broad ontologies of law: Most theoretical efforts in legal knowledge engineering have concentrated on the characteristics and roles of open-textured concepts. Although this is indeed a very important aspect, few works have been dedicated to represent the differences between normal rules of law (norms) and rights, legal positions, etc.
- The use of theories from legal philosophy: computational theories of law do not need to be divorced from philosophical ones, but rather borrow from their investigations and avoid reinventing the wheel.
- Legal knowledge and commonsense knowledge: to adopt the principle of separating legal knowledge and commonsense knowledge. Some claim that this is impossible, because law is incurably permeated with commonsense knowledge. The key for such a separation rely on a broader discussion about ontologies of law.

Finally, according to (Valente et al., 1992), the model-based approach to legal knowledge engineering can provide an alternative to traditional approaches to AI & Law by: (1) requiring a coherent domain theory based on explicit ontologies; (2) focusing on the use of knowledge-level models and (3) viewing knowledge engineering as a modeling task.

2.4 Legal Rule-Based Systems

A rule-based system consists of set of IF-THEN rules, set of facts, and some interpreter controlling the application of the rules on the given facts. Therefore, in a rule-based system, much of the knowledge is represented as rules, that is, as conditional sentences relating statements of facts with one another (Buchanan et al., 1983).

In the legal domain, any legal expert system, known as legal knowledge based systems (LKBS), must be capable of legal reasoning (Popple, 1990). Thus, the system must be based upon a model of legal reasoning by describing the norms that operate within the legal system (Popple, 1996). Legal reasoning, applied earlier in various approaches for decision making purposes, describes how legal expert system takes legal decisions with the help of rules (Valente et al., 1991). Accordingly, legal reasoning is considered as a rule-guided activity, where most part of it consists of applying legal rules to interpretations of cases (Gardner, 1987). This kind of reasoning is called rule-based reasoning performed by rule-based expert systems where the reasoning process is based on a set of if-then rule statements used to describe certain patterns of the giving domain such as legal norms.

Mainly, legal norms are expressed in textual sources such as legislations and codes and have basically the following structure (Davis et al., 1984):

```
If A1,...,A2 then B;
```

where "`A1,...,A2`" are the conditions of the norm, "B" is the legal effect and "`if...then`" is a normative conditional. This view highlights an immediate link between the concepts of the norm and the rules (Gordon et al., 2009).

This link relies on ontologies since they are used for filling the gap between document representation, expressed in natural language, and rules modeling using logical formalisms (Palmirani et al., 2009). Thus, the legal rules are considered as legal interpretation and modeling of the meaning of texts by transforming the legal norms to logical rules for permitting reasoning (Palmirani et al., 2012).

2.4.1 Evaluation of Rule-Based Systems

Basically, rule-based systems have several advantages and limitations. Concerning the advantages, the most known are presented in the following (Negnevitsky, 2002):

- Natural knowledge representation in the form of `if-then` rules that reflect the problem-solving procedure explained by the domain experts.
- Uniformity of structure where all the production rules are expressed in the same `IF...THEN` format.

- Modularity of structure where each rule is an independent piece of knowledge. The very syntax of production rules enables them to be self-documented.
- Separation of knowledge from its process. The structure of a rule-based expert system provides an effective separation of the knowledge base from the inference engine. This makes it possible to develop different applications using the same expert system shell. It also allows a graceful and easy expansion of the expert system. To make the system smarter, a knowledge engineer simply adds some rules to the knowledge base without intervening in the control structure.
- Justification of the determinations by explaining how the system arrived at a particular conclusion and by providing audit trails.

Meanwhile, rule-based systems suffer from some limitations, such as:

- Opaque relations between rules because of the uniformity and modularity of their structure.
- Inability to learn from experience.
- In this context, a limitation to cite, that concerns not only rule-based systems but all legal expert systems, which is the problem of how to model vague or "open-textured" concepts. For instance, uncertain and fuzzy legal concepts such as "reasonable" and "intentional" cannot be modeled in a way analogous to human thinking. What constitutes reasonable behavior will vary from time to time, place to place and person to person (Dove, 1996b). In this context, some authors in the field (Breuker, 1990) considered that "open texture problems are functional to adequate regulations and should not be resolved by automatic legal reasoning systems. They form the human interface between case and regulation and should be implemented as such in legal reasoning systems, i.e. as a task for the user".

Despite these limitations, there are still many problems that can be solved by rule-based systems. The goal of these systems is not to solve all legal automation problems, but there are ideal for encoding legal principles found in statutes and regulations where the law is explicit and knowable, but logically complicated.

2.4.2 Methods of Reasoning in Rule-Based Systems

It is commonly known that rule-based systems are categorized in forward chaining and backward chaining systems. Forward chaining systems are primarily data-driven, while backward chaining systems are goal-driven. In this section, we study the reasoning process of these systems.

2.4.2.1 Forward Chaining

In forward chaining systems, the purpose is to determine results from premises. The start is defined from the available information and then draw conclusions. Thus, the system analyzes the problem by looking for the facts that match the IF part of its IF-THEN rules. If there is a matching, then the rule is executed. This process continues until the goal is found. Therefore, forward chaining is considered as a data-driven approach. The forward chaining algorithm is represented as the sequence of the following steps:

1. Initial facts are inputs from the user to be set into the database (working memory);
2. Check left side of the production rules;
3. If the logical condition part of a rule (IF part) matches, then the rule fires;
4. Execute right side actions;
5. Retract old conditions/facts;
6. Input new conditions/facts;
7. Do other input-output actions, unifications etc.
8. Repeat until no other rules fire.

Matching of patterns in chaining process is also known as unification. So the application of the chaining procedure depends on unification. In the unification a set of binding of variables is considered. In a unification process two atomic sentences are compared and a unifier is returned if one of the given sentences exists.

2.4.2.2 Backward Chaining

In backward chaining systems, the start is defined from an hypothesis (expectation) and then seek evidence that supports (or contradicts) the expectation. Backward chaining works in reverse to forward chaining, and starts from the goal and tries to find data to prove its goal. Therefore, it is also called a goal-driven reasoning. After starting from the given goal, the search of THEN parts of the given rules (action part) (RHS) is conducted, and if the rule is found and its IF part (condition) matches the data in the database, then the rule is executed (fired). The backward chaining algorithm can be represented in the following form:

1. The rule that matches the goal is selected;
2. IF the condition (IF part) is empty, ask the user for information; ELSE WHILE not end, AND we have the selected rules DO,
3. Add the conditions of the rules,

4. IF the condition is not met, THEN put the condition as a goal to solve END WHILE.

2.4.3 Existent Rules Interchange Languages

In this paragraph, the most known rule interchange languages are presented for the purpose of modeling and formalizing rules in rule-based systems.

2.4.3.1 RuleML

RuleML(Rule Markup Language)[19] is an XML based language for the representation of rules. It encompasses a hierarchy of rules from reaction rules (event-condition-action rules), via integrity-constraint rules (consistency-maintenance rules) and derivation rules (implicational-inference rules), to facts (premiseless derivation rules).

RuleML is capable of specifying queries and inferences in Web ontologies, mappings between Web ontologies, and dynamic Web behaviors of workflows, services, and agents (Boley et al., 2001).

RuleML provides a way of expressing business rules in modular stand-alone units which maintains the flexibility and the extensibility of the language. Each module is meant to implement a particular feature relevant for a specific language or application (e.g., modules for various types of negation, for example, classical negation, and negation as failures). Each module is intended to refer to a semantic interpretation of the feature implemented in the module. However, RuleML does not have a mechanism to specify semantic structures on which to evaluate elements of the language (Wagner et al., 2004). Moreover, it lacks support for the use of deontic concepts, such as obligations, permissions and prohibitions (Lam et al., 2016).

In the literature, some works proposed extension and interpretation for this area, in particular for the representation of (business) contracts (Grosof, 2004; Governatori, 2005; Governatori et al., 2009).

2.4.3.2 SBVR

SBVR(Semantics of Business Vocabulary and Business Rules)[20] is a standard proposed by the Object Management Group (OMG) for the representation and formalization of business ontologies, including business vocabularies, business facts and business rules.

[19] http://www.ruleml.org, retrieved 22-12-2017
[20] http://www.omg.org/spec/SBVR/1.0/, retrieved 22-12-2017

Thus, SBVR bears on business rules, which may or may not have legal standing (Athan et al., 2014). Business rules are generally expressed in natural language. SBVR provides a logic means, called semantic formulations, for describing the structure of the meaning of rules. The formal representation is based on several logics including first order logic and deontic logic. Furthermore, SBVR adopts model theoretic interpretations for semantic formulations.

The focus of SBVR is on modeling not providing a framework for executing the rules (Gordon et al., 2009). Concerning the modeling of norms, SBVR has two main features for the modeling of norms: (1) the introduction of deontic operators to represent obligations and permissions and (2) the use of controlled natural languages for modeling norms. Unfortunately, SBVR is not suitable for representing deontic notions and conflicts since it is based on first-order-logic.

2.4.3.3 SWRL

SWRL(Semantic Web Rule Language)[21] is a W3C proposal for a rule interchange format which combines ontologies represented in OWL with RuleML. SWRL is originally proposed by (Horrocks et al., 2004) as *"a combination of the OWL DL and OWL Lite sublanguages of the OWL Web Ontology Language with the Unary/Binary Datalog RuleML sublanguages of the Rule Markup Language [RuleML]*". The SWRL language has an RDF syntax and an XML syntax based on RuleML.

SWRL extends OWL axioms to include (monotonic) Horn-like rules. It is the only approach that gathers ontology and rules in product development (Fiorentini et al., 2010) where users are permitted to write rules that can be expressed in terms of OWL concepts and that can reason about OWL individuals (O'Connor et al., 2008).

In SWRL, rules are of the form of an implication between an antecedent (body) conjunction and a consequent (head) conjunction in the following form (Antoniou et al., 2005a):

$$a_1 \wedge a_2 \wedge a_3 \ldots a_n \implies b_1 \wedge b_2 \wedge b_3 \ldots b_n;$$

where description logic expressions can occur on both sides, "$\wedge$" is an operator for the logical AND, "$\implies$" is an operator for drawing the conclusion and a_i and b_i are OWL atoms such as, among others, concepts, object properties, data properties, sameAs, and differentFrom. The intended interpretation of SWRL rules is in classical first-order logic: whenever the conditions specified in the antecedent hold, then the conditions specified in the consequent must also hold. SWRL supports a rich set of built-ins inspired by XQuery and XPath2 functions[22].

[21] https://www.w3.org/Submission/SWRL/, retrieved 22-12-2017
[22] http://www.w3.org/TR/xpath-functions/, retrieved January 3 2018

SWRL is supported by several DL reasoners such as KAON2[23] and Pellet (Sirin et al., 2005) that facilitate the DL-safe fragment of SWRL, while RacerPro (Haarslev et al., 2001) supports a SWRL-like syntax with a slightly different semantics (for instance, closed world reasoning is supported in RacerPro's variant of SWRL) (Eiter et al., 2008).

2.4.3.4 RIF

RIF(Rule Interchange Format)[24] Working Group was chartered by the World Wide Web Consortium in 2005 to create a standard for exchanging rules among rule systems, in particular among Web rule engines. Like RuleML, RIF is intended to be an extensible framework for a whole family of rule languages, possibly with different semantics (Gordon et al., 2009).

Concerning the RIF dialects, the RIF Working Group has focused on two kinds of dialects: logic-based dialects and dialects for rules with actions.

- Logic-based dialects: include languages that employ some kind of logic, such as first-order logic or non-first-order logics.
- The rules-with-actions dialects: include production rule systems, such as Jess[25], Drools[26] and JRules[27], as well as reactive (or event-condition-action) rules, such as Reaction RuleML[28] (Paschke, 2014).

RIF does not provide direct support for adequate representation of legal rules and legal reasoning since they do not support e.g. logic-based negation, non-monotonic reasoning, events and temporal metadata, among other relevant features (Athan et al., 2014).

2.4.3.5 LKIF

LKIF(The Legal Knowledge Interchange Format)[29] is a Semantic Web based language, developed in the European project ESTRELLA[30], for representing legal knowledge in order to support modeling of legal domains and to facilitate interchange between legal knowledge-based systems.

[23] http://kaon2.semanticweb.org/
[24] https://www.w3.org/TR/rif-overview/, retrieved 23-12-2017
[25] http://www.jessrules.com/
[26] http://www.drools.org/
[27] http://www.ilog.com/products/jrules
[28] http://wiki.ruleml.org/index.php/Specification_of_Reaction_RuleML_1.02
[29] http://www.estrellaproject.org/doc/D1.1-LKIF-Specification.pdf
[30] http://www.estrellaproject.org

In general, LKIF provides a direct support for representing three types of knowledge, which have been identified as most indispensable to law and legal reasoning: terminological knowledge, legal rules and normative statements.

- Terminological Knowledge: this layer is supported in LKIF through the Web Ontology Language (OWL).
- Legal Rules: LKIF does not use XML schema for modeling legal rules, such as Common Logic, RuleML, SWRL, or RIF. This layer has required more sophisticated formalism. For that reason, a partly novel rule formalism, called LKIF rules, has been developed and incorporated in LKIF.
- Normative Statements: this layer is given a direct support via the Norm module included in LKIF-Core ontology. No particular deontic logic is imposed on LKIF representation.

Thus, LKIF is a specification that includes a legal core ontology (LKIF-Core) and a legal rule language that closely represents legal knowledge and reasoning. According to (Athan et al., 2014), LKIF does not provide mechanisms to handle concurrent interpretations of a legal source; more specifically, while it might be possible to represent the content of the individual (alternative) interpretations, it is not possible to specify that these representations are mutually exclusive.

A reference inference engine for LKIF, called Carneades[31] (Gordon et al., 2007), was developed in ESTRELLA. Carneades is written in a functional style, using the Scheme programming language, and is available as Open Source software. Carneades places some restrictions on LKIF rules: The heads of rules are limited to literals (positive or negated atomic formulas) and the bi-conditional ($\Leftrightarrow$) operator and first-order quantifiers are not supported (Gordon et al., 2009).

2.4.3.6 LegalRuleML

LegalRuleML is an extension of RuleML, an XML based language for the representation of legal rules using formal semantics (Palmirani et al., 2011; Athan et al., 2013a). The goal of the LegalRuleML is to extend RuleML with features specific to the formalization of norms, guidelines, and legal reasoning. It aims to bridge the gap between natural language descriptions and semantic norms.

It reuses and extends concepts and syntax of RuleML wherever possible, and also adds novel annotations (Athan et al., 2015). LegalRuleML introduces features which are fundamental for modeling legal rules (Athan et al., 2013b; Athan et al., 2015):

[31] https://carneades.github.io/

- Isomorphism between rules and natural language normative provisions: to maintain a link between the units of natural language textual provisions and the sets of rules.
- Defeasible logics: norms are often written in a way that they admit exceptions. Defeasiblity allows for a natural representation of exceptions and permits terms to be defined in an open textured fashion.
- Jurisdiction of norms: norms emanated from different authorities, different locations, and different times. Relative to such differences, norms can produce different effects. To properly model such contextual dependence, LegalRuleML associates rules with the jurisdictions where the rules hold.
- Legal temporal parameters: LegalRuleML is able to define temporal instants and intervals that can be used to build complex legal events and situations (e.g. date of publication, interval of suspension, interval of efficacy but not applicability).
- Legal deontic operators: the function of prescriptive rules is to describe the normative effects that they produce (e.g., obligations, permissions, prohibitions, ...), the parties related to them, and the conditions under which such effects are produced.
- Qualifications of norms: legal documents can contains different types of norms (constitutive, technical, prescriptive, etc.). Some norms are intended to define the terms used in the document, others to produce normative effects, and others to describe legal procedures.
- Semantic management of negation;

The specifications of core or domain legal ontologies are out of the scope of LegalRuleML. This interchange language is independent from any legal ontology and logic framework. However it includes a mechanism, based on IRIs, for pointing to reusable classes of a specified external ontology.

2.5 Conclusion

We presented in this chapter the main research lines covering the background of the thesis which are the domains of ontology engineering and knowledge engineering and the rule-based systems.

Firstly, we analyzed the ontologies in general, their definitions, classifications, criteria and components. Furthermore, the legal ontologies are discussed as well as their roles and uses. Secondly, the most relevant ontology engineering methodologies,

tools and formalisms are surveyed. Additionally, we presented some ontology engineering support processes that can be useful during the development of ontologies. Finally, the domain of knowledge engineering is discussed. The most commonly known knowledge engineering approaches are outlined. Then, the approaches for building legal knowledge based systems are explored.

The literature review conducted in our work shows the extensive work done in the field of building ontologies. However, this domain suffers from some limitations that could make the ontology building process difficult and complicated. Actually, the outlined methodologies address the issue of development of ontologies specifically either from scratch or by reusing existent ones. Most of them have focused on core ontology development activities such as: requirements analysis, conceptualization, implementation, evaluation and maintenance. Moreover, the application-specific requirements are seen as an integral part of the requirements analysis activity of various methodologies. This will restrict the potential reusability of the developed ontologies as well as making them not well-founded.

Other limitations are noticed such as the lack of the focus on the participatory approach in the ontology engineering process. Meanwhile, such approach can effectively simplify and enrich the ontology building process specifically for complex domains such as the legal domain. Additionally, concerning the over-viewed legal ontologies, we found that most of them are not built fo reasoning purposes. Though, an essential role of legal ontologies is reasoning and problem solving capabilities, where an ontology can be used as a knowledge base for building knowledge-based systems.

We propose in our research some key challenges that could enrich the ontology building process, make it more cooperative and simple and leading to well-founded ontologies at the end. Firstly, we suggest to diversify the data sources such as existent validated ontologies to be reused as well as textual resources related to the domain of discourse, to enforce the cooperative work and to apply some support processes (discussed in chapter 3). Secondly, the use of the developed ontology in reasoning purposes such as rule-based reasoning model (discussed in chapter 5).

Chapter 3

MIROCL: A Modular Middle-Out Collaborative Approach for Building Well-Founded Domain Ontologies

Contents

3.1 Overview

The overall objective of this thesis is to build a well-founded legal domain ontology for rule-based reasoning purposes. In this chapter, a novel modular middle-out collaborative approach, named MIROCL, is presented. The aim of MIROCL is to build well-founded domain ontologies from heterogeneous data sources by applying different ontology engineering support processes such as: ontology modularization, integration, reuse and learning from texts. The proposed approach is considered as middle-out composed mainly of two complementary strategies, top-down and bottom-up. MIROCL tends to simplify the ontology building process, make it more cooperative by incorporating different contributors from various backgrounds such as domain experts, knowledge engineers and ontology engineers.

The remainder of this chapter is organized as follows: in section 3.2, we present the problems facing ontology building processes. In order to solve these problems, several issues are taken into consideration such as the concept of building well-founded ontologies which is discussed in section 3.3, the middle-out aspect of building ontologies presented in 3.4, the collaborative aspect of ontology development analyzed in section 3.5 and the data heterogeneity concept is introduced in section 3.6. Furthermore, the motivations of the proposed approach are introduced in section 3.7. The ontology building support processes that compose MIROCL, ontology modularization, ontology reuse, ontology learning from texts and ontology integration, are then discussed in sections 3.7.2, 3.7.3, 3.7.4 and 3.7.5 respectively. Finally, section 3.9 introduced the life-cycle of MIROCL and section 3.10 concludes the chapter.

3.2 Problems Facing Ontology Building Process

Actually, the ontology engineering community is facing several key problems concerning the ontology building processes. In the following, we discuss the main problems that we have noticed in this domain:

- *Build ontologies from scratch*: Most ontologies are built from scratch, rather than reusing existing ones, leading to high engineering efforts and costs (Hartmann et al., 2009). In fact, building ontologies from scratch is not easy. It is considered as a resource-intensive, time consuming and costly task. This is due to the difficulty and complexity of capturing knowledge from textual sources which are mainly unstructured documents such as legislations and codes in the legal domain.
- *Build ontologies that are not well-founded*: This problem is considered as a consequence of the aforementioned one. Actually, building ontologies from

scratch will not lead to well-founded domain ontologies. These ontologies must be represented with the support of a well-known foundational theory.

- *Build ontologies for specific purpose*: Most existing ontologies are built having a specific application scenario in mind, making them similar to custom software (Hartmann et al., 2009). When designing these ontologies, engineers focus on expected behavior in the application rather than on reuse and interoperability with other ontologies.
- *Difficulty of ontology reuse*: Ontologies trying to cover domains in the knowledge representation sense are often too big to be reused efficiently. These ontologies try to capture the complete domain knowledge whilst ontology engineers normally only need to reuse certain parts for their ontology.
- *Collaboration is missing while building ontologies*: Generally, little attention has been paid to formalisms and tools for collaborative ontology construction (Bao et al., 2004b). Most of the ontology engineering methodologies, reviewed in chapter 2 implement a centralized engineering model focusing on core ontology development activities and the participatory approach is missing.

Having presented the main problems facing the ontology building process, the following sections address the main issues that could help to solve these problems from our perspective such as the building of well-founded ontologies, the middle-out aspect of the ontology building process, the collaborative ontology development and the concept of data heterogeneity.

3.3 Well-founded Domain Ontologies

The concept of well-founded ontologies have raised mainly in Guizzardi's works, such as (Guizzardi, 2005; Guizzardi, 2007), where the author admit that ideally domain ontologies should be developed grounded in foundational (top-level) ontologies (Falbo, 2014). Therefore, concepts and relations in a domain ontology must be previously analyzed in the light of a foundational ontology. Consequently, for achieving the grounding purpose, there is a need for a well-known foundational ontology, and an ontologically well-founded conceptual modeling language for representing domain ontologies. The process of representing domain ontologies using these components is called ontology-driven conceptual modeling (ODCM).

In this thesis, UFO(see chapter 2, section 2.2.2.1) is selected as the most convenient foundational ontology for the application of ontology-driven conceptual modeling for three main reasons:

- Its successful application in a large number of domains ranging from natural science domains such as Petroleum and Gas and Electro-physiology of the

heart to social domains such as organizations, services and software (Griffo et al., 2015) (Guizzardi et al., 2010c; Genesereth et al., 1992);

- The fact that UFO comprises a rich theory of relations and complex relational properties that is absent in other foundational ontologies (Guizzardi et al., 2005a).
- The availability of a conceptual modeling language founded on this ontology (OntoUML) (see Figure 3.1).

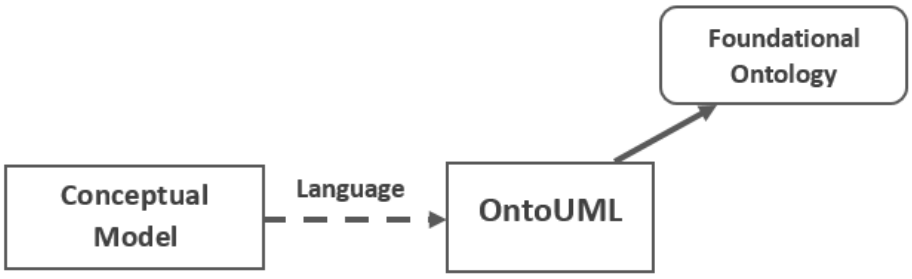


FIGURE 3.1: Conceptual modeling language founded on UFO.

In the following sections, we will discuss two essential concepts for the building of well-founded domain ontologies: the ontology-driven conceptual modeling (ODCM) and the ontologically well-founded conceptual modeling language (OntoUML).

3.3.1 Ontology-Driven Conceptual Modeling

"*A challenge to the modeling of ontologies is the lack of well-founded structural and temporal constructs of the conventional design techniques. Ontology-driven conceptual modeling has been successfully applied to overcome this issue, where ontological analysis based on a foundational ontology supports the development of well-founded ontologies*" (Moreira et al., 2016). Ontology-Driven Conceptual Modeling (ODCM) is firstly introduced by Guarino et al. (Guarino et al., 2002a). ODCM is still a relatively new research domain in the field of information systems, there is still much discussion on how the research in ODCM should be performed and what the focus of this research should be (Guizzardi et al., 2008b) and (Saghafi et al., 2014).

Generally, conceptual modeling is defined as "*the activity of representing aspects of the physical and social world for the purpose of communication, learning and problem solving among human users*" (Mylopoulos, 1992). In other words, conceptual modeling is concerned with identifying, analyzing and describing the relevant concepts and constraints of a domain with the help of a modeling language that is based on a small set of basic meta-concepts (Guizzardi et al., 2004b). In order to make conceptual modeling languages more suitable for representing the real world and less

oriented by systems, the attention of researchers have turned to philosophy where ontologies, dealing with the modeling reality, represent a branch of it (Verdonck, 2014). Therefore, the ontologies were introduced in order to provide a foundation for conceptual modeling by expressing the fundamental elements of a domain (Guarino, 1998). Moreover, ontologies are used to analyze and improve existing conceptual modeling languages (Wand, 1996). Thus, ontological or ontology-driven modeling is concerned with capturing the relevant entities of a small set of basic, domain-independent ontological categories (forming an upper level ontology) (Guizzardi et al., 2004b).

Recently, ontology-driven conceptual modeling is defined by Guizzardi as the utilization of ontological theories, coming from areas such as formal ontology, cognitive science and philosophical logics, to develop engineering artifacts (e.g. modeling languages, methodologies, design patterns and simulators) for improving the theory and practice of conceptual modeling (Guizzardi, 2012).

3.3.2 ONTOUML: Conceptual Modeling via UFO

One of the main success factors behind the use of a modeling language is its ability to provide to its target users a set of modeling primitives that can directly express relevant domain conceptualizations (Guizzardi, 2005). In order to make possible the activity of conceptual modeling via UFO(see chapter 2, section 2.2.2.1), a conceptual modeling language, named OntoUML (Ontological Unified Modeling Language) (Benevides et al., 2009b) is used. OntoUML was proposed by Guizzardi (Guizzardi, 2005) based on the need for an ontology-based language that would provide the necessary semantics to construct conceptual models using concepts faithful to reality.OntoUML uses the ontological constraints of UFO as modeling primitives and is specified above the UML2.0 meta-model (Guizzardi, 2005) (see Figure 3.2 and 3.3). The figure 3.3 presents a brief summary of some OntoUML's modeling primitives.

Stereotype	Corresponding Concept in UFO
<<category>>	Category
<<kind>>	Kind
<<collective>>	Collective Universal
<<rolemixin>>	Role Mixin
<<role>>	Role
<<mode>>	Mode Universal
<<relator>>	Relator Universal
<<event>>	Event Universal

FIGURE 3.2: A Subset of OntoUML Stereotypes (Nardi et al., 2016).

Stereotype	Description
<<kind>> A	A stereotype <<kind>> is a representation of a Substante Sortal whose instances are funcional complexes. E.g.: Person.
<<role>> B	A <<role>> represents an anti-rigid and relationally dependent universal. Every <<role>> must be connected to an association end of a <<mediation>> relation. E.g.: Student.
<<relator>> C	A <<relator>> universal is a relational moment universal. Every <<relator>> must be (directly or indirectly) connected to an association end on at least one <<mediation>> relation. E.g.: Marriage.
<<mediation>>	A <<mediation>> is a formal relation that takes place between a relator universal and the endurant universal(s) it mediates. E.g.: the relator universal Marriage mediates the role universals Husband and Wife.
derivation relation	A derivation relation represents the formal relation of derivation that exists between a material relation and the relator universal this material relation is derived from. E.g.: the material relation married to, derived from the relator universal Marriage.
<<material>>	A <<material>> relation is a relational universal which is induced by a relator universal. E.g.: a person is married to another person.

FIGURE 3.3: Some OntoUML Stereotypes (Teixeira et al., 2014).

In (Guizzardi, 2006), the authors show how a modeling language based on UFO can be used to address a number of semantic interoperability problems which cannot be handled by semantic web languages such as OWL and RDF. The Unified Foundational Ontology (UFO) is a foundational ontology that provides a sound ontological basis to evaluate and give real-world semantics to conceptual modeling language's constructs such as UML. OntoUML is a result of such evaluation. The class diagram fragment of UML 2.0 was re-designed and evaluated according to the structural layer of UFO (Guerson et al., 2015).

Over the years, OntoUML has been adopted by many research, industrial and government institutions worldwide (Guizzardi et al., 2015). OntoUML is a well-founded modeling language that allows modelers to formalize world-views in a technologically neutral way, aiding in the solution of such interoperability challenges (Benevides et al., 2009a) (Guerson et al., 2015). According to (Guerson et al., 2014), this language has been successfully employed in a number of industrial projects in several domains such as Petroleum and Gas, News Information Management, E-Government and Telecommunication.

In figure 3.4, a sample OntoUML diagram is depicted. To build, evaluate and implement OntoUML models, a model-based environment is needed such as the standalone tool OLED[1] (OntoUML Lightweight Editor) (Benevides et al., 2009a). This modeling tool has been developed as an academic effort for several years (Moreira et al., 2016). Recently, OLED has been entirely re-factored and transformed into

[1] Available at: https://code.google.com/p/ontouml-lightweight-editor/

a commercial tool, named Menthor Editor[2] (ontology-driven conceptual modeling platform) (Moreira et al., 2016).

There are two ways to model domain ontologies with the Menthor Editor (Moreira et al., 2016). First, the tool provides a class diagram interface with OntoUML stereotypes (see Figure 3.4). Second, Sparx's Enterprise Architecture[3] (EA) tool may be used for modelling, where the models may be exported to Menthor Editor using an OntoUML plug-in for EA, i.e. a UML profile that reflects OntoUML meta-model, implemented with the MDG technology[4].

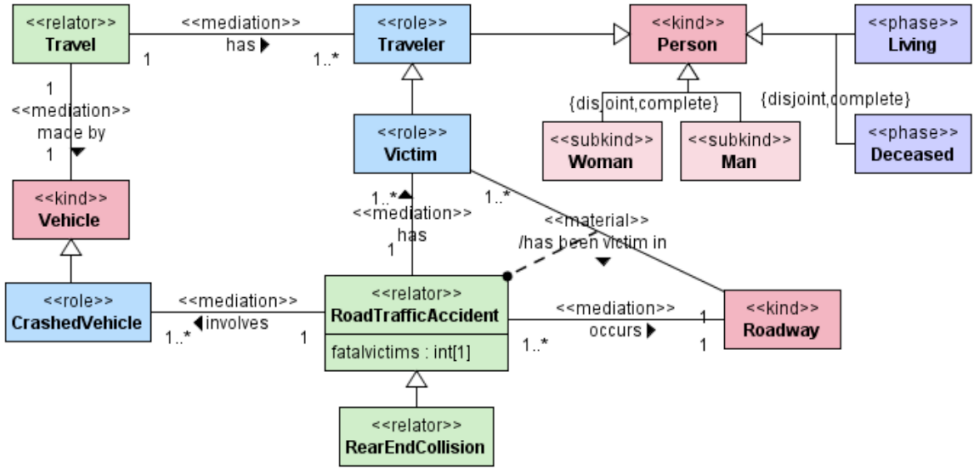


FIGURE 3.4: Example of OntoUML Diagram (Guerson et al., 2015).

Finally, the domain ontology implementation can be automatically generated in OWL and SWRL through model transformations, taking design decisions into account. Menthor Editor presents a set of settings to configure the transformation approach, including filters, axioms and data types' selection (Moreira et al., 2016).

3.4 Middle-out Ontology Engineering

Generally, ontology building methodologies are classified into two main categories: top-down and bottom-up (Francesconi et al., 2010).

- Top-down: the ontology construction starts by modeling the most generic concepts and build a structure by specialization. In this approach, the ontology building process starts by an analysis and study of relevant information sources about the given domain and then modeling the top level concepts which will be refined in next steps. This category of approaches is typically

[2]http://www.menthor.net/
[3]http://www.sparxsystems.eu/enterprisearchitect/
[4]http://www.sparxsystems.com.au/resources/mdg_tech/

carried out manually by domain experts and leads to a high-quality, reusable and shareable ontologies.

Although, a top-down approach results in better control of the level of detail. However, starting at the top can result in choosing arbitrary high-level categories which lead to a risk of less stability in the model.

- Bottom-up: start from the most specific concepts and build a structure by generalization. In this approach, the building process of the ontology usually starts with linguistic study on existing data structures forms (documents, reports, etc.) in order to extract relevant concepts of the domain and relations among them with the semi-automatic support in document analysis.

  This approach results in a very high level of detail which makes it difficult to spot commonality between related concepts and increases the risk of inconsistencies (Uschold et al., 1996). Moreover, the bottom-up approach is limited by developing domain-specific or application ontologies that are not reusable. Meanwhile, it can support the refining and expanding of existing ontologies by incorporating new knowledge emerging from texts(Francesconi et al., 2010).

According to (Francesconi et al., 2010), there is a complementarity between top-down and bottom-up approaches and preferring one approach over the other means ignoring complementary information that can help creating a better ontology. Thus, for better results in building comprehensive ontologies, there is a need to combine the two categories (Francesconi et al., 2010). Actually, this is a fact acknowledged in the literature, specifically in the studies of Uschold and Gruninger (Uschold et al., 1996), who include among their guidelines for ontology construction and merging the so-called *middle-out approach*, based on the combination of top-down and bottom-up ontology modeling. Therefore, a middle-out approach is a combination of top-down and bottom-up approaches leading to an integration of theoretical modeling and textual analysis. This approach strikes a balance in terms of the level detail. Detail arises only as necessary, by specializing the basic concepts (Uschold et al., 1996).

More recently, scholars advocating a middle-out approach to ontology construction started explicitly mentioning the "support of automatic document analysis" through which relevant lexical entries are extracted semi-automatically from available documents (Francesconi et al., 2010). The (semi-)automatic support in ontology development is nowadays referred to as *ontology learning*.

3.5 Collaborative Ontology Engineering

A collaborative ontology is defined as "two or more people interact and exchange knowledge in order to build a common, shared ontology in pursuit of a shared, collective, bounded goal" (Tudorache, 2007) (see Figure 3.5).

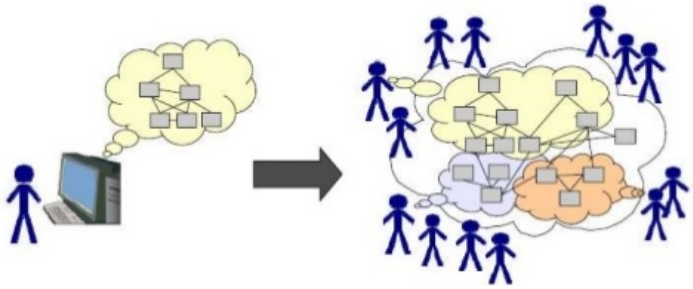


FIGURE 3.5: From stand-alone ontology to collaborative ontology (Tudorache, 2007).

In a collaborative ontology engineering scenario process, methods and tools are explicitly designed to support a decentralized group of stakeholders or community of interest in the sense of geographical dispersion, varying levels of skills, experience and responsibilities, as well as potentially divergent agendas to reach a consensus in an incremental and asynchronous fashion (Simperl et al., 2013).

A collaborative ontology engineering process typically starts with an analysis of the domain to be captured by the ontology, and of the requirements imposed by the ontology-based application as it is common in any other ontology engineering process. The team developing the shared ontology consists of stakeholders with different, and perhaps divergent, interests and complementary competencies. Classical ontology engineering distinguishes between three roles: domain experts, knowledge engineers and ontology engineers (Gomez-Perez et al., 2004).

- Domain experts are knowledgeable in the domain that is captured by the ontology; they have "intricate" knowledge about domain-relevant concepts and their attributes, as well as their interdependencies and relationships.
- Knowledge engineers try to obtain these insights from the domain experts, for instance via interviews, to create a conceptual model of the domain.
- The ontology engineers represent the conceptual model in a suitable knowledge representation language.

In this context, each member of the community can play several roles, depending on the types of contributions the respective individual is allowed to perform on the shared ontology, but also on the level of technology support in place and on the type of ontology that the project targets (Simperl et al., 2013).

In this regard, domain experts become increasingly involved in developing ontologies. This development is a natural consequence of ontologies covering domains in more detail, including information that ontology engineers simply do not know such as the domain of law, medicine, etc...

In the literature, several collaborative approaches are found such as methodology of Holsapple and Joshi (Holsapple et al., 2002), DILIGENT (Vrandecic et al., 2005) and HCOME (Kotis et al., 2004).

Approach	strategy for identifying concepts	Life cycle
Holsapple and Joshi	top-down	iterative
DILIGENT	middle-out	iterative
HCOME	bottom-up, ontology reuse, improvising of ontologies, alignment of multiple ontologies	iterative
Ontology Maturing	bottom-up	No life cycle model proposed

TABLE 3.1: A comparison of collaborative engineering methodologies.

3.6 Data Heterogeneity

In this section we overview the concept of data heterogeneity and the possible classifications founded in the literature. Generally, ontology engineering refers to the study of the "activities that concern the ontology development process, the ontology life cycle, and the methodologies, tools and languages for building ontologies" (Gomez-Perez et al., 2004). In other words, ontology engineering is the task of designing, implementing and maintaining ontology-based applications (Euzenat et al., 2007). Actually, this is the standard definition of the ontology engineering process. Meanwhile, recently, authors considered the distributed concept in the definition of this process. For instance, Euzenat (Euzenat et al., 2007) considered that the ontology engineering process represents the context where users are confronted with heterogeneous ontologies. Therefore, this activity requires support of ontology matching because ontology engineering has to deal with multiple, distributed and evolving ontologies. From this perspective, the ontology heterogeneity become an essential concept while designing an ontology for a domain of interest in the ontology engineering process. Thus, ontology-based system designers often have to integrate different ontologies, either for the sake of enforcing reuse, and thus not multiplying ontologies on the same topic, or because it is necessary for interconnecting various relevant resources (Euzenat et al., 2007).

In fact, there are different classifications of data heterogeneity. A number of researchers (Kim et al., 1991), (Kashyap et al., 1997), (Cui et al., 2000), (Hakimpour et al., 2001b), (Hakimpour et al., 2001a) and (Hakimpour et al., 2002) have classified heterogeneities into two main types: structural and semantic.

- Structural heterogeneity: means that different data systems store their data in different structures i.e. different data models.
- Semantic heterogeneity: involves discrepancies in the meaning of related data among heterogeneous systems. For example, two schema elements (i.e., classes or attributes) in two data sources can have different names, but the same meaning. Thus, during integration, these two elements may be treated differently even though they may refer to the same concept.

  According to (Goh, 1997), the semantic heterogeneity can be classified as follows:

  - Semantically equivalent concepts (the models use different terms to refer the same concept, e.g. synonymous; the properties are modeled differently by distinct systems, etc.).
  - Semantically unrelated concepts (the same term may be used by distinct systems to denote completely different concepts).
  - Semantically related concepts (generalization/specification, different classifications, etc.).

3.6.1 Ontology-based Approaches for Resolving Data Heterogeneity

Due to an increased awareness of ontology (Gruber, 1993), applications and the availability of multiple ontologies over same domain leads to semantic heterogeneities between ontologies. Ontology mapping has been the suggested solution to find semantic correspondences between similar elements of different ontologies thereby enabling semantic interoperability between them (Patel et al., 2005).

Ontology is considered to provide definitions for the term used to represent knowledge (Gruber, 1993), (Gruber, 1995), which consist of concepts, relations and their taxonomic hierarchies, also express constraints (Guarino, 1998). Therefore, explicit and formal definitions of the semantics of the terms guided researchers to apply formal ontologies (Guarino, 1998) as a potential solution to semantic heterogeneity. Thus, an ontology is considered to be suitable for information integration tasks because of its potential to describe the semantic of data sources and to solve the data meaning heterogeneity problems (Goh, 1997) and (Cui et al., 2000). Many ontology-based approaches exist for the integration of heterogeneous data sources. (Wache et al., 2001) categorized different methods using ontologies into three major approaches (see Figure 3.6):

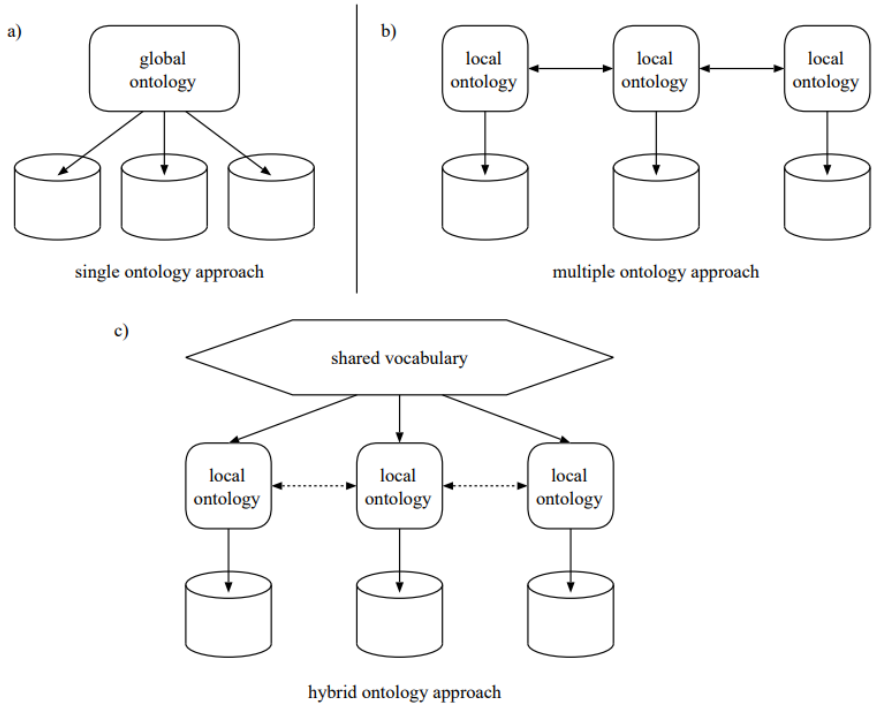


FIGURE 3.6: The three possible ways for using ontologies for resolving semantic heterogeneity (Wache et al., 2001)

- Single ontology approach (see Figure 3.6 (a)): in this approach, one global ontology provides a shared vocabulary for the specification of the semantics. All information sources are related to the global domain ontology. The global ontology can be a combination of several specialized ontologies. A reason for the combination of several ontologies can be the modularization of a potential large monolithic ontology.

  However, this approach requires that all sources have nearly the same view on a domain, with the same level of granularity. For instance, if one information source has a different view on a domain by providing another level of granularity, finding the minimal ontology commitment (Gruber, 1995) becomes a difficult task.

- Multiple ontology approach (see Figure 3.6 (b)): in this approach, the semantics of an information source is described by its own separate ontology. The source ontology can be a combination of several other ontologies but it cannot be assumed, that the different source ontologies share the same vocabulary.

  The advantage of this approach is that no common and minimal ontology commitment (Gruber, 1995) about one global ontology is needed. Each source ontology can be developed independently from the other sources or their ontologies. Thus, the ontology architecture can simplify the integration task and supports the change, i.e. the adding and removing, of sources.

On the other hand, the lack of a common vocabulary makes it difficult to compare different source ontologies. Therefore, there is a need for inter-ontology mapping in order to identify semantically corresponding terms of different source ontologies, taking into account different views on a domain.

- Hybrid ontology approach (see Figure 3.6 (c)): in this approach, information sources are described by local ontologies that are built from a global shared vocabulary that contains basic terms of a domain. This in order to make the local ontologies comparable to each other. Sometimes the shared vocabulary is also an ontology (Stuckenschmidt et al., 2009).

  The advantage this approach is that new sources can easily be added without the need of modification. It also supports the acquisition and evolution of ontologies. However, existing ontologies cannot easily be reused, but have to be redeveloped from scratch.

The following table summarizes the ontology-based approaches for resolving semantic heterogeneity:

Approach	Semantic Heterogeneity	Adding/Removing sources
Single-ontology approach	Similar view of a domain	need for some adaptation in the global ontology
Multiple-ontology approach	supports heterogeneous views	providing a new source ontology; relating to other ontologies
Hybrid-ontology approach	supports heterogeneous views	providing a new source ontology

TABLE 3.2: A comparison of Ontology-based approaches for resolving semantic heterogeneity

3.7 MIROCL Motivations

In this section, we discuss five main key challenges that MIROCL aim to solve in one cohesive approach which are: data heterogeneity, ontology modularization, ontology reuse, ontology learning and ontology integration.

3.7.1 Heterogeneity of Data sources in MIROCL

In order to prevent building domain ontologies totally from scratch, we proposed different heterogeneous data sources aiming at enriching the domain of discourse and the intended ontology as well. For this purpose, two main category of resources are identified in MIROCL:

- Textual resources related to the domain of application. For example, legislations and codes in the legal domain. From these resources, domain and domain-specific ontologies, that are not reusable, can be extracted using semi-automatic support tools.
- Existent validated ontologies such as foundational or upper-level ontologies that are abstract and common for all the domains and core ontologies that cover the higher levels of the application domain. These ontological sources are for enforcing ontology reuse and not reinventing the wheel. Core and upper-level ontologies, that can be reusable, are extracted from these sources.

Therefore, the main purposes of this heterogeneity are:

- The interconnection of various heterogeneous relevant resources aiming to enrich the ontologies to build.
- Enforcing the ontology reuse that will simplify the ontology building process.
- Encouraging the collaborative aspect of the ontology building process since the heterogeneous data sources need the intervention of various contributors such as domain experts and lexical engineers for the textual data sources and ontology and knowledge engineers for the ontology reuse process.
- The possibility of building ontologies that contain different level of details: (1) domain and domain-specific extracted from textual resources and (2) core and upper extracted from reusing existent general and core ontologies.

Therefore, facing this heterogeneity of data sources, there is a need for an ontology-based approach that could solve it. for this purpose, we will apply the Multiple ontology approach (see Figure 3.6 (b)). The main reasons for selecting this approach are:

- The possibility of describing the semantics of an information source by its own separate ontology. Each source ontology can be developed independently

from the other sources or their ontologies. This will support the collaborative building of ontologies and the reusability of the separated developed ontologies.

- The ontology architecture can simplify the integration task and supports the change, i.e. the adding and removing, of sources.

Finally, there is need for a selection process in order to identify the most relevant data sources. The textual resources should be related directly to the domain of discourse in order to express the most possibly the domain. The selection of the upper and core ontologies depend on different criteria:

- The content of the ontology to be reused.
- The possibility to apply partial reuse or extraction of ontology modules since it may be not necessary to apply a complete reuse known as *import* for the whole ontology.

The selection of the upper ontology is discussed in section 2.2.2. Concerning the core ontology, the selection of an existent validated ontology is related to the domain of application since this ontology represent the most common categories of a specific domain. We cite two main core ontologies in the literature: LKIF-Core for the law, and GENE ontology for the medicine. The selection process of the different data sources is performed by domain experts that recognize mostly the most convenient materials representing the domain. The ontology engineers have an essential role in the selection of the existent ontologies to be reused under the supervision of domain experts specifically for the core ontologies (see Figure 3.7).

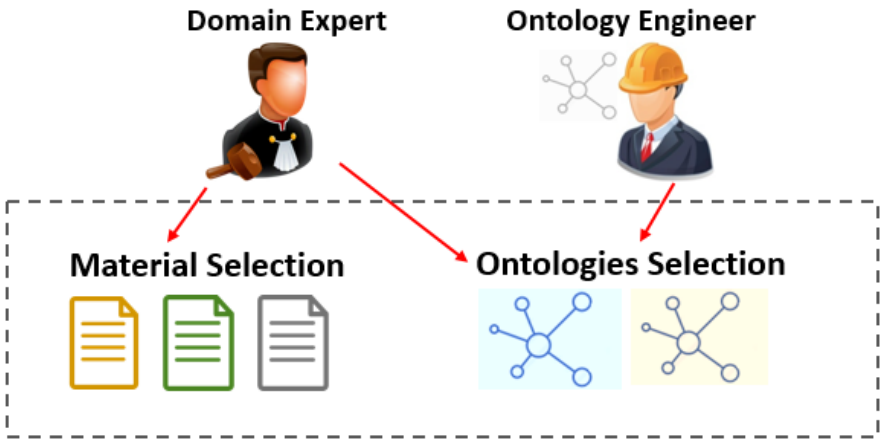


FIGURE 3.7: Contributors for the selection of heterogeneous data sources.

3.7.2 Ontology Modularization in MIROCL

Actually, interest in modularization techniques, as an ontology engineering principle, has increased in order to resolve the problems of reusability, scalability and maintenance of ontologies (d'Aquin et al., 2007). Three main reasons are discussed in the following for applying ontology modularization in MIROCL:

- Reusability: It is commonly known that ontologies aim to capture consensual knowledge of a given domain in a generic and formal way, to be reused and shared across applications and by groups of people (Corcho et al., 2007). In this context, the number of available ontologies has increased considerably in various domains such as bio-informatics, genetics, medicine and law, among many others, but they are also becoming larger and more complex to manage and reuse (Pathak et al., 2009). In the literature, several studies such as, (Turlapati et al., 2013) and (Bench-Capon et al., 1997), consider that in any realistic application, it is often desirable not to concentrate on creating one ontology for the domain, but on the creation of a library that contains several dedicated ontologies, developed independently, at different abstraction levels and supports their combination to create a composite ontology. This would allow for the modular design of large ontologies and would facilitate knowledge reuse tasks.

- Resolving data heterogeneity: The main use of ontologies is making the intended meaning of a given domain available to all agents. thus, an ontology conceptual architecture is required to represent this meaning. Meanwhile, there is a need to modularize the conceptual architecture dealing with the complexity of the domain such as heterogeneous knowledge with different levels of detail of that knowledge. Therefore, the resulted designed ontologies are obtained with a high quality (Gangemi et al., 2004).

- Collaborative aspect: By applying ontology modularization to an ontology building process, the collaborative effort is encouraged where different contributors can cooperate in building different modules independently in order to combine them at the end.

Therefore, ontology modularization has several benefits where modular representations are easier to understand, reason with, extend and reuse (Grau et al., 2007c), and using modularization reduces the complexity of designing ontologies and facilitates the ontology reasoning, development, and integration. Meanwhile, there is no universal way to modularize ontologies and that the choice of a particular technique should be guided by the requirements of the considered application (d'Aquin et al., 2007).

In MIROCL, we pursue the modularity strategy by taking into account the content and level of ontologies. Here, modular ontologies support design clarity by

specifying the different perspectives on a domain. i.e., each modularly-designed ontology provides the semantics for a particular view. As a result, the different ontologies are located in a layered architecture, connected by integration and can interact in a meaningful way. Thus, we pursued the ontology composition approach (see chapter 2, section 2.2.8.3), where the different ontology modules are developed independently, then assembled together to compose the target ontology. Note that the modularization process performed by ontology engineers and supervised by domain experts. A multi-layered modular architecture of the ontology is outlined by identifying the main modules, their level, number, type and criteria, as well as the knowledge to be represented in each module. MIROCL modularizes the domain ontology into four different modules: Upper Ontology Module (UOM), Core Ontology Module (COM), Domain Ontology Module (DOM) and Domain-Specific Ontology Module (DSOM) (see Figre 3.8).

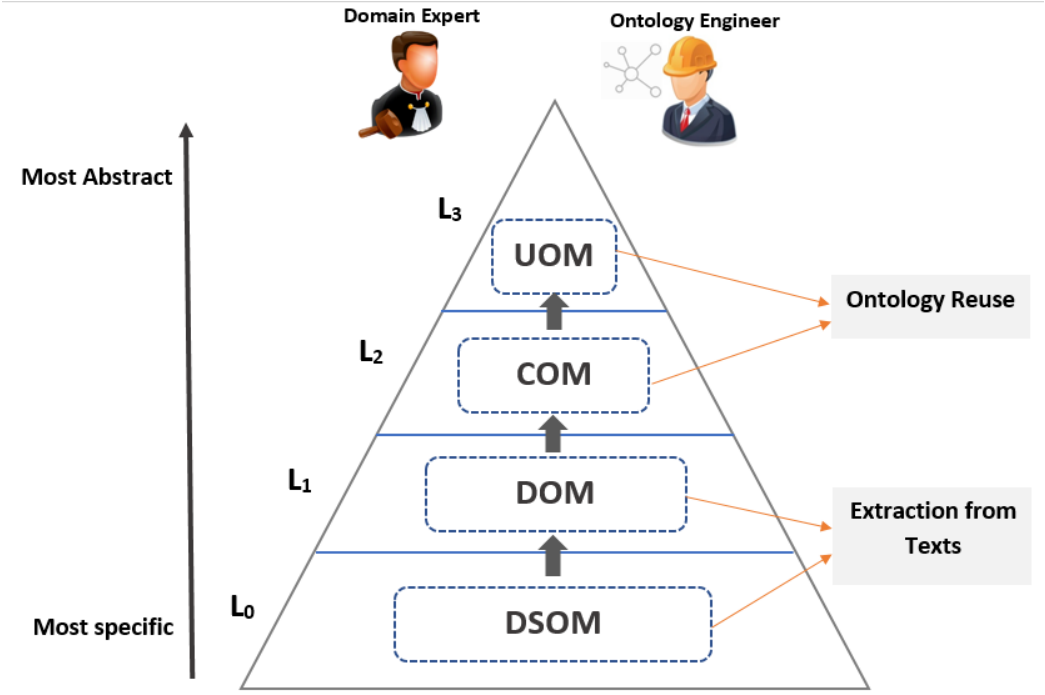


FIGURE 3.8: Ontology modularization in MIROCL.

- Upper Ontology Module (UOM): located at the higher level (L3). It consists of abstract concepts and relations which are effectively independent of any specific domain such as *Agent*, *Action*, *Event*, etc...For a well founded building of this module, a partial reuse of existent foundational, or top-level, ontology can help to facilitate and speed up the ontology development process by preventing to reinvent the wheel concerning basic categories.
- Core Ontology Module (COM): located at the level (L2). It consists of concepts and relations that are common across the domain of discourse and can provide the basis for specialization into domain and domain-specific concepts. Building COM is related to a reuse process from an existent validated core ontology

such as LKIF[5] for the legal domain or Gene Ontology[6] for the medicine.

- Domain Ontology Module (DOM): located at the level (L1). It is composed of categories that are related mainly to the domain of discourse such as criminal law for the legal domain. For building this module, two main strategies are applied:
  - specialize the concepts and relations of the core module;
  - extract the knowledge from textual resources using an ontology learning process in order to extract the relevant concepts and relations. This process is supported by semi-automatic techniques such as ontology learning tools and NLP (Natural Language processing) techniques.
- Domain-Specific Module (DSOM): located at the lower level (L0). It consists of the instances or individuals of a specific subject domain such as the Lebanese criminal system. Domain-specific ontologies are useful in systems involved with reasoning. Thus, they should be at higher level of expressiveness, in other words, rich in axioms. This module is developed using ontology learning techniques as well as manual strategy with the help of domain experts.

All the modules can be developed independently by different contributors such as domain experts, ontology engineers, knowledge engineers and lexical engineers.

3.7.3 Ontology Reuse in MIROCL

Ontology reuse process, is considered as one of the important research issues in the ontology field (Doran, 2009), and is recommended as a key factor to develop cost effective and high quality ontologies. Actually, ontology reuse reduces the cost and the time required for building ontologies from scratch (Ben Mustapha et al., 2013), (Bezerra et al., 2009). Moreover, by reusing validated ontology components, the quality of the newly implemented ontologies is increased. For the purpose of building well-founded domain ontologies, the reuse process in MIROCL is applied on two main levels:

- Building the content of the upper module by reusing general common concepts from foundational ontologies and the content of the core module by reusing domain core concepts from core ontologies.
- Grounding the upper and core modules in an existent validated foundational ontology such as UFO.

[5]https://github.com/RinkeHoekstra/lkif-core
[6]http://www.geneontology.org/

These two processes are performed starting from the most general concepts down to the core concepts. Thus, the reuse process is considered as a top-down strategy for building the upper and core ontology modules (see Figure 3.9).

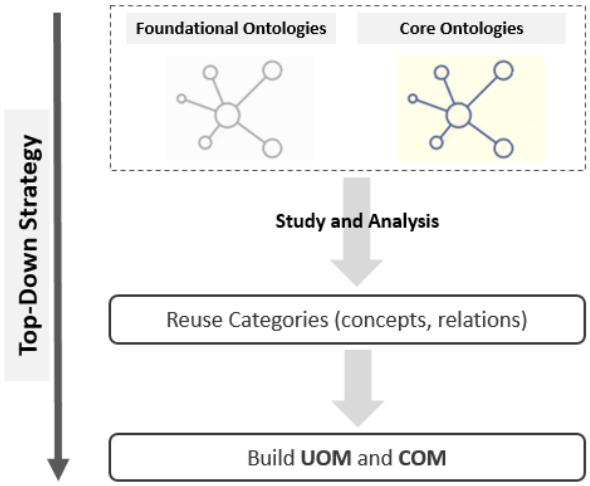


FIGURE 3.9: Reusing upper and core ontologies: top-down strategy.

3.7.3.1 Ontology Reuse for Building Ontology Modules

The ontology reuse process for building ontology modules can be performed completely of partially where an existent validated ontology can be imported with all the content or extracted partly as module extraction for instance. In MIROCL, the reuse process is performed partially where ontology categories or modules are extracted for building new ontologies. For instance, the concepts of the unified foundational ontology UFO are reused partially for building the upper ontology module (UOM). In figure 3.10, an example of simple reuse process is depicted. For building an upper ontology module, there is a need for some general concepts that are not related directly to a specific domain of application such as `Agent`, `Object`, `Action`, etc.. Such concepts can be reused from UFO (see section 2.2.2.1 for more details), specifically from the layers UFO-B and UFO-C. Then, the extracted concepts can be specialized in the target ontology or merged with other concepts in order to add new ones. This process depends mainly on the purpose of the intended ontology.

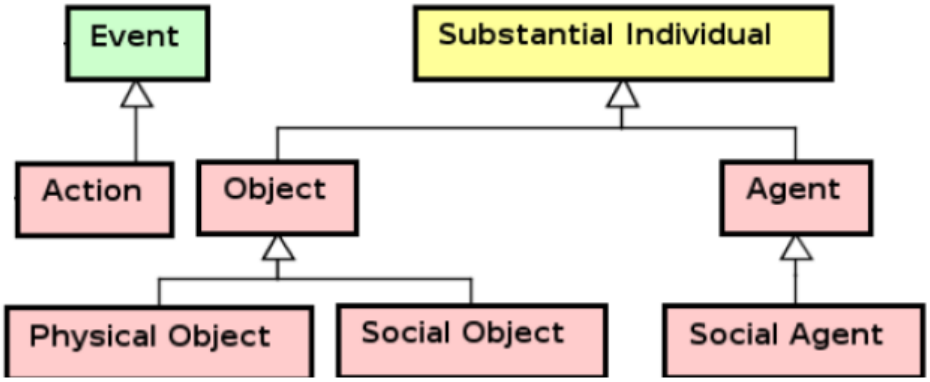


FIGURE 3.10: Simple reuse process from UFO.

3.7.3.2 Ontology Reuse for Grounding Ontologies

Ontology grounding is introduced by Harnad (Harnad, 1990) who claimed that existing approaches to ontology design pose the classical symbol grounding problem. Harnad wondered how a logical theory of a concept, that can be explicit, easier to communicate and axiomatized, is feasibly related to a human understanding of that same concept and avoiding constructing an abstract theory or model for another model. Moreover, how are its primitives grounded outside the formal system? In other words, how the semantic interpretation of a formal symbol system can be made intrinsic to the system, rather than just parasitic on the meanings in human head (Harnad, 1990). In this context, some studies such as (Kohn, 2003) illustrated the ontology grounding by avoiding resorting endlessly from one formal system to another in explaining the meaning of symbols. They claim that if ontologies are not grounded in something that their users share, they will be of very limited practical use. Therefore, ontology engineering methods have to supply a list of concepts (or at least of the kinds of concepts) considered meaningful outside the formal theories (Kohn, 2003).

Our concern is to build a well-grounded domain ontology that could be expanded, compared or merged with other ontologies. According to (Guarino, 1998), ideally domain ontologies should be grounded in foundational ontologies. Thus, ontology grounding using foundational ontologies refers to the reuse process of their basic categories. It can be expressed by the application of foundational ontologies in conceptual modeling for building domain ontologies. This process is called Ontology-Driven Conceptual Modeling (ODCM) discussed in section 3.3.1. Therefore, a foundational ontology (UFO) and a conceptual modeling language (OntoUML) are needed for grounding domain ontologies in MIROCL.

In figure 3.11, we illustrate the concept of grounding the core ontology module (COM) in UFO by reusing its basic categories such as kind, sub-kind, role and phase.

Thus, the concepts of the core module will be represented in the context of UFO. This will simplify the integration process of the upper and core ontology modules since they will be defined in the same conceptual modeling language (OntoUML).

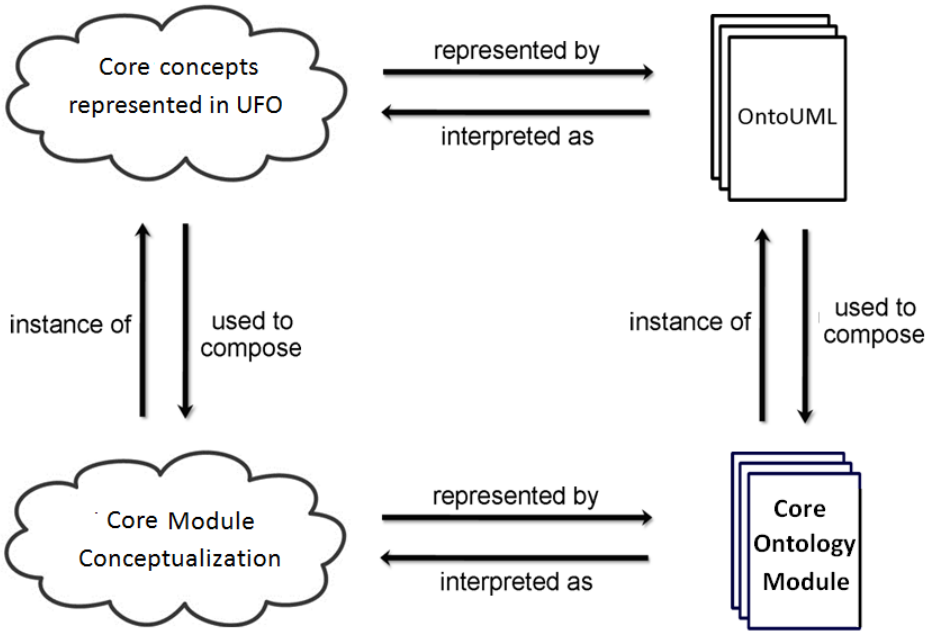


FIGURE 3.11: Representing core concepts in UFO.

3.7.4 Ontology Learning from Textual Resource in MIROCL

Generally, the knowledge expressed and conveyed in texts using domain-specific terminology does not provide a well-defined structure to be used by machines for reasoning tasks. Meanwhile, the extracting and mining of this terminology will lead to a certain domain representation model such as ontology (Madche et al., 2000).

In this context, extracting and maintaining ontologies manually remains a resource-intensive, time consuming and costly task. This is due to the difficulty in capturing knowledge, also known as the "knowledge acquisition bottleneck". In order to reduce the cost of creating and maintaining ontologies, there is a need for ontology learning (OL) supported by semi-automatic methods and tools. Ontology Learning greatly facilitates the construction of ontologies by the ontology engineer (Maedche et al., 2001). For this reason, we defined in MIROCL a bottom-up strategy that starts from textual analysis and conceptual representation of the intended meaning of the available resources by applying an ontology learning process supported by natural language processing techniques for building semi-automatically the domain and domain-specific modules of the target ontology. This process tends to extract from texts, the relevant categories of ontologies: concepts, taxonomies, relations, axioms

and instances. Thus, the main output of the OL process is a structured content represented in an explicit formal way.

However, even after a comprehensive literature review, we found a difficulty to define a complete approach or tool that can totally extract domain and domain-specific ontologies from textual resources. Additionally, there is no guarantee that the (semi-)automatically generated ontology is correct and precise enough to characterize the domain in question (Rudolph et al., 2007). For this reason, the intervention of ontology engineer and legal expert during the ontology learning process is required in order to supervise the work and to verify the obtained information. Furthermore, a re-engineering methodology is needed in order to enhance the results by transforming the resulted ontology into a new more correct, complete and expressive ontology.

From these perspectives, we defined five main phases of the ontology learning process (see Figure 3.12):

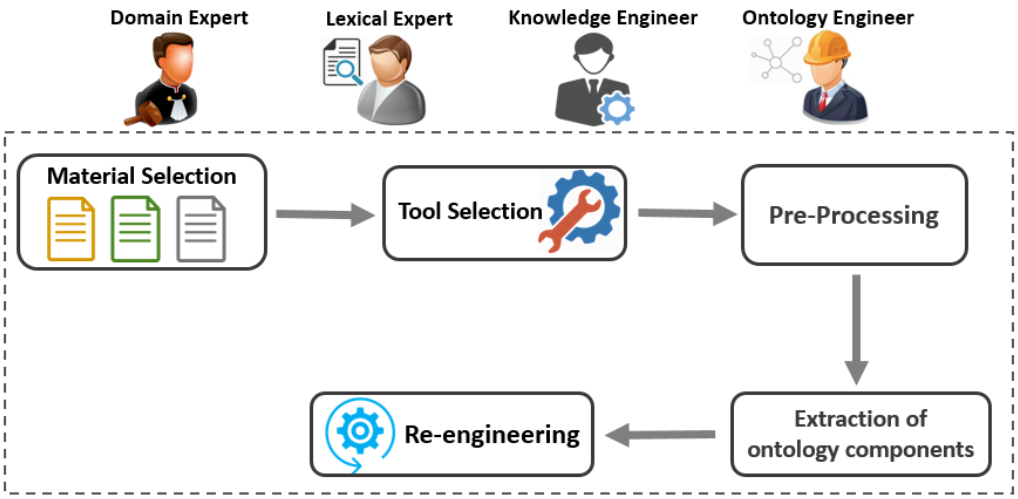


FIGURE 3.12: The main phases of ontology learning phase.

1. Material selection: in this phase, the material related to the context of interest is selected based on the requirements of the application, taking into account the type of documents and their language.

2. Tool selection: there is a need for a semi-automatic tool that supports the user in order to reduce the complexity of the ontology building process. The selection of an appropriate tool is related to the requirements of the ontology and the intended results.

3. Documents pre-processing: the purpose of this phase is to prepare the corpus and remove the ambiguity by filtering out worthless symbols and words, in order to extract meaningful textual content from the input documents. This

phase is performed using machine learning approaches with basic linguistic processing such as tokenization or lemmatizing and shallow parsing.

4. Extraction of the main components of the ontology: this phase is performed with the support of the selected tool by implementing list of algorithms and techniques.
5. Re-engineering phase: after extracting the ontology semi-automatically from textual resources, there is a need to correct, prune and enrich the results.

The contributors involved in this process are:

- Domain experts, such as lawyer or judge for the legal case for better selection of the convenient materials and pre-processing phase.
- Lexical experts and Knowledge engineers for managing the pre-processing and tool selection phases.
- Ontology engineers for the extraction of ontology components.
- The re-engineering phase can be performed under the supervision of domain experts and ontology engineers as well.

The ontology learning process from text in MIROCL is performed starting from texts to semantic level in a bottom-up strategy (see Figure 3.13).

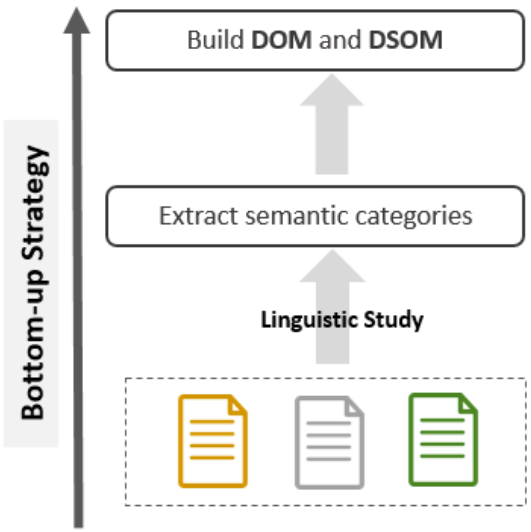


FIGURE 3.13: Ontology learning from text: bottom-up strategy.

3.7.5 Ontology Integration in MIROCL

One of the main objectives of MIROCL is to integrate information from heterogeneous sources. More specifically, integration of independently developed local

ontologies, or modules, into a global ontology.

Authors, such as, Pinto and her colleagues (Pinto et al., 1999) elaborate on issues concerning ontology integration. They clarify the term 'integration' terminologically and how it has been used in different works. Thus, three meanings of ontology 'integration' are identified (Kalfoglou et al., 2003):

- Building a new ontology by reusing (assembling, extending, specializing or adapting) other ontologies already available;
- Building an ontology by merging several ontologies into a single one that unifies all of them;
- Building an application using one or more ontologies.

In MIROCL, the integration process is established by building different ontologiy modules, semantically heterogeneous, then assembling them into one global ontology. In order to make the integration accessible through the uniform interface of the global ontology, semantic mappings are established between the global ontology and the local ontologies, or modules. Thus, this mapping process is accomplished during the construction of the global ontology, which is generated by merging the local ontologies. We consider that each local ontology (module) is merged into the global target ontology. The process of ontology merging consists of several operations:

- Copying a class and/or its properties: classes and properties that do not exist in the target ontology are copied into it.
- Class Generalization: related classes in the local and target ontologies can be generalized into a superclass.
- Class Merging: conceptually equivalent classes in the local and target ontologies are combined into one class in the target ontology.
- Property Merging: conceptually equivalent properties of a class in the local and target ontologies are combined into one property in the target ontology.
- Relationship Merging: conceptually equivalent relationships from one class c1 to another class c2 in the local and target ontologies are combined into a single relationship in the target ontology (i.e., an RDF property having c1 as its domain and c2 as its range).

Subsequently, in the resulting ontology, regions that were taken from the integrated modules can be identified. Moreover, it is essential to consider heterogeneity resolution and related ontology matching or mapping strategies to be an internal part of ontology integration (Caldarola et al., 2015). In this context, list of semantic mappings will be created among concepts of the different modules. Generally, the mapping concept is defined by (Kalfoglou et al., 2003) as a morphism. Meanwhile,

in this study a more loosely definition is used based on some works in the literature such as (Dmitrieva et al., 2011) and claims such as "simple mappings methods are sufficient and outperform more complex methods" (Ghazvinian et al., 2009). For ontology mappings, several studies, such as (Euzenat, 2007) and (Borgida et al., 2003), have proposed a number of specialized semantics. Meanwhile, for other studies, such as (Jimenez-Ruiz et al., 2008), ontology mappings are represented as OWL2 axioms of the form `subClassOf`, `EquivalentClass` and `DisjointClass` (Grau et al., 2008).

In MIROCL, the modules are located on different vertical conceptual levels, or layers, from general (upper module), located at the higher layer, to specific (domain-specific module), located at the lower layer. For this reason, the mappings will be based mainly on a parent-child, or subsumption, hierarchical (Wang et al., 2010) and instanceof relationships. These mappings are established manually as structural axiom of the form `subClassOf` and `instanceOf`. Thus, the hierarchical relationship is established among the concepts of modules. For this purpose, a linguistic-based matcher, such as WordNet, is used to deal with ontology mapping for calculating the similarity values between concepts (Miller, 1995). Then a domain expert, knowledgeable about the semantics of legal concepts, validates the proposed mappings. Given two concepts C_i and C_j from the modules UOM and COM respectively, if C_j is considered as a subclass of C_i, then the `subClassOf` axiom is added between the two concepts in the resulting ontology. Concerning the `instanceOf` axiom, it is performed between DSOM and DOM where the instances are located at the lower layer (DSOM). Given an instance I_k from DSOM, it is considered as an instance for a concept C_k from DOM, then a `instanceOf` axiom is added between them.

Therefore, the main contributors of the integration process are domain experts knowledgeable about the domain, lexical experts for the linguistic-matching part. The knowledge and ontology engineers are concerned in performing the integration process using ontology editors such as Protégé for the domain and domain-specific modules, and OLED for the upper and core modules.

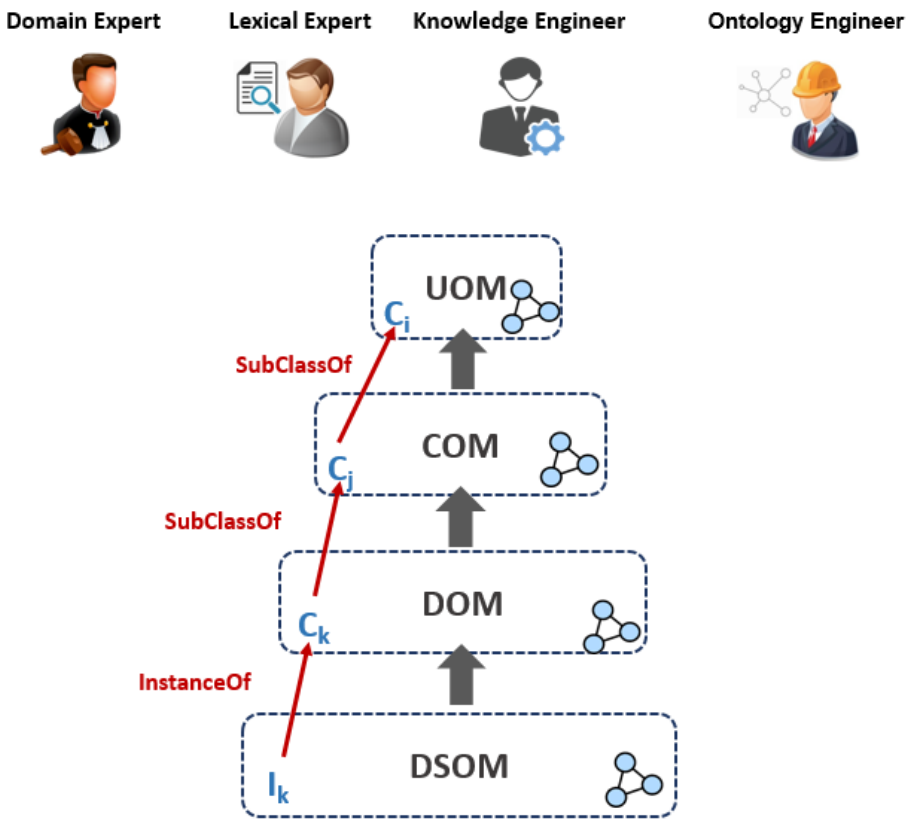


FIGURE 3.14: Example of Ontology modules integration.

3.8 MIROCL Aspects

Based on the application of the aforementioned ontology building support processes, we consider that two main aspects characterize MIROCL: middle-out and collaborative.

3.8.1 Middle-out Aspect of MIROCL

MIROCL is considered as a middle-out approach combining two independent complementary strategies: top-down and bottom-up (see Figure 3.15). This observation is based on the independent building of the different modules of the target ontology, where each module is related to a predefined data source located on a different level of detail. Then, the modules will be integrated together to compose the global ontology.

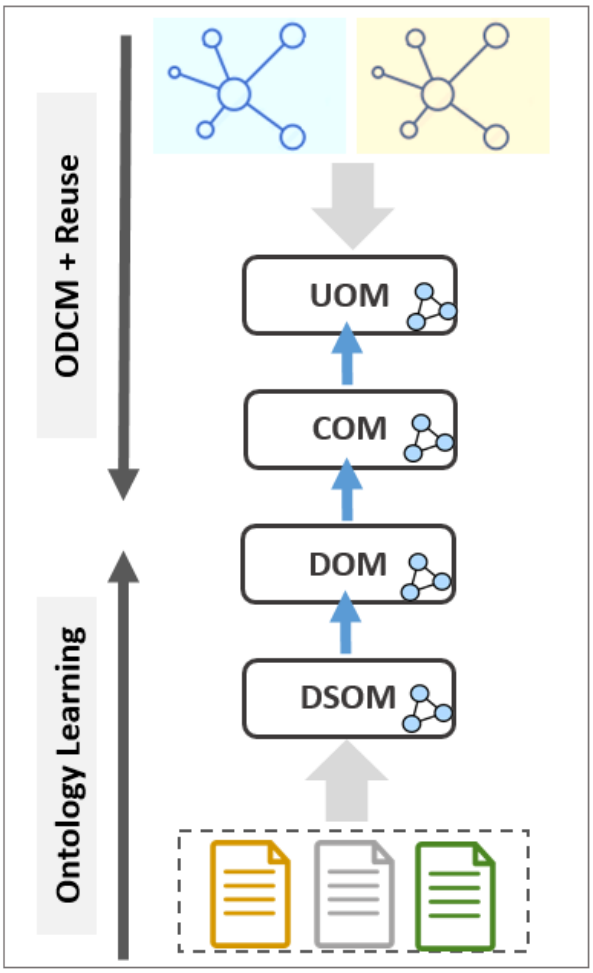


FIGURE 3.15: Middle-out approach for building ontology modules.

- Top-down: starts from the most general concepts, represents two main processes: (1) the ontology-driven conceptual modeling process (ODCM) guided by reusing the unified foundational ontology UFO for grounding the target domain ontology and (2) the ontology partial reuse of the components of the existent validated ontologies (foundational and core) for building the upper and core ontology modules.
- Bottom-up: starts from the most specific concepts that are related closely to the domain of discourse. It aims to build the domain and domain-specific modules using the ontology learning process from the available textual resources.

3.8.2 Collaborative Aspect of MIROCL

After a deep study of the available ontology engineering approaches in the literature, and inspired by different studies in the field such as (Bao et al., 2004a) and (Simperl et al., 2013), we consider that for building well-founded and rich domain ontologies,

there is a need for a truly collaborative effort carried out by different contributors handling heterogeneous data sources, or indirect cooperation through reuse or adaptation of previously published developed ontologies. Actually, the recent definitions of ontology engineering process moved toward the cooperative efforts in ontology building where users are confronted with integrating heterogeneous ontologies either for the sake of enforcing reuse, and thus not multiplying ontologies on the same topic, or because it is necessary for interconnecting various relevant resources (Euzenat et al., 2007).

Therefore, building ontologies in a collaborative and increasingly community-driven fashion has become a central paradigm of modern ontology engineering (Simperl et al., 2013). This will provide the technological support that makes it easier for non-experts to become involved in ontology-related activities beyond requirements elicitation.

In MIROCL, the collaboration is included as an integral part of the ontology development itself. It is performed on two levels: (1) direct cooperation of various contributors such as domain experts, lexical experts, knowledge engineers and ontology engineers (depicted in Figure 3.16) and (2) indirect cooperation through the ontology reuse process of existent validated ontologies (see section 3.7.3). These activities required support such as ontology modularization, matching and integration because it has to deal with multiple, distributed and evolving ontology modules (Euzenat et al., 2007) (Stuckenschmidt et al., 2009).

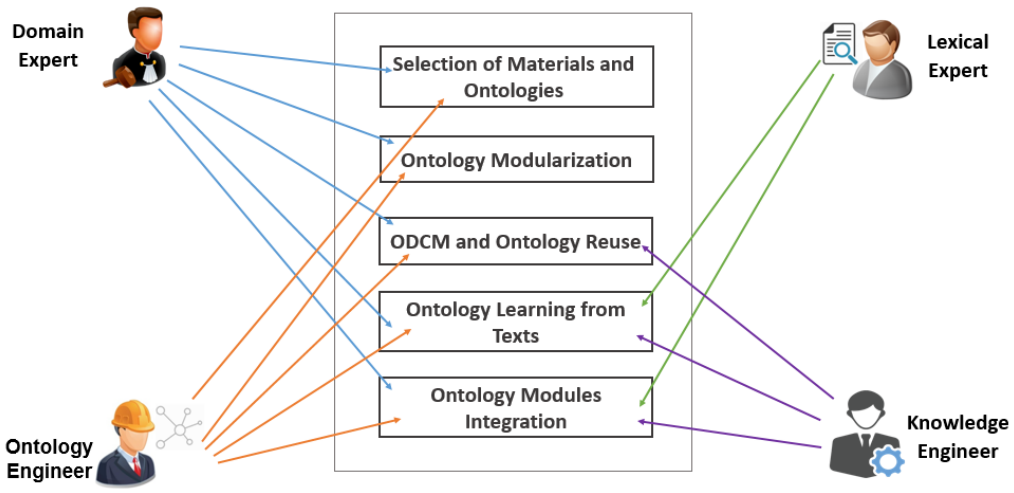


FIGURE 3.16: Collaboration of various contributors.

We conclude that there is an active involvement of domain experts during all the life-cycle of MIROCL. They lead the process, as well as providing the relevant conceptual knowledge specifically in collaboration with the ontology engineers.

3.9 Life-Cycle of MIROCL

After discussing the main key challenges that MIROCL aim to solve, which are data heterogeneity, ontology modularization, ontology reuse, ontology learning and ontology integration, its overall process can be summarized by a generation of independent local ontologies, or ontology modules, for each data source and by different contributors and then the use of a global merged ontology which defines the integrated underlying distributed heterogeneous data sources. The global merged ontology provides a unified representation of all underlying modules and will be utilized as a knowledge base for a legal reasoning and decision support system.

In such settings, typically, different participants have only partial knowledge of the domain, and hence can contribute only partial ontologies of the domain. A common task involves refinement of a predefined ontology. Another common task involves integration of several such partial ontologies to obtain a coherent ontology that covers a much larger portion of the domain. Semantic mismatches and logical inconsistencies between independently developed ontologies are unavoidable. Thus, there is an urgent need for principled approaches and flexible tools for allowing individuals to collaboratively build, refine, and integrate existing ontologies as needed in specific contexts or for specific applications e.g., data-driven knowledge acquisition from semantically heterogeneous, distributed data sources (Bao et al., 2004a). Therefore, the proposed approach MIROCL, depicted in figure 3.17, consists of four main phases:

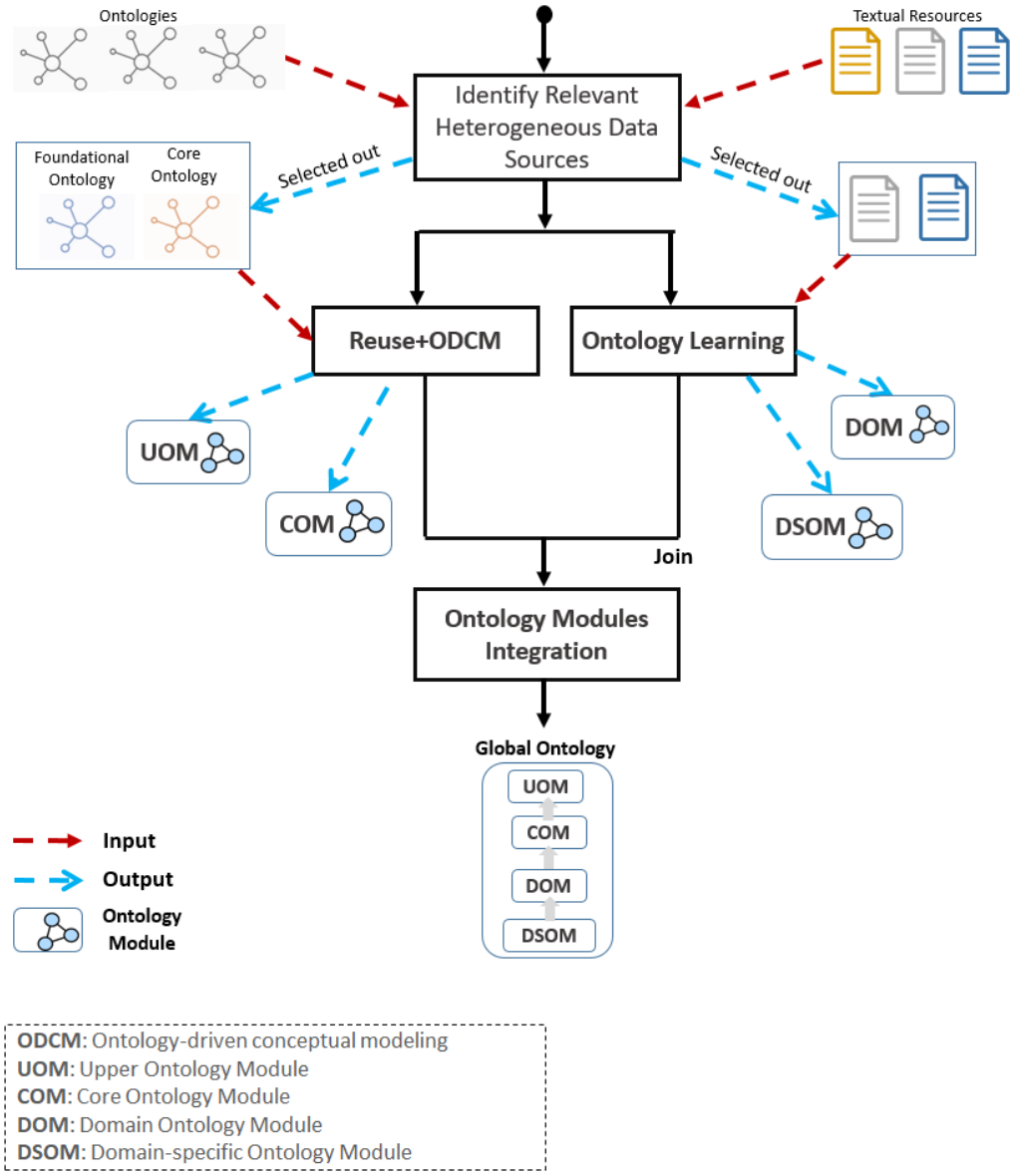


FIGURE 3.17: The life-cycle of MIROCL.

1. Identify relevant heterogeneous data sources: the identified data sources are divided mainly into two categories, pre-existing validated ontologies and textual resources related to the domain of context. In this phase, the domain experts and ontology engineers select the most relevant sources for the efficiency of the study. The output of this phase comprise a relevant textual resource that reflects accurately the domain of discourse as well as a foundational and core ontologies for reuse purposes.

2. Ontology reuse and building of ontology modules: in this phase, a partial reuse process is applied for reusing categories from the unified foundational ontology UFO and a core ontology related directly to the domain of discourse in order to build the upper and core ontology modules (UOM and COM).

Therefore, the main outputs of this phase are two independent ontology modules. The main contributors at this phase are: ontology engineers, domain experts and knowledge experts. This phase can be performed in parallel with the ontology learning phase that will be detailed in the next phase.

3. Ontology learning process and building of ontology modules: this process extracts the relevant concepts and relations from the textual data sources of the domain with the support of semi-automatic tools and NLP techniques. The main outputs of this phase are the domain and domain-specific modules (DOM and DSOM). Thus, two independent developed modules are derived from this phase. The ontology learning process should be performed under the supervision of all the contributors.

4. Integration of ontology modules: the last phase tends to integrate the different ontology modules in a global ontology. The integration process should be semantically coherent combination of the local ontologies. Thus, one global ontology is obtained at the end of this process which is considered as composite ontology composed of four modules, developed independently and located at different abstraction levels. This phase is accomplished by all the contributors.

3.10 Conclusion

The main goal of this thesis is to build a well-founded domain ontology for legal reasoning decision support system. In this chapter, we have presented and discussed a novel modular middle-out approach, named MIROCL, for building well-founded domain ontologies. Our concern is to solve some problems facing the ontology engineering domain concerning the complexity of ontology building process, the difficulty of reusing existent ontologies, as well as the lack of building well-founded ontologies and the missing of the participatory approaches in this field.

From our perspective, solving these issues can enhance and simplify the process of building well-founded domain ontologies. Based on this prospect, we suggested to enrich the ontology building process by some useful concepts in the field such as data heterogeneity, middle-out ontology building strategy, ontology-driven conceptual modeling and collaborative ontology development. For this purpose, four main ontology support processes are applied in a coherent way in order to achieve our goal. These processes are: ontology modularization, ontology reuse, ontology learning from texts and ontology integration.

Regarding the ontology modularization process, we suggested to modularize the ontology into four modules, upper, core, domain and domain-specific, to be developed independently and are located at different levels of granularity. The application of

this process have several benefits such as simplifying the building process of the ontology modules as well as of the target ontology, participating several contributors in the building process and enforcing the reusability of the ontology modules specifically the upper and core ones.

Concerning the ontology reuse process, it is applied in MIROCL on two levels, the simple one that consists of reusing concepts from existent validated ontologies (foundational and core ontologies) in order to simplify the development of ontologies and not to reinvent the wheel. The second level of the reuse process is the application of ontology-driven conceptual modeling based on existent foundational ontology such as UFO for well grounding the target ontology. This level aims to build well-founded ontologies using an ontology-based conceptual modeling language such as OntoUML.

The purpose of proposing the application of ontology learning process is to extract, semi-automatically with the support of NLP techniques, domain and domain-specific ontology components directly from textual resources that are related directly to the domain of discourse. This process is the most suitably technique for reflecting the domain under the supervision of the domain experts.

Finally, the ontology integration process is applied for combining the developed ontology modules into one global ontology using `subClassOf` and `instanceOf` axioms.

Furthermore, the validation of the proposed approach MIROCL will be accomplished in chapter 4 for building a well-founded legal domain ontology as well as in chapter 5 for testing the developed ontology in rule-based reasoning purposes.

Chapter 4

CriMOnto: A Criminal Ontology for Modeling Legal Norms

Contents

4.1 Overview

In this chapter, the building process of a well-founded legal domain ontology named CriMOnto is discussed. The aim of CriMOnto is modeling the content of legal norms. Furthermore, CriMOnto will be used as a ground for building a rule-based legal reasoning model. The domain application of this work is the Lebanese criminal system. Thus, the available domain sources are the legal norms of the Lebanese criminal code. Therefore, the concept of modeling legal norms is discussed in section 4.2, then the existent approaches are outlined in section 4.3. A proposed bi-phased ontology-based approach for modeling legal norms is presented in section 4.4. For the current, the first phase of the proposed approach is presented in section 4.6 which promotes the building of the legal domain ontology CriMOnto. Finally, section 4.7 explores the related works and section 4.8 concludes the chapter.

4.2 Modeling Legal Norms

Generally, norms are defined as an abstract mandatory command concerning rights or duties (Kelsen, 1991). For Boer and his colleagues (Boer et al., 2005), "*the norm is an epistemological concept identified by its role in a type of reasoning and not something that exclusively belongs to the vocabulary of the legal domain* ". Within a legal system, the role of norms is to specify how and when the chosen behavior agrees with the basic principles of the legal system (Boella et al., 2005). According to some studies, such as (Gostojic et al., 2013), a legal norm is a rule of conduct of people that may contain a rule on the application of a sanction in the case of its violation. This definition is limited to the regulatory form of a legal norm. Meanwhile, some studies such as (Von Wright, 1963) and (Biagioli, 1997) classified legal norms into three main types:

- Determinative or constitutive: define concepts or constitute activities that cannot exist without such norms. Legal norms are constitutive when they operate within the legal system defining it in order to describe legal institutes (Biagioli, 1991).
- Technical or procedural: state that something has to be done in order for something else to be attained.
- Regulative or normative: these norms are similar to orders. They regulate actions by making them obligatory, permitted, or prohibited in a direct and imperative manner and are normally followed by provisions providing for sanctions (Biagioli, 1997). These norms are aimed explicitly at the citizen, at the point of contact or union between the legal institute and the real regulated world, of which it represents a model (Biagioli, 1991).

The modeling of legal norms consists of interpretation of text's meaning in order to transform the norms in logical rules for legal reasoning. It is commonly known that legislative documents are semi-structured and hierarchically structured in nature. In this context, the structure of the document consists of normative parts rather than simply textual documents which facilitates the understanding of legal concepts and thus the interpretation of text (Heflin et al., 2000) (Ouksel et al., 1999).

In the legal domain, three conceptual layers are distinguished: norms, textual provisions and rules (Palmirani et al., 2012).

- Legal norms: abstract mandatory commands concerning rights or duties. They are usually expressed in written using legal text.
- Textual provisions: instantiation of the norms in one possible textual representation (sentence, article, and paragraph).
- Legal rules: interpretation of the textual provisions formalized using logical rules in the form of antecedent and consequent. Sometimes several provisions will form a single rule, or a single provision may include multiple rules (Palmirani et al., 2013).

The legal norms are expressed in textual sources such as legislations and codes and have basically the following structure (Davis et al., 1984), (Kelsen, 1991):

```
If A1, ...,A2 then B;
```

where "`A1, ..., A2`" are the conditions of the norm, "`B`" is the legal effect and "`if ...then`" is a normative conditional.

This view highlights an immediate link between the concepts of the norm and the rules (Gordon et al., 2009). This link relies on *ontologies* since they are used for filling the gap between document representation, expressed in natural language, and rules modeling using logical formalisms (Palmirani et al., 2009). Thus, the legal rules are considered as legal interpretation and modeling of the meaning of texts by transforming the legal norms to logical rules for permitting reasoning (Palmirani et al., 2012).

However, according to (Palmirani et al., 2012), the scholars in the domain of AI & Law, have focused only on the rules modeling and on the foundational logical theory, and apart the isomorphism principles (Bench-Capon et al., 1992) and neglected the ontology aspects. Actually, there is a theoretical and important debate in the AI & Law community on the interpretation of the legal textual provisions and on formalizing of the rules using logical formalisms (Boella et al., 2011).

4.3 Approaches for Modeling Legal Norms

There are a variety of approaches in the literature that deal with the translation of legal rules expressed in natural language to a machine-processable formal representation format for reasoning purposes. In this section, three main approaches are presented: Palmirani and her colleagues (Palmirani et al., 2009; Palmirani et al., 2012), (Francesconi, 2010; Francesconi, 2011) and (Wyner et al., 2013) .

Palmirani and her Colleagues: Fill the Gap between Texts and Rules The approach proposed by Palmirani and her colleagues is under the "*Fill the gap*" project (Palmirani et al., 2009; Palmirani et al., 2012) that aims to design a set of XML standards for modeling legal documents in the Semantic Web. The authors have used three main concepts: (1) the translation of textual provisions into XML using Akoma Ntoso[1] (Vitali et al., 2007), (2) developing the corresponding rules in LegalRuleML (Palmirani et al., 2011), and (3) an ontology for modeling and defining macro-concepts specific for the legal domain is developed and expressed in LKIF-Core. The approach is applied on a fragment of the legal norms of the US copyright domain.

Francesconi: Learning Legal Rules based on Semantic Model of Legislation An approach to support the acquisition and modeling of legal rules contained in legislative documents is proposed by Francesconi (Francesconi, 2010; Francesconi, 2011). The author considers that the extraction of legal rules from legislative texts can be an effective method to make it easier the implementation of rules-based systems for legal assessment and reasoning, as well as for implementing advanced search and retrieval systems for legislative documents. For this purpose, Francesconi's approach is based on machine learning and NLP techniques for extracting legal rules on the basis of a semantic model for legislative texts, which is oriented to knowledge reusability and sharing.

This approach combines two strategies: top-down and bottom-up. The top-down strategy provides a model for legal rules, while the bottom-up strategy identifies rules instances from legal texts. The bottom-up strategy is processed automatically using machine learning and NLP techniques. Therefore, the approach is composed of two main methodologies: (1) knowledge modeling and (2) knowledge acquisition.

- Knowledge Modeling: definition of a semantic model for legislative texts able to describe legal rules ; Francesconi tends to separate in his approach, oriented to interoperability and reusability, the types of knowledge to be represented by Semantic Web standards. The key-point is to represent independently

[1] http://www.akomantoso.org/

knowledge about the domain and knowledge about reasoning on the domain. In this regard, the knowledge model of the proposed approach is organized into two main components: Domain Independent Legal Knowledge (DILK) and Domain Knowledge (DK).

- Knowledge Acquisition: bottom-up strategy for the instantiation of legal rules, driven by the defined semantic model, through the analysis of legislative texts based on machine learning and NLP techniques.

Wyner and Governatori: Translating Regulatory Rules from Natural Language to Defeasible Logic Wyner and Governatori (Wyner et al., 2013) have compared two different approaches for translating regulatory rules from natural language. The first approach is manual and tends to convert the rules to Defeasible and Deontic Logic. Meanwhile, the aim of the second approach is to apply the C&C/Boxer tool (Bos, 2008) to translate regulatory statements to semantic representations. The domain application of the work is the Australia's Telecommunications Consumer Protections Code.

First, the source data is preprocessed. Furthermore, C&C/Boxer automatically parses the sentences of the Modified Source. Thus, every sentence is parsed and given a semantic representation. Furthermore, the Defeasible Logic is used to represent the defeasible rules and the Deontic Logic to represent the concepts of *obligation*, *prohibition* and *permission*. The proposed approach is based on an assumption that in the legal domain, rules are *non-monotonic* , that is, they admit of exceptions where the rule does not apply or where new information blocks the inference from the rule.

In *Description Logic*, five main key features are identified:

- Facts: indisputable statements.
- Strict rules: material implication in classical logic.
- Defeasible rules: rules from which inferences are drawn, unless the rule is defeated by superior, contrary evidence.
- Defeaters: rules that prevent conclusion of a defeasible rule from holding. They produce contrary evidence.
- Superiority relation among rules: the relation allows to draw a "winning" conclusion from rules with opposition conclusions.

After applying the approaches, the authors conclude that:

- The use of C&C/Boxer highlight some limitations: the output parse and semantic representation given by the tool must be manually checked to accurately correlate to the intended semantic interpretation of the input expression; relatedly, the outputs have not been associated with logical or machine-readable representations that could serve as requirements for the semantic representation.
- On the other hand, studies using Defeasible Logic and LegalRuleML do not systematically relate to natural language or the issues of acquiring the formal representations from the source material that is represented in natural language.

There remains, then, a significant gap between natural language source material and formal, machine-processable representations.

4.4 Ontology-based Approach for Modeling Legal Norms

After presenting the existent approaches in the literature that deal with the modeling of legal norms, we found that most of them neglect the role of ontologies. Meanwhile, the use of ontologies can enhance and simplify the modeling process.

It is commonly known that the legal domain is dominated by the use of natural language. Generally, legal norms are expressed in natural language textual sources, such as legislations and codes, which make them ambiguous since an expression can have multiple meanings. This problem causes some difficulties in interpreting them (Van Gog et al., 2001). The general rule is that any document of this domain is always embedded in a context of norms. Thus, understanding concepts of a legal norm is important to understanding other legal norms (Machado et al., 2014). Based on this perspective, we relied on the role of ontologies as a ground for modeling legal norms. The aim of building legal ontologies is to describe the facts of legal cases at a comfortable level of abstraction and the "law" of the domain application which consists of "norms" and "concepts" (McCarty, 1983).

In this context, we have relied on three main criteria, inspired by (Biagioli, 1991), (Boer et al., 2005) and (Gordon et al., 2009), for proposing an ontology-based approach for modeling legal norms:

- The modeling of legal norms is considered as developing a formalism which is not too different from the way in which the legislator expresses himself. The aim is to create models that should be defined from within the legal world and not imposed on it from outside.

- The separation of knowledge about reasoning - epistemology - and knowledge about the problem domain - domain ontology - for the reusability of the knowledge representation.
- The faithful representation of legal rules is obviously crucial for representing legislative documents, regulations, and other sources of law.

Based on these criteria, a direction that separates the modeling of the legal content of the norms from their procedural aspects where logical inferences is tracked. Therefore, a bi-phased ontology-based approach is proposed for modeling legal norms (see Figure 4.1):

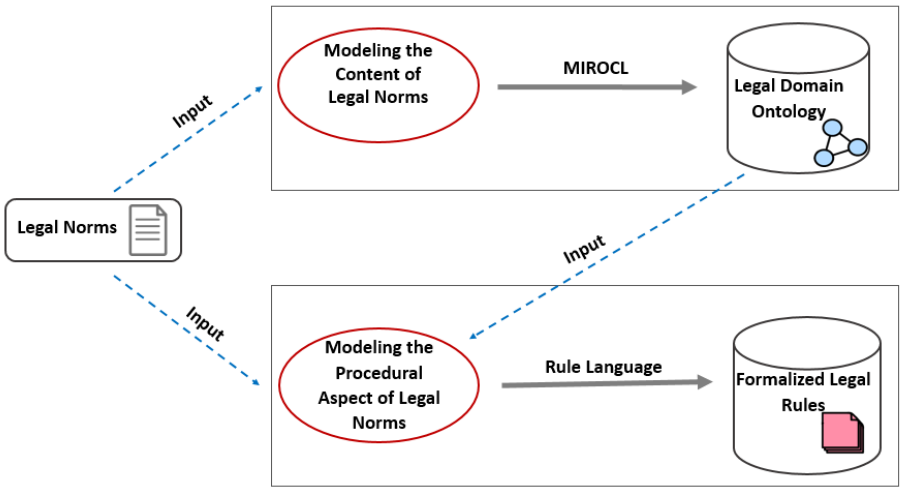


FIGURE 4.1: A bi-phased ontology-based approach for modeling legal norms.

Phase1: Modeling the content of the legal norms The aim of the first phase is to model the content of the legal norms which reflects the given domain. Thus, we obtain an ontological model, which is the intended legal domain ontology, aiming to provide a well-founded representation of concepts and semantic relationships among them in the context of the norms of the given domain. In this regard, the MIROCL approach, introduced in chapter 3 is applied for several reasons detailed in section 4.5.

Phase2: Modeling the procedural aspect of the legal norms The second phase of the proposed approach tends to model the procedural aspect of the norms using logical formalisms in order to translate the norms into a list of formalized rules. In this phase, it is preliminary to follow the isomorphism principle stated by (Bench-Capon et al., 1992; Bench-Capon et al., 2009) for connecting legal textual provisions with formalized rules. At the end, we obtain a list of logic rules formalized using a

rule language based on the legal domain ontology. This phase will be discussed in chapter 5.

4.5 Phase1: Advantages of Using MIROCL for Modeling the Content of Legal Norms

As aforementioned, the aim of the first phase of the ontology-based approach for modeling legal norms is to model their content. Therefore, we will obtain a legal domain ontology that reflects the model of the given legal norms. The target is to build a well-founded legal domain ontology. Thus, the modular middle-out approach MIROCL, proposed in chapter 3, is suggested for the following reasons:

- The most promising way to build legal ontologies is through the integration of top-down and bottom-up approaches. Such an integrated approach leads to accurate ontology construction, which cannot be achieved by either bottom-up or top-down approach alone. This is particularly true in the legal domain, where ontology construction should follow insights provided by legal theory but at the same time should guarantee textual grounding (Francesconi et al., 2010).

  Therefore, from one side linguistic resources are indeed important resources for identifying concepts and can be used as consensus references to root ontology. From the opposite point, lexical resources and semantic resources, such as ontologies, need each other and are complementary one to each other as ontologies allow to represent the complex relationship between the lexical and the semantic meaning of a term (Biasiotti et al., 2011).

- Starting the modeling activity from theoretical assumptions and semantic resources already developed by the scientific community in the field of legal ontologies can lead to a well-founded ontological model (Cherubini et al., 2008).

- Additionally, the literature suggests that legal ontologies may be distinguished by the levels of abstraction of the ideas they represent (Breuker et al., 2009), with the key distinction being between core and domain levels. The core level ontology is a model of general concepts common for all legal domains (Valente, 1995). Accordingly, research in this field should not concentrate on creating a single ontology of the legal domain but on the creation of a library that contains several dedicated ontologies at different abstraction levels and supports their mapping to create a composite ontology (Piovesan et al., 2014).

In fact, the approach MIROCL satisfies the three aforementioned points which are the middle-out aspect, the reuse of existent semantic resources in order to build

a well-founded domain ontology and the modularization or composition of the ontology.

4.6 Phase1: The Building Process of CriMOnto

CriMOnto is a well-founded legal domain ontology for modeling the content of legal norms. It is intended to be used for supporting rule-based reasoning purposes. Therefore, a conceptualization process is needed for modeling the domain in the context of the legal norms provided as input texts. In this regard, an issue must be recognized is that there is no agreement on the basic conceptualization aspects of the legal domain in general. Thus, the same domain can be conceptualized in different ways. What is needed is rather for the ontologies to be sufficiently clearly stated (Bench-Capon et al., 1997).

For the reasons mentioned in section 4.5, MIROCL will be used for building CriMOnto. Therefore, the main tasks of MIROCL are executed: (1) identification of data sources; (2) building of ontology modules by using two different strategies: bottom-up for ontology learning and top-down for ontology reuse and ontology-driven conceptual modeling; (3) integration of ontology modules; (4) ontology evaluation.

4.6.1 Identification of Data sources in CriMOnto

In MIROCL, two main data sources are identified: (1) the textual resources that are related to the domain application and (2) existent validated ontologies (foundational and core ontologies).

4.6.1.1 Textual Resources in CriMOnto

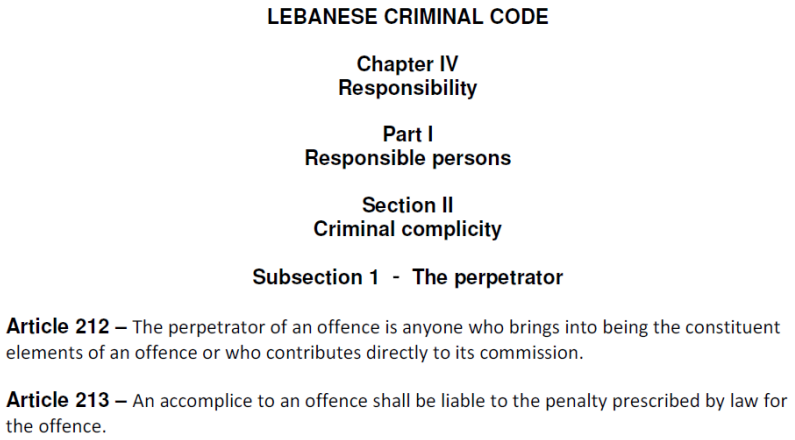
In the legal domain, the textual resources are considered as the legal discourse. For Tiscornia (Tiscornia, 2005), different levels of legal language exist, such as:

- The discourse of the legislator (laws and regulations);
- The discourse of the judges (judgments and other judicial decisions);
- The discourse of the doctrine (studies on several legal sub-domains, systematizing legislator and judges' discourses);
- The discourse of legal theory (legal works having a general content, not addressing a particular legal system).

The legal discourse in this thesis is limited to the discourse of legislator. For CriMOnto, the domain application is the Lebanese criminal system. The available

textual resources used as input are the legal norms of the Lebanese criminal code which is considered as relevant source of the discourse to be considered. This corresponds to explicit legal knowledge, codified in specific and standardized ways by the legal community (laws and articles) (Fernandez-Barrera et al., 2011).

The Lebanese criminal code (**??**, see Figure 4.2) contains the general criminal laws of Lebanon. First enacted in 1943 and it remains in effect till today. It is translated to French and English versions. Concerning the structure of the code, it is divided into two main books, *General Provisions* and *Offences*, composed of 770 articles.



LEBANESE CRIMINAL CODE

Chapter IV
Responsibility

Part I
Responsible persons

Section II
Criminal complicity

Subsection 1 - The perpetrator

Article 212 – The perpetrator of an offence is anyone who brings into being the constituent elements of an offence or who contributes directly to its commission.

Article 213 – An accomplice to an offence shall be liable to the penalty prescribed by law for the offence.

FIGURE 4.2: Excerpt of the Lebanese criminal code.

In the Lebanese criminal code, the legal norms are represented in unstructured natural language texts. Two main types of norms are identified: determinative, known also as terminological, and regulative, known also as normative.

Determinative Rules The determinative rules, as defined in section 4.2, consist in definitions of some of the concepts of the criminal domain that are used to describe the criminal facts. For instance, the concept *perpetrator* is defined in article 212 as:

"*Article 212: The perpetrator of an offence is anyone who brings into being the constituent elements of an offence or who contributes directly to its commission*".

Another example, the article 240 where the legislator clarify some terms used in the code such as *child*, *adolescent* and *minor*.

"*Article:240: For the purposes of this Code, a child means anyone from the age of 7 to the age of 12.*

An adolescent means anyone from the age of 12 to the age of 15.

A minor means anyone from the age of 15 to the age of 18 ".

Normative Rules The normative rules, as defined in section 4.2, connect the legal consequences to descriptions of certain facts and situations, such as in articles 217 and 398:

"*Article 217: Anyone who induces or seeks to induce another person to commit an offence shall be deemed to be an instigator*".

"*Article 398: Any Lebanese citizen who knew of a felony against state security and failed to report the matter at once to the public authorities shall be punishable by imprisonment for a term of between one and three years and by loss of his civil rights*".

4.6.1.2 Existent Validated Ontologies in CriMOnto

Existent validated ontologies are the second type of data sources. They are selected not only for simple reuse but for grounding the target ontology as well. Actually, two main types of ontologies are recognized: foundational and core.

In MIROCL, the Unified Foundational Ontology UFO (Guizzardi et al., 2005b) is identified as a foundation for the ontology grounding process (see chapter 3, section 2.2.2.1). Meanwhile, the core ontology is related to the domain application. For instance, in case of CriMOnto, since the domain application is law, then the core ontology should be legal core ontology. For this purpose, a selection process have to be applied in order to choose the convenient ontology. In the legal domain, three main recent legal core ontologies are identified: LKIF-Core (Hoekstra et al., 2007), OPJK (Casellas, 2008a) and UFO-L (Griffo et al., 2015).

UFO-L is still under development. Concerning OPJK, actually we consider this ontology as domain ontology because it is intended for a specific application which is the JURISERVICE web-based application. Additionally, the modelers have used existent upper and core ontologies for building OPJK. LKIF-Core is defined by the authors as legal core ontology, and directed at supporting legal inference, knowledge acquisition and knowledge exchange. Beside this, LKIF-Core is composed of modules which simplifies the ontology reusing. Therefore, LKIF-Core is the most convenient as a legal core ontology (see chapter 2, section 2.2).

4.6.2 Building of Ontology Modules in CriMOnto

As mentioned previously, the building process of the ontology modules, by following the MIROCL approach, is a combination of two main strategies: top-down and bottom-up. These strategies tends to modularize the ontology into four different modules which are themselves ontologies: *upper* (UOM), *core* (COM), *domain* (DOM) and *domain-specific* (DSOM). The *upper* and *core* modules are developed by applying the top-down strategy that represents an ontology-driven conceptual

modeling process performed by reusing existent foundational and legal core ontologies. Meanwhile, the bottom-up strategy starts from textual analysis and conceptual representation of the intended meaning of the legal norms as an ontology learning process for building the *domain* and *domain-specific* modules.

Before detailing the four ontology modules, the identified strategies are discussed in the following.

4.6.2.1 Top-down: ODCM and Reuse

Concerning the top-down strategy, an ontology-driven conceptual modeling process (ODCM) guided by reusing foundational and core ontologies is established for grounding the legal domain ontology. For this purpose, two main existent validated ontologies are reused: The unified foundational ontology UFO and the legal core ontology LKIF-Core in order to build two top modules: *upper* (UOM) and *core* (COM). In CriMOnto, the ontology-driven modeling is concerned with capturing the relevant entities of the given domain in the upper and core ontology modules using the ontology specification language OntoUML which is based on a set of basic, domain-independent ontological categories of UFO.

4.6.2.2 Bottom-up: Ontology Learning Process

"Since legal domain is strictly dependent on its own textual nature, a methodology for ontology construction must privilege a bottom-up approach, based on a solid theoretical model" (Biasiotti et al., 2011). Thus, considering the complexity of the law, the most promising way to fill the gap between conceptual models and lexical patterns extracted from texts is the use of methodologies based on ontology learning techniques (Biasiotti et al., 2011). In MIROCL, the ontology learning process is defined in five main phases (see Figure 3.12). We discuss in the following each phase.

1. Material selection: As aforementioned, the domain application for building CriMOnto is the criminal domain, and the available resources are the legal norms of the Lebanese criminal code. Thus the material is the Lebanese criminal code which is unstructured textual document. The available language is the English.

2. Tool selection: After exploring the literature and collecting the state-of-the-art for the most frequently used ontology learning tools (see chapter 2), we met some access difficulties in our experimentations. In fact, three of the tools were publicly available on the Internet to download and install: *Terminae*, *OntoGen* and *Text2Onto*. We discuss briefly the usability of each tool.

Concerning the input type, all the tools accept simple text files (.txt), *Text2Onto* and *Terminae* accept PDF files (.pdf) as well. For *OntoGen*, there are additional input file types that need to be pre-processed, such as Named Line-Document and Bag of Words. *Terminae* and *OntoGen* need preprocessing efforts.

Starting with *Terminae* where the linguistic tool extract terms automatically from the corpus based on their occurrences. Meanwhile, the rest of the steps are processed manually which is too resource demanding and too time consuming. For this reason, this tool is discarded. Furthermore, we face some difficulties while using *OntoGen*. We could not control the system that generates sequences of terms that are not well related. In addition to this, the suggestion of concepts is limited to single-word terms, proposed only from the input documents (no external resources), and the relations extraction is limited as well to taxonomic. Meanwhile, OntoGen provide a visualization and exploration of concepts only and not of the whole ontology. OntoGen is discarded too. We finished our experiments by *Text2Onto*. According to (Gherasim, 2013), *Text2Onto* is an ontology learning tool that covers the entire process of extracting OWL ontologies. Furthermore, it provides a long list of proposed concepts and relationships along with their weights in a tabular form. Meanwhile, *Text2Onto* does not have any mechanism to filter the concepts irrelevant to goal (Hatala, 2012). The user input is limited to removing concepts and relationships extracted from the supplied course. In *Text2Onto*, the visualization of the structure of the resulted ontology is missing. Regarding the external resources, *Text2Onto* uses *WordNet* to improve and enrich the algorithms of pattern-based relation extraction. However, some authors found that *WordNet* lacks the richness of named relations (Fouad, 2015). For this reason, they decided to use online ontologies as an alternative to *WordNet*.

Regarding the limitations of *Text2Onto*, this tool still answers the main requirements of our work: automatic extraction, usability, scalability and reusability. Based on this selection, we proposed to apply a re-engineering phase that consists of evaluating the ontology extracted using *Text2Onto*, correcting the detected errors, refine the ontology model and finish by enrich the semantic relations and axioms.

Tool	Terminae	OntoGen	Text2Onto
User Input	Add, Remove, Modify	Add, Remove, Modify	Remove
Visualization	NA	Concepts	NA
External Resources	NA	NA	WordNet

TABLE 4.1: List of experimented tools.

Therefore, the following phases, pre-processing and extraction of ontology elements, are performed using *Text2Onto* in order to extract the domain ontology module from texts (see Figure 4.3).

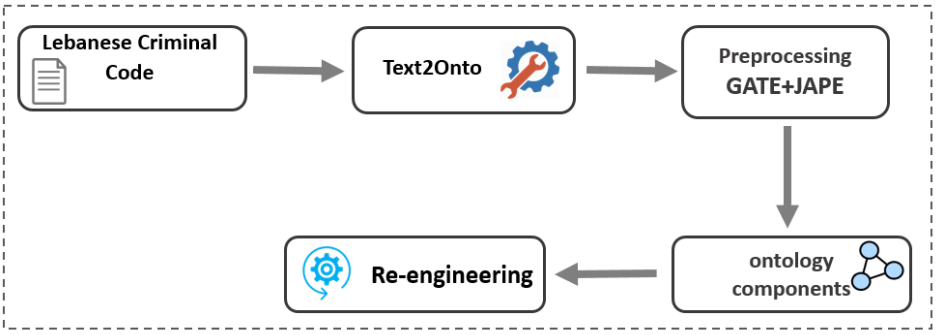


FIGURE 4.3: Ontology learning process using Text2Onto.

3. Preprocessing: The aim of this phase is to extract meaningful textual content from the input document after removing the ambiguity by filtering out worthless symbols and words. The tasks performed in this phase are: extraction of plain text, splitting text into sentences, elimination of stop words, tagging and parsing of the sentences.

   In *Text2Onto*, there is a combination of machine learning approaches with basic linguistic processing such as tokenization or lemmatizing and shallow parsing (Cimiano et al., 2005b). In addition to this, *Text2Onto* benefits from *GATE* by the integration of *JAPE* that provides finite state transduction over annotations based on regular expressions.

   Therefore, two main steps composed the preprocessing phase (see Figure 4.4):

   (a) Basic linguistic preprocessing using GATE applications; it starts by tokenization and sentence splitting, then applying the POS tagging on the resulting annotation set, and finish by lemmatizing using a morphological analyzer and a stemmer respectively.

   (b) The second step runs a JAPE transducer over the annotated corpus in order to match a set of particular patterns required by the ontology learning algorithms. JAPE patterns are developed for shallow parsing and identification of modeling primitives such as concepts, instances and relations.

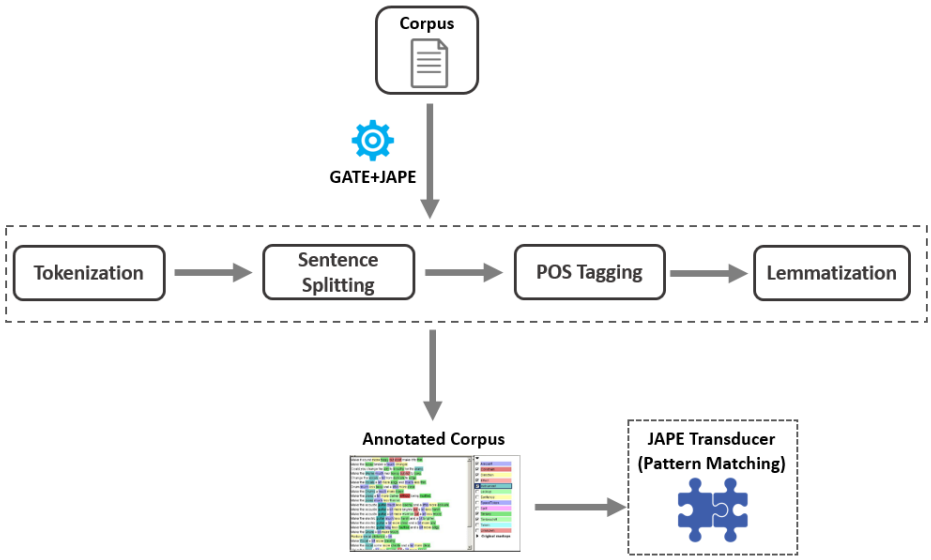


FIGURE 4.4: Pre-processing phase in Text2Onto.

4. Extraction of the main components of the ontology: This phase is performed with the support of the selected tool Text2Onto by implementing list of algorithms and techniques. In *Text2Onto*, the main extracted components or modeling primitives of the ontology are concepts, taxonomies, relations, disjoint axioms and instances. For the concepts, `RTF` algorithm is applied. The taxonomies are learned using three algorithms combined together: vertical relations, WordNet and patterns. For the relations, *Text2Onto* uses syntactic pattern matching technique to extract general relations. Finally, the extraction of the disjoint axioms is based on lexico-syntactic patterns.

   - Concepts: In our experiments, in order to extract the concepts from the corpus, we have applied the `RTF` algorithm. This algorithm calculates the relative frequency of the terms as follows:

$$rtf(t,D) = \frac{absolutetermfrequency}{maximumabsolutetermfrequency}$$

     By applying this algorithm, 486 single and multi-word concepts are extracted (see Table 4.2).

Single word	Multi-word
Probation	Constituent element
Criminal	Deportation measure
Prosecution	Drug addict
Location	Instance judgment
Selfishness	Term penalty

TABLE 4.2: Excerpt of the concepts extracted using Text2Onto.

However, some verbs are extracted as concepts such as `stay`, `incur` and `abort`. Moreover, some meaningless concepts are extracted such as `hand`, `harm`, `interior`, `lack`, `loss`, etc...

- Taxonomies: or concept inheritance (subclass-of relations), `Text2Onto` provides three main algorithms to classify concepts based on Vertical Relations, WordNet, and Patterns. Based on our experiments, we decided to combine them for achieving better results (see Table 4.3).

Domain	Range
Treatment	Manner
Offender	Person
Death penalty	Penalty
Corrective measure	Measure

TABLE 4.3: Excerpt of the taxonomies extracted Text2Onto.

- Relations: *Text2Onto* relies on an algorithm that uses syntactic pattern matching technique to extract general relations by identifying the following syntactical frames:

  Transitive: *Subject+verb+object*.

  Intransitive+prepostition phrase-complement: *Subject+verb+preposition+object*.

  Transitive+prepostition phrase-complement: *Subject+verb+object+preposition+ prepositional+object*.

  The number of extracted relations is limited to 20. In table 4.4, an excerpt of the extracted relations is depicted.

Label	Domain	Range
involve	Residence	Placement
require	Activity	License
exceed	Offence	Bound
preclude	prescription	Enforcement
commit	Society	Offence

TABLE 4.4: Excerpt of the relations extracted Text2Onto.

- Disjoint Axioms: Based on lexico-syntactic patterns, *Text2Onto* extracts 86 disjoint axioms from the corpus (see Table 4.5).

Domain	Range
Measure	Penalty
Felony	Disposal
Person	Association
Substitute	Penalty
Summary	Judgment

TABLE 4.5: Excerpt of the disjoint axioms extracted Text2Onto.

Some wrongful disjoint axioms are extracted such as `disjoint(Penalty, Fine)` and `disjoint(Person, Perpetrator)`.

- Instances: The extracted instances are limited to name of days or months, countries and languages such as `Friday`, `November`, `Lebanon` and `Arabic`. Actually this is due to the quality of the input resources written in legal language, which is authoritative and contains legal speech acts accompanied by rituals of various types. For this reason, the tool will not recognize easily the instances of the relevant domain without the help of the legal expert.

After applying Text2Onto to extract the ontology components from the Lebanese criminal code, we conclude that the generated ontology is limited in the expressiveness and mainly consists of concepts organized in hierarchies. Meanwhile, an expressive ontology, rich in axioms and instances, is required mainly for reasoning purposes. Therefore, there is a need for correcting, pruning and enriching the results by applying an ontology re-engineering process with the intervention of legal experts in order to achieve better results more correct and expressive.

5. Ontology Re-engineering: In this phase, the legal experts correct, prune and enrich the obtained results of Text2Onto. As aforementioned, the main goal is to obtain a more expressive domain ontology rich in axioms and instances.

   Generally, the knowledge re-engineering is defined as the correction and continuous reuse of preexisting knowledge for a new task (Wyner et al., 2010). For ontological purposes, the re-engineering concept is defined by Gomez-Perez (Gomez-Perez et al., 1999) as: "*the process of retrieving and transforming a conceptual model of an existing and implemented ontology into a new more correct and more complete conceptual model which is reimplemented* ". Accordingly, the ontological re-engineering process is composed of three main activities: reverse engineering, restructuring and forward engineering (see Figure 4.5).

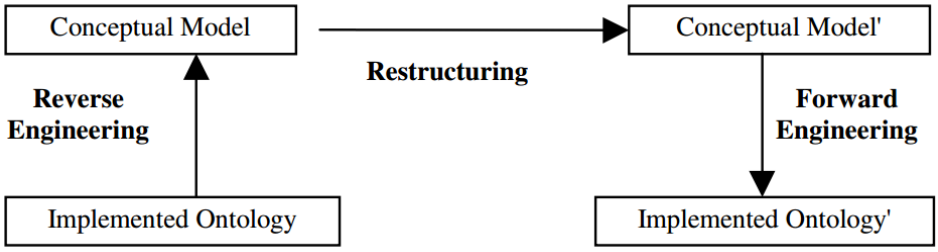


FIGURE 4.5: The re-engineering process (Gomez-Perez et al., 1999).

After applying the re-engineering process under the supervision of ontology engineers and legal experts, the domain and domain-specific modules are presented in the following paragraphs.

4.6.3 The modules of CriMOnto

4.6.3.1 Upper Ontology Module (UOM)

As a result, UOM consists of 74 concepts (classes in protégé), 54 relations (object properties in protégé) and 144 hierarchies (subClassOf axioms in protégé). UOM is composed of abstract concepts and relations which are effectively independent of any specific domain such as *Agent*, *Action*, *Event*, etc.. For a well-founded building of this module, a partial reuse of the unified foundational ontology UFO is applied. More specifically, the reuse process covers the UFO-C and UFO-B layers.

Concerning the conceptual modeling process of the upper module, the ontology modeling language OntoUML is used for representing the upper concepts reused from UFO in order to compose the upper ontology module (see figure 4.6 adapted from (Guizzardi, 2005)).

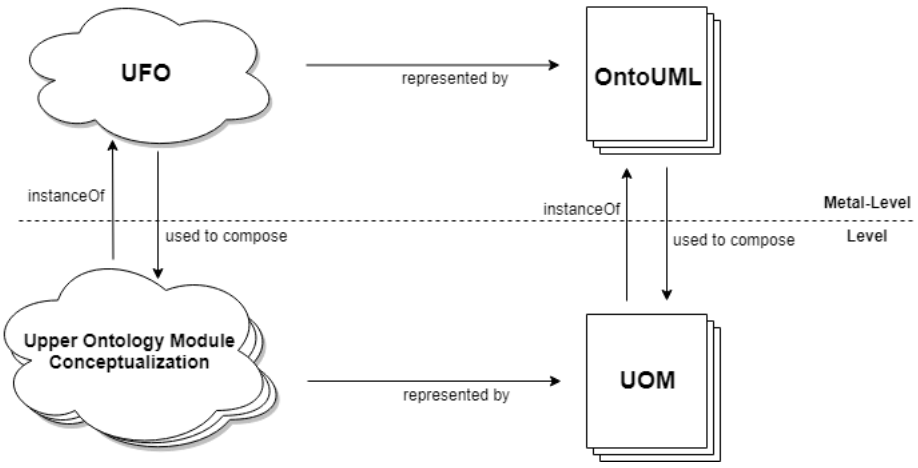


FIGURE 4.6: Conceptual modeling process of the Upper Ontology Module (UOM).

Below are detailed the following fragments:

- `Substance` in figures 4.7 and 4.8.
- `Event` in figures 4.9 and 4.10.
- `Agent` in figures 4.12 and 4.13.
- `Intention` in figures 4.16 and 4.17.

substance `Substance` category is considered as an *endurant* where `Agents` and `Objects` are disjoint substances. `Agent` can be physical (`Physical_Agent`) or social `Social_Agent`. `Natural_Person` is an example of `Physical_Agent`. `Organization` is an example of `Social_Agent`.

As `Agent`, `Object` can be physical (`Physical_Object`) or social (`Social_Object`). `Normative_Description` is considered as a `Social_Object` and is recognized by at least one `Social_Agent`. Each `Physical_Object` has a *role* (`Resource`).

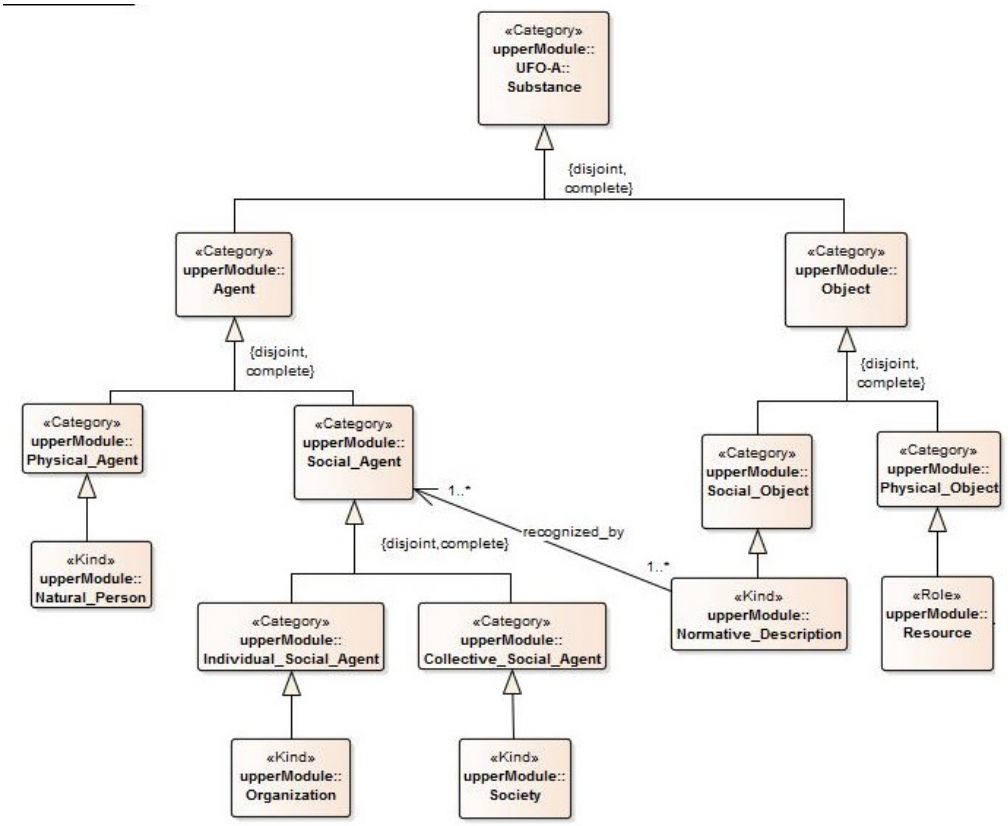


FIGURE 4.7: The concept `Substance` represented using OntoUML.

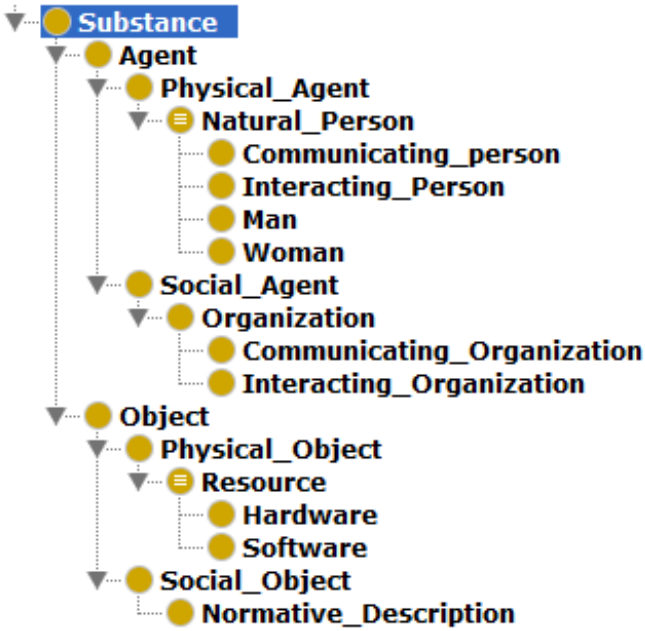


FIGURE 4.8: The concept `Substance` represented in Protégé.

Event The concept `Event` is a basic concept in the legal domain and specifically in the criminal domain. `Event` can be later instantiated into `Legal_Event` in the core module, and `Criminal_Event` in the domain module. In UFO, events can be atomic (`Atomic_event`) or complex (`Complex_Event`). While atomic events have no proper parts, complex events are aggregations of at least two disjoint events (Guizzardi et al., 2013b). Example of an event is the murder of a person which can be decomposed into sub-events such as *attack* and *death*.

Events existentially depend on the objects that participate in them (Guizzardi et al., 2013b). `Participation` is an event, and it represents the participation of one object into an event and it depends exclusively on one object.

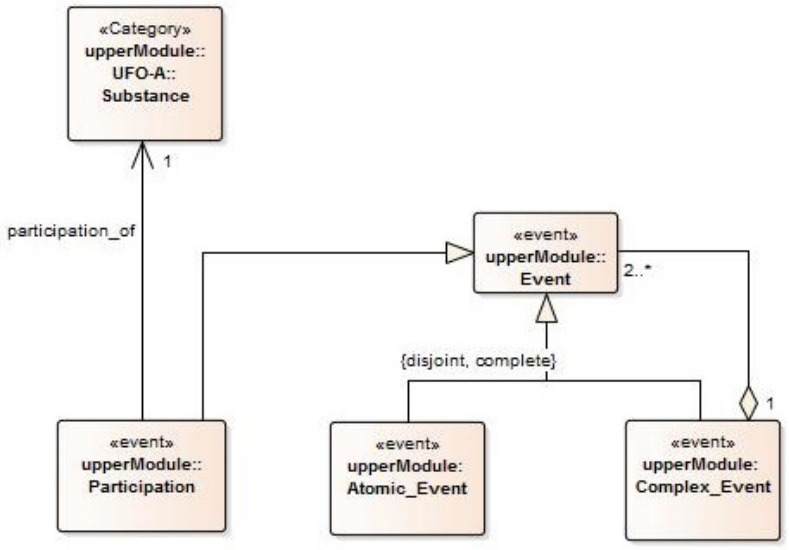


FIGURE 4.9: The concept `Event` represented using OntoUML.

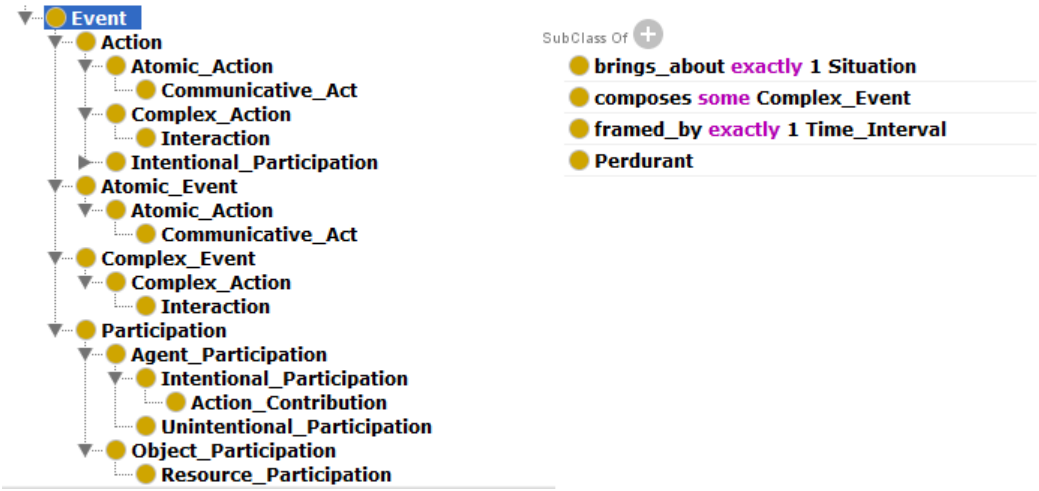


FIGURE 4.10: The concept `Event` represented in Protégé.

An event occurs in a certain situation at a certain point in time, and transforms it to another situation (Guizzardi et al., 2013b). The figure 4.11 depicts the `Situation` concept and its relationship to `Event`. Actually, `Situation` is considered as a basic concept in the ontological account of events in UFO-B.

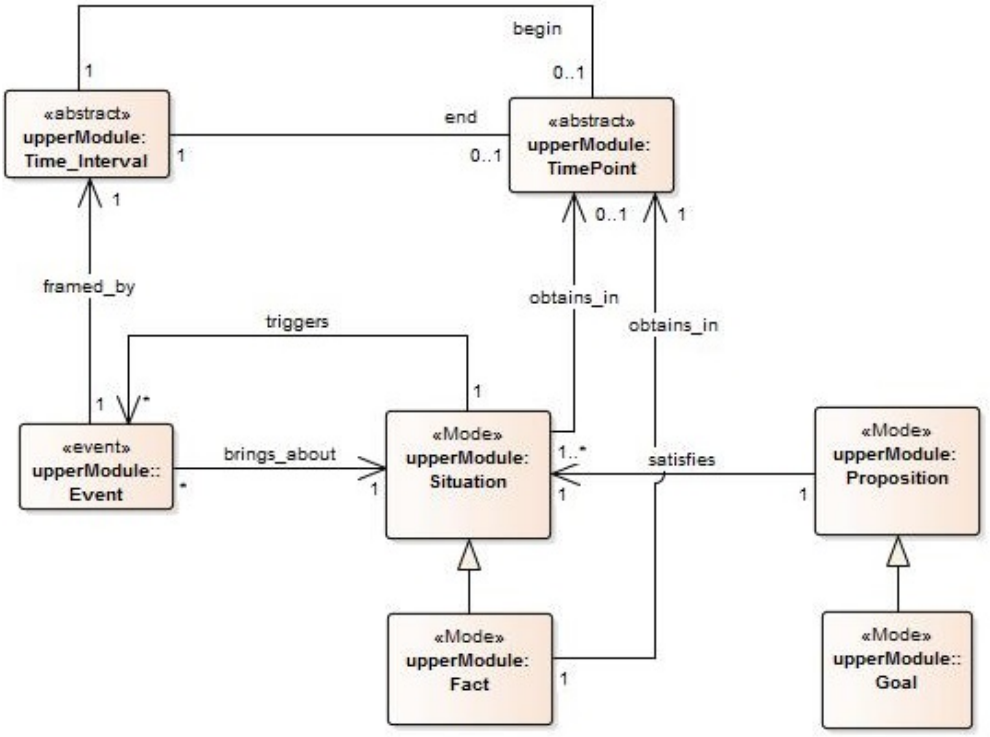


FIGURE 4.11: The concept `Situation` represented using OntoUML.

Agent Only agents are able to create actions. Actions, as events, can be atomic (`Atomic_Action`) or complex (`Complex_Action`).

Each `Agent` has a role. It can be `Communicating_Agent` or `Interacting_Agent`. A `Communicating_Agent` is the sender and receiver of a `Communicating_Act` which is considered as an `Atomic_Action` (see Figure 4.14). Meanwhile, the `Interaction` is a `Complex_Action` composed of minimum 2 `Action_Contribution` (see Figure 4.15).

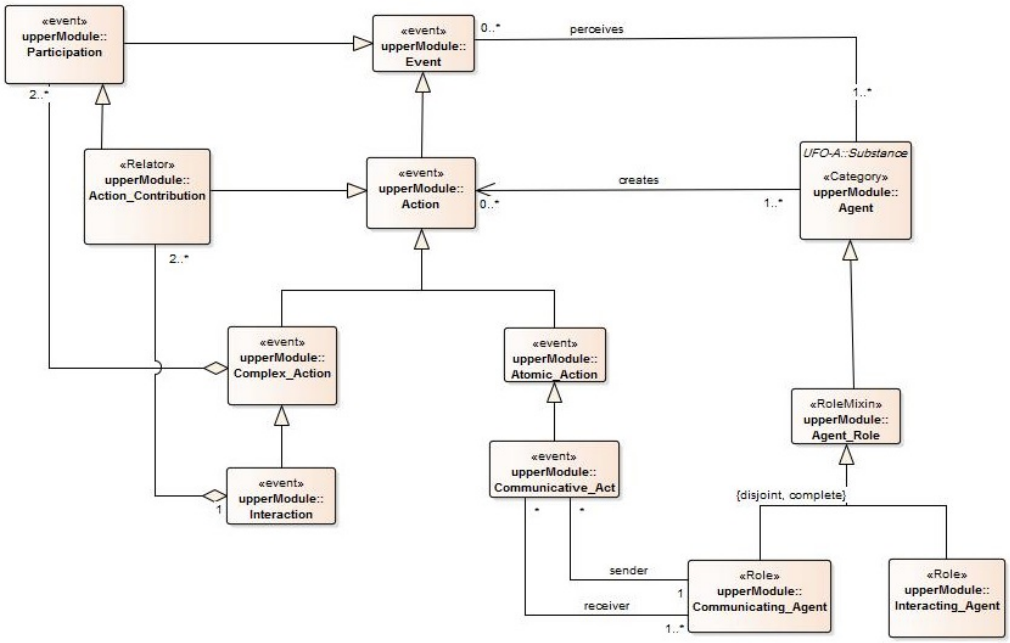


FIGURE 4.12: The concept `Agent` represented using OntoUML.

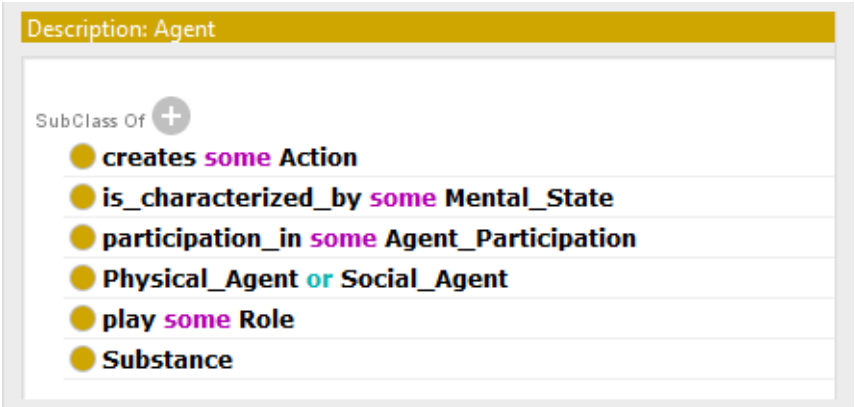


FIGURE 4.13: The concept `Agent` represented in Protégé.

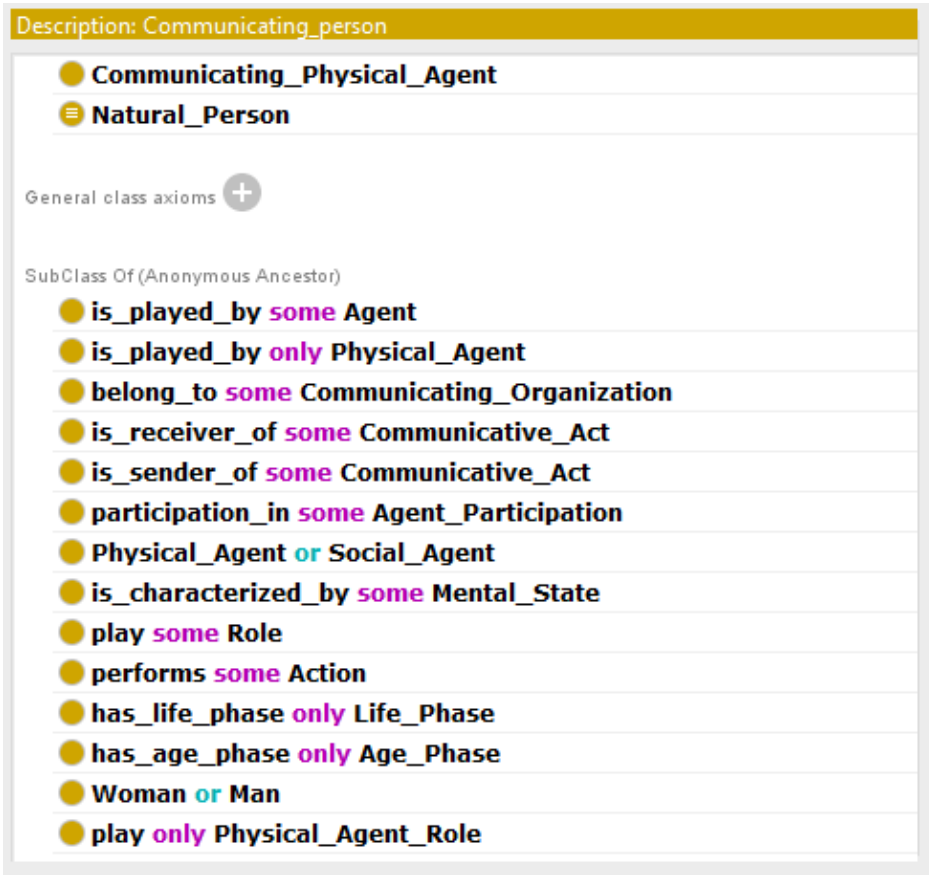


FIGURE 4.14: The concept `Communicating_Agent` represented in Protégé.

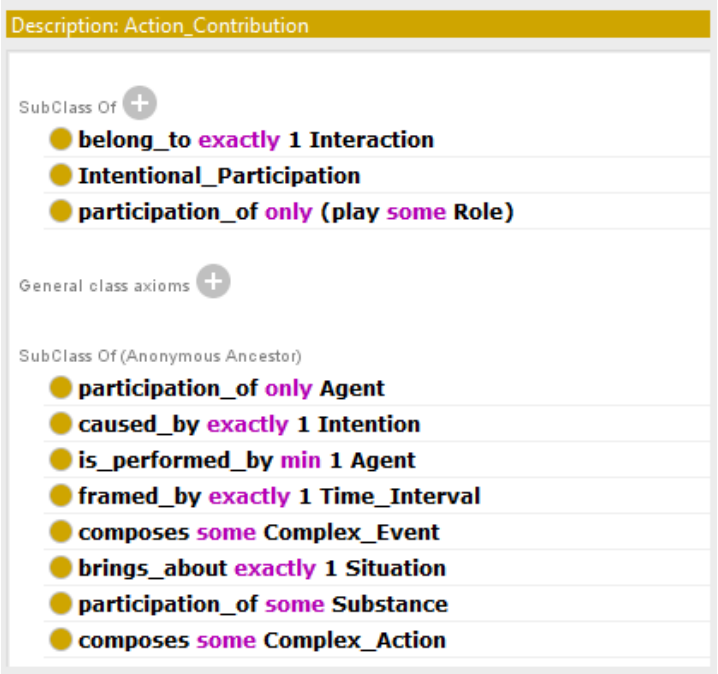


FIGURE 4.15: The concept `Action_Contribution` represented in Protégé.

Intention In UFO, agents can bear special kinds of moments named Intentional_Moment. Every `Intentional_Moment` has a type (e.g., Belief, Desire, Intention) and a `Propositional_Content`. *Intending something* is a specific type of intentionality termed `Intention`. The propositional content of an Intention is a `Goal`. Intentions cause the agent to perform `Actions`.

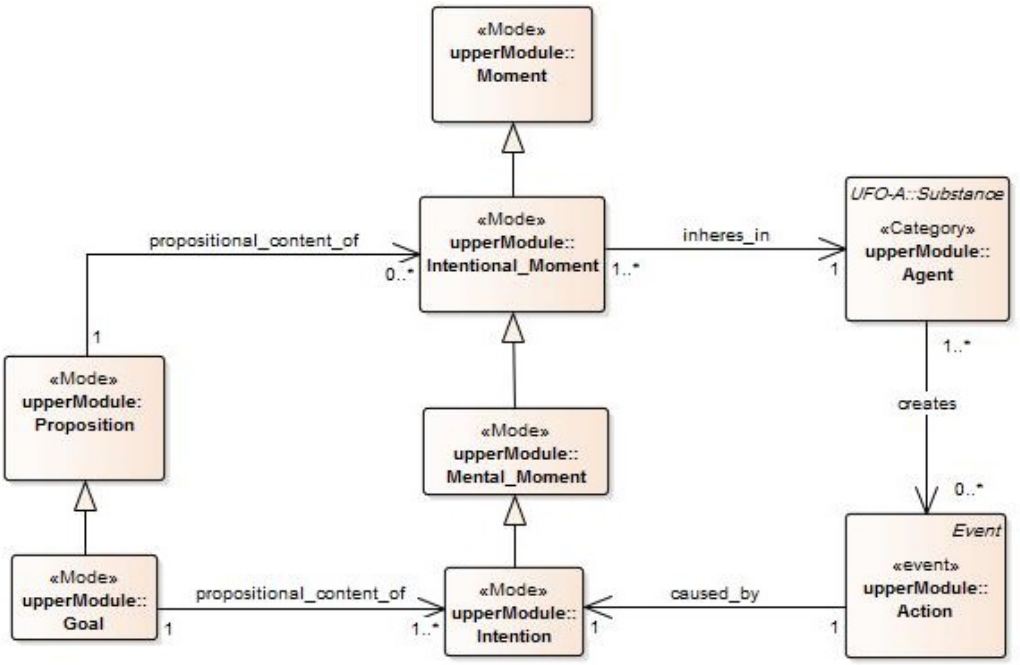


FIGURE 4.16: The concept `Intention` in OntoUML.

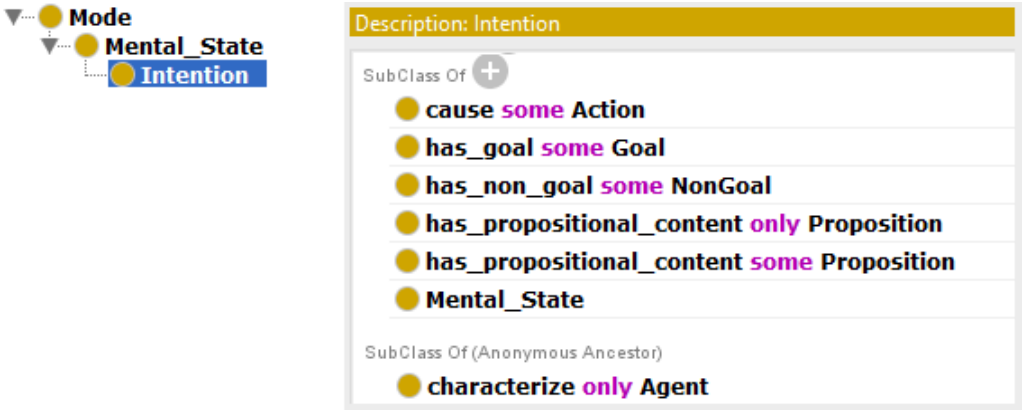


FIGURE 4.17: The concept `Intention` in Protégé.

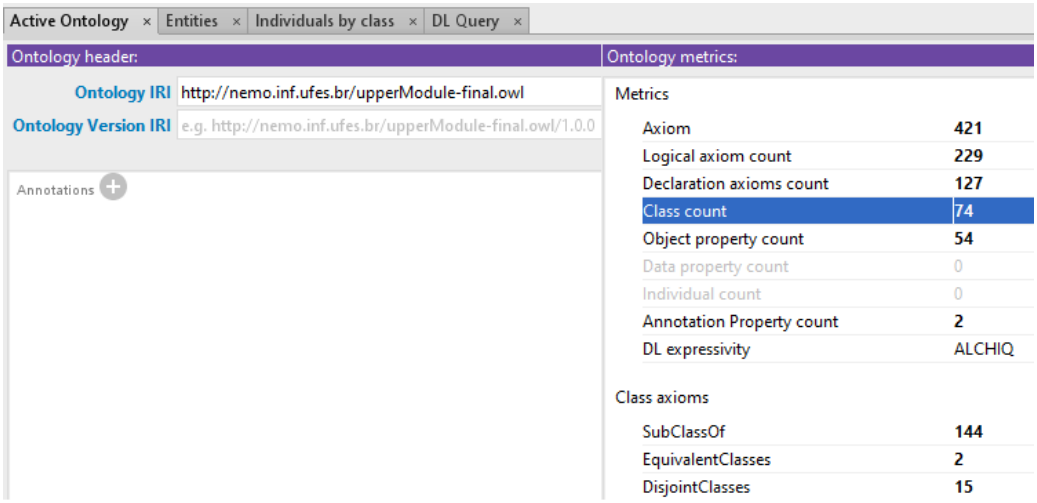


FIGURE 4.18: Metrics of the Upper ontology module.

4.6.3.2 Core Ontology Module (COM)

COM consists of 61 classes, 30 object properties and 100 subClassOf axioms. The core ontology module provides a definition of structural knowledge in the legal domain which constitutes the basis for specialization into domain and domain-specific knowledge. For instance, concepts, such as `Legal_Source`, `Legal_Act` and `Legal_Document`, are common for all the legal fields (criminal, civil, etc.). The core module is built by reusing the concepts and relations of an existent validated legal core ontology which is LKIF-Core. The core concepts are represented in the unified foundational ontology UFO, using OntoUML as an ontology modeling language, in order to compose the conceptualization of the core module which is represented by the core ontology module (see Figure 4.19).

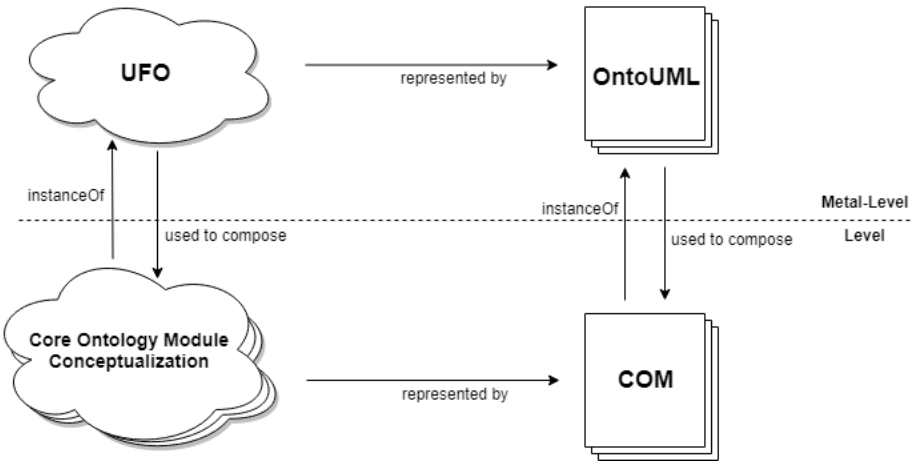


FIGURE 4.19: Conceptual modeling process of the core module.

In the following, we detail some excerpts of the core modules (COM) which are:

- `Medium` in figure 4.20.
- `Legal_Source` in 4.21.
- `Code` in figures 4.22 and 4.23.
- `Expression` in figure 4.24.
- `Process` in figure 4.25.
- `Legal_Event` in figure 4.26.
- `Legal_Role` in figure 4.27.
- `Professional_Legal_Role` in figure 4.28.

Medium A medium is a bearer of expressions, i.e. externalized propositions. Propositions become expressions once they are externalised through some medium. A `Medium` can be `Legal_Document` or `Legal_Source`. A legal document is a document bearing norms or normative statements.

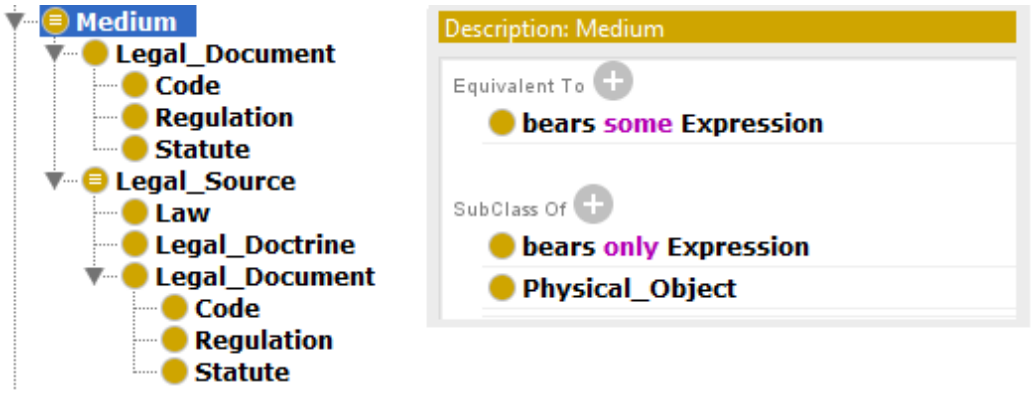


FIGURE 4.20: The concept `Medium`.

Legal_Source A legal source is a source for legal statements, both norms and legal expressions. In a sense it is literally a 'source' of law.

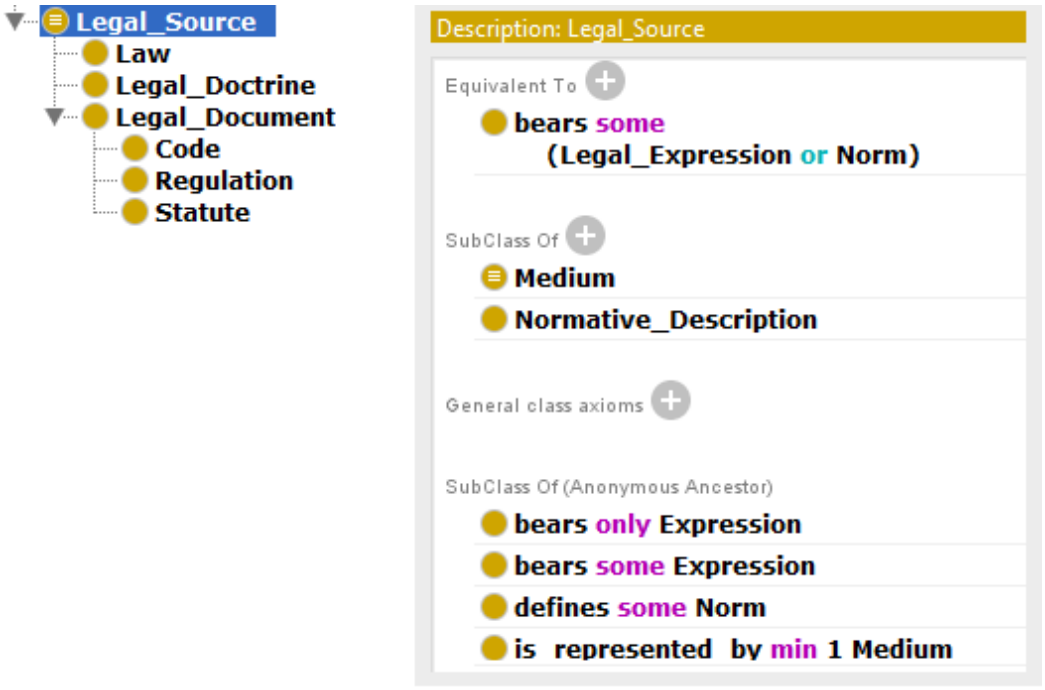


FIGURE 4.21: The concept `Legal_Source`.

Code A legal code bears one or more norms, all of which are uttered by some legislative body. It cannot bear expressions which are not uttered by a legislative body.

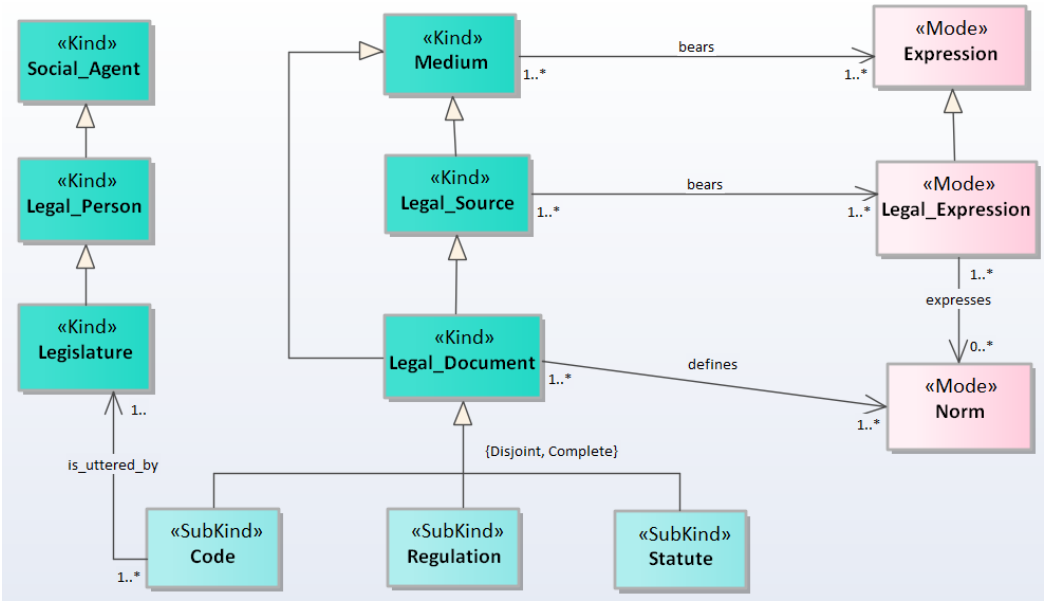


FIGURE 4.22: Representing he concept `Code` in OntoUML.

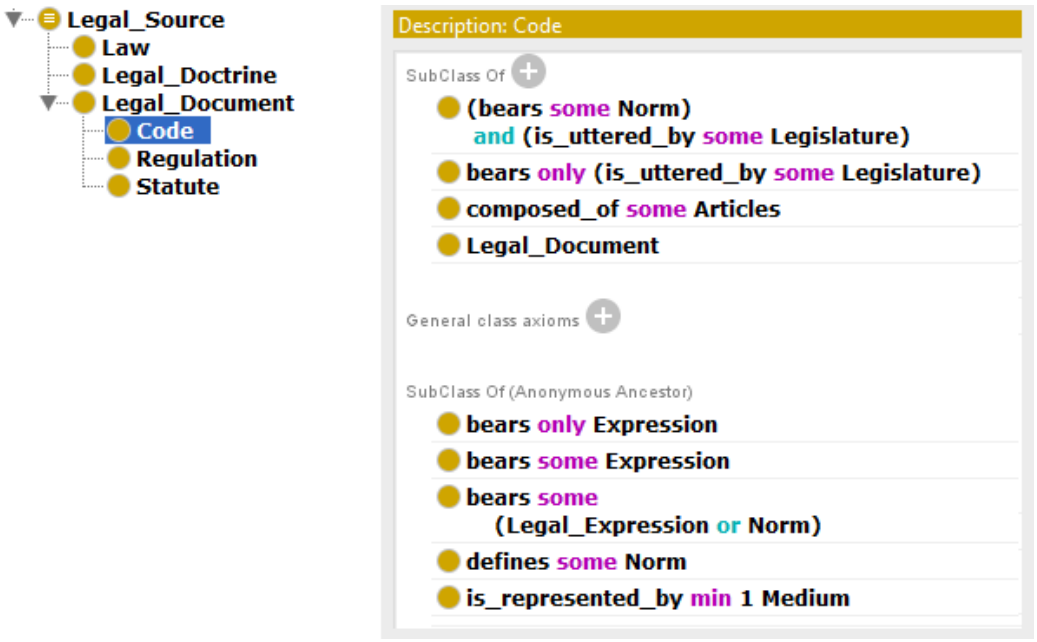


FIGURE 4.23: The concept `Code` in Protégé.

Expression An expression is a proposition beared by some medium, e.g. a document, and is stated by some communicated attitude.

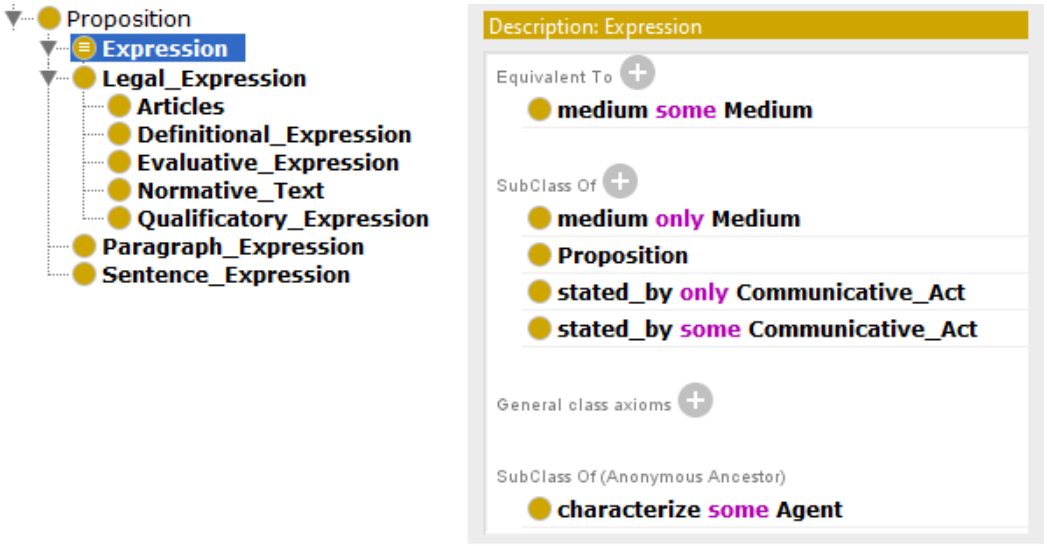


FIGURE 4.24: The concept `Expression`.

Process A process is a 'causal' change: any change which can be explained through some known or understood causal structure. Every process has some `Time_Period` as duration.

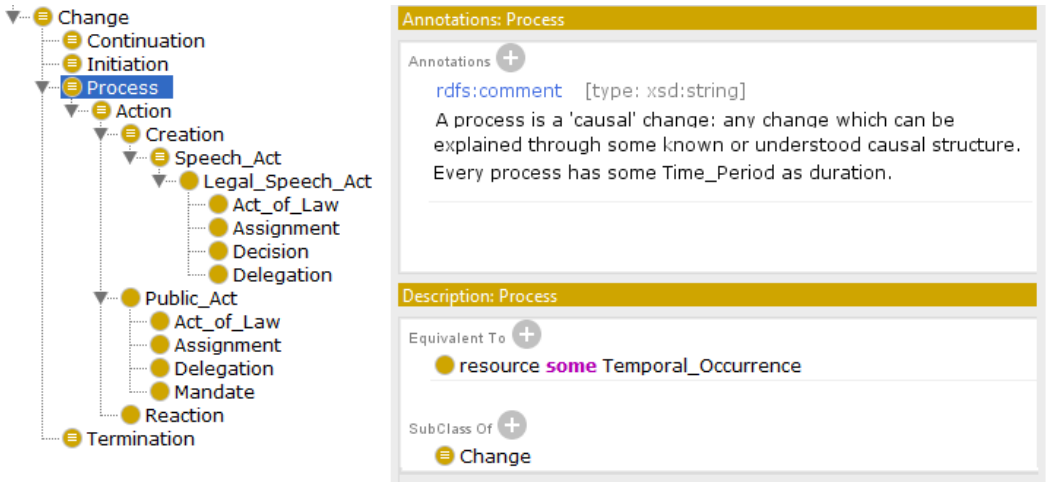


FIGURE 4.25: The concept `Process` in LIKF-Core.

Legal_Event , it is clearly stated in the upper module the `Event` concept. For this reason, we created the concept `Legal_Event` to be a core legal concept. A `Legal_Event` brings about some `Legal_Situation`.

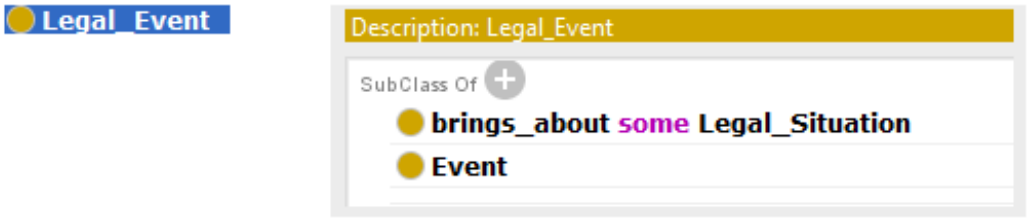


FIGURE 4.26: The concept `Legal_Event`.

Legal_Role A `Legal_Role` is a role played in a legal context. Legal role players can be both Agents and other 'things'.

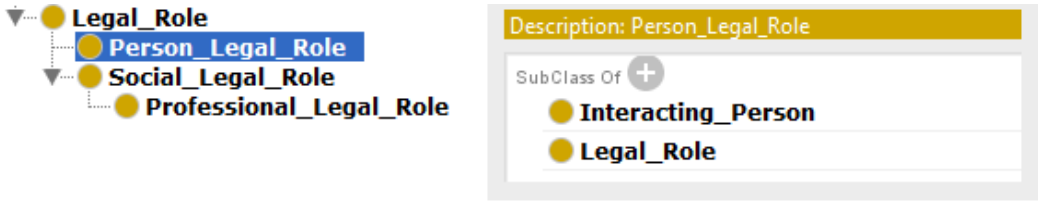


FIGURE 4.27: The concept `Legal_Role`.

Professional_Legal_Role A professional legal role is a legal profession of some person, examples: lawyer, judge etc.

FIGURE 4.28: The concept (`Professional_Legal_Role`).

4.6.3.3 Domain Ontology Module (DOM)

As a result, DOM consists of 70 classes, 54 object properties and 123 subClassOf axioms. The domain module is composed of categories that are related mainly to the criminal domain such as `Criminal_Act`, `Penalty`, `Misdemeanor`, `Violation`, etc.

Below, some domain module concepts are illustrated:

- `Criminal_Act` in figure 4.29.
- `Intentional_Felony` in figure 4.30.
- `Penalty` in figure 4.31.
- `Punishment` in figure 4.32.

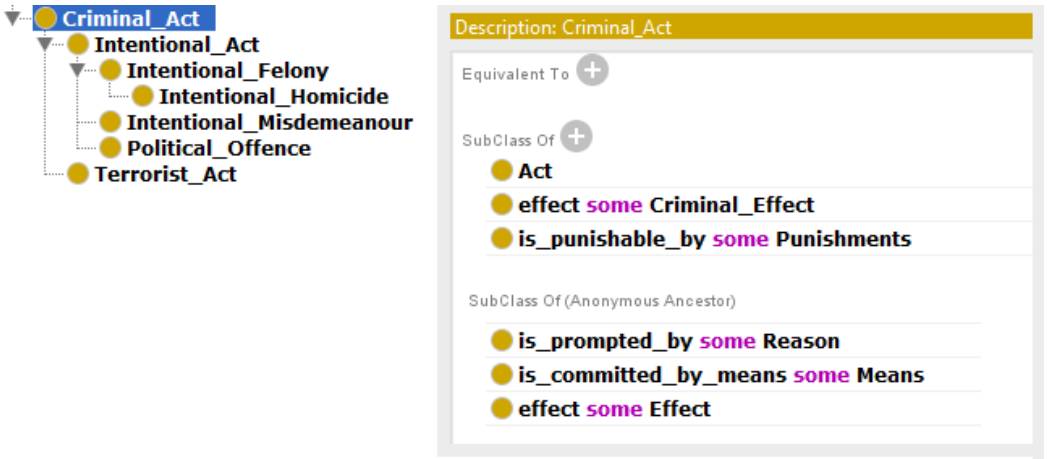


FIGURE 4.29: `Criminal_Act` in Protégé.

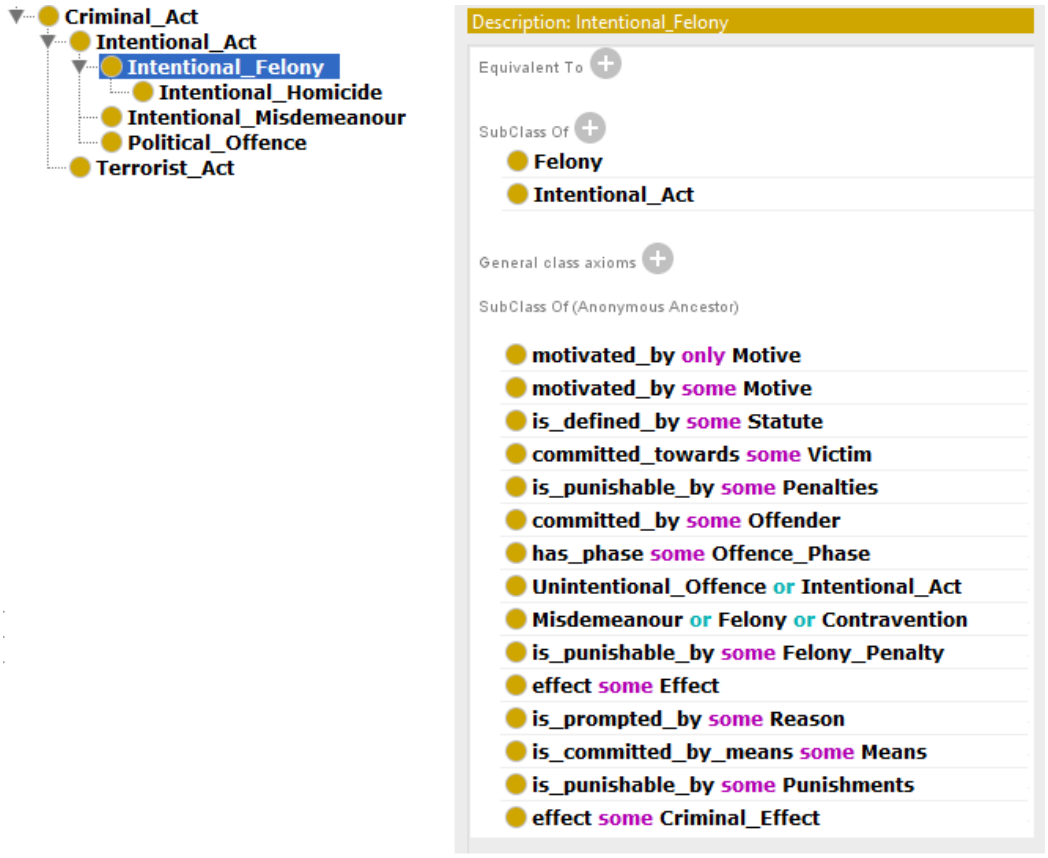


FIGURE 4.30: `Intentional_Felony` in Protégé.

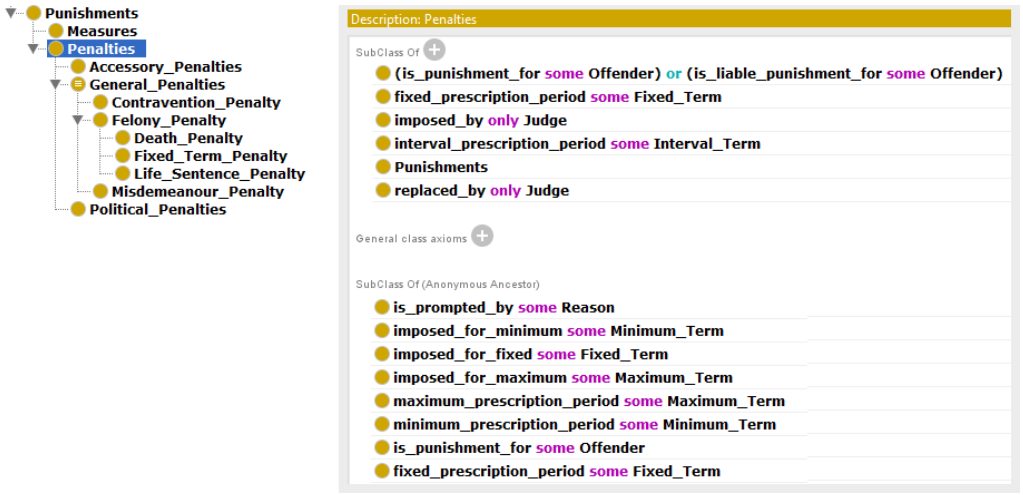


FIGURE 4.31: `Penalty` in Protégé.

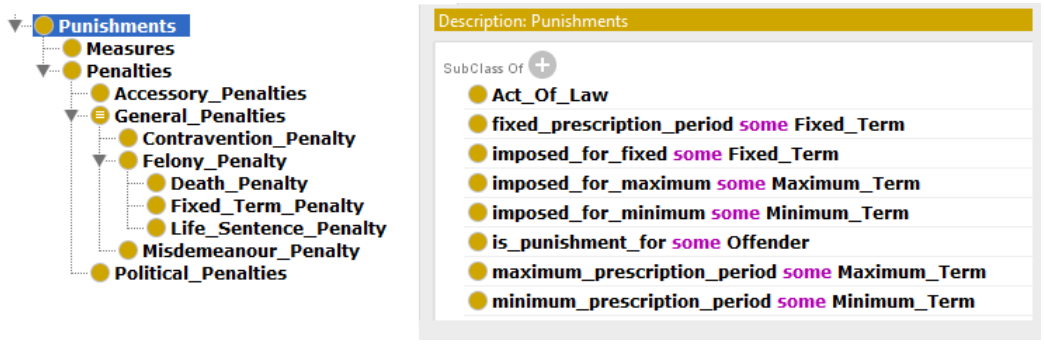


FIGURE 4.32: `Punishment` in Protégé.

4.6.3.4 Domain-specific Ontology Module

DSOM consists of 40 instances. The domain-specific module is composed of instances for concepts and relations of the domain module. These instances are specific for the Lebanese criminal domain. In figure 4.33, we present an instance of the class `Articles`, which is for representing the article 196 of the Lebanese criminal code.

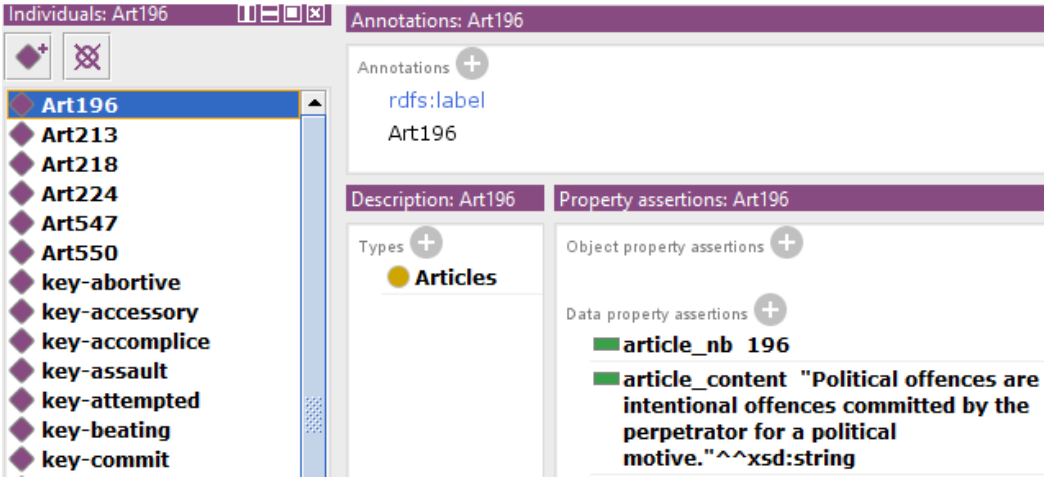


FIGURE 4.33: Instances of `Article` 196 of the Lebanese criminal code.

4.6.4 Integration of CriMOnto Modules

After building the ontology modules (upper, core, domain and domain-specific), there is a need for an integration process to combine them for composing the global ontology CriMOnto.

In CriMOnto, we will apply the integration process defined in MIROCL (see chapter 3, section 3.7.5). Thus, the modules are located on vertical conceptual levels from general (upper module) to specific (domain-specific module). For this reason, the mappings will be based mainly on a parent-child, or subsumption, hierarchical

relationship and established in Protégé as structural axiom of the form `subClassOf` and `instanceOf`.

Therefore, the hierarchical relationship is established among the concepts of modules. A domain expert, knowledgeable about the semantics of legal concepts, validates the proposed mappings. Below three examples of modules integration are detailed.

4.6.4.1 Example of Upper and Core modules mapping

In figure 4.34, we outline the legal core concept `Legal_Event` added as a subclass of the upper concept `Event`.

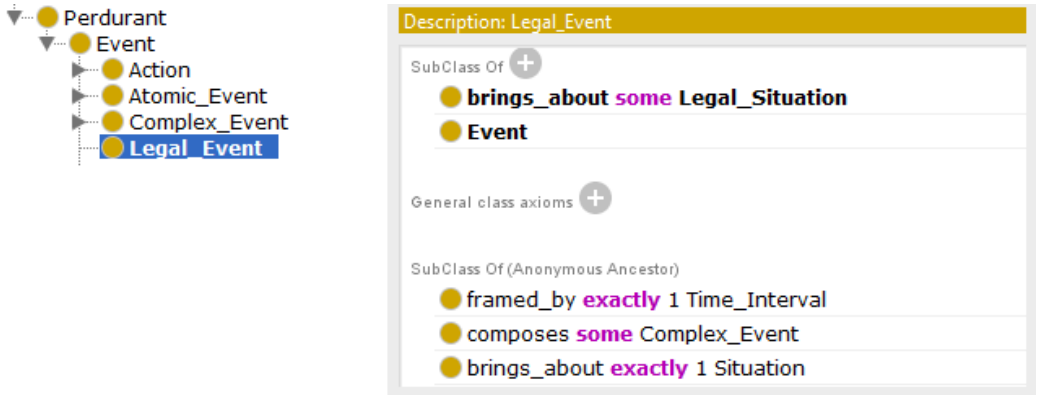


FIGURE 4.34: Example of Upper and Core modules mapping using `OWL:imports` formalism in Protégé.

In figure 4.35, we outline the legal core concept `Legal_Source` added as a subclass of the upper concept `Normative_Description`.

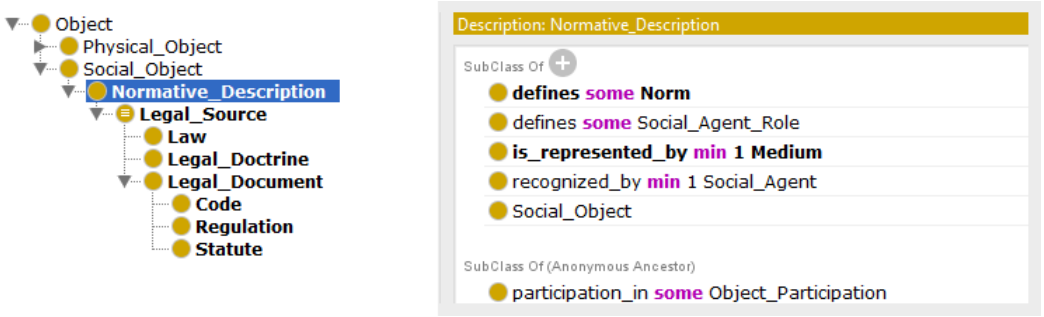


FIGURE 4.35: Example of Upper and Core modules mapping using `OWL:imports` formalism in Protégé.

4.6.4.2 Example of Core and Domain modules mapping

In figure 4.36, we outline the legal domain concept `Criminal_Act` added as a subclass of the core concept `Legal_Act`.

FIGURE 4.36: Example of Core and Domain modules mapping using `OWL imports` in Protégé.

4.6.4.3 Example of Domain and Domain-specific modules mapping

In figure 4.37, an example of the `instanceOf` mapping is depicted where the individual `Art212` added as an instance of the domain concept `Article`. The object and data property assertions are fulfilled with the appropriate values from the Lebanese criminal code.

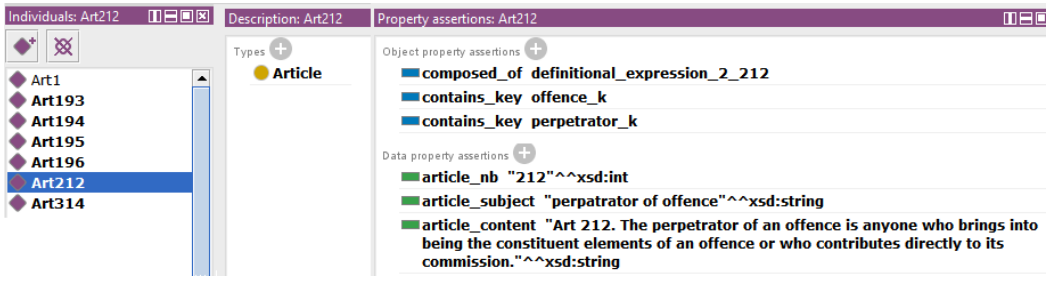


FIGURE 4.37: Example of Domain and Domain-specific modules mapping using `OWL imports` in Protégé.

4.6.5 CriMOnto Evaluation

Ontology evaluation is an important task that is needed in many situations. For example, during the process of building of an ontology, ontology evaluation is important to guarantee that what is built meets the application requirement (Tartir et al., 2010). In addition to the need during the process of building an ontology, evaluation and validation of ontologies are also useful for end users in domains where several ontologies with similar areas of interest are common. Additionally, ontology evaluation includes aspects of ontology validation and verification relating to structural, functional, and usability issues (Obrst et al., 2007). The verification ensures that the ontology is constructed correctly, and the validation ensures that the ontology represents the real world (Gomez-Perez et al., 2004).

Therefore, after building CriMOnto, there is a need to evaluate the characteristics and the validity of the resulting ontology. According to (Hartmann et al., 2004), evaluation is required during the whole life-cycle of an ontology in order to guarantee that what is built meets the requirements. Thus, we will discuss the evaluation phase of CriMOnto by referring to the studies presented in chapter 2 section 4.6.5.

From our perspective, the kind and purpose of the ontology will affect its validation. Since CriMOnto is a legal domain ontology and will be used for reasoning purposes mainly for building a rule-based legal reasoning model, its evaluation will be based on the following levels:

- Ontology Definitions: in this level, the evaluation process will cover the architecture of the ontology, its hierarchy, the syntax and the content of the definitions. For this level, a logic-based reasoner such as Pellet in Protégé can help in checking the consistency of the ontology as well as a legal expert who can verify closely all the syntactical and lexical issues.
- Application-based: as aforementioned, CriMOnto will be used for building a rule-based reasoning model for a legal knowledge-based system that will be be discussed in the chapters 5 and 6. Therefore, the results of this usage will affect the validity of CriMOnto.

4.7 Similar Works

In the literature, several studies that deal with the building of legal ontologies are found. We classified them in three main categories: ontologies for modeling legal norms, middle-out building of legal ontologies and ontologies for the criminal domain.

- Building ontologies for modeling legal norms (Cherubini et al., 2008; Gostojic et al., 2013; Machado et al., 2014):

  - (Cherubini et al., 2008) designed a preliminary ontology-based computable model of the normative framework and of the sequences of operational activities of public bodies (or enterprises) for producing services. The main methodological assumption adopted by the authors of this study is to differ between ontology of norms and ontology of procedures. Furthermore, they define a mapping procedures between the two ontologies where all the overlapping concepts and the dependency relations are represented and managed by a meta-model.
  - (Gostojic et al., 2013) described a formal model of legal norms, using their elements and elements of legal relations they regulate, intended for the development of expert systems for semi-automatic drafting and semantic retrieval and browsing of legislation. The scope of the model are general and abstract legal norms, abstract social relations, abstract subjects, abstract objects and legislation. A top-down approach is applied for the ontology development in order to identify and formally specify concepts that are essential for the description of a legal system (a system of legal norms). The general concepts are imported from the foundational ontology DOLCE and the legal concepts from the legal ontology CLO.
  - (Machado et al., 2014) proposed the building of a reference legal ontology of relationships for Brazilian civil law system using the ontology modeling language OntoUML. The main goals of the ontology are conveying a legal knowledge base and supporting reasoning knowledge-based systems.

- Application of middle-out strategies for building legal ontologies (Casellas, 2008a; Francesconi et al., 2010; Saias et al., 2005):
  - (Saias et al., 2005) tend to automatically create a legal ontology from a set of legal documents where the top-down strategy choose an already existent top-level legal ontology. Meanwhile, the purpose of the bottom-up is to identify the concepts and their properties using NLP techniques.
  - (Casellas, 2008a) developed a legal ontology, called Ontology of Professional Judicial Knowledge (OPJK), to map questions of junior judges to a set of stored frequently asked questions. for this purpose, two main strategies are identified: (1) top-down where top concepts from upper ontologies such as DOLCE Lite (CLO), SUMO, PROTON, and LKIF-Core are taken into account; (2) bottom-up where the acquisition of conceptual knowledge from textual resources is performed by ontology learning from texts. OPJK methodology focus on distributed ontology development involving different stakeholders with different purposes and needs and usually not at the same location.

  - (Francesconi et al., 2010) have proposed a methodology, which combines top-down and bottom-up strategies in a middle-out, for multilingual legal knowledge acquisition and modeling in the work of the DALOS KOS project. The top-down strategy defines the conceptual language-independent structure of the legal domain under consideration on the basis of expert judgment. This structure is language-independent, modeled manually as an ontology. Another top-down approach is the exploitation of the explicit structure of legal texts, which enables the targeted identification of text spans that play an ontological role and their subsequent inclusion in the knowledge model. Meanwhile, the bottom-up strategy uses semi-automatic NLP techniques for the knowledge acquisition from texts. The domain application of this work is the consumer protection law.

- Building ontologies for the criminal domain (Breuker, 2003; Asaro et al., 2003; Bezzazi, 2007; Rodrigues et al., 2016).

  - (Breuker, 2003) described the use of various ontologies for the information management of documents coming from and related to criminal trial hearings. Tis work is part of the e-COURT European IST project. Particularly, the author focused on the role of the *upper* ontology *LRI-core* in providing anchors and interpretation to the various legal domain ontologies. The role of LRI-core is exemplified by an ontology about Dutch criminal law (see section 2.2.3.3).

    The hard core of the OCL.NL consists of actions which are categorized in crime and punishment:

    * The criminal actions themselves (called 'offences') that are executed by the person who is successively acting as suspect, defendant, and eventually convict.

    * The punishments that are declared by the legal system.

  - (Asaro et al., 2003) outlined a construction of The Italian Crime Ontology. In the first phase of crime modeling, they have used UML class diagram to formalize the ontological concept of crime. For them, the crime is composed of offender, behavior, penalty and optionally by an event and coercions.

  - (Bezzazi, 2007) proposed the implementation of a formal ontology for criminal law and the application of the counter-factual reasoning on it. The domain application of this work is the legal texts of the French criminal law related to the cybercrime. The tools used in this study are the ontology editor Protégé, and the reasoning system Racer.

– (Rodrigues et al., 2016) proposed an ontological formalization for the Theory of Crime from Brazilian Penal Code, as well as for Property Crimes applications. The aim of this work, which is inspired by the UFO-B foundational ontology of events, is to support some decision-making process, as the agents behavior classification and the inference of punishments. UFO-B is used for modeling the different violations of individual property, in terms of events and participating states. Two main ontologies are developed in this work: the domain ontology *OntoCrimeAlpha* and the application ontology *OntoPropertyCrime*. OntoCrimeAlpha is built inspired by LKIF and UFO-B. Meanwhile, the purpose of OntoPropertyCrime is to map the allowed and prohibited behaviors of the set of articles describing the crimes against property. Thus, the concepts, relationships, properties and axioms were built based on the domain model. For developing the ontologies, the methodology *Methontology* is applied. The relevant information are extracted from official texts in a middle-out strategy: from a list of relevant terms, the specialization and/or generalization of concepts is generated.

4.7.1 Discussion

Concerning the category of ontologies for modeling legal norms, the approaches are recognized as top-down where reusing of upper ontologies is applied. They focused on modeling the hierarchical aspect of legal norms more than their content which is related mainly to the textual resources.

The category of building legal ontologies using middle-out approaches is characterized mainly the application of two strategies top-down and bottom-up. In the top-down, existent validated ontologies are reused. Meanwhile, the bottom-up strategy applied the ontology learning process.

For the category of building ontologies for the criminal domain, the works focused mainly on modeling the criminal aspect either by reusing foundational ontologies such as UFO-B (ontology of events) or LRI-CORE, or directly from textual resources.

We conclude that some limitations are recognized in these works such as the missing of the participatory approach, the lack of reusing existent ontologies that could simplify the building process, the focus on the general modeling of the norms and the deficiency of the modeling of the procedural aspect of the norms for reasoning purposes.

Therefore, by applying MIROCL for building the legal domain ontology CriMOnto, we covered the main aspects proposed in these works and enhanced them in one coherent approach. The main aspects identified for building CriMOnto are: (1) using a middle-out strategy for the ontology building process, (2) application of ontology

modularization for dividing the target ontology into modules, (3) using OntoUML as an ontology-based conceptual modeling language for modeling the upper and core modules of CriMOnto, (4) an existent foundational ontology (UFO) and legal core ontology (LKIF-Core) are reused for simplifying the ontology building process, (5) application of ontology learning process from texts for constructing the domain and domain specific modules, (6) the contribution of different stakeholders (ontology engineers, knowledge engineers, legal experts and linguistic experts) in the ontology building process.

	Middle-out	Modularization	Collaboration	Reuse
(Cherubini et al., 2008)	-	+	-	-
(Gostojic et al., 2013)	-	-	-	+
(Machado et al., 2014)	-	-	-	+
(Casellas, 2008a)	+	-	+	+
(Francesconi et al., 2010)	+	-	+	+
(Asaro et al., 2003)	-	-	-	-
(Bezzazi, 2007)	-	-	-	-
(Rodrigues et al., 2016)	+	-	-	+
CriMOnto	+	+	+	+

TABLE 4.6: Ontologies for modeling legal norms.

4.8 Conclusion

In this chapter, we have presented a bi-phased ontology-based approach for modeling legal norms. The aim of the first phase is to model the content of the legal norms in a legal domain ontology. Meanwhile, the second phase tends to model and formalize the procedural aspect of the legal norms based on the developed ontology by using a logic rule language. The focus of the current chapter is on the first phase where a modular legal domain ontology named CriMOnto for modeling legal norms of the criminal domain is obtained. For building CriMOnto, the proposed approach MIROCL (see chapter 3) is applied. The main phases of MIROCL are discussed: identification of data sources, building of ontology modules and ontology integration.

Regarding the data sources, we admitted an heterogeneity of selection where two main existent validated ontologies are selected which are UFO, as foundational ontology, and LKIF-Core, as legal core ontology. Moreover, the Lebanese Criminal Code is specified as textual source.

Concerning the building of ontology modules, the middle-out strategy is applied. The top-down tends to develop the upper and core modules using an ontology-driven conceptual modeling process and OntoUML, as conceptual modeling language, as well as a partial reuse process of the identified ontologies (UFO and LKIF-Core). Meanwhile, for building the domain and domain-specific modules, an ontology learning process defined in five main steps is performed. These steps are: material selection, tool selection, pre-processing, extraction of ontology components and re-engineering.

Finally, the developed modules are integrated using simple hierarchical mappings such as `subclassOf` and `instanceOf`.

CriMOnto will be used for building a rule-based legal reasoning model which will be discussed in chapters 5 and 6.

Chapter 5

Modeling and Formalizing the Procedural Aspect of Legal Norms

Contents

5.1 Overview

The main focus of this thesis is building a well-founded legal domain ontology using a novel collaborative approach (see chapters 3 and 4). The main purpose of the developed ontology is to be used for reasoning applications such as rule-based reasoning systems. In this chapter, we discuss the second phase of the proposed ontology-based approach which is the modeling and formalizing process of the procedural aspect of the legal norms based on the legal domain ontology CriMOnto (section 5.2). The section 5.2 analyses the concept of integration of rules and ontologies (section 5.2.1). Then, the selection of the most appropriate rule language for modeling and formalizing the legal norms of the Lebanese criminal code is outlined in section 5.2.2.1 as well as the existent rules reasoners for swrl (section 5.2.2.2). A case study that defines the application of the ontology-based approach for modeling and formalizing an excerpt of the legal norms of the Lebanese Criminal code is presented in section 5.3. Finally, the similar works are outlined in section 5.4 and the section 5.5 concludes the chapter.

5.2 Ontology-Based Approach for Modeling and Formalizing Legal Norms

Reasoning with both ontological knowledge and rule-based knowledge has recently gained in interest in the Semantic Web community (Heymans et al., 2005). Generally, extracting and formalizing norms remain a highly knowledge and labor intensive task, creating a significant bottleneck between the semantic content of the source material, expressed in natural language, and computer-based, automatic use of that content (Wyner et al., 2011). The legal domain is one of the most challenging area for developing applications based on the Semantic Web principles because of the complex nature of legal information and document workflow, as well as the peculiarities of legal user's information needs, which require advanced information retrieval and reasoning services (Francesconi, 2016). From this perspective, we have proposed an ontology-based approach for modeling and formalizing legal norms grounded mainly on a legal domain ontology in order to simplify the gap between norms represented in natural language and the formalized logic rules for reasoning purposes.

In chapter 4, the two-phased ontology-based approach for modeling legal norms is discussed, mainly, the first phase, which is the modeling of the content of the legal norms. This phase resulted in an ontological model, which is the criminal domain ontology (CriMOnto), aiming to provide a well-founded representation of concepts and semantic relationships among them in the context of the norms of the criminal domain. In this chapter, we present the second phase which is the modeling and

formalizing of the procedural aspect of the legal norms using logical formalisms in order to obtain list of formalized rules (see Figure 5.1).

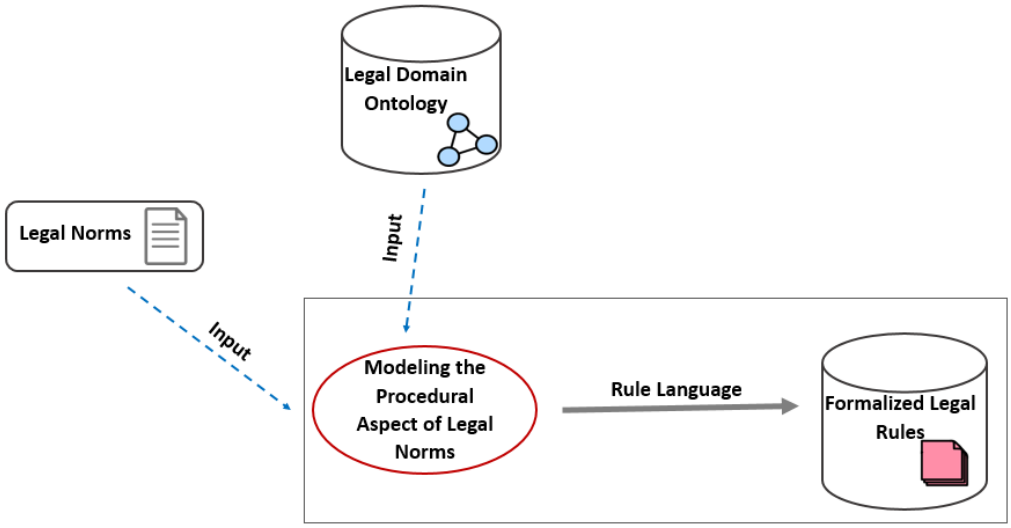


FIGURE 5.1: The second phase of the ontology-based approach for modeling legal norms.

In the proposed approach, the reasoning with rules and ontologies are based mainly on the semantic web standards and tools (see Figure 5.2). Thus, the main requirements of this approach are:

- The modeling of the procedural aspect of the the legal norms is based mainly on the developed criminal domain ontology. Thus, there is a need to integrate the ontology with the legal rules.
- There is a need for a rule language that respects this integration.

For this purpose, we will overview the domain of the integration of ontologies and rules and the existent approaches. Furthermore, the existent logic rule languages are discussed in order to select the most convenient for this study.

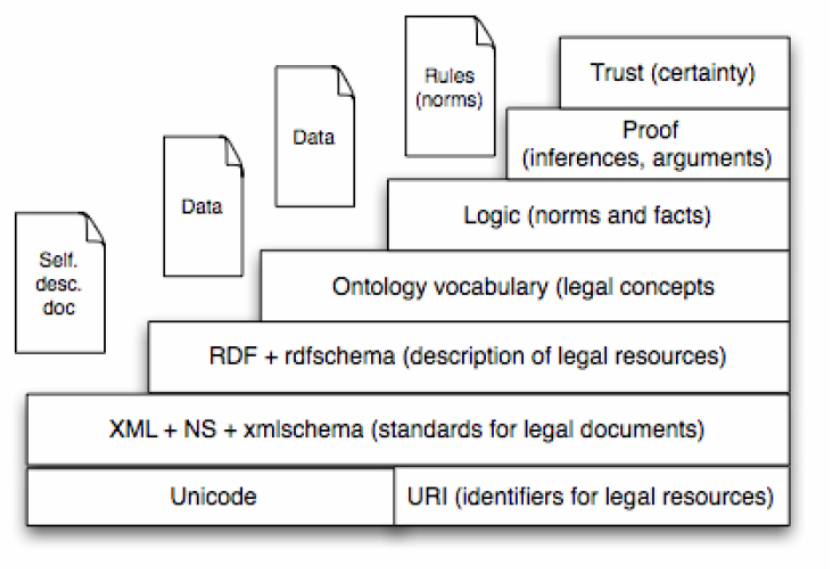


FIGURE 5.2: Levels of the legal semantic web (Biasiotti et al., 2008; Sartor, 2009).

5.2.1 Integration of Rules and Ontologies

Rules and ontologies represent two main components in the semantic web vision (Eiter et al., 2006b). They ensure that information available on the World Wide Web is machine-readable (Kaneiwa et al., 2009). Using rules in conjunction with ontologies is a major challenge for the semantic web (Golbreich, 2004). Ontologies, or conceptual models, are considered as the common and distinctive conceptualizations of a domain of knowledge (Valente et al., 1994a). They comprise five main modeling primitives: concepts, taxonomic relations (sub-class relations), non-taxonomic relations, axioms and instances (individuals). OWL (Web Ontology Language) is the standard representation language of ontologies. Meanwhile, rules are represented using inferential links according to the following pattern:

IF *precondition* THEN *conclusion*.

The role of inferential links is to govern the reasoning, they are considered as " rules for judging ". The reasoning is processed by the evaluation of the left side of the rule with reference to the knowledge base, and if this succeeds, the action specified by the right side is performed (Davis et al., 1984). Reasoning with rules and ontologies affects the Ontology and Rule layers of the semantic web layer cake (Berners-Lee, 2005).

In the literature, several approaches have been discussed for the integration between ontologies and rules such as, among others, (Grosof et al., 2003), (Eiter et al., 2004), (Eiter et al., 2006b), (Eiter et al., 2006a), (Rosati, 2006a), (Rosati, 2006b)

and (Antoniou et al., 2005a). Two main integration approaches have been distinguished: homogeneous, which are monotonic approaches, and hybrid, which are non-monotonic approaches (Antoniou et al., 2005b; Eiter et al., 2006a).

- Homogeneous: the integration between ontologies and rules is defined over a tight semantic integration where ontologies and rules are embedded in a common logical language (see Figure 5.3). Ontologies are treated as external sources of information, which are accessed by rules. Ontology concepts and properties may be defined through the rules (Antoniou et al., 2005b). The most typical homogeneous paradigms are:
  - Combination of OWL ontologies with SWRL rules (Horrocks et al., 2004) expressed in First Order Logic (FOL) which is family of monotonic Knowledge Representation (KR) formalisms.
  - Description Logic programs (DLP) which is a Knowledge Representation (KR) contained within the intersection of Description Logic (DL), which is the basis of ontology languages, and Logic Programs (LP), which is the basis of rules languages (Grosof et al., 2003).

FIGURE 5.3: Homogeneous integration of rules and ontologies.

- Hybrid: the integration between ontologies and rules is defined over a strict semantic separation where the ontology elements and the rules predicates are separated (see Figure 5.4). Ontology elements, such as concepts and properties, represent the conceptualization of the domain. Rules cannot define them but some application-specific relations. Thus, in the hybrid approach, the ontology remains unchanged and rules are built on top of ontologies (Eiter et al., 2006b). In this strategy, rules are expressed in Logic Programming LP formalism which is family of non-monotonic Knowledge Representation (KR). The most typical hybrid approaches are, among others, Answer Set programming (ASP) (Gelfond et al., 1991; Gelfond et al., 1988), dl-programs (Eiter et al., 2004) and DL+log (Rosati, 2006a).

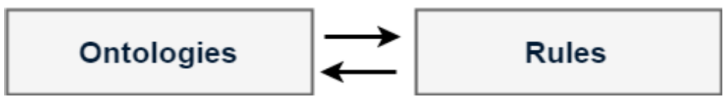


FIGURE 5.4: Hybrid integration of rules and ontologies.

5.2.2 Formalizing Legal Rules

Legal regulations are expressed in natural language. To make them automatically processable for reasoning or information extraction, they must be represented in a machine readable form (Wyner et al., 2013).

In the last years, an extensive research has been devoted for developing rule languages in the legal domain (Gordon et al., 2009). In this regard, new languages are devised, or the existing ones are adjusted, specifically for documenting and modeling the semantics of business vocabularies, facts, and rules such as RuleML (Governatori, 2005; Grosof, 2004) and SBVR[1]. Meanwhile, the faithful representation of legal rules is obviously crucial for representing legislative documents, regulations, and other sources of law (Gordon et al., 2009).

In the literature, there are significant experiences for representing legislative documents throughout XML languages such as CEN MetaLex (Boer et al., 2002b) (Boer et al., 2002a) and AKOMA NTOSO[2].These works have mostly focused on representing legal documents rather than modeling directly legal rules (Gordon et al., 2009). They are not based on robust or comprehensive conceptual models for representing legal rules to be applied in the legal domain.

In this context, few works are devoted for devising rule interchange languages specifically for the legal domain such as LKIF and LegalRuleML.

Important requirements for legal rule languages from the field of AI&Law include the following (Gordon et al., 2009) (Palmirani et al., 2011):

- Isomorphism (Bench-Capon et al., 1992; Bench-Capon et al., 2009): the term has been seen as a desirable property of executable representations of law to be used in legal knowledge based systems. Isomorphism is intended to capture the notion of creating a well defined correspondence between source documents and the representation of the information they contain used in the system. To ease validation and maintenance, there should be a one-to-one correspondence between the rules in the formal model and the units of natural language text which express the rules in the original legal sources, such as sections of legislation. This entails, for example, that a general rule and separately stated exceptions, in different sections of a statute, should not be converged into a single rule in the formal model.
- Rule Semantics: Any language for modeling legal rules should be based on a precise and rigorous semantics, which allows for correctly computing the legal effects that should follow from a set of legal rules.

[1] http://www.omg.org/spec/SBVR/, retrieved 22-12-2017
[2] http://www.akomantoso.org/, retrieved 22-12-2017

- Defeasibility (Gordon, 1995; Prakken et al., 1996): for managing exceptions, conflicting norms, different interpretations. For instance, when the antecedent of a rule is satisfied by the facts of a case, the conclusion of the rule presumably holds, but is not necessarily true.
- Contraposition (Prakken et al., 1996): Rules do not counterpose. If some conclusion of a rule is not true, the rule does not sanction any inferences about the truth of its premises.
- Normative effects: obligations, permissions and prohibitions are examples of normative effects that follow from applying rules.

5.2.2.1 Selection of a Rule Language for Modeling Legal Norms

In this section, a selection process, from the list of rule interchange languages presented in chapter 2, section 2.4.3.3, for choosing the most appropriate rule language, is discussed.

Actually, the main goal of the thesis is to build a well-founded legal domain ontology for reasoning purposes. For this purpose, the focus was set on designing a novel approach that aims to develop a well-founded legal domain ontology. Therefore, the target ontology will be considered as a ground for an ontology-based approach for modeling and formalizing the legal norms of the criminal domain (see section 5.2).

From this perspective, the following requirements define the selection of the appropriate rule language:

- The rule language should be suitable for ontology-based models. For instance, the rules can be expressed using the OWL ontology concepts, object and data properties.
- The rule language should be compatible with the OWL syntax.
- Since, the legal norms of the criminal code can almost be expressed in first-order logic, the rule language should allow to write them in form of Horn-like rules that can be expressed in terms of OWL concepts.

Thus, for the current work it is just necessary to prove the efficiency of the developed ontology in modeling and formalizing the legal norms by applying the ontology-based approach and by using an appropriate rule language. For this reason, we found that the rule language SWRL (Semantic Web Rule Language[3], see chapter 2, section 2.3.3.1) can serve in formalizing part of the legal norms of the criminal code

[3]https://www.w3.org/Submission/SWRL/

which are in the form of first-order logic. In this regard, the proposed ontology-based approach will be considered as homogeneous respecting a tight semantic integration of the domain ontology and SWRL.

In fact, SWRL represents an approach that gathers ontology and rules in product development (Fiorentini et al., 2010). It provides a way to define derivation rules based on ontology, the main purpose of which is to inference the individuals in semantic web application. Therefore, it allows to write Horn-like rules that can be expressed in terms of OWL concepts and that can reason about OWL individuals. The rules can be used to infer new knowledge from existing OWL knowledge bases (O'Connor et al., 2005; O'Connor et al., 2008). Thus, SWRL is considered as the most appropriate rule language for ontology-based models. Moreover, the SWRL specification does not impose restrictions on how reasoning should be performed with SWRL rules. Thus, variety of rule engines can be used to reason with the SWRL rules.

Meanwhile, SWRL suffers from some limitations in expressing rules such as:

- Concerning the modeling of legal rules, since SWRL rules are Horn clauses, it is not possible to model legal rules in an isomorphic way. Most legal rules would need to be modeled using several SWRL rules (Gordon et al., 2009).
- SWRL works usually under the Open World Assumption (OWA) (Fiorentini et al., 2010), since they focus on the Semantic Web, that deals with an unlimited knowledge resource. In this context, SWRL suffers from the lack of non-monotonic features, thus it does not support the negation as failure. Therefore, many common rule-like statements cannot be expressed in SWRL.

  This problem can be solved by expressing it explicitly to the system by using classical negation with the class description `owl:complementOf`.

  A class `C1` is defined as complement of class `C2`, if `C1` contains all individuals that do not belong to `C2`. `ComplementOf` in OWL is analogous to the logical negation. It is the same as if the `NOT` operator of set theory is applied to classes. A typical example of this case are the `Intentional_Legal_Event` and `Unintentional_Legal_Event` classes (see Figure 5.5).

  The class `Unintentional_Legal_Event` is defined as `NOT Intentional_Legal_Event`.

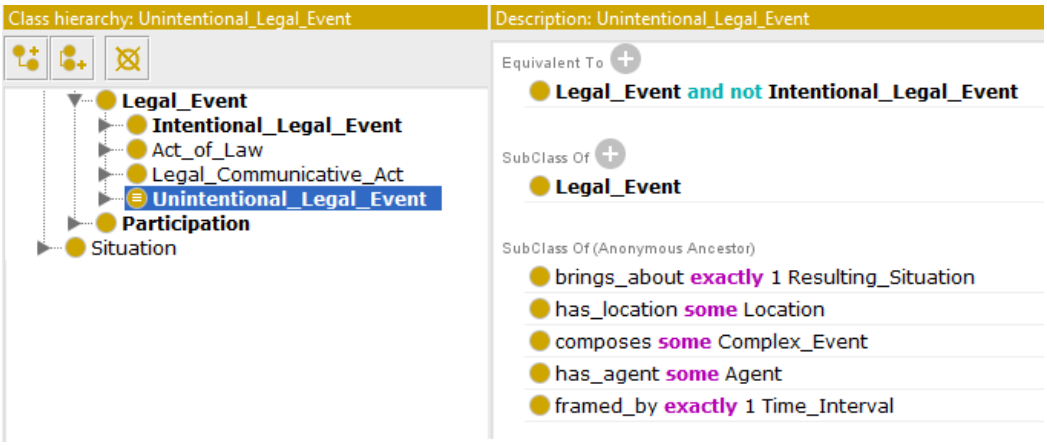


FIGURE 5.5: Example of complement classes.

- The problem of undecidability that is possibly solved by introducing the notion of DL-*safe* rules which restricts the application of SWRL rules only to individuals (Motik et al., 2005). Moreover, OWL2, successor of OWL, added new features based on DL SROIQ (Horrocks et al., 2006) which can completely internalize DL rules as decidable fragment of SWRL.

- SWRL has difficulties dealing with complex rules (Fortineau et al., 2012). For instance, it is not possible in theory to create rules with numbered predicates (Elenius et al., 2009). This limitation can be overtaken with SWRL built-ins (Rossello-Busquet et al., 2011). Built-ins allow to define a rdf list of OWL arguments, that then feed a SWRL rule as an unique antecedent. However, this method is complex.

- SWRL rules cannot be used to modify existing information in an ontology (Rossello-Busquet et al., 2011). In fact, SWRL rules originally create new knowledge, i.e. new instances or new properties between existing instances (Fortineau et al., 2012).

- SWRL does not support the disjunction of atoms. So, for example, the following rule is not possible:

  $A(?x) \text{ or } B(?x) \implies C(?x)$

  In order to solve this limitation, the following two rules will produce the intended effect:

  $A(?x) \implies C(?x)$

  $B(?x) \implies C(?x)$

Concerning the complex rules of the criminal code, such as the defeasible rules (e.g. Articles 2, 9, 28, 162, 200, 203, 204, 250 and 335) that SWRL cannot express, there is a need for non-monotonic formalisms such as Defeasible Logic (DL) (Nute, 1994) to translate them.

5.2.2.2 Rule Reasoners for SWRL

A reasoner is a program that infers logical consequences from a set of explicitly asserted facts or axioms and typically provides automated support for reasoning tasks such as classification, debugging and querying (Dentler et al., 2011). The main reasoners that support rules, specifically SWRL are: Pellet, Hermit, KAON2 and RacerPro. In this section, two reasoners, Pellet and HermiT are outlined.

Pellet (Sirin et al., 2007b; Parsia et al., 2004) is a Java-based, open source OWL-DL reasoner. It can be used with Jena and OWL API libraries. It is based on the tableau algorithm and supports expressive description logics. The reasoner allows ontologies to use XML-Schema built-in and user-defined datatypes that extend numeric and date/time types. Moreover, the query engine of Pellet is capable of answering SPARQL queries. Finally, the tool supports reasoning with DL-safe rules encoded in SWRL.

HermiT is a free (under LGPL license) Java reasoner for OWL 2/SROIQ with OWL 2 datatype support (Glimm et al., 2014). It implements a hypertableau-based decision procedure, which allows the reasoner to avoid some of the nondeterministic behavior exhibited by the tableau calculus used in Pellet, The reasoner uses the OWL API 3.0, and is compatible with the OWLReasoner interface of the OWL API. This allows HermiT to be used in any application based on the OWL API, and it also allows the Protégé editor to use HermiT as a plugin. Apart from the standard OWL 2 reasoning task of entailment checking, HermiT supports several specialized reasoning services such as class and property classification, as well as a range of features outside the OWL 2 standard such as DL-safe SWRL rules, SPARQL queries, and description graphs.

In table 5.1, we compare the reasoners based on some basics characteristics inspired from the work of (Abburu, 2012): methodology, ABOX reasoning, OWL API, Protégé support, Jena support, availability, and implementation language.

- Methodology: indicates the procedure or an algorithm that is used by the reasoner for solving basic reasoning problems in description logics. E.g: Tableau (Baader et al., 2003c), Tableaux (Horrocks et al., 2005) etc.
- ABOX reasoning: reasoning with individuals and includes instance checking, conjunctive query answering and ABox consistency checking.
- OWL API based applications (such as Protégé).
- Protégé support: indicates whether the reasoner can be used with protégé tool or not.

- Jena support: indicates whether the reasoner can be used with Jena API or not.
- Availability: indicates availability of reasoner. Many reasoners are free and open.
- Implementation language: indicates the language which is used to implement a reasoner.

Reasoner	Pellet	RacerPro	HermiT	KAON2
Methodology	Tableau based	Tableaux based	Hyper-tableau based	Disjunctive Datalog programs
ABOX reasoning	Yes	Yes	Yes	Yes (except nominals)
OWL API	Yes	Yes	Yes	Yes
Protégé Support	Yes	Yes	Yes	Yes
Jena Support	Yes	No	No	No
Availability	Open source	Commercial	Open source	Commercial
Implementation language	Java	LISP	Java	Java

TABLE 5.1: Comparison of rule reasoners that support SWRL.

5.3 Case Study: Modeling and Formalizing the Legal Norms of the Lebanese Criminal Code

In the case study, we will discuss the application of the ontology-based approach for modeling and formalizing an excerpt of the legal norms of the Lebanese Criminal code. The proposed ontology-based approach is homogeneous and monotonic established by integrating the criminal domain ontology CriMOnto and the semantic web rule language SWRL. Actually, the approach is applied on approximately 100 selected norms which can be represented using the first-order logic.

As aforementioned, the formalized SWRL rules are OWL-level constructs; the unary predicates are class expressions from CriMOnto such as `Accomplice`, `Intentional_Act` and `Instigator`, and the binary predicates are object and data properties such as `motivated_to` and `offence_phase_type("Abortive")` respectively. Additionally, the SWRL rules match on named individuals such as `hard-labour` and `death`.

The SWRL rules are written and edited in Protégé. Concerning the selection of the appropriate reasoner that can handle the SWRL rules, and after reviewing the list of reasoners in section 5.2.2.2, we found that Pellet has the best support for SWRL rules. Actually, Pellet is open-source Java based OWL DL reasoner and it can be used with Jena and OWL API libraries.

5.3.1 Application of the Ontology-based Approach using Protégé

In the following, 5 legal norms (Articles) are selected from the Lebanese criminal code for the application of the ontology-based approach: 213, 196, 547, 218 and 550. The study is performed using Protégé based on the developed legal domain ontology CriMOnto (see chapter 4).

Article 213 *- An accomplice to an offence shall be liable to the penalty prescribed by law for the offence.*

In SWRL, the Article 213 is formalized using OWL classes and object properties based on the ontology CriMOnto as following:

$$\texttt{Accomplice(?x)} \wedge \texttt{commit(?x,?y)} \wedge \texttt{is\_punishable\_by(?y,?z)} \implies \texttt{is\_liable\_to\_punished\_by(?x,?z).}$$

Where, `Accomplice` is an OWL class of the ontology CriMOnto (see figure 5.6). `commit` is an object property that characterizes the class `Accomplice` (see Figure 5.6) in the following form:

$$\texttt{Accomplice commit some Offence.}$$

`is_punishable_by` (see Figure 5.7) is an object property that characterizes the class `Offence` (see Figure 5.8) in the following form:

`Offence is_punishable_by some Penalty.`

`is_liable_to_punished_by` is an object property that characterizes the class `Accomplice` (see Figure 5.6).

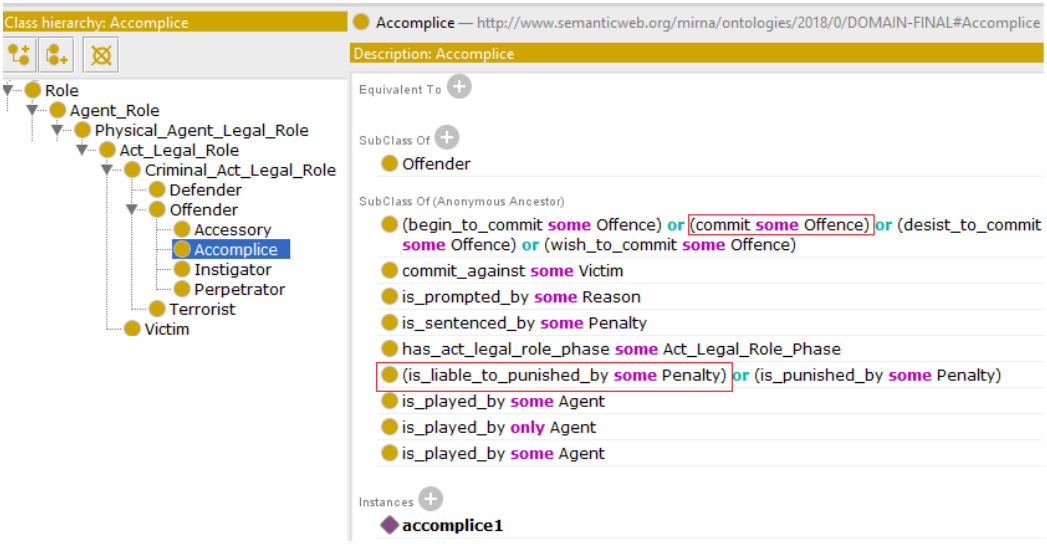


FIGURE 5.6: The class `Accomplice` in Protégé.

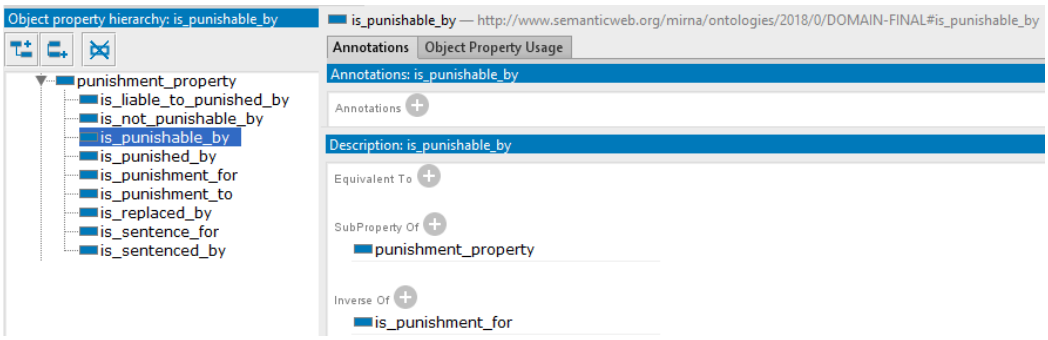


FIGURE 5.7: The object property `is_punishable_by` in Protégé.

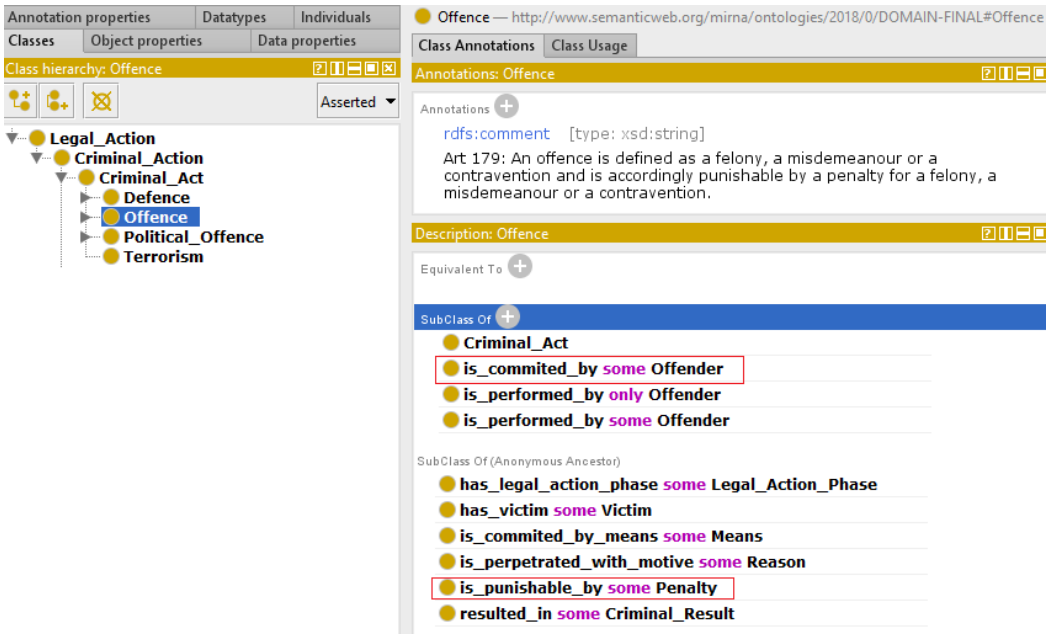


FIGURE 5.8: The class `Offence` in Protégé.

Given the following individuals: `accomplice1` for the class `Accomplice`, `offence1` for `Offence` and `penalty1` for `Penalty`. The execution of the Article 213 in SWRL based on the given individuals will be:

$$\texttt{Accomplice(accomplice1)} \wedge \texttt{commit(accomplice1, offence1)} \wedge \texttt{is\_punishable\_by(offence1, penalty1)} \implies \texttt{is\_liable\_to\_punished\_by(accomplice1, penalty1).}$$

The figure 5.9 depicts the inference of the SWRL rule of the Article 213 using the reasoner Pellet.

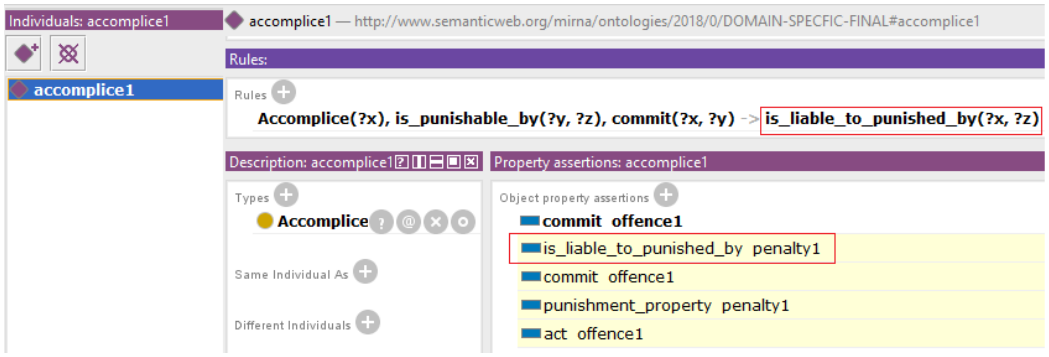


FIGURE 5.9: Article 213 using CriMOnto and SWRL in Protégé.

As shown in figure 5.9, the execution of the given rule results in inferring the object property `is_liable_to_punished_by` with the individual `penalty1` which is the head of the rule of the article 213.

Article 196 - *Political offences are intentional offences committed by the perpetrator for a political motive.*

In SWRL, the Article 196 is formalized using the OWL classes and object properties of CriMOnto as following:

$$\texttt{Offence(?x)} \wedge \texttt{Political\_Motive(?y)} \wedge$$
$$\texttt{is\_perpetrated\_with\_motive(?x,?y)} \implies \texttt{Political\_Offence(?x).}$$

Where, `Offence` and `Political_Motive` are OWL classes of CriMOnto depicted respectively in Figures 5.8 and 5.10.

The object property `is_perpetrated_with_motive` characterizes the class Offence in the following form:

$$\texttt{Offence is\_perpetrated\_with\_motive some Motive.}$$

FIGURE 5.10: The class `Political_Motive` in Protégé.

Given the following individuals: `offence2` for the class `Offence` and `political_motive1` for `Political_Motive`. The execution of the Article 196 in SWRL based on the given individuals will be:

$$\texttt{Offence(offence2)} \wedge \texttt{Political\_Motive(political\_motive1)} \wedge$$
$$\texttt{is\_perpetrated\_with\_motive(offence2, political\_motive1)} \implies$$
$$\texttt{Political\_Offence(offence2).}$$

In figure 5.11, the article 196 is formalized using SWRL and CriMOnto and inferred using the reasoner Pellet.

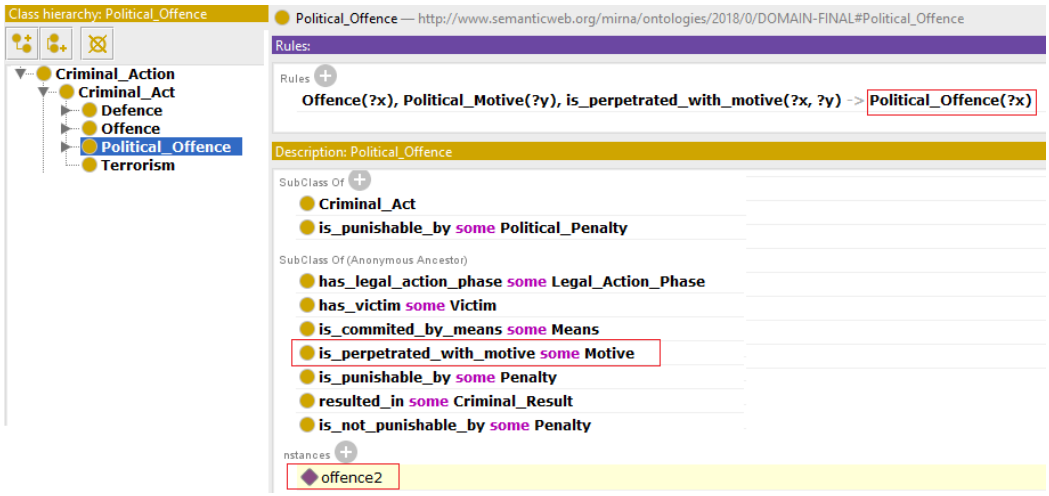


FIGURE 5.11: Article 196 using SWRL and CriMOnto in Protégé.

As shown in figure 5.11, the execution of the given rule results in inferring the head of the rule `Political_Offence` with the individual `offence2`.

Article 547 - *Anyone who intentionally kills another person shall be punishable by hard labour for a term of between 15 and 20 years.*

In SWRL, the Article 547 is formalized using the OWL classes and object properties of CriMOnto as following:

```
Rule: Intentional_Homicide(?x) ∧ Offender(?y) ∧ is_committed_by(?x,
?y) ⟹ is_sentenced_by(?y, hard_labour)∧
Felony_Penalty(hard_labour) ∧ has_max_sentence_term(hard_labour,
maximum_sentence_fixed_term1) ∧ has_minimum_sentence_term(hard_labour,
minimum_sentence_fixed_term1) ∧
sentence_fixed_term_type(maximum_sentence_fixed_term1, "year") ∧
sentence_fixed_term_value(maximum_sentence_fixed_term1, 20) ∧
sentence_fixed_term_type(minimum_sentence_fixed_term1, "year") ∧
sentence_fixed_term_value(minimum_sentence_fixed_term1, 15).
```

Where, `Intentional_Homicide`, `Offender`, `Felony_Penalty` are OWL classes of CriMOnto depicted respectively in figures 5.12, 5.13 and 5.14.

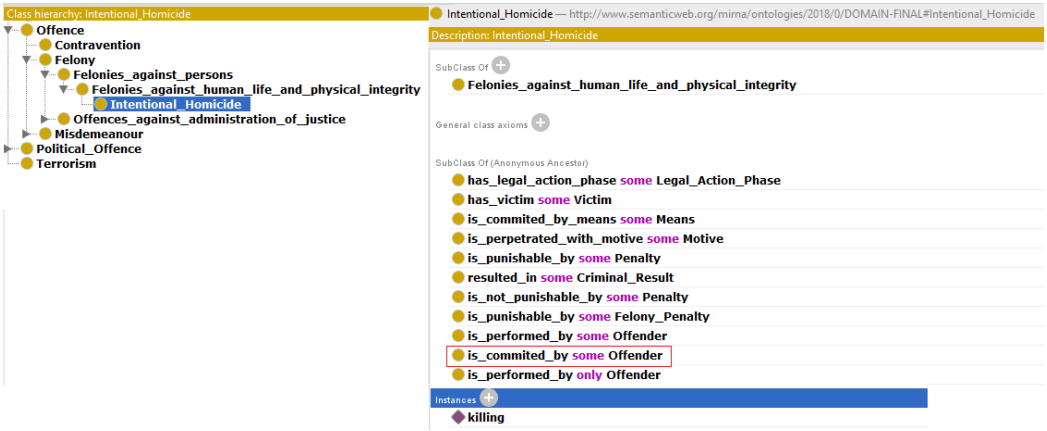


FIGURE 5.12: The class `Homicide` of CriMOnto in Protégé.

FIGURE 5.13: The class `Offender` of CriMOnto in Protégé.

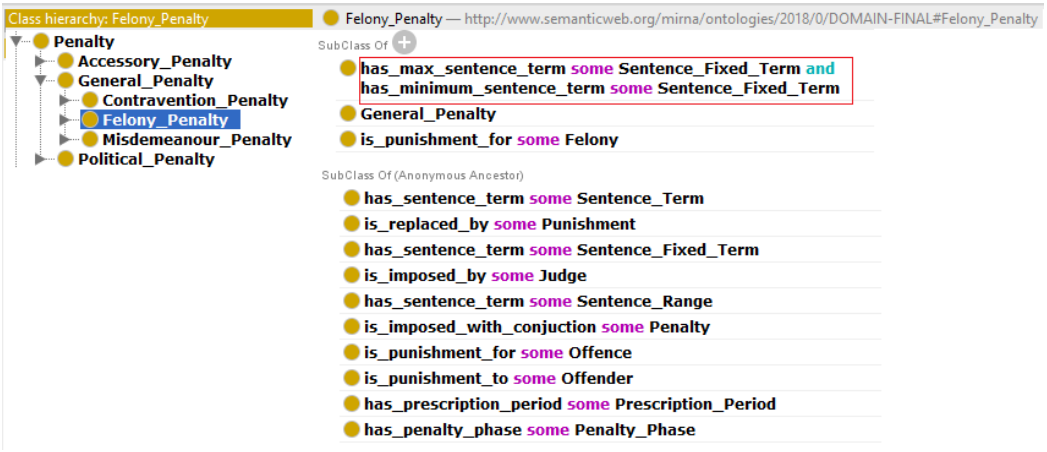


FIGURE 5.14: The class `Felony_Penalty` of CriMOnto in Protégé.

The object property `is_committed_by` characterizes the class `Intentional_Homicide` (see Figure 5.12). The object property `is_sentenced_by` characterizes the class `Offender` (see Figure 5.13). The object properties `has_max_sentence_term` and `has_min_sentence_term` characterize the class `Felony_Penalty` (see Figure 5.14).

`sentence_fixed_term_type` and `sentence_fixed_term_value` are data properties for the class `Sentence_Fixed_Term` (see Figure 5.15).

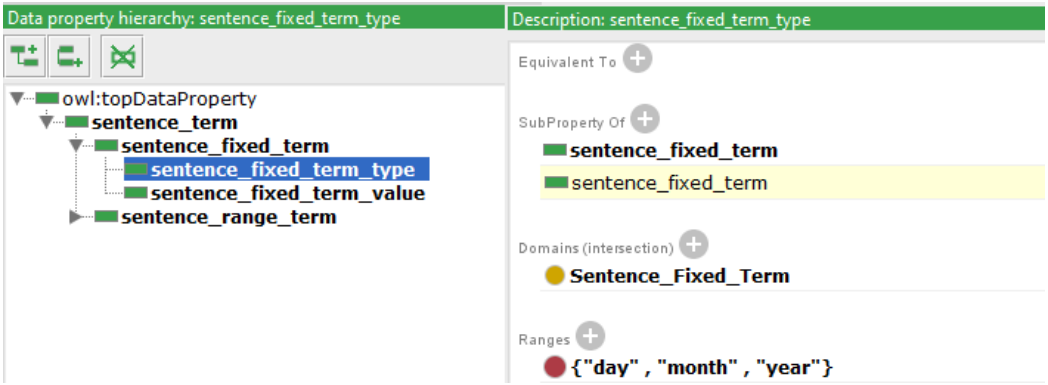


FIGURE 5.15: Data properties for the class `Sentence_Fixed_Term` in Protégé.

Given the following individuals: `killing` for the class `Intentional_Homicide` and `offender1` for `Offender`. The execution of the Article 547 in SWRL based on the given individuals will be:

```
Rule: Intentional_Homicide(killing) ∧ Offender(offender1) ∧
is_committed_by(killing, offender1) ⟹ is_sentenced_by(offender1,
hard_labour)∧ Felony_Penalty(hard_labour) ∧
has_max_sentence_term(hard_labour, maximum_sentence_fixed_term1) ∧
has_minimum_sentence_term(hard_labour, minimum_sentence_fixed_term1)
∧ sentence_fixed_term_type(maximum_sentence_fixed_term1, "year") ∧
```

$$\texttt{sentence\_fixed\_term\_value(maximum\_sentence\_fixed\_term1, 20)} \wedge$$
$$\texttt{sentence\_fixed\_term\_type(minimum\_sentence\_fixed\_term1, "year")} \wedge$$
$$\texttt{sentence\_fixed\_term\_value(minimum\_sentence\_fixed\_term1, 15).}$$

The figures 5.16 and 5.17 depict the SWRL version of the article 547 and its inference using the reasoner Pellet. It shows the inference of the rule based on the given individuals.

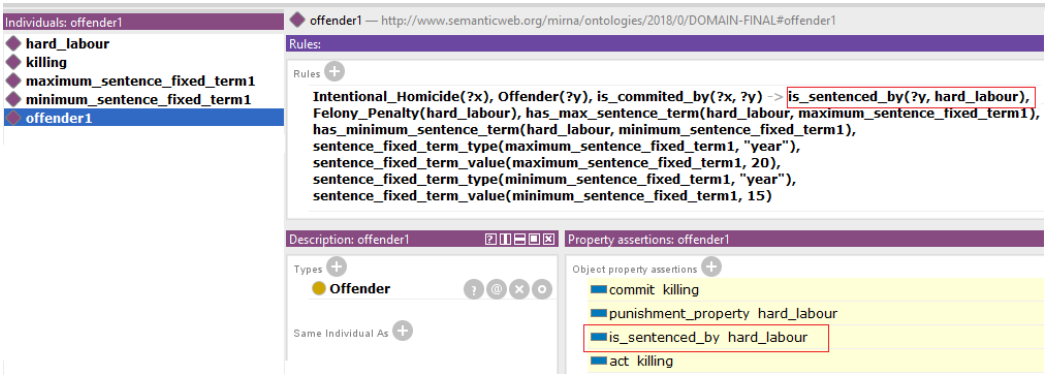


FIGURE 5.16: Inferring Article 547 using CriMOnto and SWRL in Protégé.

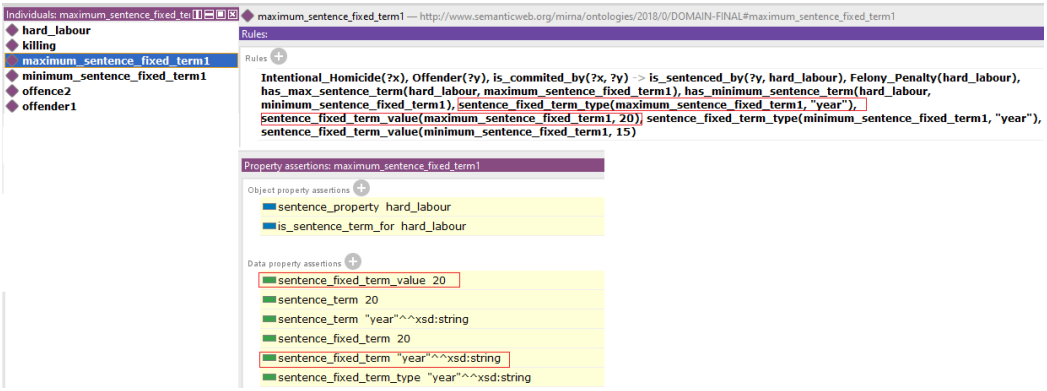


FIGURE 5.17: Inferring Article 547 using CriMOnto and SWRL in Protégé.

The figure 5.16 shows the execution of the rule and resulting with the head mainly the object property `is_sentenced_by`. Meanwhile, the figure 5.17 depicts the inference part of the data properties `sentence_fixed_term_value` and `sentence_fixed_term_type`.

Article 218 - *The instigator shall be liable to the penalty for the offence that he wished to commit, whether the offence was completed, attempted or abortive.*

In SWRL, the Article 218 is formalized using the OWL classes and object properties of CriMOnto as following:

$$\texttt{Instigator(?x)} \wedge \texttt{wish\_to\_commit(?x, ?y)} \wedge \texttt{Offence(?y)} \wedge \texttt{has\_legal\_action\_phase(?y, ?p)} \wedge \texttt{legal\_action\_phase(?p,"completed")} \wedge \texttt{is\_punishable\_by(?y, ?z)} \Longrightarrow \texttt{is\_liable\_to\_punished\_by(?x, ?z).}$$

$$\texttt{Instigator(?x)} \wedge \texttt{wish\_to\_commit(?x, ?y)} \wedge \texttt{Offence(?y)} \wedge \texttt{has\_legal\_action\_phase(?y, ?p)} \wedge \texttt{legal\_action\_phase(?p,"attempted")} \wedge \texttt{is\_punishable\_by(?y, ?z)} \Longrightarrow \texttt{is\_liable\_to\_punished\_by(?x, ?z).}$$

$$\texttt{Instigator(?x)} \wedge \texttt{wish\_to\_commit(?x, ?y)} \wedge \texttt{Offence(?y)} \wedge \texttt{has\_legal\_action\_phase(?y, ?p)} \wedge \texttt{legal\_action\_phase(?p,"abortive")} \wedge \texttt{is\_punishable\_by(?y, ?z)} \Longrightarrow \texttt{is\_liable\_to\_punished\_by(?x, ?z).}$$

Where, `Instigator` (see Figure 5.18) and `Offence` are OWL classes in CriMOnto.

`wish_to_commit` is an object property that characterizes the class Instigator. The objects properties `has_legal_action_phase` and `is_punishable_by` characterize the class Offence. The data property `legal_action_phase` define the class `Legal_Action_Phase`

FIGURE 5.18: The class `Instigator` in Protégé.

The article 218 is an example of rules with the logic operator `Or`. Three main values are given for the data property `legal_action_phase`: `"completed"`, `"attempted"` or `"abortive"`. For each value, the article is formalized using SWRL. The figure 5.19 depicts the version of the `"completed"` value.

Given the following individuals: `instigator1` for the class `Instigator`, `offence3` for the class `Offence` and `penalty1` for the class `Penalty`. The execution of the article 218 will be:

$$\texttt{Instigator(instigator1)} \wedge \texttt{wish\_to\_commit(instigator1, offence3)} \wedge \texttt{Offence(offence3)} \wedge \texttt{has\_legal\_action\_phase(offence3, phase1)} \wedge \texttt{legal\_action\_phase(phase1,"completed")} \wedge \texttt{is\_punishable\_by(offence3, penalty1)} \implies \texttt{is\_liable\_to\_punished\_by(instigator1, penalty1).}$$

The figure 5.19 depicts the SWRL version of the article 218 and its inference using the reasoner Pellet. It shows the inference of the rule based on the given individuals.

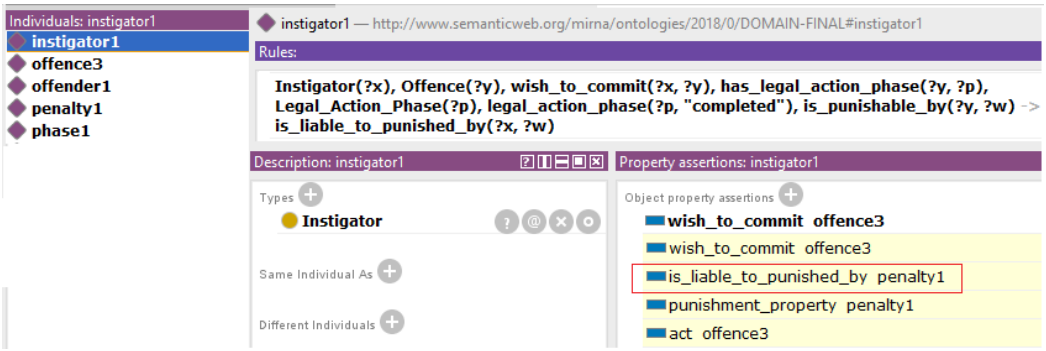


FIGURE 5.19: Article 218 using CriMOnto and SWRL in Protégé.

As shown in figure 5.19, the head of the rule is inferred which is the object property `is_liable_to_punished_by`.

Article 550 - *Anyone who causes the death of a person through beatings, violence, assault or any other intentional act without intending to kill him shall be punishable by hard labour for a term of at least five years.*

In SWRL, the Article 550 is formalized using OWL classes and object properties based on the ontology CriMOnto as following:

$$\texttt{Intentional\_Act(?x)} \wedge \texttt{resulted\_in(?x, ?y)} \wedge \texttt{has\_non\_goal(?x, ?k)} \Longrightarrow \texttt{is\_sentenced\_by(?x, ?h)} \wedge \texttt{has\_minimum\_sentence\_term(?h, minimum\_sentence\_fixed\_term2)} \wedge \texttt{sentence\_fixed\_term\_type(minimum\_sentence\_fixed\_term2, "year")} \wedge \texttt{sentence\_fixed\_term\_value(minimum\_sentence\_fixed\_term2, 5).}$$

Where, `Intentional_Act` is an OWL class in CriMOnto (see Figure 5.20). The object properties `resulted_in` and `has_non_goal` characterize the class `Intentional_Act` in the following form:

`Intentional_Act resulted_in some Criminal_Result.`

`Intentional_Act has_non_goal Goal.`

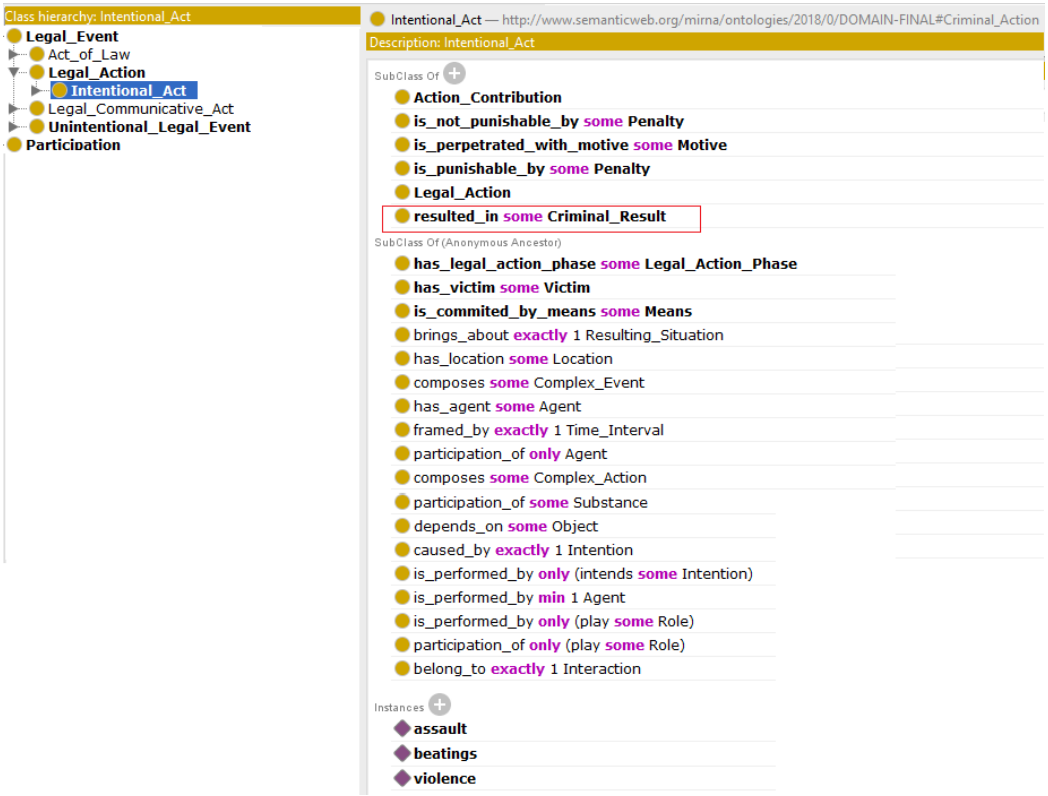


FIGURE 5.20: The class `Intentional_Act` in Protégé.

Given the following individuals: `violence` for the class `Intentional_Act`, `death` for the class `Criminal_Result`, `killing2` for the class `Goal`, `hard_labour2` for the

class `Fixed_Term_Hard_Labour` and `minimum_sentence_fixed_term2` for the class `Minimum_Sentence_Fixed_Term`.

The execution of the Article 550 in SWRL based on the given individuals will be:

$$\texttt{Intentional\_Act(violence)} \wedge \texttt{resulted\_in(violence, death)} \wedge \texttt{has\_non\_goal(violence, killing2)} \implies \texttt{is\_sentenced\_by(violence, hard\_labour2)} \wedge \texttt{has\_minimum\_sentence\_term(hard\_labour2, minimum\_sentence\_fixed\_term2)} \wedge \texttt{sentence\_fixed\_term\_type(minimum\_sentence\_fixed\_term2, "year")} \wedge \texttt{sentence\_fixed\_term\_value(minimum\_sentence\_fixed\_term2, 5).}$$

In figures 5.21 and 5.22, the article 550 is executed and inferred using Pellet.

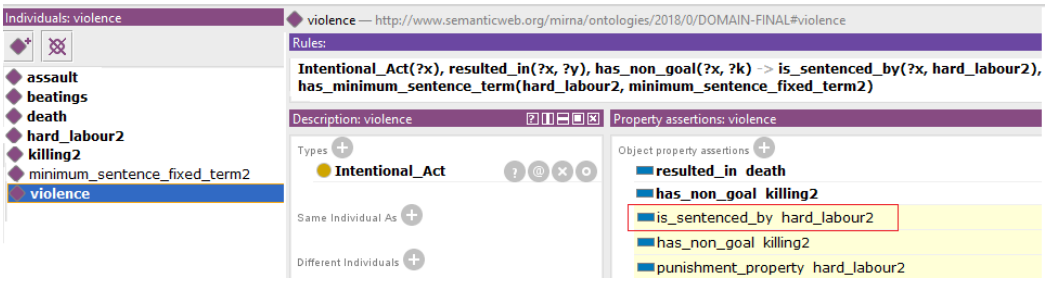


FIGURE 5.21: Article 550 using SWRL in Protégé.

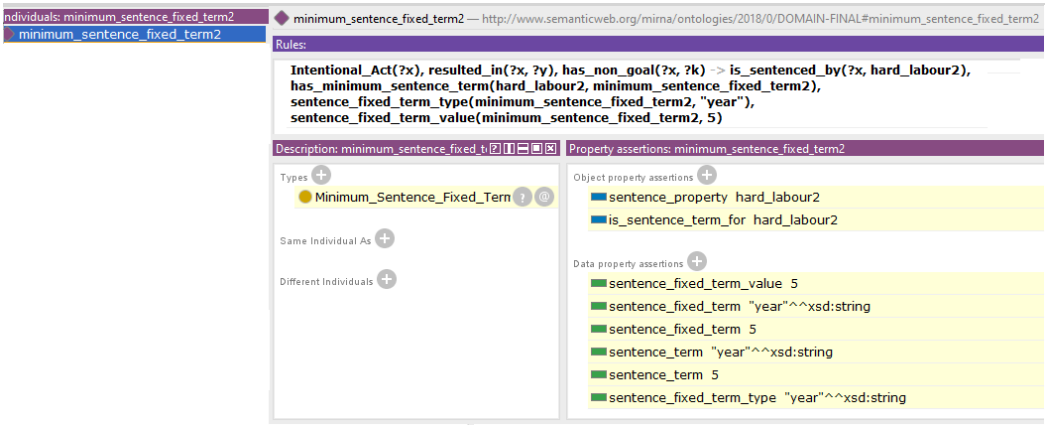


FIGURE 5.22: Article 550 using SWRL in Protégé.

As shown in figure 5.21, the execution of the given rule results in inferring the object property `is_sentenced_by` with the individual `hard_labour2` which is the head of the rule of the article 550.

5.4 Similar Works

In this section, we will overview briefly the related works concerning modeling rules using SWRL for ontology-based models.

Modeling Rules using SWRL for Ontology-Based Models In the literature, several works in different fields about modeling rules using SWRL for ontology-based models are found. Zhao (Zhao et al., 2008) provides a methodology to translate EXPRESS-driven models to OWL and SWRL. In other works, SWRL provides association rules, and allows to "associate instances to new classes and to create properties between instances", such as the works of Yang (Yang et al., 2008) and Dong (Dong et al., 2011) for the product configuration case. Elenius (Elenius et al., 2009) automates military events analysis with a SWRL-based reasoning. In the Healthcare field, Beimel (Beimel et al., 2011) have used OWL-DL to provide a model of the hospital organization, and control policies are expressed as implication rules, using SWRL, on classes and properties from the OW-DL part. Lezcano (Lezcano et al., 2011) have proposed the integrating of formal representations(ontologies) using OWL with rules expressed using SWRL providing an approach to apply the SWRL rules to concrete instances of clinical data. In another context, Rossello-Busquet (Rossello-Busquet et al., 2011) achieves automated actions with a SWRL-based model involving energy management. Finally, in the work of Fortineau (Fortineau et al., 2012), SWRL is used as a rule language for ontology-based models in the power plant design.

5.5 Conclusion

After building the legal domain ontology CriMOnto in chapter 4, by applying the first phase of the proposed ontology-based approach for modeling legal norms, this chapter addressed its employment for modeling the procedural aspect of these norms by applying the second phase of the approach. The most relevant approaches for modeling legal norms in the literature are surveyed such as (Palmirani et al., 2009; Palmirani et al., 2012), (Francesconi, 2010; Francesconi, 2011) and (Wyner et al., 2013). Then, the ontology-based approach for modeling and formalizing the procedural aspect of the legal norms is discussed. This approach is grounded on the integration of the developed ontology CriMOnto, which represents the modeling of the content of the legal norms, with a logic-based rule language. For this purpose, the domain of integration of ontologies and rules is outlined including the available approaches as well as the existent rule interchange languages needed for the formalization process such as RuleML, SBVR, SWRL, RIF, LKIF and LegalRuleML. Furthermore, a selection process for the most convenient language is made. SWRL is selected for several reasons discussed in section 5.2.2.1. Moreover, the rule reasoners for SWRL are introduced. Additionally, a case study, that employed CriMOnto and SWRL for modeling and formalizing an excerpt of the legal norms of the Lebanese criminal code, is presented. The logic rules are formalized in Protégé and inferred using the Pellet reasoner. Finally, we have reviewed some related works in the literature concerning the integration of ontologies and rules and the modeling of

rules using SWRL. The list of the formalized rules as well as CriMOnto will be used for building a rule-based legal reasoning model for a legal knowledge-based systems which will be presented and discussed in chapter 6.

Chapter 6

CORBS: Rule-Based System Grounded on CriMOnto

Contents

6.1 Overview

In this chapter, we discuss the building of a legal rule-based decision support system, named CORBS that performs rule-based reasoning for the criminal domain. CORBS is grounded on CriMOnto, the criminal domain ontology developed using a modular middle-out collaborative approach (see chapters 3 and 4), and a set of logic rules formalized in SWRL rule language (see chapter 5). The domain ontology and the rules are integrated together to compose the legal reasoning model of CORBS.

In section 6.2, we discuss the proposed rule-based system CORBS, the building approach, the model reasoning and the implementation are analysed as well. The similar works are presented in section 6.3 and section 6.4 concludes the chapter.

6.2 CORBS

In an environment of growing complexity of regulations, automated support for reasoning with regulations is becoming increasingly necessary. For this reason, we propose a legal decision support system that tends to help users in the legal domain to solve legal problems. The proposed system is an ontology-driven legal rule-based system named CORBS targeted mainly to validate the developed legal domain ontology CriMOnto (discussed in chapter 4). CriMOnto is constructed by applying the modular middle-out approach MIROCL (see chapter 3). Therefore, CORBS aims to embody the rules of the criminal domain, specifically the Lebanese criminal code as domain application, that are constructed based on CriMOnto (see chapter 5), in a prototype expert system. Thus, the system is grounded on the integration of a legal domain ontology and set of logical rules formalized using the logic rule language SWRL (see Figure 6.1).

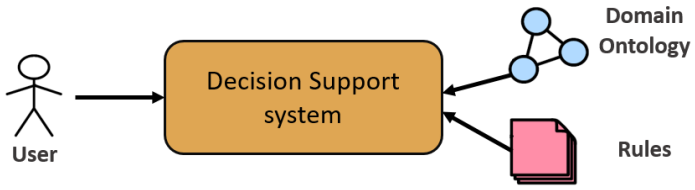


FIGURE 6.1: Architecture of CORBS combining ontologies with rules.

For developing such system, there is a need for an approach that defines its reasoning model which will be discussed in the following sections.

6.2.1 Hybrid Approach for Building CORBS

In chapter 2 section 2.3.3, we have discussed the existent approaches in the literature for building legal knowledge-based systems which are mainly: case-based, rule-based and model-based where each approach is based on a specific model of reasoning. The case-based and rule-based approaches are interested mainly in capturing the inferential aspects of legal knowledge more than in expressing the conceptual components and dependencies among kinds of knowledge. For building CORBS, a hybrid approach combining model-based and rule-based approaches is proposed.

- The rule-based approach depends on a rule-based reasoning model where the knowledge of the domain is represented in the form of productive rules in order to solve given problems.
- The model-based approach focuses mainly on two phases: modeling of the domain and the realization of the computational system. Additionally, this approach tends to separate the domain knowledge and the problem-solving knowledge. This promotes flexibility and transparency because the knowledge base can then be manipulated and examined as any other data structures (Buchanan et al., 1983).

  Therefore, this separation implies two main advantages:

  - The domain models can be developed independently of specific tasks and problems, and can therefore have a high degree of reusability;
  - The general problem solving models can also be used and adapted to domain-specific tasks, generating relatively domain-independent tasks.

  Therefore, for the development of models, two theories are maintained: a theory of domain reasoning and a theory of domain knowledge. The first specifies the specific characteristics of reasoning in the domain. This is made by specifying common reasoning modes and structures for typical tasks in the domain. The second explains how the domain is organized by containing an ontology of the domain.

Therefore, the propose hybrid is composed of two main phases: knowledge base construction phase and rule-based reasoning phase.

- Knowledge base construction phase: represents the construction of the system's model that is represented by the legal domain ontology CriMOnto (see chapter 4) and the legal formalized rules (see chapter 5). In this phase, the knowledge of the task area is separated as much as possible from the procedures that manipulate it. Thus, the building of CriMOnto is independent from the formalization process of the legal rules. Therefore, the model of the domain is represented by the criminal domain ontology CriMOnto and the

problem-solving methods are the legal rules formalized using SWRL based on CriMOnto.

- Rule-based reasoning phase: represents the reasoning strategy of the system based on the developed knowledge base.

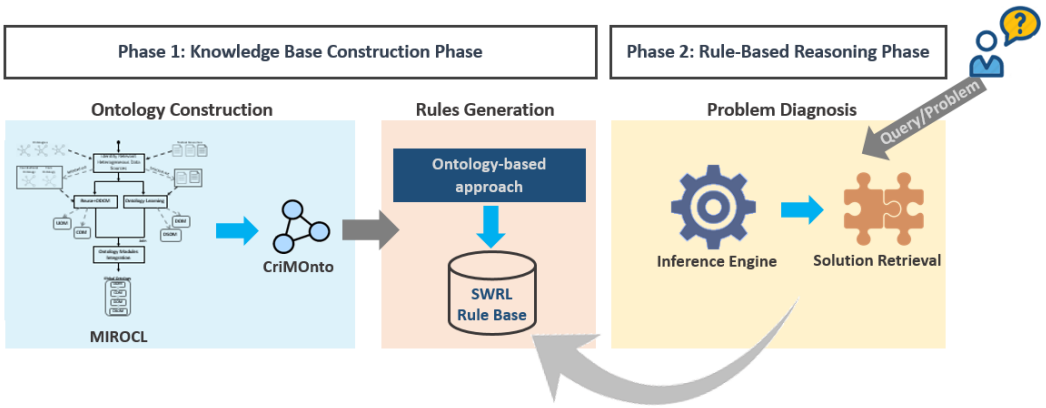


FIGURE 6.2: Hybrid approach for building CORBS.

Based on this approach, we will define the reasoning model of CORBS in the following section.

6.2.2 Reasoning Model of CORBS

Legal knowledge-based systems are built on the assumption that the user will provide the current facts to which the legal expert will apply the law (Susskind, 1986). In this context, McCarty, father of AI and law, claims that a LKBS must be able to represent the facts that involve all the complexities of daily life (human actions, beliefs, intentions, motivations, etc.) and the law that consists of a system of "concepts" and "rules" (Popple, 1996; McCarty, 1983). Most of the LKBS attempt to develop and implement complex models of legal reasoning (Popple, 1993). However, a LKBS need to be based upon a simple, but expressive, model in order to produce useful reasoning or decision. The power and utility of such systems rely on two main parts: a significant domain knowledge base and an intelligent reasoning module, or inference engine (Corsar et al., 2008).

From this perspective, and based on the hybrid approach proposed in section 6.2.1, the reasoning model of CORBS is composed of three main components (see Figure 6.3): user interface, knowledge base and inference engine.

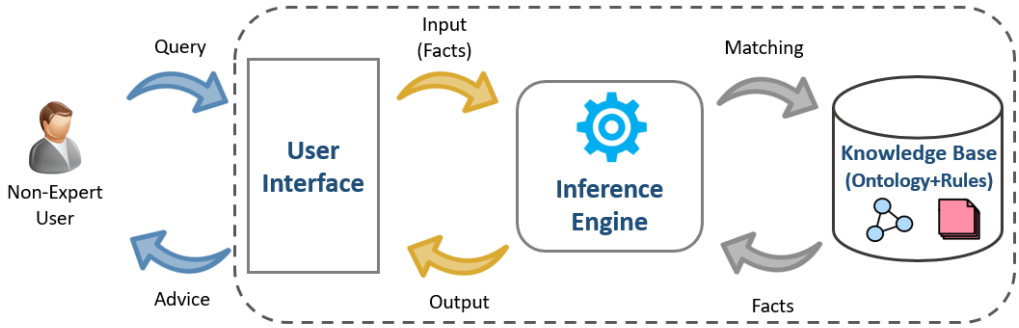


FIGURE 6.3: Reasoning Model of CORBS.

6.2.2.1 User Interface

The user interface is the means of communication between a user seeking a solution to the problem and an expert system. The communication should be as meaningful and friendly as possible (Negnevitsky, 2002). In CORBS, the user interface is an ontology-based interface. Through the interface, the user can provide the system with facts and queries that will be processed by the inference engine for reasoning purposes.

6.2.2.2 Knowledge Base

Generally, the knowledge base contains the domain knowledge useful for problem solving. In a rule-based expert system, the knowledge is represented as a set of rules. Each rule specifies a relation, recommendation, directive, strategy or heuristic and has the `IF (condition) THEN (action)` structure. When the condition part of a rule is satisfied, the rule is said to fire and the action part is executed (Negnevitsky, 2002).

In CORBS, we have defined an ontology-based approach (see chapter 5) for building these rules grounded on a legal domain ontology (CriMOnto). Thus, the knowledge base of CORBS is composed of two main categories of knowledge: ontological and problem-solving.

Ontological Knowledge The ontological knowledge tends to model in a clear and unambiguous way the application domain of the system. This knowledge is represented by the criminal domain ontology CriMOnto, developed earlier and discussed in chapter 4. CriMOnto is a modular ontology developed using a middle-out collaborative approach (see chapter 3) from heterogeneous sources, such as reusing existent validated upper ontologies and textual resources representing the domain (the Lebanese criminal code).

CriMOnto is composed of four independently developed ontology modules that are themselves ontologies (see Figure 6.4): Upper, Core, Domain and Domain-Specific. They are located on different levels from the most abstract (Upper) to the most specific (Domain-Specific).

- The Upper module contains the most general concepts such as `Mode`, `Phase`, `Role`, `Action`, `Event`,... This module includes 83 classes, 159 hierarchies and 60 relations.
- The Core module consists of legal core concepts such as `Legal_Role`, `Legal_Event`, `Legal_Action`, `Act_of_Law`, `Medium`,... This module contains 44 classes, 82 subclasses relations and 19 semantic relations.
- The Domain module represents the criminal domain concept such as `Offender`, `Perpetrator`, `Criminal_Action`, `Offence`, `Terrorism`,...This module contains 127 classes, 226 subclasses relations and 67 semantic relations.
- Finally, the Domain-Specific module that contains the instances of the DOMAIN module. These instances are related closely to the Lebanese criminal domain.

The modules are integrated together to compose the general ontology CriMOnto. The concepts of the different modules are interlinked using `subClassOf` and `instanceOf` relations. Meanwhile, the different modules can still be identified in CriMOnto.

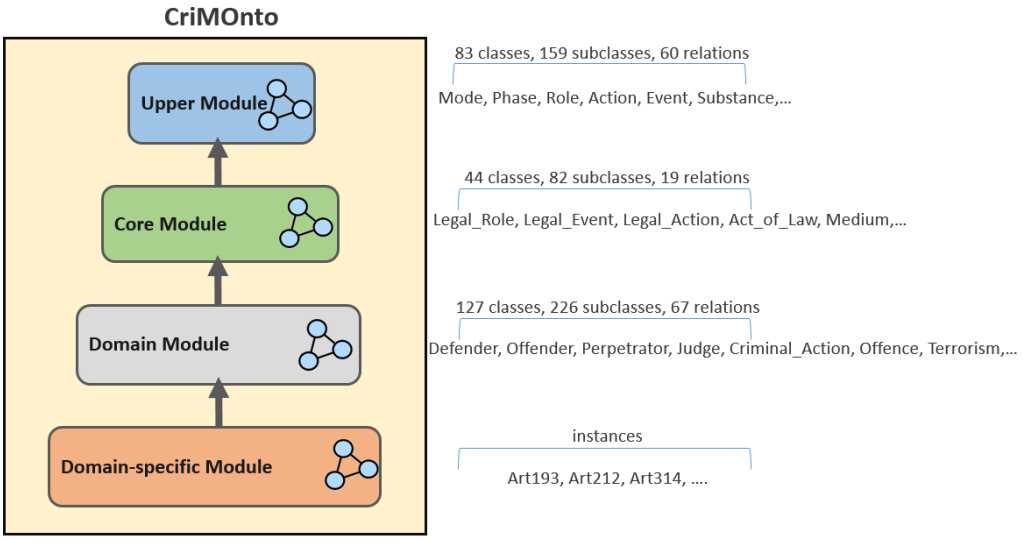


FIGURE 6.4: The criminal domain ontology CriMOnto.

Problem-solving Knowledge Problem-solving knowledge is about how to use the domain knowledge to achieve various goals. This knowledge is often in the form of a problem solving methods (PSM) which are the set of rules in a rule-based LKBS.

In CORBS, PSMs are the formalized legal rules discussed earlier in chapter 5. These rules represent the procedural aspect of the legal norms of the Lebanese criminal code. They are developed based on CriMOnto by mentioning vocabulary specified by this ontology and formalized using SWRL. Moreover, they are independent, self-contained chunks of knowledge where each rule can be changed or updated without requiring the modification of other rules or affecting the entire system. Furthermore, the performance of the system is affected by their reliability.

In the following, an excerpt of the formalized legal rules is presented.

Article 213: `Accomplice(?x)` $\wedge$ `is_punishable_by(?y,?z)` $\wedge$ `commit(?x,?y)` $\Longrightarrow$ `is_liable_to_punished_by(?x,?z).`

Article 196: `Intentional_Act(?x)` $\wedge$ `motivated_to(?m, ?x)` $\wedge$ `motive_type(?m, "Political_Motive")` $\Longrightarrow$ `Political_Offence(?x).`

Article 547: `Intentional_Homicide(killing)` $\wedge$ `committed_towards(killing, ?y)` $\wedge$ `committed_by(killing, ?x)` $\Longrightarrow$ `is_punished_by(?x, hard-labour)` $\wedge$ `imposed_for_maximum(hard-labour, max_d_2)` $\wedge$ `imposed_for_minimum (hard-labour, min_d_2)` $\wedge$ `term_value(max_d_2, 20)` $\wedge$ `term_value(min_d_2, 15)` $\wedge$ `term_type(max_d_2, "years")` $\wedge$ `term_type(min_d_2, "years").`

6.2.2.3 Inference Engine

The inference engine consists of algorithms for manipulating the knowledge represented in the knowledge base. It applies the logic contained in the knowledge base to the information input by the user and outputs advice. The inference engine carries out the reasoning whereby the expert system reaches a solution. It links the rules given in the knowledge base with the facts provided in the database (Negnevitsky, 2002).

In CORBS, we define a reasoning process for solving legal problems based on forward chaining systems. The legal rules, formalized in SWRL, are used by the inference engine or OWL reasoner, such as Pellet and Hermit (Glimm et al., 2014), in order to derive new knowledge. Note that HermiT is one of the few reasoners ported so far to OWL API 5. The reasoning process consists of the following sub-steps:

1. Select the applicable rules.
2. Match the facts, that are defined as the individuals of the OWL classes in our ontology, with the condition of the rules to determine which rules should be applied and selects the most appropriate rule.
3. Deduce new facts from the existing facts.

4. The selected rule is fired by the inference engine and the action associated with it is executed.
5. The inference engine repeats this reasoning process in a loop through all the rules and facts until no more conclusion can be reached or the termination conditions are satisfied.

Therefore, in CORBS, the inference engine drives the legal reasoning by retrieving the facts (input) submitted to the system and matching them with the rule base to identify the rule, or rules that satisfy the input.

6.2.3 Tasks of CORBS

Generally, CORBS is defined as a legal decision support system that aims to help the users to access regulations semantically, perform querying and rule-based reasoning. Therefore, the common tasks of CORBS are:

- Semantic search and querying: that would ensures the provision of useful and relevant information to the user where the user can search for results by contextual meaning of input query instead of keyword matching.
- Rule-based reasoning: that enables the reasoning over input facts such as determining the legal punishments for crime perpetrators.

6.2.4 Implementation of CORBS

For the implementation of CORBS, Eclipse[1], as a programming environment, and Java, as a programming language, are used. Moreover, for developing an ontology-based application, such as CORBS, there is a need for a Java-based library such as Jena and OWL API. In the following paragraphs, we discuss these API and determine the most relevant for our work.

Jena Framework Jena[2] is a an open source Java framework for building Semantic Web and linked data applications. It provides a programmatic environment for RDF, RDFS, OWL, SPARQL and includes a rule-based inference engine. Jena has an API to extract data from and write to RDF graphs and OWL ontologies. In Jena, all state information provided by a collection of RDF triples is contained in a data structure called a *Model*. The model denotes an RDF graph in Jena and can be created by using data from URLs, files, databases or by combining different sources. Jena includes support for a variety of reasoners through the inference API.

[1] http://www.eclipse.org/
[2] https://jena.apache.org/, last accessed 11 February 2018

Additionally, a memory and persistent storage for storing large number of RDF triples is provided in Jena. SPARQL can be used to query the model.

Concerning the ontology support, Jena provides an *Ontology* API that presents the graph using concepts from OWL and RDFS (Carroll et al., 2004) (see Figure 6.5).

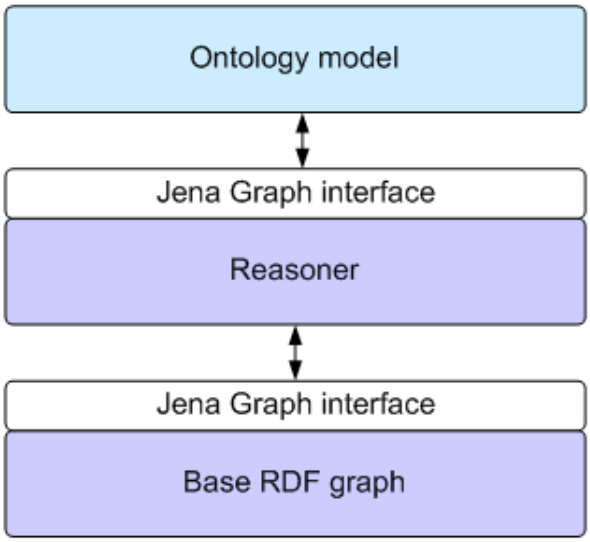


FIGURE 6.5: Handling ontologies in Jena.

OWL API OWL API[3] is a Java interface and implementation for working with OWL ontologies. The OWL API is closely aligned with the OWL 2 structural specification. It supports parsing and rendering in the syntaxes defined in the W3C specification (Functional Syntax, RDF/XML, OWL/XML and the Manchester OWL Syntax); manipulation of ontological structures; and the use of reasoning engines. The reference implementation of the OWL API, written in Java, includes validators for the various OWL 2 profiles - OWL 2 QL, OWL 2 EL and OWL 2 RL. The OWL API has widespread usage in a variety of tools and applications (Horridge et al., 2011).

The OWL API includes the following components:

- An API for OWL 2 and an efficient in-memory reference implementation.
- RDF/XML parser and writer.
- OWL/XML parser and writer.
- OWL Functional Syntax parser and writer.
- Turtle parser and writer.
- KRSS parser.

[3]https://owlcs.github.io/owlapi/, last accessed 11 February 2018

- OBO Flat file format parser.
- Support for integration with reasoners such as Pellet and FaCT++.
- Support for black-box debugging.

In OWL API, an ontology is simply viewed as a set of axioms and annotations as depicted in Figure 6.6. The names and hierarchies of interfaces for entities, class expressions and axioms in the OWL API correspond closely to the structural specification, relating the high level OWL 2 specification directly to the design of the OWL API. The OWL API supports loading and saving ontologies is a variety of syntaxes. However, none of the model interfaces in the OWL API reflect, or are biased to any particular concrete syntax or model. For example, unlike other APIs such as Jena, the representation of class expressions and axioms is not at the level of RDF triples (Horridge et al., 2011).

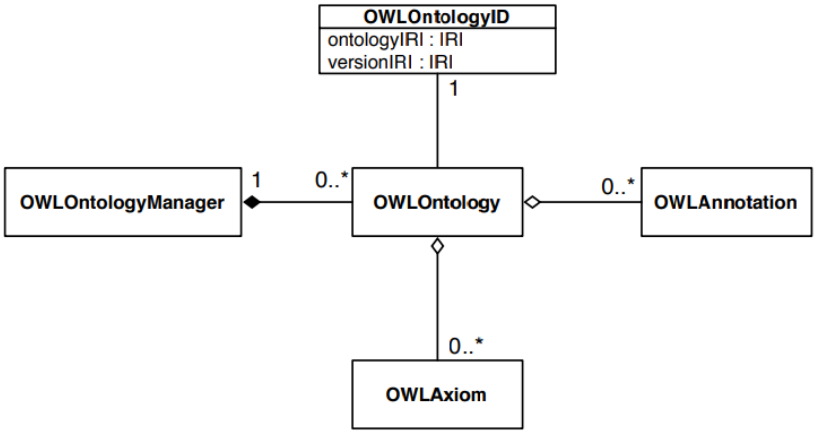


FIGURE 6.6: A UML diagram showing the management of ontologies in the OWL API (Horridge et al., 2011).

OWL API	Jena API
Java API and reference implementation	Java framework
Free and open-source	Free and open-source
OWL-centric	RDF-centric
OWL2 support	OWL2 not supported
Creating, manipulating and serializing OWL2 ontologies	API to extract data from and write to RDF graphs
Reasoner interfaces: FaCT++, HermiT, Pellet, and Racer	Pellet
Support importing ontologies (Modularity of ontologies)	Importing ontologies not supported
Reasoning support for SWRL	SWRL not supported
supports SPARQL-DL querying	supports SPARQL querying

TABLE 6.1: A comparison of OWL API and Jena framework.

Based on the aforementioned, the most relevant API for our work is OWL API (version 5) since it is OWL-centric and dedicated for manipulating ontologies mainly. Additionally, OWL2 is supported in this API. Moreover, two main important reasons for this selection are: (1) the support of modular ontologies which is the case of our developed ontology CriMOnto; (2) the support pf SWRL rules using the rule engines Pellet and Hermit.

6.2.4.1 Loading CriMOnto

The implementation of CORBS is based mainly on loading the developed legal domain ontology CriMOnto. For this purpose, the selected API should handle this process specifically the modular aspect of CriMOnto.

In OWL API, `OWLOntologyManager` is the central class for managing ontologies. It handles creating, loading and saving ontologies, the application of changes such as annotations or axiom additions. Moreover, `OWLOntologyManager` can hold more than one ontology. The most important case where this becomes relevant in our work since we are dealing with multiple ontology modules in CriMOnto. We need to import and load four main modules (Upper, Core, Domain and Domain-specific) and to handle them as unified global ontology. In OWL, this dependency is made explicitly by adding an `owl:imports` statement. For example, in CriMOnto, Domain-specific module depends on Domain module which in turn depends on Core module that depends as well on Upper Module. In OWL API, the import process of the CriMOnto modules is made as mentioned in the following code:

```
OWLOntologyManager m=OWLManager.createOWLOntologyManager(); File file1
= new File("C:/Users/Mirna/workspace/misc/src/ontology/UPPER-FINAL.owl");
```

```
IRI iri1=
IRI.create("http://www.semanticweb.org/mirna/ontologies/2018/0/UPPER-FINAL");
m.getIRIMappers().add(new SimpleIRIMapper(iri3,
IRI.create("C:/Users/Mirna/workspace/misc/src/ontology/DOMAIN-FINAL.owl")));
OWLOntology onto1=m.loadOntologyFromOntologyDocument(file1);
```

When a local copy of one or more ontologies is used, an ontology IRI mapper can be used to provide a redirection mechanism. This means that ontologies can be loaded as if they were located on the Web.

6.2.4.2 Semantic Search and Queries Executing

In OWL API, the reasoners implement the `OWLReasoner` interface. Through the `OWLReasoner` interface, a number of interesting things can be done such as, compute ontology axioms and query for sub-classes, equivalent classes and instances.

```
OWLReasonerFactory rf = new ReasonerFactory();
OWLReasoner r = rf.createReasoner(onto2);

r.precomputeInferences(InferenceType.CLASS_HIERARCHY);
```

In figure 6.7, an example of querying for sub-classes is depicted. For example, for listing the subclasses of the class `Legal_Event`, which is located in the Core module of CriMOnto, the following code is applied:

```
OWLClass per =
df.getOWLClass(IRI.create("http://www.semanticweb.org/mirna/ontologies/2018/0/
CORE-FINAL#Legal_Event"));
NodeSet<OWLClass> subClses = r.getSubClasses(per, true);
Set<OWLClass> clses = subClses.getFlattened();
System.out.println(“Subclasses of Legal Event:  ”);
for(OWLClass cls :  clses) {  System.out.println(cls); }
```

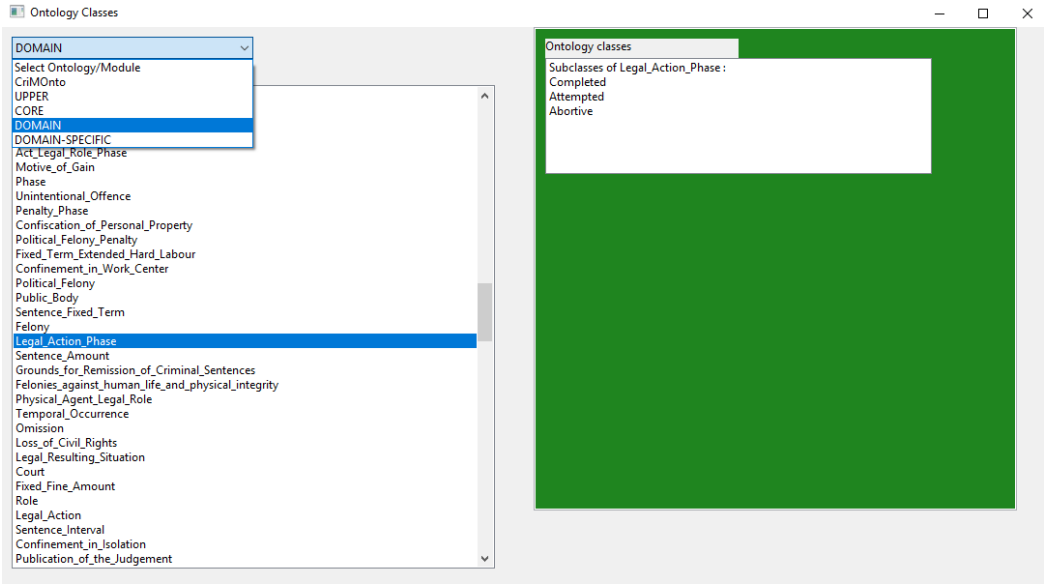


FIGURE 6.7: Example of querying ontologies (sub-classes).

Additionally, for querying CriMOnto, a query engine SPARQL-DL ia used. In the following paragraph, SPARQL-DL is presented as well as example of Java code.

SPARQL-DL API SPARQL-DL is a query language for OWL-DL ontologies(Sirin et al., 2007a). SPARQL-DL[4] query engine is settled on top of the OWL API. The library is fully aligned with the OWL 2 standard and adds a SPARQL-DL interface to every OWL API 3 reasoner (see Figure 6.8).

FIGURE 6.8: SPARQL-DL querying.

The SPARQL-DL query language is a distinct subset of SPARQL (a query language for RDF). The subset is tailored to ontology specific questions which typically come into focus when dealing with OWL. SPARQL-DL is a quite expressive language which particularly allows to mix TBox, RBox, and ABox queries.

SPARQL-DL supports two different types of queries: `ASK` and `SELECT`. An `ASK`-query returns a Boolean result whereas a `SELECT`-query returns all possible bindings of the provided variables.

[4]http://www.derivo.de/en/resources/sparql-dl-api/

`ASK`-queries:

```
ASK  [comma-separated list of atoms]
```

SELECT-queries:

```
SELECT [DISTINCT] [space-separated list of variables] [WHERE]  [comma-separated
list of atoms]
[OR WHERE  [comma-separated list of atoms] ]
```

As within SPARQL, the `DISTINCT` keyword removes automatically all redundant bindings within the result set.

SPARQL-DL Query Patterns
Class(a)
Property(a)
Individual(a)
Type(a, b)
PropertyValue(a, b, c)
EquivalentClass(a, b)
SubClassOf(a, b)
EquivalentProperty(a, b)
SubPropertyOf(a, b)
InverseOf(a, b)
ObjectProperty(a)
DataProperty(a)
Functional(a)
InverseFunctional(a)
Transitive(a)
Symmetric(a)
Reflexive(a)
Irreflexive(a)
SameAs(a, b)
DisjointWith(a, b)
DifferentFrom(a, b)
ComplementOf(a, b)
Annotation(a, b, c)
StrictSubClassOf(a, b)
DirectSubClassOf(a, b)
DirectType(a, b)
StrictSubPropertyOf(a, b)
DirectSubPropertyOf(a, b)

TABLE 6.2: Supported query patterns in SPARQL-DL.

In figure 6.9, an example of applying SPARQL-DL for querying CriMOnto is depicted. This example illustrates the selection of articles from the criminal code that contains the keyword `Perpetrator` or any other synonym.

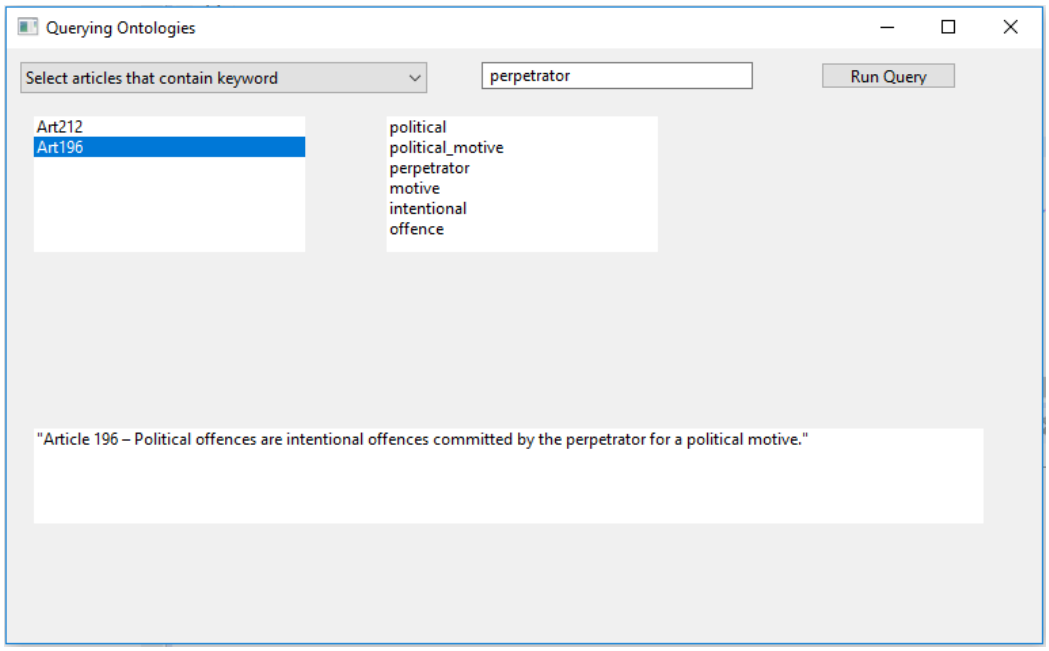


FIGURE 6.9: Example of querying CriMOnto using SPARQL-DL.

In Java, using SPARQL-DL, the following code is applied for a `SELECT` query:

```
String q= "PREFIX domain:
<http://www.semanticweb.org/mirna/ontologies/2018/0/DOMAIN-FINAL#>" +

"PREFIX core:

<http://www.semanticweb.org/mirna/ontologies/2018/0/CORE-FINAL#>" +

"PREFIX ds:

<http://www.semanticweb.org/mirna/ontologies/2018/0/DOMAIN-SPECFIC-FINAL#>"
+

"SELECT ?x WHERE PropertyValue(?x, domain:contains_key, ds:perpetrator)";

Query query = Query.create(q);

QueryResult result = engine.execute(query);
```

6.2.4.3 Rules Executing

For the executing of rules, there is a need to integrate an OWL reasoner that supports rule-based reasoning such as Pellet and HermiT. In the OWL API library, the main

interfaces for managing SWRL rules are `SWRLDataFactory`, `SWLAtom`, `SWRLRule` and `SWRLVariable`.

An example of listing the SWRL rules is presented in the following:

```
OWLDataFactory dataFactory = new OWLDataFactoryImpl();

Set<SWRLRule> rules = ontology.getAxioms(AxiomType.SWRL_RULE);

 SWRLRule parsedRule = rules.iterator().next();  System.out.println("rule:
"+rules);
```

6.3 Similar Works

6.3.1 COMUS: Context-Based Music Recommendation Ontology for Rule-Based Reasoning

COMUS (Context-based Music Recommendation) ontology consists of about 500 classes and instances, and 52 properties definitions created in Protégé (Rho et al., 2009). This ontology describes music related information about relationships and attributes that are associated with people, genre, mood, location, time, and situation events in a daily life. COMUS ontology is used to support ontology rue-based reasoning for recommending appropriate music to users. The reasoning can be brought in by specifying user-defined reasoning rules towards defining high level conceptual contexts such as "What music does the user want to listen to when he/she is stressed?" can be deduced from relevant low-level context (see Figure 6.10).

	Reasoning Rules
Wake up	(?u isLocatedIn Bedroom) ∧ (Datetime week MONDAY) ∧ (Datetime hour 7) ⇒ (?u event WAKE_UP)
Driving to work	(?u isLocatedIn Road) ∧ (?u isLocatedIn Car) ∧ (Datetime week MONDAY) ∧ (Datetime hour 8) ⇒ (?u event DRIVING_To_WORK)
Driving and Traffic Jam	(?u event DRIVING_TO_WORK) ∧ (?e TrafficJam) ⇒ (?u situation DRIVING_TO_WORK_TRAFFIC_JAM)
Situation goal	(?u situation DRIVING_TO_WORK_TRAFFIC_JAM) ∧ (?u likeMood) ⇒ (?s hasGoal ?goalMood)

FIGURE 6.10: User-defined context reasoning rules (Rho et al., 2009).

A prototype music recommendation system is developed based on COMUS and extended the music ontology to enable mood and situation reasoning in a music recommendation system (see Figure 6.11). The system provides various types of query interfaces to the users.

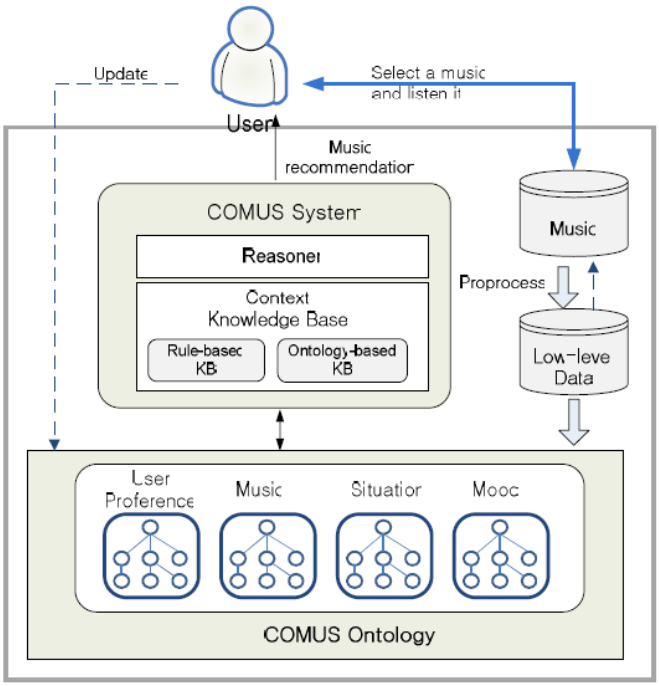


FIGURE 6.11: A prototype for COMUS (Rho et al., 2009).

6.3.2 Emotiono: Ontology for Rule-Based Reasoning for Emotion Recognition

Emotiono is an emotional and related ontology in the Affective Computing domain (Zhang et al., 2011). The Emotiono ontology defines the terms used to describe and represent emotional knowledge including basic concepts (different affective states, participants' EEG features and sampling rate) and the relationships that exist among them. Some concepts and some hierarchy of concepts in the Emotiono ontology with their subclasses include: Spatial_Parameter, Channel_Type and Electrode.

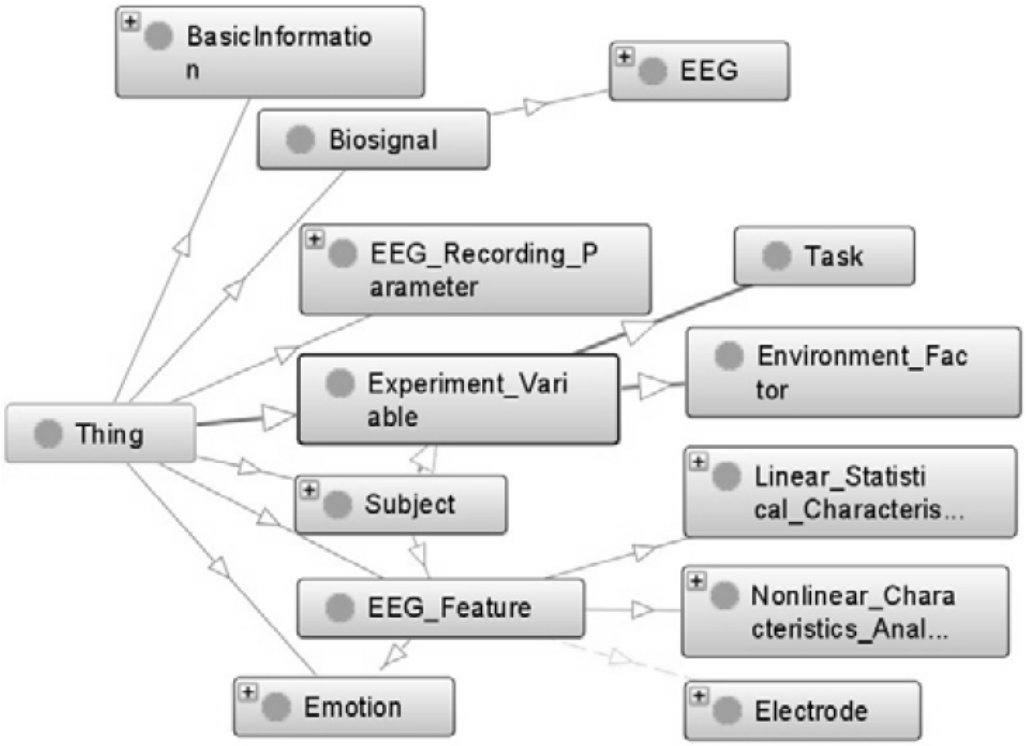


FIGURE 6.12: Excerpt of Emotiono Ontology (Zhang et al., 2011).

6.3.2.1 Rule-Based Reasoning in Emotiono

In Emotiono ontology, a user's desired emotional state is deduced from the ontology based on his situation (prevailing state), personal information, and his EEG features. In order to get the main relations between EEG features of a certain person and his affective states Generic rule reasoner, a Jena Reasoner engine is used; the reasoner consists of the reasoning engine and context-based engine. The context-based engine extracts the contexts of interrelation with input data for emotion recognition. Therefore, the Emotiono ontology relies on well-defined context definitions to arrive at the correct emotional state. When the reasoner receives the EEG signal data or user request, a context-based reasoning engine generates the query as rules to generate the correct results.

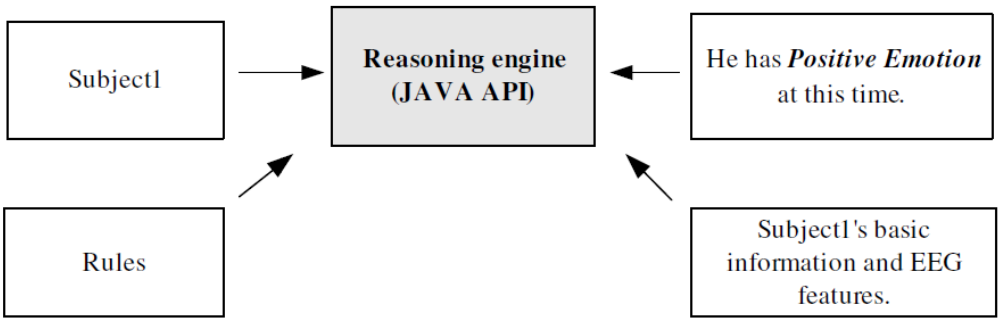


FIGURE 6.13: Emotiono Reasoning (Zhang et al., 2011).

6.4 Conclusion

In this chapter, the legal rule-based systems are discussed, their architecture, advantages and disadvantages. The most known methods of reasoning, forward and backward chaining, that exist for rule-based systems are presented. Furthermore, for building CORBS, the target rule-based decision support systems, a hybrid approach is proposed which is a combination of rule-based and model-based approaches. The reasoning model of CORBS, which is composed of three main components (user interface, knowledge base and inference engine), is analyzed. Moreover, the implementation of CORBS is introduced using the Java-based OWL API where the main tasks are discussed such as loading CriMOnto, semantic search and querying and rules executing. Finally, some recent similar works, that consider the building of ontologies for rule-based reasoning purposes, are explored.

Chapter 7

Conclusion

Contents

7.1 Thesis Overview

This thesis addresses the problem of building well-founded domain ontologies for reasoning purposes. Actually, the field of ontology engineering suffers from several limitations in the fields of knowledge acquisition and modeling and knowledge sharing and reasoning. Most of the existent approaches in the literature focus on core ontology development or build ontologies from scratch leading to non well-founded ontologies. Moreover, the approaches neglect the participatory aspect of building ontologies. Additionally, most of the developed ontologies are not built for reasoning purposes.

Therefore, aiming to tackle the addressed limitations, the thesis had four main contributions. Firstly, we provided a novel modular middle-out collaborative approach named MIROCL for developing well-founded domain ontologies. This approach supports the following features: (1) Collaborative ontology building process handling heterogeneous data sources; (2) Ontology modularization; (3) Ontology-driven conceptual modeling process; (4) Ontology learning process; (5) Ontology integration. Thus, MIROCL focused on building well-founded domain ontologies collaboratively from different heterogeneous sources such as reusing predefined existent validated ontologies and textual resources as well as simplifying the ontology development process by applying the ontology modularization and learning processes. MIROCL was presented and discussed in chapter 3. Secondly, we applied the approach MIROCL in the legal domain, specifically the Lebanese criminal domain, for building well-founded criminal domain ontology named CriMOnto. The main purpose of CriMOnto was modeling the content of the legal norms of the Lebanese criminal code as a domain application. CriMOnto is targeted to be used for rule-based reasoning purposes. The building process of CriMOnto and the results were presented and evaluated in chapter 4. Thirdly, we proposed an ontology-based approach for modeling and formalizing the procedural aspect of the legal norms of the Lebanese criminal code based on the developed ontology CriMOnto. The application of the approach and the results were analyzed in chapter 5. Finally, we put together all these works for building a legal decision support system named CORBS grounded on the integration of the legal domain ontology CriMOnto and the formalized legal rules. The main purpose of CORBS was to validate the legal domain ontology by emerging it in an ontology-based application. CORBS is developed using the Java-based library OWL API.

7.2 Future Directions

In this thesis, the novel modular middle-out approach MIROCL have been proposed, described and validated for building well-founded domain ontologies for reasoning

purposes in the legal domain. Meanwhile, we found several future directions related to our work which can be described for improving the current research. We will discuss these directions in the following:

- MIROCL is a middle-out approach that tend to solve several limitations in the field of ontology development by combining different ontology building support processes such as ontology modularization, integration, ontology-driven conceptual modeling and ontology learning. Actually, the results obtained, using the ontology learning tool Text2Onto for extracting the domain and domain-specific ontologies from textual resources, were described as lightweight and required the intervention of domain experts and ontology engineers for correcting, pruning and enriching the results. This process is described as resource intensive and time consuming. To allow for a more enhanced results obtained by applying the ontology learning process, and for minimizing the efforts exhausted for the correction phase, we suggest to develop an ontology learning tool that have the capability to extract semi-automatically semantic objects (such as hierarchies, semantic relations, axioms,..) from texts. These objects can be classified as heavyweight, rich and containing the minimum possible of errors.

- For developing our legal domain ontology, we have reused the legal core ontology LKIF-Core. For future works, we suggest the application of UFO-L which is a recent legal core ontology developed based on the unified foundational ontology UFO.

- Concerning the domain application and the language, we have applied the proposed approach MIROCL on the Lebanese criminal domain specifically the Lebanese criminal code as textual resources. We suggest its application on other legal domains such as civil code for example. Regarding the application language, we have used the English language in this research. For future works, we need to verify the application of MIROCL on other languages such as the Arabic or French.

- Apart from applications on the legal domain, the proposed approach MIROCL is indeed general and can be applied to other domains such as medicine for instance. In fact, the experience in the medical domain concerning the ontology development by applying MIROCL could offer a starting point for comparative research for building well-founded domain ontologies.

- Regarding the reasoning part, the developed legal domain ontology CriMOnto is used for modeling the procedural aspect of the legal norms by applying an homogeneous monotonic ontology-based approach. Therefore, the rule language SWRL is used for formalizing the rules. Meanwhile, SWRL suffers from some limitations which cause the difficulty of formalizing the complex legal norms of the criminal code. For this reason, we recommend for future

works to apply a non-monotonic approach where rules can be expressed using Logic Programming formalisms, such as ASP (Answer Set Programming), which is a family of non-monotonic knowledge representation.

- For the implementation part, in this work we have developed a desktop ontology-based application using the OWL API Java-based library. We can improve it by developing a web-based application using the GWT library.

Bibliography

Abbes, S. et al. (2012). "Characterizing modular ontologies". In: *7th International Conference on Formal Ontologies in Information Systems (FOIS)*, pp. 13–25.

Abburu, S. (2012). "A Survey on Ontology Reasoners and Comparison". In: *International Journal of Computer Applications* 57.17, pp. 33–39.

Agnoloni, T. et al. (2007). "Building an ontological support for multilingual legislative drafting". In: *Proceedings of the 2007 conference on Legal Knowledge and Information Systems: JURIX 2007: The Twentieth Annual Conference*. Ed. by A. Lodder and L. Mommers. Vol. 165. Frontiers in Artificial Intelligence and Applications. IOS Press, pp. 9–18.

Agnoloni, T. et al. (2009). "A two-level knowledge approach to support multilingual legislative drafting". In: *the 2009 conference on Law, Ontologies and the Semantic Web: Channelling the Legal Information Flood*. Ed. by J. Breuker et al. Vol. 188. Frontiers in Artificial Intelligence and Applications. IOS Press Amsterdam, The Netherlands. Chap. A two-level knowledge approach to support multilingual legislative drafting, pp. 177–198.

Agnoloni, T. and Francesconi, E. (2011). "Modelling Semantic Profiles in Legislative Documents for Enhanced Norm Accessibility". In: *ICAIL 11 Proceedings of the 13th International Conference on Artificial Intelligence and Law*, pp. 111–115.

Albuquerque, A. (2013). "Ontological Foudations for Conceptual Modeling Datatypes". PhD thesis. Federal University of Espirito Santo.

Angele, J. et al. (1998). "Developing Knowledge-Based Systems with MIKE". In: *Automated Software Engineering* 5.4, pp. 389–418.

Antoniou, G. et al. (2005a). *Combining Rules and Ontologies: A survey*. Tech. rep. IST506779/Linkoping/I3-D3/D/PU/a1. Linkoping University.

Antoniou, G. et al. (2005b). *Combining Rules and Ontologies - A survey*. Deliverables I3-D3, REWERSE. URL: http://rewerse.net/deliverables/m12/i3-d3.pdf.

Arpirez, J. et al. (1998). "(ONTO)2Agent: An ontology-based WWW broker to select ontologies". In: *Workshop on Application of Ontologies and PSMs*, pp. 16–24.

Arpirez, J. et al. (2000). "Knowledge and Information Systems". In: vol. 2. 4. Springer. Chap. Reference Ontology and (ONTO)2 Agent: The Ontology Yellow Pages, pp. 387–412.

Arpirez, J. et al. (2003). "WebODE in a nutshell." In: *AI Magazine* 24.3, pp. 37–47.

Asaro, C. et al. (2003). "A Domain Ontology: Italian Crime Ontology". In: *ICAIL 2003 Workshop on Legal Ontologies & Web based legal information management*.

Ashley, K. (2009). "Ontological Requirements for Anological Teleological, and Hypothetical Legal Reasoning". In: *Proceedings of the 12th International Conference on AI and Law (ICAIL)*.

Ashley, K. (2011). "Approaches to Legal Ontologies". In: ed. by G. Sartor et al. Vol. 1. Law, Governance and Technology Series. Springer, Dordrecht. Chap. The Case-Based Reasoning Approach: Ontologies for Analogical Legal Argument, pp. 99–115.

Ashley, K. et al. (2001). "Legal Reasoning and Artificial Intelligence: How Computers "Think" Like Lawyers". In: *The University of Chicago Law School Roundtable* 8.1.

Athan, T. et al. (2013a). "OASIS LegalRuleML". In: *The Fourteenth International Conference on Artificial Intelligence and Law. ICAIL13*. New York: ACM Press, pp. 3–12.

Athan, T. et al. (2013b). "Theory, Practice, and Applications of Rules on the Web. RuleML 2013". In: ed. by L. Morgenstern et al. Vol. 8035. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg. Chap. LegalRuleML: From Metamodel to Use Cases, pp. 13–18.

Athan, T. et al. (2014). "Legal interpretations in legalruleml". In: *Proceedings of the Semantic Web for the Law and Second Jurix Doctoral Consortium Workshops (SW4LAW+JURIX-DC 2014)*. Vol. 1296. CEUR-WS.org.

Athan, T. et al. (2015). "Reasoning Web. Web Logic Rules. Reasoning Web 2015". In: ed. by W. Faber and A. Paschke. Vol. 9203. Lecture Notes in Computer Science. Springer, Cham. Chap. LegalRuleML: Design Principles and Foundations, pp. 151–188.

Aussenac-Gilles, N., Biebow, B., and Szulman, S. (2000). "Revisiting ontology design: a methodology based on corpus analysis". In: *12th International Conference in Knowledge Engineering and Knowledge Management (EKAW 00)*. Vol. 1937. Lecture Notes in Artificial Intelligence. Springer, Berlin, pp. 172–188.

Baader, F. et al., eds. (2002). *The Description Logic Handbook*. Cambridge University Press.

Baader, F. et al., eds. (2003a). *Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press.

Baader, F., Horrocks, I., and Sattler, U. (2005). "Mechanizing Mathematical Reasoning". In: ed. by D. Hutter and W. Stephan. Vol. 2605. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg. Chap. Description Logics as Ontology Languages for the Semantic Web, pp. 228–248.

Baader, F. and Nutt, W. (2003b). "The description logic handbook". In: Cambridge University Press New York, NY, USA. Chap. Basic description logics, pp. 43–95.

Baader, F. and Nutt, W. (2003c). "The description logic handbook". In: Cambridge University Press New York, NY, USA. Chap. Basic Description Logics Cambridge University, pp. 41–95.

Bachimont, B., Isaac, A., and Troncy, R. (2002). "Semantic commitment for designing ontologies: a proposal". In: *EKAW 2002*. Ed. by A. Gomez-Perez and V. Benjamins. Vol. 2473. LNAI. Springer-Verlag Berlin Heidelberg, pp. 114–121.

Bao, J. and Honavar, V. (2004a). "Collaborative Ontology Building with Wiki@nt". In: *3rd Intl. Workshop on Evaluation of Ontology Based Tools at Intl. Semantic Web Conference*.

Bao, J. and Honavar, V. (2004b). *Ontology Language Extensions to Support Localized Semantics, Modular Reasoning, and Collaborative Ontology Design and Ontology Reuse*. Tech. rep. 243.

Barcelos, P. et al. (2013). "An Automated Transformation from OntoUML to OWL and SWRL". In: *Ontobras*. Vol. 1041, pp. 130–141.

Baroni, M. and Bisi, S. (2004). "Using cooccurrence statistics & the web to discover synonyms in a technical language". In: *Proceedings of the 4th International Conference on Language Resources and Evaluation*. Vol. 5, pp. 1725–1728.

Beimel, D. and Peleg, M. (2011). "Using OWL and SWRL to represent and reason with situation-based access control policies". In: *Journal of Data and Knowledge Engineering* 70, pp. 596–615.

Ben Mustapha, N. et al. (2013). "A dynamic composition of ontology modules approach: application to web query reformulation". In: *International Journal of Metadata, Semantics and Ontologies* 8.4, pp. 309–321.

Bench-Capon, T. et al. (2012). "A history of AI and Law in 50 papers: 25 years of the international conference on AI and Law". In: *Artificial Intelligence and Law* 20.3, pp. 215–319.

Bench-Capon, T. and Coenen, F. (1992). "Isomorphism and Legal Knowledge Based Systems". In: *Artificial Intelligence and Law* 1.1, pp. 65–86.

Bench-Capon, T. and Gordon, T. (2009). "Isomorphism and argumentation". In: *ICAIL 2009*, pp. 11–20.

Bench-Capon, T. and Visser, P. (1997). "Ontologies in legal information systems; the need for explicit specifications of domain conceptualizations". In: *Sixth International Conference on Artificial Intelligence and Law*, pp. 132–141.

Bench-Capon, T. et al. (1987). "Logic programming for large scale applications in law: A formalisation of supplementary benefit legislation". In: *First International Conference on Artificial Intelligence and Law*. ACM Press, pp. 190–198.

Benevides, A. and Guizzardi, G. (2009a). "A model-based tool for conceptual modeling and domain ontology engineering in OntoUML". In: *ICEIS 2009: Enterprise Information Systems*. Vol. 24. LNBIP, pp. 528–538.

Benevides, A. et al. (2009b). "Assessing Modal Aspects of OntoUML Conceptual Models in Alloy". In: *International Workshop on Evolving Theories of Conceptual Modeling (ETheCoM 2009)*. Gramado, Brazil.

Benevides, A. et al. (2011). "Validating modal aspects of OntoUML conceptual models using automatically generated visual world structures". In: *J. Univers. Comput. Sci.* 16, pp. 2904–2933.

Benjamins, V. et al. (2003). "Ontologies of Professional Legal Knowledge as the Basis for Intelligent IT Support for Judges". In: *ICAIL 2003 Workshop on Legal Ontologies & Web based legal information management*.

Benjamins, V. et al. (2005a). "Law and the Semantic Web". In: vol. 3369. LNCS. Springer-Verlag Berlin Heidelberg. Chap. Law and the Semantic Web, an Introduction, pp. 1–17.

Benjamins, V. et al. (2005b). *Law and the Semantic Web Legal Ontologies, Methodologies, Legal Information Retrieval, and Applications*. Ed. by V. Benjamins et al. Springer.

Bernaras, A., Laresgoiti, I., and Correra, J. (1996). "Building and Reusing Ontologies for Electrical Network Applications". In: *12th European conference on Artificial Intelligence (ECAI)*, pp. 298–302.

Berners-Lee, T. (2005). "Web for Real People," in: *14th World Wide Web Conference (WWW2005)*. URL: http://www.w3.org/2005/%20Talks/0511-keynote-tbl/.

Bezerra, C. et al. (2008). "ModOnto: A tool for modularizing ontologies". In: *Proc. WONTO-08*. Vol. 427. ceur-ws.org.

Bezerra, C. et al. (2009). "An approach for ontology modularization". In: *Proc. Brazil/INRIA colloquium on computation: cooperations, advances and challenges*, pp. 184–189.

Bezzazi, H. (2007). "Building an Ontology That Helps Identify Criminal Law Articles That Apply to a Cybercrime Case". In: *Proceedings of the Second International Conference on Software and Data Technologies*.

Biagioli, C. (1991). *Definitional Elements of a Language for the Representation of Statutory Texts*. Rechtstheorie, Beiheft 11, Berlin: Duncker and Humblot.

Biagioli, C. (1997). "Towards a Legal Rules Functional Micro-Ontology". In: *First International Workshop proceedings on Legal Ontologies*.

Biasiotti, M. and Tiscornia, D. (2011). "Approaches to Legal Ontologies". In: ed. by G. Sartor et al. Law, Governance and Technology Series. Springer, Dordrecht. Chap. Legal Ontologies: The Linguistic Perspective, pp. 143–166.

Biasiotti, M. et al. (2008). "Legal informatics and management of legislative documents". In: *Global Center for ICT in Parliament Working Paper* 2.

Biebow, B. and Szulman, S. (1999). "TERMINAE: A Linguistics-Based Tool for the Building of a Domain Ontology". In: *International Conference on Knowledge Engineering and Knowledge Management EKAW 1999: Knowledge Acquisition, Modeling and Management*. Vol. 1621. LNCS, pp. 49–66.

Blazquez, M. et al. (1998). "Building ontologies at the knowledge level using the ontology design environment". In: *11th International Workshop on Knowledge Acquisition, Modeling and Management (KAW 98)*. Ed. by M. M. B.R. Gaines.

Boella, G., Lesmo, L., and Damiano, R. (2005). "Law and the Semantic Web". In: ed. by V. Benjamins et al. Vol. 3369. Lecture Notes in Computer Science. springer, Berlin, Heidelberg. Chap. On the Ontological Status of Norms, pp. 125–141.

Boella, G. et al. (2011). "A Formal Study on Legal Compliance and Interpretation". In: *AICOL Workshops(2009)*. Springer, pp. 162–183.

Boer, A., Engers, T., and Winkels, R. (2003). "Using ontologies for comparing and harmonizing legislation". In: *ICAIL 03 Proceedings of the 9th international conference on Artificial intelligence and law*, pp. 60–69.

Boer, A., Engers, T., and Winkels, R. (2005). "Mixing Legal and Non-legal Norms". In: *Jurix 2005: The Eighteenth Annual Conference. Legal Knowledge and Information Systems*. Ed. by M. F. M. andP. Spyns. IOS Press, pp. 25–36.

Boer, A., Hoekstra, R., and Winkels, R. (2001). "The CLIME Ontology". In: *Second International Workshop on Legal Ontologies*.

Boer, A., Hoekstra, R., and Winkels, R. (2002a). "Metalex: Legislation in XML". In: *In Proc. JURIX 2002*.

Boer, A. et al. (2002b). "Proposal for a dutch legal xml standard". In: *EGOV2002 - Proceedings of the First International Conference on Electronic Government*.

Boley, H., Tabet, S., and Wagner, G. (2001). "Design rationale for RuleML: A markup language for Semantic Web rules". In: *Proc. SWWS'01, The first Semantic Web Working Symposium*. Ed. by I. Cruz et al., pp. 381–401.

Borgida, A. and Serani, L. (2003). "Distributed description logics: Assimilating information from peer sources". In: *J. Data Semantics* 1, pp. 153–184.

Borgo, S., Guarino, N., and Masolo, C. (1996). "Stratified Ontologies: the case of physical objects". In: *Workshop on Ontological Engineering. ECAI96*, pp. 5–15.

Borgo, S. and Leitao, P. (2004). "The role of foundational ontologies in manufacturing domain applications". In: *OTM 2004: On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE*. Vol. 3290. Lecture Notes in Computer Science, pp. 670–688.

Borst, E. (1997). "Construction of Engineering Ontologies for Knowledge Sharing and Reuse". PhD thesis. University of Twente.

Bos, J. (2008). "Wide-coverage semantic analysis with boxer". In: *Semantics in Text Processing. STEP 2008 Conference Proceedings, Research in Computational Semantics*. Ed. by J. Bos and R. Delmonte, pp. 277–286.

Bourigault, D. (1994). "LEXTER, un Logiciel d'EXtraction de TERminologie. Application a l'acquisition des connaissances a partir de textes". PhD thesis. EHESS Paris.

Bourigault, D., Jacqueminand, C., and L'Homme, M. (2001). *Recent Advances in Computational Terminology*. Ed. by D. Bourigault. John Benjamins.

Brank, J., Grobelnik, M., and Mladenic, D. (2005). "A survey of ontology evaluation techniques". In: *Conference on Data Mining and Data Warehouses (SiKDD 2005)*, pp. 166–170.

Breuker, J. (1990). "Legal KBS: aims for research and development". In: chap. Towards a workbench for the legal practitioner.

Breuker, J. (2003). "The construction and use of ontologies of criminal law in the e-Court European project". In: *Means of Electronic Communication*.

Breuker, J. (2004). "Constructing a legal core ontology: Lri-core". In: *Workshop on Ontologies and their Applications*.

Breuker, J. et al. (2007). *Owl ontology of basic legal concepts (lkif-core). Deliverable 1.4 D.1.4, ESTRELLA project*. Tech. rep.

Breuker, J., Dieng, R., and Gadon, F. (2003). "Managing legal domains: in search of a core ontology for law". In: *Workshop on Knowledge Management and the Semantic Web at KCAP-2003*.

Breuker, J. and Hoekstra, R. (2004a). "Epistemology and ontology in core ontologies: FOLaw and LRI-Core, two core ontologies for law". In: *roceedings of Workshop on Core Ontologies in Ontology Engineering in the 14th International Conference (EKAW'04)*.

Breuker, J., Valente, A., and Winkels, R. (2004b). "Legal Ontologies in Knowledge Engineering and Information Management". In: *Artificial Intelligence and Law* 12, pp. 241–277.

Breuker, J. and Winkels, R. (2004c). "Use and reuse of legal ontologies in knowledge engineering and information management". In: *Artificial Intelligence and Law*.

Breuker, J. et al. (2002). "Ontologies for Legal Information Serving and Knowledge Management". In: *In Legal Knowledge and Information Systems, Jurix 2002: The Fifteenth Annual Conference*.

Breuker, J. et al. (2009). "Law, Ontologies and the Semantic Web - Channelling the Legal Information Flood". In: IOS Press. Chap. The Flood, the Channels and the Dykes: Managing Legal Information in a Globalized and Digital World.

Breuker, J. et al. (n.d.). *IT Support for the Judiciary: Use of Ontologies in the e-Court Project*.

Brewster, C. et al. (2004). "Data driven ontology evaluation". In: *Int. Conf. on Language Resources and Evaluation*. Lisbon.

Brickley, D. and Guha, R. (2004). *RDF vocabulary description language 1.0: RDF schema*. Tech. rep. W3C Recommendation.

Buchanan, B. and Duda, R. (1982). *Principles of Rule-Based Expert Systems*. Tech. rep. Stauford University.

Buchanan, B. and Duda, R. (1983). "Advances in Computers". In: vol. 22. Elsevier, Science Direct. Chap. Principles of Rule-Based Expert Systems, pp. 163–216.

Buchanan, B. and Shortliffe, E. (1984). *Rule-Based Expert Systems*. Addison-Wesley.

Buitelaar, P. et al. (2006). "DFKI-LT - LingInfo: Design and Applications of a Model for the Integration of Linguistic Information in Ontologies". In: *Proc. of OntoLex06, a Workshop at LREC*.

Buitelaar, P. and Cimiano, P. (2007). *Bridging the Gap from Text to Knowledge. Selected Contributions in Ontology Learning and Population from Text*. Tech. rep.

Buitelaar, P., Cimiano, P., and Magnini, B. (2005a). *Ontology Learning from Text: Methods, Evaluation and Applications*. Vol. 123. Frontiers in Artificial Intelligence and Applications. IOS Press.

Buitelaar, P. and Magnini, B. (2005b). "Ontology Learning from Text: An Overview". In: *Ontology Learning from Text: Methods, Applications and Evaluation*. Ed. by P. Buitelaar, P. Cimiano, and B. Magnini. IOS Press, pp. 3–12.

Caldarola, E., Picariello, A., and Rinaldi, A. (2015). "An approach to ontology integration for ontology reuse in knowledge based digital ecosystems". In: *7th International Conference on Management of computational and collective intElligence in Digital EcoSystems*. ACM, pp. 1–8.

Capellades, M. (1999). *Assessment of the Reusability of Ontologies: A Practical Example*. Tech. rep. AAAI.

Cardoso, J. (2007). "The Semantic Web Vision: Where are We?" In: *IEEE Intelligent Systems*, pp. 22–26.

Carroll, J. et al. (2004). "Jena: implementing the semantic web recommendations". In: *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*. Ed. by S. Feldman et al. New York, NY, USA: ACM, pp. 74–83.

Casanovas, P., Noriega, P., and Bourcier, D. (2007). *Trends in Legal Knowledge. The Semantic Web and the Regulation of Electronic Social Systems*. European Press Academic Publishing.

Casellas, N. (2008a). "Modelling Legal Knowledge through Ontologies. OPJK: the Ontology of Professional Judicial Knowledge". PhD thesis. Universitat Autonoma de Barcelona.

Casellas, N. (2008b). "Modelling Legal Knowledge through Ontologies. OPJK: the Ontology of Professional Judicial Knowledge". PhD thesis. UNIVERSITAT AUTONOMA DE BARCELONA.

Chandrasekaran, B., Johnson, T., and Smith, J. (1992). "Task-Structure Analysis for Knowledge Modelling". In: *Communications of the ACM* 35.9, pp. 124–137.

Cherubini, M. and Tiscornia, D. (2008). "An ontology-based model of procedural norms and regulated procedures". In: *eGov international conference proceedings*.

Cimiano, P. (2006). *Ontology Learning and Population from Text. Algorithms, Evaluation and Applications*. Springer.

Cimiano, P., Hotho, A., and Staab, S. (2005a). "Learning concept hierarchies from text corpora using formal concept analysis". In: *Journal of Artificial Intelligence research* 24, pp. 305–339.

Cimiano, P. and Volker, J. (2005b). "Text2Onto: a framework for ontology learning and data-driven change discovery". In: *NLDB 05 Proceedings of the 10th international conference on Natural Language Processing and Information Systems*, pp. 227–238.

Cimiano, P. et al. (2004). *Ontology Learning. Handbook on Ontologies*. Springer.

Clancey, W. (1989). "The Knowledge Level Reinterpreted: Modeling How Systems Interact". In: *Machine Learning* 4, pp. 285–291.

Corcho, O., Fernandez-Lopez, M., and Gomez-Perez, A. (2003). "Methodologies, tools and languages for building ontologies. Where is their meeting point?" In: *Data & Knowledge Engineering* 46, pp. 41–64.

Corcho, O., Fernandez-Lopez, M., and Gomez-Perez, A. (2006). "Ontologies for Software Engineering and Software Technology". In: ed. by C. Calero, F. Ruiz, and M. Piattini. Springer. Chap. Ontological Engineering: Principles, Methods, Tools and Languages, pp. 1–48.

Corcho, O., Fernandez-Lopez, M., and Gomez-Perez, A. (2007). "Semantic Web Services: Theory, Tools and Applications". In: IGI Global. Chap. Ontological Engineering: What are Ontologies and How Can We Build Them?, pp. 44–70.

Corcho, O. et al. (2005). "Law and the Semantic Web, LNAI 3369". In: ed. by V. Benjamins et al. Springer-Verlag Berlin Heidelberg. Chap. Building Legal Ontologies with METHONTOLOGY and WebODE, pp. 142–157.

Corsar, D. and Sleeman, D. (2008). "Developing knowledge-Based Systems using the Semantic Web". In: *SSS-08 on Symbiotic Relationships between the Semantic Web & Knowledge Engineering*.

Cui, Z., Jones, D., and Brien, P. (2000). "Domain ontology management environment". In: *33rd Hawaii International Conference on System Sciences 2000*.

d'Aquin, M. et al. (2007). "Ontology modularization for knowledge selection: Experiments and evaluations". In: *18th International Conference on Database and Expert Systems Applica-tions (DEXA 07)*. Ed. by R. Wagner, N. Revell, and G. Pernul. Vol. 4653. LNCS. Springer, pp. 874–883.

d'Aquin, M. et al. (2009). "Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization". In: ed. by H. Stuckenschmidt, C. Parent, and S. Spaccapietra. Vol. 5445. LNCS. Springer. Chap. Criteria and evaluation for ontology modularization techniques, pp. 67–89.

Davies, N., Fensel, D., and Richardson, M. (2004). "The future of Web Services". In: *BT Technology* 22.1, pp. 118–130.

Davis, R. and King, J. (1984). "Rule Based Expert Systems-The MYCIN Experiments of the Stanford Heuristic Programming Project". In: ed. by B. Buchanan and E. Shortlie. Addison-Wesley. Chap. The origin of rule-based systems in AI, pp. 20–52.

Dean, M. and Schreiber, G. (2004). *OWL Web ontology language reference*. Tech. rep. W3C Recommendation.

Dell'Orletta, F. et al. (2014). "T2K: a System for Automatically Extracting and Organizing Knowledge from Texts". In: *proceeding of LREC*, pp. 26–31.

Dentler, K. et al. (2011). "Comparison of reasoners for large ontologies in the OWL 2 EL profile". In: *Semantic Web* 2.2, pp. 71–87.

Ding, L. et al. (2004). "Swoogle: A search and metadata engine for the semantic web". In: *Proc. CIKM*, pp. 652–659.

Dmitrieva, J. and Verbeek, F. (2011). "Modular Approach for a new Ontology." In: *5th International Workshop on Modular Ontologies WoMO*.

Dong, M., Yang, D., and Su, L. (2011). "Ontology-based service product configuration system modeling and development". In: *Journal of Expert systems with applications* 38.9, pp. 11770–11786.

Doran, P. (2006). "Ontology reuse via ontology modularization." In: *Proceedings of Knowledge-Web PhD Symposium*, pp. 1–6.

Doran, P. (2009). "Ontology modularization: Principles and practice." PhD thesis. University of Liverpool, Liverpool, UK.

Dove, I. (1996a). "LEGAL EXPERT SYSTEMS: THE END OF JURISPRUDENCE?" In:

Dove, I. (1996b). "LEGAL EXPERT SYSTEMS: THE END OF JURISPRUDENCE?" In: *The Journal of Legal Studies in Business* 5.

Eiter, T. et al. (2004). "Combining answer set programming with description logics for the semantic web". In: *The International Conference of Knowledge Representation and Reasoning (KR04)*.

Eiter, T. et al. (2006a). "Effective Integration of Declarative Rules with External Evaluations for Semantic-Web Reasoning". In: *The Semantic Web: Research and Applications. ESWC 2006*. Ed. by Y. Sure and J. Domingue. Vol. 4011. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg.

Eiter, T. et al. (2006b). "Reasoning with rules and ontologies". In: *Reasoning Web 2006*, pp. 93–127.

Eiter, T. et al. (2008). "Reasoning Web". In: ed. by C. Baroglio et al. Vol. 5224. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg. Chap. Rules and Ontologies for the Semantic Web, pp. 1–53.

Elenius, D. et al. (2009). "Reasoning about resources and hierarchical tasks using OWL and SWRL". In: *International Semantic Web Conference*. USA.

Eriksson, E. et al. (1995). "Task Modeling with Reusable Problem-Solving Methods". In: *Artificial Intelligence* 79, pp. 293–326.

Eriksson, H., Puerta, A., and Musen, M. (1994). "Generation of Knowledge Acquisition Tools from Domain Ontologies". In: *Int. J. Human-Computer Studies* 41, pp. 425–453.

Ernest, D. (1990). *Representations of Commonsense Knowledge*. Los Altos: Morgan Kaufman.

Euzenat, J. (2007). "Semantic precision and recall for ontology alignment evaluation". In: *IJCAI*, pp. 348–353.

Euzenat, J. and Shvaiko, P. (2007). *Ontology Matching*. Springer-Verlag, Heidelberg (DE).

Evans, D. (2003). "A framework for named entity recognition in the open domain". In: *the Recent Advances in Natural Language Processing (RANLP-2003)*.

Falbo, R. (2014). "SABiO: Systematic Approach for Building Ontologies," in: *ONTO.COM/ODISE@FOIS*.

Fensel, D. and Straatman, R. (1996). "Advances in Knowledge Acquisiiton, Lecture Notes in Artificial Intelligence (LNAI) 1076". In: ed. by N. Shadbolt et al.

Springer-Verlag, Berlin. Chap. The Essence of Problem-Solving Methods: Making Assumptions for Efficiency Reasons.

Fernandez-Barrera, M. and Sartor, G. (2011). "Approaches to Legal Ontologies". In: ed. by G. Sartor et al. Vol. 1. Law, Governance and Technology Series. Springer, Dordrecht. Chap. The Legal Theory Perspective: Doctrinal Conceptual Systems vs. Computational Ontologies, pp. 15–47.

Fernandez-Lopez, M. (1999). "Overview of methodologies for building ontologies". In: *Workshop on Ontologies and Problem-Solving Methods in conjunction with the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI'99)*, pp. 4.1–4.13.

Fernandez-Lopez, M. and Gomez-Perez, A. (2002). "Overview and analysis of methodologies for building ontologies". In: *Knowledge Engineering Review* 17.2, pp. 129–156.

Fernandez-Lopez, M., Gomez-Perez, A., and Juristo, N. (1997). "Ontological Engineering: Papers from the 1997 Spring Symposium AAAI97". In: ed. by A. Farquhar and M. Gruninger. Chap. Methontology: from ontological art towards ontological engineering, pp. 33–40.

Fielding, J. et al. (2004). "Ontological Theory for Ontology Engineering". In: *Proceedings of 9th International Conference on the Principles of Knowledge Representation and Reasoning*.

Fiorentini, X. et al. (2010). "An analysis of description logic augmented with domain rules for the development of product models". In: *Journal of Computing and Information Science in Engineering* 10, pp. 1–13.

Fortineau, V. et al. (2012). "Product Lifecycle Management. Towards Knowledge-Rich Enterprises. PLM". In: ed. by L. Rivest, A. Bouras, and B. Louhichi. Vol. 388. IFIP Advances in Information and Communication Technology. Springer, Berlin, Heidelberg. Chap. Swrl as a rule language for ontology-based models in power plant design, pp. 588–597.

Fortuna, B., Grobelnik, M., and Mladenic, D. (2007). "Ontogen: Semi-automatic ontology editor". In: *Proceedings of Human Interface and the Management of Information. Interacting in Information Environments, Symposium on Human Interface 2007*. Vol. 4558. Lecture Notes in Computer Science. Springer, pp. 309–318.

Francesconi, E. (2010). "Legal rules learning based on a semantic model for legislation". In: *Proceedings of the LREC 2010 Workshop on the Semantic Processing of Legal Texts (SPLeT-2010)*. Malta.

Francesconi, E. (2011). "Approaches to Legal Ontologies - Theories, Domains, Methodologies". In: ed. by P. Casanovas et al. Vol. 1. Law, Governance and Technology. Springer Berlin / Heidelberg. Chap. A Learning Approach for Knowledge Acquisition in the Legal Domain, pp. 219–233.

Francesconi, E. (2016). "Semantic model for legal resources: Annotation and reasoning over normative provisions". In: *Semantic Web* 7.3, pp. 255–265.

Francesconi, E., Spinosa, P., and Tiscornia, D. (2007). "A linguistic-ontological support for multilingual legislative drafting: the DALOS Project". In: *Proceedings of LOAIT'07, II Workshop on Legal Ontologies and Artificial Intelligence Techniques*. Ed. by P. Casanovas et al.

Francesconi, E. and Tiscornia, D. (2008). "Computable Models of the Law". In: ed. by P. Casanovas et al. Vol. 4884. LNCS. Springer-Verlag Berlin, Heidelberg. Chap. Building Semantic Resources for Legislative Drafting: The DALOS Project, pp. 56–70.

Francesconi, E. et al. (2010). "Semantic Processing of Legal Texts: where the Language of Law Meets the Law of Language". In: ed. by E. Francesconi et al. Springer-Verlag, Berlin, Heidelberg. Chap. Integrating a bottom-up and top-down methodology for building semantic resources for the multilingual legal domain, pp. 95–121.

Gamallo, P. et al. (2002). "Mapping Syntactic Dependencies onto Semantic Relations". In: *ECAI Workshop on Machine Learning and Natural Language Processing for Ontology Engineering*.

Gangemi, A., Catenacci, C., and Battaglia, M. (2004). "Inflammation ontology design pattern: an exercise in building a core biomedical ontology with descriptions and situations". In: *Stud Health Technol Inform* 102, pp. 64–80.

Gangemi, A., Sagri, M., and Tiscornia, D. (2003). "Metadata for Content Description in Legal Information". In: *ICAIL 2003 Workshop on Legal Ontologies & Web based legal information management*.

Gangemi, A. et al. (2006). "Modelling ontology evaluation and validation". In: *Proceedings of the 3rd European Semantic Web Conference (ESWC2006)*. LNCS 4011. Budva: Springer.

Gardner, A. (1987). *An Artificial Intelligence Approach To Legal Reasoning*. Ed. by E. Rissland. MIT Press. Chap. Book Review, pp. 223–233.

Gelfond, M. and Lifschitz, V. (1988). "The Stable Model Semantics for Logic Programming". In: *Proc. of ICLP'88*. Cambridge, Massachusetts: MIT Press, pp. 1070–1080.

Gelfond, M. and Lifschitz, V. (1991). "Classical negation in logic programs and disjunctive databases". In: *New Generation Computing* 9.3-4, pp. 365–385.

Genesereth, M. and Fikes, R. (1992). *Knowledge interchange format. Version 3.0. Reference Manual*. Tech. rep. Stanford University, Computer Science Department. URL: http://logic.stanford.edu/kif/Hypertext/kif-manual.html.

Genesereth, M. and Nilsson, N. (1987). *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann.

Ghazvinian, A., Noy, N., and Musen, M. (2009). "Creating mappings for ontologies in biomedi-cine: Simple methods work". In: *AMIA 2009 Symposium Proceedings*.

Glimm, B. et al. (2014). "HermiT: An OWL 2 Reasoner". In: *Journal of Automated Reasoning* 53.3, pp. 245–269.

Goh, C. (1997). "Representing and reasoning about semantic conflicts in heterogeneous information sources". PhD thesis.

Golbreich, C. (2004). "Combining Rule and Ontology Reasoners for the Semantic Web". In: *Rules and Rule Markup Languages for the Semantic Web. RuleML 2004*. Ed. by G. Antoniou and H. Boley. Vol. 3323. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg.

Gomez-Perez, A. (1995). "Some ideas and examples to evaluate ontologies". In: *11th Conference on Artificial Intelligence for Applications*.

Gomez-Perez, A. (1999). "Ontological Engineering: A State of the Art". In: *Expert Update. British Computer Society* 2.3, pp. 33–44.

Gomez-Perez, A. (2004). "Handbook on Ontologies". In: ed. by S. Staab and R. Studer. Springer Berlin Heidelberg, Berlin, Heidelberg. Chap. Ontology Evaluation, pp. 251–274.

Gomez-Perez, A., Fernandez-Lopez, M., and Corcho, O. (2003a). *Ontological Engineering*. Springer Verlag.

Gomez-Perez, A., Fernandez-Lopez, M., and Corcho, O. (2004). *Ontological Engineering with Examples from the Areas of Knowledge Management, e-Commerce and the Semantic Web*. Advanced Information and Knowledge Processing 1st ed.

Gomez-Perez, A. and Manzano-Macho, D. (2003b). *A Survey of Ontology Learning Methods and Techniques,Ontoweb Deliverable 1.5.2003.* Tech. rep.

Gomez-Perez, A. and Rojas, M. (1999). "Ontological reengineering and reuse". In: *11th European Workshop on Knowledge Acquisition, Modeling and Management (EKAW '99)*. Ed. by D. Fensel and R. Studer. Vol. 1621. Lecture Notes in Artificial Intelligence. Springer Berlin, pp. 139–156.

Gonzalez, A. and Dankel, D. (1993). *The engineering of knowledgebased systems, theory and practice*. Englewood Cliffs, NJ: Prentice Hall.

Gordon, T. (1995). "The Pleadings Game; An Artificial Intelligence Model of Procedural Justice". PhD thesis. University of Darmstadt.

Gordon, T., Governatori, G., and Rotolo, A. (2009). "Rules and Norms: Requirements for Rule Interchange Languages in the Legal Domain". In: *RuleML 2009: Rule Interchange and Applications*. Ed. by G. G. G., J. Hall, and A. Paschke. Vol. 5858. Lecture Notes in Computer Science (LNCS). Springer, Berlin, Heidelberg, pp. 282–296.

Gordon, T., Prakken, H., and Walton, D. (2007). "The Carneades model of argument and burden of proof". In: *Artificial Intelligence* 171.10-11, pp. 875–896.

Gostojic, S. and Milosavljevic, B. (2013). "Ontological Model of Legal Norms for Creating and Using Legal Acts". In: *The IPSI BgD Journal* 9.1, pp. 19–25.

Governatori, G. (2005). "Representing business contracts in RuleML". In: *International Journal of Cooperative Information Systems* 14.2-3, pp. 181–216.

Governatori, G. and Pham, D. (2009). "Dr-contract: An architecture for e-contracts in defeasible logic". In: *International Journal of Business Process Integration and Management* 5.4.

Governatori, G. and Rotolo, A. (2006). "Logic of violations: A Gentzen system for reasoning with contrary-to-duty obligations". In: *Australasian Journal of Logic* 4, pp. 193–215.

Grassl, W. (1999). "The Reality of Brands: Towards an Ontology of Marketing". In: *The American Journal of Economics and Sociology* 58.2, pp. 313–359.

Grau, B. and Kutz, O. (2007a). "Modular ontology languages revisited". In: *Proc. of the IJCAI-2007 Workshop on Semantic Web for Collaborative Knowledge Acquisition*.

Grau, B. et al. (2006). "Modularity and web ontologies". In: *KR*, pp. 198–209.

Grau, B. et al. (2007b). "A logical framework for modularity of ontologies". In: *Proceedings of IJCAI 07*. AAAI Press, pp. 298–303.

Grau, B. et al. (2007c). *Extracting modules from ontologies: Theory and practice*. Tech. rep. University of Manchester.

Grau, B. et al. (2008). "OWL 2: The next step for OWL". In: *J. Web Semantics* 6.4, pp. 309–322.

Greenleaf, G. (1989). "Legal Expert Systems - Robot Lawyers? (An Introduction to Knowledge-Based Applications to Law)". In: *Australian Legal Convention*.

Griffo, C., Almeida, J., and Guizzardi, G. (2015). "Towards a Legal Core Ontology based on Alexy's Theory of Fundamental Rights". In: *MWAIL2015 ICAIL Multilingual Workshop on AI & Law Research*, pp. 89–100.

Griffo, C., Almeida, J., and Guizzardi, G. (2016). "A Pattern for the Representation of Legal Relations in a Legal Core Ontology". In: *JURIX 2016. Frontiers in Artificial Intelligence and Applications*.

Grosof, B. (2004). "Representing e-commerce rules via situated courteous logic programs in RuleML". In: *Electronic Commerce Research and Applications* 3.1, pp. 2–20.

Grosof, B. et al. (2003). "Description logic programs: Combining logic programs with description logic". In: *The Twelfth International World Wide Web Conference (WWW 2003)*. ACM, pp. 48–57.

Gruber, T. (1993). "A Translation Approach to Portable Ontology Specification". In: *Knowledge Acquisition*, pp. 199–220.

Gruber, T. (1995). "Toward principles for the design of ontologies used for knowledge sharing". In: *International Journal of Human-Computer Studies* 43, pp. 907–928.

Gruner, R. (1986). "Thinking like a Lawyer: Expert Systems for Legal Analysis". In: *Berkeley Technology Law Journal* 1.2.

Gruninger, M. and Fox, M. (1995). "Methodology for the design and evaluation of ontologies". In: *IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing*, pp. 6.1–6.10.

Guarino, N. (1997). "Understanding, building and using ontologies". In: *International Journal of Human-Computer Studies* 46.2-3, pp. 293–310.

Guarino, N. (1998). "Formal Ontology and Information Systems". In: *Formal Ontology in Information Systems. Proceedings of FOIS 98*. Trento, Italy: IOS Press, pp. 3–15.

Guarino, N., Carrara, M., and Giaretta, P. (1994). "An Ontology of Meta-Level Categories". In: *Principles of Knowledge Representation and Reason-ing: Proceedings of KR94*. Ed. by E. S. J Doyle and P. Torasso.

Guarino, N., Oberle, D., and Staab, S. (2009). "Handbook on Ontologies". In: Springer. Chap. What Is an Ontology?, pp. 1–17.

Guarino, N. and Schneider, L. (2002a). *Ontology-Driven Conceptual Modelling: Advanced Concepts. ER 2002.* Pre-Conference Tutorials. URL: http://www.loa-cnr.it/odcm.html.

Guarino, N. and Welty, C. (2002b). "Evaluating ontological decisions with OntoClean". In: *Comm. of the ACM* 45.2, pp. 61–65.

Guerson, J., Almeida, J., and Guizzardi, G. (2014). "Support for Domain Constraints in the Validation of Ontologically Well-Founded Conceptual Models". In: *Enterprise, Business-Process and Information Systems Modeling*. Vol. 175. LNBIP. Springer, pp. 302–316.

Guerson, J. et al. (2015). "OntoUML Lightweight Editor: A Model-Based Environment to Build, Evaluate and Implement Reference Ontologies". In: *Enterprise Distributed Object Computing Workshop (EDOCW), 2015 IEEE 19th International*.

Guizzardi, G. (2005). "Ontological Foundations for Structural Conceptual Models". PhD thesis. Telematica Institut, The Netherlands.

Guizzardi, G. (2006). "The Role of Foundational Ontology for Conceptual Modeling and Domain Ontology Representation". In: *7th DB&IS, Vilnius,* IEEE Press.

Guizzardi, G. (2007). "On Ontology, ontologies, Conceptualizations, Modeling Languages and (Meta)Models". In: *Frontiers in Artificial Intelligence and Applications, Databases and Information Systems IV*. IOS Press.

Guizzardi, G. (2012). "Ontological Foundations for Conceptual Modeling with Applications". In: *CAiSE 2012: Advanced Information Systems Engineering*. Ed. by J. Ralyte et al. Vol. 7328. LNCS. Springer, pp. 695–696. URL: http://doi.org/10.1007/978-3-642-31095-9_45..

Guizzardi, G. et al. (2013a). "Towards Ontological Foundations for the Conceptual Modeling of Events". In: *ER 2013: Conceptual Modeling*. Ed. by W. Ng, V. C. Storey, and J. C. Trujillo. Vol. 8217. LNCS. Springer, Berlin, Heidelberg, pp. 327–341.

Guizzardi, G., Falbo, R., and Guizzardi, R. (2008a). "Grounding software domain ontologies in the unified foundational ontology (ufo): The case of the ode software process ontology". In: *1th Iberoamerican Workshop on Requirements Engineering and Software Environments (IDEAS 2008)*.

Guizzardi, G. and Halpin, T. (2008b). "Ontological foundations for conceptual modelling". In: *Applied Ontology* 3, pp. 1–12.

Guizzardi, G. and Wagner, G. (2004a). "A unified foundational ontology and some applications of it in business modeling". In: *In CAiSE Workshops (3)*, pp. 129–143.

Guizzardi, G. and Wagner, G. (2005a). "Applications of a Unified Foundational Ontology." In: chap. Some Applications of a Unified Foundational Ontology in Business Modeling, pp. 345–367.

Guizzardi, G. and Wagner, G. (2005b). "Towards Ontological Foundations for Agent Modelling Concepts Using the Unified Fundational Ontology (UFO)". In: *Agent-Oriented Information Systems II*. Vol. 3508. LNCS, pp. 110–124.

Guizzardi, G. and Wagner, G. (2010a). "Theory and Applications of Ontology: Computer Applications". In: ed. by R. Poli, M. Healy, and A. Kameas. springer. Chap. Using the Unified Foundational Ontology (UFO) as a Foundation for General Conceptual Modeling Languages, pp. 175–196. DOI: https://doi.org/10.1007/978-90-481-8847-5_8.

Guizzardi, G. and Wagner, G. (2012). "TUTORIAL: CONCEPTUAL SIMULATION MODELING WITH ONTO-UML". In: *2012 Winter Simulation Conference*. Ed. by C. Laroque et al.

Guizzardi, G. and Wagner, G. (2013b). "DISPOSITIONS AND CAUSAL LAWS AS THE ONTOLOGICAL FOUNDATION OF TRANSITION RULES IN SIMULATION MODELS". In: *2013 Winter Simulation Conference*. Ed. by R. Pasupathy et al.

Guizzardi, G. et al. (2004b). "An Ontologically Well Founded Profile for UML Conceptual Models". In: *Advanced Information Systems Engineering, Proceedings of16th CAiSE Conference*. Ed. by A. Persson and J. Stirna. Springer.

Guizzardi, G. et al. (2010b). "The role of foundational ontologies for domain ontology engineering". In: *International Journal of Information System Modeling and Design* 2.1, pp. 1–22.

Guizzardi, G. et al. (2010c). "The Role of Foundational Ontologies for Domain Ontology Engineering: An Industrial Case Study in the Domain of Oil and Gas Exploration and Production". In: *International Journal of Information System Modeling and Design* 1.2, pp. 1–22.

Guizzardi, G. et al. (2015). "Towards Ontological Foundations for Conceptual Modeling: The Unified Foundational Ontology (UFO) Story". In: *Applied Ontology*. DOI: 10.3233/AO-150157.

Haarslev, V. and Moller, R. (2001). "RACER System Description". In: *Proceedings of the First International Joint Conference on Automated Reasoning (IJCAR 2001)*. Vol. 2083. Lecture Notes in Computer Science (LNCS). Springer Verlag, pp. 701–705.

Haase, P. and Volker, J. (2005). "Ontology Learning and Reasoning - Dealing with Uncertainty and Inconsistency". In: *Uncertainty Reasoning for the Semantic Web*. Vol. 5327. LNCS, pp. 366–384.

Haase, P. et al. (2008). "The Neon Ontology Engineering Toolkit." In: *International World Wide Web Conference, 17, Beijing, China*.

Hakimpour, F. and Geppert, A. (2001a). *Ontologies: An Approach to Resolve Semantic Heterogeneity in Databases*. Tech. rep. Swiss National Science Foundation.

Hakimpour, F. and Geppert, A. (2001b). "Resolving Semantic Heterogeneity in Schema Integration: An Ontology Based Approach". In: *FOIS'OI, international conference on Formal Ontology in Information Systems*, pp. 297–308.

Hakimpour, F. and Geppert, A. (2002). "Global schema generation using formal ontologies". In: *21st Int'l Conf. on ConceptualModeling (ER2002)*. Ed. by S. Spaccapietra, S. T. March, and Y. Kambayashi. Vol. LNCS 2503. Springer Verlag, pp. 307–320.

Harnad, S. (1990). "The symbol grounding problem". In: *Physica D.* 42, pp. 335–346.

Harris, Z. (1968). *Mathematical structures of language*. Interscience Publishers.

Hartmann, J., Palma, R., and Gomez-Perez, A. (2009). "Handbook on Ontologies". In: International Handbooks on Information Systems book series (INFOSYS). Springer. Chap. Ontology Repositories, pp. 551–571.

Hartmann, J. et al. (2004). *Methods for ontology evaluation, Knowledge Web Deliverable D1.2.3*. Tech. rep.

Heflin, J. and Hendler, J. (2000). "Semantic interoperability on the web". In: *Extreme Markup Languages 2000*.

Heijst, G. (1995). "The Role of Ontologies in Knowledge Engineering". PhD thesis. University of Amsterdam.

Heijst, G., Schreiber, A., and Wielinga, B. (1997). "Using explicit ontologies in KBS development". In: *Int . J . Human Computer Studies* 45, pp. 183–292.

Henderson, J. and Bench-Capon, T. (2001). "Dynamic arguments in a case law domain". In: *In ICAIL 01: Proceedings of the 8th international conference on Artificial intelligence and law*. New York, USA: ACM Press, pp. 60–69.

Heymans, S., Nieuwenborgh, D., and Vermeir, D. (2005). "Nonmonotonic Ontological and Rule-based Reasoning with Extended Conceptual Logic Programs". In: *ESWC 2005: The Semantic Web: Research and Applications*, pp. 392–407.

Hoekstra, R. et al. (2007). "The LKIF Core ontology of basic legal concepts". In: *Proceedings of the Workshop on Legal Ontologies and Artificial Intelligence Techniques (LOAIT 2007)*, pp. 43–64.

Hoekstra, R. et al. (2009). "Law, Ontologies and the Semantic Web". In: vol. 188. Frontiers in Artificial Intelligence and Applications. IOS Press. Chap. LKIF Core : Principled Ontology Development for the Legal Domain, pp. 21–52.

Hofmann, T. (1999). "Probabilistic latent semantic indexing". In: *SIGIR 99 Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 50–57.

Hois, J., Bhatt, M., and Kutz, O. (2009). "Modular Ontologies for Architectural Design". In: *Proc. of FOMI-09*. Vol. 198. Frontiers in Artificial Intelligence and Applications. IOS Press.

Holsapple, C. and Joshi, K. (2002). "A collaborative approach to ontology design". In: *Communications of the ACM* 45.2, pp. 42–47.

Horridge, M. and Bechhofer, S. (2011). "The OWL API: A Java API for OWL Ontologies". In: *Semantic Web* 2.1, pp. 11–21.

Horrocks, I. et al. (2000). "OIL in a nutshell". In: *12th International Conference in Knowledge Engineering and Knowledge Management (EKAW 00)*. Ed. by R. D. Õ. Corby. LNAI 1937. Springer-Verlag, pp. 1–16.

Horrocks, I. and Harmelen, F. (2001). *Reference description of the DAML+OIL (March 2001) ontology markup language*. Tech. rep.

Horrocks, I., Kutz, O., and Sattler, U. (2006). "The even more irresistible SROIQ". In: *10th Int. Conf. On Principles of Knowledge Representation and Reasoning*. AAAI Press, pp. 57–67.

Horrocks, I. and Sattler, U. (2005). "A tableaux decision procedure for SHOIQ". In: *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005)*, pp. 448–453.

Horrocks, I. et al. (2004). *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*. W3C Member Submission. URL: http://www.w3.org/Submission/SWRL/.

Jimenez-Ruiz, J. et al. (2008). *Ontology integration using mappings: Towards getting the right logical consequences*. Tech. rep. Universitat Jaume, University of Oxford.

Kalfoglou, Y. and Schorlemmer, M. (2003). "Ontology Mapping: The State of the Art". In: *The Knowledge Engineering Review Journal* 18.1, pp. 1–31.

Kaneiwa, K. and Mizoguchi, R. (2009). "Distributed reasoning with ontologies and rules in order-sorted logic programming". In: *Journal of Web Semantics* 7.3, pp. 252–270.

Karp, P., Chaudhri, V., and Thomere, J. (1999). *XOL: An XML-based ontology exchange language. Version 0.3*. Tech. rep.

Kashyap, V. and Sheth, A. (1997). "Cooperative Information Systems: Current Trends and Directions". In: ed. by M. Papazoglou and G. Schlageter. Academic Press. Chap. Semantic Heterogeneity in Global Information Systems: The Role of Metadata, Context and Ontologies.

Keet, M. (2011). "The use of foundational ontologies in ontology development: An empirical assessment," in: *Proc. 8th Extended Semantic Web Conference*. Vol. 6643, pp. 321–335.

Kelsen, H. (1991). *General Theory of Norms*. Clarendon Press, Oxford.

Kifer, M., Lausen, G., and Wu, J. (1995). "Logical foundations of object-oriented and frame-based languages". In: *Journal of the ACM* 42.4, pp. 741–843.

Kim, W. and Seo, J. (1991). "Classifying schematic and data heterogeneity in multi-database systems". In: *IEEE Computer* 24.12, pp. 12–18.

Koepsell, D., ed. (1999). *Proceedings of the Buffalo Symposium on Applied Ontology in the Social Sciences*. Vol. 58. 2. The American Journal ofEconomics & Sociology.

Koepsell, D. (2003). *The Ontology of Cyberspace: Philosophy, Law, and the Future of Intellectual Property*. Open Court Publishing.

Kohn, W. (2003). "Grounding Ontologies". In: *COSIT 2003 Workshop on FUNDAMENTAL ISSUES IN SPATIAL AND GEOGRAPHIC ONTOLOGIES*.

Konev, B. et al. (2009). "Ontology modularization". In: ed. by H. Stuckenschmidt, S. Spacciapietra, and C. Parent. Vol. 5445. LNCS. Springer. Chap. Formal properties of modularization, pp. 25–66.

Kotis, K., Vouros, G., and JAlonso, J. (2004). "HCOME: A Tool-Supported Methodology for Engineering Living Ontologies". In: *Semantic Web and Databases. Second International Workshop - SWDB 2004*. Ed. by C. Bussler, V. Tannen, and I. Fundulaki. Vol. 3372. LNCS. Springer-Verlag, pp. 155–166.

Krisnadhi, A. and Hitzler, P. (2014). "Description Logics". In: *Encyclopedia of Social Network Analysis and Mining*.

Krotzsch, M., Simancik, F., and Horrocks, I. (2014). *Description Logics*. URL: http://korrekt.org/papers/Kroetzsch-Simancik-Horrocks_DL-Intro_IEEE-IS-2014.pdf.

Lam, H., Hashmi, M., and Scofield, B. (2016). "Enabling Reasoning with LegalRuleML". In: *10th International Web Rule Symposium (RuleML 2016)*. Ed. by N. Bassiliades et al. New York, USA: Springer International Publishing, pp. 241–257.

Lame, G. (2005). "Using nlp techniques to identify legal ontology components: concepts and relations". In: *Lecture Notes in Computer Science* 3369, pp. 169–184.

Landauer, T. and Dutnais, S. (1997). "A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge". In: *Psychological Review* 104.2, pp. 211–240.

Lassila, O. and Swick, R. (1999). *Resource description framework (RDF) model and syntax specification*. Tech. rep. W3C Recommendation.

Lehmann, J. and Voelker, J. (2014). "Perspectives on Ontology Learning". In: ed. by J. Lehmann and J. Voelker. AKA/IOS Press. Chap. An introduction to ontology learning, pp. ix–xvi.

Lezcano, L., Sicilia, M.-A., and Rodriguez-Solano, C. (2011). "Integrating reasoning and clinical archetypes using OWL ontologies and SWRL rules". In: *Journal of Biomedical Informatics* 44.2, pp. 343–353.

Lin, D. and Pantel, P. (2001). "Dirt - discovery of inference rules from text". In: *In Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 323–328.

Lin, D. and Pantel, P. (2002). "Concept discovery from text". In: *COLING 02 Proceedings of the 19th international conference on Computational linguistics*. Vol. 1, pp. 1–7.

Love, N. and Genesereth, M. (2005). "Computational law". In: *Proceedings of the 10th International Conference on Artificial Intelligence and Law, ICAIL '05, New York, NY, USA*, pp. 205–209.

Luke, S. and Heflin, J. (2000). *SHOE 1.01. Proposed specification*. Tech. rep. University of Maryland, Parallel Understanding Systems Group, Department of Computer Science. URL: https://www.cs.umd.edu/projects/plus/SHOE/onts/.

MacCartney, B. et al. (2003). "Practical partition-based theo-rem proving for large knowledge bases". In: *Proceedings of the 18th international joint con-ference on Artificial intelligence*, pp. 89–96.

MacGregor, R. (1991). "Inside the LOOM classifier". In: *SIGART Bulletin* 2.3, pp. 70–76.

Machado, A. and Oliveira, J. (2014). "A Legal Ontology of Relationships for Civil Law System". In: *1st Joint Workshop ONTO.COM / ODISE proceedings on Ontologies in Concep-tual Modeling and Information Systems Engineering*.

Madche, A. and Staab, S. (2000). "Mining ontologies from text". In: *Proceedings of EKAW 00*, pp. 189–202.

Maedche, A. and Staab, S. (2001). "Ontology learning for the semantic web". In: *IEEE Intelligent Systems* 16, pp. 72–79.

Maedche, A. and Staab, S. (2002). "Measuring similarity between ontologies". In: *Proc. CIKM*. Vol. 2473. LNAI.

Masolo, C. et al. (2003). *Wonderweb Deliverable D18 (ver. 1.0), Ontology Library*. Tech. rep.

Mazzega, P. et al. (2011). "Approaches to Legal Ontologies". In: ed. by G. Sartor et al. Vol. 1. Law, Governance and Technology Series. Springer, Dordrecht. Chap. A Complex-System Approach: Legal Knowledge, Ontology, Information and Networks, pp. 117–132.

McCarthy, J. (1980). "Circumscription - A Form of Non-Monotonic Reasoning". In: *Artificial Intelligence* 5.13, pp. 27–39.

McCarty, L. (1980). "The TAXMAN project: Towards a cognitive theory of legal argument". In: ed. by B. Niblett. Cambridge University Press. Chap. 3, pp. 23–43.

McCarty, L. (1983). "Intelligent Legal Information Systems: Problems and Prospects". In: *Rutgers Computer and Technology Law Journal* 9.2, pp. 265–294.

McCarty, L. (2007). "Deep semantic interpretations of legal texts". In: *proceeding of ICAIL*, pp. 217–224.

McDermott, J. (1993). "Readings in knowledge acquisition and learning". In: Morgan Kaufmann Publishers Inc. Chap. Preliminary steps toward a taxonomy of problem-solving methods, pp. 149–169.

Miller, G. (1995). "WordNet: A lexical database for english". In: *Communications of the ACM* 38, pp. 39–41.

Mizoguchi, R. (2004). "Tutorial on ontological engineering: part 3: Advanced course of ontological engineering". In: *New Generation Computing - Grid systems for life sciences* 22.2, pp. 198–220.

Mizoguchi, R. and Ikeda, M. (1997). "Towards Ontology Engineering". In: *Proc. of The Joint 1997 Pacific Asian Conference on Expert systems / Singapore International Conference on Intelligent Systems*. Singapore: Nanyang Tech. University, pp. 259–266.

Mizoguchi, R. and Kozaki, K. (2009). "Handbook on Ontologies. International Handbooks on Information Systems". In: ed. by S. Staab and R. Studer. Springer, Berlin, Heidelberg. Chap. Ontology Engineering Environments, pp. 315–336.

Mizoguchi, R., Vanwelkenhuysen, J., and Ikeda, M. (1995). "Task Ontology for Reuse of Problem Solving Knowledge". In: *Towards Very Large Knowledge Bases: Knowledge Building & Knowledge Sharing*, pp. 46–59.

Mommers, L. (2003). "Application of a knowledge-based ontology of the legal domain in collaborative workspaces". In: *ICAIL 03 Proceedings of the 9th international conference on Artificial intelligence and law*. UK, pp. 70–76.

Mommers, L. (2010). "Theory and Applications of Ontology: Philosophical Perspectives". In: ed. by R. Poli and J. Seibt. Springer. Chap. Ontologies in the Legal Domain, pp. 265–276.

Moreira, J. et al. (2016). "Menthor Editor: An Ontology-Driven Conceptual Modeling Platform". In: *JOWO@FOIS*.

Morik, K. (1993). "Balanced cooperative modeling". In: *Machine Learning* 11.1, pp. 217–235.

Motik, B., Sattler, U., and Studer, R. (2005). "Query answering for owl-dl with rules". In: *Journal of Web Semantics: Science, Services and Agents on the World Wide Web* 3.1, pp. 41–60.

Motik, M., Patel-Schneider, P., and Parsia, B. (2009). *OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax*. Tech. rep. W3C Recommendation.

Musen, M. and Schreiber, A. (1995). "Architectures for intelligent systems based on reusable components". In: *Artificial Intelligence in Medicine*.

Mylopoulos, J. (1992). "Conceptual Modelling, Databases and CASE: An Integrated View of Information Systems Development". In: ed. by P. Loucopoulos and R. Zicari. Wiley. Chap. Conceptual modeling and telos, pp. 49–68.

Nardi, J. et al. (2016). "A commitment-based reference ontology for services". In: *Information Systems* 56, pp. 133–134. DOI: https://doi.org/10.1016/j.is.2015.01.012.

Neches, R. et al. (1991a). "Enabling Technology for Knowledge Sharing". In: *AI Magazine*, pp. 36–56.

Neches, R. et al. (1991b). "Enabling Technology for Knowledge Sharing". In: *AI Magazine* 12.3, pp. 36–56.

Negnevitsky, M. (2002). *Artificial Intelligence. A Guide to Intelligent Systems*. Addison-Wesley, Harlow.

Negnevitsky, M. (2005). *Artificial Intelligence A Guide to Intelligent Systems*. Pearson Education Canada.

Noy, N. and McGuinness, D. (2001). *Ontology development 101: A guide to creating your first ontology*. Tech. rep. Stanford University School of Medicine.

Nute, D. (1994). "Handbook of Logic for Artificial Intelligence and Logic Programming". In: ed. by D. Gabbay and C. Hogger. Vol. III. Oxford University Press. Chap. Defeasible logic, pp. 353–395.

Obrst, L. et al. (2007). "Semantic Web". In: ed. by C. Baker and K. Cheung. Springer. Chap. The Evaluation of Ontologies, pp. 139–158.

O'Connor, M. et al. (2005). "Supporting Rule System Interoperability on the Semantic Web with SWRL". In: *ISWC 2005: The Semantic Web-ISWC 2005*. Vol. 3729. LNCS, pp. 974–986.

O'Connor, M. et al. (2008). "Developing a Web-Based Application using OWL and SWRL". In: *AAAI Spring Symposium*. Stanford, CA, USA.

Ouksel, A. and Sheth, A. (1999). "Semantic interoperability in global information systems". In: *ACM SIGMOD Record*. Vol. 28, pp. 5–12.

Palmirani, M., Contissa, G., and Rubino, R. (2009). "Fill the Gap in the Legal Knowledge Modelling". In: *RuleML 2009*.

Palmirani, M., Ognibene, T., and Cervone, L. (2012). "Legal rules, text, and ontologies over time". In: *RuleML@ECAI 2012*.

Palmirani, M. et al. (2011). "LegalRuleML: XML-Based Rules and Norms". In: *RuleML America 2011*. Ed. by F. Olken, M. Palmirani, and D. Sottara. Vol. 7018. Lecture Notes in Computer Science. Springer, pp. 298–312.

Palmirani, M. et al. (2013). *RAWE: A Web Editor for Rule Markup in LegalRuleML*.

Pantel, P. and Lin, D. (2001). "A Statistical Corpus-Based Term Extractor". In: *Conference of the Canadian Society for Computational Studies of Intelligence. AI 2001: Advances in Artificial Intelligence*. Vol. 2056. Lecture Notes in Computer Science, pp. 36–46.

Parsia, B. and Sirin, E. (2004). "Pellet: An OWL-DL Reasoner". In: *The 3rd Int. Semantic Web Conference (ISWC 2004)*.

Paschke, A. (2014). "Rules on the Web. From Theory to Applications. RuleML 2014". In: ed. by A. Bikakis, P. Fodor, and D. Roman. Vol. 8620. Lecture Notes in Computer Science. Springer. Chap. Reaction RuleML 1.0 for Rules, Events and Actions in Semantic Complex Event Processing, pp. 1–21.

Patel, C. et al. (2003). "OntoKhoj: A Semantic Web Portal for Ontology Searching, Ranking and Classification". In: *5th ACM Int. Workshop on Web Information and Data Management*, pp. 58–61.

Patel, M. et al. (2005). *Semantic Interoperability in Digital Library Systems*. Tech. rep. UKOLN, University of Bath.

Pathak, J., Johnson, T., and Chute, C. (2009). "Modular ontology techniques and their applica-tions in the biomedical domain". In: *Integrated Computer-Aided Engineering* 16.3, pp. 225–242.

Pinto, S., Gomez-Perez, A., and Martins, J. (1999). "Some Issues on Ontology Integration". In: *IJCAI-99 Workshop on Ontologies and Problem-Solving Methods(KRR5)*.

Piovesan, L., Molino, G., and Terenziani, P. (2014). "An ontological knowledge and multiple abstraction level decision support system in healthcare". In: *Decision Analytics* 1.8.

Popp, W. and Schlink, B. (1975). "JUDITH, A COMPUTER PROGRAM TO ADVISE LAWYERS IN REASONING A CASE". In: *Jurimetrics Journal* 15.9, pp. 303–314.

Popple, J. (1990). "Legal expert systems: The inadequacy of a rule-based approach". In: *Thirteenth Australian Computer Science Conference (ACSC-13)*.

Popple, J. (1993). "SHYSTER: A Pragmatic Legal Expert System". PhD thesis. Australian National University.

Popple, J. (1996). *A pragmatic legal expert system*. Dartmouth Publishing Company.

Porzel, R. and Malaka, R. (2004). "A task-based approach for ontology evaluation". In: *ECAI 2004 Workshop Ont. Learning and Population*.

Prakken, H. and Sartor, G. (1996). "A dialectical model of assessing conflicting argument in legal reasoning". In: *Artificial Intelligence and Law* 4.3-4, pp. 331–368.

Puerta, A. et al. (1992). "A multiple-method knowledge-acquisition shell for the automatic generation of knowledge-acquisition tools". In: *Knowledge Acquisition* 4.2, pp. 171–196.

Rho, S. et al. (2009). "COMUS: Ontological and Rule-Based Reasoning for Music Recommendation System". In: *PAKDD 2009: Advances in Knowledge Discovery and Data Mining*, pp. 859–866.

Rodrigues, C., Freitas, F., and Azevedo, R. (2016). "An Ontology for Property Crime based on Events from UFO-B Foundational Ontology". In: *2016 5th Brazilian Conference on Intelligent Systems (BRACIS)*. IEEE.

Rogger, M. and Thaler, S. (2010). *Ontology Learning*. Seminar paper, Applied Ontology Engineering. University Innsbruck.

Rosa, D. et al. (2012). "Using events from UFO-B in an ontology collaborative construction environment." In: *CEUR-WSX 938*, pp. 278–283.

Rosati, R. (2006a). "DL+log: Tight Integration of Description Logics and Disjunctive Datalog". In: *The 10th International Conference on Principles of Knowledge Representation and Reasoning (KR 2006)*. Ed. by P. Doherty, J. Mylopoulos, and C. Welty. AAAI Press, pp. 68–78.

Rosati, R. (2006b). "Reasoning Web, Second International Summer School 2006, Tutorial Lectures". In: ed. by P. Barahona et al. Vol. 4126. Lecture Notes in Computer Science (LNCS). Springer. Chap. Integrating Ontologies and Rules: Semantic and Computational Issues, pp. 128–151.

Rossello-Busquet, A., Brewka, J., and Dittman, L. (2011). "OWL ontologies and SWRL rules applied to energy management". In: *International Conference on Modelling and Simulation*.

Roussey, C. et al. (2011). "Advanced Information and Knowledge Processing". In: Springer. Chap. An Introduction to Ontologies and Ontology Engineering, pp. 9–38.

Rudolph, S., Volker, J., and Hitzler, P. (2007). "Supporting lexical ontology learning by relational exploration". In: *Proceeding of ICCS*, pp. 488–491.

Sabou, M. et al. (2005). "Learning Domain Ontologies for Web Service Descriptions: an Experiment in Bioinformatics". In: *Proceedings of the 14th International World Wide Web Conference (WWW2005)*.

Saghafi, A. and Wand, Y. (2014). "Do Ontological Guidelines Improve Understandability of Conceptual Models? A Metaanalysis of Empirical Work". In: *In System Sciences (HICSS)*, pp. 4609–4618.

Sagri, M., Tiscornia, D., and Bertagna, F. (2004). *Jur-WordNet*.

Saias, J. and Quaresma, P. (2003). "Using NLP techniques to create legal ontologies in a logic programming based web information retrieval system". In: *ICAIL 2003 Workshop on Legal Ontologies & Web based legal information management*.

Saias, J. and Quaresma, P. (2005). "Law and the Semantic Web". In: vol. 3369. Lecture Notes in Computer Science. Springer, Verlag. Chap. A Methodology to Create Legal Ontologies in a Logic Programming Information Retrieval System, pp. 185–200.

Salton, G. and Buckley, C. (1988). "Term-weighting approaches in automatic text retrieval". In: *Information Processing and Management: an International Journal* 24.5, pp. 513–523.

Sanderson, M. and Croft, B. (1999). "Deriving concept hierarchies from text". In: *SIGIR 99 Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*.

Sartor, G. (2006). "Fundamental Legal Concepts: A Formal and Teleological Characterisation." In: *Artificial Intelligence and Law* 14.1-2, pp. 101–142.

Sartor, G. (2009). "Legal Concepts as Inferential Nodes and Ontological Categories". In: *Artif. Intell. Law* 17.3, pp. 217–251.

Sartor, S. and Branting, L. (1998). "Introduction: Judicial Applications of Artificial Intelligence". In: *Artificial Intelligence and Law*.

Sattler, U., Schneider, T., and Zakharyaschev, M. (2009). "Which Kind of Module Should I Extract?" In: *Proc. of DL*.

Savini, M. (2011). *JOHANNES CLAUBERG METHODUS CARTESIANA ET ONTOLOGIE*. VRIN.

Schlicht, A. and Stuckenschmidt, J. (2007). "Criteria-based partitioning of large ontologies". In: *Proceedings of the 4th International Conference on Knowledge Capture (K-CAP 2007)*. Ed. by D. H. S. K̃. Barker, pp. 171–172.

Schreiber, A., Wielinga, B., and Breuker, J., eds. (1993). *KADS. A Principled Approach to Knowledge-Based System Development, Knowledge-Based Systems*. Vol. 11. Academic Press, London.

Schreiber, G. and Akkermans, H. (2000). *Knowledge engineering and management: the CommonKADS methodology*. MIT Press.

Schubert, R. (2001). *Bones, Holes, and Scales - on the Need for a Spatial Ontology for Anatomy*.

Shamsfard, M. and Barforoush, A. (2004). "Learning ontologies from natural language texts". In: *International Journal of Human-Computer Studies* 60.1, pp. 17–63.

Simperl, E. and Rosch, M. (2013). "Collaborative ontology engineering: a survey". In: *The Knowledge Engineering Review* 29.1, pp. 101–131.

Sirin, E. and Parsia, B. (2007a). "SPARQL-DL: SPARQL Query for OWL-DL". In: *3rd OWL Experiences and Directions Workshop (OWLED-2007)*.

Sirin, E. et al. (2005). *Pellet: A practical OWL-DL reasoner*. Tech. rep. 68. UMIACS, University of Maryland.

Sirin, E. et al. (2007b). "Pellet: A practical OWL-DL reasoner". In: *Web Semantics* 5.2, pp. 51–53.

Skuce, D. (1995). "Conventions for reaching agreement on shared ontologies". In: *Know ledge Acquisition for Knowledge Based Systems Workshop*.

Smith, M., Welty, C., and McGuinness, D. (2004). *OWL Web Ontology Language Guide*. Tech. rep. W3C Recommendation 10 February 2004.

Sowa, J. (1984). *Conceptual Structures. Information Processing in Mind and Machine Reading*. MA: Addison Wesley.

Steels, L. (1990). "Components of Expertise". In: *AI Magazine* 11.2, pp. 29–49.

Steve, G., Gangemi, A., and Pisanelli, D. (1997). "Integrating medical terminologies with ONIONS methodology". In: *Information Modelling and Knowledge Bases VIII*.

Stuckenschmidt, H. and Klein, M. (2003). "Integrity and Change in Modular Ontologies". In: *18th International Joint Conference on Artificial Intelligence*, pp. 900–905.

Stuckenschmidt, H. and Klein, M. (2007). "Reasoning and change management in modular ontologies". In: *Data Knowledge Eng.* 63.2, pp. 200–223.

Stuckenschmidt, H., Parent, C., and Spaccapietra, S., eds. (2009). *Modular Ontologies - Concepts, Theories and Techniques for Knowledge Modularization*. Springer.

Studer, S., Benjamins, V., and Fensel, D. (1998). "Knowledge engineering: Principles and methods". In: *Data & Knowledge Engineering* 25, pp. 161–197.

Sure, Y., Staab, S., and Studer, R. (2003). "Handbook on Ontologies". In: ed. by S. Staab and R. Studer. Springer-Verlag, Berlin Heidelberg New York. Chap. On-To-Knowledge methodology, pp. 117–132.

Sure, Y. and Studer, R. (2002a). *On-to-knowledge methodology - final version. Project Deliverable D. 18*. Tech. rep. Institute AIFB, University of Karlsruhe.

Sure, Y. et al. (2002b). "OntoEdit: collaborative ontology engineering for the semantic web". In: *First International Semantic Web Conference (ISWC'02)*. Vol. 2342. Lecture Notes in Computer Science. Springer, Berlin, pp. 221–235.

Sure, Y. et al. (2004). *D.7.1.1 sekt methodology: Survey and initial framework*. SEKT IST-2003-506826 Deliverable 7.1.1, SEKT, EU-IST Project IST-2003-506826, Institute AIFB, University of Karlsruhe.

Susskind, R. (1986). "Expert Systems in Law: A Jurisprudential Approach to Artificial Intelligence and Legal Reasoning". In: *Modern Law Review*.

Swartout, B. et al. (1997). "Toward Distributed Use of Large-Scale Ontologies". In: *Ontological Engineering*, pp. 138–148.

Tartir, S., Arpinar, I., and Sheth, A. (2010). "Theory and Applications of Ontology: Computer Applications". In: ed. by R. Poli, M. Healy, and A. Kameas. Springer. Chap. Ontological Evaluation and Validation, pp. 115–130.

Teixeira, M., Falbo, R., and Guizzardi, G. (2014). "Analyzing the Behavior of Modelers in Interpreting Relationships in Conceptual Models: An Empirical Study". In: *ONTO.COM/ODISE@FOIS 2014.*

Tiscornia, D. (2005). "Multilingual Semantic Metadata for Law". In: *In Quaderni CNIPA, 2005, 3rd Workshop on Legislative XML*.

Tudorache, T. (2007). *Collaborative Ontology Development in Protégé*. Ontolog forum invited talk. URL: http://ontolog.cim3.net/file/resource/presentation/TaniaTudorache_20071004/CollaborativeProtege--TaniaTudorahce_20071004.pdf.

Turlapati, V. and Puligundla, S. (2013). "Knowledge Engineering and the Semantic Web". In: vol. 394. Communications in Computer and Information Science. Springer Berlin Heidelberg. Chap. Efficient module extraction for large ontologies. Pp. 162–176.

Uschold, M. (1996). "Building Ontologies: Towards a Unified Methodology". In: *16th Annual Conf. of the British Computer Society Specialist Group on Expert Systems*.

Uschold, M. and Gruninger, M. (1996). "Ontologies : Principles, methods and applications". In: *Knowledge Engineering Review* 11.2.

Uschold, M. and King, M. (1995). "Towards a Methodology for Building Ontologies". In: *IJCAI 95 Workshop on Basic Ontological Issues in Knowledge Sharing*, pp. 6.1–6.10.

Valente, A. (1995). *Legal Knowledge Engineering: A Modelling Approach*. IOS Press.

Valente, A. (2005). "Law and the Semantic Web". In: vol. 3369. LNCS. Springer. Chap. Types and Roles of Legal Ontologies, pp. 65–76.

Valente, A. and Breuker, J. (1991). "Law functions: Modeling principles in legal reasoning". In: *Legal Knowledge Based Systems, Model-based legal reasoning, JURIX 91*.

Valente, A. and Breuker, J. (1992). "A MODEL-BASED APPROACH TO LEGAL KNOWLEDGE ENGINEERING". In: *Legal knowledge based systems JURIX 92 Information Technology and Law*.

Valente, A. and Breuker, J. (1994a). "Ontologies: the Missing Link Between Legal Theory and AI & Law". In: *Jurix 94 Proceedings*, pp. 138–149.

Valente, A. and Breuker, J. (1994b). "Towards a global expert system in law". In: ed. by G. Bargellini and S. Binazzi. CEDAM Publishers. Chap. A functional ontology of law.

Valente, A. and Breuker, J. (1996). "Towards Principled Core Ontologies". In: *Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW 96)*. Ed. by B. Gaines and M. Musen, pp. 33/1–33/20.

Valente, A. and Breuker, J. (1999a). "Legal modeling and automated reasoning with on-line". In: *International Journal of Human-Computer Studies* 51.6, pp. 1079–1125.

Valente, A. et al. (1999b). "Building, Using and Reusing an Ontology of Air Campaign Planning". In: *IEEE Intelligent Systems* 14.1, pp. 27–36.

Van Gog, R. and Van Engers, T. (2001). "Modelling Legislation Using Natural Language". In: *2001 IEEE Systems Proceedings, Man and Cybernetics Conference*.

Van Kralingen, R. (1995). *Frame-based Conceptual Models of Statute Law*. Computer/Law. Kluwer Law International.

Van Kralingen, R., Visser, P., and Bench-Capon, T. (1999). "A principled approach to developing legal knowledge systems". In: *Int. J. Human-Computer Studies*, pp. 1127–1154.

Velardi, P. et al. (2005). "Ontology Learning from Texts: Methods, Applications and Evaluation". In: Frontiers in Artificial Itelligence and Applications 123. IOS Press. Chap. Evaluation of OntoLearn, a metodology for automatic population of domain ontologies, pp. 92–106.

Ven, S. et al. (2008). "Automated Legal Assessment in OWL 2". In: *Proceedings of Jurix 2008*. Ed. by E. Francesconi, G. Sartor, and D. Tiscorina. IOS Press, pp. 170–175.

Verdonck, M. (2014). "Providing guidance for conceptual modelling using core ontologies". In: *PhD symposium in 33rd International on Conceptual Modeling Conference (ER).*

Visser, P. (1995). *Knowledge Specification for Multiple Legal Tasks; A Case Study of the Interaction Problem in the Legal Domain, Computer/Law Series*. Kluwer Law International.

Vitali, F. and Zeni, F. (2007). "Towards a Country-Independent Data Format: The Akoma Ntoso Experience". In: *Proceedings of the V Legislative XML Workshop, 6786. European Press Academic Publishing*.

Von Wright, G. (1963). *Norm and Action*. Routledge, London.

Vrandecic, D. (2009). "Handbook on Ontologies". In: ed. by A. Staab and R. Studer. Springer Berlin Heidelberg. Chap. Ontology Evaluation, pp. 293–313.

Vrandecic, D. et al. (2005). "The diligent knowledge process". In: *Journal of Knowledge Management* 9.5, pp. 85–96.

Wache, H. et al. (2001). "Ontology-Based Integration of Information - A Survey of Existing Approaches". In: *IJCAI-01 Workshop: Ontologies and Information Sharing*. Ed. by H. Stuckenschmidt, pp. 108–117.

Wagner, G. et al. (2004). "The abstract syntax of RuleML - towards a general web rule language framework". In: *Proc. Web Intelligence 2004*. IEEE, pp. 628–631.

Wand, Y. (1996). "Ontology as a foundation for meta-modelling and method engineering". In: *Information and Software Technology* 38.4, pp. 281–287.

Wang, Y., Liu, W., and Bell, D. (2010). "A Concept Hierarchy Based Ontology Mapping Approach". In: *KSEM*, pp. 101–113.

Waterman, D. and Peterson, M. (1980). "Rule-Based Models of Legal Expertise". In: *Proc. First Nat'1 Conf. Artificial Intelligence, AAAI*.

White, M. (1992). "Legal Complexity and Lawyers' Benefit from Litigation". In: *International Review of Law and Economics* 12, pp. 381–395.

Wyner, A. (2008). "An ontology in OWL for legal case-based reasoning". In: *Artificial Intelligence and Law* 16.4, pp. 361–387.

Wyner, A. (2009). "An OWL Ontology for Legal Cases with an instantiaiton of Popov v. Hayashi". In: *Pre-conference workshop on Modelling Legal Cases at the 12th International Conference on AI and Law (ICAIL 2009)*.

Wyner, A. and Governatori, G. (2013). "A Study on Translating Regulatory Rules from Natural Language to Defeasible Logic". In: *Proceedings of RuleML 2013*. Seattle, WA.

Wyner, A. and Hoekstra, R. (2010). "A Legal Case OWL Ontology with an Instantiation of Popov v. Hayashi". In: *The Knowledge Engineering Review* 14.2, pp. 1–24.

Wyner, A. and Peters, W. (2011). "On Rule Extraction from Regulations". In: *JURIX 2011*.

Yang, D. et al. (2008). "Product configuration knowledge using ontology web language". In: *Journal of Expert Systems with Applications* 40, pp. 863–878.

Zamborlini, V., Goncalves, B., and Guizzardi, G. (2008). "Codification and Application of a Well-Founded Heart-ECG Ontology". In: *Third Workshop on Ontologies and Metamodeling in Software and Data Engineering - WOMSDE 2008*.

Zeleznikow, J. and Hunter, D. (1992). "Rationales for the Continued Development of Legal Expert Systems". In: *Journal of Law and Information Science*.

Zeleznikow, J. and Stranieri, A. (2001). "An Ontology for the Construction of Legal Decision Support Systems". In: *Second International Workshop on Legal Ontologies*.

Zeng, Y. et al. (2005). "Knowledge Representation for the Intelligent Legal Case Retrieval". In: *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems KES 2005, Lecture Notes in Computer Science book series (LNCS, volume 3681)*.

Zhang, X. et al. (2011). "Emotiono: An Ontology with Rule-Based Reasoning for Emotion Recognition". In: *ICONIP 2011: Neural Information Processing*, pp. 89–98.

Zhao, W. and Liu, J. (2008). "OWL/SWRL representation methodology for EXPRESS-driven product information model". In: *Computers in industry* 59, pp. 580–589.