Deploying Access and Flow Control in Distributed Workflows

Samiha Ayed, Nora Cuppens-Boulahia and Frederic Cuppens

TELECOM Bretagne
Cesson Sevigne 35576, France
{Samiha.ayed, Nora.cuppens, Frederic.cuppens}@telecom-bretagne.eu

Workflows are operational business processes. Workflow Management Systems (WFMS) are concerned with the control and coordination of these workflows. In recent years, there has been a trend to integrate WFMS in distributed inter-organizational systems. In this case malfunctioning of one WFMS can affect more than one organization, making the correct functioning of a WFMS a critical issue. Thus, an important function of WFMS is to enforce the security of these inter-organizational workflows. Several works have been done to integrate the security aspects in the workflow specification. Unfortunately, these research works generally adopt a centralized management approach and are based on static access control models. Therefore, they do not deal with flow control, a very important requirement in WFMS. In this paper, we suggest a decentralized and dynamic approach to handle a security policy in workflows taking into account access and flow control. The last part of the paper deals with the workflow satisfiability problem. It studies the complexity of the problem which asks whether a set of users can complete a constrained workflow in some system configuration.

Keywords: WFMS, OrBAC, DTE, Petri Nets, Security Policy
ACM Classifications: H.1, H.2, H.3, H.4, H5

1. INTRODUCTION

A workflow is a process consisting of several tasks to be executed by respecting specific constraints. WFMS are based on representing processes as workflows. A workflow representation implies that tasks composing it are interdependent and are communicating control information and data to each other. For example, let us consider a workflow composed of tasks T_1, T_2 and T_3 which must be executed in a sequential order. If we suppose that these three tasks act on the same documents, the access to these documents must be controlled according to the execution order of tasks. In other words, this access control must be synchronized with execution progression of the workflow. In addition, the execution of a task is related to the execution of precedent tasks. So an access control model is needed. On the other hand, workflow tasks can be executed within the same organization or within different ones. In the first case, the workflow security is maintained by the same organization which has the authority to manage the security of its different roles and their interactions and to control the access to its different objects. In the second case, a more rigorous treatment must be done to ensure a secure execution of the workflow. So, security measurements are not limited to the access control which must be applied but also it concerns the need to apply a

Manuscript received: 18 July 2008
Communicating Editor: Ljiljana Brankovic

flow control to manage interactions between different components of the WFMS. If a piece of information Info flows from a role R_1 to another role R_2 of the same organization or a different one, we must be sure that R_2 has privilege allowing it to get an access to Info. If R_2 must not access to Info and R_1 transmits the information Info to R_2, a leakage of information has happened. The problem is more dangerous if the transferred data is confidential or has a certain sensitivity degree in general. Such a problem is known as the "confinement problem" (Boebert, 1996; 1985) defined as a leakage of data. In order to resolve such a problem, flow control models have to be used. So, a workflow specification must be correlatively defined with a security policy which takes into account access control and information flow control simultaneously. Many research works have been interested in studying this issue but no study has addressed the subject completely. Several proposals (Adam, 1998; Atluri, 1996; Huang, 1999; Hung, 1999; Bertino, 1999) tried to address this problem and proposed different access control models to manage security in WFMS without taking into account information flow control. Their propositions consist in (1) specifying the global workflow security policy (an access control policy) and (2) defining a *centralized* management procedure that controls the execution of the workflow so that it remains compatible with its associated security policy. Since these approaches are generally based on the RBAC model (Ferraiolo, 1992) which only provides means to specify static security requirements, they present some limitations. Further that they present static approaches, they do not care about information flow control when defining their policy. Other works were done in order to converge access and information flow control models away from the WFMS environment. Several authors have discussed the relationship between RBAC and MLS lattice based systems (Nyanchama, 1994, 1996; Sandhu, 1996, 1998; Osborn, 1997, 2000; Kuhn, 1998; Demurjian, 2001; Atluri, 1997, 2000). Their basic idea that we retain is that the majority of them are founded on a mapping of RBAC notions and MLS notions. In these models, subject clearances are used as security levels to be assigned to roles in a role-based system. Beyond the limitations of MLS models, we consider that a such correspondence is a misunderstanding of MLS notions. To go in further details the reader can refer to Ayed (2007). In this paper, we suggest a different approach to manage security in WFMS. Our security policy associated to the workflow execution is specified using the OrBAC model (Abou, 2003). OrBAC provides means to define dynamic and contextual security requirements. For this purpose, this model defines two useful notions. The first notion is the *organization* which can be seen as an organized group of active entities. Workflow tasks may be executed in the same or different organizations. If they are executed within the same organization, the policy has to manage security in this organization. The notion becomes more useful if workflow tasks are executed in independent organizations. In this case, flows between different organizations must be managed. The second interesting notion defined in OrBAC is the *context*. A context is used to express permissions, prohibitions or obligations that apply in specific circumstances. Each context has a name and its definition depends on the organization (Cuppens, 2003). Therefore, the term "context" corresponds to any constraint or extra conditions that join an expression of a rule in the access control policy. OrBAC classifies contexts according to their type. *Prerequisite context* aims to restrict or extend privileges granted to a role depending on some conditions. So, this context category is useful in WFMS to specify constraints associated with the workflow process execution. The *provisional context* depends on previous actions the subject has performed in the system. In other words, it corresponds to a history of execution. Provisional contexts are very interesting in the domain of WFMS since the execution of a task depends on the history of execution of precedent tasks. Also, it permits the definition of a dynamic security policy according to contexts, a very useful requirement in WFMS.

In this paper, using OrBAC notions we present a Petri net based model to represent workflows. Then, we define the WFMS security policy that we have to associate with the workflow model. Such a policy deals with access and information flow control. It is based on OrBAC rules. The information flow control part introduces a DTE (Domain Type Enforcement) approach. It uses DTE principles, in particular "Entry point" notion, to define flow control rules. Afterwards, we show how to manage this WFMS security policy to control the workflow execution in a *distributed* manner. For this purpose, we define an algorithm to generate the *local* security policy associated with the execution of each task that composes the workflow. The global policy and the Petri net model are provided as inputs of the algorithm. Our approach remedies to the static and centralized aspect of models already proposed.

The last part of this paper deals with the workflow satisfiability problem. Such a problem can be defined as follows: "Having a set of constraints associated to the workflow, can this workflow be executed and finished using a specific system configuration?" We are interested in studying the complexity of such a problem. For this reason, we divide the problem into three parts. First, we study the complexity of checking if a constraint set is consistent. Second, when the set of constraints is consistent, we study the complexity of finding a valid assignment of users to different tasks which satisfies all constraints. We show that it is a NP complete problem. Finally, we suppose that we have a workflow instantiation and a set of consistent constraints and we study the complexity of checking if this assignment satisfies or not all constraints. We show that it has a polynomial complexity.

The paper is organized as follows. Section 2 introduces definition of WFMS and related works studying security in these systems. Section 3 introduces the OrBAC model. Section 4 presents our DTE formalism that we use to define flow control rules. We define our WFMS Petri net based model in Section 5. This model provides means to specify fine-grained execution modes between tasks using different temporal constraints. Section 6 specifies our security policy which we must associate with the workflow to ensure a secure execution environment. Section 7 addresses the issue of distributed workflow execution and defines an algorithm to generate the local policies required to execute the workflow in a secure distributed way. It also discusses the algorithm through an example. Section 8 deals with the complexity of the workflow satisfiability problem. Finally, we conclude the paper and outlines future work.

2. WFMS AND RELATED WORKS

2.1 WFMS

A workflow is a representation of a process. This process is divided into many tasks having an inter-dependent execution. The execution of these different tasks may include temporal constraints or conditions. Actually, we can suppose different manners to constrain tasks execution: two tasks may be executed in sequential, parallel or concurrent mode. A workflow is composed of two phases: a building phase and an execution phase. The first phase defines the formal representation of the process. The second one is an instantiation of the workflow which is based on the specification defined in the first phase.

A WFMS is a system which supports the specification, control, coordination and administration of processes using workflows. This is done through the execution of software which is based on the logic of workflows. These systems are used in different domains: research, industry, commerce, etc. Their management can be either centralized or distributed, according to the tasks of the process and the operation mode of the system. These systems may include many subjects or roles and many resources when managing and executing different workflows. This introduces the need to care about the security in these systems.

Tasks of a workflow may be executed by the same subject or by different ones having different roles and must access to different resources of the system. So, to ensure a secure execution, tasks must be executed by authorized persons or subjects and according to their execution order specification. In addition, subjects must have access only to objects available and authorized. This access must be limited to the period of task execution. Thus, granting or revoking privileges must be synchronized with the workflow execution progress. All these requirements lead us to combine the conception of the workflow and the security policy associated with it. Since security is essential and represents an integral part of a workflow, this security policy must ensure many properties: integrity, authorization, availability, confidentiality, authentication and separation of duty.

2.2 Related Works

Studying security in WFMS has been a topic of several research works. WFMC (Workflow Management Coalition) focuses on the development of workflows through standards that provide connectivity between different products. WFMC suggests WfRM as a model of reference for workflow. It presents a management system of workflow and its different interfaces. WFMC defines different services to ensure a secure environment: identification, authentication, authorization, confidentiality, integrity and non repudiation. WFMC (1995) studies these services using WfRM. In WFMC (1998), WFMC suggests a simple security model which defines a set of security operations. Also, it investigates inter-operability between two parties. Hence, it presents an extension to the inter-operability protocol to support the workflow service authentication data and the workflow inter-operability protocol data. However, this protocol does not consider the flow of authorizations among parties, tasks and resources during the workflow execution.

Atluri (1996) and Adam (1998) propose a conceptual and a logic model WAM (Workflow Authorization Model) to enforce the authorization flow based on the inter-dependencies between tasks using coloured and temporized Petri nets. The model defines an authorization template (AT). These ATs are defined during the building phase and they are used to derive the actual authorization during the execution phase. WAM tries to synchronize the flow of authorization with the workflow and to specify temporal constraints using a static approach. Different algorithms and interpretation of modelling are presented in Adam (1998). This model does not consider either the order of execution of tasks for the same object or the authorization access to resources. Their approach remains also static. Based on the WAM model, Alturi and Huang have developed in Huang (1999) a model called SecureFlow for WFMS. The model uses a simple language of 4G to specify different constraints on authorization. Huang (1999) defines the specification of constraints and the security policy associated with this model. Both WAM and SecureFlow are only adequate for centralized management of workflow security. Also they do not consider integrity and availability properties. Hung and Karlapalem (1999) have presented an authorization model for WFMS. This model addresses three security properties: integrity, authorization and availability. The model is based on defining a set of invariants. These invariants concern several aspects: agents, events and data of the workflow. The model is defined as an abstract machine having three layers: workflow layer, control layer and data layer. The first contains authorizations which can be granted to or revoked from an agent. The second deals with events which can be generated during the execution phase. The last layer manages granting and revoking authorizations to documents. Security requirements are then transformed into a set of invariants in different layers. Using these invariants, authorizations functions are defined in order to ensure system operation. In spite of supporting concurrent execution of tasks, the model does not address different temporal constraints which can be present in a workflow system specification.

Bertino (1999) studies authorizations in WFMS considering different constraints. Based on the constraints evaluation time, they define three classes of constraints: static constraints, dynamic constraints and hybrid constraints. This work is based on assigning users and roles to tasks. So, this ensures the non violation of constraints but does not address temporal constraints and constraints based on events. Also, it supposes that tasks are executed in a sequential mode. In the case where two tasks are executed in concurrent mode, it is impossible to define constraints on a task based on the success or failure of tasks. Thus, this model cannot be considered complete and does not address the issue of distributed management of workflow security requirements.

In this work, Bertino (1999) describes the specification and enforcement of authorization constraints in workflow management systems. The suggested model defines a workflow role specification to be a list of task role specifications. A task role specification defines a task, specifies all roles authorized to execute the task and the number of activations of the task that are permitted in a workflow instance. So, this work enumerates and defines all possible configurations that satisfy the workflow accomplishment and all constraints. In the last part of this paper we focus on calculating the complexity of the workflow satisfiability problem. This problem was the topic of Qihua (2007), but in this work, the problem is not very well exposed. Authors do not deal with the consistence of the set of constraints.

The definition of the security policy of workflows in most of the aforementioned works is based on the RBAC model. This model is restricted to positive authorizations, namely permissions. Also, it defines security policies using a static approach which is not appropriate to WFMS known to have a dynamic behaviour. To remedy this lack, we choose to base our security policy on OrBAC model as further explained in Section 6. We first present the OrBAC model and our DTE formalism that we will use.

3. ORBAC IN BRIEF

To define a WFMS model and its secure execution environment, we suggest using the OrBAC model and its concepts. Thus, before presenting the workflow model and how to express security requirements associated with the workflow execution, we first briefly recall basic concepts suggested in the OrBAC model.

In order to specify a security policy, the OrBAC model (Abou, 2003) defines several entities and relations. It first introduces the concept of *organization* which is central in OrBAC. An organization is any active entity that is responsible for managing a security policy. Each organization can define its proper policy using OrBAC. Then, instead of modelling the policy by using the concrete implementation-related concepts of subject, action and object, the OrBAC model suggests reasoning with the roles that subjects, actions or objects are assigned to in an organization. The role of a subject is simply called a role as in the RBAC model. The role of an action is called activity and the role of an object is called view. Each organization can then define security rules which specify that some roles are permitted or prohibited to carry out some activities on some views. Particularly, an organization can be structured in many sub organizations, each one having its own policy. It is also possible to define a generic security policy in the root organization. Its sub organizations will inherit from its security policy. Also, they can add or delete some rules and so, define their proper policy. The definition of an organization and the hierarchy of its sub organizations facilitate the administration (Cuppens, 2004). The security rules do not apply statically but their activation may depend on contextual conditions (Cuppens, 2003). For this purpose, the concept of context is explicitly included in OrBAC. Contexts are used to express different types of extra conditions or constraints that control activation of rules expressed in the

access control policy. So, using formalism based on first order logic, security rules are modelled using a 6-places predicate.

Definition 1: an OrBAC security rule is defined as: security_rule (type, organization, role, activity, view, context) where type ∈{permission, prohibition, obligation}.

An example of this security rule can be: security_rule (permission, a_hosp, nurse, consult, medical_record, urgency) meaning that, in organization a_hosp, a nurse is permitted to consult a medical record in the context of urgency.

4. DTE FORMALISM

To take into account flow control in our policy we use a DTE approach. Domain and Type Enforcement (DTE) (Badger, 1995; Kiszka, 2003) is a technique originally proposed to protect the integrity of military computer systems and was intended to be used in conjunction with other access control techniques. As with many access control schemes, DTE views a system as a collection of active entities (subjects) and a collection of passive entities (objects) and groups them into domains and types. This classification not only reduces the size of the access control matrix but also simplifies the management. DTE uses two tables to define an access control policy. The first table is a global table called Domain Definition Table (DDT). It governs the access of domains to types (domain-type controls). Each row of the DDT represents a domain and each column represents a type. When a subject attempts to access an object, we must verify the entry corresponding to the domain of the subject and the type of the object in the DDT. If the access needed is defined in the matrix then the access is allowed, if not, the access is denied. The second table is called Domain Interaction Table (DIT). It governs the interaction between domains (Inter-domain controls). Each row and each column of the DIT represents a domain. The intersection cell denotes the access privilege that the domain corresponding to column possesses on the domain corresponding to row. To be stronger, DTE has defined the manner to be used to pass from a domain to another. So, if a subject S belongs to a domain D_1 then it wants to transit to a domain D_2, it must refer the DIT. The intersection of D_1 and D_2 in DIT should contain an entry indicating the activity or the program that S must perform to access D_2. This entry is called the entry point. Thus, each domain has one or many entry points which consist in programs or activities to invoke by a subject in order to enter this domain. Any subject belonging to another domain must execute an entry point of the destination domain to be able to access this domain. When passing from a domain to another, a subject looses all its privileges of the source domain and gets a privileges set of the destination domain. This notion of entry point makes the inter-domain communication more strict and precise.

To use DTE as an approach to express our flow control, we propose to formalize the model. For this purpose let us introduce the following formal definitions.

Definition 2: (domain) S is a set of all system subjects (active entities). S is divided into equivalence classes. Each class represents a domain D including a set of subjects having the same role in the system.

Definition 3: (type) O is a set of all system objects (passive entities). O is divided into equivalence classes. Each class represents a type T including a set of objects having the same integrity properties in the system.

Definition 4: (Entry Point) An entry point is a program or an activity which must be executed to pass from a domain D_1 to a domain D_2, denoted EP(D_1, D2) or $EP_{\{1,2\}}$. An entry point implies two

Symbol	Temporal Constraint
$\mathcal{BW}(T_i , T_j)$	T_i must begin with T_j
$\mathcal{BA}(T_i , T_j)$	T_i must begin after T_j
$\mathcal{EW}(T_i , T_j)$	T_i must end with T_j
$\mathcal{EB}(T_i , T_j)$	T_i must end before T_j
$\mathcal{BJA}(T_i , T_j)$	T_i must begin just after T_j

Table 1: Different temporal constraints

rules: (1) subjects passing from D_1 to D_2 obtain a set of privileges depending on the entry point they execute, (2) subjects passing from D_1 to D_2 loose all their D_1 privileges.

The first rule means that the execution of an entry point defines the set of privileges that subjects will obtain when transiting from a domain D_1 to a domain D_2. These privileges are included into or equal to the set of privileges that D_2 subjects have. Each domain can have more than one entry point. The execution of these different entry points implies different privilege sets. The second rule means that if a subject leaves a domain it cannot return to it only by executing one of its entry points. So, another definition of an entry point is:

Definition 5: (Entry Point) An entry point is a program or an activity which must be executed to pass from a domain D_1 to a domain D_2, denoted $EP(D_1, D_2)$ or $EP_{\{1,2\}}$. Its execution must correspond to one of these two cases: (1) restrict the privileges set of D_2 defined in DDT to subjects transiting from D_1 to D_2, (2) extend the privileges set of D_2 defined in DDT to subjects transiting from D_1 to D_2.

5. MODELLING WFMS

5.1 Petri Net Based Model

Our WFMS model is based on Petri-Nets (David, 1992) which provide an expressive formalism to represent synchronous activities. With the graphic representation that they offer, they are considered simple and easy to understand. In addition, they have a mathematical foundation which provides means to formally analyse obtained models. Thus, Petri nets allow a flexible transition from the conceptual level to the implementation of test, where the system can be simulated and validated before proceeding to a detailed conception and implementation.

Definition 8: a Petri net is defined as a 5-tuple, $PN = (P, T, F, W, M_0)$, where: (1) P is a finite set of places, (2) T is a finite set of transitions, (3)$F \subseteq (P \times T) \cup (T \times P)$: a set of arcs from places to transitions and from transitions to places, (4) $W : F \rightarrow \{1, 2, 3, ...\}$: a weight function, it defines weights assigned to different arcs, (5) M_0: $P \rightarrow \{0, 1, 2, ...\}$: the initial marking, it describes initial place contents.

Definition 9: in a synchronized Petri net, each transition is associated with an event and the release of the transition will happen (1) if the transition is valid, (2) when the event occurs.

Definition 10: an interpreted Petri net has the following three characteristics: (1) It is synchronized, (2) It is P-temporized, (3) It contains an operative part where the state is defined by a set of variables V. This state is modified by operations Op which are assigned to places. It determines the truth value of conditions (predicates) C which are associated with transitions.

The model that we suggest uses interpreted Petri nets. This choice is motivated by the requirements of processes we want to represent. In fact, choosing interpreted Petri nets is related to

the operative part of the process execution. Therefore, we assign operations or tasks to different places of the Petri net. So, we need an operative part in the Petri net representation. On the other hand, we need synchronized Petri net to manage information flow control.

5.2 Constraints Requirements

A workflow specification is correlatively defined with an execution order of tasks or with temporal constraints between different tasks. In our WFMS model, we do not want to restrict execution modes of two tasks simply to sequential and parallel execution. Instead we take into account different temporal constraints that can be present between two tasks. Table 1 defines these atomic temporal constraints. Using a combination of these different cases we can reconstruct different interval relationships present in Allen (1993) which defines a complete set of execution modes of two tasks having each an execution time interval.

5.3 WFMS Petri Net Model

In this section, we use the OrBAC concepts to define our WFMS Petri net model. It is based on a formal representation of workflows using Petri nets. Our modelling is constructed in term of roles, views, activities, organizations and contexts. Roles, views and activities notions are used as they are defined in OrBAC. These concepts are defined within the formalism of Petri nets to express functional aspects of a business process. Then, such a structuring will allow us to define merely our security policy. So, roles, views and activities defined in OrBAC are respectively interpreted within the Petri net context: (1) R: a finite set of roles defined in the process, (2) V: a finite set of views used during the execution of the process, (3) A: a finite set of activities associated with places. An activity can be an atomic operation or a composed operation. A place can be validated only if all operations associated with this place are executed.

We also use concepts of context and organization suggested in OrBAC. The concept of organization is associated with places. Different operations defined in relation with a place are executed within an organization. Different places can belong to the same or to different organizations. Also, a hierarchy may exist between different organizations (Cuppens, 2004). Managing these different organizations can be either centralized or distributed. In the first case, a root organization must manage security with different other organizations which can be sub organizations of the root organization. Then, they receive or inherit their security policy from the root organization. In the second case, each organization defines and manipulates its proper security policy. In this case, information can be exchanged between the set of organizations to be able to know what is happening globally.

The notion of contexts is also associated with places. OrBAC defines several categories of contexts (Cuppens, 2003). The *Prerequisite context* aims to restrict or extend privileges granted to a role depending on some conditions. So, this context category is useful in WFMS to specify constraints associated with the workflow process execution. For instance, if P_1 and P_2 are two places of a workflow, we can associate the place P_2 to a context same_subject(P_1) to constrain subjects who are executing activities assigned to place P_1 and P_2 to be equal.

Also, the *Provisional context* that depends on previous actions the subject has performed in the system is relevant for WFMS. In fact, in a workflow execution, a task execution depends on the execution history of precedent tasks. Hence, we can associate each place with a context. It represents the context of execution of operations associated with this place. It can be defined in conjunction with prerequisite context and in conjunction with different temporal constraints which could be present between different tasks. So, a context $Cont_i$ associated to a place P_i is a triple (Pre_context,

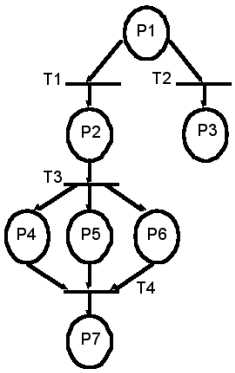

Figure 1: Example of a Petri Net

temp_context, exec_context). The first context contains conditions related to prerequisite context. The second one deals with temporal constraints defined between different tasks. The third context concerns the execution context. It defines requirements associated to P_i execution. It indicates precedent tasks that must be executed before the execution of operations associated with P_i.

However, in our model, we suppose that, when defining the security policy of the WFMS, the execution context of each place is not explicitly defined because it can be derived from the Petri net representing the workflow. Thus, in the policy, the execution context of a place is only containing the place itself. Then it will be enriched.

Finally, we define a specific type of tokens which are used in our model. A token is a triple <r, v, cont(P)>. For a token $<r_i, v_i, cont(P_i)>$ placed in a place P_i, the first parameter (r_i) indicates the role eligible to execute the operation associated with P_i. The second parameter (v_i) designates the view or the type of object which will be used in the operation associated with P_i. The last parameter (cont(Pi)) represents the context of the place P_i.

5.4 Example

To clarify the model proposed, we apply it to an application of initiating a mission for an enterprise employee. This application will need to reserve a fly, a hotel and a car. First, we introduce essential elements to define the Petri net model that we propose to represent the application. Then, we present the model in Figure 2.

- Roles defined in the model:
  - r_1: traveller
  - r_2: checker
  - r_3: a responsible of the agency of car rental
  - r_4: an agent of the hotel
  - r_5: a responsible of the airline
- Types of used objects :
  - o_1: slip of a mission
  - o_2: demand of travel
  - o_3: slip of reservation
  - o_4: slip of validation
  - o_5: mail of information

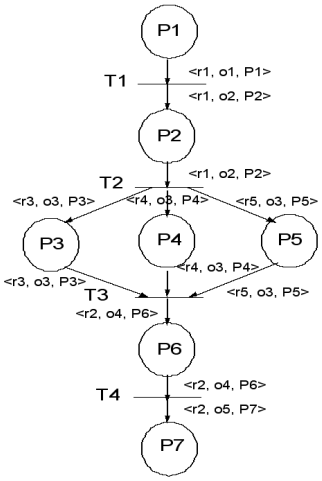

Figure 2: Petri Net representation of the application

- Operations associated with different places, an operation Op_i is associated with a place P_i:
  - Op_1: creating a mission
  - Op_2: submit the demand of travel
  - Op_3: reservation of a car
  - Op_4: reservation of a hotel
  - Op_5: reservation of a flight
  - Op_6: validation of the mission
  - Op_7: informing the traveller

Using these elements, we represent the initial model in reference with the application. This representation (Figure 2) introduces the model before the execution of the process. So, contexts contain just places to which they are associated. With the execution progression of the process, they are changed dynamically.

For example, if we consider T_3, this transition will be valid and so we will be able to validate the mission (Op_6) only if we accomplish Op_3, Op_4 and Op_5.

When the process starts its execution, contexts change with it. These changes are explained in the algorithm proposed in the next section. This Petri net modelling is the first input of our algorithm presented in the sequel. The second input will be the security policy that we specify in Section 6. So, a system manager must define a Petri net structuring of its WFMS application according to our proposed model approach.

6. SPECIFYING WFMS GLOBAL SECURITY POLICY

Once we define the Petri net modelling our WFMS application, we define the security policy that we must associate to the model. To ensure a secure execution environment of the workflow, we must take into account three aspects:

1. We have to control access to objects during tasks executions.
2. We have to ensure and enforce different execution modes defined in the workflow process.
3. We have to deal with information flow control.

Dealing with these different aspects provides means to make the security policy more effective and increases assurance that it is correctly specified. So, next objective is to specify the Petri net workflow modelling and its associated security policy in order to generate a dynamic policy that is updated with the workflow execution progress.

6.1 Access Control Policy

Our access control policy is premised on a security policy to manage access to different system objects and on a coordination security policy to enforce temporal constraints and execution modes of different tasks.

6.1.1 WFMS Security Policy

Our WFMS security policy defines access control rules. These rules are expressed using the OrBAC model. So a security rule is defined as a 6-tuple: *security-rule (type, organization, role, activity, view, context)*. These rules use the specific context defined in our Petri net model (see Section 5.3). Thus, let us consider the Petri net of Figure 1. A security rule defined as a permission granted to a role R_5 within the organization Org_5 to execute an activity A_5 associated to the place P_5 by the same subject that has executed A_4 and must end before A_4 in this Petri net and using a view V_5 will be expressed as follow: *security-rule (permission, Org_5, R_5, A_5, $view_5$, (same_subject(A_4), EB (A_4, A_5), P_5))*. We have already supposed that our initial execution contexts contain just the place itself. It is not necessary to explicitly specify the execution context associated to this rule. This execution context will be automatically derived from the workflow description using the algorithm presented in the following section. In this policy we exploit provisional and prerequisite contexts defined in OrBAC.

6.1.2 WFMS Coordination Security Policy

The WFMS security policy manages access control but it does not deal with different tasks execution modes. So to complete our access control policy we propose a coordination security

Temporal Constraints	Associated Policy
$\mathcal{BJA}$ (T_i , T_j)	*P*(begin(T*i*), default) *O*(begin(T*j*), done(T*i*))
$\mathcal{BW}$(T_i , T_j)	*P*(begin(T*i*), default) *P*(begin(T*j*), default) *O*(begin(T*i*), start(T*j*)) *O*(begin(T*j*), start(T*i*))
$\mathcal{BA}$ (T_i , T_j)	*P*(begin(T*i*), default) *P*(begin(T*j*), default) *O*(end(T*i*), done(T*j*)) *O*(end(T*j*), done(T*i*))
$\mathcal{EW}$(T_i , T_j)	*P*(begin(T*j*), default) *O*(begin(T*i*), doing(T*j*))
$\mathcal{EB}$(T_i , T_j)	*P*(begin(T*j*), default) *O*(end(T*i*), doing(T*j*))

Table 2: WFMS coordination security policy

policy which ensures a secure environment to execute workflows. This coordination security policy is complementary to our WFMS security policy. This policy controls different temporal constraints presented in Section 5.2. Also, it enforces different task dependencies and so it preserves the well functional execution of workflows.

Coordination policy is based on temporal contexts. These contexts can be used in conjunction with other contexts or conditions. They depend on the time at which the subject is requesting for an access to an object or a view. With temporal contexts, it should be possible to express that a given action made by a given user on a given object is authorized only at a given time or during a given time interval or also after or before another task execution (Cuppens, 2003). To express our coordination contexts, we reuse predicates defined in the Nomad model (Cuppens, 2005): *start(T)* ("starting T"), *doing(T)* ("doing T") and *done(T)* ("finishing T"), T is a task.

To each task we associate two specific activities called *begin* and *end*. begin(T_i) means the activity to start the task T_i. end(T_i) means the activity to finish the task T_i. Our coordination policy is defined as a set of rules associated with each case of different temporal constraints. To ensure these execution orders, we are lead to use not only permissions to control our workflow execution but also obligations imposed by temporal constraints. For example, to enforce that two tasks start together we have to apply obligations to these tasks to start at the same time, thus we are sure that their beginning is simultaneous. We present our coordination policy in Table 2. This policy makes explicit different temporal constraints presented in Section 4.2. Thus, to each case, we associate its coordination security policy that we must respect and apply. For simplicity, we represent these coordination rules using just the activity and the temp_context predicates. But these rules are defined in reality within an organization, for a specific role and using a specific view. Also, we use *"default"* context to define a context where neither conditions nor constraints on the activity are required.

6.2 Flow Control Security Policy

As we have presented, OrBAC is an efficient model to express access control rules. Our objective is then to express both access and flow controls using the same model. The DTE formalism offers SR_DIT rules to define information flow control policy.

In the sequel, we present our approach to define information flow control based on OrBAC and using the DTE approach. For the sake of simplicity we assume that all that is not permitted is denied (closed policy) and we do not deal with obligations.

Based on an OrBAC rule, SR_DIT [*SR_DIT = (rule_type, domain1, domain2, entry point)*] can be seen as a particular OrBAC rule. This rule expresses the transition between different domains and uses the entry point concept in order to preserve secure information flows and to keep DTE aspects. Therefore, to consider this particular rule we suppose the following hypotheses: (1) the source domain is considered as a role in the OrBAC rule (we have already said that the two notions have equivalent meaning), (2) the destination domain is considered as a view in the OrBAC rule, (3) the transition between two domains can be expressed as an OrBAC activity since the basic meaning of this specific rule is to handle interactions between domains. For this purpose, we define the OrBAC Enter activity, (4) the entry point defines the manner to enter into a domain. Thus, we can consider it as a condition of rule validation. Therefore, an entry point can be defined as a specific context in an OrBAC rule denoted through($E_{\{i,j\}}$). This context specifies that the rule is valid only through the $E_{\{i,j\}}$ execution.

Thus, such a rule is expressed, in a specific organization org and handling transition between D_1 and D_2, as follows: *SR (permission, org, D_1, Enter, D_2, through($E_{\{1,2\}}$))*. If there is no need to execute

any entry point to transit to another domain, the *auto* mode is used instead of the *through($E_{\{1,2\}}$)* context. Thus the rule will be *SR (permission, org, D_1, Enter, D_2, auto)*. These flow control rules can be enriched with other OrBAC contexts. Indeed, the *through($E_{\{i,j\}}$)* context can be used in conjunction with different OrBAC contexts, for example temporal contexts, to express more restrictive or conditioned flow control. Also, a transition from a domain D_1 to a domain D_2 is possible only if there is the corresponding permission in the policy. Such transitions between domains correspond to a context change. Since transiting to another domain corresponds to the activation of a new context, new rules will be activated. These rules are those for which this context is valid. This dynamic management of the security policy and the closed policy hypothesis guarantee the loss of source domain privileges during transition. New granted privileges are defined once the entry point is executed. The entity *organization* is useful to control the flow in the inter-organizational environment. This will be developed in a forthcoming paper. The set of these specific rules forms the information flow security policy that we have to associate with our Petri net model. These information flow control rules are especially associated to different transitions of the Petri net model. Thus the transiting between places which correspond to different roles is controlled by these rules and so by the execution of entry points.

7. DEPLOYING WFMS SECURITY REQUIREMENTS

7.1 Decentralized Control

Section 6 showed how to specify the workflow security policy as a set of OrBAC security rules. We apply access control rules to places and information flow control rules to transitions in our WFMS Petri net model. It is straightforward to define a management procedure compliant with a given global policy if we assume that this workflow is *centrally* managed. In this case, a request by a subject to perform an action on an object in this workflow is authorized if (1) this request is permitted by the global security policy and (2) this action can be activated according to the workflow current marking.

However, if we assume that the workflow management is distributed on several components, we can no longer assume that each component has a complete view of the workflow. Thus, our objective in this section is to derive, from the global policy, the local policy to be managed by such distributed components. For this purpose we define an algorithm that takes as input the Petri net description of the workflow and the global security policy associated with this workflow. This algorithm provides as output the local policy. This local policy is conditioned with the context of the place. A place P_i is valid, and so its operations will be executed, only if its context contains the place P_i. After the end of operations execution associated with P_i, the place P_i is deleted from its own context. Thus, it will be not valid until another execution or another event in the Petri net adds the place in its context. The local policy dynamically changes in relation with contexts. This policy synchronizes the global policy with the execution of the Petri net. So, access to different system objects and temporal constraints are secured with this local policy. In fact, when a place P_i is using a document d and another place P_j must use the same document after being used by P_i, P_i must not have access to this document when it is used by P_j. This is to ensure the integrity of the system. Using the local security policy, this risk is eliminated since access to different documents and objects is controlled by contexts. Our security policy does not only deal with access control but also with information flow control. Our information flow control is based on transitions of our WFMS Petri net model. Since we use interpreted Petri nets, we can synchronize transitions to events. In our model, we consider that these events correspond to the activation of information flow control rules. A transition creates a relation between two places. If operations associated with these two places

have to be executed by roles R_1 and R_2 respectively and by the same subject, a transition from role R_1 to R_2 is needed. This transition is controlled using our information flow rules associated with transitions. Thus a transition in our WFMS Petri net will be valid or activated if and only if all operations associated to its entry places finish and the information flow control rule associated with it is applied.

7.2 Algorithm to Deploy WFMS Security Requirements

To define our algorithm, we first introduce some hypotheses, variables, functions, sets and tables. We then present the algorithm. It introduces two levels of security policy: a global security policy and a local security policy. The first one is considered an input of the algorithm. The second is provided as output to follow the process execution. We can qualify the global security policy as static and the local security policy as dynamic since it depends on contexts of different places.

Hypothesis: The initial marking of the Petri net is a token in the first place. This comes from the definition of a workflow.

Variables: (1) i, j, k, n, m: integer, (2) Bool, result: Boolean (3) $Cont(P_i)$: context of the place P_i (4) A token $<r_k, v_k, Cont(P_k)>$, $Cont(P_k) = (Pre\_context(P_k), temp\_context(P_k), exec\_context(P_k))$

Functions: (1) Card (E) = |E|: returns the cardinality of the set E. (2) M (P_j): returns the set of tokens of the place P_j. (3) Index (E): returns index of different elements of the set E. (4) Valid_transition (bool, i): function which verifies if the Petri net contains a valid transition. If this valid transition exists, bool will be true and i will return its index.

Sets: (1) P: set of places (2) T: set of transitions (3) |P| = m et |T| = n (4) $°P_j$: set of input places of the transition T_j (5) $P_j°$: set of output places of the transition T_j (6) $°T_j$: set of input transitions of the place P_j (7) $°T_j$: set of output transitions of the place P_j (8) $P°_{\{i, j, \ldots, k\}} = P_i° \cup P_j° \cup \ldots \cup P_k°$, (identically for other sets) (9) $Cond_j = cond(T_j) = \{C_{ex}, C_1, \ldots, C_k\}$: set of conditions associated with the transition T_j where: $C_{ex} = \{C_{exi}, C_{exj}, \ldots\}$: set of conditions of execution of input places of the transition T_j. $Cond_j$ = true only if all conditions associated with T_j are true. C_{ex} is true only if all conditions of execution of input places are true.

Tables: (1) Valid[1..n]: table of Boolean which indicates validity of transitions. (2) Org[1..m]: table indicating organizations of different places of the Petri net. (3) R[1..m]: table indicating roles associated with different operations of places. (4) A[1..m]: table indicating different activities (operations) associated with different places. (5) V[1..m]: table indicating different views (objects) associated with places of the Petri net.

Proposed algorithm: The execution of the Petri net is described in algorithm 1. This algorithm assumes a Petri net representing a workflow and a global security policy as inputs. It securely executes the process by generating local contextual policies. These local policies are associated to different places of the Petri net.

The algorithm starts by initializing all execution conditions to false since no operation is being executed until now. Also, each context contains initially only the place to which it is associated. The processing of the first place is done separately. Indeed, the test for all other processing of places is done on transitions. So, if the first transition is valid (it contains at least one token) we check if the token of this place verifies the global security policy defined as an input of the algorithm. In other words, for a token $<r_k, v_k, Cont(P_k)>$ we check if the role r_k has access to the view v_k in the

organization Org[k] associated with the place P_k. After that, we construct the local security policy containing rules which manage execution of operations associated with each place. So, we check if execution_context(P_k) contains the place P_k. If it is the case, the local policy is activated. Thus, r_k can execute activities associated with P_k using the view v_k. After finishing execution, the local policy is deactivated by subtracting P_k from the set exec_context(P_k). Later, we apply the coordination policy according to policies defined in Section 5.1. These policies are presented in Table 2. To each case presented in the temp_context(P_k), we have to apply the set of rules associated with it in Table 2. Then, we have to update (1) execution condition of transitions in relation with P_k to true, (2) transitions in relation with P_k if all conditions associated with them are true, (3) contexts of places following P_k in the order of execution, (4) the marking of the Petri net by removing $<r_k, v_k, Cont(P_k)>$ from the set $M(P_k)$. The next transition releases according to information flow control rules associated with it. This processing is repeated for all places of the Petri net. So, using this algorithm we can ensure a secure execution of the process.

The algorithm considers two bases of security rules:

1. Static base of security rules (global security policy): security rules using a default context, without taking into account conditions or circumstances of execution and coordination security rules. It represents a security aspect. It defines permissions of roles on objects within an organization. It is an input of the algorithm.
2. Dynamic base of security rules (local security policy): security rules generated according to the workflow execution, so depending on contexts generated or built during workflow execution. It presents a security and a functional aspect. It is an output of the algorithm. It changes dynamically during the workflow execution. To add a contextual security rule to this base we must verify or check if there is a corresponding non contextual rule of this rule in the static base.

To have a global view of the workflow specification, we can refer to the dynamic base of security rules. Contexts of the security rules include a history of different tasks that must have been executed before each task. So, grouping whole contexts we can redesign the workflow scheme. Since these contexts express different temporal relations between workflow tasks, this design is simple and logic to be reconstructed.

The algorithm does not affect initial properties of the Petri net representation. Indeed, Petri net workflow modelling preserves its properties of reachability, liveness and boundedness. These properties can be studied using matricial representation of the Petri net marking and graph theory. In complexity terms, the algorithm has a polynomial execution time (in $O(n^3)$, n number of tasks).

7.3 Example

To clarify the algorithm, we consider an application represented by the Petri net of Figure 3. This application includes four tasks that must be executed in a specific order. With these tasks we define the following temporal constraints:

- T_2 and T_3 start when T_1 ends
- T_2 ends before T_3
- T_4 starts when T_3 ends

The application is defined within an organization org and includes four roles (R_1, R_2, R_3 and R4) and four views (V_1, V_2, V_3 and V_4).

We define our input static global policy of this application in Table 3. The execution of the algorithm generates a local dynamic policy presented in Table 4. This policy is the output of our proposed algorithm during the process execution.

Algorithm 1: Main Algorithm

```
1  for i in [1 ... |T|] do
2  |  C_ex (T_i) ← false   /* initialisation of execution conditions */
3  end
4  for i in index (P) do
5  |  exec_context(P_i) ← { P_i }    /* initialisation of contexts */
6  end
7  /* processing for the first place */
8  if (permission, Org[1], R[1], V[1], A[1], (Pre_context(P_1), temp_context(P_1), -)) ∈ SB then
9  |  DB ← DB ∪ (permission, Org[1], R[1], V[1], A[1], (Pre_context(P_1), temp_context(P_1),
   |  exec_context(P_1)))
10 end
11 if (|^o T_1 | = 0 & M(P_1)<>{} & P_1 ⊆ exec_context(P_1)) then
12 |  Execute (A[1], result)
13 end
14 for j in index (T_1^o) do
15 |  if (result = true) then
16 |  |  C_ex1 (T_j) ← true;    /* execution validation associated to P_1 */
17 |  |  Apply(flow control policy(T_j)) ;    /* information flow control rules */
18 |  end
19 |  if (cond_j = true) & valid (flow control policy(T_j)) then
20 |  |  Valid (j) ← true;
21 |  end
22 end
23 for k in index (P^o_index(T_1^o)) do
24 |  exec_context(P_k) ← exec_context(P_k) ∪ P_1;
25 end
26 exec_context(P_1) ← exec_context(P_1) \ {P_1};
27 repeat
28 |  Valid_transition (bool, i);    /* it returns a valid transition in the Petri net and its index i*/
29 |  if (bool = true) then
30 |  |  for k in index (P_i^o) do
31 |  |  |  if (permission, Org[k], R[k], V[k], A[k], (Pre_context(P_k), temp_context(P_k), -)) ∈ SB then
32 |  |  |  |  DB ← DB ∪ (permission, Org[k], R[k], V[k], A[k], (Pre_context(P_k), temp_context(P_k),
   |  |  |  |  exec_context(P_k)))    /* local dynamic policy */
33 |  |  |  end
34 |  |  |  /*updating RdP marking*/
35 |  |  |  M(P_k) ← M(P_k) ∪ <r_k, v_k, Cont(P_k)> ; /* Cont(P_k) = (Pre_context(P_k), temp_context(P_k),
   |  |  |  exec_context(P_k)
36 |  |  |  if (M(P_k)<> {} & P_k ⊆ Cont(P_k)) then
37 |  |  |  |  Apply ( Coordination Policy(temp_context(P_k)));    /*Coordination policy*/
38 |  |  |  |  Execute (a_k, result);
39 |  |  |  end
40 |  |  |  for j in index (T_k^o) do
41 |  |  |  |  if (result = true) then
42 |  |  |  |  |  C_ex_k (T_j) ← true;
43 |  |  |  |  |  Apply(flow control policy(T_j));
44 |  |  |  |  end
45 |  |  |  |  if (cond_j = true)& valid (flow control policy(T_j)) then
46 |  |  |  |  |  Valid (j) ← true;
47 |  |  |  |  end
48 |  |  |  end
49 |  |  |  for i in index (P^o_index(T_k^o)) do
50 |  |  |  |  exec_context(P_i) ← exec_context(P_i) ∪ P_k;
51 |  |  |  end
52 |  |  |  exec_context(P_k) ← exec_context(P_k) \ {P_k}; /* updating RdP marking */ M(P_k) ← M(P_k)
   |  |  |  \ {<r_k, v_k, Cont(P_k)>};
53 |  |  end
54 |  end
55 until (bool = false) ;
```

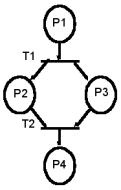

Figure 3: Petri net application

Global security policy	
General security policy	***Coordination security policy***
P(org, R1, execute, V1, default)	P(begin(T1), default)
P(org, R2, execute, V2, default)	O(begin(T2), done(T1) $\wedge$ start(T3))
P(org, R3, execute, V3, default)	O(begin(T3), done(T1) $\wedge$ start(T2))
P(org, R4, execute, V4, default)	P(begin(T2), default)
	P(begin(T3), default)
	O(end(T2), doing(T3))
	O(begin(T4), done(T3))

Table 3: Algorithm input policy

Local and dynamic security policy
P(org, R1, execute, V1, (T1))
P(org, R2, execute, V2, ($\mathcal{BA}$(T1, T2)))
P(org, R3, execute, V3, ($\mathcal{BA}$ (T1, T3)))
P(org, R4, execute, V4, ($\mathcal{BA}$ (T4, $\mathcal{BA}$(T1, $\mathcal{BW}$(T2, T3) $\wedge$ $\mathcal{EB}$(T2, T3)))))

Table 4: Algorithm output policy

We remember that in our security policy obligations have higher priority than permissions. So there is no conflict that can be generated between obligations and permissions. Only conflicts between obligations have to be managed.

8. THE COMPLEXITY OF THE WORKFLOW SATISFIABILITY PROBLEM

A workflow specification is usually defined with a set of constraints that introduces different relations and dependencies that must be managed either between subjects executing workflow tasks or between workflow tasks themselves. An important known requirement in WFMS is separation of duty, where two different users must execute two different tasks. Many other constraints can be defined with the workflow specification such as the binding of duty constraints, in which the same user is required to perform two different tasks. Satisfying all these constraints when executing the process can be a complex task in some cases and can even lead to the non accomplishment of the workflow or to the violation of some constraints especially if the constraint set is initially not consistent. Once the workflow specification is done, the workflow manager will be wondering about its execution and whether it is possible to execute the workflow without violating any constraint.

In this part we deal with calculating the complexity of the workflow satisfiability problem. To do so, we divide the problem into three problem statements. In the following we introduce these statements and we study their complexity. Let us consider these two definitions:

Definition 11: (Workflow) A workflow W is a set of n tasks {T1, T2…Tn} that must be executed in a specific order and satisfy a number m of constraints, where m and n are positive integers. Let us consider W = <{T1, T2, ..., Tn}, C> and C = {C1, C2, …, Cm}.

Definition 12: (Constraints) A constraint can be a relation conditioning the execution of two or more tasks in the workflow. These conditions can concern either roles or subjects executing tasks or also the execution order of these tasks. Such a set is called constraint base.

Problem statement 1:

1. Input: a set C of m constraints. C={C1, C2, …, Cm}
2. Question: Check if this set of constraints is consistent. It means that these constraints are satisfiable and they are not contradictory.
3. Complexity: this problem has a polynomial complexity in time ($O(m^3)$).
4. Proof: Let us consider this constraint base:
   - same_subject(T1, T2)
   - same_subject(T3, T6)
   - same_subject(T2, T3)
   - different_subject(T1, T6)

   We can easily notice that these constraints cannot be consistent since tasks T1 and T6 must be done simultaneously by different users (constraint 4) and by the same user (combining constraints 1, 2 and 3). To study the complexity of this problem, we suppose that a constraint base contains a set of relations and their opposite. First of all, we focus on one type of relation. Then the same work can be done for the rest of relation types.

   W = {T1, T2, .., Tn}, a workflow of n tasks.

   C = {ρ1(T1, T2), ρ2(T4, T6), ρ1(T3, T4), $\overline{\rho}$2(T3, Tn), $\overline{\rho}$1(T1, T4), $\overline{\rho}$1 (T1, T2), …, $\overline{\rho}$2 (T4, Tn)} a constraint base of m elements where ρi belongs to {same_subject, same_departement, different_team, …} and $\overline{\rho}$i is the opposite relation of ρi (for instance, same_subject and different_subject are two opposite relations).

   Let us consider Ci different sets which contain different relations ρi and $\overline{\rho}$i where i in [1..k]. We divide Ci into two groups G1 and G2 which contain respectively the positive (ρi) and the negative ($\overline{\rho}$i) relations. G1(i) and G2(i) indicate the relation number i in the group G1 or G2. Studying the intersection between these different relations, we can detect different conflicts that can be present between constraints. This detection mechanism of conflicts is defined by the algorithm below. The complexity of checking if a set of constraints is consistent is similar to the complexity of this algorithm. So it is a polynomial complexity in time ($O(m^3)$, m is the number of constraints).

   The functions First_arg() and Second_arg() used in the algorithm return respectively the first and the second argument of a binary relation in the constraint base. For example if we have the relation ρ = same_subject(T1, T4) we will have First_arg(ρ) = {T1} and Second_arg(ρ) = {T4}.

Problem statement 2:

1. Input: a workflow W of n tasks, a set of users and a set of m consistent constraints.
2. Question: can we find a valid assignment of users to different tasks which satisfies all constraints?
3. Complexity: this problem is NP-complete.
4. Proof: This problem is similar to the problem called Constraint Satisfaction Problem (CSP) defined as "We give a set of variables, a finite and discrete domain for each variable, and a set of constraints. Each constraint is defined over some subset of the original set of variables and limits the combinations of values that the variables in this subset can take. The goal is to find one assignment to the variables such that the assignment satisfies all the constraints" (Kumar, 1992).

Many optimized backtracking techniques are proposed to solve this problem. To proof that it is a NP-complete problem, we use a reduction of the problem to a colourability problem. We represent the workflow as a graph where its vertices represent different workflow tasks and its edges represent the constraints defined between the tasks connected by these edges. We associate each relation of the constraint base with a graph where we represent just this specific relation. So the whole graph representing the workflow will be represented as a set of sub graphs representing different relations of the constraint base. Each sub graph contains just the relation (ρi) and its opposite ($\overline{\rho}$i).

For example let us consider a workflow W = {T1, T2, T3, T4, T5, T6} and a constraint set containing these constraints:

- same_subject(T1, T5)
- same_subject(T3, T6)
- different_subject(T1, T2)
- different_subject(T1, T5)
- different_subject(T5, T6)

The Figure 4 represents the sub graph corresponding to the "same_subject" relation of the constraint set (we represent this relation using the symbol =). Then we transform this graph by gathering tasks that must be executed by the same subject in the same vertices. This transformation leads to the graph in Figure 5. We apply the same treatment to all relations in the constraint set. So, searching an assignment that does not violate these constraints can be expressed as a set of graph colourability problems which is known in its simplest form as a way of colouring the vertices of a graph such that no two adjacent vertices share the same colour. In this reduction, vertices in a graph are mapped to steps in the workflow, while colours are mapped to users. The graph colourability problem is NP-complete. So, finding a valid assignment of users to different tasks which satisfies all constraints is also NP-complete.

Problem statement 3:

1. Input: a workflow instantiation of n tasks and a set of m consistent constraints.
2. Question: This assignment satisfies all constraints or not?
3. Complexity: this problem has a polynomial complexity in time. (O(nxm), n the number of tasks and m the number of constraints).
4. Proof: To study the complexity of this problem, we represent the workflow as a deterministic turing machine. Different tasks represent different states of the machine. The assignment of different subjects to different tasks corresponds to the entry of the turing machine. At each state of the turing machine, we check if the assignment of the first subject to the task of this state

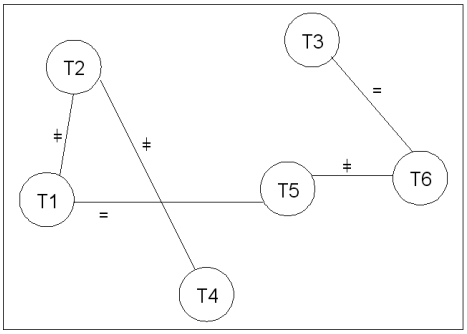

Figure 4: Initial graph of the "=" relation

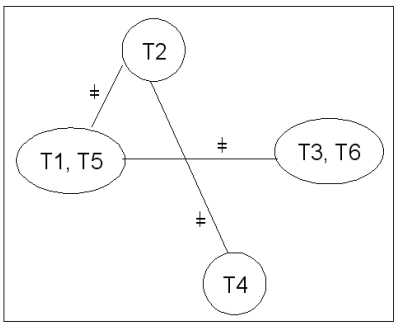

Figure 5: Reduction of the "=" relation graph

Algorithm 2: Main Algorithm

```
for l in index(ρ) do
    G1 ← ρ_l
    G2 ← ρ̄_l
    for k in 1,2 do
        i ← 1
        repeat
            S_k(i) ← First_arg(Gk(1)), Second_arg(Gk(1))
            Gk ← Gk \ Gk(1)
            j ← first_index(Gk)
            repeat
                Loop
                if (First_arg(Gk(j)) ∈ S_k(i)) or (Second_arg(Gk(j)) ∈ S_k(i)) then
                    S_k(i) ← S_k(i) ∪ {First_arg(Gk(j)), Second_arg(Gk(j))}
                    Gk ← Gk \ Gk(j)
                    j ← First_index(Gk)
                    GOTO Loop
                j ← next_index(Gk)
            until (j = last_index(Gk)) ;
            i ← i+1
        until (Gk = ∅) ;
    i ← 1
    coherence ← true
    repeat
        j ← 1
        repeat
            if (S1(i) ∩ S2(j) ≠ ∅) then
                coherence ← False
            j ← j + 1
        until (j = last_index(S2)) or (coherence = False) ;
        i ← i + 1
    until (i = last_index(S1)) or (coherence = False) ;
```

Algorithm 3: Main Algorithm

```
Entry_chain ← affectation_subjects
result ← True
repeat
    cursor ← i
    Affect (T_i, Entry_chain[1])
    Check (constraint_base)          – m operations
    if check(constraint_base) then
        Entry_chain ← Entry_chain \ Entry_chain[1]
    if not (check(constraint_base)) then
        result ← False
until (i = n)or (result=false) ;
```

satisfies all constraints associated with the workflow. Then this subject will be deleted from the entry and then the new entry will be passed to the next state of the machine. These steps are done until the end of the workflow. This functioning is described by the following algorithm. The complexity of this algorithm is polynomial (nxm; n: number of tasks and m: number of constraints). In each state we must do m checks with the constraint base. This operation is repeated n times, the number of states (equal to the number of tasks).

9. CONCLUSION

In this paper, we have presented a Petri net based model for modelling workflows and we have defined the security policy that we associate to it. This model and security policy are based on the OrBAC model. Thus, they reuse organization and context notions given in this access control model. Our security policy takes into account different possible execution modes of two tasks. It is composed of a general security policy, a coordination security policy and an information flow policy. In a second part, we have presented an algorithm allowing us to synchronize authorization flows with workflow execution. This algorithm defines how to execute the suggested model in a *distributed* WFMS environment. The last part of this work has dealt with the complexity of the workflow satisfiability problem. We have shown that, given a consistent constraint base, searching for an assignment of users to tasks that not violate all constraints is NP-complete. But checking if a given assignment does not violate the constraint set has a polynomial complexity. As part of future work, we will enrich our algorithm by handling information flows between different organizations. Indeed, organizations must exchange flows to have knowledge of what is happening globally in the system. These flows must be managed in order to keep a secure execution environment of the process. In fact, exchanging flows between organizations may imply a confinement problem (Boebert, 1996; Boebert, 1985). Thus, these exchanges have to be controlled in order to keep a secure environment of execution processes.

10. REFERENCES

ABOU EL KALAM, A., EL BAIDA, R., BALBIANI, P., BENFERHAT, S., CUPPENS, F., DESWARTE, Y., MIEGE, A., SAUREL, C. and TROUESSIN, G. (2003): Organization based access control. *IEEE 4th International Workshop on Policies for Distributed Systems and Networks*, Lake Como, Italy.

ADAM, N.R., ATLURI, V. and HUANG, W-K. (1998): Modeling and analysis of workflows using Petri nets. *Journal of Intelligent Information Systems, Special Issue on Workflow and Process Management*, 10.

ALLEN, J.F. (1993): Towards a general theory of action and time. *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*.

ATLURI, V. and HUANG, W. (1996): An authorization model for workflows. *Proceedings of the Fifth European Symposium on Research in Computer Security*, Rome, Italy, 44–64.

ATLURI, V. and HUANG, W-K. (1997): Enforcing mandatory and discretionary security in workflow management systems. *Journal of Computer Security*, 5(4):303–339.

ATLURI, V., HUANG W-K. and BERTINO, E. (2000): A semantic based execution model for multilevel secure workflows. *Journal of Computer Security*, 8(1).

AYED, S., CUPPENS-BOULAHIA, N. and CUPPENS, F. (2007): An integrated model for access control and information flow requirements. *Proceedings of the Twelfth Annual Asian Computing Science Conference Focusing on Computer and Network Security*, Doha, Qatar.

BADGER, L., STERNE, D.F., SHERMAN, D.L., WALKER K.M. and HAGHIGHAT, S.A. (1995): Practical domain and type enforcement for Unix. IEEE *Symposium on Security and Privacy*, Oakland, CA, USA.

BERTINO, E., FERRARI, E. and ALTURI, V. (1999): The specification and enforcement of authorization constraints in workflow management systems. *ACM Transactions on Information and System Security*.

BOEBERT, W. E., KAIN, R. Y. and YOUNG, W. D. (1985): The extended access matrix model of computer security. *ACM Sigsoft Software Engineering Notes*, 10(4).

BOEBERT W.E. and KAIN R.Y. (1996): A further note on the confinement problem. *Proceedings of the IEEE 1996 International Carnahan Conference on Security Technology*, New York: IEEE Computer Society.

CUPPENS, F. and MIEGE, A. (2003): Modelling contexts in the Or-BAC model. *19th Annual Computer Security Applications Conference*, Las Vegas.

CUPPENS, F., CUPPENS-BOULAHIA, N. and MIEGE, A. (2004): Inheritance hierarchies in the Or-BAC model and application in a network environment. *Second Foundations of Computer Security Workshop (FCS'04)*, Turku, Finlande.

CUPPENS, F., CUPPENS-BOULAHIA, N. and SANS, T. (2005): Nomad: A security model with non atomic actions and deadlines. *18th IEEE Computer Security Foundations Workshop (CSFW'05)*, Aix-en-Provence, France.

DAVID, R, and ALLA, H. (1992): *Du grafcet aux réseaux de Petri*, Hermμes. Paris, 2nd edition.

DEMURJIAN, S. (2000): Implementation of mandatory access control in role-based security system. CSE367 Final Project report.

FERRAIOLO, D. and KUHN, R. (1992): Role-based access controls. *15th National Computer Security Conference*, 554–563.
HUANG, W. and ATLURI, V. (1999): SecureFlow: A secure web-enabled workflow management system. *Proceedings of the fourth ACM workshop on Role-based access control*, 83–94, Fairfax, Virginia, United States.
HUNG, P. and KARLAPALEM (1999): A secure workflow model. *AISW (Australian Information Security Workshop)*.
KISZKA, J. and WAGNER, B. (2003): Domain and type enforcement for real-time operating systems. *Proccedings ETFA '03, Emerging Technologies and Factory Automation*.
KUHN, R.D. (1998): Role based access control on MLS systems without kernel changes. *Proceedings of the third ACM Workshop on Role-Based Access Control*, 25–32. Fairfax, Virginia, United States.
KUMAR, V. (1992): Algorithms for constraint-satisfaction problems: A Survey. *AI Magazine*, 13(1).
NYANCHAMA, M. and OSBORN, S. (1994): Information flow analysis in role-based security systems. *International Conference on Computing and Information, Proc. of the 6th International Conference on Computing and Information (ICCI)*, 1368-1384, Peterborough, Ontario, Canada.
NYANCHAMA, M. and OSBORN, S. (1996): Modeling mandatory access control in role-based security systems. *IFIP Workshop on Database Security*, New York, United States.
OSBORN, S. (1997): Mandatory access control and role-based access control revisited. *Proceedings of the second ACM workshop on Role-based access control*, 31–40, Fairfax, Virginia, United States.
OSBORN, S., SANDHU, R. and MUNAWER, Q. (2000): Configuring role-based access control to enforce mandatory and discretionary access control policies. *ACM Transactions on Information and System Security*, 3(2):85–106.
QIHUA, W. and NINGHUI, L. (2007): Satisfiability and resiliency in workflow systems. 90–105. In *Proc. Of ESORICS 2007*, 12th European Symposium On Research In Computer Security, Dresden, Germany.
SANDHU, R. (1996): Role hierarchies and constraints for lattice-based access controls. *Proc. Fourth European Symposium on Research in Computer Security*, Rome, Italy.
SANDHU, R.S., COYNE, E.J., FEINSTEIN, H.L. and YOUMAN, C.E. (1996): Role-based access control models. *Computer*, 29(2).
SANDHU, R. and MUNAWER, Q. (2000): How to do discretionary access control using roles. In *Proc. of the 3rd ACM Workshop on Role Based Access Control (RBAC-98)*, Fairfax, VA, USA.
WORKFLOW MANAGEMENT COALITION (1995): The workflow reference model. WFMC-TC-1003.
WORKFLOW MANAGEMENT COALITION (1998): Workflow security considerations. White Paper. Document number WFMC-TC-1019, Document Status – Issue 1.0.

BIOGRAPHICAL NOTES

Samiha Ayed

Samiha Ayed obtained her Masters degree in networks from the National Higher School in Computer Science of Tunisia in 2006. Since 2006, she has been a PhD candidate at TELECOM Bretagne, and is also a member of the security team SERES in the department RSM where she has done research for her thesis in the field of security of information systems. Her research interests include especially studying (specifying, managing, deploying, etc.) access and flow control within Workflow Management Systems.

Nora Cuppens-Boulahia

Nora Cuppens-Boulahia is a teaching researcher at the TELECOM Bretagne. She holds an engineering degree in computer science, a PhD from ENSAE and an Habilitation thesis. Her research interest includes formalization of security policies and security properties, cryptographic protocol analysis and formal validation of security properties. She is giving courses in various domains of computer security including network, operating systems and database security, risk analysis and security evaluation criteria. She has published more than 50 technical papers in refereed journals and conference proceedings. She shares the responsibility of "Information system security" pole of the SEE and she is the French representative at IFIP TC11 « Information Security ».


Frédéric Cuppens

Frédéric Cuppens is a full professor at the TELECOM Bretagne. He holds an engineering degree in computer science, a PhD and an Habilitation thesis. He has been working for more than 15 years on various topics of computer security including definition of formal models of security policies, access control to network and information systems, intrusion detection and formal techniques to deploy and analyze security policies and prove security properties. He has published more than 100 technical papers in refereed journals and conference proceedings. He served on several conference program committees and was the Program Committee Chair of ESORICS 2000 and IFIP SEC 2004.