Delegation Protocols in Human-Centric Workflows

Khaled Gaaloul and H.A. (Erik) Proper
CRP Henri Tudor
L-1855 Luxembourg-Kirchberg, Luxembourg
{khaled.gaaloul,erik.proper}@tudor.lu

François Charoy
LORIA - Nancy University - UMR 7503
BP 239, F-54506 Vandœuvre-lès-Nancy Cedex, France
charoy@loria.fr

***Abstract*—Organisations are facilitated and conducted using workflow management systems. Currently, we observe a tendency moving away from strict workflow modelling towards dynamic approaches supporting *human interactions* when deploying a workflow. One specific approach ensuring human-centric workflows is *task delegation*. Delegating a task may require an access to specific and potentially sensitive data that have to be secured and specified into authorisation policies.**
In this paper, we propose a modelling approach to secure delegation. In doing so, we define delegation *protocols* supporting specific constraints based on both workflow and access control systems. Moreover, we develop an advanced access control framework to integrate delegation constraints within existing policies. The novelty consists in the proactivity aspect of our framework to cope with dynamic delegation of authority in authorisation policies.

Keywords: Workflow, Delegation, Access Control, Policy.

I. Introduction

In the classical software engineering approach the organisational context and related security requirements are often considered during the design phase of a workflow, but internal controls are later defined and implemented in a manual fashion disjoint from predefined models [1]. Workflows model and control the execution of business processes in an organisation. Internal controls depend on the organisational aspect to support human-centric interactions and to analyse resources specifications. Dealing with that, organisations establish a set of authorisation policies that regulate how processes and resources should be managed within a workflow [2].

Moreover, workflow systems are determined by a mix of ad-hoc as well as human-centric processes. Human interactions are also related with the environment in which organisations cooperate dynamically, but are strictly constrained by regulations and policies at the same time [3]. This highly dynamic environment must be supported by mechanisms allowing flexibility, security and on-the-fly shift of rights and responsibilities both on a (atomic) task level and on a (global) process level [4]. One specific set of mechanisms ensuring human-centric interactions is that of task delegation.

We define task delegation as a means of assigning tasks and its access rights *(privileges)* from one user to another user. 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". Delegation can be very useful for real-world situations where a user who has to perform a task is either unavailable or too overloaded with work to successfully complete it. It is frequently the case that delaying these tasks executions would violate time constraints, thereby impairing the entire process execution. Therefore, delegation works on ensuring flexible execution to cope with unavailable workflow's actors and the organisation rigidity. Additionally, we aim to ensure a user-to-user delegation with heavily human interactions. To that end, we need to support additional requirements related to the organisation flexibility and the authorisation policy definition when granting privileges to access delegated resources.

Delegating a task consists of the definition of a user-to-user dialogue to issue a delegation request. Here, human interactions depict delegation protocols involving the main principals, the delegated task and eventually the granted privileges to access task's resources. In addition, we have to consider additional constraints with regards to the process definition and the authorisation policy. In doing so, we have to come up with a new modelling approach that bridge the gap between workflow (process) and access control (policy) systems. Further, we leverage our approach to align the security requirements for delegation within secure workflows. Moreover, we develop an advanced access control framework supporting the dynamic aspect of authority when delegating privileges.

The remainder of this article is organised as follows. Section 2 presents fundamental concepts of the organisational management and security in workflows. In section 3, we motivate our work with an eGovernmental scenario supporting task delegation. Section 4 presents our contribution in terms of delegation protocols ensuring dynamic and secure human interactions within a workflow. In section 5, we develop an access control framework to specify delegated privileges into authorisation policies. In Section 6, we conclude our approach and outline future work.

II. Background

The fundamental concepts of this work are related to the organisational management as well as the security enforcement in workflow systems. We aim to address issues related to the organisational needs and security requirements for modelling and securing task delegation.

A. Organisational management in workflows

In the context of heavily human-centric workflows, business processes are determined by a mix of ad-hoc as well as process-based interactions. We currently observed, when substituting members of an organisation during workflow deployment, a move away from the predefined modelling workflow towards approaches supporting flexibility on the organisational level. This highly dynamic environment must be supported by mechanisms allowing the monitoring, secure and on-the-fly shift of rights with respect to an ongoing human interactions both on a (atomic) task level and on a (global) process level [5].

The requirements for interactions and monitoring can be summarised as transparency and control [6]. Transparency addresses the revelation of the control flow dependencies. This allows to react accordingly to exceptions and compensations during execution. Control fosters the behaviour of organisations according to their defined policies. One type of transparency and control supporting mechanism in human-centric workflows is that of task delegation. We do believe that user-to-user delegation is a suitable approach to handle such interactions.

During the design time, the workflow application designer has to design both the structure of the business process to be automated, and the structure of the resources that carry out the process [3]. Resources and workflow's tasks are linked through the construct role. From a process perspective, a role is a subject to authorisations that define permissions (operations) for the execution of a task. From a resource perspective, a role represents a granted authorisation for a workflow's actor (so-called user). In this paper, we aim to address issues related to the user's assignments and resource's access for the organisational management in workflow systems.

B. Access control over workflows

Typically, organisations establish a set of security policies, that regulate how business processes and resources should be managed within a workflow. A security policy defines the expected standard of security enforcement using access control mechanisms [7]. An access control has to be defined to check the authorisation of the initiated user (the *subject*).

We present an access control framework (ACF) to support authorisation policies within workflows. ACF is mostly based on the eXtensible Access Control Markup Language (XACML) specifications [8]. The main components are described as follows:

- **Policy Manager**: It allows an administrator to define policies. Through a graphical user interface, the administrator can navigate through the policy document, select document elements (e.g., targets, authorisation rules, obligations), and specify values for selected elements.
- **ACF**: An access control framework (ACF) is defined as a set of software components which accept requests to access resources, analyse these against policies representing actual access rights to resources, and return a response based on this analysis. To illustrate the original architecture of an ACF, a request is issued by the requestor, which is received by the *Receiver* component in ACF. This is then sent to the *Analyser* component that queries policies stored in a policy database. A response is generated by the *Responder* component, which defines a decision (permit, deny, or not applicable) that is sent back to the requestor. It should be noted, that the above appears asynchronous for the requestor; they provide the request, and a response is produced.

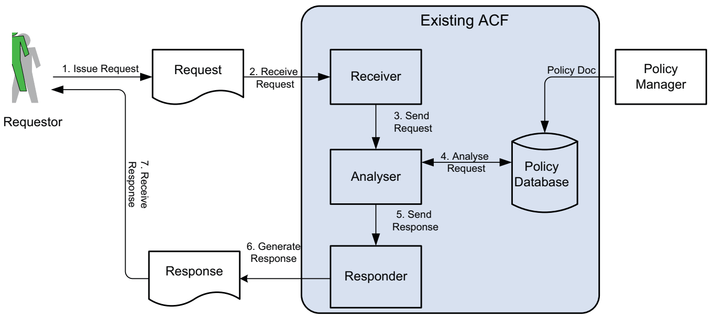

Fig. 1. Access control framework

Authorisation policies are predefined in the access control framework. Significant contributions to ensure dynamic policy enforcement can be found in [9]. While much of the work is limited to role-based access control, the goal of our paper is to consider task delegation constraints in workflow systems. Delegation constraints needs to tackle several issues with regards to workflow's invariants in terms of users, tasks and resources. Moreover, we need to support dynamic changes of authority when cancelling/revoking delegation and its granted privileges.

III. Delegation in eGovernment Context

To understand the motivation of our research, we present a scenario from an eGovernment case study supporting delegation [5]. Mutual Legal Assistance (MLA) defines a workflow scenario involving national authorities of two European countries regarding the execution of measures for witness protection in a criminal proceeding. Here we describe the MLA process part in the Eurojust organisation A. Users with roles *Prosecutor* and *Assistant* are assigned to execute activities of the MLA process which are represented as tasks (see Fig. 2).

A. Task delegation scenario

The MLA process consists of receiving the request of assistance from Europol member in order to process it and send it to the concerned authority in Eurojust B. The task "Translate Documents" T3 is originally only accessible (read/write) by the user Alice member of role *Prosecutor*, a fact defined in the workflow security policy. We define a workflow policy as a means for defining access rights to a task's resources also called *authorisation policy*.

Let *P* an authorisation policy for the MLA process. Task T3 is a long-running task and is expected to take 5 working days to complete. The *Prosecutor* Alice is unavailable to execute this task due to illness, and will delegate it to a subordinate

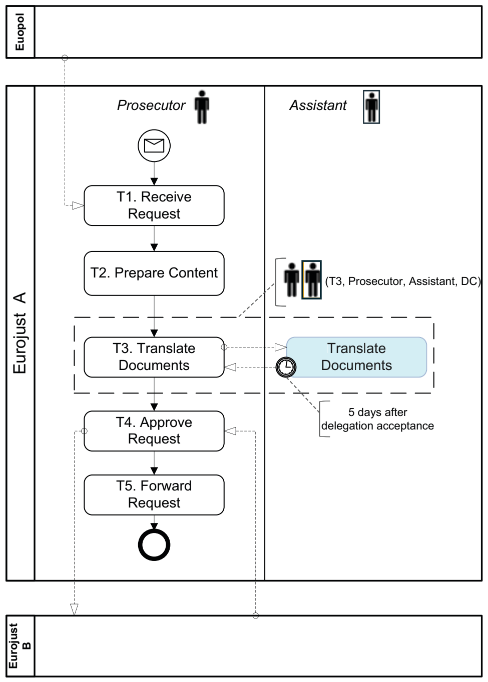

Fig. 2. MLA delegation scenario

involved in the MLA process. *Assistant* is a subordinate to *Prosecutor* in the organisational role hierarchy.

During delegation, the policy P is updated so that user Bob with role *Assistant* is now allowed to complete task T3. The policy P will grant an access right to Bob to execute the task T3 and therefore to access task's resources (e.g., legal documents). As such, users with roles *Prosecutor* and *Assistant* are here the delegator and the delegatee, respectively.

In the meanwhile, Alice interrupts her sick leave and returns to work. Once again, Alice is able to claim T3. Due to qualification considerations, it is decided that Alice should complete the task, and that her *Assistant* Bob should revoke his actions, and free the task. The policy P needs to be updated to reflect that only Alice has access to the task. As such, the grant access to Bob would now evaluate to a deny decision which has to be integrated dynamically in the existing policy.

B. Problem statements

The authorisation policy P needs to reflect the new requirements for delegation. In order to derive a delegation policy from the existing policy, we have to specify additional authorisation rules to support delegation, where a rule defines the policy decision effect (e.g., permit, deny). Delegation rules depend on several delegation constraints such as time (i.e., 5 days) and have to be included in the delegation policy to define specific conditions to validate the policy decision effect.

Delegation constraints define the delegation behaviour within the business process. This behaviour interprets human interactions when deploying a workflow. Accordingly, it is not possible to foresee a deny rule for revocation during the policy definition. Moreover, a manual review of the current access control rights and task executions is costly, labor intensive, and prone to errors.

Based on the delegation scenario, we aim to address two important issues related to task delegation in workflows. At the *organisational level*, we have to identify the list of potential delegatees having the ability to execute the delegated task. At the *security level*, we have to decide how to ensure the delegation of authority to a delegatee to access task's resources. The delegation of authority has to be defined in the access control system. It expresses new delegation policies enforcement defined in a dynamic manner and integrated in the existing policy. Finally, we have to take into account how delegation requests are issued.

Crampton et al. [10] identified two modes to issue a delegation request. The *pull* mode assumes that a delegator has at his disposal a pool of delegatees to be selected to work on his behalf. The *push* mode assumes that a delegator is waiting for an acceptance from a potential delegatee. In this paper, we consider both modes and explain how we model them in a secure manner within workflows.

IV. Modelling and Securing Delegation

Delegation defines a mechanism to support user-to-user interactions. We introduce delegation protocols to support both *push* and *pull* modes. Delegation protocols define two different models that depict the dialogue between a delegator and a delegatee. These protocols will ensure the delegation of authority in access control systems.

A. Security alignment

We aim to address the modelling issue of delegation since we have to cope with the security constraints when granting new rights (privileges) and enforcing new policies decisions. In doing so, we align the security requirements for delegation with the workflow specification. To that end, we have to consider the delegated task status and the delegatee's worklist. The delegated task has to be managed by a specific component to control its states (e.g., starting, execution, cancellation). The worklist defines a list of work items (tasks) associated with a given workflow participant (users) [11].

Our approach is aligned with the fundamental concepts introduced in Sect. 2. In others words, our modelling technique will bridge the gap between workflow and access control systems. We leverage the ACF mechanism as the main Authorisation Component (AC) in the delegation protocols to support delegation policies. Some of these policies may contain dynamic constraints depending on the current task state and the process history. To get this information the decision point will ask a workflow component for the current process state. This component is the Task Service Manager (TSM) that retrieves all relevant state information and returns it to the authorisation decision point in the AC component. For instance, if the task is already assigned and being executed, no further assignment will be allowed.

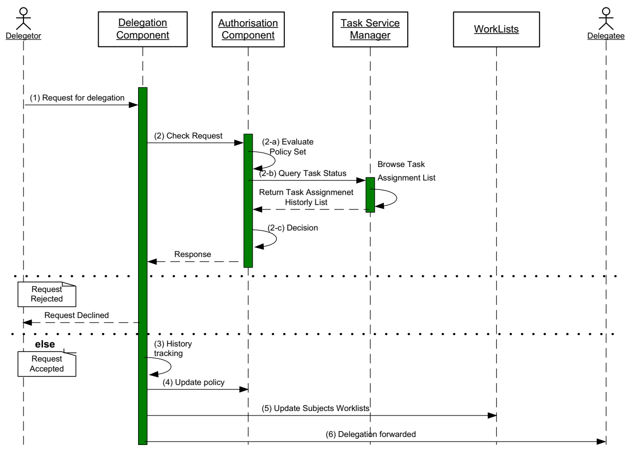

Fig. 3. Task delegation pull model

In the following, we model delegation protocols using a sequence diagram in Unified Modelling Language (UML). It is a kind of interaction diagram that shows how actors, workflow's components and access control system operate with one another and in what order.

B. Pull delegation

We define a subject as an assigned user who is member of a role to claim a task instance. Then the task is added to the subject worklist. Here, the pull delegation model is based on a direct allocation of the task through a delegation without any notion of role. This model associates implicitly an authorisation to the selected subject (see Fig. 3). When a subject holding a task initiates a delegation request, then the following procedure manages it:

1) First the delegator is sending a request for delegation to the Delegation Component for a specific task (delegated task) and a specific subject (the delegatee).
2) The Delegation Component checks with the help of the Authorisation Component (AC) if the delegator can delegate and the delegatee can receive the request.
    - a) The AC first retrieves the attributes affecting the policy and conducts an initial evaluation regarding the delegator's right to delegate. This is due to the fact that certain task assignments are exclusive and are not allowed to be delegated. In the context of an access control policy, it is defined as an obligation to a rule effect (e.g. accept, deny) [8].
    - b) The AC checks then the task status with the Task Service Manager (TSM) component which browses the current task assignment list to check the availability of the task (e.g. executed, aborted).
    - c) The AC receives the history list from TSM. Finally, the AC sends a response to the delegation component based on the intermediate results received.
3) The Delegation Component then keeps track of the current delegation within internal history records.
4) The delegation component updates the appropriate policy in the policy repository.
5) The delegation component updates the appropriate worklists (delegator and delegatee).
6) The delegation request is forwarded to the corresponding delegatee.

C. Push delegation

The push model is based on an allocation of the task through a delegation to a role and not directly to a subject (see Fig. 4). When a subject holding a task initiates a delegation request, then the following procedure manages it:

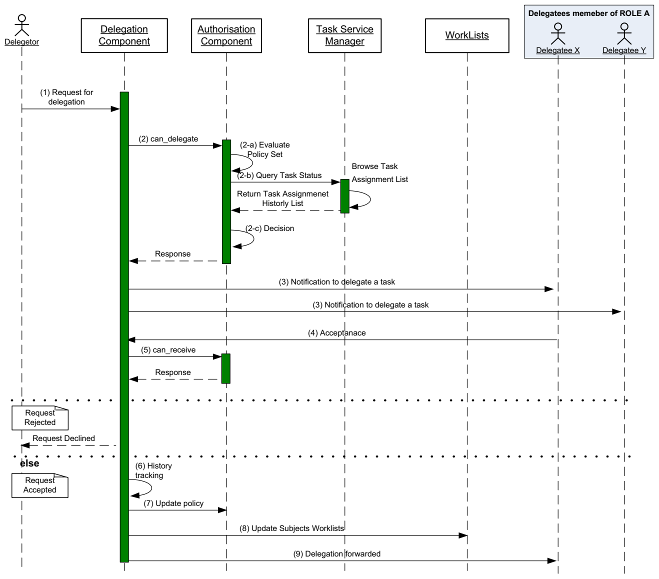

Fig. 4. Task delegation push model

1) First the delegator is sending a request for delegation to the delegation component for a specific task and a specific role (Role A).
2) The delegation component checks with the help of the Authorisation Component (AC) if the delegator can actually delegate based on his policy attributes, then with the task service manager regarding the delegated task status.
3) The delegation component notifies all the subjects belonging to the role (Role A) of the availability of the task.
4) The first one to respond is allocated with the task.
5) The delegation component checks with the help of the AC if the delegatee can actually receive the task.
6) The delegation component then keeps track of the current delegation within internal history records.
7) The delegation component updates the appropriate policy in the policy repository.
8) The delegation component updates the appropriate worklists (delegator and delegatee).

9) The delegation is forwarded to the corresponding delegatee.

D. Access control enforcement

The access control component needs to express the associated authorisation for delegation and update it in the existing policy.

- In the pull model, the delegatee is not granted with a new role. The delegation request does not need further control of the permission; the access control model handles normally a delegated task and does not need to be modified. There is no need of granting additional privileges for the delegatee.
- In the push model, the link of the authorisation with the role is kept. It allows us to reuse the established role-based access control model [12].

The pull model answers the requirements defined in the delegation scenario in the MLA process. User Bob exists in the delegatees list of the delegator Alice and is directly assigned to the task T3. In this case, Bob as a delegatee can claim achieving the task as he is provided with the same access rights (privileges) as the previous owner of the task (the delegator Alice) to execute an instance of the task T3.

Moreover, the access control enforcement has to take into account new changes to be integrated in the existing policy afterwards. Policy changes are related to delegation constraints during task execution. Such constraints depend on user-to-user interactions when executing a delegated task. For instance, Alice is back to work before T3 delegation is done. Alice asks Bob to revoke his work. Subsequently, the policy *P* has to change dynamically the previous policy decision (permit) for the delegatee Bob, thereby cancelling his work and assigning back T3 to user Alice. This dynamic policy enforcement is discussed in the next section.

V. A Secure Framework for Task Delegation

We present a modular architecture ensuring dynamic delegation of authority and show how delegation policy decisions will be implemented on the existing access control framework (ACF). In the context of delegation, when a request is issued, it is stored along with details of how to inform the requestor (the delegatee) if the policy decision to the request changes. When a policy is changed, previous requests are re-evaluated, and the requestor is informed that his access rights have changed. To support this approach, we propose an extension to the ACF architecture that permits proactive enforcement of policies.

A. Architecture overview

We describe the main components of the task delegation framework supporting proactive policy enforcement. We detail what parts were changed and what the new extended architecture looks like (see Fig. 5).

- **Policy Manager**: see section II-B.
- **ACF**: see section II-B.

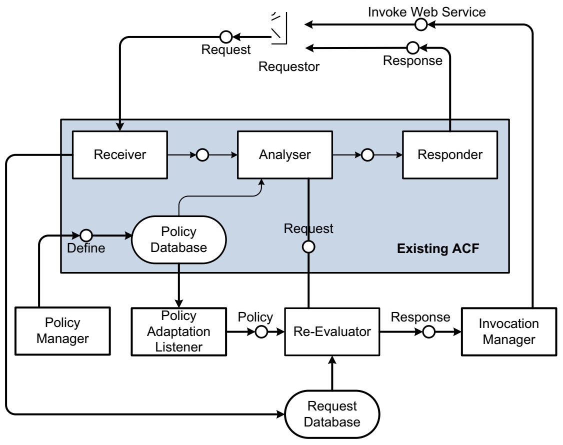

Fig. 5. Architectural extension supporting delegation policies

- **Dynamic Policy Enforcement Component**: It implements our approach to support proactive policy decisions. We extend the ACF architecture with additional components related to the policy database. When the *Receiver* receives a request, it sends this to a Request Database that stores this request. A *Policy Adaptation Listener* component polls the *Policy Database* and sends an event to a *Re-evaluator* component when a policy has changed. This queries the *Request Database* to retrieve the previous requests made, and sends this to the *Analyser* component for re-evaluation. The *Analyser* then sends back a new response to the *Re-evaluator*, which queries the *Request Database* to see if this is different to the response given to the request being analysed. If this is a different response, the *Invocation Manager* component invokes the "contact point" provided by the requestor (and stored in the Request object) with the new response.

B. Architecture requirements

On an *architectural level*, as requests are required to be re-evaluated upon a policy change, a storage mechanism of previous requests and the given response are needed. If a previous request is re-evaluated and a different response to the one stored is produced, the ACF must inform the requestor of the new result. Thus, a mechanism must exist that triggers a re-evaluation when it detects a policy change. These effects of the policy change would be automatically captured, and conveyed back to the requestor for appropriate actions. In addition, an invocation component is needed that actually marshals this information to the requestor.

On a *language level*, the approach would require new constructs to describe the invocation method that the ACF can use to contact the requestor. As such, this acts as a "contact point" for the requestor. In a service-oriented architecture (SOA), this contact point could consist of the endpoint of a service that could be invoked (see the Invoke Web Service in Fig. 5). Subsequently, all access policies must be centralised and referenced by the SOA architecture which is protected. We give an SOA a single point of access and we let the services register with our ACF. Since services are essentially black boxes, we define how to contact them and to sort out what it means when policy changes.

On a *technical level*, the Policy Manager generates policies and subsequently embed credentials attributes for authentication and authorisation purposes. Credentials providers such as certification of authorities issue digital certificates to the requestor in order to compute his request by the ACF. At this stage, the *Receiver* component acts as a policy enforcement point to perform access control by making decision request and enforcing decisions. For instance, an attribute certificate is issued to the delegatee for authentication and authorisation purposes [9]. Attribute certificates will ensure integrity, protection and non-repudiation through a digital signature. The *Receiver* gets his attributes certificate and checks his permissions afterwards. The retrieved attributes are validated against the policy (e.g. subject attributes, validity time). Once the delegatee has been successfully authenticated, he will attempt to perform specified actions on task resources. At each attempt, the *Receiver* passes the access request to the *Analyser* to decide. Decisions results (permit, deny, or not applicable) are then sent via the *Responder*.

A new re-evaluation of a policy defines new attributes for further request with regards to policy changes. As a first solution, a service is invoked to contact the delegatee and based on the mutual agreement between delegation principals, appropriate actions will take place. For instance, the cancellation of his work and the log off from the system.

VI. Conclusion

In this paper, we developed a solution to model task delegation within workflows and to specify delegation constraints into access control systems. The motivation of this work was based on human-centric workflow from an eGovernment case study. We defined delegation protocols to support user-to-user interactions. In doing so, we modelled delegation protocols with regards to workflow's invariants in terms of users, tasks and resources. We then explained how to align delegation constraints between both workflow and access control systems. Moreover, we developed an extension to an existing access control framework in order to ensure dynamic delegation of authority.

The next stage of our work is the implementation of our framework using XACML (eXtensible Access Control Markup Language) standard supporting delegation constraints. Additionally, we are looking to enrich the access control framework with a *listener* component that permits dynamic changes within existing policies.

References

[1] M. Clavel, V. Silva, C. Braga, and M. Egea, "Model-driven security in practice: An industrial experience," in *ECMDA-FA '08: Proceedings of the 4th European conference on Model Driven Architecture*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 326–337.

[2] V. Atluri and J. Warner, "Supporting conditional delegation in secure workflow management systems," in *SACMAT '05: Proceedings of the tenth ACM symposium on Access control models and technologies*. New York, NY, USA: ACM, 2005, pp. 49–58.

[3] M. Zur Muehlen, *Workflow-based Process Controlling. Foundation, Design, and Application of workflow-driven Process Information Systems*. Logos Verlag Berlin, 2004.

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

[5] T. A. R4eGov, "Towards e-administration in the large," 2006, http://www.r4egov.eu/.

[6] C. Wolter, M. Menzel, A. Schaad, P. Miseldine, and C. Meinel, "Model-driven business process security requirement specification," *System Architecture.*, vol. 55, no. 4, pp. 211–223, 2009.

[7] C. P. Pfleeger and S. L. Pfleeger, *Security in Computing (4th Edition)*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2006.

[8] Tim Moses, "eXtensible Access Control Markup Language (XACML) Version 2.0," 2005, committee specification, OASIS.

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

[10] J. Crampton and H. Khambhammettu, "Delegation in role-based access control," in *Proceedings of the Computer Security - ESORICS 2006, 11th European Symposium on Research in Computer Security, Hamburg, Germany, September 18-20, 2006*, ser. Lecture Notes in Computer Science. Springer, 2006, pp. 174–191.

[11] WFMC, The Workflow Management Coalition, "Workflow Management Coalition Terminology and Glossary," 1999, document Number WFMC-TC-1011.

[12] 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.