

# A GTRBAC Based System for Dynamic Workflow Composition and Management

Basit Shafiq, Arjmand Samuel, and Halima Ghafoor  
Purdue University  
{shafiq, amsamuel}@ecn.purdue.edu

## Abstract

*In this paper, we propose an architecture for adaptive real-time workflow-based collaborative system. Such a system is needed to support real-time communication and sharing of information among predefined or ad hoc team of users collaborating with each other for the execution of their respective tasks in the workflow. A key requirement for real-time workflow system is to provide the right data to the right person at the right time. In addition, the workflow needs to be reconfigured if a subtask of a workflow cannot be executed within the due time. We use the generalized temporal role-based access control (GTRBAC) model to capture the real-time dependencies of such workflow applications. In addition, support for triggers in GTRBAC allows dynamic adaptation of workflow based on the occurrence of certain events. Such adaptations may include rescheduling of workflow tasks, reassignment of users to scheduled tasks based on their availability and skill level, and abortion of incomplete tasks.*

## 1. Introduction

Technological advancement in networking and databases have enabled development of distributed workflow based applications in various domains including e-commerce, digital government, healthcare, power systems, air traffic control, manufacturing and many others. Workflow applications in these domains are not restricted to the administrative boundaries of a single organization and may require inter-organization information and resource sharing for the execution of the tasks comprising the workflow [1, 6, 8]. The tasks in a workflow need to be performed in a certain order and often times are subject to real-time constraints and dependencies [5, 4, 3]. Real-time constraints may be in the form of strict deadlines that must be met by the system to operate safely. Violations of these deadlines may cause malfunctioning of the system and in some cases may imply loss of human life and health. For

instance, in air traffic control systems, any delay in the transfer of flight parameters from the local air traffic control to the zonal flight control may have adverse affect on air space management at the zonal level. On the other hand, in certain workflow applications systems such as on-line banking, flight reservation, and inventory control occasional violation of real-time constraints can be tolerated. A key requirement for real-time workflow systems is to provide the right data to the right person at the right time. This requirement motivates for dynamic adaptations of workflows. For instance, a pre-assigned user for a given workflow task may not be able to complete the task within due time. Consequently, the workflow needs to be adapted to deal with such exceptions.

For any workflow instance, it is essential that the underlying workflow tasks are executed by authorized users only. Therefore, appropriate access control mechanisms need to be employed to meet this requirement. Traditional access control models such as Discretionary and Mandatory Access Control (DAC and MAC) lack the capability to capture the time-based dependencies of workflow applications. The recently proposed Generalized Temporal Role-Based Access Control (GTRBAC) [7] model provides a suitable approach for specification of security and access control requirements of real-time workflow applications. GTRBAC uses the abstraction of roles for specification of authorization constraints. A role is a collection of permissions needed to perform a certain task or a job function. Users are assigned appropriate roles based on their qualifications and responsibilities. In the context of workflow, a given task is associated with one or more roles and can only be executed by users or agents authorized for the corresponding roles. Since in any application domain, the number of roles is generally smaller than the number of users, therefore use of role-based authorization considerably reduce the management overhead in terms of policy specification. In addition, role based authorization is particularly beneficial in workflow environments in facilitating

dynamic load balancing when a task can be performed by several users [2]. However, the most distinguishing feature of GTRBAC is the support for temporal constraints which is essential for modeling the real-time dependencies. Additionally, GTRBAC allows specification of separation of duties (SoD), cardinality, and event dependency constraints that are required in many workflow based applications.

In this paper, we propose an architecture for dynamic composition of workflows with real-time constraints. The architecture supports GTRBAC based workflow specification and allows dynamic adaptation of active workflows depending on the execution status of workflow tasks and environmental context. Adaptations in a workflow may include rescheduling of certain workflow tasks, reassignment of users to the scheduled tasks based on their availability, authorization, and skill level, or abortion of certain tasks that cannot be completed under the current system state. Several possibilities may exist for adaptation. The workflow adaptation/configuration module is responsible for finding the best possible adaptation according to some pre-defined optimality criterion.

The remainder of the paper is organized as follows. In Section 2, we present a brief overview of GTRBAC model and in Section 3 we describe the proposed software architecture for dynamic composition and management of adaptive workflows. Section 4 provides an illustrative example for explaining the usability of proposed architecture. Section 5 concludes the paper and presents some future research directions.

## 2. Overview of GTRBAC Model

In this section, we provide relevant background on the RBAC and GTRBAC models that we refer to in this paper. The RBAC model as proposed by Sandhu *et. al.* in [9], currently being used as the basis for the NIST RBAC model, consists of the following four basic components: a set of users Users, a set of roles Roles, a set of permissions Permissions, and a set of sessions Sessions. A user is a human being or a process within a system. A role is a collection of permissions associated with a certain job function within an organization. A permission is an access mode that can be exercised on a particular object in the system. A session relates a user to possibly many roles. When a user logs in the system he establishes a session by activating a set of enabled roles that the user is entitled to activate at that time. If the activation request is satisfied, the user issuing the request obtains all the permissions associated with requested role. On Roles, a hierarchy is defined, denoted by  $\geq$ . If  $r_i \geq r_j$ ,  $r_i, r_j \in$

Roles then  $r_i$  inherits the permissions of  $r_j$ . In such a case,  $r_i$  is a senior role and  $r_j$  a junior role.

The RBAC model does not explicitly model different states of a role and hence does not capture various events that are typical of an RBAC system. Such event based approach was used by Bertino *et. al.* in TRBAC model [3] and later extended by Joshi *et. al.* in GTRBAC [7], primarily to capture the different transitional actions needed in the context of temporal constraints. The GTRBAC model provides a temporal framework for specifying an extensive set of temporal constraints and uses a language-based framework [7]. GTRBAC allows various types of temporal constraints such as *temporal constraints on role enabling/disabling, temporal constraints on user-role and role-permission assignments/de-assignments, role activation-time constraints, etc.* In addition support for event triggers in GTRBAC allows specification of various dependency and precedence constraints. These constraints are useful in capturing the dynamic behavior of emerging workflow based applications.

The temporal constraints in GTRBAC are expressed by a generic form  $(I, P, D, E)$ , where  $I$  is an interval and  $P$  is a set of infinite intervals.  $(I, P)$  represents the set of all the intervals of  $P$  that are contained in  $I$ . For example,  $(I, P) = ([1/1/2005, 12/31/2005], \text{Mondays})$  consider all the *Mondays* of the year 2005.  $D$  specifies the duration in which the event  $E$  is valid. The trigger expression has the form  $E_1, \dots, E_n, C_1, \dots, C_k \rightarrow pr:E$  after  $\Delta t$ , where  $E_i$ 's are simple event expressions or run time requests,  $C_i$ 's are status predicates,  $pr:E$  is a prioritized event expression, and  $\Delta t$  is a duration expression.. The following example illustrates the GTRBAC specification of an access control policy of a medical information system.

**Example:** Consider the GTRBAC access control policy of Table 1, from a medical information system. In row 1A, the enabling times of DayDoctor and NightDoctor roles are specified as a periodicity constraint. The  $(I, P)$  forms for *DayTime* (9am-9pm) and *NightTime* (9pm-9am) are as follows: *DayTime* =  $([12/1/2005, \infty], \text{all.Days}, + 10.Hours \triangleright 12.Hours)$ , and *NightTime* =  $([12/1/2005, \infty], \text{all.Days}, + 12.Hours \triangleright 12.Hours)$ .

Referring to table 1, row 1B states that, *Adams* is assigned to role DayDoctor on *Mondays, Wednesdays* and *Fridays*, whereas *Bill* is assigned to it on *Tuesdays, Thursdays, Saturdays* and *Sundays*. The assignment in 1C indicates that *Carol* can assume the DayDoctor role everyday between 10am and 3pm. In 2A, users *Ami* and *Elizabeth* are assigned roles NurseInTraining and DayNurse respectively, without any periodicity or duration constraints. In other words, their assignments are valid at all the times.

**Table 1. Example GTRBAC access policy for a medical information system**

	Policy Specification	Explanation
1	A ( <i>DayTime</i> , enable DayDoctor), ( <i>NightTime</i> , enable NightDoctor)	Enable DayDoctor at <i>DayTime</i> , Enable NightDoctor at <i>NightTime</i>
	((M, W, F), assign <sub>U</sub> Adams to DayDoctor), ((T, Th, S, Su), assign <sub>U</sub> Bill to DayDoctor),	User Adams is assigned the role of a DayDoctor on Monday, Wednesday and Friday, Similarly user Bill is assigned the role of DayDoctor on Tuesday, Thursday, Saturday and Sunday.
	(Everyday between 10am - 3pm, assign <sub>U</sub> Carol to DayDoctor)	User Carol is assigned the role of DayDoctor everyday from 10 PM to 3 PM
2	(assign <sub>U</sub> Ami to NurseInTraining); (assign <sub>U</sub> Elizabeth to DayNurse)	User Elizabeth is assigned the role DayNurse and user Ami is assigned role NurseInTraining
	$c1 = (6 \text{ hours}, 2 \text{ hours}, \text{enable NurseInTraining})$	The role NurseInTraining is enabled after every 6 hours for 2 hours
3	(enable DayNurse $\rightarrow$ enable $c1$ )	When DayNurse role is enabled, constraint $c1$ is also applied
	(activate DayNurse for Elizabeth $\rightarrow$ enable NurseInTraining after 10 min)	When the role of DayNurse is activated, NurseInTraining role is enabled after 10 minutes
	(enable NightDoctor $\rightarrow$ enable NightNurse after 10 min); (disable NightDoctor $\rightarrow$ disable NightNurse after 10 min)	When the NightDoctor role is enabled, the Night nurse is also enabled after 10 minutes; similarly, disabled
4	(10, active <sub>R_n</sub> DayNurse);	At the most 10 active users in the role DayNurse
	(5, active <sub>R_n</sub> NightNurse);	At the most 5 active users in the role NightNurse
	(2 hours, active <sub>R_total</sub> NurseInTraining)	Nurse in training role can be activated for a total of 2 hours

Row 2B in Table 1 specifies a duration constraint of 2 hours on the enabling time of the NurseInTraining role, but this constraint is valid for only 6 hours after the constraint  $c1$  is enabled. Consequently, once the NurseInTraining role is enabled, *Ami* will be able to activate the NurseInTraining role at the most for two hours.

Trigger 3A indicates that constraint  $c1$  is enabled once the DayNurse is enabled. As a result, the NurseInTraining role can be enabled within the 6 hours. Trigger 3B indicates that 10 minutes after *Elizabeth* activates the DayNurse role, the NurseInTraining role is enabled for a period of 2 hours. As a result, a nurse in training can then have access to the system only if *Elizabeth* is present in the system. In other words, once the roles are assumed, *Elizabeth* acts as a training supervisor for a nurse in training. It is possible that *Elizabeth* activates the DayNurse role a number of times within 6 hours after the DayNurse role is enabled. The activation constraint 4C limits the total activation time associated with the NurseInTraining role to 2 hours. The constraint set 4 shows additional activation constraints. For example,

constraint 4A indicates that there can be at most 10 users activating DayDoctor role at a time, whereas 4B shows that there can be at most 5 users activating the NightDoctor role at a time.

### 3. Software Architecture

Figure 1 depicts the proposed software architecture for dynamic workflow composition and management. The architecture consists of three key components, including, workflow composition module, workflow management module, and access control module.

#### 3.1. Workflow Composition Module

The workflow composition module (WCM) provides an authoring tool for specification of workflow tasks and the interdependencies between these tasks. In addition, other dynamic constraints including separation of duties and task execution cardinality constraints can be specified for the underlying workflow tasks. GTRBAC formalism is used to specify all the workflow constraints. The

consistency analyzer component in WCM is responsible for checking the consistency and the correctness of the composed workflow in terms of task dependencies, deadlines, and constraint conflicts.

### 3.2. Access Control Module

The access control module (ACM) is responsible for determining the authorization of users for execution of workflow tasks. The authorizations of users are determined based on their assigned roles. This assignment may be pre-defined in the access control policy or may be performed dynamically based on users' credentials and the context parameters. The credentials facilitate in authentication process and are supplied by the user upon the request of ACM. In addition, user credentials also help in determining the qualification and skill level of the user. The context parameters may be specific to the user, such as user location, time of access, and the current resource capacity. Additionally, environmental context such as system load and execution status is also considered in determining the user assignment. The user-specific context is extracted from the information supplied by the user at the time of access request and the environmental context is provided by the state monitoring module. In a distributed workflow environment users executing workflow tasks may belong to different organizations or administrative domains. To enable collaboration in such workflow environment, dynamic role mapping is needed that defines the relationship between the roles assumed by the users in their own organizations and the roles assigned to users for workflow execution. The role mapping/assignment component in ACM is also responsible for creating such mapping.

### 3.3. Work Flow Management Module

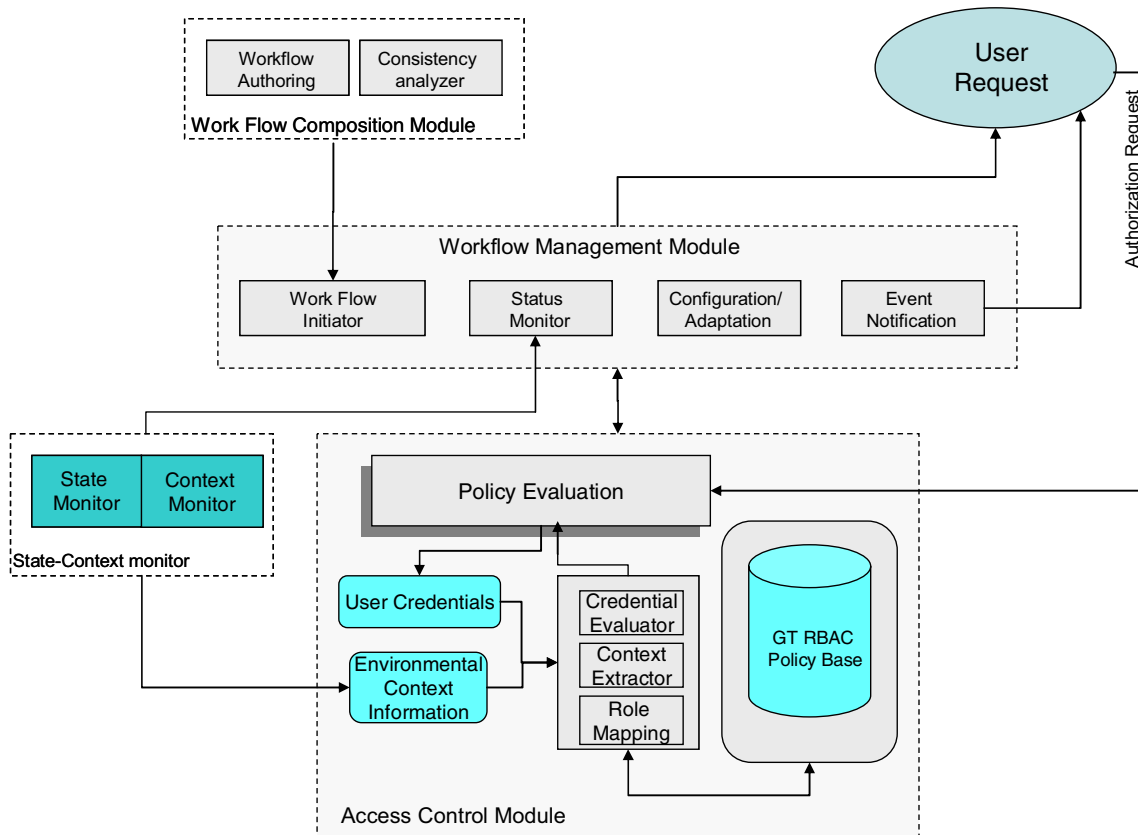
The workflow management module (WMM) is the most important component of the proposed architecture. It consists of following key sub-components: workflow invocation component, execution status monitor, event notification component, and workflow reconfiguration/adaptation component. The workflow invocation component in WMM is responsible for instantiation of a workflow upon the request of authorized user. The authorization of workflow instantiation request is determined by ACM. ACM also performs user to role assignment for the different tasks of the instantiated workflow. Upon receiving the authorization approval the workflow invocation component creates an instance of the

workflow with the user to role bindings for the associated workflow tasks. After the instantiation of the workflow, the corresponding users are notified for the execution of the tasks assigned to them. The event notification component is responsible for sending out such notification messages to the corresponding users at the appropriate time.

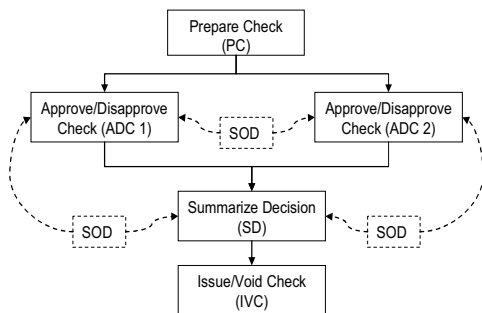
The execution of an instantiated workflow may not proceed as planned in the invocation phase. Changes in the environmental or user context or the occurrence of certain unpredictable events may block the execution of some tasks in the workflow. Consequently, the workflow needs to be reconfigured for execution of blocked tasks. The workflow adaptation/reconfiguration component is responsible for such adaptations. Workflow adaptation is triggered by the status monitor component which continuously checks the execution status of all active and pending tasks. Adaptations in a workflow may include rescheduling of certain workflow tasks, relaxation of policy constraints, reassignment of users to the scheduled tasks based on their availability, authorization, and skill level, or abortion of certain tasks that cannot be completed under the current system state. Depending on the execution status of workflow tasks and environmental context, several adaptation possibilities may exist for an active workflow. The adaptation/reconfiguration component need to find an adaptation that optimizes the overall performance under the given constraints.

## 4. Illustrative Example

In this section, we illustrate the usability of the proposed architecture using an example of a tax refund workflow process with dynamic dependencies and temporal constraints. Figure 2 shows the different tasks of the workflow and the relative order in which these tasks are executed. The workflow has following four tasks, namely: prepare check (PC), approve/disapprove check (ADC), summarize decision (SD), and issue/void check (IVC). To prevent any fraud, separation of duty (SoD) constraint is defined between the tasks approve/disapprove check and summarize decision, implying that the same person cannot perform these two tasks for a given workflow instance. In addition, check needs to be approved by two different managers, which is represented in the workflow of Figure 2 by two parallel tasks ADC1 and ADC2 with a SoD constraint ( $SOD(ADC1, ADC2)$ ) defined between them.



**Figure 1. Software architecture for dynamic workflow composition and management**



**Figure 2. Tax refund process workflow**

The constraint specification in GTRBAC formalism for the workflow of Figure 2 is listed in Table 2. This workflow specification with the associated GTRBAC constraints is composed in the workflow composition module of Figure 1.

The authorizations for different tasks of tax refund workflow are specified in the access control policy of the organization dealing with tax refunds. This access control policy is defined using GTRBAC model with four roles: General Manager (GM), Technical Manager (TM), Refund Manager (RM), and Refund Clerk (RC) and five users:  $u_1$ ,  $u_2$ ,  $u_3$ ,  $u_4$ , and  $u_5$ . Users  $u_1$  and  $u_2$  are

conflicting users for role GM, implying that  $u_1$  and  $u_2$  cannot assume these roles simultaneously in the same workflow instantiation. This constraint is specified using the predicate  $USOD(u_1, u_2, GM)$  in Table 3.  $u_1$ ,  $u_2$ , and  $u_3$  are conflicting for TM and  $u_2$  and  $u_4$  are conflicting for RM. The access control policy also has dependency constraints on the activation of the roles. The first dependency constraint is for user  $u_1$ , for the roles GM and RM. According to this dependency constraint user  $u_1$ , can activate role TM only if  $u_1$ , has activated the role GM. Similarly, the second dependency constraint prohibits user  $u_2$  to activate role RM without assuming the role GM. All the hierarchy, SoD, and dependency constraints are specified in GTRBAC formalism in Table 3. This access control policy is stored in the policy base of the access control module.

The tax refund workflow is instantiated with the preparation of the refund check by a user assuming role RC. With this instantiation of the workflow, all the user to role assignments for the remaining tasks are performed by the access control module. The execution of the instantiated workflow may result in a state in

**Table 2. GTRBAC specification of workflow constraints**

1	a	(enable PC $\rightarrow$ enable ADC 1 after 2 minutes for 4 minutes)
	b	(enable PC $\rightarrow$ enable ADC2 after 2 minutes for 4 minutes)
	c	(enable ADC 1 $\wedge$ enable ADC 2 $\rightarrow$ enable SD after 4 minutes for 4 minutes);
	d	(enable IVC $\rightarrow$ for 5 minutes);
2	SOD(ADC1, ADC2), SOD(ADC1, ADC2, SD), SOD(ADC1, ADC2, PC), SOD(ADC1, ADC2, IVC) and SOD(SD, PC), SOD(SD, IVC)	

**Table 3. GTRBAC specification of access control policy**

1	GM $\geq$ TM; GM $\geq$ RM; RM $\geq$ RC;	Hierarchal Constraints
2	USOD( $u_1, u_2, u_3, TM$ ); USOD( $u_1, u_2, GM$ ); USOD( $u_2, u_4, RM$ ); USOD( $u_1, u_2, TM$ ); USOD( $u_1, u_3, TM$ );	SOD Constraint
3	(Activate GM by $U_1 \rightarrow$ Activate TM by $U_1$ ); (Activate GM by $U_2 \rightarrow$ Activate RM by $U_2$ );	Dependency Constraint

which some of the active or pending tasks may not be completed. For instance, consider a workflow instantiation in which  $u_1$  assuming the role GM performs the task ADC1 and  $u_4$  assuming the role RM performs the task ADC2. In this particular workflow instantiation, no other users can assume any of the manager roles (GM, RM, TM) because of the SoD constraints specified in the underlying access control policy. The workflow instance cannot be completed because of the unavailability of a user with a manager role to execute the task SD. Note that none of the user  $u_1$  and  $u_4$  can execute the task SD because of the workflow constraint prohibiting same user to approve check and summarize the final decision. At this point the state monitor will detect the deadlock in the workflow and will trigger the re-configuration/adaptation sub module. A number of workflow reconfiguration steps can be taken to successfully complete or terminate the workflow. One possibility could be the abnormal abortion of the workflow. This option may not be desirable depending on the underlying workflow application. Another option is to relax the SoD constraint between  $u_1$  and  $u_3$  for the role TM. This would allow  $u_3$  to execute the blocked task SD. In this case, the access control module needs to assign user  $u_3$  to the role TM for execution of task SD. Yet another option for reconfiguration is to bypass the task ADC1 and let the user  $u_1$  to execute task SD directly. This reconfiguration amounts to partial completion of the workflow instance with the abortion of blocked task. There can be other possibilities for workflow reconfiguration which are not discussed because of space limitations. The reconfiguration/adaptation module must select the optimal solution amongst the many possible reconfiguration choices.

## 5. Conclusion

We have presented a software architecture for composition and management of adaptive real-time workflows. We use GTRBAC model to capture the dependencies and temporal constraints of such workflow specifications. In addition the organization policies defining the access control and security policy can also be specified using GTRBAC formalism. The authorizations of the users for execution of the instantiated workflow tasks are determined based on the access control policy. The system also supports dynamic assignment of workflow tasks to users based on their credentials and context parameters. The architecture also allows dynamic adaptations of instantiated workflows based on the execution status of workflow tasks, system state, and environmental context. There can be several possibilities for workflow adaptation and a selection mechanism is needed for optimal adaptation, which needs further research.

## 6. References

- [1] Richard D. Holowczak, S. A. Chun, F. J. Artigas, V. Atluri, "Applications: Customized Geospatial Workflows for E-Government Service," in *Proc. of the 9<sup>th</sup> ACM International Symposium on Advances in Geographic Information Systems*, Nov. 9 – 10, 2001, pp. 64 – 69.
- [2] E. Bertino, E. Ferrari, and V. Atluri, "The Specification and Enforcement of Authorization Constraints in Workflow Management Systems," *ACM Transactions on Information and System Security*, Vol. 2, No. 1, Feb. 1999, pp. 65-104.

- [3] E. Bertino, P. A. Bonatti, E. Ferrari, "TRBAC: A Temporal Role-based Access Control Model," *ACM Transactions on Information and System Security*, Vol. 4, No. 3, Aug. 2001, pp.191-233.
- [4] E. Chang, E. Guatama, and T.S. Dillon, "Extended Activity Diagrams for Adaptive Workflow Modelling," in *Proc. of the 4<sup>th</sup> International Symposium on Object-Oriented Real-Time Distributed Computing*, May 2001, pp. 413 – 419.
- [5] K. A. Delic. L. Douillet, and U. Dayal, "Towards an Architecture for Real-Time Decision Support Systems: Challenges and Solutions," in *Proc. International Symposium on Database Engineering and Applications*, 2001, pp. 303 – 311.
- [6] H. A. James, K. A. Hawick, and P. D. Coddington, "An Environment for Workflow Applications on Wide-Area Distributed Systems," in *Proc. Hawaii International Conference on System Sciences*, January 2001.
- [7] J. B. D. Joshi, E. Bertino, U. Latif, and A. Ghafoor, "Generalized Temporal Role-Based Access Control Model," *IEEE Transaction on Knowledge and Data Engineering*, Vol. 17, No. 1, January 2005, pages. 4 - 23.
- [8] S. K. Shrivastava, S. M. Wheeler, "A Transactional Workflow based Distributed Application Composition and execution Environment," in *Proc. of the 8<sup>th</sup> ACM SIGOPS European Workshop on Support for Composing Distributed Applications*, 1998, pp. 74 – 81.
- [9] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-Based Access Control Models," *IEEE Computer*, Vol. 29, Issue 2, Feb. 1996, pp. 38 – 47.