A Modeling Approach supporting Access Control Delegation in a Disaster Management Context

Khaled Gaaloul[1] and Hend*erik* A. Proper[1,2]
[1]Public Research Centre Henri Tudor, Luxembourg
[2]Radboud University Nijmegen, the Netherlands
{khaled.gaaloul,erik.proper}@tudor.lu

***Abstract*—Disaster management can be defined as the organization and management of resources and responsibilities for dealing with all humanitarian aspects of emergencies. This paper is about organizational policies when assigning responsibilities during a flood scenario. Specially, we focus on dynamic access control policies supporting delegation. Delegation is a dynamic behavior involving a user passing his access control authorizations to other users within organizations. This defines one aspect of collaboration at the organizational level.**

In this paper, we propose to model the delegation access control approach from the flooding business process modeling to its deployment. In doing so, we describe the delegation process, define the access control model, and implement the authorization policies within a disaster management framework.

Keywords: Disaster management, Business processes, Access control, Task delegation.

I. Introduction

Coordination between many highly independent organizations is still a challenging topic in disaster management. Strategic coordination in disaster management means that many highly independent organizations are coordinated towards a common goal within a complex process network ([1]). Different highly independent organizations execute and change their plans frequently, and have to interact in an ad-hoc fashion. These interactions between business processes can hardly be predefined and planned.

In this ad-hoc environment, disaster management applications have to deal with exceptions which are difficult to foresee when modeling inter-organizational processes. One specific approach for exceptions handling is that of task delegation [2] within organizations. An important requirement of task delegation is to support organizational flexibility for governing emergent process planning and enactment in the disaster response. Such a requirement implies dynamic authorization when delegation access control rights, and has to be integrated following the business/IT alignment of the organization.

Enormous amounts of data flow along business processes and are shared by many different users. Protecting application data through access control policies has been widely discussed ([3], [4], [5]). Sandhu et al. proposed a series of access control models ([6]): RBAC96 models where the central idea is that access rights are associated with roles, to which users are assigned in order to get appropriate authorizations. Chadwick et al. ([7]) investigated how an authorization management system based on the eXtensible Access Control Markup Language (XACML) can be extended to support delegation mechanisms. Authors focused on the administration side of the authorization policy while overriding access control in XACML. Administrating delegation policies is, however, stateless and lacks of reactivity to support policy's change when delegating a task in a business process.

The contribution of the paper is to model the delegation access control approach using a flooding simulation scenario, then, to integrate the delegation process within a disaster management framework. The idea is to model the delegation interactions and to leverage this model's specifications (i.e. concepts, relations) to derive security requirements supporting delegation in the main framework. To that end, we model the flooding process and identify the delegation requirements. This step will be important when delegating resources and so expressing access control over task delegation. Once the delegation process modeled and secured, its concepts and relations have to interact with the overall picture of the disaster management framework. This final step is about delegation policies (authorization policies) deployment within the framework.

The remainder of this paper is organized as follows. Section 2 presents a disaster management case study and discuss delegation motivations and constraints. In section 3 we model a task-based access control model supporting secure delegation. Section 4 presents the integration of the delegation process within the crisis management framework. Section 5 presents related work. In section 6 we conclude and discuss future work.

II. Context and problem statements

To understand the motivation of our research, we present a chemical plant protection scenario from the SoKNOS project[1]. A Fire brigade unit will need assistance to isolate a contaminated zone and protect civilians. They will require geographical analysis of current situations by external partners. The scenario is defined using a BPMN model (see figure 1). A fire brigade command center *Si* needs information regarding expected flooding simulation from experts at the University. *Si* issues a request for assistance to the expert *Ei*. The request

[1]More information on the publicly funded SoKNOS project can be found at: http://www.soknos.de

978-1-4799-4249-7/14 $31.00 © 2014 IEEE
DOI 10.1109/WETICE.2014.37

IEEE computer society

Authorized licensed use limited to: Radboud University Nijmegen. Downloaded on April 23,2021 at 07:40:04 UTC from IEEE Xplore. Restrictions apply.

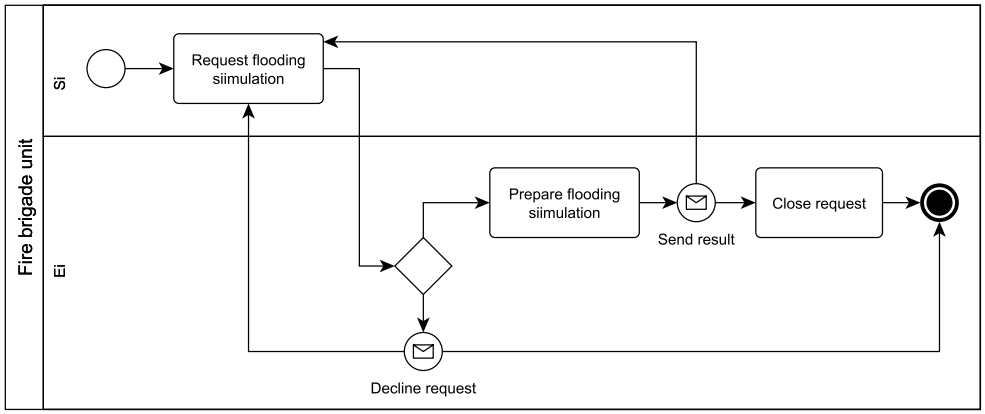

Fig. 1. A business process for flood simulation

message includes information related to the situation, a link to the geographical resources including respective rights and time constraints (e.g. 30 minutes for a confirmation reminder). The Message Engine will send the request. *Ei* confirms the request acceptance and returns results. In addition, *Ei* informs about the end of execution and resources will be revoked automatically.

In this scenario, the task "Prepare flooding simulation" is assigned to the user *Ei* with the rights to access task's resources (e.g. geographical resources). These rights define access control permissions as facts defined in the authorization policy. Facing an unexpected situation where *Ei* is unavailable to execute this task. This expert decides to delegate the task "Prepare flooding simulation" to the user at the command center *Si*. The policy is updated so that user *Si* is now allowed to access and complete this task. As such, users *Ei* and *Si* are here the delegator and the delegatee, respectively. User *Si* claims the task, and issues an access control request, is granted access, and executes the task.

In traditional access control frameworks however, no mechanism support task delegation requirements ([8], [5]). At present, we can enforce delegation access rights via policy adaptation (i.e. permitting the delegatee to perform the delegated tasks). If however, the delegation request does not meet task assignment requirements, it is difficult to foresee it in the policy. Existing models in which users get permissions through roles do not permits users to get permissions from tasks and so ignore task context awareness. This inquires the need to support specific interactions and the access control architecture that they run on. Specific interactions are meant to be delegation constraints over access control models.

Returning to the example, with a constrained access control model for delegation, User *Si* would be automatically verified based on task's resources requirements. His required authorization (permissions) against task assignment request will be computed and validated for authentication and authorization purposes. Moreover, users are granted privileges just at the start of tasks, which are revoked as soon as the tasks are finished. So privilege leakage caused by granting permission too early or revoking permission too late have to be avoided. Hence, securing delegation requires the integration of a task-based access control model within the existing framework while taking into account the business process specification.

In the following section, we motivate the need of an access model supporting task delegation and explain how such a model interacts with existing components in SoKNOS framework.

III. A TASK-BASED ACCESS CONTROL MODEL

We propose a task-based access control model to support authorization requirements. Authorization information will be inferred from access control data structures, such as user-role assignment and task-role assignment relations ([6], [9]). We leverage the different task requirements regarding human and material resources and model it in a set of relationships building our model (see figure 2).

A. Model definition

Formally, we define sets U, R, OU, T, P, S and TI as a set of users, roles, organizations units, tasks, permissions, subjects and task instances respectively.

Definition 1: We define *P* is a set of permissions. A permission *p* is a pair *(f,o)* where *f* is a function and *o* is a business object: p $\subseteq$ f $\times$ o.
P defines the right to execute an operation on a resource type.

Definition 2: We define RH (Role Hierarchy), where RH is a partial order on R. $(r_i, r_j) \in R$, RH denotes that r_i is a role superior to r_j, as a result, r_i automatically inherits the permissions of r_j.

Definition 3: We define RM (Role Mapping), where RM is a partial order on R belonging to a set of roles defined in the involved organizations hierarchies (OU), where:
$r_k \in OU_k$ and $r_l \in OU_l$, RM denotes that r_l is a role mapped to r_k, as a result, r_l automatically inherits the permissions of r_k.

Authorized licensed use limited to: Radboud University Nijmegen. Downloaded on April 23,2021 at 07:40:04 UTC from IEEE Xplore. Restrictions apply.

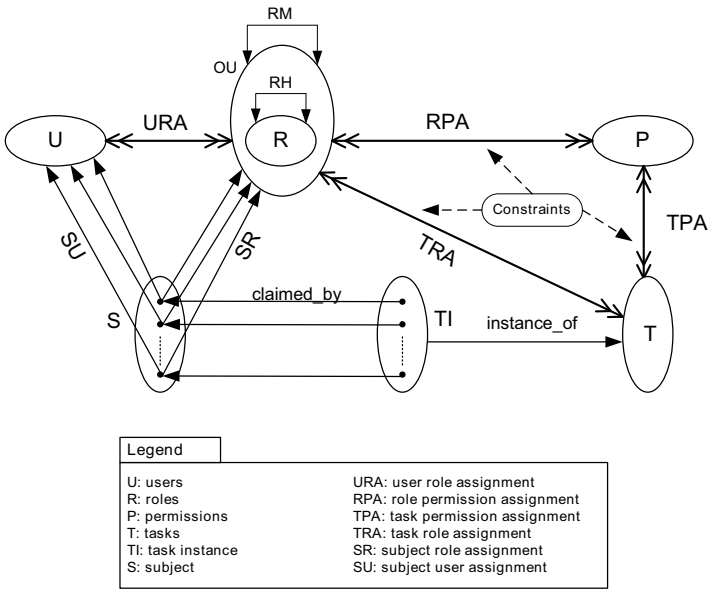

Fig. 2. Task-based access control model

RM defines external roles accessing distributed resources cross-organizations.

Definitions of Map Relations:

- $URA \subseteq U \times R$, the user role assignment relation mapping users to roles they are member of.
- $RPA \subseteq P \times R$, the permission role assignment relation mapping roles to permissions they are authorized to.
- $TPA \subseteq T \times P$, the task permission relation mapping tasks to permissions. This defines the set of permission required to execute a task.
- $TRA \subseteq T \times R$ the task role assignment relation mapping roles to tasks they are assigned to.

Definitions of Functions:

- $SU\colon S \rightarrow U$ a function mapping a subject to the corresponding user.
- $SR\colon S \rightarrow 2^R$, a function mapping each subject to a set of roles, where $SR(s_i) \subseteq \{r|(SU(s_i), r) \in URA\}$ and subject s_i has the permissions; $\cup_{\{r \in SR(s_i)\}}\{p|(p, r) \in RPA\}$.
- $instance_of\colon TI \rightarrow T$, a function mapping a task instance to its task type.
- $claimed_by\colon TI \rightarrow S$, a function mapping a task instance to a subject to execute it, where: $claimed_by(t_i, s_i) = \{t_i|instance_of(t_i, t), (r, u) \in URA|(SR(s_i) = r \bigwedge SU(s_i) = u), (t, r) \in TRA\}$.

Definitions of Constraints:

Here we discuss Separation of duty (SoD) and Binding of duty (BoD) constraints. We define exclusive relation between tasks for SoD, and binding relation between tasks for BoD as follows:

$TT_{SOD} = \{(t_i, t_j) \in T|t_i$ *is Exclusive with* $t_j\} \subseteq$ TxT
$TT_{BOD} = \{(t_i, t_j) \in T|t_i$ *is Binding with* $t_j\} \subseteq$ TxT, where $t_i \leq t_j$.
If $(t_1, t_2) \in TT_{SOD}$, then t_1 and t_2 cannot be assigned to the same subject, and if $(t_1, t_2) \in TT_{BOD}$, then t_1 and t_2 must be assigned to the same subject.

B. Task Assignment Conditions

We model permission assignment relations for task and role in order to support both human and material resources. *(P,T,R)* specifies TRA, TPA and RPA many-to-many relationships which are specifics to the task execution context. The remaining relations are generic relations based on the role-based access control model: RBAC ([6]).

Definition 4: A task can only be assigned to a role if and only if: $(t, r) \in TRA \Rightarrow \{p \in P|(t, p) \in TPA\} \subset \{p|(p, r) \in RPA\}$.

The main contribution is to specify the task assignment conditions based on the RPA and TPA requirements (see Definition 3). Two conditions have to be verified to satisfy the TRA relation. The first condition is related to task resources requirements. The user's permissions defined in RPA need to satisfy the permissions defined in TPA. If this condition is satisfied, the task is executed if and only if the user is assigned to it.

C. Delegation Constraints

Task delegation is aligned with the task assignment conditions (see Definition 4). We remind that the user who performs a delegation is referred to as a *delegator* and the user who receives a delegation is referred to as a *delegatee*.

Authorized licensed use limited to: Radboud University Nijmegen. Downloaded on April 23,2021 at 07:40:04 UTC from IEEE Xplore. Restrictions apply.

Definition 5: We define a task delegation relation *RD = (T,u_1,u_2,C)*, where *T* is the delegated task, u_1 the delegator, u_2 the delegatee, and *C* the delegation constraints. Constraints refer to the condition of delegating accordingly to the global policy.

We provide an optimized method to compute the delegated privileges based on the current requirements of the task instances (resources requirements). The aforementioned access control model (see figure 2) defines the list of potential delegatees (RPA) that may satisfy the delegated task requirements (TPA). For instance, u_1 and u_2 are members of roles r_1 and r_2 respectively:
(t,u_1,u_2,C) $\in$ *RD iff* $(t, r_2) \in TRA \Rightarrow \{p \in P|(t, p) \in TPA\} \subset \{p|(p, r_2) \in RPA\}$.

Returning to the example, the task "Prepare flooding simulation" is assigned a set of permissions (e.g. *query(), update()*) via the TPA relation in order to carry out this task. Once this task is claimed, TRA is assigned to roles that are authorized to claim it. If a different user from another command center defined in the list of the external partners, the access control model will enforce another access constraints to see whether this user belongs to the TRA set and so he will be assigned in the authorization policy of the business process.

IV. Integrating delegation within SoKNOS framework

In this section, we integrate the delegation process within SoKNOS framework. We develop a delegation component interacting with the existing components when issuing a delegation request. We present a delegation protocol that depicts the dialogue between a delegator and his corresponding delegatee under the specific constraints defined in section 3.C. We model the protocol using UML sequence diagrams and colored in grey the main components supporting the delegation process (see figure 3).

The integration context of our delegation prototype is composed of two main components namely the Delegation Component (DC) and the external Delegation Access Point (DAP) acting respectively as back-end and proxy component. Furthermore, we develop a fine grained access control solution to support delegation. The Authorization Component supports policy decision-making based on the access control mechanism defined in section 3. Furthermore, we need to authenticate and then to authorize the requester, in this case the delegatee, who will accept and perform the request.

The Access Control Enforcement (ACE) component checks the delegatee credentials. Once authenticated, rights are issued to the delegatee and access request authorization is computed in the the Policy Decision Point (PDP) engine. ACE component implements the Central Authentication Service (CAS). It is a single sign-on protocol for the web, its role is to permit a user to access multiple applications while providing their credentials (such as user ID and password) only once. The PDP component is responsible for evaluating access decision requests and returning a respective access decision response (i.e. accept, deny). PDP engine implements the Java core class for the access control engine, providing the starting point for request evaluation.

We briefly detail the main blocks of the SoKNOS framework integration as depicted in figure 3:

- **Block 1**: We determine the first interaction with the existing components. The delegator is accessing the Geographical toolbox in order to select the required resources to be delegated (see section II). The Geographical Information (GI) system includes the GI component and the GI Plugin that will set the URL link to be attached to the request. Steps 1 to 4 summarize the first block of our diagram.
- **Block 2**: The DC will secure the request. To that end, DC creates a secure URL (SecURL) and the required credentials in order to open the request. This step is very important in the delegation process since it restricts the access to the delegated resource from undesirable users (see Definition 3 in section 3). This part is supported using the ACE component. Delegation request is then sent using the Message Plug-in of the main SoKNOS architecture. Steps 5 to 10 summarize the second block of our diagram.
- **Block 3**: The second delegation component (DAP) will act as a proxy with the GI component. At this stage, the request is accepted by the delegatee and an authentication of the requester and a computing of his delegated privileges need to be checked. This part is supported using the PDP component (steps 11-17). Additional steps regarding the tracing (the log file) and the forwarding of the request execution (the GI component) in order to grant or reject the authorization are developed in the delegation protocol (steps 18-21).

Securing delegation architecture does not require major modifications of the existing architecture components such as the ACE and PDP components. It uses the different models specifications such as the business process model, the access control model, and the SoKNOS sequence diagram.

Note that revocation is an important process that must accompany the delegation. It is the subsequent withdrawal of previously delegated task. For simplification, our model of revocation is related to the delegation model where a revocation is issued from the delegator in order to cancel/end the delegated privileges.

V. Related Work

Sandhu et al. proposed a series of access control models [6], [5]: RBAC96 models. Access rights are associated with roles, to which users are assigned in order to get appropriate authorizations. It also involves the role hierarchy that enables the permission heritage. Since the roles in organizations are relatively stable and the number of roles is much more smaller than that of users, the work of administrators can be greatly relieved by applying the concept of roles. Thus it is more adaptable to dynamic environments to a certain extent.

Authorized licensed use limited to: Radboud University Nijmegen. Downloaded on April 23,2021 at 07:40:04 UTC from IEEE Xplore. Restrictions apply.

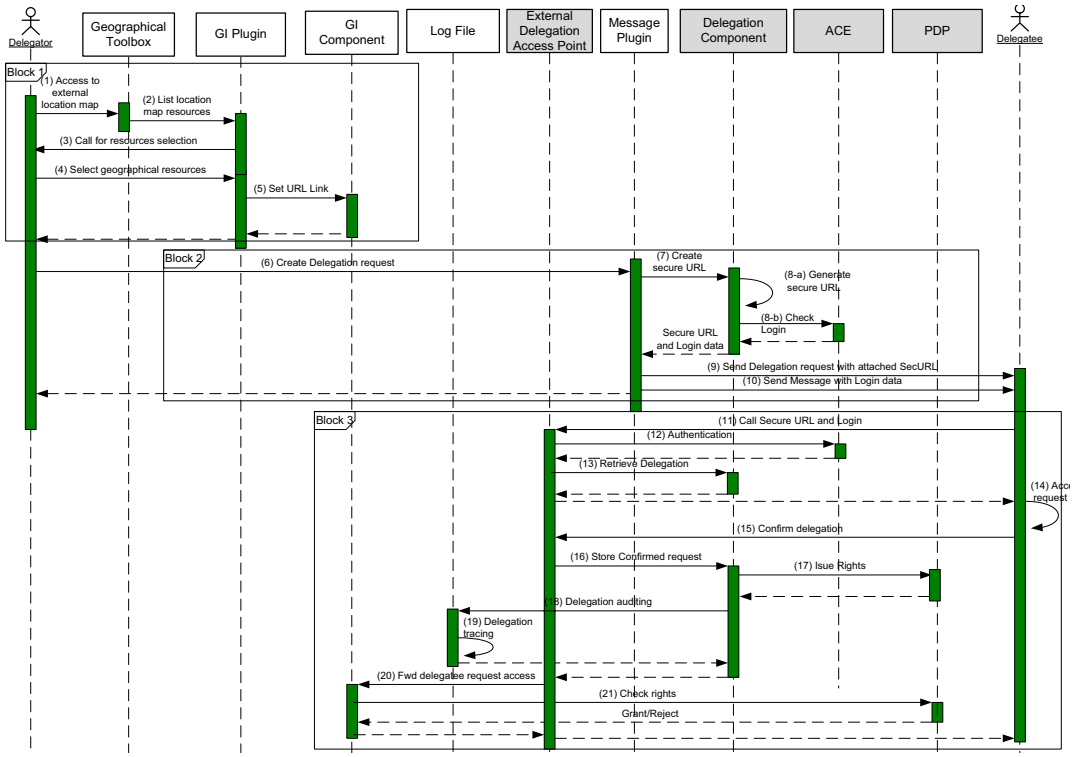

Fig. 3. Delegation integration within SoKNOS

However, there is no concept of tasks in RBAC, which makes it difficult to support task delegation process [10].

There exists several attempts to support delegation in access control models. In [5], [11], authors extend the RBAC model by defining some delegation's rules. Zhang et al. proposed a flexible delegation model named Permission-based Delegation Model (PBDM). However, PBDM supports only role-to-role delegation, thereby ignoring task delegation constraints.

The eXtensible Access Control Markup Language (XACML) was developed in order to provide a uniform way of specifying access control policies in XML [4]. Policies comprising Rules, possibly restricted by Conditions, may be specified and targeted at Resources, Subjects and Actions. Seitz et al. [12] investigated how an authorization management system based on XACML can be extended to use flexible delegation mechanisms. They developed a separate policy administration point component that specifies allowed modifications on different elements of an XACML policy for different users. Authors focused on the administration side of the policy while overriding access control in XACML. Administrating delegation policies remain, however, stateless to support dynamic authorization when delegating task.

VI. Conclusion

In this paper, we have analyzed the need of a delegation mechanism for governing emergent process planning and enactment exceptions in the disaster framework. The analysis depends on the roles of different components of the framework. These components are described via models to express their business interpretations. In doing so, we have represented the different models involved in the delegation process: from the business model (BPMN) to the access control model (RBAC). These models interactions (UML sequence diagram) aim to ensure the organizational flexibility and the dynamic authorization thereby supporting business/IT alignment within the disaster management system SoKNOS.

Future work will look also at extending our approach to larger organizations views such as enterprise architecture models. The goal is to illustrate the interrelationship of enterprise business, information, and technology environments. In addition, the assessment of such alignment when integrating new requirements such as auditing.

References

[1] T. Drabek, *Strategies for Coordinating Disaster Responses*, ser. Program on Environment and Behavior. Institute of Behavior Sciences, 2003. [Online]. Available: http://books.google.lu/books?id=psaYAAAACAAJ

[2] K. Gaaloul, H. Proper, E. Zahoor, F. o. Charoy, and C. Godart, "A logical framework for reasoning about delegation policies in workflow management systems," *International Journal of Information and Computer Security*, vol. 4, no. 4, pp. 365–388, 2011.

[3] P. Rao, D. Lin, E. Bertino, N. Li, and J. Lobo, "Fine-grained integration of access control policies," *Computers & Security*, vol. 30, no. 2-3, pp. 91–107, 2011.

[4] XACML-V3.0, "eXtensible Access Control Markup Language (XACML v3.0), note = Standard, Organization for the Advancement of Structured Information Standards (OASIS): http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf," 2013.

Authorized licensed use limited to: Radboud University Nijmegen. Downloaded on April 23,2021 at 07:40:04 UTC from IEEE Xplore. Restrictions apply.

[5] X. Zhang, S. Oh, and R. Sandhu, "PBDM: a flexible delegation model in RBAC," in *SACMAT '03: Proceedings of the eighth ACM symposium on Access control models and technologies*. New York, NY, USA: ACM Press, 2003, pp. 149–157.

[6] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models," *IEEE Computer*, vol. 29, no. 2, pp. 38–47, 1996.

[7] D. W. Chadwick, S. Otenko, and T.-A. Nguyen, "Adding support to xacml for multi-domain user to user dynamic delegation of authority," *Int. J. Inf. Sec.*, vol. 8, no. 2, pp. 137–152, 2009.

[8] K. Gaaloul, E. Zahoor, F. Charoy, and C. Godart, "Dynamic authorisation policies for event-based task delegation," in *Advanced Information Systems Engineering, 22nd International Conference, CAiSE 2010, Hammamet, Tunisia*, 2010, pp. 135–149.

[9] X. Liao, L. Zhang, and S. C. F. Chan, "A task-oriented access control model for wfms," in *Proceedings of the First international conference on Information Security Practice and Experience*, ser. ISPEC'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 168–177. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-31979-5_15

[10] K. Gaaloul, "A Secure Framework for Dynamic Task Delegation in Workflow Management Systems. Ph.D. thesis, The University of Henri Poincaré, Nancy, France," 2010.

[11] L. Zhang, G.-J. Ahn, and B.-T. Chu, "A rule-based framework for role-based delegation and revocation," *ACM Transactions on Information and System Security*, vol. 6, no. 3, pp. 404–441, 2003.

[12] L. Seitz, E. Rissanen, T. Sandholm, B. S. Firozabadi, and O. Mulmo, "Policy administration control and delegation using xacml and delegent," in *6th IEEE/ACM International Conference on Grid Computing (GRID 2005), November 13-14, 2005, Seattle, Washington, USA, Proceedings*, 2005, pp. 49–54.

Authorized licensed use limited to: Radboud University Nijmegen. Downloaded on April 23,2021 at 07:40:04 UTC from IEEE Xplore. Restrictions apply.