Integration of rules and policies for Semantic Web Services

Nima Kaviani*

Simon Fraser University Surrey, Canada
E-mail: nkaviani@sfu.ca
Website: http://www.sfu.ca/~nkaviani
*Corresponding author

Dragan Gašević

Athabasca University, Canada
E-mail: dgasevic@acm.org
Website: http://www.sfu.ca/~dgasevic

Marek Hatala

Simon Fraser University Surrey, Canada
E-mail: mhatala@sfu.ca
Website: http://www.sfu.ca/~mhatala.

David Clement

Visiphor Corporation, Canada
E-mail: david.clement@visiphor.com

Gerd Wagner

Brandenburg University of Technology, Cottbus, Germany
E-mail: wagnerg@tu-cottbus.de
Website: http://www.informatik.tu-cottbus.de/~gwagner/
Website: http://www.rewerse.net/i1

Abstract: Multimedia service providers on the web need their services to be well protected and easily accessible worldwide. This has initiated several lines of research to provide semantically rich policy-aware web services. However, the existence of various technologies to define and protect services has made the communication of services questionable. Currently, it is ambiguous how a broker agent looking for a special object would communicate to an online service provider, with different policy and service definition languages, to obtain the appropriate object. Here, we propose a solution to such problems by introducing an interchange framework to transform business rules and policies between different enterprises using *REWERSE Rule Markup Language*.

Keywords: semantic web rule languages; SWL; SWRL; policy languages; semantic web services; policy-driven communication; rule interchange format; RIF.

Reference to this paper should be made as follows: Kaviani, N., Gašević, D., Hatala, M., Clement, D. and Wagner, G. (2007) 'Integration of rules and policies for Semantic Web Services', *Int. J. Advanced Media and Communication*, Vol. 1, No. 4, pp.404–423.

Biographical notes: Nima Kaviani is a Master's student in the School of Interactive Arts and Technology (SIAT), at Simon Fraser University, Canada. His research interests are mainly in the area of trust and policy management for semantic web services. He has also an extensive background in the area of robotic and multi-agent systems.

Dragan Gašević is an Assistant Professor in the School of Computing and Information Systems at Athabasca University. He received his BSc, MSc and PhD Degrees in Computer Science from the University of Belgrade. His research interests are in the areas of model-driven (software) engineering, semantic web, web engineering, service-oriented architectures, software engineering knowledge management, learning technologie, and Petri nets.

Marek Hatala is an Associate Professor in the School Interactive Arts and Technology at Simon Fraser University and the Director of the Laboratory for Ontological Research. He received his MSc in Computer Science and PhD in Cybernetics and Artificial Intelligence from the Technical University of Kosice. He is also the Leader of Theme 1 of Canada's LORNET Research Network. His interests are in areas of knowledge representation, ontology and semantic web, user modelling, intelligent information retrieval, organisational learning and e-learning.

David Clement is the Chief Technology Officer of Visiphor Corporation. He is the visionary Chief Architect and Designer of Visiphor's Briyante integration software solutions. He has 25 years prior experience in Canada, the USA, UK and Australia driving software solutions from conception to delivery. His background includes computer graphics, distributed computing and large-scale system architecture.

Gerd Wagner received a PhD Degree in philosophy in 1993 from the Free University of Berlin and a Habilitation Degree in Informatics in 1998 from the University of Leipzig. He is currently a Professor of Internet Technology at Brandenburg University of Technology at Cottbus, Germany. His research interests include agent-oriented modelling and agent-based simulation, foundational ontologies, (business) rule technologies and the semantic web. He is the coordinator of a work package on rule modelling and markup in the European research network REWERSE.

1 Introduction

Semantic Web Services (SWS), as the augmentation of web service descriptions through semantic web annotations, facilitate the higher automation of service discovery, composition, invocation and monitoring on the web (Payne and Lassila, 2004). Semantic web ontologies and the ontology languages (OWL and RDF(S)) are recognised as the main means of knowledge representation for SWS (Sheth et al., 2006). Such ontology-enriched web service descriptions are later used in the negotiation process

between service clients and service providers, which is defined by a set of abstract protocols of Semantic Web Service Architecture (SWSA) (Burstein et al., 2005).

However, the current proposed standards for describing SWS (i.e., OWL-S (Martin et al., 2004), WSDL-S (Akkiraju et al., 2005), web Service Modelling Ontology (de Bruijn et al., 2005) and Semantic web Service Language (SWSL) (Battle et al., 2005)) demonstrate that it is important to use a rule language in addition to ontologies. This allows run-time discovery, composition and orchestration of SWS by defining preconditions or post-conditions for all web service messages exchanged (Kagal et al., 2006). For example, OWL-S recommends using OWL ontologies together with different types of rule languages (SWRL, KIF or DRS), WSMO uses F-Logic, while WSDL-S is fully agnostic about the use of a vocabulary (e.g., UML, ODM, OWL) or rule language (e.g., OCL, SWRL, RuleML). Usually, SWS descriptions use only parts of rules representing logical formulas that may have a Boolean result. It is important to point out that there is no agreement upon which rule language to use for SWS or what type of reasoning (open or closed world assumption) should be supported.

Besides satisfying clients goals when using SWS, trust is another important aspect that should be established between a client and service. Addressing this problem, researchers proposed the use of policy languages. A policy is a rule that specifies under which conditions a resource (or another policy) might be disclosed to a requester (Olmedilla et al., 2004). To define polices on the Semantic web, various policy languages have been proposed such as PeerTrust (Olmedilla et al., 2004), KAoS (Uszok, et al., 2003; Bradshaw et al., 1997), Rei (Kagal et al., 2004), and PROTUNE (Bonatti and Olmedilla, 2005). As (Olmedilla et al., 2004) reports, trust management policies are also defined as parts (most commonly preconditions) of SWS descriptions.

It is obvious that besides various SWS description languages, we have various policy languages and various rule languages. All these languages are based on different syntactic representations and formalisms with no explicitly defined mapping between them. This hampers the use of SWS from two different perspectives. One perspective is automatic negotiation between service client agents and service provides and automatic matchmaking, where agents and matchmakers should be able to 'understand' various rule/policy/service web service description languages. Another perspective is that of a knowledge management worker who needs to be able to express the rules and policies in a single form rather than in a broad variety of forms. To attempt to represent the same rules and policies in many forms is cumbersome, time consuming and error prone but it is the only choice currently available if a broad base of interoperability is required.

In the paper, we start from the presumption that rules encoded in policies and SWS are parts of business rules (Wagner, 2002; Bonatti and Olmedilla, 2005), and that we should be able to share them by using the same representation. Such a rule language should enable modelling different types of rules such as reaction rules considering the event-driven nature of web services (or so-called Event-Condition-Action (ECA) Rules), derivation rules considering the importance of inferring new facts (such as RuleML), and integrity rules considering the deontic, i.e., 'must', nature of policy languages. Unfortunately, current web rule markup languages such as RuleML and SWRL are unable to express all these types of rules.

In our approach, we propose the use of R2ML (Wagner et al., 2006), which addresses almost all use-cases and requirements for a Rule Interchange Format (RIF) (Ginsberg, 2006), along with a set of transformations between SWS description (e.g., WSDL-S), rule and trust management policy languages. We illustrate the benefits

of our approach using a SWSA example where R2ML is used to share SWS descriptions and policies in the process of matchmaking and trust negotiation. In the next section, we motivate our research by describing an example based on the present solutions to SWS.

In Section 2 we motivate our discussion by giving an example on where and how an interchange format will be needed. In Section 3 we provide the background information needed to understand the concepts of this paper. Section 4 is the introduction to our approach on how to provide an interchange format. This is followed by our suggested solutions for SWS description languages and policy languages in Sections 5 and 6 respectively. Finally in Section 7 we discuss some of the advantages and disadvantages of our approach and conclude.

2 Motivation

SWS registries are the new generation of service registries used to facilitate the process of service discovery. Referring to the semantically enriched description of web services, provided in a SWS registry, a broker agent will be able to check the needs of the requesting entity with the functionalities a service offers, and thus can better find the appropriate service. On the other hand different policies may be involved in the process of service discovery to check the constraints that either the service provider or the requesting entity specify to protect the process of communication or sharing the information.

Let us consider a broker agent trying to find a series of songs, movies and photos for a requesting user. Having a collection of service providers known to a set of service registries, with the information about their media collections annotated with regards to some known ontologies, the broker agent must be able to use the web service offered by each service provider, to search for the media it is looking for. Moreover, through referring to the ontology by which the web service has been semantically annotated, the broker agent can understand the exact purpose of the web service as well as the information required by the web service to more effectively search the media collections. This includes, for example, identifying how and where the web service refers to the name of the actors, directors and producers of movies.

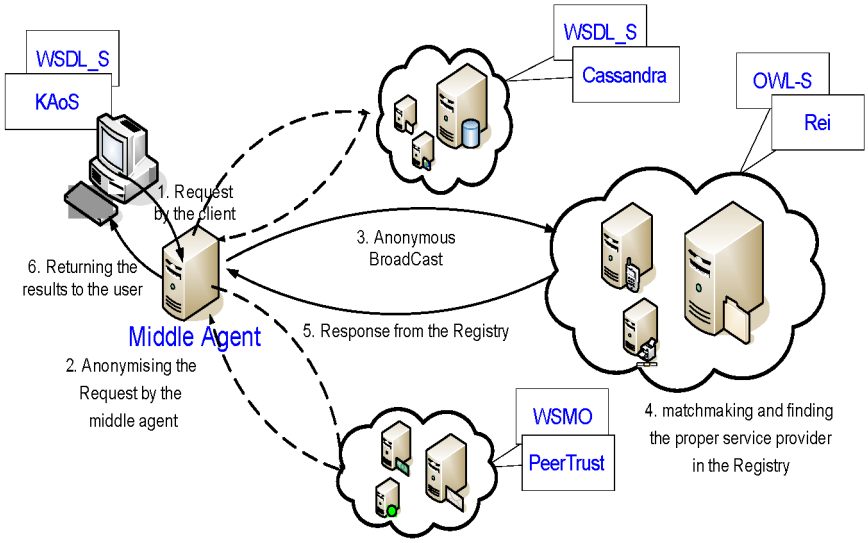
Furthermore, the service providers may need to check for some criteria (e.g., the age of the requesting users) to let the users access some parts of the collections (e.g., thriller movies). Policies can be employed here to compare the identification information of the users, such as Social Insurance No. (SIN) or Credit Card No., with the required constraints. Contrarily, the requesting entity might have policies that declare its reluctance to expose SIN or Credit Card information to service providers that are not vouched by a trusted party such as VeriSign.

A negotiation between the service provider and the service requester may eventually lead to finding the desired song or movie in case both sides of the negotiation can understand each other and can satisfy the constraints defined in their policies. However, due to the diversity of the languages available to define the services and policies, it is possible for different service providers to define their services and policies in dissimilar languages. Olmedilla et al. (2004) assumes that both the requester agent and the service provider have defined their policies in PeerTrust and their SWS description in WSMO. Kagal et al. (2004) gives another suggestion in which Rei, as the policy

language, is used together with OWL-S as the SWS description ontology to describe the service.

As it is shown in Figure 1, different service providers, even those placed in a single registry, may have been using different SWS with different policy languages. Currently proposed architectures and solutions do not deal with how to match the requests and advertisements where the policy languages and the SWS descriptions do not match.

Figure 1 Different entities with dissimilar Semantic Web Service (SWS) description and policy languages may need to communicate and share the information



In the current proposals, the requestors, the broker agents, the registries and the service providers must have different transformers from each policy language or SWS language to the others. This is hard to achieve due to the vast variety of these languages. Moreover, as the flexibility of a mapping from one language to another has not been considered in defining policy and web description languages, chances are that we encounter some information loss during the transformation from one language to another which may result in unwanted changes in the policies of either of the peers and compromise their privacy. In order to solve the problem, the focus should be on finding an interchange format language with the highest possible degree of interoperability to cope with different available SWS languages and policy languages. In the rest of the paper we propose a new solution to interchange the policy languages and SWS description languages, from one to another, with as small information loss as possible.

3 Background

This section gives background knowledge of all concepts relevant for the subject understudy, namely: SWS description languages, policy languages, current approaches to using policy languages with SWS, and web rule languages.

3.1 Semantic Web Service (SWS) description

Considering the main feature of SWS is to have semantically enriched web service descriptions, researchers have so far proposed several solutions to this problem. Here we briefly touch the submitted solutions to W3C.

OWL-S is an OWL-based ontology for describing SWS. The ontology consists of three main parts: the service profile for advertising and discovering services; the process model for detailed descriptions of services' operation; and the grounding, for providing details on how to interoperate with a service via messages (Martin et al., 2004). Some of functionalities that OWL-S provides are: annotating types of input and output parameters of services with OWL-ontology concepts, defining preconditions and effects of input and output messages by logical expressions encoded as literals and XML literals and specifying different types of composite services (e.g., sequence, split and join-split).

Web Service Modelling Ontology (WSMO) consists of ontologies for defining terminology; web service functional and behavioural descriptions, user goals and mediators that automatically handles interoperability between different WSMO element. Comparing to the OWL-S, WSMO defines its own ontology language, while it also defines its own rule language for defining logic expressions. That language has backward compatibility with F-Logic. It is important to note that WSMO introduces post-conditions, besides preconditions and effects that are also present in OWL-S. Post-conditions cover the data output while effects specify general state changes (Grønmo et al., 2005).

Semantic Web Service Ontology (SWSO) is a conceptual model for describing web services (Battle et al., 2005). The complete axiomatisation is given in first-order logic, using the SWSL consisting of SWSL-FOL (First Order Logic) and SWSL-Rules. In fact, SWSO presents a first-order ontology for web services, expressed in SWSL-FOL and its partial translation expressed in SWSL-Rules. SWSO also includes material about grounding its process models with WSDL. It is important to say that SWSL-FOL and SWSL-Rules language are serialised by using RuleML.

WSDL-S is a rather evolutionary approach to SWS that only introduces a few WSDL extensions, and thus preserves the full compatibility with the standard WSDL (Akkiraju et al., 2005). WSDL-S is a full ontology and rule language agnostics meaning that one can annotate WSDL XML schema types with any vocabulary language (e.g., OWL or UML) and define preconditions and effects (there are no post-conditions) with any rule language (e.g., OCL or R2ML).

3.2 Policy languages

As mentioned earlier, there are several policy languages proposed so far with the goal of protecting the privacy of information and authorising requesters by providing different levels of access to the available resources and information. The syntax of these languages varies from rigid ordinary logic languages such as Cassandra (Becker and Sewell, 2004), which is based on Constraint Language Programming, PeerTrust (Olmedilla et al., 2004) and PROTUNE (Bonatti and Olmedilla, 2005), which use a Prolog meta-interpreter, to more relaxed markup languages such as KAoS (Bradshaw et al., 1997; Uszok et al., 2003) and Rei (Kagal et al., 2006). Referring to all of the available policy languages is beyond the purpose of this paper and here we just mention ones the most relevant to our problem.

PeerTrust is a trust negotiation engine for semantic web and P2P networks (Olmedilla et al., 2004). PeerTrust's language is based on first order Horn rules (definite Horn clauses), i.e., rules of the form:

$$\text{lit}_0 \leftarrow \text{lit}_1, \ldots, \text{lit}_n$$

where each lit_i is a positive literal of the form $P_j(t_1 \ldots, t_n)$, P_j is a predicate symbol, and t_i ($i = 1 \ldots, n$) are the arguments of this predicate (e.g., Figure 6). It can be combined with WSMO to define policy-aware web services as it is proposed in Olmedilla et al. (2004).

Cassandra is another policy-based language based on CLP (Becker and Sewell, 2004). It uses a policy language based on Datalog with constraints and its expressiveness can be adjusted by changing the constraint domain. Policies are specified using the following predicates which govern access control decisions: *permits*(*e*, *a*) specifies who can perform which action; *canActivate*(*e*, *r*) defines who can activate which role (*e* is a member of *r*); *hasActivated*(*e*, *r*) defines who is active in which role; *canDeactivate*(*e*, *r*) specifies who can revoke which role; *isDeactivated*(*e*, *r*) is used to define automatically triggered role revocation. Although aggregation of Cassandra with SWS descriptions has not been proposed yet, its declarative nature makes it suitable to combine it with some of the available SWS, such as WSMO.

Rei is a policy framework that permits specification, analysis and reasoning about declarative policies defined as norms of behaviour (Kagal et al., 2004). Rei adopts a rule-based approach to specify semantic policies. Rei policies restrict domain actions that an entity can/must perform on resources in the environment, allowing policies to be developed as contextually constrained deontic concepts, i.e., right, prohibition, obligation and dispensation. The current version of Rei (2.0) adopts OWL-Lite to specify policies and can reason over any domain knowledge expressed in either RDF or OWL.

KAoS is a framework that provides policy and domain management services for agent and other distributed computing platforms (Uszok et al., 2003). It has been deployed in a wide variety of multi-agent and distributed computing applications. KAoS policy services allow for the specification, management, conflict resolution and enforcement of policies within agent domains. KAoS adopts an ontology-based approach to semantic policy specification. In fact, policies are mainly represented in OWL as ontologies which make it possible to combine them with SWS and then use them to define the policies of a web service provider.

3.3 Semantic Web Service (SWS) policies

As we discussed earlier in the motivation section, policies and SWS description ontologies are combined to provide a policy-aware specification of SWS. The most prominent efforts in this area have been done in Kagal et al. (2004) and Olmedilla et al. (2004). In Kagal et al. (2004), OWL-S and Rei have been chosen as web service ontology and policy language respectively. The main reason in selecting OWL-S seems to be the syntactical consistency between OWL-S and Rei.

In Olmedilla et al. (2004) the authors have chosen WSMO as their description for SWS and mixed it with PeerTrust to add policies. WSMO has been selected because it allows the use of arbitrary logical expressions in the description of the service functionality which gives the authors more complete way of expressing their terms as compared to other approaches. It also uses F-Logic to describe the logical expressions

used in the description of the services which in turn makes it possible to align the trust policies described in PeerTrust with functionality descriptions in WSMO.

3.4 Web rule languages

The current semantic web standards cover defining vocabularies and ontologies by using Resource Description Framework Schema (RDFS) and web Ontology Language. However, there is no standard for defining and sharing rules on the semantic web. The most important initiative is called RIF (Ginsberg, 2006), which defines a set of requirements and use cases for sharing rules on the web. It is important to point out, that the purpose of this language is to serve as an intermediary language between various rule languages, but it should not provide a formally defined semantic foundation for reasoning on the web such as OWL for ontologies. Here we name two most prominent rule languages and briefly describe their characteristics.

RuleML is a markup language for publishing and sharing rule bases on the world wide web (Hirtle et al., 2006). RuleML builds a hierarchy of rule sublanguages upon XML, RDF, XSLT, and OWL. The current RuleML hierarchy consists of derivation (e.g., SWRL, FOL) and production rules (e.g., Jess). RuleML is based on Datalog. RuleML rules are defined in the form of an implication between an antecedent and consequent, with the meaning whenever the logical expression in the antecedent holds, then the consequent must also hold. However, an important constraint of RuleML is that it can not fully represent all the constructs of various languages such as OCL or SWRL.

The *Semantic Web Rule Language (SWRL*) is a proposed rule language based on the W3C web Ontology Language OWL (Horrocks et al., 2004). Similar to RuleML rules, a SWRL rule is also in the form of an implication and is considered another type of an axiom on top of the other OWL axiom types. This means that SWRL rules are usually used for defining integrity constraints similar to OCL in UML. Both consequent and antecedent are collections (i.e., conjunctions) of atoms. We should say that the purpose of SWRL is not to be a universal rule syntax for interchanging rules, as it can not represent many linguistic constructs of other rule languages (e.g., F-Logic, Rei, or OCL) utilised in SWS descriptions.

4 Our approach

We propose using one general rule representation language with the syntactical capacity to represent various rule constructs mentioned in the previous section. More specifically, we propose using the R2ML language (Horrocks et al., 2004). R2ML is a general purpose rule interchange language that possesses expressivity for representing four types of rules, namely, integrity, derivation, reaction and production rules. Besides rules, R2ML has its own set of constructs for representing vocabularies and domain ontologies similar to UML or OWL. Having in mind such an expressivity, we can use R2ML to represent the following artefacts related to SWS discussed in the previous section:

- R2ML reaction rules can be used to model SWS descriptions such as WSDL-S, WSMO and OWL-S
- R2ML integrity and derivation rules can be used to represent trust management policy rules such as the ones defined in Rei, PeerTrust, PROTUNE and KAoS

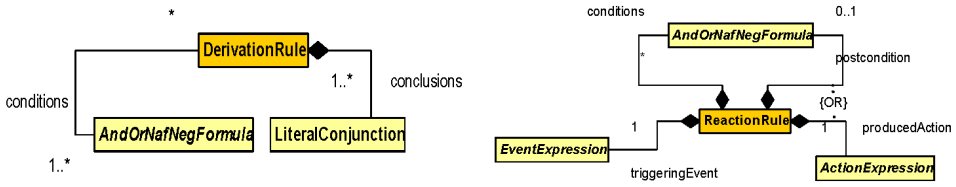
- R2ML vocabularies that are used for annotating types of SWS can be used for representing domain vocabularies expressed as UML class diagrams or OWL ontologies
- R2ML itself can be used to represent other rule languages used in Semantic web applications such as RuleML, SWRL or OCL.

Along with R2ML, a set of bi-directional transformations between R2ML and all the languages discussed in Section 4 need to be developed for our proposed solution. Given such as a set of transformations, we can transform SWRL rules (via R2ML) into OCL, or a UML vocabulary into OWL ontology (Milanović et al., 2006). In the rest of this section we give a brief overview of R2ML.

4.1 R2ML: a brief overview

R2ML is a general rule interchange language that tries to address all RIF requirements (Ginsberg, 2006). Its current version is 0.4 (R2ML specification, 2006). The abstract syntax of the R2ML language is defined with a metamodel by using OMG's Meta-Object Facility (MOF). This means that the whole language definition can be represented by using UML diagrams, as MOF uses UML's graphical notation. The full description of R2ML in the form of UML class diagrams is given in R2ML specification (2006), while more details about the language can be found in Wagner et al. (2006). In Figure 2, we give an excerpt of the metamodel that defines derivation (a) and reaction rules (b) which we will use in Sections 5 and 6.

Figure 2 The R2ML definition for: (a) derivation rules and (b) reaction rules



(a) (b)

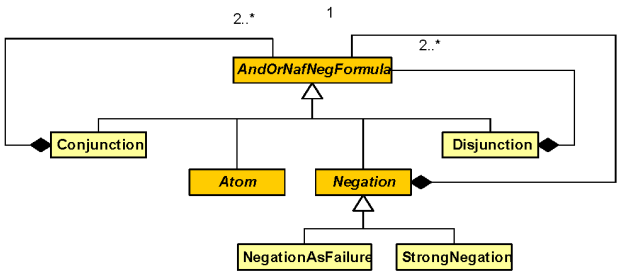
A *derivation rule* has conditions and a conclusion (see Figure 2(a)) with the ordinary meaning that the conclusion can be derived whenever the conditions hold. While the conditions of a derivation rule are instances of the *AndOrNafNegFormula* formula (Figure 3) class, representing quantifier-free logical formulas with conjunction, disjunction and negation; conclusions are restricted to quantifier-free disjunctive normal forms without NAF (Negation as Failure, i.e., weak negation). An example of an integrity rule is:

Example 1: If the requested movie belongs to the category of thriller movies then the requesting user must be older than 18.

A *reaction rule* (Figure 2(b)), also called an (ECA rule, consists of a triggering event, a list of conditions, a triggered action and an optional post-condition, which formalises the state change after the execution of the action. Here we give an example of a reaction rule:

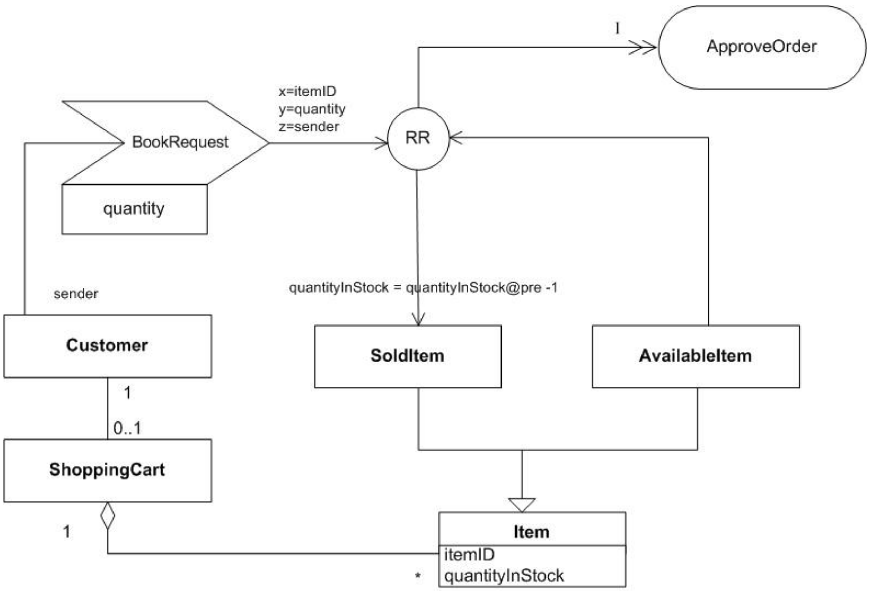
Example 2: On customer book request, if the book is available, then approve order and decrease amount of books in stock by order quantity.

Figure 3 The UML representation of R2ML's AndOrNafNegFormula



The next component of R2ML is a textual concrete syntax defined by an XML schema (so called R2ML XML). The purpose of this schema is to enable sharing R2ML rules among different applications by using XML. Another concrete syntax is a graphical UML-based Rule Language (URML) that extends the UML metamodel with rule concepts (e.g., derivation rules) from R2ML, while URML represents vocabularies by using standard UML classes and their relationships. In Figure 4, we give a URML representation for the reaction rule from Example 2. We should point out that there is a plug-in for Fujaba (a well-known UML tool), called Strelka, for modelling rules by using URML. Strelka serialises URML models in the R2ML XML format.

Figure 4 An example of a reaction rule defined in URML



Several transformations between R2ML and other languages (e.g., RuleML, F-Logic, OWL/SWRL, Jess and UML/OCL) have been implemented. These transformations have been implemented with XSLT or the ATLAS Transformation Language (ATL)

(http://oxygen.informatik.tu-cottbus.de/rewerse-i1/?q=node/15). In the rest of the paper, we show how this set of transformations can be extended by providing transformations between R2ML and SWS description languages and R2ML and trust policy management languages.

5 Mapping R2ML and Semantic Web Service (SWS) descriptions

In this section, we describe how R2ML reaction rules are mapped to SWS descriptions, logical formulas used in their descriptions, and ontology vocabularies referred to from such descriptions. The approach is demonstrated on WSDL-S, while differences with OWL-S and WSMO are discussed later in the section. In a nutshell, WSDL-S (Akkiraju et al., 2005) has the following features:

- it is defined as an extension of the standard WSDL with the goal to be fully compatible with standard web service specifications and tools
- it enables annotating datatypes using domain ontologies through the modelReference and schemaMapping attributes, but still data types are defined by using XML Schema
- precondition XML element defines a set of assertions that must be met before a web service operation can be invoked
- effect XML element can make statements about what changes in the state are expected to occur upon invocation of the service
- it is fully agnostic about the language used for defining domain ontologies, referred to in modelReference (e.g., UML, ODM and OWL) and rules used in precondition and effect elements (e.g., OCL, SWRL, RuleML, R2ML).

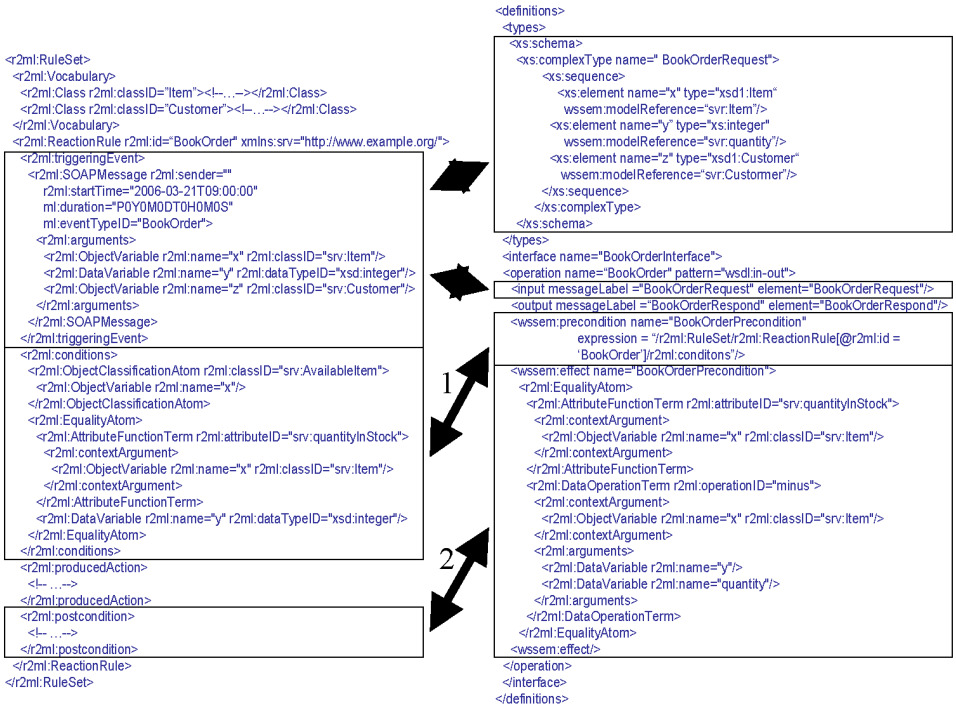
As can be seen from the listed features of WSDL-S, this language has many concepts similar to the concepts of reaction rules described above. A WSDL interface operation corresponds to an R2ML reaction rule in the following way:

- a WSDL input element is mapped to an incoming message that represents the triggering event of an R2ML reaction rule
- a WSDL output element is modelled as an outgoing message representing the triggered action of an R2ML reaction rule
- R2ML reaction rule conditions and post-conditions are mapped to the WSDL-S preconditions and WSDL-S effects, respectively
- a WSDL outfault element is mapped to an alternative action (in the form of an outgoing message) bound to a corresponding error condition in a reaction rule of the form ON-IF-THEN-ELSE.

Since WSDL-S does not have any presumption about rule language that will be used in its preconditions and effects, we can transform R2ML conditions and postconditions onto WSDL-S preconditions and effects, respectively, to be expressed in any rule language for which there are transformations implemented with R2ML (e.g., RuleML, SWRL, OCL and F-Logic). The same is applied to the opposite direction of transformations

(i.e., from WSDL-S to R2ML). Our recommendation is that WSDL-S preconditions and effects should be represented in R2ML and then, based on the need in a specific situation, translated to a target rule language. However, there is one constraint in WSDL-S caused by the XML schema definition of WSDL-S about precondition and effect XML elements. Both these elements have the expression attribute for defining logical conditions, which can only be set in the form of a text-based syntax (e.g., OCL or F-Logic). This means that we can not used an XML format (e.g., SWRL and R2ML) that will be wrapped by, for example, the effect XML element of WSDL-S. So, the R2ML conditions can only be addressed by using XPath referring to specific parts of rules defined in R2ML XML documents (arrow 1 in Figure 5). Another approach is to change the definition of the WSDL-S precondition and effect XML elements (having in mind that it is still only a W3C member submission), so that they can also wrap in an XML literal like it is shown with arrow 2 in Figure 5.

Figure 5 Mapping between (a) R2ML and (b) WSDL-S represented in XML format



(a) (b)

We can also apply similar mappings between R2ML and OWL-S. Definition of OWL-S preconditions and effects can be both literal and XML literals, which is fully compatible with two alternatives discussed for WSDL-S (arrows 1 and 2 in Figure 5). Furthermore, vocabularies can be defined in different ontology languages, thus R2MlL transformation can increase the interoperability level. However, OWL-S contains the Process ontology that is used for modelling atomic and composite (e.g., sequence, split, split + join and any-order) processes. These different types of process execution are represented in R2ML reaction rules by using EventExpressions – see Figure 2 (e.g., SequenceEventExpression

or ChoiceEventExpression). A similar situation is with mappings between WSMO and R2ML. R2ML vocabulary elements fully capture WSMO ontologies, while WSMO rules that are based on F-Logic are also fully captured by R2ML rules. Actually, there is already a transformation between R2ML and F-Logic (http://oxygen.informatik.tu-cottbus.de/rewerse-i1/?q=node/15). However, when mapping WSMO service descriptions to R2ML, there is one difference compared to mappings between R2ML and WSDL-S and between R2ML and OWL-S. It is related to output messages where WSMO has two properties, effect and post-condition. Post-conditions are related to the data output, whereas effects are used to define general state changes caused by the execution of the output. We can say that R2ML, OWL-S, and WSDL-S do not distinguish between these two different types of output conditions and both of them are captured with OWL-S and WSDL-S effect and R2ML post condition. However, to provide automatic transformations between R2ML and WSMO, R2ML should be slightly be extended to distinguish between data only and general state change conditions.

6 Policy languages and R2ML

In the previous section we showed that SWS description languages can be modelled using reaction rules. The last step in providing a powerful interchange format between different policy-aware SW services is to map policy rules from a source policy language to R2ML and from R2ML to the target policy language.

Policies for SWS are generally divided into privacy policies, in which the global constraints over accessing a service are defined, and authentication policies in which certain privileges are given to each user in accordance to the credentials he/she provides. Privacy policies can be perfectly modelled as integrity rules in R2ML. Each integrity rule is a sequence of logical predicates which, through the resolution process, can eventually hold a value of either true or false. This logical value can be neatly evaluated as a privacy policy to determine whether or not the process of information exchange can happen after resolving the constraints of the integrity rule. Authentication policies on the other hand can be efficiently modelled as derivation rules. Providing a set of credentials by a peer may satisfy the sequence of constraints in the condition part of a derivation rule and consequently a series of privileges are given to the peer.

To give the reader a clear understanding of how the policy rules can be exchanged using R2ML, we provide some mappings from policy languages to R2ML and vice versa. However, due to the length limitations in the paper, we limit our samples to authentication policies which are more general than privacy policies in the sense of dealing with conditions and consequences. We choose Rei and PeerTrust to define policies, we consider the same scenarios mentioned in Kagal et al. (2004) and Olmedilla et al. (2004), then we convert them to identical R2ML definitions and discuss the similarities of the results.

The first scenario based on Olmedilla et al. (2004) defines an authentication policy in which a discount is given to a buyer only if he/she proves that he/she is a student in a valid university (Figure 6). As it is obvious in Figure 6 the user should prove that he/she belongs to a valid university and the university must certify the validity of the student ID the user provides with regards to her or his name and information. It is worth mentioning that the object following '$' in PeerTrust illustrates the requester to whom the results will be returned and the object following '@' is the entity responsible for certifying

the expression that appears before '@'. For example the last constraint in Figure 6 considers University as the certifying authority for the expression studentId(Buyer). The expressivity of R2ML enables us to define the above rule in several different ways; however, we finally chose the transformation in Figure 7 as the most suitable one.

Figure 6 An authentication policy defined in PeerTrust

```
discount(BookTitle) $ Buyer ←
     studentId(Buyer) @ University @ Buyer,
     validUniversity(University),
     studentId(Buyer) @ University.
```

Figure 7 R2ML representation of the authentication policy of Figure 6

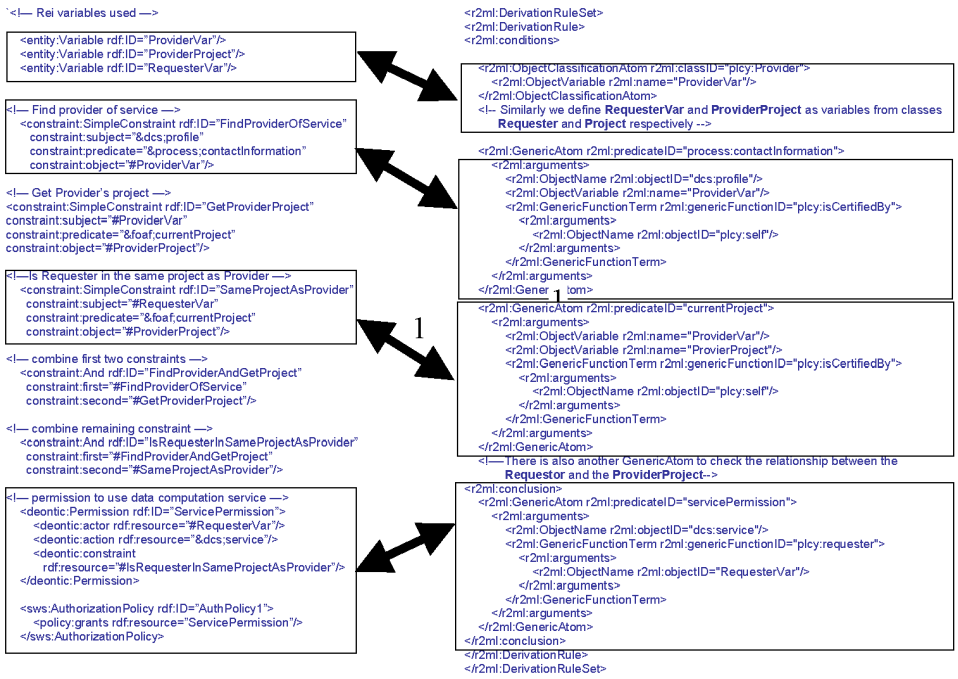
```
<r2ml:DerivationRuleSet>
<r2ml:DerivationRule xmlns:plcy="http://www.services.org/Policy/">

<r2ml:conditions>
   <r2ml:ObjectClassificationAtom r2ml:classID="plcy:Buyer">
        <r2ml:ObjectVariable r2ml:name="buyer"/>
   </r2ml:ObjectClassificationAtom>
<!-- Similarly we define university and book as variables from classes University and Book
respectively -->

   <r2ml:GenericAtom r2ml:predicateID="plcy:Buyer:studentID">
        <r2ml:arguments>
            <r2ml:ObjectVariable r2ml:name="buyer"/>
            <r2ml:GenericFunctionTerm r2ml:genericFunctionID="plcy:isCertifiedBy"
             r2ml:typeCategory="order">
                <r2ml:arguments>
                    <r2ml:ObjectVariable r2ml:name="buyer"/>
                    <r2ml:ObjectVariable r2ml:name="university"/>
                </r2ml:arguments>
            </r2ml:GenericFunctionTerm>
        </r2ml:arguments>
   </r2ml:GenericAtom>

   <r2ml:GenericAtom r2ml:predicateID="plcy:validUniversity">
       <r2ml:arguments>
            <r2ml:ObjectVariable r2ml:name="university"/>
            <r2ml:GenericFunctionTerm r2ml:genericFunctionID="plcy:isCertifiedBy">
               <r2ml:arguments>
                 <r2ml:ObjectName r2ml:objectID="plcy:self" />
               </r2ml:arguments>
            </r2ml:GenericFunctionTerm>
       </r2ml:arguments>
   </r2ml:GenericAtom>

<!—The last predicate is similar to the others -->

<r2ml:conclusion>
    <r2ml:GenericAtom r2ml:predicateID="plcy:discount">
        <r2ml:arguments>
            <r2ml:ObjectVariable r2ml:name="book"/>
            <r2ml:GenericFunctionTerm r2ml:genericFunctionID="plcy:requester">
                <r2ml:arguments>
                    <r2ml:ObjectVariable r2ml:name="buyer"/>
                </r2ml:arguments>
            </r2ml:GenericFunctionTerm>
        </r2ml:arguments>
    </r2ml:GenericAtom>
</r2ml:conclusion>

</r2ml:DerivationRule>
</r2ml:DerivationRuleSet>\
```

In Figure 7, we have mapped each predicate in PeerTrust to a GenericAtom in R2ML with the variables of each predicate defined as an ObjectVariable in R2ML. Certification

authorities of a predicate in PeerTrust are also modelled as GenericFunctionTerms in which the FunctionID refers to the role of the authority and an ObjectVariable refers to the certifying authority itself. To increase the expressivity and flexibility of the mapping we have explicitly defined a certification authority for the second constraint of Figure 6 as 'self' which is implicit in the original model defined in PeerTrust. 'self' in PeerTrust, and correspondingly in our mapping, refers to the entity that is defining the policy. Digging into the sample provided in Figure 6 and our conventions in transforming the concepts, the reader may find it easy to transfer a similar concept from PeerTrust to R2ML and back.

Now that a transformation from a rule in PeerTrust to R2ML has been provided, we follow a completely different scenario mentioned in Kagal et al. (2004) for a policy defined in Rei. Figure 8(a) shows the sample scenario in Rei.

Figure 8 Mapping between a (a) Rei policy and (b) its counterpart in R2ML



In this example, permission in using a service will be granted to the requesting entity in case the requester is working on the same project as the provider. Rei is built upon the concepts of OWL and RDF. So, the relationships are binary as opposed to PeerTrust which can have predicates of the form ternary or higher. Moreover, Rei does not address the certifying authorities as explicitly as PeerTrust and the authorities are dealt with again through defining the binary relationships between the constraints and the entities involved. Figure 8(b) is showing the transformation from the sample in Figure 8(a) to R2ML.

Finally, it should be noticed that there are a variety of possible mappings from each policy language to R2ML. For example, a SimpleConstraint in Rei (as it has been marked

with 1 in Figure 8) can easily and expressively be mapped to a ReferencePropertyAtom in R2ML which is likely even more expressive than the format that we have already come up with (Figure 9). However, transforming the derived R2ML definition to another language, say PeerTrust, is not easy as there is no room for the concept of certification authority in a ReferencePropertyAtom.

Figure 9 A Simple constraint in Rei represented as a *ReferencePropertyAtom* in R2ML

```
<r2ml:ReferencePropertyAtom referencePropertyID="currentProject">
    <subject>
        <r2ml:ObjectVariable r2ml:name="RequesterVar"/>
    </subject>
    <object>
        <r2ml:ObjectVariable r2ml:name="ProviderProject"/>
    </object>
</r2ml:ReferencePropertyAtom>
```

7 Preliminary evaluations

As we have explained in Section 5, we use reaction rules to define the mappings between WSDL and R2ML. The input or in-fault constructs for a WSDL document can be considered equal to triggeringEvent in R2ML's reaction rule, the output or out-fault construct of a WSDL document can be considered equivalent to a producedAction in R2ML's reaction rule and etc. Exploiting URML as a modelling language to define R2ML rules, we could use its modelling notation to generate R2ML reaction rules for a specific business process and then to convert it to WSDL in order to introduce a service (Lukichev et al., 2007). As an example we considered the business rule that we defined in the second example of Section 4, which states: "*On customer book request, if the book is available, then approve order and decrease amount of books in stock by order quantity*."

Again as we mentioned in Section 4, through URML notations we modelled it as shown in Figure 4, which made it a lot more comprehendible. The obtained model was then transformed to R2ML XML and then, having the ATL transformation engine and its appropriate ATL transformations (http://oxygen.informatik.tu-cottbus.de/rewerse-i1/?q=node/15), to WSDL XML. The obtained document proved to be fully compatible with the definition of the WSDL and XML Schema languages and could be deployed on any given web server to represent the above business rule as a service. Figure 10 shows an excerpt of the obtained WSDL XML code after applying the ATL transformations to the model of Figures 4 and 11 shows a part of its WSDL XML type declaration. Note also that there is another ATL transformation that transforms WSDL definitions of web services to the R2ML reaction rules, and thus we can do reverse engineering of web services. Our future goals are to extend this transformation to support WSDL-S (or Semantic Annotations for WSDL and XML Schema – SAWSDL) and other SWSL (e.g., OWL-S), so that we can fully experiment with the proposed approach for modelling SWS.

Figure 10 A part of the WSDL XML for the business rule of Figure 4 obtained through ATL transformations

```
<wsdl:interface name="BookAvailabilityService">
      <wsdl:operation name="AvailableItem"
        pattern="http://www.w3.org/2006/01/wsdl/in-out">
          <wsdl:input element="BookRequest"/>
          <wsdl:output element="ApproveOrder"/>
      </wsdl:operation>
</wsdl:interface>
```

Figure 11 Part of the WSDL XML element declaration from the type section

```
<wsdl:type>
   <xs:element name="Item">
         <xs:complexType>
               <xs:sequence>
                     <xs:element name="ItemID"
                      type="xs:ID"/>
                     <xs:element name="quantityInStock"
                      type="xs:nonNegativeInteger"/>
               </xs:sequence>
         </xs:complexType>
   </xs:element>
</wsdl:type>
```

Further to transforming the WSDL documents, the most important feature of R2ML, as a rule interchange language, is the possibility to bridge between different rule languages. In the previous sections, we have shown how different policy languages (i.e., PeerTrust and Rei) can be transformed to R2ML, thanks to its rich set of rules, atoms and terms. Having the transformations between R2ML and rule languages such as F-Logic implemented, we applied these transformations to the R2ML rules obtained from the policy languages. Our preliminary evaluations showed that the generated F-Logic or Jess rule from the R2ML rules, obtained by converting the policy rules, was valid to be reasoned over using the reasoners for F-Logic and Jess. Figure 12 shows the F-Logic rule for the Rei rule of Figure 8, obtained through applying our automatic transformations from R2ML to F-Logic. This approach opens the doors to enabling different parties with different rule languages to share and communicate their policies regardless of the policy language they use, and thus will provide a solution to the problems that we mentioned in Section 2. Another important issue that we plan to explore is how to reason over both policy and service rules in the same rule engine (e.g., Jess), as this can be a promising way to support policy drive compositions of SWS.

Figure 12 The F-logic representation for the Rei rule of Figure 8

```
<flogic>
   <!-- -->
   <pre>

  FORALL ?RequesterVar,?ProviderProject,?ProviderVar,?dcs:service :
      [hasActor->?RequesterVar; hasAction->?dcs:service]
      <-
      ?RequesterVar[foaf#currentProject->?ProviderProject]AND
      ?ProviderVar[foaf#currentProject->?ProviderProject] AND
      [process#contactInformation->?ProviderVar].

</pre>
</flogic>
```

8 Discussion and conclusion

In this paper, we have introduced an intermediary framework capable of providing transformations between different business rule and policy languages. We have shown how R2ML, a comprehensive rule markup language covering the concepts of declarative and descriptive logics, can be used to capture the notions represented in dissimilar policy and rule languages. Our proposed solution shows how various SWS description languages (e.g., OWL-S and WSDL-S) and Policy languages (e.g., PeerTrust and Rei) can be transformed to R2ML representations with similar structures. We have also demonstrated how the identical representation of these languages in R2ML can help to transform these representations to other SWS description or policy languages. This may lead to providing an interchange format that can make the inter-enterprise or inter-agent communications possible.

Referring to the example of Section 2 and assuming that our proposed framework has been placed in a Universal Description, Discovery and Integration (UDDI) registry, we can see that the broker agent, looking for a music or movie, will be no longer concerned about the language used by the service provider to describe the services and the policies. The policies and the descriptions can be transformed to the languages understandable by the broker agent through using R2ML. The agent will then use the information on how the service provider refers to the directors or actors of a movie and also the conditions required to access the desired category. Similarly, the conditions and requirements of the requesting client will be translated and presented to the service provider. Upon an agreement on the requirements and conditions of both sides of the communication, the access to the desired categories will be granted and the broker agent can use the web service of the service provider to search its collection. The broker agent will be able to communicate to all service providers that are using languages for which there are transformations to/from R2ML.

There are several transformations from R2ML to other business rule and policy languages already defined, including, OCL, OWL, F-Logic, RuleML, JBoss Rules, Jess and SWRL. The existence of a transformation engine from R2ML to F-Logic makes it possible to transform the R2ML representations for various policy languages to F-Logic and to integrate it with WSMO. Consequently, the problem of combining WSMO and PeerTrust mentioned in Olmedilla et al. (2004) is solvable by using our approach.

By using Strelka as a graphical tool to define R2ML rules through UML diagrams, a wide range of users, even those with no background on policy languages or SWS description languages, will be able to annotate their web services and define the policies over the web services. The obtained R2ML rules then can be transformed to any policy or SWS description language based on the needs and requirements.

Despite all the positive points listed above, this paper should be considered a starting point for the whole framework that we are developing. This paper is mostly a conceptualisation of the possibilities and most of the proposed ideas are targets of the future work. First and foremost, the amount of information loss during transformation from one language into R2ML and then into another language should be investigated more deeply. Although we presume that R2ML is capable of defining a wide variety of the concepts and although it is open to expansion, we still need to investigate which parts of the language need to expand. Our ultimate goal is to design a framework capable of conforming to the concepts in all business rule languages. This framework,

when fully implemented, will provide an easy way to convert all rule languages to R2ML and then reason over their concepts.

Acknowledgments

The research of Simon Fraser University is supported in part by Canada's NSERC-funded Research Network, while the research of Brandenburg University of Technology at Cottbus is supported in part by the EU IST-funded REWERSE Network of Excellence. The authors would like to thank Adrian Giurca and Sergey Lukichev of Brandenburg University of Technology at Cottbus for their valuable comments on the ideas presented in the paper.

References

Akkiraju, R., Farrell, J., Miller, J., Nagarajan, M., Schmidt, M., Sheth, A. and Verma, K. (2005) *WSDL-S Web Services Semantics – WSDL-S*, W3C Member Submission, www.w3.org/Submission/WSDL-S/.

Battle, S., Brenstein, A., Boley, H., Grosof, B., Gruninger, M., Hull, R., Kifer, M., Martin, D., McIIraith, S., McGuinness, D., Su, J. and Tabet, S. (2005) *SWSL, Semantic Web Service Language*, W3C Member Submission, www.w3.org/Submission/SWSF-SWSL/.

Becker, M.Y. and Sewell, P. (2004) 'Cassandra: flexible trust management, applied to electronic health records', *Proc. 17th IEEE Workshop on Computer Security Foundations*, pp.139–154.

Bonatti, P. and Olmedilla, D. (2005) 'Driving and monitoring provisional trust negotiation with metapolicies', *Proc. 6th IEEE Int'l Workshop. on Policies For Dist. Sys. and Networks*, Washington DC, pp.14–23.

Bradshaw, J.M., Dutfield, S., Benoit, P. and Woolley, J.D. (1997) 'KAoS: toward an industrial-strength open agent architecture', *Software Agents*, pp.375–418, http://portal.acm.org/citation.cfm?id=267985&coll=&dl=&CFID=15151515&CFTOKEN=6184618.

Burstein, M., Bussler, C., Zaremba, M., Finin, T., Huhns, M.N., Paolucci, M., Sheth, A.P. and Williams, S. (2005) 'A semantic web services architecture', *IEEE Internet Computing*, Vol. 9, No. 5, pp.72–81.

de Brujin, J., Bussler, C., Domingue, J., Fensel, D., Hepp, M., Keller, U., Kifer, M., Konig-Ries, B., Kopecky, J., Lara, R., Lausen, H., Oran, E., Polleres, A., Roman, D., Scicluna, J. and Stollberg, M. (2005) *WSMO Web Service Modelling Ontology (WSMO)*, W3C Member Submission, www.w3.org/Submission/WSMO/.

Ginsberg, A. (2006) *RIF use Cases and Requirements*, W3C Working Draft, http://www.w3.org/TR/rif-ucr/.

Grønmo, R., Jaeger, M.C. and Hoff, H. (2005) 'Transformations between UML and OWL-S', *Proc. 1st European Conf. on Model Driven Architecture: Foundations and Applications*, Nuremberg, Germany, pp.269–283.

Hirtle, D., Boley, H., Grosof, B., Kifer, M., Sintek, M., Tabet, S. and Wagner, G. (2006) *Schema Spec. RuleML 0.91*, http://www.ruleml.org/spec/.

Horrocks, I., Patel-Schneider, P.F., Boldey, H., Tabet, S., Grosof, B. and Dean, M. (2004) *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*, W3C Member Sub., http://www.w3.org/Submission/SWRL/.

Kagal, L., Finin, T., Joshi, A. and Greenspan, S. (2006) 'Security and privacy challenges in open and dynamic environments', *IEEE Computer*, Vol. 39, No. 6, pp.89–91.

Kagal, L., Paolucci, M., Srinivasan, N., Denker, G., Finin, T. and Sycara, K. (2004) *Authorization and Privacy for Semantic Web Services, AAAI 2004 Spring Symposium on Semantic Web Services*, Stanford University.

Lausen, H., Ding, Y., Stollberg, M., Fensel, D., Hern{\'a}ndez, R.L. and Han, S. (2005) 'Semantic web portals: state-of-the-art survey', *J. Know. Man.*, Vol. 9, No. 4, pp.40–49.

Lukichev, S., Guirca, A., Wagner, G., Gasevic, D. and Ribaric, M. (2007) 'Using UML-based rules for web services modeling', *2nd International Workshop on Service Engineering at the 23rd International Conference on Data Engineering (ICDE 2007)*, Istanbul, Turkey.

Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIIraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N. and Sycara, K. (2004) *OWL-S: Semantic Markup for Web Services*, W3C Member Submission, http://www.w3.org/Submission/OWL-S/.

Milanovic, M., Gasevic, D., Giurca, A., Wagner, G., and Devedzic, V. (2006) 'On Interchanging between OWL/SWRL and UML/OCL', *Proc. 6th Workshop on OCL for (Meta-)Models in Multiple Application Domains (OCLApps)*, Genoa, Italy, pp.81–95.

Olmedilla, D., Lara, R., Polleres, A. and Lausen, H. (2004) 'Trust negotiation for semantic web services', *Proc. of the 1st Int'l Workshop on Semantic Web Services and Web Process Composition*, San Diego, CA, USA, pp.81–95.

Payne, T. and Lassila, O. (2004) 'Guest editors' introduction: semantic web services', *IEEE Intelligent Systems*, Vol. 19, No. 4, pp.14, 15.

Sheth, A., Verma, K. and Gomadam, K. (2006) 'Semantics to energize the full services spectrum', *Comm. ACM*, Vol. 49, No. 7, pp.55–61.

Uszok, A., Bradshaw, J., Jeffers, R., Suri, N., Hayes, P., Breedy, M., Bunch, L., Johnson, M., Kulkarni, S. and Lott, J. (2003) 'KAoS policy and domain services: toward a description-logic approach to policy representation, deconfliction, and enforcement', *Proc. 4th IEEE I'l Workshop on Policies for Distributed Systems and Networks*, pp.93–96.

Wagner, G. (2002) 'How to design a general rule markup language?', *Proc. Workshop on XML Tech. fur das Semantic Web*, Berlin, Germany, pp.19–37.

Wagner, G., Giurca, A. and Lukichev, S. (2006) 'A usable interchange format for rich syntax rules integrating OCL, RuleML and SWRL', *Proc. Workshop Reasoning on the Web (RoW2006)*, Edinburgh, UK.

Websites

R2ML Specification, http://oxygen.informatik.tu-cottbus.de/R2ML/.

R2ML Translators, http://oxygen.informatik.tu-cottbus.de/rewerse-i1/?q=node/15.