Integration of Clinical Workflows with Privacy Policies on a Common Semantic Platform

Jan Werner[1], Bradley Malin[1,2], Yonghwan Lee[1], Akos Ledeczi[1] Janos Sztipanovits[1]

[1]Department of Electrical Engineering and Computer Science, School of Engineering
[2]Department of Biomedical Informatics, School of Medicine
Vanderbilt University, Nashville, TN 37232 USA
{jan.werner, b.malin, yohnghwan.lee, akos.ledeczi, janos.sztipanovits} @vanderbilt.edu

Abstract. As healthcare organizations (HCOs) migrate to electronic systems, they must ensure compliance with complex data protection legislation, such as the Health Insurance Portability and Accountability Act (HIPAA). Legislation specifies rules that must be enforced, but regulatory language is often imprecise, forcing HCOs to define local policies and procedures, as well as specific enforcement technologies. It is difficult for HCOs to ensure requirements are correctly translated across the enterprise, a problem compounded by the constant growth and evolution of deployed information technology (IT), such as clinical information systems (CISs). The consequence is HCOs frequently rely on *ad hoc* IT configurations, which are unverified and potentially conflict with an HCO's policy. Thus, it is crucial to develop (1) formal and computable representations of rules and requirements in data protection legislations, and (2) CISs that automatically enforce such specifications. This paper introduces a solution to these challenges by integrating HIPAA policy rules with a domain-specific model-integrated computing suite, tailored to the clinical enterprise. We present a detailed description of the policy-modeling process, the enforcement mechanism, and illustrate how to implement several policies, including mandatory access control and emergency access. All policies are formally specified through Prolog, but their enforcement is dependent on when their compliance can be evaluated. Static policies are enforced at design-time by mapping them to the structural constraints of system models. In contrast, dynamic policy rules, enforced at run-time, are loaded into a Prolog-based Policy Decision Point and Policy Enforcement Point, our extension to the standard SOA execution platform, which controls access to all services reliant upon protected health information. All models are sufficiently rich for integrating a CIS on a standard Service Oriented Architecture platform.

1 Introduction

Healthcare organizations (HCOs) are building and deploying electronic health record systems (EHRS) to improve service quality [1], reduce costs [2] and eliminate medical errors [3]. As HCOs become increasingly dependent on electronic systems, it is necessary to ensure that a patient's information can flow between disparate clinical enterprise systems. Yet, the healthcare industry has been slow to adopt clear standards, such as DICOM [4] and HL7 [5], thus making interoperability a significant challenge. To overcome such limitations, the software engineering community has initiated solutions based on the Service-Oriented Architecture (SOA), which aim to remove interoperability problems [6].

SOA-based solutions for the healthcare industry need to comply with varying federal and state data protection legislation and, thus, must integrate mechanisms to ensure medical records security and privacy. In the United States, for instance, the Privacy Rule of the Health Insurance Portability and Accountability Act (HIPAA) defines a base level of protections for patient-specific information, and EHRS in general, that must be addressed at administrative and technical levels [7]. Clinical information systems (CISs) must provide formal representation and mechanisms for compliance verification of privacy requirements, but current implementations are not satisfactory to achieve this goal. Traditional access control mechanisms do not cover the range of data protection requirements. As a result, various advanced requirements and mechanisms have been proposed, including the 4-eyes principle [8], emergency access [9], context-based access control [10], and organization-based access control [11]. Some approaches, such as the UCON control model [12] and the SECTET Security Framework [13, 14] have emerged and have been deployed within model-based computing environments; however, it is unclear how such mechanisms interfere with the execution of medical workflows.

In this paper, we illustrate that the separation of privacy constraints from business logic can lead to unexpected, and potentially harmful, behavior in a CIS. Alternatively, we demonstrate that the integration of policy abstractions with workflow models enables model verification and CIS correctness. We propose a solution for the integration of privacy requirements with CIS workflow models through a common semantic platform. This work integrates the following components: (1) Contextual Integrity [16, 17], a conceptual framework for describing and reasoning about privacy expectations and their implications; (2) the Model Integrated Clinical Information System (MICIS) framework [18], a model-based software toolkit with high-level modeling abstractions to represent complex clinical workflows in a SOA paradigm; and (3) Structural Semantics of Domain-Specific Modeling Languages [15], an approach for representing models and imposing constraints on a common semantic platform. We integrate privacy policy abstractions directly onto MICIS metamodels, reusing existing workflows, document models, and additional elements designed explicitly to represent privacy requirements. Privacy policies are enforced at the model-level using a logic-based model checker. Additionally, runtime enforced policies are generated from models and enforced by a Policy Enforcement Point.

The remainder of the paper is organized as follows. Section 2 provides an overview of research related to security and privacy in healthcare systems. Section 3 describes the system architecture and integration of the verification methods. Section 4 presents the methods, as well as several examples of healthcare-specific privacy policies within our solution. Finally, Section 5 concludes, discusses potential applications of our solution, and future work.

2 Background and Related Work

In this section, we review the applicability of existing solutions for capturing and enforcing privacy and security requirements in electronic healthcare environments. Specifically, we discuss standard access control models, present the SECTET framework for Model Driven Security, and the MICIS framework, the basis of this work. More detailed assessments and analysis of the access control models can be found in [14] and [19].

Access Control

Standard access control models are inappropriate for point-of-care healthcare environments. Some models, such as Discretionary Access Control (DAC) [20], are an ill-fit because health data does not exhibit the clear ownership required for its implementation. Other models, such as Mandatory Access Control (MAC) [20], are theoretically feasible, but their static nature and administrative overhead (e.g., management of extensive access control lists) in a dynamic environment, such as large academic medical centers, is impractical for every day usage. The most well-known approach, Role Based Access Control (RBAC) leverages the fact that “roles” are a normal abstraction of the clinical environment. RBAC is robust and supports many schemas, including DAC and MAC [21], and has been successfully applied in specialized healthcare domains [22]; however, RBAC does not gracefully address all the complex use cases that manifest in healthcare environments, such as rights delegation.

To overcome the aforementioned deficiencies, a new breed of access control models, such as Context-Aware Access Control [10] and Organization Based Access Control [11], have been proposed. These models extend RBAC with the concepts of organization, context, and contextual constraints. In addition, they provide a more fine-grained authorization mechanism. Though more suitable for dynamic healthcare environments than standard access control models, they are not sufficiently expressive to address advanced access control use cases. For instance, the “break glass” policy [25] allows for system users to elevate their access privileges when the need arises (e.g., in emergency cases) and such actions are subsequently audited by an HCO’s policy officials. Yet, there is no clear approach for integrating this mechanism with the proposed access control models.

Model-Based Security

From a model-based perspective, the SECTET framework for Model Driven Security [26] is a platform for modeling and deploying secure inter-organizational workflows. Modeling is supported through a domain-specific language based on UML, whereas security requirements are represented in a predicative OCL-based language. The SECTET framework has been successfully applied in various domains, including e-government [27] and e-health [28], which validates the appropriateness for a model-based approach extended with privacy and security constraints.

The SECTET tool suite was recently extended to support the UCON security model, and currently provides the most comprehensive framework for modeling and deploying secure workflows. The UCON framework [12] extends access control models with the concept of usage control. It provides mechanisms to control access during the action execution and on runtime attribute changes. The base concepts of UCON are authorizations, obligations, and conditions. The UCON model relies on trust between the server and the client, and enforcement of certain behaviors on the client-side using reference monitors. This mechanism allows for creating complex static and dynamic policies using UCON constraints. Nevertheless, the integration of privacy policies with the workflow models in SECTET lacks the foundations for formal verification.

Policy Specification

A theoretical framework for describing workflows and privacy requirements was recently proposed by Barth et al. [17]. This framework formalizes certain concepts of Contextual Integrity [16], using temporal logic to describe and reason about communication norms. The approach provides sound verification methods, but it is restricted in its workflow descriptions, which limits its application within real, complex healthcare environments and CIS implementations.

Model-Integrated Clinical Information Systems

MICIS [18] is a framework for the rapid development of experimental CISs. MICIS consists of two principle components: (1) a Model Integrated Platform for building workflow, document, organization, and policy models, as well as generating computer interpretable executables from the models; and (2) a Component Integration Platform for deploying clinical workflows. The Domain Specific Modeling Language (DSML) for MICIS allows for building workflow, document, organization and policy models that are tailored to the healthcare environment. These models are translated into executables, such as service descriptions, Business Process Execution Language (BPEL) workflows, deployment descriptions, and policy descriptions. These constructs are set up on a reference SOA platform with policy enforcement mechanisms. Privacy policies in MICIS are composed with workflows in the form of policy models and enforced in runtime using extensions of the reference platform.

The MICIS modeling platform provides a base for formal verification of the integration of privacy policies with the workflow logic. This paper extends MICIS with precise representations of privacy requirements and sound verification procedures of the resulting framework.

3 Methods

In this section, we describe the integration of privacy policies with system models. We specify the language composition that enables the verification of the resulting integrated workflow models. We present the formalization of DSML Structural Semantics and show how this method can be applied to represent and verify privacy policies in CIS models.

Structural Semantics

Structural Semantics is an approach for formalizing the structural semantics of a DMSL [15]. In this approach, a metamodel is translated into set of domain constraints, such that additional requirements can be directly injected into the resulting interpretation of the metamodel. This formalization allows for the formal analysis of such models. Models, the instances of the metamodel, are translated into a set of terms, which are verified using the domain constraints to prove model correctness.

In this work, we borrow the notation and definitions of [29] and define a *domain* as a formalization of a DMSL's rich syntax. A domain utilizes extended Horn logic to represent the constructs and invariants of modeling artifacts. Let $f(\bullet)$ denote an unary function over an universe that satisfies no additional equalities. We define $\sum$ as an infinite alphabet of constants. A *term* is either a constant drawn from $\sum$ or an application of some uninterpreted function of arbitrary arity on elements of the alphabet or other terms. A *signature* Y is a finite set of n-ary function symbols. A *term algebra* $T_Y(\sum)$ is an algebra where all symbols from the signature are uninterpreted. Functions are one-to-one with disjoint images and may be applied to all terms.

Syntactic rules are captured using Horn clauses defined over $T_Y(\sum)$. A Horn clause is a pair (h, T), where the head h is a term with variables and the tail T is a set of terms with variables. Semantics of Horn clauses are an evaluation of the set of clauses Θ over the set of terms T as follows: for each clause, variables in the tail are matched with the terms from T and if the valid substitution can be found, the head is added to the set of terms.

Formally, we define a *domain* as a structure $D = \{Y, \sum, \Theta\}$. This definition allows for specifying well-formed and malformed domain realizations using the *malform*(•) and *wellform*(•) invariants. A realization of the domain is well-formed if any *wellform*(•) term can be derived from the realization. Similarly, a model instance is well-formed if no malform(•) term can be derived from the realization. We use the notion of positive and negative domains, where positive domains instances derive the *wellform*(•) term, and negative domains do not derive any *malform*(•) term.

Integration of Privacy Policies in Terms of Structural Constraints

We now show how to define the workflow metamodel with privacy policy constraints. SOA provides a rich basis for mapping model abstractions and building inter-organizational workflows. The modeling language is constructed to reflect multiple elements of SOA, including services, messages, control, and dataflow. To represent privacy policies, the language is extended with abstractions that map message type, purpose of communication, and relations between the entities responsible for services. The metamodel of a simple workflow language is depicted in Figure 1.

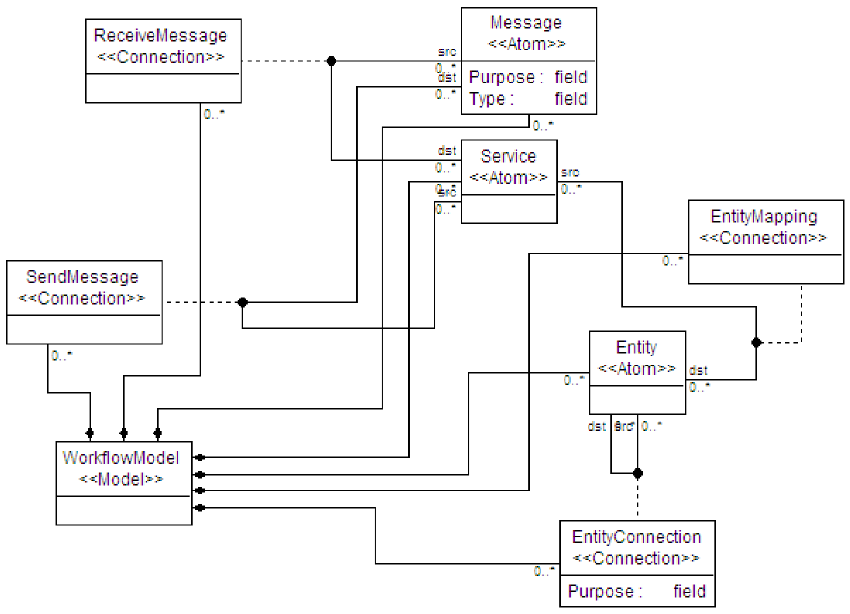

Figure 1 Simplified workflow language metamodel describing workflow abstraction

According to the metamodel in Figure 1, **Services** are connected through **Messages** using **SendMessage** and **ReceiveMessage**. **Services** relate to **Entities** using **EntityMappings** and **Entities** are connected using the **EntityConnections.** Each **Message** contains two attributes: **Type** and **Purpose**. This language is a frame for embedding privacy constraints represented as well-formedness rules. We use the Structural Semantics approach to describe the rich syntax of the language, describe privacy policies and verify and enforce the policies. This metamodel can be encoded using the following signature:

Y = {*workflowModel*(•), *sendMessage*(•,•), *receiveMessage*(•,•), *service*(•), *message*(•), *type*(•,•), *purpose*(•,•), *entityMapping*(•,•), *entityConnection*(•,•), *entity*(•), *type*(•,•), *purpose*(•,•)}

For verification and enforcement of privacy policies, we examine the workflow language as a negative domain such that any policy violation should invalidate the model. The policies are represented as a set of *malform*(•) terms denoting syntactical constraints of the language. The malformedness rules, *malform*(•), are denoted in Horn clause syntax. Two simple constraints stating that a model is malformed if (1) there is no message in the model (2) the message is not originated from the service; are presented below.

(1) *malform*(message(X)):- \+ *message*(*X*).
(2) *malform*(message(X)):- *message*(*X*), \+ *sendMessage*(*Y*, *X*)

We depict an example model realization in Figure 2. This model can be encoded using the set of terms:

M = {*service*(*Data Provider*),
service(*De-Identification*),
message(*Message*),
type(*PHI record*, *phi*),
purpose(*PHI record*, *de-identification*),
entity(*Covered Entityentity*(*Business Associate*),
sendMessage(*sm1*, *Data Provider*, *PHI record*),
receiveMessage(*rm1*, *PHI record*, *De-Identification engine*),
entityMapping(*em1*, *Data Provider*, *Covered Entity*),
entityMapping(*em2*, *De-Identification engine*, *Business Associate*),
entityConnection(*ec1*, *Business Associate*, *Covered Entity*) }

This set of terms represents the structure of the model and form a basis for verification of the privacy policies. To prove the well-formedness of this model and, consequently, its compliance with privacy policies, no *malfrom*() term defined for this domain should be derivable from the set of terms *M*.

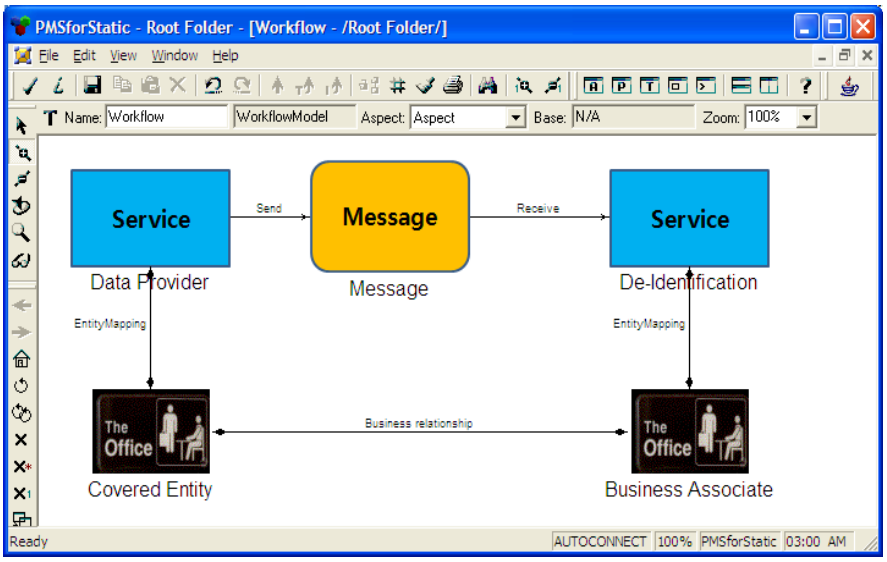

Figure 2 Workflow model, instance of the language presented in Figure 1

A set of constraints imposed on the model should guarantee workflow utility (e.g., message is sent from valid origin to valid destination) and privacy (e.g., message should not reach unauthorized entity). We can prove that a workflow achieves the utility goal: a message is sent from a data provider to a data receiver; however it does not satisfy the privacy goal (e.g., data receiver should not receive a message).

Similarly we can construct a model that satisfies the privacy requirements, but fails to achieve the utility goal. Integration of the business logic with the privacy requirements allows for analysis of both goals simultaneously and building correct models.

Integration of Privacy Policies Enforced in runtime

Privacy policies often depend on the specific contents of a message, such that their enforcement requires analyzing message instances. This analysis cannot be performed in the modeling phase. To address the problem of representation and enforcement of runtime evaluated privacy policies, we have extended the syntax of the workflow modeling language presented in [18] with the abstractions that are used to generate policy documents. To enforce these policies in runtime, we built an extension to the Web Service container Axis2 [30]. Our solution allows for evaluation of the policies using the contents of the messages and the history of the workflow executions. Furthermore, additional conditions may be fulfilled on the server side using obligations defined for the policy.

Obligations define additional actions executed upon the policy decision. The policy language associated with the workflow and document model allows for describing policies using Horn logic and for relating exchanged messages. A policy model defined with the metamodel is depicted in Figure 3. This model stores the policy document in the **Annotation** variable, relations using **typeRef** elements of document model connected with **AttributeMappings**, and configurations of the policy enforcement point. Policy documents stored in annotations are in the form of Prolog rules.

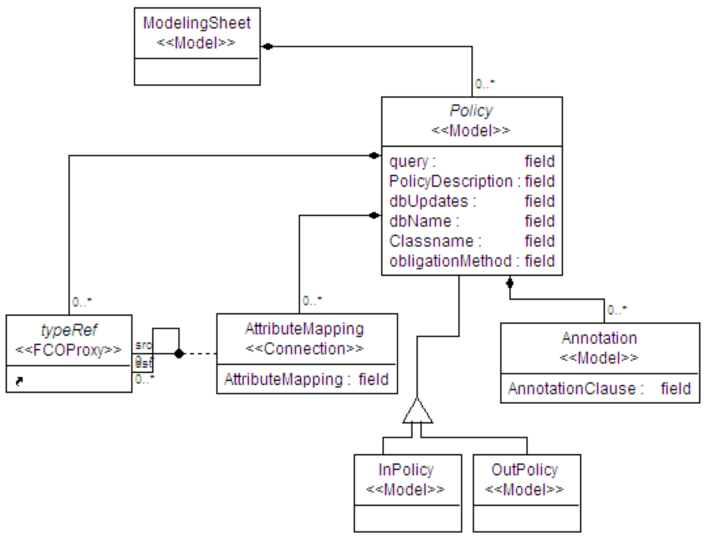

Figure 3 Metamodel of the privacy policy constructs, integrated with the MICIS modeling language

This approach allows for building very expressive rule-based policies; however, it requires knowledge of the Prolog syntax by the policy implementer. Variables from the Document Model are used to configure the Policy Enforcement Point to extract the necessary information from the passing message and provide it to the Policy Decision Point (PDP) for policy evaluation. Additional options specify the point of the policy enforcement (e.g., inbound or outbound message), information of the workflow execution history, and the obligations. The policy translator generates the policy documents and policy descriptions from workflow, document, and policy models.

Policy documents, along with the policy descriptions are deployed in the Web Service Container that hosts the protected services. The deployment architecture is based on Web Service protocols and standards. In Figure 4, we depict details of the architecture extension and relationship to the modeling environment. The set of Web services that implement the workflow logic is protected using the Context Handler, which intercepts all incoming and outgoing SOAP messages. The Policy Enforcement Point (PEP) is located above the Web Service security layer, which handles the basic security requirements, including confidentiality, integrity and availability. The Context Handler first intercepts the service invocation and loads the policy description that instructs how to handle the message. If the policy should be applied to the incoming message, then the PEP extracts the message context and prepares the request for the PDP. The PDP restores the saved state (if such state has been stored), loads the policy document and evaluates the access request. Based on the decision from the PDP, the PEP executes the obligations and passes the invocation to the service or drops the message if the access has been denied by the PDP. A similar procedure is executed for the service response; however, for the outbound

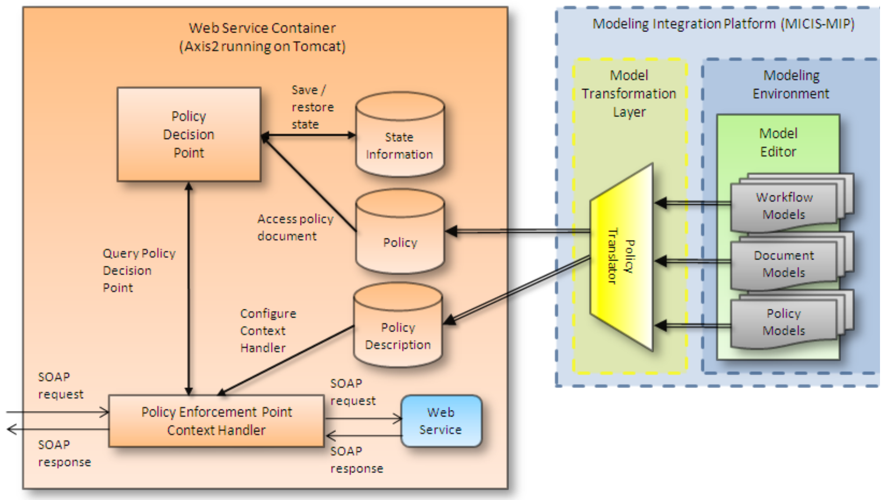

Figure 4 System Architecture

message, the PDP may access both the SOAP request and reply, and base the decision on the results provided by the service.

4 Results

We present the results of the integration of the workflow models with the privacy policies represented as structural constraints. We present two examples, one based on the HIPAA Privacy Rule policy of data sharing for de-identification purposes and one based on the “break-glass” emergency access scenario.

Compliance at Modeling Scenario: De-identification via Business Relations

The policy presented in this example is derived from section 164.502.d of HIPAA , which specifies the conditions of uses and disclosure of de-identified Protected Health Information (PHI). The plaintext rule is presented in Appendix 1. The meaning of the rule can be described as follows:

> *A covered entity may send protected health information to a business partner for de-identification purposes only if there exists a contractual agreement between the communicating entities.*

The model depicted in Figure 2 represents a communication instance satisfying the requirements of the rule. The **Medical Record Database,** which belongs to the **Covered Entity,** sends a message that includes **PHI record** with attributes representing the purpose and type of the message (not shown on the figure), to the **De-Identification engine,** which belongs to the **Business Partner**. There exists an entity connection between the **Covered Entity** and **Business Partner**, which represents the existing relation between these two entities. The workflow model is represented as a negative domain, so to prove compliance of the modeled system with the privacy policy, it should be impossible to derive the *malform*(•) rule. The privacy policy is described as the following malformedness rule in Horn clause form:

no_connection(*E*1, *E*2) *:- \+ entityConnection*(*X, E*1*, E2*)

malform(message(*M*)) *:-* *message(M),sendMessage*(*MF, S1, M*),
receiveMessage(*MF2, M, S2*),*entityMapping*(*EM1, S1, E1),*
entityMapping(*EM2, S2, E2*),*type*(*M,'phi'*), *no_connection*(*E1, E2*).

To derive the malform clause, all of the terms in the tail must be satisfied. The model needs to connect two services with the message with attribute type equal to ‘phi’, which denotes that the message contains protected health information, and the entities containing communicating services may not be connected. The constraint (*no_connection*) is an example of negation as a failure extension of Horn logic. This constraint is only satisfied when there are no terms that can be substituted for variables *X*, *E*1, *E*2 in the rule *no_connection*(*X, E1, E2*). In other words, the system invalidates the workflow if the communicating entities have not established a formal business relationship. The verification of the model proves that the *malform*() term cannot be derived and, consequently, the model is classified as complying with the privacy policy.

Compliance Evaluation at Runtime Scenario: Break-Glass
We present the enforcement of the runtime policies using the following example:

Access to the patient's medical record should only be granted to primary care physicians listed in medical record, or in case of emergency situation access should be provided to any physician following the "Break Glass" policy.

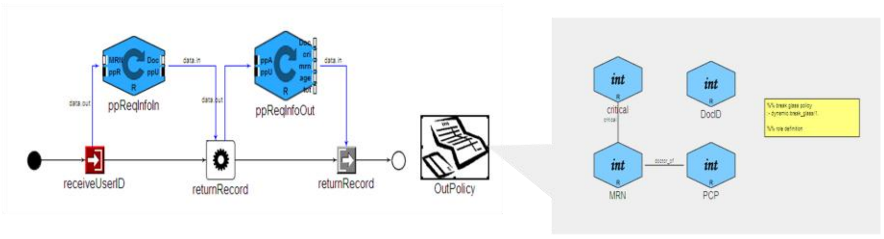

Figure 5 Sample model of a service presenting workflow and dataflow with attached

This policy requires a fine-grained access control for description of the relationship between patient and physician, as well as arbitrary conditions for application of the "Break Glass" [25] policy. This policy is represented in SOA terms as follows. The **RetrieveData** service may provide the medical record to the requestor if the request is issued in an emergency context or if the response of the service contains the list of physicians that includes the credentials of the physician from the request. Additionally, if the request was allowed using the break-glass policy, the PEP should create an audit trail, executed as an obligation, for administrative review. We built a model of a service providing medical records, and introduced the privacy policy, that regulates access to the service (See Figure 5). This model is translated using the model interpreters to the pair of documents presented below – policy description [configuration for the Policy Enforcement Point]:

```
<PolicyList>
  <PolicyDescription>
    <methodName>RetrieveData</methodName>
    <query>retrievedata(MRN,DocID)</query>
    <policyDbName>'C:/Policy/factsdb.pl'</policyDbName>
    <inPolicy>False</inPolicy>
    <outPolicy>True</outPolicy>
    <requestFields>MRN, critical </requestFields>
    <replyFields>DocID, </replyFields>
    <relations>is_critical(MRN ,critical), treats(MRN,PCP)</relations>
   <dbUpdates> accessed(DocID, MRN)</dbUpdates>
   <classname> edu.vanderbilt.isis.bpeltools.logCriticalAccess</classname>
   <obligationMethod>obligationmethod</obligationMethod>
  </PolicyDescription>
</PolicyList>
```

and the policy document

```
%% definition of helper symbols
   :- dynamic treats/2.
   :- dynamic critical/2.
%% access rule
   :- dynamic retrievedata/2.
%% define break glass rule allow access in emergency context.
   break_glass(RecordNo):- is_critical(RecordNo, X), X>0.
%% access control rules for the record
%% only physicians who treats the patient can access his record
%%access is provided if patient is marked as in critical state
   retrievedata(RecordNo,DocId):-  treats(RecordNo, DocId);break_glass(RecordNo).
```

The presented policy contains only positive permissions (i.e., access is granted if the query *retrievedata*(*MRN*,*DocID*) can be deduced); however, representation of the policies in the form of Prolog rules allows for an arbitrary policy composition. The policy designer can employ multiple strategies to resolve possible conflicts between the policies, as well as include negative polices that deny access for specific conditions.

5 Conclusion and Further Research

In this paper we proposed how to integrate formal logical representations of privacy policies with workflow models through a common semantic platform. We demonstrated how a Structural Semantics approach can be applied to formalize and verify privacy requirements within this framework. Technically, we showed that a domain-specific model integrated computing framework, the Model Integrated Clinical Information System, can be leveraged to develop clinical information systems that comply with privacy legislation in a verifiable manner. We illustrated how to apply this approach through several examples specific to well-known regulatory requirements. We offered an extension to the Service-Oriented Architecture platform that allows for enforcing privacy policies in runtime. We are currently working on classifying HIPAA rules that can be represented and enforced using Structural Semantics approach. In future work, we plan to evaluate overhead introduced by the runtime policy enforcement and applicability in large-scale distributed environments. we hope to develop tools for model generation that synthesize clinical workflow models using the specification of requirements and policies

Acknowledgements

This work was supported in part by TRUST (Team for Research in Ubiquitous Secure Technology), which receives support from the National Science Foundation (NSF award number CCF-0424422) and the following organizations: AFOSR (#FA9550-06-1-0244), BT, Cisco, ESCHER, HP, IBM, iCAST, Intel, Microsoft, ORNL, Pirelli, Qualcomm, Sun, Symantec, Telecom Italia, and United Technologies.

References

1. Committee on Quality of Health Care in America. Institute of Medicine. Crossing the Quality Chasm: A New Health System for the 21st Century. National Academy Press, Washington, DC (2001)
2. Gunter, D.T., Terry, P.N.: The emergence of national electronic health record architectures in the United States and Australia: models, costs, and questions. Journal of Medical Internet Research **7** (2005) ,
3. Kohn, T., Corrigan J.M., Donaldson, M.S.: To Err is Human: Building a Safer Health System. Washington DC, National Academy Press, 2000.
4. Digital Imaging and Communications in Medicine Standard ftp://medical.nema.org/medical/dicom/2008/
5. Health Level Seven Standard. http://www.hl7.org/
6. Vogl, R., Breu, M., Schabetsberger, T., Wurz, M.: Architecture for a distributed national electronic health record in Austria aiming at an open source solution. In Proc. 24th International EuroPACS Conference EuroPACS 2006 (2006) 67–77
7. Health Insurance Portability and Accountability Act http://www.hhs.gov/ocr/hipaa/
8. Vogt, G.: Multiple authorization - a model and architecture for increased, practical security. In Proc. IFIP/IEEE Eighth International Symposium on Integrated Network Management (IM2003), Colorado Springs, CO, IFIP/IEEE, Kluwer Academic Publishers (2003) 109–112
9. Joint NEMA/COCIR/JIRA Security and Privacy Committee (SPC). ”breakglass – an approach to granting emergency access to healthcare systems”. ”http://www.nema.org/prod/med/security/”.
10. Tzelepi, S.K., Koukopoulos, D.K., Pangalos, G.: A flexible content and context-based access control model for multimedia medical image database systems. In Proc. 2001 Workshop on Multimedia and Security: New Challenges (2001)
11. Kalam, A.A.E., Baida, R.E., Balbiani, P., Benferhat, S., Cuppens, F., Deswarte, Y., Miege, A., Saurel, C., Trouessin, G.: Organization based access control. In: Proc. IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2003) (2003) 120-131
12. Park, J., Sandhu, R..: Towards usage control models: beyond traditional access control. In: Proc. 7th ACM symposium on Access control models and technologies (SACMAT ’02), ACM Press, New York, NY (2002) 57–64
13. Hafner, M., Agreiter, B., Breu, R., Nowak, A.: SECTET: an extensible framework for the realization of secure inter-organizational workflows. Journal of Internet Research (2006) 16
14. Alam, M., Hafner, M., Memon, M., Hung, P.: Modeling and enforcing advanced access control policies in healthcare systems with SECTET. In: Proc. ACM/IEEE Workshop on Model-Based Design of Trustworthy Health Informaton Systems (2007)

15. Jackson, E.K., Sztipanovits, J.: Towards a formal foundation for domain specific modeling languages. In: Proc. 6th ACM International Conference on Embedded Software (EMSOFT'06), Seoul, South Korea (2006)
16. Nissenbaum, H.F.: Privacy as contextual integrity. Washington Law Review 79 (2004).
17. Barth, A., Datta, A., Mitchell, J.C., Nissenbaum, H.: Privacy and contextual integrity: framework and applications. In: Proc. 2006 IEEE Symposium on Security and Privacy (2006)
18. Mathe, J., Werner, J., Lee, Y., Malin, B., Ledeczi, A.: Model-based design of clinical information systems. Methods of Information in Medicine. Forthcoming.
19. Ferraiolo, D., Kuhn, D.R., Hu, V.C.: Assessment of access control systems. Technical Report NISTIR 7316, National Institute of Standards and Technology, US Department of Commerce (2006)
20. National Institute of Standards and Technology. Role Based Access Control. http://csrc.nist.gov/groups/SNS/rbac/
21. Mavridis, I., Pangalos, G., Khair, M.: eMEDAC: Role-based access control supporting discretionary and mandatory features. In: Proc. IFIP Workshop on Database Security (1999) 63-78.
22. Beznosov, K.: Requirements for access control: US Healthcare domain. In: Proc. 3rd ACM Workshop on Role-Based Access Control, Fairfax, Virginia (1998)
23. Schabetsberger, T., Ammenwerth, E., Breu, M., Breu, R., Mair, R., Penz, R., Vogl, R.: Reference implementation of a shared electronic health record using medical data grids with an RBAC based security model. In: Proc. of the 2nd AGRID Symposium, in conjunction with 6th Austrian-Hungarian Workshop on Distributed and Parallel Systems (2007)
24. Hu, J., Weaver, A.C.: Dynamic, context-aware access control for distributed healthcare applications. In: Proc. Pervasive Security, Privacy, and Trust Workshop (2004)
25. Joint NEMA/COCIR/JIRA Security and Privacy Committee (SPC). ”breakglass – an approach to granting emergency access to healthcare systems”. http://www.nema.org/prod/med/security/
26. Hafner, M., Agreiter, B., Breu, R., Nowak, A.: SECTET: an extensible framework for the realization of secure inter-organizational workflows. Journal of Internet Research **16** (2006)
27. Breu, R., Hafner, M., Weber, B., and Nowak, A.: Model driven security for inter-organizational workflows in e-government. In: Proc TCGOV (2005) 122-133.
28. Alam, M., Hafner, M., Breu, R., Hafner, M.: Modeling permissions in a (u/x)ml world. In: Proc IEEE ARES (2006)

29. Jackson, E., Schulte, W., Sztipanovits, J. The power of rich syntax for model-based development. Technical Report MSR-TR-2008-86, Microsoft Research. Redmond, WA (2008)
30. Apache Axis2 http://ws.apache.org/axis2/

Appendix 1: HIPAA privacy rule 164.502.d:

Standard: Uses and disclosures of de-identified protected health information

(1) Uses and disclosures to create de-identified information. A covered entity may use protected health information to create information that is not individually identifiable health information or disclose protected health information only to a business associate for such purpose, whether or not the de-identified information is to be used by the covered entity.