Electronic Contracting for Inter-Enterprise Collaboration

María Pérez
This thesis is submitted to the Department of Interaction and System Design, School of Engineering at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Computer Science. The thesis is equivalent to 24 weeks of full time studies.

Contact Information:
Author(s):
Maria Perez
E-mail: maria.perez.sonora@gmail.com

University advisor(s):
Andreas Jacobsson
Paul Davidsson
Keith Clark
Department of Systems and Software Engineering

Department of Systems and Software Engineering
Blekinge Institute of Technology
SE – 372 25 Ronneby
Sweden

Internet : www.bth.se/tek
Phone : +46 457 38 50 00
Fax : + 46 457 102 45
ABSTRACT

A Virtual Enterprise (VE) is a temporal alliance between two or more Small and Medium Enterprises (SMEs) geographically dispersed, which collaborate together in order to reach new business opportunities that would be unreachable in other ways. It is a business collaboration paradigm that aims at responding to the uncertainty and instability of the current global economy. VE requires support for electronic contract management, since electronic contracts formalize the agreements between the participating enterprises and coordinate their behavior. Although there is an abundance of previous work on electronic contracts, there is a lack of models and approaches related to VE contracts, which have an intrinsic dynamic and flexible nature, since they regulate independent behavior of diverse parties, and also aim at high automation in the formation and execution. This thesis aims at contributing to the VE contracting challenge. It includes a state of the art survey that identifies useful technologies and describes the most significant or relevant approaches. The state of the art survey identifies three main contracting issues: Contract specification, which determines the structure, content and performance of the contract; deontic logic norms, which represent the contractual interactions between the parties in terms of obligations, prohibitions and permissions; and ontology, which provides contracts with semantic meaning and allows interoperability. Furthermore, a simple XML-based VE Contract Representation Language and a Layered Contract Ontology, which provides common vocabulary to the contractual parties, are presented. Finally a VE scenario, including the associated contract, is described as an illustrative example.
CONTENTS

Abstract ........................................................................................................................................... 1

Chapter 1: Background .................................................................................................................. 4

Chapter 2: Problem definition/Goals ......................................................................................... 5

Chapter 3: Methodology ............................................................................................................. 7
  3.1 Introduction ......................................................................................................................... 7
  3.2 Contract specification ......................................................................................................... 7
  3.3 Introduction ......................................................................................................................... 7

Chapter 4: State of the art in electronic contracting .............................................................. 9
  4.1 Introduction ......................................................................................................................... 9
  4.2 Contract specification ......................................................................................................... 10
    4.2.1 Related work on contract specification ....................................................................... 10
  4.3 Deontic logic and norms ................................................................................................... 12
  4.4 Ontologies ........................................................................................................................ 14
  4.5 Relevant approaches related to electronic contracts ....................................................... 16

Chapter 5: Representation language for contracting ........................................................... 19
  5.1 Overview ........................................................................................................................... 19
  5.2 Contract structure .............................................................................................................. 19
  5.3 Header .................................................................................................................................... 20
    5.3.1 Rolelist ............................................................................................................................ 20
    5.3.2 Participants .................................................................................................................... 20
    5.3.3 Duration .......................................................................................................................... 21
  5.4 The core of the contract ..................................................................................................... 22
    5.4.1 ServiceList ....................................................................................................................... 22
    5.4.2 Clauses ............................................................................................................................ 22
    5.4.3 Trusted Third Party ......................................................................................................... 24
  5.5 Contract example ................................................................................................................ 24

Chapter 6: Layered Contract Ontology ................................................................................... 29
  6.1 Layered Structure ............................................................................................................... 29
  6.2 Top Level Contract Ontology ........................................................................................... 29
    6.2.1 VE_Contract ..................................................................................................................... 30
    6.2.2 Enterprise ....................................................................................................................... 30
    6.2.3 Participant_Enterprise ..................................................................................................... 30
    6.2.4 Role .................................................................................................................................. 31
    6.2.5 Deontic_Assignment ....................................................................................................... 31
    6.2.6 Obligation ........................................................................................................................ 31
    6.2.7 Permission ....................................................................................................................... 31
    6.2.8 Prohibition ....................................................................................................................... 32
    6.2.9 Clause_State ..................................................................................................................... 32
    6.2.10 Committed ...................................................................................................................... 32
    6.2.11 Fulfilled ........................................................................................................................ 32
    6.2.12 Not_Fulfilled ................................................................................................................. 32
    6.2.13 Activating_Condition ................................................................................................. 32
6.2.14 Action................................................................. 33
6.2.15 Contact............................................................. 33
6.2.16 Third_Party.......................................................... 33
6.3 Domain Level Contract.................................................. 34
   6.3.1 Seller............................................................... 34
   6.3.2 Buyer............................................................... 34
   6.3.3 Deliver............................................................. 34
   6.3.4 Pay................................................................. 35
   6.3.5 Product............................................................. 35

Chapter 7: Validation......................................................... 36

Chapter 8: Discussion......................................................... 45

Chapter 9: Results and conclusions........................................ 38

References................................................................. 47

Appendix........................................................................ 51
1. **BACKGROUND**

Due to the instability, turbulence, aggressiveness and globalization of the current economy, enterprises (especially Small and Medium Enterprises (SMEs)) require new collaboration models and to look for temporal alliances between each other in order to increase their competence and survive in the business market [41]. This phenomenon has provoked the creation of a business collaboration paradigm: the Virtual Enterprise, which is a temporal, flexible and dynamic alliance between SMEs geographically dispersed, supported by the latest ICT technologies and aiming at new business opportunities that are unreachable in other ways [4].

VEs allow geographically dispersed enterprises to use each other’s competencies and complement each other. Therefore they allow producing best quality results spending less time and less money [42]. VEs can be characterized in ways ranging from simple subcontracting networks and supply chains to dynamic alliances of independent enterprises sharing resources between each other [4].

This VE business collaboration model has a promising relevance in the near future, since it had been predicted that in the next ten years, almost all SMEs will be part of a business collaboration network in order to set up alliances between each other and establish VEs [32].
2. PROBLEM DEFINITION/GOALS

VEs aspire high automation in their whole life cycle; that is in their definition, formation, operation and termination. The first step in the VE life cycle is always an arising of a business opportunity, the selection of suitable participants to achieve that business opportunity, assigning roles to the participants. For instead, a VE may be formed to cover a market demand of a certain product, hence factories of that product will be selected to play the producer role; and stores will be selected to play the retailer role. The next stage consists of the negotiation of the terms and conditions of the alliance, such as dates, payments, amount of product to produce, etc. This process is named contract negotiation and it concludes with the contract establishment. Once the contract is established and signed by all the participants, the operation stage starts and the participants begin to carry out the processes specified in the contract [4].

It is easy to observe that contracts are crucial and essential elements for VE performance. Contract establishment is the main goal during the formation stage and contracts regulate and control VE performance during the operation stage. As a consequence, an approach that aims at high VE performance automation should begin to consider electronic VE contracts automation.

Electronic contracts have been used to enable new business relationship models, to optimize and globalization existing business relationships and to enable contractual relationships with less cost, less time and less restrictions. For that reason, there exists already a large research on electronic contracting, especially in the business-to-business area. There is also a considerable amount of literature related to the VE concept and to VE technological support. However, there is a lack of formal models and frameworks for VE contract support [5].

Existing electronic contract approaches are not enough for VE contract support, since they have special characteristics and specific dynamism and flexibility requirements. Therefore new contractual frameworks and models specially tailored for VE need to be designed.

VE contracts are electronic documents that should be managed and negotiated by several diverse parties with different features and different characteristics; therefore enabling interoperability; that is, to allow the diverse parties to work together, is one of the main challenges for VE contracting. Furthermore, VE contracts do not deal with a fixed domain, but should be domain independent and should be able to adapt to different domains and different situations. For example, a VE contract within a cookies sale and purchase VE should be able to support clauses that deal with concepts such as the cookies ingredients and cookies production, however, the clauses of a contract within a construction company VE should deal with concepts such as buildings and lands [27]. Moreover, a VE should represent the reality of the business alliances, therefore VE contracts can never be static and rigid, but, as mentioned before, they should be dynamic and flexible. In fact, they should consider the possibility of parties deciding not to fulfill contract clauses, and they should enable new participants to join or leave the alliance at runtime, and even should enable modification of contractual clauses during the VE operation [5].

This research aims at contributing to the development of a representation contractual framework for VEs, which meet mentioned requirements. The thesis consists of a comprehensive theoretical study about the existing solutions and approaches related to electronic contracting, and an analysis of which of these
solutions and approaches can be useful for VE contract representation and how can they be applied. Therefore the research questions that this study tries to address are the following:

- Which is the current level of development and the main approaches and solutions in the electronic contracting area?
- Which of these approaches and solutions can be applied in VE contract representation?
- How can these approaches and solutions be applied in VE contracts?

These questions were answered by means of a literature study and analysis, and proposing a VE Contract Representation Language model and a VE Contract Ontology.
3. **Methodology**

This section addresses the methodology chosen to answer the presented research questions. This methodology consists of three main steps: collection of information, analysis of the information collected, and knowledge application.

3.1. **Collection of Information**

The study starts with the identification of significant literature and research. The main and more relevant projects, books and articles about electronic contracting and VE are selected and a comprehensive review is carried out. The main goal of this literature and project review is to provide data and information to be analyzed, but it also provides a deep theoretical background and contributes to improve the understandability of the study area.

3.2. **Information Analysis**

The literature and project review provides plenty of information that needs to be analyzed and interpreted to become useful knowledge. The information is divided and analyzed separately. The information about electronic contracting is used to provide knowledge about the level of development in e-contracting technologies; that is, about relevant e-contracting technologies, models and approaches, and about the functionalities and advantages of each approach, as well their lacks and weaknesses. However, the information about VE contracting is used to identify the special features and requirements of VE contracts.

3.3. **Knowledge Application**

Once that knowledge is obtained by means of information collection and analysis, it is time to apply it. In this study, the knowledge will be used to generate three different results or products: a survey on the state-of-the-art in electronic contracting, a VE Contract Representation Language and a VE Contract Ontology.

The state-of-the-art survey on electronic contracting is a theoretical document that addresses the conclusions of the information collection and analysis processes. Therefore it identifies the most relevant technologies in the electronic contracting area and their characteristics, and finds out which of these technologies that can be applied to satisfy the VE contracts requirements.

Moreover, the study is expected to provide a Contract Representation Language specially tailored for VEs, which takes into account some of the most significant approaches identified in the analysis, and a VE Contract Ontology that provides semantic meaning to the Contract Representation Language as well as interoperability and domain independence. Note that both, the VE Contract Representation Language and the VE Contract Ontology are simple approaches concentrated on the main elements, since their scope is to identify suitable technologies for VE contracting, how they can be applied and how can they satisfy the special VE requirements. As a consequence, they do not cover all the issues and challenges related to VE electronic contracting, which could be the scope for future and larger projects. Furthermore both the VE Contract Representation Language and the VE Contract Ontology are complemented with examples and a VE scenario that illustrate the approach and prove its applicability in real situations.
Once the knowledge application process is finished, the generated results or products are expected to answer the research questions that motivate this study. In fact, the state-of-the-art is expected to provide answers for the first two questions: Which is the current level of development and the main approaches and solutions in the electronic contracting area? And which of these approaches and solutions can be applied in VE contract representation? Whereas the VE Contract Representation Language and the VE Contract Ontology is expected to answer the last one: how can these approaches and solutions be applied in VE contracts?

Finally, an explanatory graph is enclosed to illustrate the proposed methodology, the relationships between the different processes and the output and input:
4. STATE OF THE ART IN ELECTRONIC CONTRACTING

4.1. INTRODUCTION

A contract, in general terms, can be defined as a legal and enforceable agreement between two or more parties, in which the parties commit to fulfill certain obligations with the purpose of receiving certain rights in return[1].

Since the ancient times, contracts have been used to formalize diverse types of relationships, specially, those related to business and trade. In the current age of information and technology, the traditional businesses and trades are combined with the latest technologies. As a consequence, new models and methods of business, which make an intensive use of communication networks and other technologies, are arising [2].

VEs are collaboration models between enterprises that prove this trend. These collaboration models make use of computer networks, web services and diverse new computing technologies, in order to perform efficient and flexible business between enterprises that could be geographically dispersed [3]. Electronic contracting is one of the main pillars that support these new business models and these collaborations between enterprises. It aims to improve and support traditional contracting.

An electronic contract is a virtual representation of a traditional contract [2]. It can be a plain digitalization (“shallow e-contracting”) or a more complex, dynamic and flexible structure with much more potential (“deep e-contracting”). The most recent research in “deep e-contracting”, aims for the high automation in the contract establishment, contract management and contract fulfillment. This automation represents new possibilities and advantages for the contracted parties [1].

A VE is a temporal alliance between two or more Small and Medium Enterprises (SMEs), supported by computer networks and Information Communication Technology (ICT), that aims to reach new business opportunities. For example, a group of small airlines, each one offering a reduced number of routes, may collaborate together and form an airline VE, which will offer more routes and will be more competitive in the airline market; or, in order to exploit a touristic destination, a hotel chain may collaborate with an airline and a travel agency, offering the clients whole touristic packages. It is a promising and attractive business collaboration model, since, when an opportunity arises, a specifically tailored VE is rapidly formed. Moreover, it covers the demand of flexible and agile approaches of the competitive and changeable current market [4].

VE contracts are frequently related to the legal concept of “consortium contract”, which is used to define collaboration agreements between several partners [6]. Although there is an abundance of research and information about plain electronic contracts that regulate resource exchange, little literature is dedicated to the complex electronic contracts used in VE formation and operation. Due to the lack of research and their complex nature, special attention to these contracts is needed [5].

The multi-agent technology has a main importance in the vast majority of the approaches that aim for implementation of VE processes. Multi-agent technology may provide an automatic per-order VE formation based on electronic contracting, which,
compared to traditional business alliances negotiated by humans, saves money and
time, since computational agents are frequently more effective in finding beneficial
contracts [19]. These approaches conceive a VE as a group of intelligent agents, each
one acting on behalf of one enterprise, that collaborate to achieve a goal, and whose
behavior is regulated by electronic contracts [4]. Therefore, from the VE point of
view, an electronic contract is a formal regulation of the behavior of a group of agents
that work together to carry on a business activity [5].

4.2. CONTRACT SPECIFICATION

Electronic contracts need a previously settled format and structure, designed to
provide fixed support for automatic negotiation, establishment and enhancement. This
structure is necessary to organize the information in the contract and to facilitate the
creation, processing and maintenance of the contract [1].

The content of an automatic electronic contract should include: Identification of all
the participants, specification of the involved services and products, description of the
activities that each participant should complete, and time and precedence constraints
[5]. Contracts usually have a set of roles to assist task allocation in typified business
relationships, and a group of clauses, which form the core of the contract and regulate
the interactions between the agents [2]. For example, consider a VE set up by an
airline, a hotel chain and a travel agency. The corresponding contract should include
the airline, hotel chain and travel agency details, such as name of the company, contact
person, etc; and the services and activities that each company will carry out in terms of
roles and clauses; that is, the airline will play the hauler role and will provide go and
return flights to the destination; the hotel chain will play the accommodation role,
providing accommodation; and the travel agency will play the retailer role and will sell
the travel packages to the clients.

To implement all these issues, languages and tools that meet the requirements
imposed by both technology and business are used. Angelov et al present in [1] an in-
depth study about the requirements on an e-contract specification language based on
the 4W e-contracting framework [30]. Among other requirements, it is stated that an
efficient e-contract specification language should include at least three different
constructs: data constructs, rule constructs, and process constructs; should allow the
specification of the information to be exchanged and the activities to be performed;
and should provide mechanisms to specify and manage changes in the contract
elements. Furthermore, a specification e-contract language should provide a contract
structure and a semantic agreed between the parties [1].

Concerning VE contract specification, it has to consider the particular
characteristics and requirements of this kind of business agreement. Specifically, VE
contracts should be dynamic and flexible, in order to coordinate several parties and
support the integration and expulsion of parties at runtime, oriented to manage cross-
organizational workflow and close collaboration between the parties, and oriented to
represent ongoing relationships. Cardoso et al discuss in [5] some of these
requirements.

4.2.1.RELATED WORK ON CONTRACT SPECIFICATION

In the following, existing approaches related to electronic contract specification
are summarized. The frequent use of XML in most of the approaches is noteworthy.
This is due to the advantages that XML offers. It allows for simple document
processing, easy internet integration and plain language standardization [1]. Some
previous contract specification languages have been already proposed, for example
TpaML or BCL. In addition, the Sweetdeal project can be mentioned as a relevant project about contract specification.

The Trading Partner Agreement Mark-up Language (TpaML) is an Extensible Markup Language (XML) for contract specification submitted by IBM (International Business Machines) to OASIS (Organization for the Advancement of Structured Information Standards). The Trading Partner Agreement (TPA) is the main concept of the approach, which is an electronic contract that specifies the security and communication infrastructure, and the interactions between the parties involved in the agreement. TpaML also provides several tools and services to assist the TPAs’ establishment and execution monitoring [7]. Therefore it may be a useful tool for contract specification. However, in order to provide implementation indepndency, it omits a proper definition for the internal processes related to the agreement [8].

The Business Contract Language (BCL) is an XML-based language developed to support the contract’s semantics. It is the result of the combination of Enterprise Contract Level (ECL), which aims at providing contract monitoring with sophisticated support; and the Business Contract Architecture (BCA), which aims at providing support for the whole contract lifecycle in a flexible and adaptable manner [9]. The purpose of the BCL is to specify Electronic Contracts in a way that allows automatic management, and specifically, contract monitoring and enforcement based on the event concept (an event can be an action of a party, a temporal occurrence, a change of state, or a contract violation). BCL is an event-driven language that also covers concepts related to communities and policies, which define behavioral constraints and structural aspects, as the definition of the clauses and sub-clauses that compose the contracts [10, 11].

The SweetDeal project is an approach that combines RuleML and ontologies. It intends to represent business contracts that can be automatically created, evaluated, negotiated and executed by software agents [12]. The relation between the elements of the contract and its semantic, and the representation of contract rules in a way that allows automatic processing and interpretation are remarkable aspects in SweetDeal that can be used as inspiration for future approaches. Nevertheless, the project also has weaknesses, such as the lack of comprehensive support for process specification and all rule types. Furthermore, there is not any specification of contract structure [1].

Two relevant approaches concerning business standardization are ebXML and RossetaNet, which are frameworks that assist the development of electronic business systems and ensure participants compatibility [36]. Both are also successful standards and reference models in industry; and both provide digital structures for the specification of business interactions between partners, which may be inspirational for a contract specification development.

The ebXML standardization consists of an exhaustive XML-based specification for the main B2B issues. Especially relevant to electronic contracting is the partnership profile and agreement specification, which basically specifies two types of XML-documents: Collaboration Partner Profiles (CPPs) and Collaboration Partner Agreements (CPAs). An ebXML CPP describes the details, capacities and offered services of a certain organization, so that other partner can access to this information, interact with the organization, and maybe agree to carry out a business relationship. In this case a CPA, which is the result of merging the CPPs of both parties, is formed. CPAs can be considered as electronic contracts between two trading partners, since they are documents that describe an agreement between two parties, specifying the business processes and governing their execution. The ebXML’s Business Process Specification (BPPS) is also noteworthy and closely related to the CPP and CPA.
specification, since it is used to define the interactions between parties carrying out a business process [35].

RossetaNet is also a standardization effort, whose main contribution is a common language for business processes that allows business process automation. The main elements of this language are the Partner Interface Processes (PIPs), which are specialized XML-based dialogs between systems. PIPs define business processes between trading partners and specify the corresponding business process automation in terms of business documents, business document sequences, and quality of service [26].

Nevertheless, a specific E-Contract specification language, specially tailored for VE is still missing, although there are some efforts in that direction, for example the CrossFlow project.

The CrossFlow project aims at providing automatic support for Virtual Organization focusing on contracts that specify the interactions between the parties in a high level, whereas low-level issues are undefined and left open. Crossflow defines an e-contract specification language based on XML that provides the constructs to be used in the structure of the Crossflow contracts. This e-contract specification language is mainly focused on data and process specification, allowing high automatic process management. However, other issues such as the specification of rules, constraints, rights and duties are hardly supported. Therefore the contracts specified by this language may be efficient for describing the VE performance, but they are not useful as legal documents [1].

Finally, this summary about related work is completed by mentioning approaches related to rules specification, such as RuleML and Prolog, since rule specification may be necessary when specifying iterations between partners in an e-contract. The Rule Markup Language (RuleML) is an emerging standard that aims at an open and XML-based rule representation language [14], and Prolog is a declarative language for logic programming, used often for rule specification in contracts. It is also worth mentioning the work about business rules specification in [15]. This research analyzes rules representation requirements, and presents a representational approach, which includes a knowledge representation formalism called Courteous Logic Programs (CLP) and a XML encoding for rules called Business Rules Markup Language [15].

4.3. DEONTIC LOGIC AND NORMS

Norms can be defined as rules used in artificial societies of agents to regulate agents, avoid malicious behavior, and provide coordination, security and trust [16]. Including normative systems (sets of behavioral norms applied to a group of agents) in VEs is an outstanding issue related to the control of agents, and thereby related to the formation and operation of VEs [4].

In the context of VEs as normative systems, electronic contracts are also based in normative concepts and normative statements, and specification of behavioral norms is included in the contract clauses. When two or more agents establish a contract between each other, the new norms inside the contract come into play determining the behavior of the contractual agents. Frequently, Deontic Logic, the logic of the normative concepts, is used to model the contract norms. According to the Deontic Logic approach, three different types of norms can be found inside a contract: obligations, permissions and prohibitions. Obligation implies that an agent must fulfill a specific action. Obligations usually include a temporal or precedence constraint and a sanction, since obligations in VEs are not absolute. Permission implies that an agent has the
right of perform a specific action if necessary. Prohibition implies that an agent is not allowed to perform a specific action. Prohibitions are usually modeled as negated permissions, that is, a prohibition means that an agent has not the permission to do such action [5]. Therefore, contracts create obligations, permissions and prohibitions among agents, in other words, they introduce new normative relationships between agents. With normative statements based on deontic logic, it is possible to model almost all kind of electronic contracts [16]. For example, in the context of the VE alliance between the airline, the travel agency and the hotel chain described before, the contract will probably contain clauses representing the following deontic statements: the travel agency is permitted to sell the airline flights and reserve rooms in the hotels of the hotel chain, the airline is obliged to provide go and return flights to the destination, the hotel chain is obliged to provide accommodation, the travel agency is obliged to pay the airline and the hotel chain for their services, etc.

Sanctions are also relevant, mainly in approaches focused on high-automation contract formation, or multi-agent system approaches based on the autonomous agent paradigm. As mentioned before, obligations in electronic contracts are often not absolute. They depend on the associated sanctions. This means that an agent can choose not to fulfill an obligation, but in that case a certain sanction will be applied [16]. For instead, the airline can decide not to fly to the destination on a certain date, probably due the high risk of storms and accidents. In that case the airline is breaking an obligation and therefore, a sanction (maybe to pay a monetary compensation to the travel agency) should be applied.

Several recent research papers and approaches outline the benefits of imposing sanctions. Especially significant is the decommitting contracting mechanism that establishes Leveled Commitment Contracts. This mechanism allows the agents to reason about future events and if necessary to unilaterally decommit the obligations of the contract at any moment. Leveled Commitment Contracts include in their content, a specification of the decommitment penalties for the agents, therefore, once the contract is established, these penalties can be used to choose a certain level of commitment. Contrary to full commitment contracts, the leveled commitment feature allows accommodating properly the future events and establishing locally reasonable contracts. Therefore it increases the payoff as a consequence [18, 19].

Another outstanding issue in normative VEs approaches is the concept of Electronic Institution (EI). Inside a VE, the Electronic Institution is the entity in charge of regulating the interactions between the agents and providing the necessary level of trust. To do so, it includes a contractual framework with a reference normative system for contractual relationships and iterations between enterprises (agents) [5]. The relation between the Electronic Institution and the contracts is hierarchical. The EI imposes norms to the contractual relationships, and the contracts use the EI as support and can inherit some of its norms [16].

An illustrative approach about norms in contracts is the one presented in [5]. In this paper, Cardoso et al present and study a normative framework for validation an enforcement of electronic contracts focused on the contractual necessities of VEs. The framework contains a norm hierarchy of three levels: institutional, constitutional and operational. Institutional norms are the norms that validate contracts, regulate generic contractual tasks and solve any other issue that is not specified in lower levels. Constitutional norms express cooperation agreements and are used to impose and control complicity with operational contracts. These are the norms that, in the electronic contract, regulate general issues concerning the contractual relationship, such as the duration, the rules for entrance and exit and other cooperation terms. Operational norms are the norms that specify the actions that each of the contractual
party should realize, these norms can be followed and imposed during the VE operation phase [5].

4.4. ONTOLOGIES

Ontology is a formal explicit description of concepts in a domain [21], which provides standardized representation and common understanding for that domain [22]. Nowadays many research on ontology development are being carried out and ontology based solutions are common in artificial intelligent projects. This is due to the fact that ontologies are useful for many reasons: they provide interoperability, since they allow independent actors to understand each other; they provide reusability, since once an ontology for a certain domain has been developed, the rest of the researchers working in that domain can reuse that ontology if needed; they facilitate the development of independent domain applications, which can be applied in different domains, since they separate the domain knowledge from the operational knowledge; they allow to make domain assumptions, which are easy to implement, understand and change, instead of traditional hard-programming assumptions; and they are a good tool for introducing new users into a domain area. In addition, it is worth mentioning that ontologies and their associated knowledge bases are frequently used as data and semantic providers by software agents, applications and problem-solving methods [21].

From a VE development (and specifically, contract management) point of view, ontology-based solutions are also important. VE contracts have to deal with different users and methodologies, each one with its own terminology and interpretation of the domain, and with information that has different origins and different formats [23]. For the support of these requirements, ontologies provide common interpretation of the terminology, support for interoperability between the diverse participants of the VE, foundations for the development of data bases and new functionalities, and a base for normalization [22].

Ontology ensures that all the parties committing a contract have the same understanding about the contractual terms. For example, if the owner of a house hires a painter committing to a contract that establishes that the painter should paint the walls of the house, it is crucial for the painter and the owner to have the same idea about the meaning of the term walls of the house and the action paint the walls of the house. Otherwise misunderstanding will lead to an unsatisfactory result and a failure of the contracted agreement [40].

An ontology can be just a simple vocabulary that defines the terms required in a certain domain, or a complex ontology that provides in addition constraints to the terms and relations between components. Nevertheless, almost all formal ontologies consist of classes, slots, facets and instances as main elements. Classes or concepts are the central elements of most ontologies. They represent the main concepts of the domain, structured in a hierarchy. Slots or properties are the properties that characterize the classes of the domain. Facets or role restrictions are constraints to the slots. Instances are representations of individual elements of the real life. Instances of a class are defined by filling the slots of the class with specific values. Frequently, an ontology that includes a set of instances is considered to be a knowledge base; nevertheless, the concepts of knowledge base and ontology are closely related and barely separated [21].

As mentioned before, reusability is a relevant concept related to ontology, since ontologies are developed to be used and reused. Therefore, when designing a new ontology in a domain area, it is especially important to consider previous ontologies
and ontology-based projects in order to reuse them. Furthermore, the previous ontologies in the domain can be modified, extended and even combined with each other, to become a new ontology specially tailored for the new requirements.

Although little research has been done to model the semantics of contracts in the form of ontology, some contractual ontologies can be found in the literature. It is worth mentioning the Multi Tier Contract Ontology presented in [27], which is in fact a structured layered framework consisting of three layers: Upper level Contract Ontology that defines the essential elements in every contract type; Specific Domain Level Contract Ontology that defines more precise terms related to the specific contract type; and Template Level Contract Ontology that defines standard contractual obligations and their associated fulfillment patterns. This layered framework allows defining ontologies specially tailored for each situation and requirements, since the Specific Domain Level Contract Ontology and the Template Level Contract Ontology extend the Upper Level Contract Ontology with the specific domain and template terms required in each situation [27].

Concerning the development of an ontology for VE contracts, business-oriented ontologies and enterprise ontologies can be particularly useful, since they can provide relevant classes and concepts of the business domain. Although there are not many ontologies defining this area, following describes some of the most representative business ontologies:

The Toronto Virtual Enterprise’s ontology (TOVE) provides a formal representation for the enterprise domain. It was developed in the scope of the TOVE project, whose goal was to provide a set of ontologies for the modeling of enterprises. It is formalized in first order logic, and it consists of a set of terms defining the main enterprise concepts, and a set of axioms that enrich and constrain the term interpretations, allowing answering questions. The concepts are grouped in six top-level concepts that segment the enterprise into general categories and group the rest of the concepts: Activity, States, Causality, Time, Resources and Organizational structure [22].

The AIAI enterprise ontology was developed in the scope of the Enterprise Project, which aimed at developing tools for business modeling. It provides definition for the most frequent terms in the enterprise domain, which are grouped into five top-level classes: activity and processes class, which models activities and processes, including resources and skills; organization class, which includes the terms representing the enterprise actors and their responsibilities; strategy class, which represents purposes; marketing class, which describes sales; and time class. The AIAI enterprise ontology does not define specific business terms; therefore, when used in specific business applications, it has to be enriched [22].

The Resource-Event-Agent (REA) ontology is an enterprise domain ontology proposed for ontology-based enterprise systems development. REA was originally an account data model, that was progressively extended; first into an enterprise information architecture, and finally into the current enterprise ontology. It is also an event ontology, which means it relays on the definition of business events, including involved agents, affected resources and regulating policies. REA is frequently used as a reference for modeling business processes and business chains and it has been used in several standardization efforts for e-collaboration systems. However, REA has an inconvenient lack of adequate formal representation, even though it has a comprehensive theoretical framework related to events, accounting and micro economy. Due to this lack, different formats and conceptualizations of the REA
ontology can be found in the literature, which causes ambiguity and imprecision in definitions and interpretations [24].

The Business Process Management Ontology (BPMO) is an ontology written in OWL whose goal is to improve the technological support of businesses providing exhaustive semantic definitions of business processes. BPMO allows defining several business concepts following the UN/CEFAT Modeling Methodology (UMM) for business process and information modeling. Therefore it provides support for the definition of business entities and business objects for the process modeling, as well other concepts definitions as services, business processes and business collaborations. In addition, BPMO also allows defining tasks using the notion of Process Task Concept Type. In this way tasks are defined and related with the role that performs the task, the associated entities and businesses, and the resources that the task consumes [22].

Apart from enterprise ontologies, policy ontologies have been also identified as potential reusability candidates for contractual ontologies development. Two relevant examples of this ontology type are the Rei Policy Ontology and the KAos Policy Ontologies (KPO). The Rei Policy Ontology was developed in the scope of the Rei policy framework, whose goal was to provide support for the specification, analysis and reasoning of policies. It relays on the Rei Policy Ontology, which is an application independent ontology oriented to policies representation. It is based on deontic logic concepts, therefore the main represented concepts in the Rei Policy Ontology are the deontic concepts of obligation, permission and prohibition, although it also incorporates other important concepts such as policy rule or action [26]. The Kaos Policy Ontologies are just DAM+OIL representations of policies developed in the scope of the Kaos architecture for software agents [26].

Furthermore, if setting the defined concepts in a general background is required, top-level or upper ontologies may be also considered. This type of ontology is domain independent and defines very generic terms including almost everything. Some relevant upper ontology examples are: SUMO, Mikrokosmos, OpenCyc and Sowa’s top-level ontology [22].

Finally, it is worth mentioning other valuable projects related to semantic and ontology, such as the Core Enterprise Ontology (CEO) project, which is part of the BORO program and aims at improving the semantics of the enterprise information systems providing a core enterprise ontology [24]; and the Sweetdeal project, which is a worthy approach due to its investigations about the connections between e-contract elements and ontologies [1]. Furthermore, Sweetdeal has developed a process ontology and contract ontology to provide semantic support to businesses processes and contracts [12].

4.5. RELEVANT APPROACHES RELATED TO ELECTRONIC CONTRACTS

To complete this state of the art in electronic contracting, the most significant approaches, which may be used as an inspiration in the development of VE contracts support, are described:

First of all is worth to mention a conceptual framework that may be useful when specifying constraints and requirements in contractual systems: the 4w framework, which is a conceptual framework that provides a general and abstract vision of the business-to-business (b2b) contract concepts, including content specification concepts and contracting processes concepts [30].
The Business Contract Architecture (BCA) is also noteworthy. It was one of the first approaches trying to provide exhaustive support to automatic contract management. The relevance of the BCA lays in the dynamism and flexibility of its contracts, properties that turn BCA into a suitable approach to consider when developing support for VE contracts. This is due to the BCA Contracts’ capability of easily modifying their clauses just by adding references to new reusable clauses, and also due to the capability of setting up agreements in a simple and fast manner, since BCA has a repository where standard contracts (that can be used as contract templates) and reusable clauses can be stored to be retrieved later. Furthermore BCA allows contracts to be digitally signed and provides special support for contract enforcement and business process monitoring [28].

ForEV (Virtual Enterprise Formation) is an agent-based platform developed to provide support to VEs focusing in the formation stage; that is, in the contract negotiation and commitment processes. Main processes of this platform are the negotiation protocol and the partner’s selection automatic process that it provides. The negotiation protocol is called q-negotiation algorithm and it is an iterative, adaptive, multi-attribute algorithm using qualitative argumentation, capable of selecting the proposal with the most favorable conditions [2].

The Cosmos project provides a set of services, which facilitate e-contract management, focusing in the aspects of e-contracting that can be automated efficiently. It provides a basic architecture and a outlining meta-model for contract negotiation, amongst other issues [29].

The E-Alliance project is a software infrastructure oriented towards the management of flexible alliances. It provides support for formation of alliances, negotiation formalization, contract establishment, and for the maintenance of the alliance during all the contract life cycle. The main contribution of this approach is the autonomy of the parties involved in the alliance, since they can make independent decisions, and even decide to join or leave the alliance if needed [31].

TrustCoM is a framework for contract management for Virtual Organizations (VO) that evolves dynamically, whose most remarkable property is the emphasis in trust and security. Its main component is a service-oriented reference architecture for trust and contract management, which allows dynamic Virtual Organizations collaborations and implements the latest solutions, related to policies and distributed computing. [34].

Finally the ECOLEAD and CONTRACT project are described, since both are recent and significant projects related to contracts and business collaborations. The ECOLEAD (European Collaborative network Organizations LEADership) project considers that in the next ten years, in order to adapt to the fast changing market conditions, most enterprises will be part of a collaboration network, looking forward to set up alliances with other enterprises and form VEs. As a consequence, ECOLEAD aims at providing foundations and mechanisms for establishing advanced web-based collaboration between industries in Europe. ECOLEAD focuses in three main pillars: VO Breeding Environments, Dynamic VOs, and professional Virtual Communities. The project deal with plenty of the main issues related to business collaboration. Some relevant results are: fast VO formation derived from an effective exploitation of business opportunities; e-services supporting enterprise alliance establishment and operation, a contract negotiation protocol that rapidly defines the rights and duties of the involved parties, and a breeding environment platform. Moreover, ECOLEAS presents reference models, guidelines and new paradigms related to business collaboration and collaborative networks [32].
The CONTRACT project develops frameworks, components and tools for modeling, implementing, verifying and monitoring distributed e-business systems. It is focused on inter-organizational electronic contracts, which describe the behavior of each individual service and the whole system. The main aspects covered by this project are: specification of b2b interactions in e-contracts; dynamic establishment and management of contracts at runtime; and contract verification and monitoring techniques. Furthermore CONTRACT provides theoretical models and a reusable contracting language specification [33].
5. REPRESENTATION LANGUAGE FOR CONTRACTING

5.1. OVERVIEW

This thesis presents a simple human comprehensive and at the same time computer enforceable Contract Representation Language for Virtual Enterprises. This solution is based in the specification of Electronic Trading-Partner Agreements for E-Commerce (TPAs) [37], and it also incorporates deontic logic concepts among others.

This design has been elaborated looking for a simple, easy to understand and easy to implement solution, and keeping in mind the dynamism and flexibility requirements of the VE contracts. It intends to be a pre-study, which identifies the most important issues and requirements of the VE electronic contracts, as well as the main contracting elements and most suitable technologies and solutions.

The contracts that this specification defines are electronic XML documents. XML is a declarative language that offers plenty of advantages, which make it a good candidate for the VE contract representation. It allows easy interoperability and www-world integration. It is easy to parse, generate, edit and translate due to the variety of XML tools. Although they have been hardly used in this specification, hyper-text (links) aspects can be useful, mainly in future and more complex specifications, which would be oriented towards contract execution and validation. Furthermore XML has available diverse ontologies in XML language (including the e-commerce domain), and there are several e-commerce components that speak XML. As a conclusion, XML-based approaches are very relevant for electronic contracting [15].

5.2. CONTRACT STRUCTURE

The proposed contracts start with the <Contract> tag, which has a “Contractid” attribute that uniquely identifies the contract. The contracts consist of two main fields: the header, which includes general information and background, and the contract core, which contains all the information regarding the interactions between the parties: the definition of the actions that the parties agree to carry out, the definition of the relevant conditions and states, and the set of clauses or statements that normalize the interactions between the parties. Following, includes the syntax of the Contract structure:

```xml
<Contract Contractid=identifier>
  <Header>
    <Rolelist> <!--Definition of the VE roles-->
    </Rolelist>
    <Participants> <!--Definition of the VE participants-->
    </Participants>
    <Duration> <!--Definition of the VE duration-->
    </Duration>
  </Header>
  <Core>
    <ServiceList>
```

19
5.3. HEADER

The Header of the contract is composed of three sections: Rolelist, Participants and Duration.

5.3.1. ROLELIST

This section includes the definition of the roles that take part in the business activities regulated by the contract, and the corresponding matches with the parties that play them. Each role definition consists of the name of the role and a reference to the party that plays it. The mapping of roles with parties is crucial since the statements that regulate the interactions between the parties are specified in terms of roles. Therefore the role distribution between the parties decides and determines the obligations, permissions and prohibitions between the parties (see Clauses section). The syntax of the Rolelist definition is as follows:

```
<RoleList>
  <Role Roleid=identifier> <!--One or more-->  
    <RoleName>role_name</RoleName>
    <RolePlayer Partyid=identifier/>
  </Role>
</RoleList>
```

Where the “Roleid” attribute of the tag <Role> is an identifier that uniquely identifies the role in the contract; The <RoleName> tag contains a string with the name of the role, and the <RolePlayer> tag contains a reference attribute that points to the party that plays that role in the contract.

Keeping in mind that VE contracts need to be flexible, a party should be allowed to play more than one role. Another possibility, which has not been included in this specification but could be worth mentioning, consists in two or more parties sharing activities, and consequently, playing the same role. In this case, instead of a single party reference, the role definition should include a list of the references to the parties playing that role. Moreover, all the parties defined in the contract in the <Participants> section must be associated with at least one of the defined roles, otherwise there will be no statements in the contract core associated to that party and its participation in the VE will be null.

5.3.2. PARTICIPANTS

The Participants section includes the details of all the participants of the VE. For each participant the contract keeps information about the company name, the telephone number, the address and the details of one or more contact persons. The syntax is as follows:
<Participants>
  <Party Partyid=identifier> <!--One or more-->
    <PartyName>party_name</PartyName>
    <CompanyTelephone>telephone</CompanyTelephone>
    <Address>
      <AddressLine>address_line1</AddressLine>
      <AddressLine>address_line2</AddressLine>
      <!--As many address lines as needed -->
      <City>city_name</City>
      <Region>region_name</Region>
      <Postalcode>postal_code</Postalcode>
      <Country>country_name</Country>
    </Address>
    <Contact Type=type> <!--One or more-->
      <!--Type can just be primary or secondary--> 
      <LastName>name</LastName>
      <FirstName>name</FirstName>
      <Title>title</Title>
      <ContactTelephone 
        Type=type>telephone</ContactTelephone>
      <!--One or more-->
      <Email Type=type>email</Email>
      <!-- One or more-->
    </Contact>
  </Party>
</Participants>

Most of the tag names are self-explanatory. If no specific rules are given for a tag value, any alphanumeric string is permitted. The “Partyid” attribute of the <Party> tag is an alphanumeric string that uniquely identifies the enterprise. The telephones included in <CompanyTelephone> and <ContactTelephone> tags are numeric strings containing the area code followed by the phone number. The “Type” attribute used in <Contact>, <ContactTelephone> and <Email> tags can just be two values: “primary” or “secondary”. It is used to differentiate primary contacts, telephone numbers and emails from the secondary ones. By default the “primary” value is assumed.

5.3.3. DURATION

The Duration section defines the time interval during which the contract is valid. The syntax is as follows:

<Duration>
  <Start> <!--Optional-->  
    <Date>date</Date>
    <Time>time</Time>
  </Start>
  <End> <!--Optional-->  
    <Date>date</Date>
    <Time>time</Time>
  </End>
</Duration>

Where the dates are expressed as three numeric integer strings separated by slashes: dd/mm/yyyy, times are expressed using a 24-hours clock, as Coordinated Universal Time in the form: hh:mm:ss (hours, minutes, seconds).
The `<Start>` tag specifies the exact time when the contract begins to be valid. It is an optional field, and if omitted, it is assumed that the contract is valid immediately. In a similar way, the `<End>` tag specifies the certain time after which the contract is no longer valid. This field can also be omitted, which means that there is no limit for the contract validity. To allow contract consistency, all dates defined in the contract core statements should belong to the time period in which the contract is valid. Moreover, the actions defined in the contract must start before the `<Start>` time and none of them can continue once the `<End>` time has reached.

5.4. THE CORE OF THE CONTRACT

The core of the contract is composed of two main sections: ServiceList and Clauses.

5.4.1. SERVICE LIST

In this contract section, the list of all the services or actions that the parties agree to carry out during the contract lifetime is defined. For each action an identifier, a name and a list of parameters are specified. Furthermore, each action is associated with a namespace, which will provide the definition of the action name and the action parameters. XML namespaces are markup vocabularies to be used in different XML documents. They provide a simple way of qualifying elements and attributes by linking them with their URI reference [38]. The syntax of the Service List is as following:

```
<ServiceList>
  <Action xmlns=uri_address Actionid=identifier> <!--One or more-->
    <ActionName>action_name</ActionName>
    <ParameterList>
      <Parameter Parameterid=identifier> <!--One or more-->
        <ParameterName>parameter_name</ParameterName>
        </Parameter>
      </ParameterList>
  </Action>
</ServiceList>
```

Where uri_address must be the URI reference to the namespace where the potential values of the name and parameters of the action are defined; the “Actionid” attribute of the tag `<Action>` and the “Parameterid” attribute of the tag `<Parameter>` are identifiers that identify that action and that parameter uniquely in the contract; the tags `<ActionName>` and `<ParameterName>` contain respectively the name of the action and the parameter. Both names must be defined in the namespace referenced in the `<Action>` tag.

5.4.2. CLAUSES

This section consists of a group of clauses, which are modeled as XML representations of behavioral norms, and which will guide the interactions between the parties [2]. These norms are modeled based on deontic logic, since it has been identified as one of the most important paradigms for the formalization of contracts [3]. The representation and formalization of these norms has been inspired by the Contract Model in the Electronic Contract Framework for Contractual Agents proposed by Mathias Sallé in [39]. In this approach, contracts are modeled as a set of normative statements that are defined as follows:
ns:ϕ \rightarrow \theta_{s,b}(\alpha < \psi)

Where:
ns is a label to identify and reference the statement.
Φ is the activation condition. It is not always required and it entails that the statement is only valid when the condition is true.
θ is the deontic operator. There are only three possible values: obligation (O), permission (P) and prohibition (F).
s is the subject, or the role who assumes the statement fulfillment.
b is the beneficiary, or the role to whom the statement is fulfilled.
α is the action to perform.
ψ indicates a deadline to the statement fulfillment, it can be a date or a condition.

The statements can be read as follows: if Φ is true, then s has the \{obligation, prohibition, permission\} \{to/by\} b to do α before ψ. Obligation entails that the subject has the obligation to the beneficiary to do the action α before ψ. Permission entails that the subject is allowed to do the action α until ψ, and prohibition entails that the subject is forbidden to do the action α until ψ.

It is also worthy to include sanctions on the obligations and prohibitions of the contract core. In this way, these obligations and prohibitions are not absolute, but relative to their sanctions. Therefore, the parties of the contract can consider breaking a contract clause if they find it necessary, but always abiding to the consequences described by the sanction. This sanction component in the contracts aims at providing dynamism and flexibility, since they are inherent properties of the Virtual Enterprise nature.

Following includes an example of the normative statement representing the obligation of a party s to deliver a quantity of a certain product to a party b, before a delivery date. Also the example of the corresponding sanction is included, which expresses that in case of the delivery is not carried out properly or in time, the party s must pay a monetary compensation to the party b.

c1:Os,b(deliver(s,b,product,quantity)<deadline)
c2:not_fulfilled(c1)->Os,b(pay(s,b,price,currency)<deadline)

Regarding to the Contract Representation that is specified in this approach, it proposes to specify the deontic statements described above using XML language as follows:

```
<Clauses>
  <Statement Nd=identifier> <!--More than one-->
    <ActivatingCondition Stateid=stateid/> <!--Optional--> 
    <DeonticOperator>deontic_operator</DeonticOperator>
    <Subject Roleid=roleid/>
    <Beneficiary Roleid=roleid/>
    <Action Actionid=actionid>
    <Deadline>deadline</Deadline>
    <Fulfilled Stateid=stateid>boolean</Fulfilled>
    <NotFulfilled Stateid=stateid>boolean</NotFulfilled>
  </Statement>
</Clauses>
```
Where the “Nd” attribute of the <Statement> tag identifies each statement of the contract and allows to future references in other parts of the document using reference attributes. The <ActivatingCondition> tag is optional and includes a reference to the state of another contract statement. It indicates that the statement has not validity until the referenced state is achieved (until the value of the referenced state is true). The <DeonticOperator> tag can only contain three values: “obligation”, “permission” or “prohibition”. <Subject> and <Beneficiary> tags contain a reference attribute “Roleid” to one of the roles included in the <RoleList> section of the document, and both are used to indicate respectively the subject and the beneficiary of the statement in terms of roles. The <Action> tag has an “Actionid” attribute that references to one of the actions defined in the <ServiceList> section; the <Deadline> tag is the statement fulfillment deadline date. It is optional and it contains a date expressed as three numeric integer strings separated by slashes, dd/mm/yyyy. Furthermore two different states are defined for each statement: fulfilled and not fulfilled, which are specified under the <Fulfilled> and <NotFulfilled> tags. Both tags contain a Boolean (true or false) value that is initially false. During the contract runtime, the <Fulfilled> tag value turns into true if the statement is satisfied, and the <NotFulfilled> tag value turns into true if the deadline date is reached and the statement is not satisfied properly. References to <NotFulfilled> states are frequently used to declare sanctions.

5.4.3. TRUSTED THIRD PARTY

Lack of trust is a main problem in business relationships, for this reason is also important to include in contracts a neutral third party trusted by all the participants, which will act as a legal authority in case of contract breach or dispute. For instead, during the contract execution parties can disagree if a contract obligation has been fulfilled or not. In these situations a Trusted Third Party such a court, a notary or an auditing company, is required for arbitration and resolution. Parties present to the Trusted Third Party their vision about the situation and their evidences, and accept the final resolution [44]. The tag <ThirdParty> is used in the Contract Specification Language to specify the entity that will provide trust to the agreement.

```xml
<ThirdParty>trusted third party name</ThirdParty>
```

Where the <ThirdParty> tag contains a string with the name of the Trusted Third Party.

5.5. CONTRACT EXAMPLE

This section includes an illustrative example, which describes a VE scenario as well as the corresponding XML contract.

In this scenario two SMEs, a shoe factory and a shoe store, want to collaborate with each other in order to be better off the increase of the demand in the shoe market. Therefore they set up a VE and they commit to an electronic sales contract. Following are the statements that the SMEs commit that are expressed in natural and logic language:

The shoe factory, which plays the seller role, should produce and deliver to the shoe store, which plays the buyer role, 200 pairs of shoes before the 15th of February of 2009.

```xml
c1: O<shoe factory, shoe store>(deliver(shoe, 200)<15/02/2009)
```
Once the shoe store has properly received the requested shoes, it should pay 6000 Swedish Kronor to the shoe factory, before the 1st of March of 2009.

\[\text{c2: fulfilled(c1) -> O shoe store, shoe factory (pay(6000, Swedish Kronor) < 01/03/2009)}\]

If the requested shoes are not delivered by the shoe factory properly or on time, then the shoe factory should pay to the shoe store a compensation of 3000 Swedish Kronor, before the 20th of February of 2009.

\[\text{c3: not_fulfilled(c1) -> O shoe factory, shoe store (pay(3000, Swedish Kronor) < 20/02/2009)}\]

If the shoe store does not pay on time, then it should deliver the 200 pair of shoes back to the factory shop and furthermore, pay a compensation of 3000 Swedish Kronor.

\[\text{c4: not_fulfilled(c2) -> O shoe store, shoe factory (deliver(shoe, 200))}\]
\[\text{c5: not_fulfilled(c2) -> O shoe store, shoe factory (pay(3000, Swedish Kronor))}\]

The XML contract associated with this scenario is the following:

```xml
<?xml version="1.0"?>
<Contract xmlns="contract.xsd" Contractid="contract01">
  <Header>
    <RoleList>
      <Role Roleid="role01">
        <RoleName>"seller"</RoleName>
        <RolePlayer Partyid="party01"/>
      </Role>
      <Role Roleid="role02">
        <RoleName>"buyer"</RoleName>
        <RolePlayer Partyid="party02"/>
      </Role>
    </RoleList>
    <Participants>
      <Party Partyid="party01">
        <PartyName>"Shop"</PartyName>
        <CompanyTelephone>911234567</CompanyTelephone>
        <Adress>
          <AdressLine>"Minervavägen 22"</AdressLine>
          <City>"Kalrksrona"</City>
          <Region>"Blekingue"</Region>
          <Postalcode>37215</Postalcode>
          <Country>"Sweden"</Country>
        </Adress>
        <Contact Type="primary">
          <LastName>"Karlson"</LastName>
          <FirstName>"David"</FirstName>
          <Title>"Owner"</Title>
          <ContactTelephone Type="primary">911234567</ContactTelephone>
          <Email Type="primary">dkarl@gmail.com</Email>
        </Contact>
      </Party>
    </Participants>
  </Header>
</Contract>
```
<Party Partyid="party02">
  <PartyName>"Factory"</PartyName>
  <CompanyTelephone>917654321</CompanyTelephone>
  <Address>
    <AdressLine>"Jaerngatan11"</AdressLine>
    <City>"Ronneby"</City>
    <Region>"Blekingue"</Region>
    <Postalcode>35123</Postalcode>
    <Country>"Sweden"</Country>
  </Address>
  <Contact Type="primary">
    <LastName>"Ohlson"</LastName>
    <FirstName>"James"</FirstName>
    <Title>"Director"</Title>
    <ContactTelephone Type="primary">917654321</ContactTelephone>
    <EmailType="primary">"james@gmail.com"</Email>
  </Contact>
</Party>
</Participants>

<Duration>
  <Start>
    <Date>01/09/2008</Date>
    <Time>08:00:00</Time>
  </Start>
  <End>
    <Date>01/04/2009</Date>
    <Time>17:00:00</Time>
  </End>
</Duration>

<ThirdParty>"The auditing company"</ThirdParty>
</Header>

<ServiceList>
  <Action xmlns="http://www.w3.org/1999/xlink"
    Actionid="action01">
    <ActionName>deliver</ActionName>
    <ParameterList>
      <Parameter Parameterid="parameter01">
        <ParameterName>shoes</ParameterName>
      </Parameter>
      <Parameter Parameterid="parameter02">
        <ParameterName>200</ParameterName>
      </Parameter>
    </ParameterList>
  </Action>
  <Action xmlns="http://www.w3.org/1999/xlink"
    Actionid="action02">
    <ActionName>pay</ActionName>
    <ParameterList>
      <Parameter Parameterid="parameter03">
        <ParameterName>6000</ParameterName>
      </Parameter>
      <Parameter Parameterid="parameter04">
        <ParameterName>26</ParameterName>
      </Parameter>
    </ParameterList>
  </Action>
</ServiceList>
<ParameterName>SwedishKronor</ParameterName>
</Parameter>
</ParameterList>
</Action>
<Action xmlns="http://www.w3.org/1999/xlink"
Actionid="action03">
<ActionName>pay</ActionName>
<ParameterList>
  <Parameter Parameterid="parameter05">
    <ParameterName>3000</ParameterName>
  </Parameter>
  <Parameter Parameterid="parameter06">
    <ParameterName>Swedish Kronor</ParameterName>
  </Parameter>
</ParameterList>
</Action>
<Action xmlns="http://www.w3.org/1999/xlink"
Actionid="action04">
<ActionName>deliver</ActionName>
<ParameterList>
  <Parameter Parameterid="parameter07">
    <ParameterName>shoes</ParameterName>
  </Parameter>
  <Parameter Parameterid="parameter08">
    <ParameterName>200</ParameterName>
  </Parameter>
</ParameterList>
</Action>
</ServiceList>
<Clauses>
<Statement Nd="c1">
  <DeonticOperator>"obligation"</DeonticOperator>
  <Subject Roleid="role01"/>
  <Beneficiary Roleid="role02"/>
  <Action Actionid="action01"/>
  <Deadline>15/02/2009</Deadline>
  <Fulfilled Stateid="state01">false</Fulfilled>
  <NotFulfilled Stateid="state02">false</NotFulfilled>
</Statement>
<Statement Nd="c2">
<ActivatingCondition Stateid="state01"/>
  <DeonticOperator>"obligation"</DeonticOperator>
  <Subject Roleid="role02"/>
  <Beneficiary Roleid="role01"/>
  <Action Actionid="action02"/>
  <Deadline>01/03/2009</Deadline>
  <Fulfilled Stateid="state03">false</Fulfilled>
  <NotFulfilled Stateid="state04">false</NotFulfilled>
</Statement>
<Statement Nd="c3">
<ActivatingCondition Stateid="state02"/>
  <DeonticOperator>"obligation"</DeonticOperator>
  <Subject Roleid="role01"/>
  <Beneficiary Roleid="role02"/>
</Statement>
<Action Actionid="action03"/>
<Deadline>20/02/2009</Deadline>
<Fulfilled Stateid="state05">"false"</Fulfilled>
<NotFulfilled Stateid="state06">"false"</NotFulfilled>
</Statement>
<Statement Nd="c4"/>
<ActivatingCondition Stateid="state04"/>
<DeonticOperator>"obligation"</DeonticOperator>
<Subject Roleid="role02"/>
<Beneficiary Roleid="role01"/>
<Action Actionid="action04"/>
<Fulfilled Stateid="state07">"false"</Fulfilled>
<NotFulfilled Stateid="state08">"false"</NotFulfilled>
</Statement>
<Statement Nd="c5"/>
<ActivatingCondition Stateid="state04"/>
<DeonticOperator>"obligation"</DeonticOperator>
<Subject Roleid="role02"/>
<Beneficiary Roleid="role01"/>
<Action Actionid="action05"/>
<Fulfilled Stateid="state09">"false"</Fulfilled>
<NotFulfilled Stateid="state10">"false"</NotFulfilled>
</Statement>
</Clauses>
</Core>
</Contract>
6. LAYERED CONTRACT ONTOLOGY

Finally, this approach presents a layered ontology specially tailored to provide semantic meaning to the proposed Contract Representation Language.

Contracts in VE represent agreements between very different and diverse parties, which do not need to have anything in common a priori. Therefore a common vocabulary, which defines and formalizes all the terms in a precise and unambiguous way is needed. Ontologies provide this formal vocabulary.

The presented Layered Contract Ontology, has a flexible layered structure and is encoded in OWL (Web Ontology Language). It was constructed using Protégé 3.3.1 and the consistency was checked using Racer 1.9.0.

The OWL Web Ontology Language is a language for defining and instantiating ontologies. It was produced by the W3C Web Ontology Working Group and it is one of the most relevant technologies supporting the Semantic Web, thus it has attracted both academic and commercial interest. It is based on description logics and has attractive computational properties [43].

6.1. LAYERED STRUCTURE

Each VE commitment has a different nature and deals with a different domain. It can be a sale and purchase agreement between a factory and a retailer (for example, the shoes sale and purchase scenario described in the Contract Representation Language chapter); an agreement between a hotel chain, an airline and a travel agency that aims at exploiting a touristic destination; or an agreement between a group of small stores and an ICT company that aims at offering online product sales. In order to maintain the VE heterogeneity, VE contracts require adapting their knowledge to different domains. The layered structure of the proposed ontology intends to satisfy this important requirement.

The proposed Layered Contract Ontology has a layered structure that consists of two layers: Top Level Contract Ontology and Domain Level Contract Ontology. The Top Level Contract Ontology provides common, general vocabulary related to contracts and defines all the required elements that a contract needs to be valid. It is domain independent, thus is fixed and common for all type of contracts. Whereas the Domain Level Contract Ontology layer defines domain specific terms and it changes and evolves to adapt to a particular domain.

In fact, the concepts defined by the Domain Level Ontology, are a domain specialization of the general terms described by the Top Level Contractual Ontology. The Domain Level Ontology inherits the terms from the Top Level Contractual Ontology, and extends and adapts them to the domain [27].

This layered structure allows an extensible, dynamic, flexible and coherent contract ontology that evolves to adapt the knowledge to the required situation, and that fits perfectly with the dynamism and flexibility requirements of the VEs.

6.2. TOP LEVEL CONTRACT ONTOLOGY

It consists of fifteen concepts or classes and its relationships: VE_Contract, Enterprise, Participant_Enterprise, Role, Deontic_Assign, Obligation, Permission,
6.2.1. VE_Contract

**VE_contracts** constitutes an agreement between two or more enterprises, which commit to certain deontic assignments (obligations, prohibitions or permissions) in order to carry out business collaboration. It has six attributes: id, which is a string to identify the contract; start, which is the exact time when the contract begins to be valid; end, which is the exact time when the contract is not valid anymore; has_Participant, which is a relationship with the Participant_Enterprise class and constitutes the enterprises participating in the enterprise and their corresponding roles; has_Deontic_Assignment, which is a relationship with the Deontic_Assignment class and constitutes the deontic assignments that compose the contract; and has_Third_Party, which is a relationship with the Third_Party class and constitutes the authority providing trust and solving disputes in the contract.

6.2.2. Enterprise

**Enterprise** is an organization created for business ventures. It has ten properties: name, telephone, address, city, region, country (which are all self explanatory); and has_Contact, which is a relationship with the Contact class and defines the details of the enterprise contact persons.

6.2.3. Participant_Enterprise

**Participant_Enterprise** constitutes an enterprise participating in a VE and therefore committing a contract. It is a sub class of the concept Enterprise, and it has two properties: has_Role, which is a relationship with the class Role and determines
the role that the enterprise plays in the VE; and contract, which is a relationship with the class VE_Contract, and determines the contract that the enterprise commits to.

6.2.4. Role

Role is a function expected or required of a person or group.

6.2.5. Deontic_Assignment

Deontic_Assignment is defined as an obligation, prohibition or permission assigned a role to perform an action to another role. It has six properties: has_Activating_Condition, which is a relationship with the Activating_Condition class and indicates a condition that activates the deontic assignment; subject, which is a relationship with the Role class and specifies the role that assumes the statement fulfillment; beneficiary, which is a relationship with the Role class and specifies the role to whom the statement is fulfilled; has_Action, which is a relationship with the Action class and specifies the action to perform; deadline, which is the deadline date for the assignment fulfillment; and has_State, which is a relationship with the Clause_State class and defines the current state of the deontic assignment.

6.2.6. Obligation

Obligation is a subclass of Deontic_Assignment, and constitutes an obligation assigned to a role to perform an action to another role.

6.2.7. Permission

Permission is a subclass of Deontic_Assignment, and constitutes a permission assigned to a role to perform an action to another role.
6.2.8. Prohibition

**Prohibition** is a subclass of Deontic_Assignment, and constitutes a prohibition assigned to a role to perform an action to another role.

6.2.9. Clause_State

**Clause_State** constitutes the way a Deontic_Assignment is with respect of its fulfillment.

6.2.10. Committed

**Committed** is a subclass of Clause_State, and constitutes a state in which the action associated with a clause is not completed yet and the deadline has not been reached.

6.2.11. Fulfilled

**Fulfilled** is a subclass of Clause_State, and constitutes a state in which the action associated with a clause is carried out properly.

6.2.12. Not_Fulfilled

**Not_Fulfilled** is a subclass of Clause_State, and constitutes a state in which the action associated with a clause is not carried out properly or on time.

6.2.13. Activating_Condition

**Activating_Condition** constitutes a condition that a specific deontic assignment has a certain state. Therefore it has two properties, has_Deontic_Assignment and
has_Clause_State, which are relationships with the classes Deontic_Assignment and Clause_State respectively.

6.2.14. Action

**Action** constitutes something done or to be done.

6.2.15. Contact

**Contact** constitutes a person responsible for the enterprise. It has six properties: last_name, first_name, title, telephone, email and represents_enterprise, which is a relationship with the class Enterprise and specifies the enterprise that the contact is responsible for.

6.2.16. Third_Party

**Third_Party** is a legal authority that provides trust and solves contract breaches or disputes.

Following includes a diagram modeling the classes and relationships of the Top Level Contract Ontology:
6.3. DOMAIN LEVEL CONTRACT ONTOLOGY

As mentioned before, the Domain Level Contract Ontology defines terms that specialize in a domain the generic terms from the Top Level Contract Ontology. It is variable and extensible, since it contains different concepts as convenient and thus modeling knowledge for specific contract types.

As an illustration, following depicts a Domain Level Contract Ontology for the sale-purchase domain. This ontology specifies the term Role in the upper level in two subclasses: Seller and Buyer, and specifies the term Action also in two classes: Deliver and Pay. Moreover it includes a new class: Product.

6.3.1. Seller

**Seller** is a subclass of Role. It is the exchange products for money role.

6.3.2. Buyer

**Buyer** is a subclass of Role. It is the acquire products by financial transaction role.

6.3.3. Deliver

**Deliver** is a subclass of Action. It is to bring an amount of a product to a destination. It has two attributes: amount and product.
6.3.4. Pay

Pay is a subclass of Action. It is to pay a certain amount of money. It has two attributes: amount and currency.

6.3.5. Product

Product is an artifact that has been created by someone or some process.

Following includes a diagram modeling the whole Layered Contract Ontology, including both the Top Level Contract Ontology and the Domain Level Contract Ontology for the sale-purchase domain (in red).

The ontology depicted above would be the ontology used for knowledge representation in the shoe sales scenario presented in the Contract Representation Language specification. An appendix with the OWL coding of the Layered Contract Ontology for this domain is included at the end of the thesis.
7. **VALIDATION**

Finally, the Contract Representation Language and the Contract Ontology are applied in a real electronic contract to check the validity of the approach and discover possible lacks and deficiencies.

The contract used is a real business contract presented in the paper “Requirements on a B2B E-contract Language” by Angelov and Grefen [1]. It has been chosen for its brevity while containing complete and diverse information, which can be useful for a general study of the applicability of the proposed solution in this thesis.

Despite this is a real business contract, the names and data of the company does not appear for privacy reasons.

**Contract for HyperText Markup Language Formatting**

<table>
<thead>
<tr>
<th>File Name</th>
<th>File Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
To study how to represent this contract using the proposed Contract Representation Language and the Contract Layered Ontology, it is necessary to identify: the specific domain terms in the contract, which are the terms that the Domain Level of the Contract Layered Ontology should represent; and the obligations, permissions and prohibitions that the contract establishes between the parties.

In lines from 1 to 4, the two contracting parties are defined and matched with roles. Company A is matched with The Contractor role and Company B with The Author role. This information suits perfectly in the RoleList section of the Contract Representation Language, where the roles are declared and matched with the respective parties.

Furthermore, in lines from 36 to 42 the contact data and details of the contracting parties are stated. These details will be stated in the Participants section of the Contract Representation Language.

```
<RoleList>
  <Role Roleid="role01">
    <RoleName>"contractor"</RoleName>
    <RolePlayer Partyid="party01"/>
  </Role>
  <Role Roleid="role02">
    <RoleName>"author"</RoleName>
    <RolePlayer Partyid="party02"/>
  </Role>
</RoleList>

<Participants>
  <Party Partyid="party01">
    <PartyName>"Company A"</PartyName>
    <CompanyTelephone>telephone</CompanyTelephone>
    <Adress>
      <AdressLine>adress</AdressLine>
      <City>city</City>
      <Region>region</Region>
      <Postalcode>postalcode</Postalcode>
      <Country>country</Country>
    </Adress>
    <Contact Type="primary">
      <LastName>last_name</LastName>
      <FirstName>first_name</FirstName>
      <ContactTelephone Type="primary">telephone</ContactTelephone>
      <Email Type="primary">email</Email>
    </Contact>
  </Party>
  <Party Partyid="party02">
    <PartyName>"Company B"</PartyName>
```
In lines from 6 to 12, the Author commits to the obligation of providing the contractor with files. The Author also commits to a providing method: upload the files through internet to the Contractor’s ftp directory and the characteristics that these files should have: complete and with instructions. Furthermore a prohibition is also stated, it is forbidden for the Author to provide files with errors or viruses. These obligations and prohibition can be represented in the following deontic based statements:

\[ c1: O_{author, contractor}(provide(files, upload_ftp, complete))\]

\[ c2: O_{author, contractor}(provide(instructions))\]

\[ c3: F_{author, contractor}(provide(files, errors, virus))\]

Obviously, the action provides, the providing method: upload_ftp, the concept of file and the file characteristics: complete, with instructions, errors and viruses, should be included in the Domain Level Contract Ontology, which will provide semantic meaning to these concepts and allow the contract participants to have a common understanding of the terminology.

Following are the clauses of the contract that refer to the statements above. The actions are declared in the ServiceList section and then the corresponding deontic based statements are declared in the Clauses section.

```xml
<ServiceList>
  <Action xmlns="http://www.linktotheontology.owl" Actionid="action01">
    <ActionName>provide</ActionName>
    <ParameterList>
      <Parameter Parameterid="parameter01">
        <ParameterName>files</ParameterName>
      </Parameter>
      <Parameter Parameterid="parameter02">
        <ParameterName>upload_ftp</ParameterName>
      </Parameter>
      <Parameter Parameterid="parameter03">
        <ParameterName>complete</ParameterName>
      </Parameter>
    </ParameterList>
  </Action>
</ServiceList>
```
<ParameterName>
</ParameterName>
</Parameter>
</ParameterList>
</Action>
<Action xmlns="http://www.linktotheontology.owl"
Actionid="action02">
<ActionName>provide</ActionName>
<ParameterList>
  <Parameter Parameterid="parameter04">
    <ParameterName>instructions</ParameterName>
  </Parameter>
</ParameterList>
</Action>
</ServiceList>
<br>...
<Clauses>
  <Statement Nd="c1">
    <DeonticOperator>"obligation"</DeonticOperator>
    <Subject Roleid="role02"/>
    <Beneficiary Roleid="role01"/>
    <Action Actionid="action01"/>
    <Fulfilled Stateid="state01">"false"</Fulfilled>
  </Statement>
  <Statement Nd="c2">
    <DeonticOperator>"obligation"</DeonticOperator>
    <Subject Roleid="role02"/>
    <Beneficiary Roleid="role01"/>
    <Action Actionid="action02"/>
    <Fulfilled Stateid="state02">"false"</Fulfilled>
  </Statement>
  <Statement Nd="c3">
    <DeonticOperator>"prohibition"</DeonticOperator>
    <Subject Roleid="role02"/>
    <Beneficiary Roleid="role01"/>
    <Action Actionid="action03"/>
    <Fulfilled Stateid="state05">"false"</Fulfilled>
  </Statement>
</Clauses>
In line 19 the contract states that the duration of the agreement is one year. In the Contract Representation Language this can be represented using the Duration section, in which the end date should be one year after the start date.

In lines from 14 to 16, the Contractor commits to the obligation of formatting the files into Hyper Text Markup Languages and therefore to obtain the final product. The formatting should be done following the instructions provided by the Author and should be compliant with html 2.0 protocols. The contract also establishes that the Contractor is forbidden to make changes in the files. This obligation and prohibition can be represented with the following deontic based statements:

C4: Ocontractor, author (format(files, html, html2.0_compliant, instructions))

C5: Fcontractor, author (change(files))

The actions format and change; and the formatting characteristics: Hyper Text Markup Language, html 2.0 protocol compliant and instructions, should be included in the Domain Level Contract Ontology, which will provide semantic meaning to these concepts and allow the contract participants to have a common understanding of the terminology.

Following are the clauses of the contract that refer to the statement above. The format action is declared in the ServiceList section and then the corresponding deontic based statements are declared in the Clauses section.
In lines from 18 to 23, the contract establishes the payment conditions. Specifically, The Author commits to give The Contractor a discount on the product sales for one year. Furthermore, the contract establishes the right or permission for The Author to distribute the product through the Internet for one year. These obligations and permissions are only valid once the contractor has finished the formatting task and can be represented in the following deontic-based statements:

C6: fulfilled(c3)->O_{author,contractor} (discount(files, amount, currency))

C7: fulfilled(c3)->P_{author} (distribute(files, internet))

The actions discount and distribute and the method of distribution: internet, as well as the concepts of amount and currency should be included in the Domain Level Contract Ontology, which will provide semantic meaning to these concepts and allow the contract participants to have a common understanding of the terminology.

Following are the clauses of the contract that refer to the statements above. The provided actions are declared in the ServiceList section and then the corresponding deontic-based statements are declared in the Clauses section.
Furthermore, a Domain Level Contract Ontology that extends the Top Level Contract Ontology and defines the specific domain terms is needed to provide semantic meaning to the contractual concepts and understanding between the participants.
This Domain Level Contract Ontology extends the Role class with two subclasses: Contractor: someone (a person or firm) who contracts to do something; and Author: someone who originates or causes or initiates something.

Furthermore, the Action class is also extended with five subclasses: Provide, Format, Change, Discount and Distribute. Provide is to supply a product that is desired or needed. It has two attributes: product and method (upload through ftp protocol). Format is to organize something according to preset specifications. It has four attributes: product, specifications (html), compatibility (html 2.0 protocols) and instructions, Change is the action of modifying something. It has one attribute: product. Discount is the act of reducing the selling price. It has three attributes: amount, currency and product. Distribute is to disperse something widely. It has two attributes: product and method (through internet). The concepts of Product, Instructions and File also have a main importance in this ontology. Product is an artifact that has been created by someone or some process. File is a subclass of Product and it is a set of related records (either written or electronic) kept together. It has three attributes: complete, errors and virus. Instruction is a message describing how something is to be done.
This “translation” of a real business contract comes to the conclusion that the Contract Representation Language is suitable to represent and define the details of contractual parties and their interactions in a contract. However, there are some deficiencies when dealing with legal issues, which cannot be represented in terms of prohibitions, obligations and permissions but are also important to include in contracts. For example, the proposed Contract Representation Language does not represent the copyright and ownership provisions defined in lines from 25 to 27 in the real business sample contract, and it does not represent the requirement of a distribution license defined in line 29 either. Furthermore, the Contract Representation Language establishes general agreements between contracts and has problems in contracts that deal with very specific elements. For example, in the presented real business sample contract the Author commits to provide specific files, whose name, size and description are stated in lines from 33 to 35. However, the Contract Representation Language only represents that the author commits to provide files. Therefore this Contract Representation Language can be extended in future work to provide support for legal issues, specific agreements and also digital signing; since it does not provide any support for the parties’ signature.
8. **DISCUSSION**

In this section the approach is compared with existing ones in order to identify the contributions and the original work that the thesis provides.

Many approaches represent contracts as XML documents, for example TpaML, ebXML or CrossFlow contracts. Nevertheless, neither of them establishes the rights, prohibitions and permissions between the participant parties.

However, the Business Contract Language (BCL) incorporates these notions of obligations, prohibitions and permissions. But BCL uses them to define policies that limit the behavior of the parties whereas the approach suggested in this thesis proposes contract clauses based on deontic logic statements which specify the interactions between the parties. Furthermore BCL is a domain specific language, which only represent generic contract domain terms, whereas the approach suggested in this thesis aims at domain independency and adaptability.

Similar to the proposed Contract Representation Language, SweetDeal uses ontologies to provide semantic meaning to contracts. But it lacks contract structure and deontic notions.

It is also important to mention that, although VEs typically are mentioned in the context of SMEs (SMEs are the companies who most need to look for business collaboration and business opportunities to survive in the competitive market), large enterprises can also take benefit from this approach since there is not any restriction on the size of the company.

As a conclusion, this approach presents a flexible interoperable XML-based representation for contracts with domain independent semantic meaning that allows understanding between the participants and focuses on the establishment of obligations, prohibitions and permissions. Furthermore, for the best of my knowledge, this approach proposes a unique representation for contracts that combines an XML structure with role specification and task allocation, deontic logic and domain-independent ontology.
9. RESULTS AND CONCLUSIONS

Finally the study obtains significant knowledge about electronic contracting for VE that answers the research questions stated in the problem definition. The main conclusions that could be relevant for the development of a VE contractual framework that enables automation in the VE formation stage are summarized below:

Electronic contracting is an area supported by a considerable amount of research. This study identifies three main paradigms related to electronic contracts: contract representation; that is, specification of the contract format, structure and content; contract normative statements; that is, specification of the contract clauses that regulate participants’ behavior; and semantic contract issues; that is, specification of how to provide semantic meaning to the concepts inside the contract.

XML based approaches, role based architectures and contract templates are the most remarkable approaches related to electronic contract representation. Role based architecture enables simple and easy-to-modify task distribution, since the contractual clauses are specified in terms of roles and not in terms of participants. Furthermore, Role based architecture together with contract templates management provide a fixed structure that enables automatic contract negotiation and establishment. In addition, XML provides interoperability and its link and reference properties provide dynamism to the structure and enable links with ontologies.

Concerning the representation of normative clauses, deontic logic is the indisputable main trend. It allows representing business relationships between enterprises with the concepts of obligation, permission and prohibition. Moreover, the concept of sanction is frequently included, which enables participant to make their own independent decisions.

Relating to semantic meaning issues, ontologies are the best solution. They provide semantic meaning to the contract knowledge as well as interoperability. Furthermore a layered dynamic structure is a suitable solution for VE domain independence requirements.

Besides the issues already commented in the validation, further issues for future work related to this thesis may be: specifying of contract templates to support contract establishment and negotiation; proposing a contract negotiation protocol; and extending this solution for supporting automatic contract execution and automatic contract fulfillment monitoring.
REFERENCES


APPENDIX

This appendix contains the OWL coding of the whole Layered Contractual Ontology for the sales and purchase domain. This code was created using Protégé 3.3.1 and the consistency was checked using Racer 1.9.0.

```xml
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns="http://www.owl-ontologies.com/Ontology1218989694.owl#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:p1="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://www.owl-ontologies.com/Ontology1218989694.owl#">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="Fulfilled">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Clause_State"/>
    </rdfs:subClassOf>
    <owl:disjointWith>
      <owl:Class rdf:ID="Not_Fulfilled"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Committed"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:ID="Pay">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Action"/>
    </rdfs:subClassOf>
    <owl:disjointWith>
      <owl:Class rdf:ID="Deliver"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:about="#Clause_State">
    <owl:disjointWith>
      <owl:Class rdf:ID="Contact"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="VE_Contract"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Deontic_Assignment"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Activating_Condition"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Enterprise"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#Action"/>
    </owl:disjointWith>
  </owl:Class>
</rdf:RDF>
```
<owl:disjointWith>
  <owl:Class rdf:about="#Role"/>
</owl:disjointWith>
<owl:disjointWith>
  <owl:Class rdf:about="#Action"/>
</owl:disjointWith>
<owl:disjointWith>
  <owl:Class rdf:resource="#Product"/>
</owl:disjointWith>
<owl:disjointWith>
  <owl:Class rdf:about="#Enterprise"/>
</owl:disjointWith>
</owl:Class>
<owl:Class rdf:ID="Seller">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Role"/>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:ID="Buyer"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="#Deontic_Assignment">
  <owl:disjointWith>
    <owl:Class rdf:about="#Action"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Activating_Condition"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Enterprise"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#Product"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Role"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#VE_Contract"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Contact"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#Clause_State"/>
</owl:Class>
<owl:Class rdf:about="#Activating_Condition">
  <owl:disjointWith rdf:resource="#Clause_State"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Action"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#Deontic_Assignment"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Enterprise"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#Product"/>
  <owl:disjointWith rdf:resource="#VE_Contract"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Role"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Contact"/>
  </owl:disjointWith>
</owl:Class>
<owl:disjointWith/>
<owl:Class rdf:about="#Committed">
  <dfs:subClassOf rdf:resource="#Clause_State"/>
  <owl:disjointWith rdf:resource="#Not_Fulfilled"/>
  <owl:disjointWith rdf:resource="#Fulfilled"/>
</owl:Class>
<owl:Class rdf:ID="Obligation">
  <dfs:subClassOf rdf:resource="#Deontic_Assignment"/>
  <owl:disjointWith>
    <owl:Class rdf:ID="Prohibition"/>
    <owl:disjointWith>
      <owl:Class rdf:ID="Permission"/>
      <owl:disjointWith rdf:resource="#Obligation"/>
    </owl:disjointWith>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="#Prohibition">
  <owl:disjointWith>
    <owl:Class rdf:about="#Permission"/>
    <owl:disjointWith rdf:resource="#Obligation"/>
    <rdfs:subClassOf rdf:resource="#Deontic_Assignment"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="#Deliver">
  <owl:disjointWith rdf:resource="#Pay"/>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Action"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#Role"/>
    </owl:disjointWith>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Buyer">
  <dfs:subClassOf>
    <owl:Class rdf:about="#Role"/>
    <owl:disjointWith rdf:resource="#Seller"/>
  </dfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Contact">
  <owl:disjointWith rdf:resource="#Activating_Condition"/>
  <owl:disjointWith rdf:resource="#Deontic_Assignment"/>
  <owl:disjointWith rdf:resource="#Clause_State"/>
  <owl:disjointWith rdf:resource="#VE_Contract"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Enterprise"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#Product"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Action"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Role"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="#Role">
  <owl:disjointWith rdf:resource="#Activating_Condition"/>
  <owl:disjointWith rdf:resource="#Deontic_Assignment"/>
  <owl:disjointWith rdf:resource="#Clause_State"/>
<rdfs:range rdf:resource="#Role"/>  
<rdfs:domain rdf:resource="#Participant_Enterprise"/>  
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>  
</owl:FunctionalProperty>

<owl:FunctionalProperty rdf: ID="has_Clause_State" >
 <rdfs:range rdf:resource="#Clause_State"/>
 <rdfs:domain rdf:resource="#Activating_Condition"/>  
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>  
</owl:FunctionalProperty>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 3.3.1, Build 430)  http://protege.stanford.edu -->