THE DELEGATION AUTHORIZATION MODEL: A MODEL FOR THE DYNAMIC DELEGATION OF AUTHORIZATION RIGHTS IN A SECURE WORKFLOW MANAGEMENT SYSTEM

Karin Venter
Department of Computer Science, Rand Afrikaans University
PO Box 524, Auckland Park, Johannesburg, 2006 South Africa
e-mail:kv@adam.rau.ac.za

Martin S Olivier
Department of Computer Science, Rand Afrikaans University
PO Box 524, Auckland Park, Johannesburg, 2006 South Africa
e-mail: molivier@rkw.rau.ac.za

Abstract

A *workflow* is a coordinated arrangement of related tasks in an automated process, the systematic execution of which, ultimately achieves some goal. Tasks that comprise the workflow process are typically dependent on one another. Security, in a workflow context, involves the implementation of *access control security mechanisms* to ensure that task dependencies are coordinated and that tasks are performed by authorized subjects only. A *Workflow Authorization Model* (WAM) [AH96b] has already been developed to enforce security principles on workflows, by addressing the granting and revoking of authorizations in a Workflow Management System (WFMS). This WAM satisfies most criteria required for an optimal access control model for workflows, some of which cannot be met through pure role-based access control (RBAC) mechanisms. This paper addresses the *delegation* of task authorizations within a workflow process by subjects in the organizational structure. The proposed The *Delegation Authorization Model* (DAM) will work within the security constraints imposed by the WAM when deciding whether delegations will be approved or denied. It will also take into account the dynamically determined constraints imposed by the DAM itself.

1. INTRODUCTION

The pace at which business is conducted has increased dramatically over recent years, forcing many companies to re-evaluate the efficiency of their business processes. The restructuring of organizational policies and methods for conducting business has been termed “Business Process Re-engineering" (BPR). These refined business processes are automated in *workflows* that ensure the secure and efficient flow of information between activities and participants that constitute the business process.

Securing the information and procedures associated with workflows is an area of increasing concern. It can almost be said that the success of a workflow implementation largely depends on its secure execution. This is particularly true when one considers the need to manage authorization rights when dealing with sensitive data, information, or procedures. A *Workflow Authorization Model* (WAM)[AH96b, HA99] has already been developed to enforce security principles on workflows, by addressing the granting and revoking of authorizations. The WAM satisfies most criteria required for an optimal access control model for workflows, those being the enforcement of separation of duties, the handling of temporal constraints, a role-based application, and the synchronization of workflow with authorization flow. Some of these conditions cannot be met through pure role-based access control (RBAC) mechanisms [SCFY96].

This paper addresses the *delegation* of task authorizations within a workflow process by subjects to other individuals in the organizational structure. The proposed Delegation Authorization Model (DAM) will work within the security constraints imposed by the WAM, whilst taking into account dynamically defined constraints relevant to specific delegation procedures. As of yet, this area has not received much attention.

The paper is structured as follows. Firstly, we will introduce workflow concepts fundamental to our understanding. An overview of security concepts relevant to workflows will then be given, after which the Workflow Authorization Model will be discussed briefly. Finally the Delegation Authorization Model will be presented, followed by an evaluaion of the model.

2. WORKFLOW

Workflow concepts are not new. They have primarily been associated with business process re-engineering systems but have also been encompassed in various other types of technologies such as image processing tools, document management technology, electronic mail facilities, and transaction-based applications.

A *workflow* is formally defined in [WFMC99] as “the automation of a *business process*, in whole or in part, during which documents, information or tasks are passed from one participant to another for action, according to a set of pro-

cedural rules". In this definition, a *business process* refers to the association of activities that, when executed in a systematic way, ultimately achieve some goal or objective. A *workflow management system* (WFMS) is responsible for the automation of business processes and executing them as workflows. *Activities* represent the smallest unit of work in the workflow. Two types of activities exist, *manual* or *automated*. **Manual activities** are those activities that aren't, or cannot be, automated. They are tasks that are executed by humans or constitute manual work. Consequently, such activities do not form part of the workflow, as they are not capable of computer automation. **Automated activities** can be automated and managed by a Workflow Management System. Automated activities are commonly referred to as "workflow activities". We will use the terms activities and tasks interchangeably, to refer to automated activities. A *process definition* specifies the workflow process in a manner which makes it possible to be managed by a workflow management system. It depicts the associations between activities in the workflow as well as the relationships that govern them. It is worth noting that the process definition includes the workflow definition as well as the manual definition of those activities that cannot be automated [WFMC95].

The relationships between activities and the objectives they achieve, are typically defined according to organizational policies and structures. Thus, the order in which activities are executed is important. It is usually the case that the results of one task execution dramatically affects a subsequent activity, and in more serious cases, failure to complete one task may prevent the possible, or successful execution of another. Such conditions that are imposed on these activities are termed *task dependencies*. There are primarily two types of task dependencies, *static* or *dynamic*. **Static task dependencies** are specified in advance and do not change during the execution of the workflow. **Dynamic dependencies**, however, change throughout the workflow execution, and cannot be predetermined [AH96a]. It is important for a workflow security system to take task dependencies into account when securing the workflow process. The next section will discuss some of the security issues relevant to workflows.

3. SECURITY IN WORKFLOW

The security service of authorization (access control) is of primary relevance in the context of workflows. Access control security mechanisms need to ensure that task dependencies are coordinated and that tasks are performed by authorized subjects only. The maintenance of object integrity is important, in particular, *semantic integrity*, which concerns the consistency of information with business rules [LR00]. When considering access control in workflows, three important issues need to be addressed, namely, *separation of duty, strict least privilege,* and *order of events*. Addressing these issues is referred to as

context-sensitive access control because each emerges upon the initiation of a specific process [LR00]. An effective workflow authorization model should address the following issues.

Separation of duties constraints that reflect the business rules of the organization, should be enforced. This concept ensures that the semantic integrity of information is sustained through the prevention of fraud, by specifying that certain individuals may not perform a combination of tasks.

The concept of *strict least privilege* should be addressed. This principle allows a user to receive the least possible permissions required to perform a task [CBE00]. The enforcement of this concept prevents users from acquiring unnecessary privileges, and in doing so, reduces the risk that information integrity may be compromised during task executions.

An authorization model should also implement *event-based authorizations*. Tasks in a workflow process are typically dependent on the outcome of preceding tasks. Thus, the granting of authorizations is also dependent on task outcomes and they need to be consistent with the order of events [AH96b] [LR00].

Implementing a *role-based access control* structure will simplify the expression of business rules that are specified as constraints [AH96b].

Supporting *delegation* of authority will allow delegators to transfer and revoke privileges to other individuals [Mof98].

Finally, an authorization model should implement some sort of *reporting structure* to facilitate supervision and review. When implementing a delegation model supervision and review becomes particularly relevant as delegators need to ensure that delegatees have successfully executed the tasks assigned to them [Mof98].

The specification of *constraints* plays an important role in the enforcement of the above principles. There are primarily two types of constraints, *static*, or *dynamic* [BFA99]. **Static constraints** remain consistent throughout process execution, and thus, can be referred to in the absence of a process instance. By contrast, **dynamic constraints** are defined according to the history of the process execution, and therefore, can only be checked in the context of an executing process. Separation of duties is an example of a dynamic constraint, and the specification thereof is important for enforcing security on workflows. The next section elaborates on a model that caters to the enforcement of security principles whilst accommodating the dynamic constraints imposed on the workflow.

4. THE WORKFLOW AUTHORIZATION MODEL (WAM)

A *Workflow Authorization Model* (WAM) already exists that enforces security principles on a workflow process by addressing the granting and revoking

of authorizations in a WFMS. This section refers to the WAM proposed in [AH96b], and the extended model discussed in [HA99]. This model will provide a security framework within which our proposed delegation model will operate. The WAM enables the dynamic granting of authorizations, whilst taking into account, the time intervals of valid authorizations with respect to the time during which task execution occurs. This will ensure that task authorizations are only valid for the time periods of the task durations. To summarize, the WAM satisfies most criteria required for an optimal access control model, those being the enforcement of separation of duties, the handling of temporal constraints, a role-based application, and the synchronization of workflow with authorization flow. Some of these conditions cannot be met through pure RBAC mechanisms [SCFY96]. This section explores the most important concepts of this model.

The Authorization Template (AT). The WAM makes use of an *Authorization Template* (AT) that is associated with each task in the workflow process. Authorization Templates define the static rules for authorizations, which can be defined during the workflow design process. When a task executes in a workflow, the associated authorization template is consulted to determine if a new authorization may be granted. These *authorizations* are defined in [AH96b] as 4-tuples $A = (s, o, pr, [\tau_b, \tau_e])$, where subject s is granted access on object o with permission pr at time τ_b, and this privilege is revoked at time τ_e.

ATs help ensure that appropriate task authorizations are granted and revoked for the time period of task execution only, thereby achieving synchronization between the authorization flow and the workflow. [HA99] defines an *Authorization Template* (AT) as 4-tuple $AT(tw_i) = ((r_i, -), (\gamma_i, -), pr_i, [\tau_{li}, \tau_{ui}])$. The subject hole $((r_i, -))$ can be filled by a subject (s_i) and object hole $(\gamma_i, -)$ can be filled by an object (o_i). The type of privileges (pr_i) granted, as well as the time interval $([\tau_{li}, \tau_{ui}])$ during which the task must be executed are also parameters. The addition of the time interval parameter distinguishes this approach from earlier access control models.

Separation of Duties Constraints imposed on the WAM. The Workflow Authorization Model effectively handles separation of duties constraints, by dynamically calculating a set of *relevant constraints* (C_{twi}) from a set of *potential authorizations* (PA_i) that contains all possible authorizations for a particular task instance. Because these sets are calculated during run-time, they handle dynamic constraints such as separation of duties. Once relevant constraints have been identified, a set of *eligible subjects* is formed in a manner that takes into account the dynamically calculated separation of duties constraints [HA99].

The granting and revoking of authorizations. Authorizations are granted and revoked according to the *Authorization Derivation Rule for extended WAM* [HA99]. In the *Grant Rule* specifies that only those subjects that are contained within the set of *eligible subjects* may gain authorization. This process directly references the AT for the task at hand. The time interval for the validity of the authorization is then adjusted according to the time of task commencement and the time needed for task completion. When an authorization must be generated, the AT of the task must be consulted at run time. The set of eligible subjects must be consulted to assign a subject to the task, but only when an object of the type specified in the AT is received by the task. Authorizations are revoked according to the *Revoke Rule*. Quite simply, this rule specifies that authorizations are revoked upon task completion.

We will now present our Delegation Authorization Model (DAM) that builds on this WAM.

5. THE DELEGATION AUTHORIZATION MODEL (DAM)

The *Delegation Authorization Model* (DAM), aims to facilitate the delegation of task activities between subjects in the organizational hierarchy. The model will be implemented in conjunction with the WAM discussed in Section 4, to ensure that security constraints are fully implemented.

The task of delegation can be very useful for real-world situations where a user that is authorized to perform a task is either unavailable or too overloaded with work to successfully complete it. This can occur, for example, when certain subjects are sick or on leave. It is more often than not the case that delaying these task executions will violate the time restrictions placed thereon, thereby preventing the successfull implementation of subsequent tasks and hence the entire workflow execution. Thus, delaying a task execution can undermine the primary advantage associated with workflow automations, which is to make business processes more efficient.

In RBAC systems, multiple individuals are assigned to a role set, and hence task delegations would simply mean selecting another subject that is authorized to assume that role. However, in cases where dynamic constraints are considered, such as separation of duties, the set of subjects eligible to perform a role may be very small and hence a task may need to be delegated. Similarly, when sensitive objects need to be accessed or important decisions need to be taken, the set of eligible subjects with the authority to perform such roles will be considerably smaller.

The Delegation Authorization Model (DAM) that we propose, will accommodate the activity of delegation by adding it to the workflow process definition as an activity that can occur anywhere in the workflow process. This implies that

any subject may delegate task responsibility to another individual, at any point in the workflow process, without creating any additional task dependencies.

For the purposes of this paper, we will define the person delegating responsibility to another subject as the *delegator* (denoted as do). The subject being delegated to, will be referred to as the *delegatee*, and will be denoted in further formal definitions as s.

There are a number of issues that arise when facilitating delegation in a workflow management system. Most of them can be addressed through the specification of static and dynamic constraints. This paper will attempt to address some of them.

Fistly, it is important to ensure that delegation rules cater to *dynamic constraints* in order for them to be effective. Thus, when selecting a delegatee, separation of duties constraints must be taken into account.

Secondly, care must be taken that delegation procedures are *controlled* so that subjects aren't swamped with too much work or authority that has been delegated.

Related to the issue above is that of *refusal*. Can a delegatee refuse the delegation in the event that they have too much work? If so, under which circumstances can this be done?

Can *upward delegation* take place? In other words, it should be decided whether subordinates can delegate to superiors in the organizational structure. The subject of upward delegation raises numerous issues that we will address in future works. For this paper, we will not restrict our DAM to enforce constraints on upward delegation.

The presence of delegation raises the very important issue of *accountability*. Should the delegator maintain responsibility for the success of the task execution, or should there be cases when it is to be transferred to the delegatee?

Finally, the WAM places a *time restriction* for each task execution, that is synchronized with the time validity of authorizations. When a task is delegated, it should be considered whether these time constraints should be altered or assumed.

This paper will address the facilitation of dynamic constraints in our DAM as we consider it to be of primary importance. Future works will attempt to address others of the above-mentioned issues.

5.1. THE DELEGATION AUTHORIZATION TEMPLATE (DAT)

Our DAM introduces the concept of a *Delegation Authorization Template* (DAT), that will define the static rules used to determine which subjects are authorized to delegate authorizations for task executions. The definition of the

DAT for each task, is based on the definition of an *Authorization Template* taken from [HA99].

Definition 5.1 *Given a task tw_i, a* **Delegation Authorization Template** $DAT(tw_i)$ *is defined as a 4-tuple*

$$DAT(tw_i) = ((do_i, -), (de_i, -), pr_i, [\tau_{li}, \tau_{ui}])$$

where:

1. $(do_i, -)$ *is a subject hole which can be filled by a delegtor* do_i *where* $G(s_i) = do_i$,
2. $(de_i, -)$ *is a subject hole which can be filled by a subject (delegatee)* s_i *where* $F(s_i) = de_i$,
3. pr_i *is the privilege to be delegated to* de_i *by* do_i.
4. $[\tau_{li}, \tau_{ui}]$ *is the time interval during which the task must be executed.*

>From the definition above, the time parameter $[\tau_{li}, \tau_{ui}]$ remains consistent with the interval defined in the $AT(tw_i)$. This implies that in the event of a delegation, the time allowed for a task execution is not altered.

It is worth noting that the DAT may be implemented in a variety of ways, for example, an access control list or an access control matrix. The implementation chosen will be left to the discretion of the organization within which the model will operate. For now it is sufficient to view it as a template that specifies static delegation relationships between roles in the organization. Each task (tw_i) can have one or more DATs associated therewith. Up to now, our DAM only caters to static delegation constraints that are predefined in Delegation Authorization Templates (DATs). The next section will address dynamic constraints.

5.2. HANDLING DYNAMIC CONSTRAINTS

Our DAM will calculate a set of eligible delegatees ($S_i^{de}(o)$) for each task, in a manner that takes into account exclusive dynamic constraints, such as separation of duties. This set will be directly referenced at a later stage when we formulate an approach for the granting and revoking of authorizations.

We will refer to the authorization base AB_{NT}, that was specified in [HA99], in our formulation of the set of eligible delegatees ($S_i^{de}(o)$) for task i (tw_i). Our definition of an exclusive constraint remains consistent with the one defined by [HA99]. Similarly we will reference the set of relevant constraints (C_{twi}), for the task in question (tw_i), as calculated by the WAM.

The set of eligible delegatees is formed by adapting the definition of the set of eligible subjects ($S_i^e(o)$) specified in [HA99], as follows:

Definition 5.2 *Given a delegation authorization template:* $DAT(tw_i) = ((do_i, -), (de_i, -), pr_i, [\tau_{li}, \tau_{ui}])$ *we define a set of eligible delegatees* $S_i^{de}(o)$ *as:*

1 $S_i^{de}(o) = S_{de_i}$, *if* $C_{twi} = \emptyset$

2 $S_i^{de}(o) = S_1 \cap S_2 \cap S_3 \ldots \cap S_n$,

where each
$S_k = S_{ri} - s(q_i)$, *if* $c_k : qi \leftarrow p_j \in C_{twi}$ *is an* **exclusive constraint** *and* p_j *is true with respect to* AB_{NT}

The definition above states that to be an eligible delegatee, one must either be a statically authorized delegatee which is denoted in the DAT for the specific task i (tw_i), or one may be any subject that is not restricted by an exclusive constraint. This exclusive constraint is a dynamic constraint that the WAM has calculated for the task in question (tw_i).

We have specified static constraints in the DAT, and catered to dynamic constraints through the calculation of the set of eligible delegatees. We can now formulate an approach to determine whether a subject may indeed perform a task that he/she has been assigned, by specifying a rule for the granting and revoking of authorizations.

5.3. APPROVING AUTHORIZATIONS IN THE PRESENCE OF DELEGATION

An authorization will be granted to subject s, for task tw_i if:

1 The subject s is in the set of eligible subjects ($S_i^e(o)$), or

2 the subject s is in the set of eligible delegatees ($S_i^{de}(o)$)

>From this logic, it can be seen that the DAT need only explicitly record the possibility for a delegator to delegate to a subject that is *not* authorized to perform the task according to the WAM. This is because delegation is immediately allowed if the delegatee is already in the set of eligible subjects $S_i^e(o)$ authorized to perform task tw_i.

Mathematically, we can formulate a *Delegation Authorization Rule* that is adapted from the *Authorization Derivation Rule for Extended WAM* [HA99], that was briefly discussed earlier. This rule will specify the grantng and revoking of authorizations in the presence of delegation.

Definition 5.3 *[Delegation Authorization Rule] Given an Authorization Template* $AT(tw_i) = ((r_i, -), (\gamma_i, -), pr_i, [\tau_{li}, \tau_{ui}])$ *and a Delegation Authorization Template* $DAT(tw_i) = ((do_i, -), (de_i, -), pr_i, [\tau_{li}, \tau_{ui}])$ *an authorization* $A_i = (s, o, pr_i, [\tau_{bi}, \tau_{ei}])$ *for task* tw_i*, is derived as follows:*

Grant Rule: *Suppose object x is sent to subject y at τ_{ai} to start tw_i.*
If $x \in O_{\gamma i}$ and ($y \in S_i^e(x)$ or $y \in S_i^{de}(x)$) and $\tau_{ui} \geq \tau_{ai}$,
$S_i \leftarrow y$, $o_i \leftarrow x$, $pr_i \leftarrow pr(AT)$;
$\tau_{ei} \leftarrow \tau_{ui}$,
if $\tau_{li} \geq \tau_{ai}$,
$\tau_{bi} \leftarrow \tau_{li}$;
else $\tau_{bi} \leftarrow \tau_{fi}$.

Revoke Rule: *Suppose w_i ends at τ_{fi}, at which point o_i leaves tw_i.*
If $\tau_{ui} \geq \tau_{fi}$,
$\tau_{ei} \leftarrow \tau_{fi}$.

From Definition 5.3, a few comments are in order. Firstly, the time allocated to the delegatee for task execution $[\tau_{li}, \tau_{ui}]$, is kept consistent with the interval originally defined in the AT for the task in question (tw_i). This will ensure that no delays to the workflow will be incurred due to tasks being delegated. More importantly, dynamic security constraints are addressed by referencing two distinct sets, the set of eligible subjects ($S_i^e(x)$), and the set of eligible delegatees ($S_i^{de}(x)$). The authorization model (WAM) dynamically determines which subjects are authorized to perform the task in question through the formation of the set of eligible subjects ($S_i^e(x)$). The delegation authorization model (DAM) dynamically determines the set of eligible delegatees ($S_i^{de}(x)$) in the event of a delegation (See Definition 5.2).

5.4. ANALYSIS OF THE DELEGATION AUTHORIZATION MODEL (DAM)

There are a number of advantages to the WAM that can also be associated with the DAM. The WAM enforces security principles on a workflow process by addressing the granting and revoking of authorizations. The WAM caters to dynamic task dependencies whilst ensuring that these tasks are executed within the time constraints imposed thereon. Thus, the validity of authorizations is restricted to time constraints that are dynamically determined. Synchronization of authorization flow with the workflow is a major advantage of the model. Consequently, the WAM is an appropriate authorization model for workflows, where the prediction of task execution time intervals is very difficult. A key element of the WAM is its formation of a set of *eligible subjects* by considering *separation of duties constraints*. This is where another strength of the model lies. Finally, this model facilitates the assignment of authorizations to organizational *roles*, thus enabling one to apply it to any role hierarchy within an organization.

Integrating the WAM with this DAM ensures that a delegation model may be effectively implemented whilst still satisfying the most important security requirements of workflow systems. The model ensures that delegations are

only valid for authorization time intervals which are in turn synchronized with the workflow to ensure that principles of least privilege are implemented. Furthermore, through the dynamic specification of constraints, separation of duties restrictions can be catered to. The DAM is applied through the specification of constraints that directly determine the set of eligible subjects. Thus, only minor adjustments need to be made to the authorization derivation rule [HA99] in order for the DAM to be accommodated. By these means, it enhances the WAM without complicating it unecessarily. Most importantly, however, implementing the DAM within the architecture of the WAM will allow the DAM to maintain the characteristics of an optimal access control workflow model.

6. CONCLUSION

Workflow technology has a significant impact on the operations of business processes. Thus, it is important that this technology is optimized in its implementation through the enforcement of sound security principles. The Workflow Authorization Model (WAM) [AH96b] and the extended model defined in [HA99], is designed to do just that. The WAM satisfies most criteria required for an optimal access control model, those being the enforcement of separation of duties, the handling of temporal constraints, a role-based application, and the synchronization of workflow with authorization flow. Our Delegation Authorization Model (DAM) works with the WAM and hence, it inherits the security advantages associated with the impementation thereof. Our DAM also caters to dynamic constraints imposed on the activity of delegation. Future work aims to address other issues that were identified.

References

[AH96a] V. Atluri and W-K Huang. (1996). “Modeling and Analysis of Workflows using Petri Nets." *Technical report.* CMIC, Rutgers University.

[AH96b] V. Atluri, W-K Huang. (1996). “An Authorization Model for Workflows", In: *Proceedings of the Fifth European Symposium on Research in Computer Security.* Rome, Italy. pp. 44-64.
http://cimic.rutgers.edu/atluri/esorics96.ps

[BFA99] E. Bertino, E. Ferrari, V. Atluri. (1999). “Specification and enforcement of authorization constraints in workflow management systems." In: *ACM Transactions on Information and Systems Security, 2(1).* pp. 65-104.
http://www.acm.org/pubs/articles/journals/tissec/1999-2-1/p65-bertino/p65-bertino.pdf

[CBE00] D.G. Cholewka, R.A. Botha, J.P. Eloff. (2000) “A context-sensitive access control model and prototype implementation." In: *Information Security for Global Information Infrastructures: IFIP TC 11 Sixteenth Annual Working Conference on Information Security.* Beijing, China. pp. 341-350. (Aug 2000)

[HA99] W-K Huang and V. Atluri. “SecureFlow: A Secure Web-enabled Workflow Management System." In: *Proceedings of the fourth ACM workshop on role-based access control*. Fairfax, VA USA, pp. 83-94. http://www.acm.org/pubs/citations/proceedings/commsec/319171/p83-huang

[LR00] F. Leyman and D. Roller. (2000). *Production Workflow: Concepts and Techniques*. Prentice-Hall.

[Mof98] J.D. Moffett. (1998). “Control Principles and Role Hierarchies." In: $3^r d$ *ACM Workshop on Role Based Access Control (RBAC).* George Mason University, Fairfax, VA, 22-23 October 1998.

[SCFY96] R. Sandhu, E. Coyne, H. Feinstein, C. Youman. (1996). “Role-based Access Control Models." In: *IEEE Computer 29(2).* pp. 38-47. http://dlib.computer.org/co/books/co1996/pdf/r2038.pdf

[WFMC95] Workflow Management Coalition, Author: David Hollingsworth. (1995). *The Workflow Reference Model*. Issue 1.1, No TC00-1003.

[WFMC99] Workflow Management Coalition. (1999). *Terminology and Glossary*. Issue 3.0, No WFMC-TC-1011.

K. Venter and M. S. Olivier, “The Delegation Authorization Model: A model for the dynamic delegation of authorization rights in a secure workflow management system,” in *ISSA2002*, Muldersdrift, South Africa, 2002. Published electronically.

Source: http://mo.co.za