Formal Specification of Web Service Contracts for Automated Negotiations and Compliance Checking

Abstract

Service-oriented computing as a concept for providing interoperability and flexibility within heterogeneous environments has gained much attention within last years. Dynamically integrating external Web services into enterprise applications requires automatic contracting between service requestors and providers and automatic contract monitoring. This paper suggests semi-automatic approach since in the current legal environment full automation is not feasible. We elaborate on the content of Web service contracts from a legal perspective and derive a set of legal requirements. Based on these requirements we propose an ontology-based representation of contract clauses as well as monitoring information. We can thus automatically evaluate whether a service execution meets the requirements expressed in a contract.

1. Introduction

Information systems of the future will be combinations of loosely-coupled services. In service-oriented architectures (SOA), application systems are assembled as required by pulling together various services. In this context, a service is a software component which can be used by means of standard internet technologies. The implementation of services is encapsulated and numerous service providers may provide the same functionality. Hence, a customer may choose from a variety of implementations depending on his preferences. So as to make sure that a service meets the requirements, customers and providers have to agree on terms of a contract.

The area of electronic contracting received considerable attention in recent years [1]. The work can be structured according to the contracting lifecycle [2]: First, in the **Information Phase** information about the product, other parties, etc. is gathered. Then, terms and conditions of an agreement are determined in the **Agreement Phase**. Finally, in the **Settlement Phase** contracts are executed and the fulfillment of the contractual clauses is monitored.

In order to support automation of the entire contracting life cycle formal machine-understandable representation of contract information is required. Since this is hardly achievable in an inter-organizational setting, we focus on semi-automatic contracting where an umbrella contract is manually closed with different service providers and only some of the terms are negotiated for each service invocation dynamically. In order to support the settlement phase monitoring information as well as knowledge how to interpret the contract is required. Usually, the latter is available only as tacit knowledge of legally educated persons and thus has to be externalised into a machine readable and executable form.

Our paper is structured according to the typical ontology engineering process [3]. In section 2 we introduce a scenario which provides a use case for the subsequent sections. Then in section 3 requirements for representing Web service contracts are discussed from a legal perspective. After presenting general design considerations of our ontology framework in section 4, we demonstrate how a Web service contract (section 5) and monitoring information (section 6) are represented. In addition, we show in section 7 how to evaluate whether services are delivered as specified in the contract. Finally, we discuss related work in section 8 and conclude in section 9.

2. Scenario

In order to reduce credit risk and to select profitable customers, many companies have always relied on credit information. The latest legal developments around risk management such as Sarbanes Oxley or related legal regulation have forced the companies to have a closer look at the management of financial risk. Financial information relating to the creditworthiness of companies, the profitability of their business or the quality of their senior management helps companies to assess the risk of doing business with each other and respond to increased or decreased risk. Companies as Dun & Bradstreet or Creditreform collect and sell credit information. Based on credit information,

companies will decide whether to start business with another company or determine and adapt lines of credit.

In the past, such lines of credit have often been adapted too late as buying of credit information was done manually and not always on a continuous basis. Thus, integrating critical credit information into existing enterprise applications allows for risk decisions based on externally provided and permanently updated data. Standard internet technologies such as Web services can be used to retrieve this credit information.

We distinguish between three degrees of automation of the contracting lifecycle, namely a manual, semi-automatic and automatic approach. In the manual case human beings are involved in all steps of the contract lifecycle. Full automation is the situation in which the complete contract lifecycle can be supported by software agents - a scenario which has been investigated only for very simple contracts. For reasons illustrated in the following, a semi-automatic approach is more appropriate.

3. Legal Perspective

In this section, we discuss the legal concept of semi-automatic contracting and monitoring. Finally, we explain the possible contents of a contract negotiated and concluded by software agents based on these considerations.

3.1. Semi-automatic contracting of credit information services

Semi-automatic contracting can be seen as an interim solution, where contracts made based on a manually negotiated **umbrella agreement** are being negotiated by software agents. The 'umbrella agreement' is directly negotiated by human beings. The contract negotiated by the software agents is referred to as '**individual contract'**.

The umbrella agreement is presently necessary to define the legal conditions under which software agents can enter into binding agreements as not all jurisdictions acknowledge negotiating and contracting by software agents. In our scenario we assume that the umbrella agreement provides for German law to be applicable to the umbrella agreement, the individual contract and any action of the parties via its software agents. Under German law, declarations of software agents are deemed to be human declarations under a legal concept called "declaration of a blank form" [4].

In our scenario, the service requestors agree on an umbrella agreement with several credit information companies. The umbrella agreement will therefore define the framework for several software agents to negotiate the individual contracts. The umbrella agreement specifies the beginning of the contractual relations between all parties, how long the umbrella agreement is valid and how and when it can be terminated. The credit information services to be negotiated as well as the timeframe for negotiations (preferably 24/7) are agreed on in the umbrella agreement. The umbrella agreement will also define those credit information services which the software agents are entitled to negotiate and to contract for. For a better understanding, such a clause is included in this paper:

"Scope of this umbrella agreement is the provision of the following Credit Information Service, hereinafter referred to as Credit Information and Credit Information Services on the basis of individual contracts negotiated by either party's software agents. For the purposes of this umbrella agreement and the individual contracts, Credit Information means
(a) Information about ownership, history and principals of a company, and the operations and location of the company, hereinafter, referred to as Business Background Information.
(b) In-depth information about the business of a company, the situation of the company in the market and the senior management of the company, referred to as Quality of Company Information.
(c) Scoring information about the likelihood that a company becomes insolvent within the next 12 months. The scores are based on a scale of 1 to 100, any 10 point increase relates to a doubling of the risk, referred to as Credit Score Calculation.
(d) Calculation of the credit limit for a company based on the risk level agreed on between the parties, either a conservative limit or an aggressive limit, referred to as Credit Limit Calculation."

The umbrella agreement also provides for auxiliary duties of the parties such as the obligation to treat customer information confidentially or maintenance and service duties. These clauses lay the ground for the later provision of credit information services and form the continuous contractual relations between the parties. As their content spans more than one provision of credit information, they are not negotiated every time an individual credit information service is needed.

The individual credit information service, the license type, how up-to-date credit information is, which guarantees for the accuracy are given as well as the price and the payments terms are negotiated whenever there is a need for a service.

This wide array of terms that are negotiated distinguishes the semi-automatic contracting from a simple call-off order. A call-off order is placed against an umbrella agreement which defines all the conditions except delivery time and amount and, in some cases,

prices. The option to negotiate not only these basic terms allows the parties to flexibly adapt the services as well as the price to changing conditions.

After closing a contract in the settlement phase the participants monitor whether the contractual duties are fulfilled. However, full automation of the monitoring step is impossible since assessing the quality of a credit information service can only be done by taking external and not quantifiable factors into account. Nevertheless some monitoring tasks can be done by the system automatically. For instance, it can be assured that an individually contracted service is provided at all and in the negotiated timeframe. For this purpose, all clauses that are relevant to evaluate whether the contract is met also have to be represented in our formal representation language.

3.2. Content of the individual contract on provision of Credit Information Services

The individual contract is negotiated by the software agents. In such an individual contract the software agents will negotiate which credit information service should be delivered and paid.

Credit Information Service (§ 1). In our scenario we distinguish between so called 'Business Background Information', 'Quality of Company Information', 'Credit Score Calculation', 'Warning Information' and 'Credit Limit Calculation'. In a physical contract between human beings such content would be covered in a clause '§ 1 Scope of Agreement'.

Update Periods (§ 2). Usually, it is price relevant how old the credit information is. The software agents therefore negotiate the update periods of credit information. For example, Business Background Information is updated either every month or once a quarter. An example of a contract clause (§ 2) is given in the section contract representation.

Use of Information and Licenses (§ 3). The individual contract will specify how the customer may use the information. In our scenario the service provider grants a license. A contract clause specifying such use may grant a transferable license to use the information or a non-transferable license and define further to what extent the customer may use the credit information within its company or towards third parties. The extent of the usage can be negotiated by the software agents.

Warranties (§ 4). The software agents will negotiate which warranties the service provider gives for the individual credit information service. Negotiating warranties is a legally very complex task even when human beings are involved. Standard terms and conditions, used by companies to facilitate negotiations and transactions in routine business, to a large extent consist of clauses to limit/extend liability and to describe warranties. This shows that there is a need to negotiate warranties while it is not efficient to negotiate them for small or routine transactions. However, warranties are price relevant. We let the software agents negotiate about the warranty level but not about the legal obligations resulting from a breach of warranty. The legal complexity, including the restrictions by law to contract out certain statutory warranties and liabilities, does not allow negotiating the legal obligations by software agents at present. When negotiating a warranty we work with the following scheme: (1) The service provider does not give any warranty as to the accuracy of the information. (2) The service provider does not warrant the accuracy of the information, but warrants that it has put the information together with utmost care and state-of-the-art-methods (3) The service provider guarantees that the information is 100% correct.

Delivery Time (§ 5). The delivery of credit information service can be negotiated by the software agents in a way that the service has to be provided immediately after the individual contract is concluded or at a later, negotiated time. The legal consequences of non- or late delivery however are set forth in the umbrella agreement for the same reasons as specified above under warranties.

Prices and Payment Terms (§ 6). Finally, the software agents will negotiate the prices and payment terms. While the parties defined the details of invoicing in the umbrella agreement, the software agents may negotiate the price for the individual credit information service as well as the due date of the payment.

4. General Ontology Framework

In order to be able to pass down the contract negotiation and execution to the system level knowledge about the contracts and their interpretation has to be expressed in a machine interpretable way. Thus, a well-defined, formal representation is required that allows heterogeneous systems to understand, close, and enforce the contracts.

In recent years, *ontologies* emerged as state of the art for knowledge sharing in distributed, heterogeneous environments. An ontology is a set of logical axioms that formally define a shared vocabulary [5]. By committing to a common ontology agents can make assertions or ask queries that are understood by the other agents. By featuring logic-based representation languages ontologies provide executable calculi that allow querying and reasoning during run-time, which is required for automatic contract enforcement. Furthermore, by using a standardized ontology language the maximal degree of interoperability

between different heterogeneous systems can be ensured. This is crucial when dealing with open environments as in the case of Web services.

In the remainder of this section we briefly introduce the formalism as well as the ontology framework used to define the contract ontology.

4.1. Ontology Formalism

In order to guarantee that the formal definitions are understood by other parties in the web, the underlying logic has to be standardized. The Web Ontology Language (OWL) standardized by the World Wide Web Consortium (W3C) is a first effort in this direction. OWL-DL is a decidable fragment of OWL and is based on a family of frame-based knowledge representation formalisms called *Description Logics* (DL) [6]. The meaning of the modeling constructs (such as concepts, data types, individuals and data values) provided by OWL-DL [7] is formally defined via a model-theoretic semantics, i.e. it is defined by relating the language syntax to a model consisting of a set of objects, denoted by a domain, and an interpretation function, which maps entities of the ontology to concrete entities in the domain [7].

In order to define the contract ontology, we require additional modeling primitives not provided by OWL (e.g. triangle relations between concepts). The Semantic Web Rule Language (SWRL) [8] allows us to combine rule approaches with OWL and thus model such knowledge. Since reasoning with knowledge bases that contain arbitrary SWRL expression usually become undecidable [9], we restrict ourselves to *DL-safe rules* [10]. A rule is *DL-safe* in case each variable occurring in the rule also occurs in a non-DL-atom in the body of the rule. This means the identity of all objects referred to in the rule has to be known explicitly. To query and reason over a knowledge base containing OWL-DL as well as DL-safe SWRL axioms we use the KAON2 inference engine [11].

For the reader's convenience we define DL axioms either in DL abstract syntax [6] or informally via UML class diagrams [12], where UML classes correspond to OWL concepts, UML associations to object properties, UML inheritance to subconcept relations and UML attributes to OWL data type properties. For representing rules we rely on the standard rule syntax as done in [8].

4.2. Modeling Basis

As shown in Figure 1, instead of engineering a contract ontology entirely from the scratch we rely on the foundational ontology DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) [13] and the DOLCE modules Ontology of Descriptions and Situations (DnS), Ontology of Plans (OoP), Ontology of Information Objects (OIO).

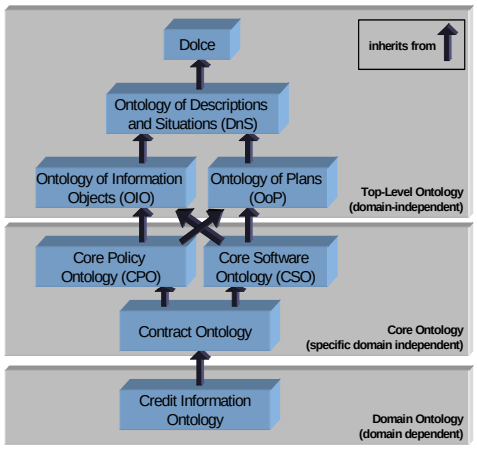

Figure 1 Ontology framework

DnS provides a theory of contextualization by introducing the distinction between descriptive and ground entities. Descriptive entities are aggregated by a *DnS:Description* and represent non-physical objects like product descriptions or legal norms.[1] Ground entities derived from DOLCE constitute a *DnS:Situation* that represents information about a concrete state of affaire in the world such as a concrete web service invocation or a legal case. Furthermore, the *DnS:satisfies*-relation between a *DnS:Situation* and a DnS:Description specifies if the descriptive entities "describe" the *DnS:Situation* according to specified rules. Moreover, we rely on the ontology module Ontology of Plans (DDOP) to describe social and cognitive plans such as goals and task and the module Ontology of Information Objects (DDIO) which introduces primitives for describing information items. These top level ontologies are discussed in [14].

Based on the DOLCE modules, we reuse further 'core ontologies' which are still domain-independent but geared towards a specific topic. The Core Software Ontology (CSO) provides a clear distinction between information (e.g. software or data) and the digital realization of information in information systems. The Core Policy Ontology [15, 16] allows the specification of obligations and rights which are used in the Contract Ontology to define contractual clauses. The structure of the Core Policy Ontology (CPO) is in line with the ontology design pattern DnS. In general, CPO distinguishes between a *CPO:Policy* specializing a *DNS:Description* and a concrete *DnS:Situation* which is evaluated according to the *CPO:Policy*. As depicted

[1] Concepts, relations and rules contained in the ontology are highlighted in *italics*. For concepts and relations that are introduced directly in the contract ontology, namespaces are omitted. For those that are derived from other ontologies, the corresponding namespace is mentioned explicitly.

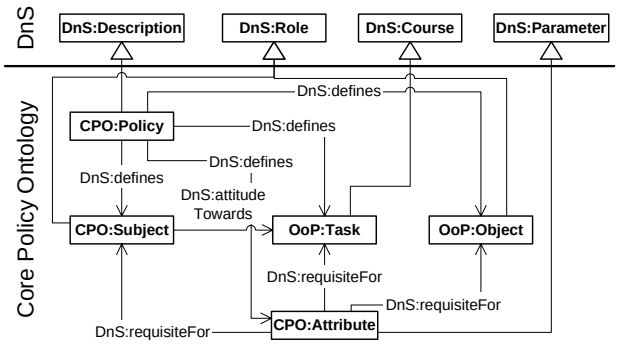

Figure 2: Core Policy Ontology

in Figure 2, a *CPO:Policy* contains *CPO:Subjects* and *CPO:Objects* which are modeled as *DnS:Roles* to allow specifying policies on an abstract level without referring to concrete entities. This is also essential for modeling contracts, since they are usually applicable in many different settings. Further, the *CPO:Policy* determines the *OoP:Task* that is regulated as well as *CPO:Attributes* defining the constraints that have to be met to fulfill a certain *CPO:Policy*. By refining the *DnS:attitudeTowards*-relation we can model different deontic relations between *CPO:Subject* and *OoP:Task,* e.g. we can say that a *CPO:Subject* either has the *DnS:rightTo* or is *DnS:obligedTo* execute a certain *OoP:Task*. In order to find out if a certain *DnS:Situation* conforms to a *DnS:Policy* a specialization of the *DnS:satisfies*-relation is used. For more in-depth discussion of the CPO please refer to [15, 16].

The remainder of the paper is organized in line with the structure of the CPO. In the next section we show how *CPO:Policies* can be used to model an ontology for representing Web service contracts. Subsequently in section 6 monitoring information is modeled as a *DnS:Situation* and finally the *DnS:satisfies*-relation is refined to check whether the *DnS:Situation* conforms to the contract (section 7).

5. Contract Representation

We introduce the contract ontology in three parts: First, we present the body of the contract which contains regulations about who has to accomplish which task. After that we discuss concrete contract clauses that impose certain conditions on regulations in the contract. Finally, we show how domain ontologies can be used to apply the contract ontology in a concrete scenario.

5.1. Basic contractual concepts

A contract can be seen as a "legally enforceable agreement in which two or more parties commit to certain obligations in return for certain rights" [17]. In a manual contract this could read as follows:

"The Provider shall provide the [specified] Credit Information Services to the Customer and grant the [specified] license and right of use of the Credit Information. The Customer shall pay for Credit Information Services [the specified amount] to the Provider."

That means contracts define obligations as well as permissions that are binding on all contractors as well as the sequence in which they enter into force. In the case of Web services we restrict ourselves to contracts between exactly two parties, namely *Provider* and *Customer*. We consider a *Contract* as a *DnS:Description* containing at least one CPO:Policy that regulates the transaction. This fact is reflected by the following DL-axiom:

$$Contract \equiv DnS{:}Description \sqcap \exists DOLCE{:}part.CPO{:}Policy$$

In the context of contracts, a CPO:Policy represents either an *Obligation* or a *Permission*. An *Obligation* is a *CPO:Policy* where the *DnS:attitudeTowards*-relation is refined to *DnS:obligedTo* and a *Permission* a *CPO:Policy* where it is refined to *DnS:rightTo.* This can be formalized using the following SWRL rules:

$$Obligation(x) \leftarrow DnS{:}defines(x,y), Subject(y), DnS{:}defines(x,z), Task(z), DnS{:}obligedTo(y,z)$$

$$Permission(x) \leftarrow DnS{:}defines(x,y), Subject(y), DnS{:}defines(x,z), Task(z), DnS{:}rightTo(y,z)$$

Consequently, as depicted in Figure 3 the most elementary contract about purchasing Web services in exchange for money results in two simple *Obligations*: (i) A *Provider Obligation* that specifies that the *Provider* is obliged to make certain functionality accessible to the *Customer.* We call this activity *Service Task*. In this case the *CPO:Subject* is specialized to a *Provider, the OoP:Task* to *Service Task* and the *CPO:Object* to *Trading Object*. (ii) A *Customer Obligation* which specifies that the *Customer* is obliged to compensate the *Provider* for using the Web service. This activity is called *Compensation Task* and mostly involves the transfer of a certain amount of money. To define a *Compensation Task* the *CPO:Subject* is specialized to a *Customer, the OoP:Task* to *Compensation Task* and the *CPO:Object* to *Compensation Object*. The two *Obligations* forming a *Contract* are shown in Figure 3**.**

Note that the distinction between *Service Tasks* and *Trading Objects* allows modeling the promised functionality of a service using either explicit or implicit capability representation [18]. This enables our contract ontology to support all major efforts striving for semantic web service descriptions such as WSMO [19], OWL-S [20] and WSDL-S [21].

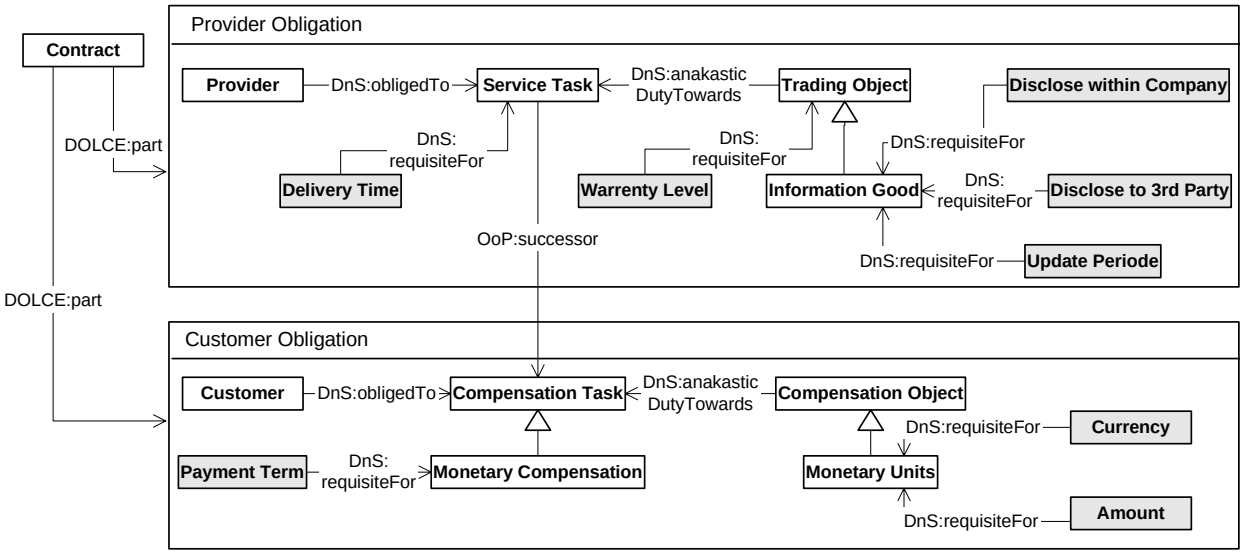

Figure 3 Contract ontology. Note that plotting UML classes within an Obligation-class illustrates a DnS:defines-relation between the Obligation and the contained classes.

Usually contracts also specify in which sequence obligations have to be fulfilled and rights are obtained. In our sample contract, for instance, the *Send Information* task has to be executed before the *Compensation Task* of the *Customer*. Hence, means for representing sequences of *OoP:Tasks* are required. We reuse the Ontology of Plans which provides primitives for modeling complex processes, e.g. *Sequential Tasks*, *Parallel Tasks, Loop Tasks,* etc. In this context, the primary ordering relation for *OoP:Tasks* are *OoP:directSuccessor* and its transitive version *OoP:successor*. In Figure 3 for example, we use the *OoP:successor*-relation to state that the *Service Task* has to be executed before the *Compensation Task.* Such information is crucial for figuring out which party violated a *Contract*.

5.2. Individual Contract Clauses

As discussed above, a contract imposes further conditions that have to be fulfilled by the contractors. These conditions can be modeled as constraints on the concepts contained in a *CPO:Policy*. We realize this by means of the *CPO:Attribute* concept as follows:

Update period (§ 2) As specified above, update period*s* are warranted by the provider. A legal text negotiated by human beings could read as follows:

"The Provider warrants that it reviews and, if necessary, updates Business Background Information every month."

Since this is a property of the *Trading Object* we introduce *Update Period* as a subclass of *CPO:Attribute* which constrains the *Trading Object Information Good*. It is *DnS:valuedBy* a *DOLCE:Region XSD:Integer*.

$$InformationGood \sqsubseteq TradingObject$$

$$UpdatePeriod \sqsubseteq CPO{:}Attribute \sqcap \exists DnS.requisiteFor.InformationGood \sqcap \forall DnS.requisiteFor.InformationGood \sqcap \exists DnS{:}valuedBy.XSD{:}Integer$$

Allowed usage of purchased information (§ 3): Typically licenses regulate how information can be used. A negotiated legal text could read as follows:

"The Provider grants the customer a non-transferable license to use the Credit Information delivered under the terms of this contract. The Customer may freely copy or forward Credit Information within its company. The Customer may not disclose or make the Credit Information otherwise available to third parties without prior consent of the Provider. "

The license specifies if the right to use a certain *Information Good* is *Transferable,* if the *Customer* may disclose the *Information Good* within the company (*Disclose within Company*) or to external third parties (*Disclose to 3rd Party*). These usage terms are also modeled as *CPO:Attributes* of *Information Goods.* The following DL axiom formalizes the *CPO:Attribute Transferable* which is *DnS:valuedBy* a *DOLCE:Region* containing the two *XSD:Strings* "*yes*" and "*no*". The *CPO:Attribtues Disclose within Company* and *Disclose to 3rd Party* are formalized analogously.

$$Transferable \sqsubseteq CPO{:}Attribute \sqcap \exists DnS.requisiteFor.InformationGood \sqcap \forall DnS.requisiteFor.InformationGood \sqcap \exists DnS{:}valuedBy.\{\text{“}yes\text{”},\text{“ }no\text{”}\}$$

Warranties (§ 4) : As explained above, warranties are an important part of negotiations. A legal warranty clause negotiated by human beings could contain the following element:

"The Provider warrants that the credit information is 100% accurate."

Therefore, we add a *DnS:Attribute Warranty Level* which is valued by a *DOLCE:Region* reflecting the three different warranty levels defined in the legal discussion above. Since the warranty can be considered as a fundamental property of a *Trading Object* we model *the Warranty Level* as a *DnS:Attribute* of *Trading Object*.

$$\begin{aligned} &\mathit{WarrantyLevel} \sqsubseteq \mathit{CPO{:}Attribute} \sqcap \\ &\quad \exists \mathit{DnS.requisiteFor.TradingObject} \sqcap \\ &\quad \forall \mathit{DnS.requisiteFor.TradingObject} \sqcap \\ &\quad \exists \mathit{DnS{:}valuedBy}.\{1,2,3\} \end{aligned}$$

Note that since the *Warranty Levels* are only vague terms a conversion-rule has to be used to map the levels to concrete accuracy guarantees.

Delivery Time (§ 4):

"The Provider shall deliver the credit information service 5 seconds after conclusion of the contract."

The *CPO:Attribute* Delivery Time specifies the period in which the Service Tas*k* has to be executed. Hence, it is modeled as a constraint of *Service Task* which is *DnS:valuedBy* an *XSD:Integer*.

$$\begin{aligned} &\mathit{DeliveryTime} \sqsubseteq \mathit{CPO{:}Attribute} \sqcap \\ &\quad \exists \mathit{DnS.requisiteForServiceTask} \sqcap \\ &\quad \forall \mathit{DnS.requisiteFor.ServiceTask} \sqcap \\ &\quad \exists \mathit{DnS{:}valuedBy.XSD{:}Integer} \end{aligned}$$

Prices and Payment Terms (§ 6):

"The price for the Business Background Information is EUR 15."

Mostly the provision of goods and services is compensated by monetary payments. We call a *Compensation Task* where *Monetary Units* are transferred *Monetary Compensation. Monetary Units* are mandatory described by the DnS:Attributes *Amount* which is *DnS:valuedBy* a *XSD:Float* representing the number of *Monetary Units* to be transferred and *Currency* which is *DnS:valuedBy* a *XSD:String* representing the currency the *Amount* is specified. This is formalized by the following axioms.

$$\begin{aligned} &\mathit{MonetaryUnits} \sqsubseteq \mathit{CompensationObject} \sqcap \\ &\quad \exists \mathit{DnS.anakasticDutyTowards.CompensationTask} \sqcap \\ &\quad \forall \mathit{DnS.anakasticDutyTowards.CompensationTask} \sqcap \\ &\quad \exists \mathit{DnS{:}requisites.Amount} \sqcap \exists \mathit{DnS{:}requisites.Currency} \end{aligned}$$

$$\begin{aligned} &\mathit{Amount} \sqsubseteq \mathit{CPO{:}Attribute} \sqcap \\ &\quad \exists \mathit{DnS{:}requisitesFor.MonetaryUnits} \sqcap \\ &\quad \forall \mathit{DnS{:}requisitesFor.MonetaryUnits} \sqcap \\ &\quad \exists \mathit{DnS{:}valuedBy.XSD{:}Float} \end{aligned}$$

$$\begin{aligned} &\mathit{Currency} \sqsubseteq \mathit{CPO{:}Attribute} \sqcap \\ &\quad \exists \mathit{DnS{:}requisitesFor.MonetaryUnits} \sqcap \\ &\quad \forall \mathit{DnS{:}requisitesFor.MonetaryUnits} \sqcap \\ &\quad \exists \mathit{DnS{:}valuedBy.XSD{:}String} \end{aligned}$$

Furthermore, a contract usually contains a *Payment Term* that specifies in which timeframe a *Monetary Compensation* has to take place. We model the *Payment Term* as a *CPO:Attribute* constraining the *Monetary Compensation* task.

$$\begin{aligned} &\mathit{PaymentTerm} \sqsubseteq \mathit{CPO{:}Attribute} \sqcap \\ &\quad \exists \mathit{DnS{:}requisiteFor.CompensationTask} \sqcap \\ &\quad \exists \mathit{DnS{:}requisiteFor.CompensationTask} \sqcap \\ &\quad \exists \mathit{DnS{:}valuedBy.DOLCE{:}Temporal\text{-}Region} \end{aligned}$$

Of course, all regulations specified above can be extended in case it is required by a certain application. This is realized either by introducing new *CPO:Attributes* within an existing *CPO:Policy* or by adding further *CPO:Policies* to the *Contract*.

5.3. Domain Ontology

In order to apply the contract ontology in a concrete application scenario, domain ontologies are required to introduce concepts and relations required for specializing *Trading Objects* as well as *Service* and *Compensation Tasks.*

Since in our credit information scenario we deal with information services, the functionality of a service can be specified by introducing the *Send Information* and *Monetary Compensation* tasks, which concretize *Service Task* and *Compensation Task*, respectively. Furthermore, in order to define the functionality of a service specifying the service output is required, which can be done by means of a reference to a specific type of information. In the following we discuss a domain ontology dealing with *Trading Objects* in the credit information services example.

As introduced above, there are five main categories of *Credit Information* which are named in an exemplary clause of the umbrella agreement. Ontologically, information mentioned above is represented by the concept *CSO:Data* that is *OIO:realized* by a *CSO:Computational Object* within the information system. CSO:Data is a *OIO:Information Object* which is – in contrast to *CSO:Software* – not executable. Hence, we formally define different types of *Credit Information* as follows:

CreditInformation ⊑ *CSO:Data* ⊓
∃*DOLCE:part.CompanyIdentifier* ⊓
∀*DOLCE:part.(BusinessBackgroundInformation* ⊔
CreditScoreCalculationInformation ⊔
QualityOfCompanyInformation ⊔
CreditLimitCalculation ⊔ *WarningInformation*)

BusinessBackgroundInformation ⊑ *CSO:Data* ⊓
∃*DOLCE:part.CompanyIdentifier* ⊓
∀*DOLCE:parts.(OwnershipInformation* ⊔ *History* ⊔
Principles ⊔ *Operations* ⊔ *Location*)

CompleteBusinessBackgroundInformation ⊑ *CSO:Data*
⊓ ∃*DOLCE:part.CompanyIdentifie r*⊓
∃*DOLCE:part.Ownership* ⊓ ∃*DOLCE:part.History*
⊓ ∃*DOLCE:part.Principle*s ⊓
∃*DOLCE:part.Operations* ⊓ ∃*DOLCE:part.Location*

OwnershipInformation ⊑ *CSO:Data* ⊓
∃*DOLCE:part.CompanyIdentifier* ⊓
∃*DOLCE:part.Ownership*

The other types of *Credit Information* are defined analogously. Note that this modeling approach enables a DL-reasoner to automatically infer a *Credit Information* hierarchy, e.g. it is inferred that *Ownership Information* is a subclass of *Business Background Information* as well as *Credit Information* or that *Complete Business Background Information is a subclass of Business Background Information.*

6. Representation of Monitoring Information

In the last section, we presented contract information as a collection of *CPO:Policies* which are modeled by refining *DnS:Descriptions*. In this section, we extend this approach in order to represent *Monitoring Information*. Since this is information about real world activities, it is ontologically modeled as a *DnS:Situation*. As sketched in Figure 4, *DnS:Situations* use DOLCE ground entities to model concrete activities in a real world information system, i.e. in our case a concrete web service invocation. According to the Core Software Ontology (CSO) [16] such activities are called *CSO:Computational Activities. CSO:Computational Activities* are *DOLCE:Perdurants* that involve *CSO:Computational Objects* which realize either the *CSO:Data* or *CSO:Software*. While *CSO:Software* can be seen as an active entity that *executes CSO:Computational Activities*, *CSO:Data* is only passively *involved in CSO:Computational Activities.* Therefore, we introduce the two relations *executes* and *involved in* as follows:

executes(x,y) ← *DOLCE:participateIn(x,y),*
CSO:ComputationalObject(x),
CSO:ComputationalActivity(y),
OIO:realizes(x,z),CSO:Software(z)

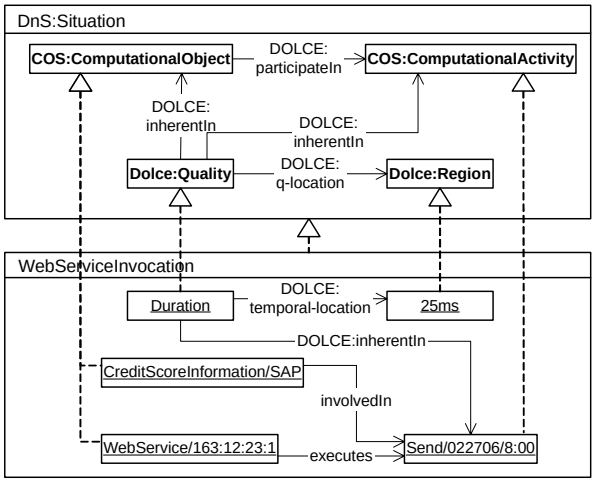

Figure 4 Monitoring information as DnS:Situation. Note that plotting UML classes within a DnS:Situation-class illustrates a DnS:settingFor-relation between the DnS:Situation and the contained classes.

involvedIn(x,y) ← *DOLCE:participateIn(x,y),*
CSO:ComputationalObject(x),
CSO:ComputationalActivity(y),
OIO:realizes(x,z), CSO:Data(z)

Moreover, computational entities inherently exhibit certain *DOLCE:Qualities* which can be seen as attributes that characterize *DOLCE:Endurants* and *DOLCE:Perdurants,* e.g. the duration of an activity or the size of an object. *DOLCE:Qualities* are *DOLCE:localized in DOLCE:Regions*. Consequently, the notion of *Monitoring Information* is defined by the following DL axiom:

Monitoring Information ⊑ *DnS:Situation* ⊓
∃*settingFor.CSO:ComputationalActivity* ⊓
∃*settingFor.CSO:ComputationalObject* ⊓
∀*settingFor.*(*CSO:ComputationalActivity* ⊔
CSO:ComputationalObject ⊔ *DOLCE:Quality* ⊔
DOLCE:Region)

Providing information via a Web service leads to a *CSO:Computational Activity* where one party transfers a *CSO:Computational Object*, e.g. the realization of certain credit information, to another party. In executing this activity various types of monitoring information about the activity itself as well as about participating objects can be measured or perceived, which are represented as *DOLCE:Qualities*.

Furthermore, Figure 4 introduces a concrete example which represents information about a Web service invocation as an instance of *Monitoring Information.* The *CSO:Computational Activity* that is

monitored is a specific *Send*-activity carried out on February 27th, 2006 at 8am. The activity was executed by a Web service with the IP-address 163:12:23:1 and involved the digital representation of *Credit Score Information* of the company SAP. Moreover, the *DOLCE:Quality Duration* of the *Send*-activity is measured and results in a *DOLCE:Region* of *25ms*. Of course, other *DOLCE:Qualitites* of the *CSO:Computational Activity* as well as the *CSO:Computational Object* beyond *Duration* can be determined in the monitoring step. However, introducing methods for Web services monitoring is not in the focus of this paper. More information about how to derive monitoring information can be found in [22], for instance.

7. Contract Evaluation

After introducing the notion of Contract as specialization of DnS:Description and Monitoring Information as specialization of DnS:Situation we can use the DnS:satisfies-relation that holds between DnS:Situations and DnS:Descriptions in order to determine whether the two parts match according to a specified rule. However, the DnS:satisfies-relation stemming from DnS is defined on a very abstract level and thus has to be adapted to our concrete application, i.e. contract evaluation.

However, evaluating contracts is highly complex. This is mainly due to context-dependent and fuzzy interpretations that are required in the evaluation process. For example, certain actions have to be done "immediately" or "with utmost care". Nevertheless, lawyers typically use guidelines how to interpret statements and expressions which can be captured by the *DnS:satisfies*-relation and thus used for automatic contract evaluation. For instance, in case the term "immediately" is used to specify a timeframe in which a response of the service is expected, one could use a rule of thumb that says the response is received "immediately" only if it is received within 5 sec. after sending the request. Of course, such rules are always highly domain dependent and have to be verified regularly for their validity. For instance, it might be necessary to adapt them to new court decisions.

Subsequently, we exemplify this approach using the following *Provider Obligation*: The credit information service provider *X* has to provide a complete set of *Business Background Information* of Company *Z* to customer *Y*. This has to be done *immediately* after closing the contract. Therefore, we derive the following formal definition of the *Provider Obligation:*

Obligation(ProviderObligationX)
Provider(X)
DnS:defines(ProviderObligationX,Y)
TradingObject(BBInformation/Z)
DnS:defines(ProviderObligationX,BBInformation/Z)
OoP:Task(Transfer)
DnS:defines(ProviderObligationX,Transfer)
DeliveryTime(responseTime)
DnS:defines(ProviderObligationX,responseTime)
DnS:Region('immediately')
DnS:defines(ProviderObligationX, 'immediately')
DnS:valuedBy(responseTime, 'immediately')
DnS:obligedTo(X,Transfer)
DnS:anakasticDutyTowards(BBInformation/Z,Transfer)
DnS:requisiteFor(responseTime,Transfer)

After executing the *Contract* above the customer has monitored the *Web Service Invocation* formalized in Figure 4.

Based on the *Monitoring Information* the customer evaluates the *Provider Obligation* presented above. The following questions have to be answered within this evaluation process:

1. Is the requested trading object delivered?
To answer this question we have to find out if information is delivered by the provider and – in case it is – if the delivered information is complete with respect to the agreement in the *Contract*. We realize this by comparing the delivered *CSO:Computational Object* contained in the *Monitoring Information* with the *CSO:Computational Object* agreed on in the contract. This is done by the following SWRL-rule:

$$correctInformation(x,y) \leftarrow DnS:MonitoringInformation(x),\ DnS:Policy(y),\ DnS:defines(y,t),\ TradingObject(t),\ DnS:playedBy(t,d_1),\ CSO:Data(d_1),\ DnS:settingFor(x,d_2),\ CSO:Data(d_2),\ subsumes(d_2,d_1)$$

Note that *subsumes* is implemented as a built-in predicate that verifies if the first argument *classifies* the second, or in other words if the class of the second argument is either an explicit or inferred subclass of the first argument. This verification is done with a DL-reasoner. By using this predicate we allow the provider to send more information than required, while making sure that the information agreed in the contract is provided.

2. Was the correct service task executed?
The SWRL-rule below answers this question by comparing the executed *CSO:Computational Activity* stated in the *Monitoring Information* with the *CSO:Computational Activity* agreed upon in the *Contract*. Again we use the *subsumes*-predicate to allow a general activity description in the contract to be fulfilled by a specific activity. For example, a contract might specify that certain information has to be *transferred*. How this should be done is not specified

exactly. Therefore, *sending by mail* or *telling on the phone* might be admissible since both are certain types for transferring information.

correctActivity(x,y) ← *DnS:MonitoringInformation(x), DnS:Policy(y), DnS:defines(y,t), OoP:ServiceTask(t), DnS:sequences(t,a_1), CSO:ComputationalActivity(a_1), DnS:settingFor(x,a_2), ComputationalActivity(a_2), subsumes(a_1,a_2)*

3. Was the task executed within the required timeframe?

According to the Contract Ontology a *Service Task* has to be executed within a certain *Delivery Time.* This is verified by the rule below, which compares the *Quality* specifying the response time of the invocation with the *DeliveryTime* in the contract.

activityInTime(x,y) ← *DnS:MonitoringInformation(x), DnS:Policy(y), DnS:defines(y,t), OoP:ServiceTask(t), DeliveryTime(d), DnS:requisiteFor(d,t), interpretDeliveryTime(d,r_1), DOLCE:Region(r_1), DnS:settingFor(x,a), ComputationalActivity(a), DOLCE:inherentIn(q,a), DOLCE:Quality(q), DOLCE:q-location(q,r_2), DOLCE:Region(r_2), subsumes(r_1,r_2)*

However, since *DeliveryTime* is expressed by a *XSD:String* rather than a *DOLCE:Temporal-Region* we need the following conversion-rule. Note that this interpretation of the term "immediately" is not content of the contract but rather common sense knowledge.

interpretDeliveryTime(d,r_2) ← *DeliveryTime(d), DnS:valuedBy(d,r_1), DOLCE:Region(r_1), SWRLB:equals(r_1,'immediately'), DOLCE:Temporal-Region('<5s'), DOLCE:Temporal-Region(r_2), SWRLB:equals(r,'<5s')*

After the conversion both measures are expressed via *DOLCE:Temporal-Regions* and thus can compared by the *subsumes*-relation. A *DOLCE:Temporal-Regions subsumes* another one if the timeframe is larger. Thus, *Durations* that are shorter than the period specified in *Delivery Time* are admissible.

If one of the above questions is not evaluated to true the contract is considered as violated. Therefore, the following conjunctive rule specializes *DnS:satisfies* and thereby aggregates the results of questions above.

conformsTo(x,y) ← *DnS:satisfies(x,y), DnS:MonitoringInformation(x), DnS:Policy(y), correctInformation(x,y), correctActivity(x,y), activityInTime(x,y)*

If we have more complex contracts than in this example (e.g. more than one *CPO:Policy* in the *Contract)* the *conformsTo*-rule has to be adapted accordingly. Although it would be desirable to define the *conformsTo-relation* in a more general way, this is not possible with the formalism at hand. Since we do not have universal quantification in SWRL, we have to explicitly iterate over all *DnS:Concepts* in a *CPO:Policy* and over all *CPO:Policies* in a *Contract.* However, we belief that remaining in a standardized and decidable fragment of the logic justifies this limitation.

8. Related Work

There are several approaches that strive for electronic contract representation in B2B scenarios, ranging from EDI to XML-based languages. However, proprietary approaches such as EDI are not suitable for open, fast-changing environments like the Internet, because the introduction of these technologies requires high investments which could easily lead to lock-in effects.

XML-based languages tackle this problem (e.g. ebXML [23]). They introduce machine-readable contract models that can be considered as a structured contract document. Some of them are designed for a Web service scenario. The Web Service Level Agreement (WSLA) project focuses on the quality of service aspect within a contract [24]. It also addresses the monitoring of an agreement. However, the project covers only some specific elements of a contract. While WS-Agreement takes a similar approach, it is not restricted to quality aspects since it is based on WS Policy [25]. However, all the approaches mentioned above do not provide formal and explicit semantics. But this is required to facilitate interoperability in an open environment where standardization is not always possible and desirable. In addition, pure XML-based languages do usually not support the logical reasoning which is required during contract evaluation.

To meet this requirement several logic-based contract languages have been proposed. Most of them are based on deontic logic (e.g. the ODP Enterprise Language [26]) which extends first order logic by modal operators like 'may' and 'must'. However, these languages are not based on internet standards and thus are not directly applicable to Web service contracting. RuleML is a first attempt to establish a standardized syntax for exchanging logical rules [27]. Under the umbrella of RuleML different kinds of logics can be used to express contracts (e.g. courteous logic programs [28] or defeasible logic [29]). However, RuleML does only deal with standardizing the syntax and not with standardizing the underlying logic. Thus, interoperability still seems to be a problem.

By building on a standardized knowledge representation language like OWL as well as a sound and highly axiomatized toplevel ontology we are able to derive a clean and extensible conceptual model that provides interoperability in open environments and facilitates the integration of heterogeneous modeling

efforts. We extend this approach for expressing rules by adding a decidable fragment of the emerging standard SWRL (called DL-safe rules). This provides us with an expressive – but still decidable - language for representing Web service contracts.

9. Conclusion

We presented a modeling framework for Web service contracts. The approach does not aim towards full automation, but rather enables semi-automatic contracting. In our opinion full automation is – at least for the moment – not feasible across organizational boundaries. Therefore, we suggest combining an umbrella contract covering static aspects with a formalized description of clauses that are dynamically negotiated and monitored. In contrast to related approaches, for the formalization of the contract we rely on internet standards to facilitate interoperability in a Web environment and on foundational ontologies in order to derive a sound conceptual model.

10. References

[1] Angelov, S. and P. Grefen, *B2B eContract Handling - A Survey of Projects, Papers and Standards*, Technical Report, University of Twente, The Netherlands, 2001.

[2] Lindemann, M. A. and B. F. Schmid, *Elements of a Reference Model for Electronic Markets*, In HICSS '98: Proceedings of the Thirty-First Annual Hawaii International Conference on System Sciences-Volume 4, Hawaii, USA, 1998.

[3] Pinto, H. S. and J. P. Martins, *Ontologies: How can They be Built?* Knowledge Information System, 6 (2004), pp. 441-464.

[4] Sester, P. and T. Nitschke, *Software-Agent mit der Lizenz zum.?* Computer und Recht (CR) (2004), pp. 548-554.

[5] Gruber, T. R., *A translation approach to portable ontologies*, Knowledge Acquisition, 5 (1993), pp. 199-220.

[6] Baader, F., D. Calvanese, D. McGuinness, D. Nardi and P. F. Patel-Schneider, eds., *The Description Logic Handbook: Theory Implemenation and Applications*, 2003.

[7] Horrocks, I., P. F. Patel-Schneider and F. v. Harmelen, *From SHIQ and RDF to OWL: The making of a web ontology language*, Journal of Web Semantics: Science, Services and Agents on the World Wide Web, 1 (2003), pp. 7-26.

[8] Horrocks, I., P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof and M. Dean, *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*, Technical Report, W3C Submission, 2004.

[9] Horrocks, I. and P. F. Patel-Schneider, *A proposal for an OWL rules language*, In WWW '04: Proceedings of the 13th international conference on World Wide Web, New York, NY, USA, 2004.

[10] Motik, B., U. Sattler and R. Studer, *Query Answering for OWL-DL with Rules*, Journal of Web Semantics: Science, Services and Agents on the World Wide Web, 3 (2005), pp. 41-60.

[11] Kaon, *Karlsruhe Ontology Framework 2 (KAON2)*, http://kaon2.semanticweb.org, 2004,

[12] Brockmans, S., R. Volz, A. Eberhart and P. Loeffler, *Visual modeling of OWL DL ontologies using UML*, *Proceedings of the Third International Semantic Web Conference (ISWC2004)*, Springer, Hiroshima, Japan, 2004.

[13] Masolo, C., S. Borgo, A. Gangemi, N. Guarino, A. Oltramari and L. Schneider, *The WonderWeb library of foundational ontologies: preliminary report*, Padova, Italy, 2002.

[14] Gangemi, A., S. Borgo, C. Catenacci and J. Lehmann, *Task taxonomies for knowledge content*, Technical Report, Metokis deliverable D07, 2004.

[15] Lamparter, S., A. Eberhart and D. Oberle, *Approximating Service Utility from Policies and Value Function Patterns*, In 6th IEEE Int. Workshop on Policies for Distributed Systems and Networks, Stockholm, Sweden, 2005.

[16] Oberle, D., S. Lamparter, S. Grimm, D. Vrandecic and A. Gangemi, *Towards Ontologies for Formalizing Modularization and Communication in Large Software Systems*, To appear in Journal of Applied Ontology (2006).

[17] Reinecke, J., G. Dessler and W. Schoell, *Introduction to Business - A Contemporary View*, Allyn and Bacon, Boston, 1989.

[18] Sycara, K., M. Paolucci, A. Ankolekar and N. Srinivasan, *Automated Discovery, Interaction and Composition of Semantic Web Services*, Journal of Web Semantics: Science, Services and Agents on the World Wide Web, 1 (2003).

[19] WSMO, *Web Service Modeling Ontology*, http://www.wsmo.org, 13 April 2005, 2005,

[20] *OWL-S*, http://www.daml.org/services/owl-s/, November 2004,

[21] WSDL-S, *Web Service Semantics - WSDL-S*, W3C Submission, http://www.w3.org/Submission/WSDL-S/, 7 November 2005, W3C, 2005,

[22] Sahai, A., V. Machiraju, M. Sayal, A. v. Moorsel and F. Casati, *Automated SLA Monitoring for Web Services*, In Proceedings of the 13th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management: Management Technologies for E-Commerce and E-Business Applications, 2002.

[23] Organization for the Advancement of Structured Information Standards, *Enabling a global electronic market: ebXML*, http://www.ebxml.org,

[24] IBM, *Web Service Level Agreements (WSLA) Project*, http://www.research.ibm.com/wsla/about.html,

[25] Forge, *Grid Resource Allocation Agreement Protocol. Web Services Specification*, https://forge.gridforum.org/projects/graap-wg/document/WS-AgreementSpecification,

[26] *RM-ODP Enterprise Language [X.911 I IS 15414]*, http://www.joaquin.net/ODP/DIS_15414_X.911.pdf,

[27] *The Rule Markup Initiative, February 26th, 2005*, http://www.ruleml.org/,

[28] Grosof, B. and T. Poon, *SweetDeal: Representing agent contracts with exceptions using XML rules, ontologies, and process descriptions*, In Proceedings of the 12th World Wide Web Conference, Budapest, Hungary, 2003.

[29] Governatori, G., *Representing business contracts in RuleML*, International Journal of Cooperative Information Systems, 14 (2005), pp. 181-216.