ESAC-BPM: Early Security Access Control in Business Process Management

Mahmoud F. Ayoub
Computer and Systems Engineering
Alexandria University
Alexandria, Egypt
Email: mfayoub@alexu.edu.eg

Riham Hassan
Computer Science
Virginia Tech
Blacksburg, VA, USA
Email: rhabdel@cs.vt.edu

Hicham G. Elmongui
Computer and Systems Engineering
Alexandria University
Alexandria, Egypt
Email: elmongui@alexu.edu.eg

***Abstract*—Business process modeling notations do not provide explicit means to model security aspects such as access control, integrity and confidentiality. Business analysts who are not typically security experts are incapable of modeling security aspects that could not be modeled in business process modeling notations. In this paper, we propose systematic means to model access control explicitly in business process models. More specifically, we used Business Process Modeling Notation (BPMN) as a graphical notation to represent processes. Our proposed technique exploits BPMN by employing business rule activities to carry the access control logic as If-Then rules with conflict detection capabilities. We prove the validity of ESAC-BPM formally. Further, we demonstrate the technique using a case study for a reservation process for a movie store by telephone, that needs data access control policies to be applied on the process model.**

***Index Terms*—Business process management, security data access control, business rule activities.**

I. Introduction

Business Process Management (BPM) is the process of optimizing business processes and aligning all the organizational aspects with the requirements needed to be in a software. BPM concerns about the validity, performance and agility of business processes. A business process is a network of activities done by collaborators to achieve a business goal.

Business Process Modeling Notation (BPMN) [1] is used to represent business processes. With this unified graphical representation, all business stakeholders can easily understand business processes and can adjust any business modifications quickly and in a standard way. It is very similar to activity diagrams of Unified Modeling Language (UML). Therefore, it is intuitive to business users and technical engineers, and hence it can enclose the gap between them.

BPMN consists of graphical constructs and objects that can be mapped to execution languages like Business Process Execution Language (BPEL) as in [2], [3], and [4]. As a result, the pace of processes development has increased and process management became easier.

While modeling, a business analyst finds problems in expressing all the business requirements. Many of such requirements are about security. Engineering a secure software is a challenging problem. In industry, almost security is the last aspect to be considered and it is added to the software in an adhoc manner. The analysts are the best ones in the software cycle to know about the security holes and how to handle them. However, they do not have a direct control on security policies. Instead, they have to forward the required changes to software engineers.

One important aspect of software security is the data access control, which is a necessary and crucial design element for any secure application. In general, an application should protect its data and system resources against unauthorized access by implementing access control restrictions on what users can do. Access Control refers to the much more general way of controlling access to data, including restrictions based on things like the time of day, age, gender, the IP address of the HTTP client browser, or any other derived variables that can be extracted or calculated easily. Simpler access control models often cannot adequately meet the complex access control requirements that such relationships require, and so more granular, powerful, dynamic models and mechanisms are needed to address these new realities. In short, increasingly complex data access and sharing drive the need for increasingly complex access control models and mechanisms.

Attribute Based Access Control (ABAC) is one model of access control. It uses attributes in a structured language to define access control policies. There are 3 kinds of attributes: subject, object and environment. 1) The subject represents who requests data access. In typical applications, It can be a user, application, or process. Subject related attributes are like age, name, gender or role. 2) The object is the target identity to be secured. It can be a file, database, data object or web resource. Object related attributes are like name, size or URL. 3) The environment is the context where the access request happens. Environmental attributes are like time of day, weather, season or place of request.

Process modeling languages and security policies languages are both used to document organizational policies and procedures. While process modeling languages describe a procedural sequence of activities, security policy languages often rely on a declarative description of security constraints. Understanding the relationship between the two languages would maximize benefit, avoid content duplication, and reduce their overall effort [5]. In this paper, we present a provably systematic and easily deployable approach for embedding security access control, as one example of security aspects,

that is designed for BPMN diagrams. Our approach is based on a novel usage of business-rule activities which are in the BPMN specification. So, we do not need to modify the BPMN meta-model. It is based on putting all the security logic as If-Then rules. Additionally, we integrate our approach with an option to detect conflicts [6] between access control policies. Access control policy conflicts usually happen as a result of their complexity. We adopt ABAC as it is the most flexible access control model, and it can be a replacement to any access control model as shown later. In ABAC, access control policies are given as boolean expressions, which are easily written and understood. They are comprised of conditional attributes and boolean operators ($\neg$, $\wedge$ and $\vee$). They are mapped to business rules and are wrapped by a business rule activity. Upon activation, a business rule activity activates the associated business rules and on their completion, the activity terminates. A business rule activity hides all the security logic in its underlying properties. As a result, all complex logic is not visible in the graphical view of a process model, and so it is still easily understood. In such a way, we provide a complete software solution by considering early security while modeling business processes. Our evaluation shows that the proposed approach can achieve significant secrecy gain with minimal additions compared to the state of the art.

The rest of the paper is organized as follows. In Section II, we discuss the related work. In Section III, we present our approach. In Section IV, we prove the validity of our approach. In Section V, we provide a complete test case example examining our approach. And, finally, the conclusion and future work are in Section VI.

II. Related Work

Concerning different related work of embedding security in business processes, there are many ways to do so with different approaches. A first popular approach is to add text annotations to different BPMN constructs. These annotations contain a formal specification of security policies and need to be parsed afterwards at runtime to enforce security requirements. There were many security policy specifications languages used and are presented here. Second approach changes the meta-model of BPMN to add new constructs for security requirements. These constructs are translated into security policies to be enforced at runtime. Our approach does not change the meta-model, but it creates new fragments based on business rule activities and some gateways. So it is easily deployable.

Mülle et al. [14] and Darnianou et al. [15], propose languages for security policies specification. A business analyst should study it for usage. In [18], security elements and process models are integrated. The language constructs can be inserted as a text annotation associated with BPMN constructs. Annotations are to be compiled and enforced at the process execution stage. The language supports access control by providing authorization, delegation, information filtering and refrain policies. Additionally, they provide obligations, basic constraints, meta policies and policy composition. After integrating the process model with security elements, the graphical representation of the process is complex and hard to be understood. It is better to externalize all the security logic.

In [16], a Policy Description Language (PDL) is proposed. It is declarative and consists of 3 categories: 1) Set of events. 2) Set of actions. 3) Set of functions to evaluate the environment. Policy rule propositions take the following form.

event **causes** *action—event(s)* **if** *condition*

This means that if *event* occurs in a situation where *condition* is true, then the action or consequential events specified will be executed. The language is a generic language that can be used to describe any type of policies and not only security policies. This type of formalization will allow representing obligations, and prohibitions.

Rodríguez et al. [10] and Menzel et al. [11], concern about the graphical representation of security constructs. Additionally, they add to the meta-model of BPMN. Security requirements are included in the extended meta-model. They do not cover how the underlying layer works for enforcing such security constraints. Additionally, they map each security requirement to a BPMN construct and add a mark to it. Supported security aspects are non-repudiation, attack harm detection, integrity, privacy, access control, security role, and security permissions. In contrast to our proposition, there are no clues about how security constraints are to be enforced. They model the requirements and leave it up to the developer to decide how to implement them, which may introduce security holes.

Wolter et al. [12] and [13], constitute their generic security model that specifies security goals, policies, and constraints based on a set of basic entities, such as objects, attributes, interactions, and effects. It is a general description regardless the notation used. For BPMN, they visualize the security requirements as artifacts like a text annotation to a message link for message confidentiality. They use the group artifact on some activities to express the separation of duties (SoD) security aspect.

In [17], a similar methodology, to our approach, for modeling security requirements in business processes is proposed. They cover both the design-time modeling and run-time enforcement of security requirements for business processes. Such requirements are translated to XACML policies which are enforced by generated Policy Enforcement Points (PEPs). Additionally, Policy Decision Points (PDPs) are generated to decide if a certain request is granted or not. A prototype is implemented based on Activiti (http://www.activiti.org/), extending the Eclipse designer and process engine. Some security requirements are represented as new constructs in BPMN and others like access control are represented by domain-specific user interface.

In [19], security needs are supported by the commitments view, which consists of a set of commitments between actors. The conversation and choreography diagrams of BPMN 2.0 are targeted. An overview of intercompany processes between several partners is given. Hence, annotations are used to show which conversations and related participants the requirements

apply.

III. Formal Method of Security Access Control in BPMN

While accessing data objects, we authorize the incoming read/write request based on the access control policies of the requested object. For accessing data base object, we do the same for the CRUD operations. Our proposed approach adopt ABAC because of its flexibility and context awareness. Access control policies are represented by boolean expressions comprised of conditional attributes and boolean operators ($\neg$, $\wedge$ and $\vee$). Boolean expressions are transformed to business rules. Our approach does not modify the meta-model of BPMN's, because there is a native support for business rules via business rule activities. When a model token activates such activity, the associated business rules are called. On completion, the business rule activity completes and the result is the actions specified in the body section of rules. In our case, for handling access control policies, the action is either permit or deny the incoming request.

Our approach uses system-wide rules by business analysts to model security aspects (mandatory access control policies). As illustrated in Figure 1, when the regional manager receives a message of a loan application, it is reviewed using the loan data file, and then the manager notifies the customer with the result of application either by acceptance or refusal. In this example, a business analyst is incapable of adding constraints on accessing the data file like restricting the access time in a certain duration of a day ($6am \leq time \leq 6pm$) or other constraints.

Our proposed approach solves this problem by adding a construct to be activated before accessing data objects. As depicted in Figure 2, the secure read sub process is inserted before accessing loan data file. This process can throw an error event if it denies the incoming access request, and the business analyst should handle the thrown error in a proper way. In Figure 3, we give a further insight into our construct. It contains a business rule activity that has all the logic of the access control policy. The activity is followed by a gateway to differentiate between a granted access and a denied one. If the access is accepted, the subprocess finishes normally. If it is denied, the subprocess throws an intermediate error event to the calling process.

It is better to externalize the decision logic in an external decision table like business rules rather than organizing them among gates that control the model's token path. In the former approach, business analyst can express complex policies, easily review and validate the business rules controlling a certain data object. On the other hand, the latter approach is subject to policy changes and requires significant maintenance.

IV. Validity Proof of Formal Method

To prove the validity of our approach, we split the proof into 3 steps as follows. 1) Traditional access control models can be converted to ABAC. 2) We can transfer any ABAC policy to some business rules. 3) Conflict detection. We present each of these steps in detail in the following subsections.

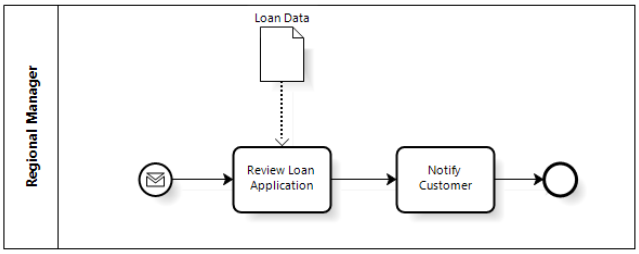

Fig. 1. Example before secure read activity

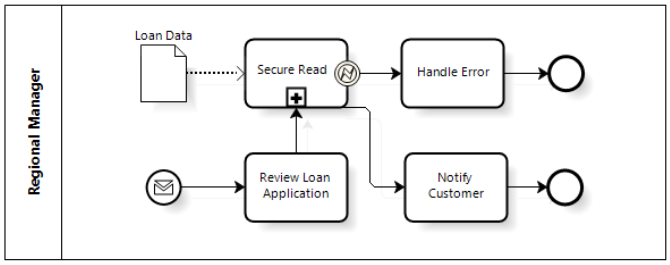

Fig. 2. Example after secure read activity

A. Traditional Access Control Models to ABAC

ABAC is flexible enough among all other traditional models, as presented in [7] and [8]. So we can express various access control policies. Because they all depend on attributes of 3 domains: subject, object and environment. 1) Subject represents the identity who requests for a data access. Typical attributes are like age, name, gender or role. 2) Object is the target identity to be secured. It can be a file, database, data object or web resource. 3) Environment is the context where the access request happens. So for using ABAC, we have to convert the given domain to its attributes and then create the ABAC policies. We do a proof by complete induction on all access control models from [9] and convert each one to the corresponding ABAC model as follows.

1) Role Based Access Control (RBAC) to ABAC: RBAC only concerns about roles in the system. So one of the inherent

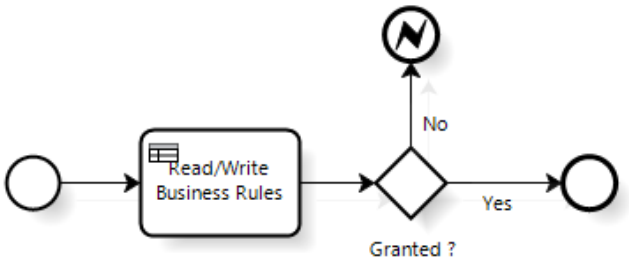

Fig. 3. Collapsed secure read subprocess

TABLE I
MODELING RBAC RULES

Conditions	Result
Role	**Access Granted**
Manager	Yes
Clerk	No
TA	Yes

TABLE II
MODELING BLP READ RULES. THE SIMPLE SECURITY PROPERTY (NO READ-UP). EXPRESSION: $SubjectLabel \in \{Owner, Manager\}$ AND $Operation = "Read"$

Conditions		Result
Subject Label	**Operation**	**Access Granted**
Owner	Read	No
Manager	Read	Yes
Clerk	Read	Yes
Customer	Read	Yes

TABLE III
MODELING BLP WRITE RULES. THE STAR PROPERTY (NO WRITE-DOWN)

Conditions		Result
Subject Label	**Operation**	**Access Granted**
Owner	Write	Yes
Manager	Write	Yes
Clerk	Write	No
Customer	Write	No

limitations of RBAC is the single dimension of roles. As a result, if we want to model multiple attributes, the number of roles needed to encode these attributes will grow exponentially. Additionally, RBAC does not support environment attributes and can not model BellLaPadula model. Table I shows how to model an RBAC to an ABAC one.

2) BellLaPadula (BLP) Model to ABAC: BLP is used in military organizations where there are object security labels and subject clearances. They build the finite state machine of the model. Assume that we have a model with the following specifications. Object labels are ordered from top secured to lower level as follows.

1) TOP SECRET
2) SECRET
3) CONFIDENTIAL
4) PUBLIC

The subject clearances are ordered in the same way as follows.

1) Owner
2) Manager
3) Clerk
4) Customer

Tables II and III show how to model this BLP model to an ABAC one assuming we have a SECRET object to be secured.

This policy can be expressed by two attributes: $SubjectLabel$ and $Operation$ as follows

$$(SubjectLabel \in \{Owner, Manager\} \wedge Operation = "Read") \vee (SubjectLabel \in \{Manager, Clerk, Customer\} \wedge Operation = "Write") \quad (1)$$

TABLE IV
BUSINESS RULES FOR BPL MODEL

Condition	R1	R2	R3	R4	R5	R6
Subject Label	Owner		Manager		Clerk, Customer	
Operation	Read	Write	Read	Write	Read	Write
Access Granted	Yes	No	No	Yes	No	Yes

B. ABAC Policy to Business Rules

In this subsection, we have an input ABAC policy expressed as a boolean expression. We want to transform it to some business rules that satisfy the expression. We deal with business rules because BPMN has a complete support to them. This is our main contribution that we make use of business rules in embedding security in business processes. So there is no need to modify the meta-model of BPMN.

The input boolean expression will consist of attribute variables each of which belongs to a certain range of the whole attribute domain and logical operators ($\neg$, $\wedge$ and $\vee$). A family of disjoint sets of each attributes can be detected from the given boolean expression. Union of these sets is the whole attribute domain and there is no intersection between them. These sets make up the business rules to be put in business processes. The cardinality of the number of business rules is given as follows:

$$NumberOfBusinessRules = \prod_{i=1}^{N} |A_i| \quad (2)$$

where N is the number of attributes in policy, A_i represents the i^{th} attribute and $||$ returns the number of disjoint sets of the given attribute. For example, in Equation 1, there are three sets for the $SubjectLabel$ attribute: {Clerk, Customer}, {Owner} and {Manager}. For $Operation$ attribute, there are two sets: {Read} and {Write}. Hence, we have six business rules and we can put them in a decision table as in Table IV. Decision trees can be used for visualizing business rules. But they are very brittle when rules change and require significant maintenance. Additionally, they are more complex when each parameter potentially has a large number of different values where each possible parameter value becomes a node at a branching point in the tree. On the other hand, decision tables, as in Table IV, can be used instead which are more compact and intuitive when many rules are needed to analyze many combinations of attribute values.

C. Conflict Detection

Several policies control access to each object. Each policy consists of some business rules that adheres what the policy specifies. Each rule has an action (whether permit or deny) to access the object. Due to enterprise business processes and complex access control policies, some conflicts may arise. A conflict arises when two policies having some conditional attributes in common but different in their actions.

We adopt the approach in XACML access control policies [6], as in Algorithm 1, by representing each policy in d-dimensional space, where d is the number of conditional

attributes. The plane sweep algorithm, as in [20], is used for detecting pairwise conflicts. It has three steps: 1) project each policy on a specified dimension plane and sort the projected values. 2) sweep the plane. 3) report intersections.

Algorithm 1 Conflict Detection Algorithm

```
map all policies to the d-dimensional space
determine start and end of the range every policy covers
for each dimension do
    determine all intersections via plane sweep algorithm
    prune all policies that cannot conflict another one
end for
report all pairwise conflicts
```

The complexity of the plane sweep algorithm is $O(n * log(n))$. And due to the fact that it is used d times in the conflict detection algorithm. So the overall complexity of the conflict detection algorithm is as in Equation 3

$$= d * n * log(n) \tag{3}$$

where n is the number of policies and d is the number of dimension.

We have developed many techniques to resolve the conflicts among the rules. However, we omit their details due to the space limitations. These techniques depend on how the business rule engine works. Some techniques can call the business rules in serial manner and finish when it takes a decision from the calling business rules. Other techniques can call all the business rules in parallel, and then aggregates the results by voting algorithm, permit dominates, deny dominates or others. A business analyst can do the proper modifications to remove any conflict.

V. Case Study

Our illustrative example describes a typical business process for a movie rental shop. Reservations are done via telephone calls by customers to store clerks. There are two requirements to be considered for business security and management. 1) Basic requirement in which access control is based on customers' age and the movies content ratings. Ratings are Restricted (R), Parented Guidance Strongly Cautioned (PG-13) and General Audience (G). 2) Advanced requirement which introduces membership classes (Premium, Regular), which enforces a new policy that only Premium users can view new releases. Figure 4 shows the details of what a movie store clerk does to securely create a right order for the incoming call request according to the store policies. The diagram also handles any possible attack for the data objects.

Figure 4 describes a business process of a movie store that is triggered by a phone call. The clerk responds, and then review the customer data. Before accessing *Customer Data* file, we can apply access control policies on the subject role, username, time of request, or type of operation. As shown in Table V, it restricts the accessing of file to either clerks and managers. Additionally, it puts additional time constraint

TABLE V
Decision table for accessing the customer data object file

Conditions		Result
Role	**Time**	**Granted**
Clerk	$8am \geq time \leq 4pm$	Yes
Manager	Any	Yes

for each role. On activation the *Secure Read* business rule activity, the associated business rules in Table V are activated and produce the result, depending on the business rule engine, either with permit or deny the incoming access control request. If the business rule denies the request, the process terminates to prevent an unauthorized access.

In case of granting the access control request, the process execution continues and checks whether the customer already exists in the system or not. If not, the process registers the customer into the system. In this case, we may not need any access control policies because they are already granted at first. If the customer already exists, the process proceeds with writing the incoming order. The process should enforce the management requirements. So, the *Secure Order* business rule activity is inserted before accessing the movies data file. The ABAC policies for the basic and advanced requirements are represented as boolean expressions in Equations 4 and 5, respectively.

$$(Age \geq 21 \wedge Rating \in \{R, PG13, P\}) \vee (21 > Age \geq 13 \wedge Rating \in \{PG13, P\}) \vee (Age < 13 \wedge Rating \in \{P\}) \tag{4}$$

$$(MemberType = "Premium") \vee (MemberType = "Regular" \wedge MovieType\neg = "NewReleased") \tag{5}$$

The *Secure Order* business rule activity either permits or denies the incoming order request. If it denies the request, it throws an intermediate error event and *Deny Request* task is activated. If it permits the request, it continues normally and activate the *Accept Request* task. In both directions, the process terminates.

Finally, business analysts can model the security and management policies using our approach. They are easy deployable into a process model definition. It is easy to add new policies or attribute values without adding any constructs to the BPMN diagram. This is because we put all the security and management logic in business rule activities. In this way, we still keep the diagram easily understood without embedding complex constructs, artifacts and fragments.

VI. Conclusion and Future Work

We presented a novel approach for embedding security in business processes in a systematic manner. Therefore, we can provide a complete software solution without the need of post-adhoc security to be considered. We consider data access control as an aspect of security. It is crucial in most business

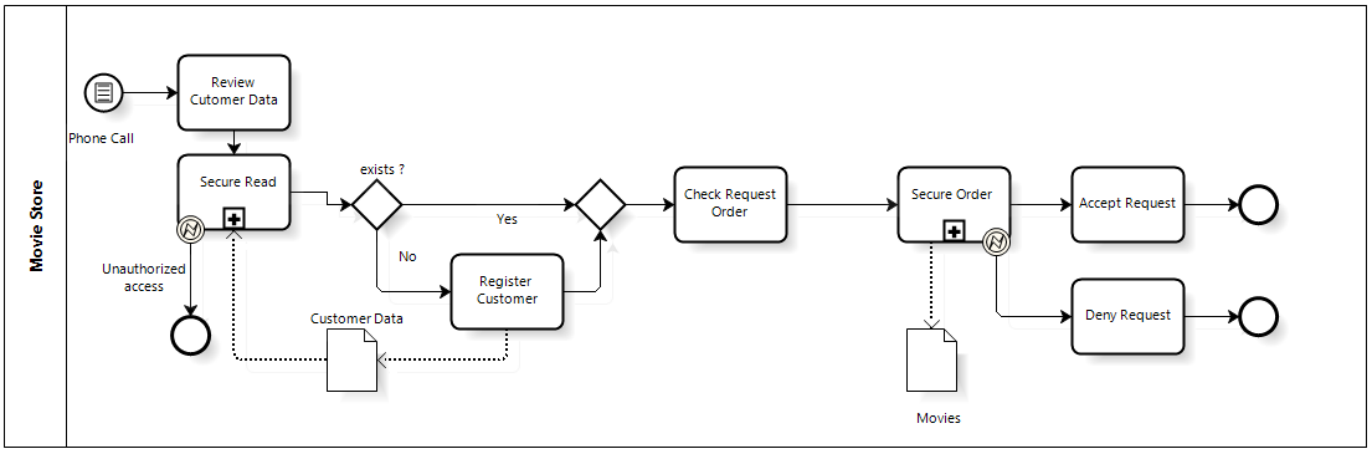

Fig. 4. Movie store example

TABLE VI
BASIC REQUIREMENT DECISION TABLE

Conditions		Result
Age	**Movie Rating**	**Granted**
≥ 21	{R, PG13, P}	Yes
$21 > age \geq 13$	{PG13, P}	Yes
<13	{P}	Yes

TABLE VII
ADVANCED REQUIREMENT DECISION TABLE

Conditions		Result
Member Type	**Movie Type**	**Granted**
Premium	Any	Yes
Regular	Not new released	Yes

processes where business analysts do not know either about access control models or how to add them to their processes while modeling. We used Business Process Modeling Notation (BPMN) as a graphical notation to represent processes. In this approach, we make use of business rule activities of the notation via putting all the security logic to be put as If-Then rules with conflict detection. This comes with minimal overhead for business analysts. Business processes diagrams are still readable and easily understood. We prove the validity of the approach.

Our ongoing work is to consider other aspects of security like confidentiality, integrity, and availability. We plan to transform the new inserted fragments to BPEL in a systematic way in order to provide a complete software solution with early security consideration from the beginning.

REFERENCES

[1] BPMN, Business Process Model and Notation (BPMN). *In http://www.omg.org/spec/BPMN/2.0/PDF/*, 01.7.2012.

[2] C. Ouyang, M. Dumas, A. H. M. ter Hofstede, and W. M. P. van der Aalst, "From BPMN process models to BPEL web services," In *Proceedings of the 4th International Conference on Web Services (ICWS06)*, pp. 285-292, Chicago, Illinois, USA, September 2006. IEEE Computer Society.

[3] J. Recker and J. Mendling, "On the Translation between BPMN and BPEL: Conceptual Mismatch between Process Modeling Languages," In *Proceedings of the 18th International Conference on Advanced Information Systems Engineering*, pp. 521-532, 2006.

[4] J. Blox, "BPMN 2 BPEL," *Eindhoven: Eindhoven University of Technology*, 2009.

[5] M. zur Muehlen and M. Indulska, "Modeling languages for business processes and business rules: A representational analysis," *Information Systems*, pp. 379-390, Elsevier, 2010.

[6] F. Huonder, "Conflict Detection and Resolution of XACML Policies," *Master Thesis, University of Applied Sciences Rapperswil*, 2010.

[7] E. Yuan and J. Tong, "Attributed Based Access Control (ABAC) for Web Services," In *ICWS05: IEEE International Conference on Web Services, Orlando*, pp. 561-569, 2005.

[8] B. Lang, I. Foster, F. Siebenlist, R. Ananthakrishnan, and T. Freeman, "A Flexible Attribute-Based Access Control Method for Grid Computing," *Journal of Grid Computing*, vol. 7, no. 2, pp. 169-180, 2009.

[9] P. Samarati and S. D. C. di Vimercati, "Access Control: Policies, Models, and Mechanisms," In *FOSAD II*, pp. 137-196, Springer-Verlag, 2001.

[10] A. Rodríguez, E. Fernández-Medina, and M. Piattini, "A BPMN Extension for the Modeling of Security Requirements in Business Processes," *IEICE Transactions on Information and Systems*, pp. 745-752, 2007.

[11] M. Menzel, I. Thomas, and C. Meinel, "Security requirements specification in service-oriented business process management," In *ARES*, pp. 41-48, 2009.

[12] C. Wolter, M. Menzel, and C. Meinel, "Modelling security goals in business processes," *Proc. GI Modellierung*, pp. 197-212, 2008.

[13] C. Wolter and A. Schaad, "Modeling of task-based authorization constraints in bpmn," In *BPM*, pp. 64-79, 2007.

[14] J. Mülle, S. von Stackelberg, and K. Bohm, "A Security Language for BPMN Process Models," *Technical Report, Karlsruhe Institute of Technology (KIT)*, 2011.

[15] N. Darnianou, N. Dulay, E. Lupu, and M. Sloman, "The Ponder Policy Specification Language," *International Workshop, Policies for Distributed Systems and Networks*, pp.29-31, 2001.

[16] J. Lobo, R. Bhatia, and S. Naqvi, "A Policy Description Language," *Proc. of National Conference of the American Association for Artificial Intelligence*, pp. 291-298, July 1999.

[17] A. D. Brucker, I. Hang, G. Lückemeyer, and R. Ruparel, "SecureBPMN: Modeling and Enforcing Access Control Requirements in Business Processes," In *ACM Symposium on Access Control Models and Technologies*, pp. 123-126, 2012

[18] J. Mülle, S. von Stackelberg, and K. Bohm, "Modelling and Transforming Security Constraints in Privacy-Aware Business Processes," In *SOCA*. pp. 1-4, 2011.

[19] E. Paja, P. Giorgini, S. Paul, and P. H. Meland "Security Requirements Engineering for Secure Business Processes," In *10th International Conference BIR 2011*, LNBIP, pp. 77-89, 2012.

[20] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. "Computational Geometry, Algorithms and Applications," *Springer*, 1998.