

Rule Modeling to Unify Policies and Processes in Service-Oriented Health Information Systems

Nima Kaviani¹, Dragan Gasevic², Moeen Karimifar¹, Marek Hatala¹, Shahin Sheidaei¹

¹Simon Fraser University, Canada

²Athabasca University, Canada

{nkaviani, mooenk, mhatala, ssheidaei}@sfu.ca, dgasevic@acm.org

Abstract. The service-oriented architecture (SOA) approach offers many promises for the development of high-quality Health Information Systems (HISs) that utilize information such as Electronic Health Records (EHRs) from different parties. However, extreme privacy preservation requirements and the variety of deployment systems make development of HISs particularly challenging. To support HIS developers, we propose effective mechanisms and tools to become part of the development process. We use model-driven engineering to switch the focus of the developers from technologies used for implementing SOA to domain specific concepts of the HISs under study. Our approach relies on a modeling framework based on Web rules, which allows us to represent business processes and policies of service-oriented HISs in a unified rule framework. The framework supports development of tools for a formal analysis of existing services and their policies by using rule-based reasoning engines (e.g., HP's Jena). The tools that we are developing can identify inconsistencies between Web services from different parties, propose solutions for mismatched policies, and verify security and privacy requirements of service-oriented HISs.

1 Introduction

The recent research in the area of Health Information Systems (HIS) is realizing the necessity for developing software systems that can help with faster, less erroneous, and more productive information exchange (e.g., exchanging Electronic Health Records - EHR) and collaboration between different Health Care Establishments (HCE). This provides more effective and efficient patient care, especially in emergency cases, where a quick exchange of EHR is needed [11]. Service Oriented Architecture (SOA) introduces a sound method for developing service-oriented HIS with regards to interoperability and reliability [10]. Benefiting from the interoperability solutions of SOA, one can address the business requirements for the entities in an HIS through orchestration of their business processes (e.g., the emergency unit can fetch the latest status of a patient's EHR from her family doctor to better treat the patient in need).

What distinguishes an HIS from systems with similar interoperability requirements is its extreme constraints for privacy preservation of the patients' EHRs [12]. Making proper decisions on where, when, and to what extent, the EHR of a patient should be released to the organizations in charge has recently been subject to a lot of debates. Health Insurance Portability and Accountability Act (HIPAA) [16] in the United States and Personal Health Information Protection Act (PHIPA) [1] in Canada are examples of the efforts conducted by the governments to assure the patients with

the confidentiality of their EHRs. These concerns have initiated several research lines in merging security and privacy approaches with SOA-based HIS [18, 40].

Designing a secure and reliable SOA-based system which satisfies the interoperability requirements of an HIS is a matter of dealing with a variety of technologies at a low level of abstraction. This includes technologies to design business process relations for the Web services offered by different HCEs (e.g., BPMN [39] and WebML [7]), to declare information exchange between those services (e.g., by using WSDL [8]), to specify privacy and confidentiality rules over the EHRs (e.g., by using WS-Policy [37], Rei [21], or XACML[30]), and to dictate the policies over the services in act (e.g., by using reasoning and enforcement engines such as Margrave [13]). The problem of orchestrating Web services becomes even more challenging when we realize that each HCE may use a different set of technologies to define its business process relations and policies. Ideally, the design of SOAs for HISs should also allow for dynamic discovery of services and even run-time orchestration of services. In such situations, it is important to have (run-time) mechanisms for checking whether it is possible to orchestrate some services or not (e.g., based on the different policies).

To address the problem of using many different technologies in the development of service-oriented HIS, we propose the use of model-driven engineering (MDE). MDE enables for abstracting away the complexity of the underlying architectures and focusing on the problem under study [4]. This is done by defining and using a domain-specific modeling language that can precisely model different aspects of service-oriented HISs, i.e., business processes and their corresponding vocabularies and policies. However, to be able to undertake both design- and run-time analyses of policy-driven SOAs for HIS, the modeling language must be based on a suitable formalism.

In this paper, we start from the fact that business processes, vocabularies, and policies can be modeled by means of rules. In our approach, we propose the use of Web rules [35] defined over vocabularies. In particular, we use the *UML-based Rule Modeling Language (URML)* [2] as an MDE solution to model Web rules (and consequently policy-driven SOAs for HIS). The URML stands for a graphical concrete syntax of REVERSE Rule Markup Language (R2ML) [32] which is one of the major proposals for sharing rules on the Web. We use URML as a unifying framework into which we transform business processes of involved HISs and their associated policies [23]. We use this unified rule space to develop reasoning tools supporting SOA-based HIS application designers with analysis and validation in order to resolve information exchange conflicts among services and their policies at design time as well as to monitor privacy during information exchange. We also show how to exploit these transformation capabilities to resolve the issues that frequently rise due to the diversity of the underlying technologies used by collaborating HCEs.

The rest of this paper has been organized as follows. In section 2, we provide a motivational example of the requirements of incorporating different HCEs into one HIS application. Section 3 gives a background on the existing technologies to be used for establishing an HIS together with our modeling approach. Section 4 elaborates on using URML for designing and validating a distributed HIS. In Section 5, we discuss the advantages of our approach compared to other existing techniques, which is followed by a conclusion in Section 6.

2 Motivations

It is highly desirable to enable different HCEs to share their business processes and EHRs. In the case of an emergency, obtaining accurate EHRs from family doctor's HIS can play a critical role in saving patient's life. For an elderly patient, exchanging her EHR between her family doctor's HIS and the long term care unit's HIS in the retirement house can provide for the better quality care with an ability to continually monitor the patient's health. The HIS applications developed for any of these HCEs have to be able to discover and connect to other HIS that hold the vital EHR information, including the emergency HCEs, the family doctor medical clinics, insurance companies, paramedics, or any health clinic that keeps or uses patients' records. Typically, in the distributed HIS, collaborating organizations establish their information exchange network upfront and before getting into any exchange of EHRs. They synchronize their resources and develop shared platform on which the HIS applications are developed. This process is slow, requires commitment from all parties, and makes it difficult to engage HISs in HCEs that were not part of the original agreements.

We share the vision of SOAs where HIS applications are built on top of independent HIS services provided by different HCEs. Rather than developing a shared platform between HCEs we equip the HIS application developer with the set of mechanisms and tools that facilitate the application development, including resolving interoperability issues, addressing policy constraints, and satisfying privacy requirements. In Fig. 1 we describe a rather typical scenario of an HIS application. A nurse in the emergency unit (organization C) obtains the driver's license number, name, and address of a car accident patient. Her application interrogates known insurance companies' HIS services for the knowledge of the patient. Once the service is discovered (organization A), another service is invoked at the discovered HIS to obtain Personal Identifiable Information (PII) of the patient, such as his Care Card Number. If available, also the identifier of the family doctor's clinic for the patient is discovered at the same time. Otherwise, a similar interrogation over the medical clinics is carried forward with patient's Care Card Number. Next, the discovered medical clinic HIS is

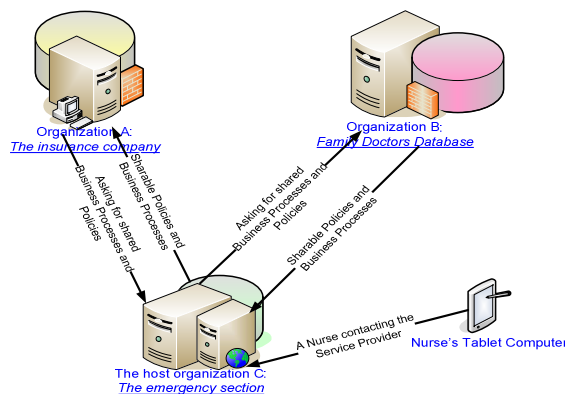


Fig. 1. A security enabled HIS to protect the confidential information of a patient while giving appropriate privileges to the involved resources

queried for the patient's EHR. After the treatment, the EHR at the family doctor's HIS is updated with the latest patient information including recommended follow-up treatments.

This example scenario highlights several challenges that have to be resolved either at HIS application development time or at runtime. For organization C to be able to access the information of organizations A and B, it needs to know what services are offered by these

two organizations, what constraints each organization defines over sharing its information, and how the services can be orchestrated to obtain intended information out of the EHRs. Moreover, as the selection of organizations A and B is dynamic based on a concrete patient identification, the HIS application has to be able to communicate with a wide variety of HIS systems. To achieve a smooth exchange and throughput of data, the interoperability mechanisms, to align the business processes in all involved organizations, have to be designed at the application development time. Additionally, each of these organizations may have its own set of privacy and security policies placed over the shared EHRs. For instance, a policy by the insurance company may prohibit any call from organization C to see any information beyond the Care Card Number of the patient. It may also open the access to the Care Card Number only for the doctors of an emergency section and not for the nurses. Similarly, the HIS keeping the EHR of the patient may consent to revealing the results for special medical tests merely to the doctors of organization C (i.e. emergency section) by obligating them to send a notifying email to the corresponding patient. It should also be noted that different HCEs may use different languages to define their EHRs, business processes, and policies (cf. Section 3) which makes collaboration among them even more difficult.

To the best of our knowledge, there has been no practical effort that puts into the same context the design and verification of the existing services, policies, and business rules, so that one can potentially propose some possible workflows or analyze the defined workflows (i.e., orchestrations.) Such a method requires a reasoning process over different business processes provided by the HCEs. In this paper, we suggest URML as a rule modeling language that can assist with modeling both business processes and policy rules as through its rich rule-based graphical notations. Strong ties between URML and REVERSE Rule Markup Language (R2ML) then enables us to convert our URML-based models to R2ML and then to any rule or policy language [23, 26], thanks to the language capabilities. Also transformations between R2ML and various rule languages (e.g. F-Logic [24] and JBoss [27]) enable us to transform the URML models to any of these languages (through R2ML) and then reason over them for the sake of validation and verification of a HIS application at design time. Furthermore, such models can be later deployed into highly dynamic SOAs. The rest of this paper will be devoted to provide a clear definition of our modeling approach for designing, verifying, and validating HIS applications in distributed HISs.

3 Background and Related Work

The pervasive use of the Internet in today's applications and its adoption by HCEs now benefits from a collection of standards set by national and international organizations for the purpose of keeping and transferring EHRs. In this section we review some of the specifications and languages developed for defining and exchanging EHRs, business processes, and policies defined over the business processes.

3.1 Medical Data Exchange Standards

Digital Imaging and Communications in Medicine (DICOM) is an internationally accepted specification for storage, manipulation, and transmission of diagnostic and

therapeutic images, and image-related information [11]. DICOM defines a file format and a network protocol for the exchange of medical images.

Health Level 7 (HL7) [16] is another standard set which defines a message model to be used in transmission of textual data in HCEs. Based on these standards, a number of medical data exchange and communication architectures have been developed, the most well-known one of which is briefly overviewed below.

Veterans Health Information Systems and Technology Architecture (VistA) is an HIS used for EHR data exchange adopted by the Veterans Health Administration (VHA), one of the biggest health organizations in the world. This system, widely used in the United States, is based on Bidirectional Health Information Exchange (BHIO), a comprehensive set of protocols for peer-to-peer transmission of large amounts of coordinated healthcare information. VistA is developed in accordance with HL7, and is believed to be the first nationwide medical information exchange system [6].

3.2 EHR Exchange Languages

Medical Markup Language (MML) is a markup language for facilitating EHR exchange between different HCEs. In order to exchange information, the information stored in the database of a specific HCE is converted into the MML format, and using the single MML interface of that HCE, it is exchanged with other HCEs [15].

Virtual Patient Records System (XVPRS) [41] is a centralized approach based on a web server from which medical information flows to all connected HCEs. A Communication Virtual Machine (CVM) is responsible for the exchange of EHRs, and can have policy and privacy requirements such as HIPPA enforced on it.

3.3 Service Languages

Web Service Description Language (WSDL) is an XML-based language that is used for information exchange in distributed systems [8]. WSDL allows those who seek the services to find out the function calls to a service and how to access them. WSDL describes web service capabilities as collections of communication endpoints that can exchange messages. It is extensible in the sense that regardless of the message formats or network protocols, WSDL fully describes the endpoints and their messages.

Business Process Execution Language (BPEL) is an XML-based execution language that is used in distributed systems for sharing tasks carried out by different entities in the system [9]. Using a combination of web services, BPEL enables intra, and inter-communication between parts of a decentralized system.

3.4 Service development languages

Business Process Modeling Notation (BPMN) is a standardized modeling language, with a graphical notation, for defining business processes in a workflow [39]. BPMN provides a mechanism to help with generating executable business processes (in BPEL) from the business level notation. This mechanism has been used in tools such as BPMN2BPEL¹.

¹ <http://www.bpm.fit.qut.edu.au/projects/babel/tools/>

Web Modeling Language (WebML) [7] is a modeling language for designing web applications, which most importantly consists of the content model (i.e. ER modes) and hypertext model (i.e. including navigation and presentation). WebML has extensions for modeling service-oriented Web applications by extending its hypertext models with concepts for modeling Web service interactions based on the WSDL 1.2 message exchange patterns. To the best of our knowledge, no practical attempt has tried to discuss policy related issues (e.g., access control) in terms of WebML.

3.5 Policy Languages

eXtensible Access Control Markup Language (XACML), is a relatively new and promising paradigm in encoded data exchange [30]. This XML-based markup language is used for expression and enforcement of access control policies using a single language. XACML can be used to restrict access to certain EHRs in an HIS, and has the capability to deploy authorization policies that comply with its resources and business use cases. XACML supports rules which perform conflict resolution amongst conflicting policies. Although XACML supports a fine granularity of access control specification, the policy is verbose and not intended for human interpretation.

Rei is a policy framework that permits to specify, analyze, and reason about declarative policies defined as norms of behavior [21]. In order to specify policies, Rei adopts a rule-based approach. The policies defined by Rei govern the actions (e.g. *access control*) that an entity (e.g. a *health professional*) can perform on resources in the environment. 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.

Knowledgeable Agent-oriented System (KAoS) is a framework that provides policy and domain management services for agents and other distributed computing platforms [34]. It has been deployed in a wide variety of multi-agent and distributed computing applications similar to HISs. KAoS policy services allow for the specification, management, conflict resolution, and enforcement of policies within agent domains. KAoS uses Web Ontology Language (OWL) for Semantic policy specification.

3.6 General Rule Markup Languages

The main purpose of a rule markup language is to permit reuse, interchange, and publication of rules that administer the behavior of entities in a system. Rule markup languages are the vehicle for using rules on the Web and other distributed systems [38]. They play an important role in facilitating business-to-customer (B2C) and business-to-business (B2B) interactions over the Internet. Here, we refer to two of the widely used markup languages that can represent policies in a high level of abstraction.

RuleML represents an initiative for creating a general rule markup language that will support different types of rules and semantics that constitute policies [5]. RuleML is built on top of logic programming paradigm of first order logic (i.e. predicate logic). It builds a hierarchy of rule sublanguages upon XML, RDF, XSLT, and OWL. The current RuleML hierarchy consists of derivation (e.g., SWRL, FOL), integrity (e.g., OCL), reaction (i.e., Even-Condition-Action), and production rules (e.g., Jess).

REVERSE Rule Markup Language (R2ML) is another rapidly emerging web rule language which serves as a means for enabling the deployment, execution, publi-

cation, and communication of rules on the Web. R2ML is defined by a Meta-Object Facility (MOF) metamodel. It considers five main types of rules to model a system, i.e., integrity, derivation, production, reaction, and transformation rules. In this paper, we explain two types of rules that are later used in Section 4. An R2ML derivation rule has conditions and a conclusion (Fig. 2a) with the ordinary meaning that the conclusion can be derived whenever the conditions hold. A reaction rule (Fig. 2b) is a statement of programming logic that specifies the execution of one or more actions in case of occurrence of a triggering event and satisfaction of its conditions. Optionally, after executing the action, post-conditions may need to be satisfied. Reaction rules therefore have an operational semantic (formalizing state changes, e.g., on the basis of a state transition system formalism). Fig. 2 shows an excerpt of the R2ML metamodel for a derivation rule and a reaction rule. The R2ML vocabulary can be defined as a combination of *Basic Content Vocabulary*, *Relational Content Vocabulary*, and *Functional Content Vocabulary* to represent objects and classes, relations between them, and also different functions.

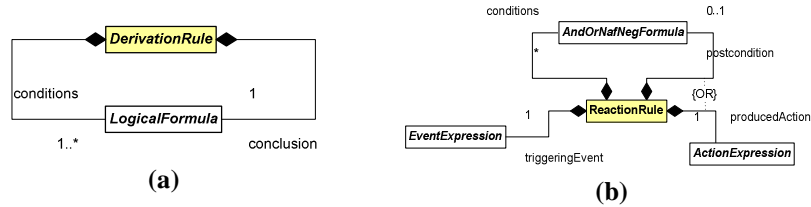


Fig. 2. The meta model for (a) a derivation rule, and (b) a reaction rule in R2ML

R2ML has both a UML-based graphical and an XML-based textual concrete syntax. Given that R2ML fully follows MDE principles, it is the best candidate to be used as a general policy modeling framework in any distributed system. The full description of R2ML in the form of UML class diagrams is given in [32], while more details about the language can be found in [38]. Several transformations between R2ML and other rule languages (e.g., RuleML, F-Logic, OWL/SWRL, Jess, UML/OCL, Drools, and Jena2) and policy languages (e.g., Rei and KAoS) have been implemented by using XSLT or the ATLAS Transformation Language (ATL) [33].

4 URML to define Medical Policies and Processes

UML-based Rule Modeling Language (URML) [2] has been designed by benefiting from the use of MOF/UML *conceptual modeling* extended with additional *visual notations* that enable *assertion specifications*. URML is backed up by R2ML and provides one-to-one mappings from its graphical notations to the concepts of R2ML. Consequently, all of the rules represented in R2ML (i.e. derivation, reaction, production, and transformation) can be graphically modeled using URML (N.B., integrity rules are modeled as OCL invariants). All types of rules in URML are depicted using circles with identifiers. Conditions are depicted as arrows from a conditioned model element to a rule circle (i.e. \rightarrow). A condition element can be one of the UML classifiers: *class*, *association*, and *association end*. A negated condition is depicted using a crossed arrow (i.e. \rightarrow). Conditions can also be defined by using OCL filters, that is,

OCL expressions that further constrain the conditions defined by means of UML classifiers. There is a plug-in for Fujaba (a well-known UML tool), called *Strelka*, for modeling rules by using URML. Strelka serializes URML models in the R2ML XML format. Fig. 5 and Fig. 6 show examples of URML rules.

Our prior research in the area of business process modeling and policy transformation, via rules, shows that derivation rules are the most suitable rule constructs to carry policies [22] while reaction rules are the most appropriate ones to model business processes [26]. Building on top of our previous research work and exploiting the use of graphical notations of URML, in this section we present the details of our hybrid modeling framework to combine business processes and policies of a SOA-based HIS in one single context. This provides us with a framework in which we can build distributed systems with upfront verification and validation of their business processes, resolution of policy conflicts, and detection of information leakage and vulnerabilities of the system under design.

4.1 URML to model Business Processes

Business processes in a system represent a set of interwoven tasks which address a managerial, operational, or supporting process. Business processes may have different behavioral patterns, that is, they may or may not accept an input message and may or may not generate an output message. In cyber space, the ultimate goal of a business process is accomplished by a set of one or more Web services. Based on the behavior of the business process, the WSDL specification defines eight different message exchange patterns (MEPs) between SOAP nodes, including, *In-Only*, *Robust In-Only*, *In-Out*, *In-Optional-Out*, *Out-Only*, *Robust Out-Only*, *Out-In*, and *Out-Optional-In*.

Considering the behavior of a Web service in terms of the triggering events, the set of input data, pre- and post-conditions for executing a Web service (strongly supported by Semantic Web service specifications [36]), and the ultimate output of the Web service, it can be neatly modeled as an Event-Condition-Action (ECA) rule or, better to say a reaction rule, in R2ML. In this case, a web service input message is a triggering event of a reaction rule, while a web service output message can be modeled as an action of an R2ML reaction rule. Although the standard WSDL specification does not support pre-conditions or post-conditions, almost all proposals for Semantic Web services include these concepts, and thus further justify the fact that R2ML reaction rules can be used to model Web services.

In URML, a reaction rule has additional elements that are specific only to this type of rules. A *message event type* model element is connected to the rule circle with the help of a triggering event arrow with a solid head². A rule event is depicted by an arrow from an event class to a rule circle. A postcondition is depicted as an outgoing arrow with a double head (to denote a state change) from the rule circle to the postcondition classifier (class, association, or association end). A postcondition arrow supports annotation with filter expressions, specifying the state change of the system.

² Note that here we show one particular type of event expressions (i.e., message event type) supported in R2ML and URML. However, as shown in Fig. 2, triggering events are of type *EventExpression*, which is further divided into composite (e.g., parallel or sequential events) and atomic event types. Messages event type is a subtype of *AtomicEventExpression*.

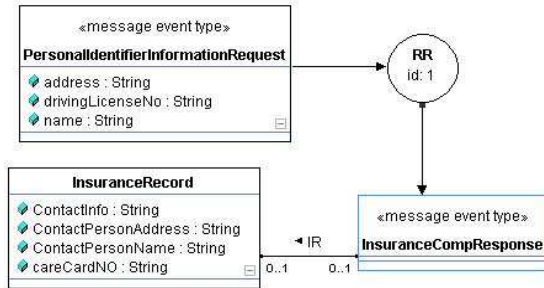


Fig. 3. The reaction rule model for an In-Out Web service example

In this case, a request of type *PersonalIdentifierInformationRequest* containing the name, the address, and the driving license number of a patient is sent to a Web service, and the Web service sends back an *InsuranceRecord* containing the Care Card Number and the information for the contact person of the patient. Next, this obtained information can be used to request the patient's EHR (e.g., expressed in a language such as MML) by querying other services. In that case, our service model will be referring to MML concepts as parts of output messages to be exchanged via SOAP. This clearly shows an *In-Out* MEP for the corresponding Web service. Considering that we have one-to-one mapping between the model of Fig. 3 and the R2ML reaction rule, the model can be fully mapped to an R2ML reaction rule. We also have developed two-way mappings between R2ML and WSDL which would easily enable us to convert the obtained R2ML rule to WSDL, and way back from WSDL to R2ML [26].

4.2 URML to model Policy Specifications

As we stated earlier, policies are best modeled through using derivation rules. This is mainly because policies clearly show the derivative meaning of granting privileges based on a set of constraints that satisfy the conditions. The policy constraints are placed in the condition part of a derivation rule and the final conclusion would be an authorization or authentication decision. The generality of R2ML enables us to convert most of the existing languages to R2ML. Moreover, we can provide transformations from R2ML to different policy languages as well. Detailed discussion of interchanging policies through using R2ML can be found in [22]. Compliance and conformance of URML to R2ML enables us to extend the mappings originally defined between policy languages and R2ML to policy languages and URML as well. Thus one can model various policy languages using URML, and then convert these URML models to R2ML. The approach gets completed by applying the transformations from R2ML to different policy languages which would eventually transform our abstract models for policies to machine understandable policy representations.

For our R2ML models to be able to fully cope with the meaning of policy languages, we have defined a high-level conceptual ontology that entails the most important and necessary elements for a policy language. We use the instances of these ontological concepts in the conclusion of an R2ML derivation rules to capture the meaning of the policy rules modeled with R2ML derivation rules. Fig. 4 shows the

To show the capability of our framework, here we demonstrate a Web service based on the example given in Section 2.

We mentioned in the scenario of Section 2 that the insurance company of a patient may need to send the Care Card Number as well as the contact person of a patient to a nurse in the emergency unit. Fig. 3 represents the reaction rule model for a Web service representing this scenario.

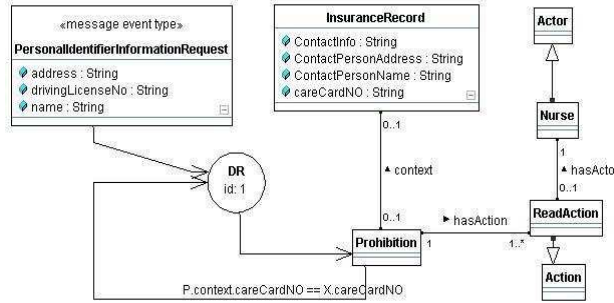


Fig. 5. The URML model of a policy prohibiting the nurse to access any information beyond what is allowed by ReadAction.

metamodel for the abstract ontology that we developed for conceptualizing policies in R2ML. We also defined this ontology using the R2ML vocabulary. As shown in Fig. 4, we consider four main types of policies for our current ontology, *Permission*, *Prohibition*, *Obligation*, and *Dispensation*. The basic attributes which our policy ontology entails consist of the to-be-protected action, the context where the action is applied, the priority for the policy, and the effect of executing the policy. Moreover, for obligation or dispensation policies, we have the additional concept *obliged* which places an obligation action upon execution of a policy rule. All these elements refer to the concept of OWLTHING for their range. However, we can further specialize them by defining specific classes for action, and context. In particular we define the *Action* class and define a *hasActor* role for this class which represents the actor of an action and has the class *Actor* for its range. This class can be further specialized in order to represent users and roles (e.g. we have *Nurse* inherited from the class *Actor* in Fig. 5).

To exemplify the capabilities of our modeling framework, let us try to apply our modeling approach to the policies of Section 2. We have mentioned that, the insurance company *A* might set a policy that *prevents an emergency section's nurse from accessing any other information beyond the Care Card No of a patient*. Leveraging the graphical notation of URML, this rule can be modeled as shown in Fig. 5. According to the figure, there is a prohibition element, taken from our R2ML policy ontology, in the conclusion of the rule with its *context*, and *hasAction* attributes related to instances from *InsuredRecord* and *ReadAction* respectively. The policy defines that, any request from a *RequestingEntity* for which the requested action is not of type *ReadAction*, i.e., the action consented over among the collaborating organizations beforehand, is prohibited. Fig. 6 shows modeling of the other scenario of Section 2, stating that *only the doctor from the emergency section is allowed to access the medical test results of a patient and a notification email is required to be sent to the patient*.

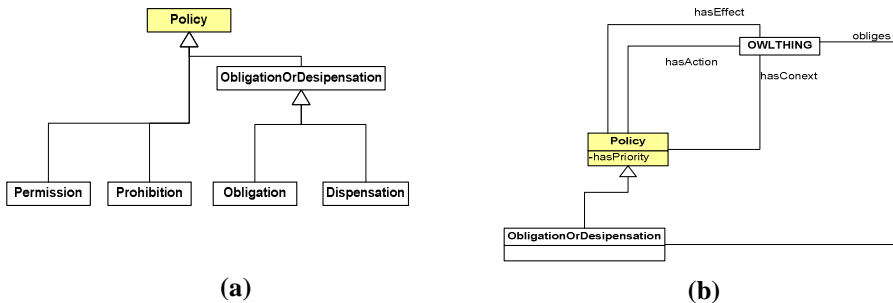


Fig. 4. The abstract policy ontology used in R2ML together with its internal definition

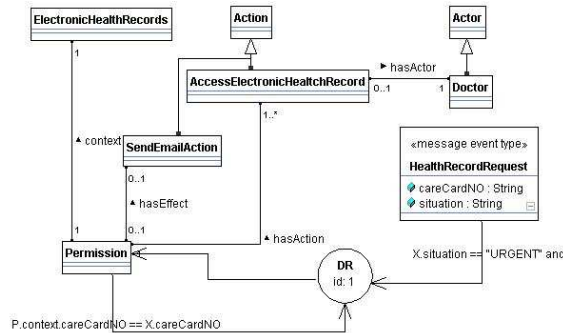


Fig. 6. The URML model of a policy allowing the doctor to access the medical record of a patient with a notification email as its effect.

As stated before, there is clear a one-to-one mapping from URML to R2ML. Consequently, we should be able to transform the models above to R2ML. Strelka supports automatic generation of R2ML rules from URML. Fig. 7 shows the R2ML derivation rule obtained from transforming the model of Fig. 6. As Fig. 7 shows, for every single element of URML we have a corresponding R2ML construct. For example, the equality check linked as a condition to the derivation rule of Fig. 6 has been mapped to the *EqualityAtom* of R2ML. Similarly,

What distinguished the policy of Fig. 6 from that of Fig. 5 is the *hasEffect* attribute of the policy that can be instantiated by instances of class *EmailAction*. This reflects the effect of permitting the doctor to access the medical test results of a patient.

As stated before, there is clear a one-to-one mapping from URML to R2ML. Consequently, we should be able to transform the models above to R2ML. Strelka supports automatic generation of R2ML rules from URML. Fig. 7 shows the R2ML derivation rule obtained from transforming the model of Fig. 6. As Fig. 7 shows, for every single element of URML we have a corresponding R2ML construct. For example, the equality check linked as a condition to the derivation rule of Fig. 6 has been mapped to the *EqualityAtom* of R2ML. Similarly,

we have implemented mappings from R2ML to KAoS and Rei policy languages. [22] demonstrates how policies expressed in the form of R2ML rules can be further transformed to policy languages such as KAoS and Rei. Besides the benefits of this approach in terms of facilitating the definition and design of policies, the actual strength appears when we combine this approach with what we provided in the previous section, that is, modeling the business processes by using URML reaction rules.

```

<r2ml:DerivationRule>
  <r2ml:conditions>
    <r2ml:ObjectClassificationAtom r2ml:classID="Request">
      <r2ml:ObjectVariable r2ml:name="X">
        <r2ml:ObjectClassificationAtom>
          <!-- similarly for other elements as well -->
        </r2ml:ObjectClassificationAtom>
      </r2ml:ObjectVariable>
    </r2ml:ObjectClassificationAtom>
    <r2ml:EqualityAtom>
      <r2ml:ReferencePropertyFunctionTerm r2ml:referencePropertyID="careCard">
        <r2ml:contextArgument>
          <r2ml:ObjectVariable r2ml:name="MTR" r2ml:classID="MedicalTestResult">
            <r2ml:contextArgument>
              <r2ml:ReferencePropertyFunctionTerm>
                <r2ml:ReferencePropertyFunctionTerm r2ml:referencePropertyID="careCard">
                  <r2ml:contextArgument>
                    <r2ml:ObjectVariable r2ml:name="X" />
                    <r2ml:contextArgument>
                      <r2ml:ReferencePropertyFunctionTerm>
                        <r2ml:EqualityAtom>
                      </r2ml:ReferencePropertyFunctionTerm>
                    </r2ml:ContextArgument>
                  </r2ml:ReferencePropertyFunctionTerm>
                </r2ml:ContextArgument>
              </r2ml:ReferencePropertyFunctionTerm>
            </r2ml:ContextArgument>
          </r2ml:ObjectVariable>
        </r2ml:ContextArgument>
      </r2ml:ReferencePropertyFunctionTerm>
    </r2ml:EqualityAtom>
  </r2ml:conditions>
  <r2ml:conclusion>
    <r2ml:ObjectDescriptionAtom r2ml:classID="Permission">
      <r2ml:subject>
        <r2ml:ObjectVariable r2ml:name="P">
          <r2ml:subject>
            <r2ml:ObjectSlot r2ml:referencePropertyID="context">
              <r2ml:object>
                <r2ml:ObjectVariable r2ml:name="MTR" r2ml:classID="MedicalTestResult">
                  <r2ml:object>
                    <r2ml:ObjectSlot>
                      <r2ml:ObjectSlot r2ml:referencePropertyID="hasAction">
                        <r2ml:object>
                          <r2ml:ObjectVariable r2ml:name="AMT" r2ml:classID="AccessMedicalTest">
                            <r2ml:object>
                              <r2ml:ObjectSlot>
                                <r2ml:ObjectSlot r2ml:referencePropertyID="hasEffect">
                                  <r2ml:object>
                                    <r2ml:ObjectVariable r2ml:name="SEA" r2ml:classID="SendEmailAction">
                                      <r2ml:object>
                                        <r2ml:ObjectSlot>
                                          <r2ml:ObjectDescriptionAtom>
                                        </r2ml:ObjectSlot>
                                      </r2ml:ObjectVariable>
                                    </r2ml:ObjectSlot>
                                  </r2ml:ObjectSlot>
                                </r2ml:ObjectSlot>
                              </r2ml:ObjectSlot>
                            </r2ml:ObjectSlot>
                          </r2ml:ObjectSlot>
                        </r2ml:ObjectSlot>
                      </r2ml:ObjectSlot>
                    </r2ml:ObjectSlot>
                  </r2ml:ObjectSlot>
                </r2ml:ObjectSlot>
              </r2ml:ObjectSlot>
            </r2ml:ObjectSlot>
          </r2ml:ObjectSlot>
        </r2ml:ObjectVariable>
      </r2ml:Subject>
    </r2ml:ObjectDescriptionAtom>
  </r2ml:conclusion>

```

Fig. 7. The corresponding R2ML excerpt for the policy of Fig. 4

5 Discussion

The unified representation of policies and processes into the single reasoning space is essential both for the dynamic discovery and orchestration of service as well as improving development of HIS applications. By embedding intelligent reasoning into the software development process we plan to speed up the development of the HIS applications as well as provide mechanisms for achieving highly usable applications satisfying the stringiest privacy requirements. We are developing three applications currently based on our framework.

Validator for Service Orchestration. The main purpose of the Validator is to run simulations on the application invoking the services from the different domains. As the developer models the HIS application, which invokes Web services from different HISs, the application business process and policies associated with the services (possibly expressed in different policy languages) are transformed into the unified rule framework. The Validator invokes the policy rules following the application business process and executes them testing the flow through the system for the predefined combinations of input parameters and ranging over the set of runtime selection choices, e.g. registered family medical centers and their policies.

Policy Analyzer/Bottleneck Spotter. The Policy Analyzer is an add-on to the Validator process. The Analyzer is invoked when the Validator Simulation is halted because of the conflicting policies. The rules representing policies are analyzed with respect to the constraints and expected parameters. Three possible types of conflicts can be identified: i) vocabulary conflict requiring remapping of the attributes to 'conflicting' policy ontology, ii) missing information conflict requiring redesign of the business process, and iii) unresolvable policy conflict requiring developers to engage in the bilateral negotiation with the service provider and reaching agreements leading to the redefining or extending the policies.

Privacy Guardian. In the heart of the third application are specific privacy protection rules set by general privacy laws and health privacy standards. For example, the rule can specify that the insurance company can receive only patient information required to determine the level of coverage for the specific current treatment. The generic privacy rules are amended with the rules stemming from the bilateral agreements for information exchange between HCEs and internal rules for the home HCE. The Guardian application simulates the workflow and monitors the flow of the information in the service orchestration with respect to the privacy rules and constrains specified in the policies' post-conditions identifying any potential violations of the patient's privacy.

The first two applications above focus on increasing the productivity of the developers in the highly sensitive domains such as HIS but their applicability is much broader. The third one clearly focuses on HIS but is designed modularly which enables to replace HIS specific privacy rules with a set of rules from different domains. It should be also noted that the three applications benefit from the unified models that bring all involved workflows and policies into single space. From the technical point of view the applications themselves are expert systems that reason over the distributed HIS. Another benefit of the unified modeling environment for the workflows and policies is that it increases maintainability of the developed applications, in particular their extensibility. With clearly modeled extension points of the application the three

applications can be easily re-run on the extended models and as such preserve the consistency of the final application. The same approach can be used as a prophylactic procedure to ensure the application is well synchronized with external Web services.

6 Conclusions

Several research projects have addressed the use of models in an HIS-like domain which can be characterized through its extreme privacy requirements. [3, 28] represent Model Driven Security Engineering (MDSE) as a method for representing both the business process relations and policies via UML profiles and Object Constraint Language. Jones et. al. in [19] suggest the use of MDA together with formal methods for the purpose of verification, validation, and correction of the designed system. [] represents some efforts in using UML to define security based distributed systems.

Despite the important contributions made in the abovementioned efforts, none of them precisely addresses the verification and incorporation of different HCEs in a service-oriented and policy-driven HIS application. This leads to the most important contribution of our framework with the ability to model policies and process of service-oriented HIS in a unified way. The main consequence is that we can then analyze both policies based on the same formalism, that is, rules that have well-studied principles of reasoning, validation and verification. Other related model-driven approaches require transformations of their models into suitable validation formalisms [20], or when they are using several different model languages (e.g., UML diagrams or UML profiles), which is the most common case, all these different aspects should be translated into the same formal framework to be validated [31]. In our case, we are leveraging the use of R2ML to be a general interchange language, and hence we are able to analyze models of policy-driven SOAs for HIS into many different rule-based engines such as Jena. Fig. 8 shows a sample of the automatically generated Jena rules from the corresponding R2ML rules of Fig. 3 and Fig. 6.

<pre>[(?IR rdf:type InsuranceRecord) <- {?PIIR rdf:type PersonIdentifierInformationRequest}]</pre> <p style="text-align: center;">(a)</p>	<pre>[(?P context ?EHR) (?P hasAction ?AMT) (?P hasEffect ?SEA) <- (?X rdf:type Request) (?N rdf:type Nurse) (?X situation URGENT) equal(careCardNO(?EHR),careCardNO(?X)) (?AMT hasActor ?N) (?SEA hasActor ?N)]</pre> <p style="text-align: center;">(b)</p>
--	--

Fig. 8. The Jena representation of the R2ML rules shown in a) Fig. 3. and b) Fig.6.

However, given that we have developed two way transformations between R2ML and various policy and rule languages, we are actually able to analyze the existing policy and service definitions by transforming them into a rule-based representation as well (i.e. to reverse engineer the already developed systems into our modeling framework). For example, we can translate policy languages such as KAoS and Rei onto R2ML derivation rules, while service descriptions (e.g., WSDL) can be translated into R2ML reaction rules. Thus, we can reason over the existing policies and services when integrating them into new HIS application. This also offers a flexible

approach to redesigning the existing service-oriented HIS in order to reflect changes over either policies or processes, done by any of the parties involved in a service-oriented HIS. Rule-based models of policy and process can also be used directly for run-time validation of the execution of SOAs, while the overall unified rule-based approach can be used for dynamic composition of policy-driven SOAs, in the cases when there is no predefined service compositions.

Our future work will focus on the implementation of the tools discussed in Section 5, namely, Validators for Service Orchestration, Policy Analyzer/Bottleneck Spotter, and Privacy Guardian. This will include defining mappings between R2ML production and reaction rules (such as JBoss' Drools or HP's Jena), mappings between standard policy languages (i.e., XACML and WS-Policy) and R2ML in addition to the already supported policy languages (i.e., KAOs and Rei), defining mappings between service composition languages (e.g., WS-CDL and BPEL) and R2ML, and experimenting with the rule-based reasoning services for verification of policy-driven service orchestrations.

References

1. "Personal Health Information Protection Act (PHIPA)", <http://www.peelregion.ca/corpserv/hipa/index.htm>.
2. "URML: a UML-Based Rule Modeling Language." [Online] Available: <http://oxygen.informatik.tu-cottbus.de/reverse-1/?q=node/7>
3. Alam, M., et. al, "Modeling Permissions in a (U/X) ML World," *Availability, Reliability and Security, 2006. The First Int'l Conference on*, pp. 8, 2006.
4. Bézivin, J., "On the unification power of models," *Software and Sys. Modeling*, vol. 4, no. 2, pp. 171–188, 2005.
5. Boley, H., Tabet, S., Wagner, G. "Design Rationale of RuleML: A Markup Language for Semantic Web Rules." *In Proceedings of Semantic Web Working Symposium* (2001).
6. Brown, S. H. (2003). "VistA, U.S. Department of Veterans Affairs national scale HIS," *International Journal of Medical Informatics 69: 135-156, Bethesda, MD (USA)*.
7. Ceri, S., Fraternali, P., and Bongio, A., "Web Modeling Language (WebML): a modeling language for designing Web sites," *Computer Networks*, vol. 33, pp. 137-157, 2000.
8. Christensen, E., Curbera, F., Meredith, G., Weerawarana, S., (2001). "Web Services Description Language (WSDL) 1.1". [Online] Available: <http://www.w3.org/TR/wsdl20>.
9. Curbera, F., et. al. "Business Process Execution Language for Web Services (BPEL4WS 1.0)," *At WWW*, vol. 106
10. Daniel Austin, Abbie Barbir, Christopher Ferris, Sharad Garg, "Web Services Architecture Requirements", [online] Available: <http://www.w3.org/TR/wsa-reqs/>
11. Eichelberg, M., Aden, T., Riesmeie, J., Dogac, A., Laleci, G. (2005). "A survey and analysis of Electronic Healthcare Record standards," *ACM Comput. Surv.* 37(4): 277-315 (2005).
12. Evered, M., Bögeholz, S. (2004). "A Case Study in Access Control Requirements for a Health Information System," *Australasian Information Security Workshop* (2004).
13. Greenberg, M. M., et. al. (2005) "The Soundness and Completeness of Margrave with Respect to a Subset of XACML Tech Report CS-05-05," *Dept. of Comp. Sci., Brown Univ.*
14. Griffin, R., et. al. (2002). "Conformance Program Specification for the OASIS Security Assertion Markup Language (SAML)". *OASIS*, November (2002).
15. Guo, J., et. al. (2004). "The Development of MML (Medical Markup Language) Version 3.0 as a Medical Document Exchange Format for HL7 Messages," *J. Med. Syst.* 523-533.

16. HL7, "HIPAA Claims and Attachments Preparing for Regulation," [Online] Available: www.hl7.org/memonly/downloads/Attachment_Specifications/HIPAA_and_Claims_Attachments_White_Paper_20040518.pdf.
17. Hristidis, V., et. al. (2006). "A flexible approach for electronic medical records exchange," *In Proc. of the int'l WSh on Healthcare inf. and Knowledge Management*, Arlington, Virginia, USA, November 11, 2006. HIKM '06. ACM Press, New York, NY, 33-40.
18. Hung, P. C. K., et. al. (2007) "Research Issues of Privacy Access Control Model for Mobile Ad Hoc Healthcare Applications with XACML," *3rd Int'l WSh on Web and Mobile Inf. Services (WAMIS'07)*, Niagara Falls, Canada, May 21-23, 2007.
19. Jones, V. M., et. al., "A formal MDA approach for mobile health systems}," *EWMDA-2, Second European Workshop on Model Driven Architecture (MDA) with an Emphasis on Methodologies and Transformations, Canterbury, England*, 2004, pp. 28-35.
20. Jürjens, Jan, *Secure Systems Development with UML*, Springer, Berlin-Heidelberg, 2004.
21. Kagal, L., Joshi, T. (2003). "A policy language for a pervasive computing environment." *In Proc. of IEEE 4th Int'l WSh Policies for Dist. Sys and Net POLICY 2003*, pp. 63-74.
22. Kaviani, N., et. al. "Web Rule Languages to Carry Policies," *In Pro. of the 8th IEEE Int'l WSh on Policies for Dis. Sys and Nets*, Bologna, Italy, 2007, pp. 188-192.
23. Kaviani, N., et. al. "Integration of Rules and Policies for Semantic Web Services," *Int'l Journal of Advanced Media and Communication (IJAMC)*, 1(4) 2007.
24. Kifer, M., Lausen, G., Wu, J. (1995). "Logical Foundations of Object-Oriented and Frame-Based Languages," *J. ACM* 42(4): 741-843 (1995).
25. Loddarstedt, T., Basin, D., and Doser, J., "SecureUML: A UML-Based Modeling Language for Model-Driven Security," *UML*, vol. 2460, pp. 426-441, 2002.
26. Lukichev, S., et. al. (2007). "Using UML-based Rules for Web Services Modeling," *Proc. of the 2nd Int'l WSh on Services Engineering*, Istanbul, Turkey, April 2007, pp. 290-298.
27. M. Fleury and F. Reverbel, "The JBoss Extensible Server," *Int'l Middleware Conf.*, 2003.
28. M. Hafner, et. al., "Sectet: an extensible framework for the realization of secure inter-organizational workflows," *Internet Research*, vol. 16, pp. 491-506, 2006.
29. Milanović, M., et.al. "Business Process Integration by using General Rule Markup Language", *11th IEEE Int'l Enterprise Dist. Object Computing Conf.* Annapolis, USA, 2007.
30. Moses, T., (2005). eXtensible Access Control Markup Language (XACML) Version 2.0. OASIS Standard (2005).
31. Paige, R. F., Brooke, P. J., and Ostroff, J. S. 2007. Metamodel-based model conformance and multiview consistency checking. *ACM Trans. Softw. Eng. Methodol.* 16, 3.
32. R2ML Specification [Online] Available: <http://oxygen.informatik.tucottbus.de/R2ML/>
33. R2ML: R2ML to/from Other Rule Languages. [Online] Available: <http://oxygen.informatik.tu-cottbus.de/reverse-i1/?q=node/6>.
34. Uszok, A., et. al. "KAoS: A Policy and Domain Services Framework for Grid Computing and Semantic Web Services Trust Management," *2nd Int'l Conf. (iTrust 2004)* 16-26.
35. W3C RIF Working Group [Online] Available: <http://www.w3.org/2005/rules/wg>.
36. W3C Semantic Web Services Language Requirements. [Online] Available: <http://www.daml.org/services/swsl/requirements/swsl-requirements.shtml>
37. W3C Specification for WS-Policy, Web Services Policy 1.2 - Framework (WS-Policy) Reference: April2006 [online]. Available: <http://www.w3.org/2002/ws/policy/>
38. Wagner, G., et. al. "A Usable Interchange Format for Rich Syntax Rules: Integrating OCL, RuleML and SWRL," *Reasoning on the Web Workshop at WWW2006* (2006).
39. White, S.A., "Using BPMN to Model a BPEL Process". (2005) *BPTrends* 3 1-18.
40. Wozak, F., Schabetsberger, T., Ammenwerth, E. (2007) "End-to-end Security in Tele-medical Networks," *A Practical Guideline, Int'l J of Medical Informatics* (2007) 484-490.
41. Xiaou, Z., Keng P.H., "XML-Based Virtual Patient Records System for Healthcare Enterprises," *4th Int'l Conf. on Enterprise Info. Sys.* Ciudad-Real, Spain, April 3-6 2002.