Locale-Based Access Control: placing collaborative authorization decisions in context *

William J. Tolone
Department of Software and Information System
University of North Carolina at Charlotte
Charlotte, NC, U.S.A
wjtolone@uncc.edu

Robin A. Gandhi
Department of Software and Information System
University of North Carolina at Charlotte
Charlotte, NC, U.S.A.
rgandhi@uncc.edu

Gail-Joon Ahn
Department of Software and Information System
University of North Carolina at Charlotte
Charlotte, NC, U.S.A.
gahn@uncc.edu

Abstract - *Collaboration systems require an appropriate authorization model to specify and maintain policies that not only facilitate group activities but also enforce restrictions and accountability. Existing models fail to incorporate adequately into authorization decisions the rich notion of context that is inherent to any collaborative setting. In this paper we present the Locale-based Access Control (Locale-BAC) model for collaborative systems, a model whose design is based upon the application of Fitzpatrick's Locale Framework for collaboration to the problem of access control. This model encapsulates the notion of context using locales, allowing for a natural representation of collaborative authorization decisions.*

Keywords: Access Control, Locale Framework, and Collaborative Systems.

1 Introduction

Collaborative systems have long recognized the usefulness of role-based specification of collaborative activities. The use of roles within collaborative systems, however, often is not driven by security requirements, but rather by the complexities of collaborative activity specification and maintenance – i.e., many collaborative activities evolve more slowly than user participation in those activities. In fact, during the research and design process, collaborative systems traditionally pay little attention to security issues. This is further complicated by the fact that often the goals of collaborative systems and security are competing. Collaborative systems emphasize the building of connections among people, tools, and applications; while security emphasizes the building and enforcing of boundaries in order to ensure confidentiality, integrity, and availability.

In [19], we survey existing solutions for collaborative access control. From preliminary work in access matrix and role-based models to recently proposed frameworks, traditional access control models have been extended using a variety of concepts. One of the most significant characteristics of these efforts is an emerging recognition of the importance of utilizing contextual information in authorization decisions. Context is one of the most defining attributes of collaborative environments because it encapsulates not only all types of environment variables (participants, resources, tasks, etc.) but also the dynamism and unpredictability associated with them. This, as well as the fact that context also embodies parameters such as time, place, presence, awareness, etc., is leading designers to incorporate broader notions of context into access control models for collaborative systems.

While the concept of role-based access control is well accepted, context introduces another level of consideration because a user in different roles could be active in different contexts and these contexts could change with his/her location, presence of other users in the same location or remote location, the roles in which other users are present and active, etc. Such a view finds strong support within social science communities. For example, Suchman argues that one cannot effectively separate actions from the context in which they are performed without losing the meaning or implications of those actions [16].

Covington et. al. [3] introduce the notion of environment roles in order to provide for security in context-aware applications. These roles are activated based on environment conditions at the time of request. Such roles have been shown useful in ubiquitous computing where environment sensitive information is pervasive. However, it is insufficient to state that a user can perform an operation only if s/he is active in a particular role. Rather, a richer notion of context is needed. For instance, it may be the case that the user can perform the operation only if his/her current context includes (or possibly does not include) certain people, tools, resources, etc. For example, a professor might be able to read/view a grade sheet within a shared collaboration space as long as no students are simultaneously present in that same space.

* 0-7803-7952-7/03/$17.00 © 2003 IEEE.

To facilitate authorization decisions based on a richer notion of context required by collaborative systems, we propose a new access control model called Locale Based Access Control (Locale-BAC). This model is grounded in Fitzpatrick's Locale Framework [5], a multi-faceted collaboration framework that, among other things, provides insight to the system designer in regards to the multi-dimensional nature of collaboration. We believe the Locale Framework offers a rich basis for describing collaboration context and, consequently, it guides our efforts to develop a more effective authorization model for collaborative systems.

Thus, in this paper we introduce the notion of a locale as a means of extending and relating sessions in RBAC [12] to provide a richer representation of context for authorization decisions. In addition, we present the formal definition of Locale-BAC and demonstrate its usefulness by newly identified constraints that are naturally expressible within Locale-BAC and required by many collaborative environments. Examples of such constraints include the *principle of all privileged* and the *principle of greatest authority*. We conclude with example scenarios that demonstrate the basics of Locale-BAC as well as these newly identified constraints.

2 Related Work

In this section, we provide a brief overview of existing approaches to access control for collaborative systems. A significant characteristic of several of these approaches is an emerging recognition of the importance of utilizing contextual information in authorization decisions.

By far the most prevalent approach to collaborative access control is models based on Access Matrices, e.g., [4], [10], [8], and [14]. Motivated by simplicity, access matrices provide an elegant model for associating subjects to permissions. At the same time, access matrices unfortunately incorporate very limited notions of context into authorization decisions. As a result, these models artificially restrict the utilization of context with collaborative systems. That is, the definition of context within a collaborative system is bounded by the limitations of access matrices.

Collaborative systems have long recognized the usefulness of role-based specifications of collaborative activities (e.g., MPCAL [7], ICICLE [1], SUITE [13], ConversationBuilder [9], PREP [11], and WORLDS [20]). Role-based Access Control (RBAC) Models [12] provide a slightly better notion of context than access matrices. Within RBAC, users exercise permissions in the context of an "active" role. An active role is one that is associated with a user's active session. From this perspective, model elements of roles in sessions begin to introduce a notion of context into authorization decisions. Yet, this notion of context remains weak since the scope of context in RBAC is limited to a single user.

Others have extended RBAC in order to define broader notions of context. Team-based Access Control (TMAC) [17] extends the notion of subject to groups of users, i.e., teams. Context-based TMAC (C-TMAC) [7] goes a step farther by introducing a context object that is associated with teams of users. This context object contains information such as time and place that are relevant to team definition. Covington et. al. [3] extend RBAC with the notion of environment roles in order to provide a broader notion of context within context-aware applications.

Other, more unique efforts to define access control for collaborative systems include Task-Based Access Control (TBAC) [18] and SPACE Access Control [2]. Access control within TBAC is granted in steps that are related to the progression of tasks. The SPACE model utilizes the notions of boundaries and access graphs to capture access capabilities among related collaborative environments.

While many of these efforts recognize the importance of context for authorization decisions, each fails to capture the multi-dimensional nature of context as expressed by the Locale Framework – TBAC focuses on process, TMAC and C-TMAC focus on groups, SPACE and Covington focus on organizational context. In the following section, we overview the Locale Framework as we believe it provides a rich, multi-dimensional view of the inherent role of context within collaborative activities, and therefore, collaborative access control.

3 Locale Framework

To facilitate authorization decisions based on a rich notion of context required by collaborative systems, we are developing a new model for access control called Locale Based Access Control (Locale-BAC). This model is grounded in Fitzpatrick's Locale Framework [5].

The purpose of the Locale Framework is multi-facetted. First, a locale provides a window by which to view, and therefore better understand, the interaction needs of groups of users committed to collection action (c.f., Straus' social worlds [15]). Second, a locale provides basis for characterizing the site and means utilized for collective action. As such, the framework serves as a bridge between communities and their environments providing a pathway for characterizing and understanding the mutual influence of community and environment (see Figure 1). This understanding impacts not only how we understand collective action but also informs the design of site and means for collective action. It is from this perspective that we ground our efforts to develop a more

effective authorization model for access control. In the following, we briefly outline the principles of the Locale Framework.

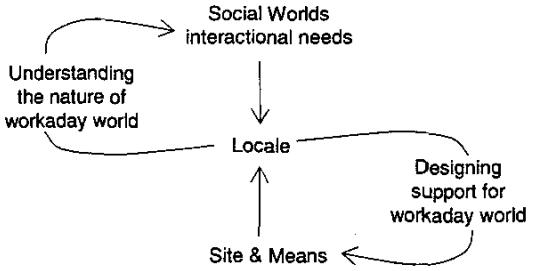

Figure 1. The Locale Framework [5]

Fitzpatrick defines the Locale Framework in terms of five important dimensions.

- *Locale Foundations*: the domains, objects, tools, media, and resources used to facilitate group activities
- *Civic Structure*: the broader context of locales, locale lifecycle processes, how locales are structured and related, and how interactions are supported among locales
- *Individual Views*: a user perspective over a locale(s)
- *Interaction Trajectory*: the past, present, and imagined future of group activities and the mutual influence of locales and group activities
- *Mutuality*: the presence and awareness (the complement of presence)

Collectively, these dimensions offer a rich basis for describing collaboration context and thus guide our efforts to develop a more effective authorization model for collaborative systems.

4 Locale-Based Access Control Model

Locale-based Access Control (Locale-BAC) is a-collaborative access control model grounded in Fitzpatrick's Locale Framework and derived from well-known role-based access control (RBAC) model. To derive Locale-BAC, fundamental changes were made to $RBAC_0$, the baseline reference model to the well known family of RBAC reference models proposed by Sandu et. al. [12] Changes were made to $RBAC_0$ to account for the access control requirements unique to collaborative systems. Despite these derivations, however, Locale-BAC supports the complete $RBAC_3$ reference model [12], which is the most expressive model in the Sandu et. al. family of RBAC reference models. Key differences between Locale-BAC and $RBAC_0$ exist in the definition of sessions and the introduction of locales. A locale is a group place [6], which has meaning only in relationship to the group of sessions that use it. A locale is used as a means of relating the sessions in context to support interactions among the users. Access control is supported by membership functions on the locale that grant or deny access to resources depending on system-defined membership levels. The definition of locale greatly simplifies the specification of access control constraints required by most collaboration environments.

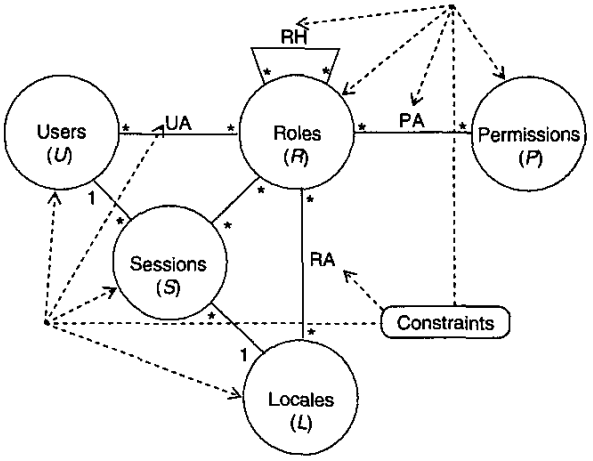

Figure 2. Locale-Based Access Control Model

As shown in Figure 2, the Locale-BAC model consists of five entities, users (U), roles (R), permissions (P), locales (L) and sessions (S).

Users, Roles and Permissions: These entities are interpreted in the same manner as in the consolidated $RBAC_3$ [12] model with role hierarchies and constraints.

Locale: A locale is a long-lived, multi-session, collaborative environment. The session mapping of one user to possible many roles according to role-to-user assignment UA is restricted by the role-to-locale assignment RA.

Sessions: A user establishes a session in context of each locale in which s/he wants to be active. While establishing a session, a user may activate a subset of the roles s/he is assigned as restricted by the roles assigned to the locale. In order to be active in a locale, a user must be able to create a non-empty session in the locale. Each session maps one user to possibly many roles, but a single locale. For a given session, multiple roles may be activated as restricted by the role-to-locale and user-to-role assignment relations. The set of permissions available to a user in a locale is defined as the union of permissions from all roles activated in that user's locale session. While it also may be useful for a user to be able to assume

multiple identities (i.e., sessions) in a single locale, we suggest constraining users such that, for a given locale, a user has at most one session (see Section 5).

4.1 Definition

The formal definition of the Locale-BAC model as shown in Figure 2 has the following components

- U, R, P, S, L (users, roles, permissions, sessions and locales, respectively)
- $PA \subseteq P \times R$, a many-to-many permission-to-role assignment relation
- $UA \subseteq U \times R$, a many-to-many user-to-role assignment relation
- $RA \subseteq R \times L$, a many-to-many role-to-locale assignment relation
- $RH \subseteq R \times R$ is a partial order on R called the role hierarchy or role dominance relation, also written as $\geq$
- *user*: $S \rightarrow U$, a function mapping each session s_i to the single user, *user*(s_i) (constant for the session's life time)
- *locale*: $S \rightarrow L$, a function mapping each session s_i to the single locale, *locale*(s_i) (constant for the session's life time)
- *sessions*: $L \rightarrow 2^S$, a function mapping each locale l_i to a set of sessions (which may change with time), $sessions(l_i) = \{s \mid locale(s) = l_i\}$
- *roles*: $S \rightarrow 2^R$, a function mapping each session s_i to a set of roles (which may change with time), $roles(s_i) \subseteq \{ r \mid (\exists r' \geq r) [((user(s_i), r') \in UA) \wedge ((r, locale(s_i)) \in RA)]\}$
- *roles*: $L \rightarrow 2^R$, a function mapping each locale l_i to a set of roles (which may change with time), $roles(l_i) = \cup_{s \in sessions(l_i)} \{ r \mid r \in roles(s) \}$
- *permissions*: $S \rightarrow 2^P$, a function mapping each session s_i to a set of permissions (which may change with time), $permissions(s_i) = \cup_{r \in roles(s_i)} \{ p \mid (\exists r' \leq r) [(p, r') \in PA]\}$
- *permissions*: $L \rightarrow 2^P$, a function mapping each locale l_i to a set of permissions (which may change with time), $permissions(l_i) = \cup_{s \in sessions(l_i)} \{p \mid p \in permissions(s)\}$
- *capability*: $L \rightarrow 2^P$, a function mapping each locale l_i to a set of permissions, $capability(l_i) = \cup_{(\forall r) [(r, l_i) \in RA]} \{p \mid (\exists r' \leq r) [(p, r') \in PA]\}$
- *constraint-safe*: $L \rightarrow$ {true, false}, a function mapping each locale l_i to a Boolean. Informally, *constraint-safe*(l_i) is true when none of the locale's constraints are violated. (Note: a formal definition of the Locale-BAC constraint model is identified as future work.)

Permission sets are used to define membership functions for locales to facilitate the definition of unique, collaboration-oriented constraints, such as the *principle of greatest authority* and the *principle of all privileged* that are presented in this paper.

4.2 General Access

Based on the definition of Locale-BAC, access determinations are defined as a function that maps a session and permission to a Boolean as follows:

- *access*: $(S \times P) \rightarrow$ {true, false}, a function mapping a session s_i and permission p_i to a Boolean, $access(s_i, p_i) \Rightarrow (p_i \in permissions(s_i) \wedge constraint\text{-}safe(locale(s_i)))$

Narrative: A user may invoke a permission in a locale if the permission is a member of the permission set for the user's session in the locale and there are no constraint violations.

4.3 Locale Participation

Before an invocation request can be satisfied, a user must participate in a locale. The role assignment (RA) relation defines those locales within which a user may participate. This can be defined as:

- For a given user u_i and locale l_i, $participate(u_i, l_i) \Rightarrow (\exists r' \leq r)[((u_i, r) \in UA) \wedge ((r', l_i) \in RA) \wedge constraint\text{-}safe(l_i)]$

Narrative: If there exists roles r and r′, where $r' \leq r$, such that the user is assigned to the role r and the role r′ is assigned to the locale and no other constraint is violated, then a user may participate in the locale.

If the first two clauses in the consequence are true, yet there exists a constraint violation, then locale policies (see Section 7) dictate what protocols may be used to allow the user to enter the locale, if at all.

Locale participation is then defined as:

- *participate*(u_i, l_i) $\Leftrightarrow$ ($\exists$s) [(u_i = *user*(s)) $\wedge$ (l_i = *locale*(s))]

Narrative: A user is participating in a locale if and only if that there exists a session for the given user and locale.

5 Authorization Constraints

An important implication of applying the Locale Framework to collaborative access control is that authorization decisions always occur in context. Our observations show that several authorization decisions depend heavily on context. For example, consider a scenario where in order for a user to invoke a permission in a collaborative setting, it is required that all individuals, active in that context be equally privileged. – e.g., a Faculty Member may be able to read/view student grade sheets only when all individuals in the current context have the same privilege. In other collaborative contexts, the invocation of certain permissions may depend on the presence or absence or specific individuals, resources, etc. – e.g., an Assistant Project Manager may be authorized to enact design changes in absence of the Project Manager.

Authorization decisions similar to those presented above are often required for collaborative access control. We believe the natural way to express these requirements is as constraints within Locale-BAC. In the following, we highlight three such constraints: i) *single user session per locale*; ii) the *principle of all privileged*; and iii) the *principle of greatest authority*.

5.1 Single User Session per Locale

For some collaboration environments it might be useful to enforce a constraint restricting users to assume only a single identity (i.e., session) per context. This feature can be easily expressed by allowing only a single user session per locale.

- ($\forall$s $\in$ *sessions*(l_i)) ($\neg\exists$ s′ $\in$ *sessions*(l_i) [(s $\neq$ s′) $\wedge$ (*user*(s) = user(s′))])

Narrative: For any given locale, a user will have at most one session.

5.2 Principle of All Privileged

Defining membership levels for permissions in the capability list for a locale allows natural expression of constraints required by most collaboration environments. One such constraint is the *principle of all privileged* which requires the existence of a set of permissions, *all_privileged*(l_i) $\subseteq$ *capability*(l_i), for each locale l_i such that:

- ($\forall$p $\in$ *all_privileged*(l_i), s $\in$ *sessions*(l_i)) [*access* (s, p) $\Rightarrow$ ($\forall$s′ $\in$ *sessions*(l_i) [p $\in$ *permissions*(s′)])]

Narrative: For a session to invoke a permission in a locale, where the permission is a member of the "all privileged" set for the locale, all sessions active in the locale at the time of invocation must have the permission in their permission set.

5.3 Principle of Greatest Authority

In many collaborative settings the ability to invoke a particular permission may be deferred to the user(s) with the greatest authority as specified by the role hierarchy (RH) relation. This newly identified constraint requires the existence of a set of permissions, *greatest_authority*(l_i) $\subseteq$ *capability*(l_i), for each locale l_i such that:

- ($\forall$p $\in$ *greatest_authority*(l_i), s $\in$ *sessions*(l_i)) [*access*(s, p) $\Rightarrow$ ($\exists$r $\in$ *roles*(s) [($\exists$r′ $\leq$ r)[(p, r′) $\in$ PA]) $\wedge$ (($\neg\exists$r″ $\in$ *roles*(l_i))[r < r″])])]

Narrative: For a user to invoke a permission in a locale, where the permission is a member of the "greatest authority" set for the locale, the user must be active in a role in the locale that both is related directly or indirectly (via the role dominance relation) to the permission and for which there is no other user in an active role of greater authority (i.e., strict dominance) in the locale.

The usefulness of these constraints is demonstrated using example scenarios in the next section.

6 Example Scenarios

In this section, we present five example scenarios to demonstrate the effectiveness of the Locale-BAC Model. To aid in this demonstration, we have constructed an example Locale-BAC policy for a hypothetical academic institution. For each scenario, assume the role hierarchy as shown in Figure 3.

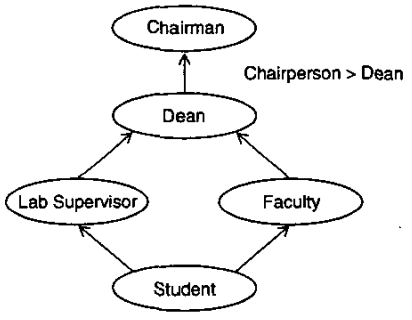

Figure 3. Example Role Hierarchy

The user to role assignment (UA) and locale to role assignment (RA) for these scenarios are expressed in Table 1 and Table 2.

Table 1. User to Role Assignment (UA)

		Roles				
		Chairperson	Dean	Faculty	Student	Lab Supervisor
Users	A	✓				
	B		✓			
	C			✓		
	D			✓		
	E				✓	
	F				✓	
	G					✓

Table 2. Locale to Role Assignment (RA)

		Roles				
		Chairperson	Dean	Faculty	Student	Lab Supervisor
Locales	Registrar's Office	✓	✓	✓		
	Classroom			✓	✓	
	Laboratory				✓	✓

Table 3 and Table 4 depict the permission assignment relation (PA) for the *Registrar's Office* locale and the *Classroom* locale, respectively.

Table 3. Permission to Role Assignment (PA) for *Registrar's Office* Locale

Objects in context of locale *Registrar's Office*	Operations	Roles	Set membership for locale *Registrar's Office*
Student_Graduation _Approval.doc	Write	Chairperson	Greatest_Authority
	Read, Lookup	Chairperson, Dean, Faculty	All_privileged
Student_Dissertation _Evaluation.doc	Write	Chairperson, Dean, Faculty	Greatest_Authority
	Read, Lookup	Chairperson, Dean, Faculty	All_privileged

Table 4. Permission to Role Assignment (PA) for Locale 2

Objects in context of locale *Classroom*	Operations	Roles	Set membership for Locale *Classroom*
Student_ Evaluation.xls	Write	Faculty	All_privileged
	Read, Lookup	Faculty, Student	All_privileged
Student_ Thesis.doc	Read	Faculty, Student	All_privileged
	Lookup	Faculty, Student	All_privileged

Using these tables as reference, consider the following scenarios which illustrate the effectiveness of the Locale-BAC model to support collaborative access control.

Scenario 1 (Participation Denied): Assume user E tries to create a session S_E with the active role of a *Student* to enter the *Registrar's Office* Locale.

This action is prohibited according to definition of the role assignment relation (RA), which specifies the roles that can be active for a session in the context a given locale. This scenario demonstrates the impact of the role-to-locale assignment (RA) relation on the *Registrar's Office Locale* by the manner in which it narrows down the list of potential sessions that can be created in its context.

Scenario 2 (Uniform Participation): Assume users A, B, C, and D try to create sessions S_A, S_B, S_C, S_D, respectively, each with the active role of a *Faculty* to enter the *Registrar's Office* Locale.

This action is allowed as (*Faculty, Registrar's Office*) is an element of RA, and the users A, B, C, and D all assume the role of *Faculty* according to the user assignment (UA) and role hierarchy (RH) relations. The role hierarchy allows users to create a session to enter a locale with any combination of roles junior (less dominant) to the users own roles as long as the junior roles of the session have also been assigned to the locale by way of the role assignment relation (RA). This is evident in the definition of the function (*roles*: $S \rightarrow 2^R$). Also the permissions available to user A through S_A will be those associated with *Faculty* and *Student* roles by the permission assignment relation (PA) but not a *Chairperson*, which is apparent in the definition of the function (*permissions*: $S \rightarrow 2^P$).

Scenario 3 (Diverse Participation): Assume user B creates a session S_B with the role of a *Dean*, users C and D create session S_C and S_D respectively with the role of a *Faculty* to enter *Registrar's Office*.

In this scenario, B through S_B will be granted the permissions assigned to the role of a *Dean*, *Lab Supervisor*, *Faculty*, and *Student*; whereas C and D, through S_C and S_D, respectively, are only granted the permissions assigned to the role of a *Faculty* and *Student*. This scenario is used to demonstrate that the permissions a user is able to invoke in the context of a locale depend on the role the user activates in the session to enter the locale and the roles assigned to the locale.

Considering the same scenario for the *Classroom Locale*, user B is able to establish session S_B in the role of a *Dean* as the role assignment relation (RA) for the *Classroom Locale* consists of roles *Faculty* and *Student*, only. However, user B may enter the *Classroom Locale* by establishing session S_B in the role of a *Faculty* or *Student*.

Scenario 4 (Principle of All Privileged): In the context of the *Classroom Locale*, assume users *C* and *D* create session S_C and S_D, respectively, with the role of *Faculty* and users *E* and *F* create sessions S_E and S_F, respectively, with the role of a *Student*.

For this scenario, assume users *C*, *D*, *E*, and *F* wish to invoke permissions on the locale object entitled *Student_Evaluation.xls*. Since the permissions on this object are members of the *all_privileged* set, when these users are present in the locale as assumed, the only permissions that may be invoked on the object are the *Read* and *Lookup* permissions. Thus, although the permission assignment relation (PA) states that users active in the *Faculty* role may invoke the *Write* permission, the *all_privileged* constraint prohibits its invocation. Table 5 presents the access table for various combinations of active roles in the *Classroom Locale.*

Table 5. Scenario 4 Access to Student_Evaluation.xls

Roles active in the Locale *Classroom*	Role	Effective access to *Student_Evaluation.xls*
Faculty	Faculty	Write, Read, Lookup
Faculty, Student	Faculty	Read, Lookup
	Student	Read, Lookup
Student	Student	Read, Lookup

Scenario 5 (Principle of Greatest Authority): In the context of Locale *Registrar's Office*, assume user *A* creates session S_A with the role of a *Chairperson*, User *B* creates session S_B with the role of a *Dean*, User *C* and *D* create session S_C and S_D respectively with the role of a *Faculty.*

In this scenario, only user *A* can exercise a *Write* permission for *Student_Dissertation_Evaluation.doc* as this permission is a member of the *greatest_authority* set for the *Registrar's Office Locale.* This is the case even though sessions S_B, S_C and S_D have same the permission through the user and permission assignment relations. The access table for the various combinations of roles present in the *Registrar's Office Locale* is shown in Table 6.

Table 6. Scenario 5 Access to Student_Dissertation_Evaluation.doc

Roles active in Locale *Registrar's Office*	Role	Effective access to *Student_Dissertation_ Evaluation.doc*
Chairperson, Dean and Faculty	Chairperson	Write, Read, Lookup
	Dean	Read, Lookup
	Faculty	Read, Lookup
Dean and Faculty	Dean	Write, Read, Lookup
	Faculty	Read, Lookup
Faculty	Faculty	Write, Read, Lookup

7 Locale Policies

The above constraints and example scenarios highlight that permission invocation is not always a discrete event. Some operations, such as *read*, are continuous and, thus, must be continuously reassessed with changes to the invocation context, i.e., the locale. When and how changes are allowed to a locale is thus a very important issue.

Consider the situation where an *all_privileged* permission, such as *read*, is being appropriately invoked within a particular locale. What should happen when an additional user attempts to enter the locale, but for whom the session s/he can or wishes to create does not contain the *all_privileged* permission in its permission set? We envision several possible solutions to this situation.

- The system could deny the user access to the locale.
- The system could terminate the invocation of the *all_privileged* permission and then allow the user access to the locale.
- The system could prompt users present in the locale, informing them of the situation and inquiring whether they wish to deny entry or to terminate the permission invocation and allow entry.

These and many other protocols could be enacted when changes to locales are requested or simply occur. We believe that the manner in which locales respond to such situations are context dependent and therefore must be configurable. Thus, we propose the use of an extensible set of locale policies to account for these situations.

8 Conclusions and Future Work

A fundamental goal of any collaborative system is to cater effectively to the collaboration needs of users while maintaining the required security and privacy for users and resources. Realizing this goal is essential to the success of such systems.

We believe that placing authorization decisions in context is not achieved easily by building complex policies within existing authorization models. Rather, we contend that baseline changes to existing models are necessary to meet the unique requirements of collaborative systems. In this paper, we presented the Locale-Based Access Control (Locale-BAC) model as a baseline derivation to the traditional RBAC model and demonstrated the usefulness and appropriateness of this model through several newly identified authorization constraints and supporting scenarios.

While Locale-BAC is an important first step in the effort to incorporate context more fully into authorization decisions, much work remains to be done. Future work includes: designing a standard Locale-BAC constraint specification language, incorporating administrative functions, identifying additional collaboration-specific constraints, and broadening the scope of Locale-BAC to support process, time, and individual views effectively. In addition, our current efforts are focused on developing a proof-of-concept implementation of the Locale-BAC model using type definition languages such as XML Schema. Such an implementation will validate the model and assist collaborative system designers in their effort to incorporate Locale-BAC into existing collaborative infrastructures.

9 References

[1] Brothers, L., Sembugamoorthy, V., Muller, M., ICICLE: Groupware for Code Inspection. *CSCW 90: Proc. of the Conf. on Computer-Supported Cooperative Work*, Los Angeles, CA ACM, 1990, pp. 169-181.

[2] Bullock, A., Benford, S. An access control framework for multi-user collaborative environments, In *Proc. of ACM GROUP '99*, Phoenix, Arizona, USA.

[3] Covington, M., Long, W., Srinivasan, S., Dey, A.K., Ahamad, M., Abowd, G. D. Securing Context-Aware Applications Using Environment Roles, In *Proc. of ACM SACMAT'01*, May 2001, Chantilly, VA, USA.

[4] Ellis, C. A., Gibbs, S. J., Rein, G. L. Design and use of a group editor. In *Proc. of IFIP WG 2.7 Working Conference on Engineering for Human-Computer Interaction*, pages 13-28, 1989.

[5] Fitzpatrick, G. *The Locales Framework: understanding and Designing for Cooperative Work.* Ph.D. Thesis. The Univ. of Queensland, Australia, 1999.

[6] Fitzpatrick, G., Mansfield T., Kaplan S., Locales Framework: Exploring foundations for collaboration support. *IEEE Proceedings of OzCHI'96.*

[7] Georgiadis, C. K., Mavridis, I., Pangalos, G., Thomas, R.K. Flexible Team-Based Access Control Using Contexts, In *Proc. of ACM SACMAT'01*, May 2001, Chantilly, VA, USA.

[8] Grief, I., Sarin, S. Data Sharing in Group Work. *Computer-Supported Cooperative Work: A Book of Readings*, Irene Grief, ed. San Mateo, CA: Morgan Kaufman 1988, pp. 477-508.

[9] Kaplan, S.M., Tolone, W.J., Bogia D.P., Bignoli, C. Flexible Active Support for Collaborative Work with Conversation Builder. *Proceedings of the Conference on Computer-Supported Cooperative Work*, Toronto, Ontario: ACM, pp. 378-385, 1992.

[10] Lampson, B. W. Protection. In *Proc. 5th Princeton Symposium on Information Sciences and Systems*, Princeton University, March 1971, pp 437-443, reprinted in *Operating Systems Review*, 8(2), pp18-24, January 1974.

[11] Neuwirth C. M., Kaufer, D. S., Chandhok, R., Morns, J. Issues in the Design of Computer Support for Co-authoring and Commenting. *Proceedings of the Conference on Computer-Supported Cooperative Work* Los Angeles, CA ACM, pp. 183-195, 1990.

[12] Sandhu, R. S., Coyne, E. J., Feinstein, H. L., Youman, C. E. Role-Based Access Control Models. *IEEE Computer*, 29(2), pp.38-47, Feb. 1996.

[13] Shen, H., Dewan, P. Access control for collaborative environments. *Proceedings of the Conference on Computer-Supported Cooperative Work*, Toronto, Ontario: ACM, pp.51-58, 1992.

[14] Sohlenkamp, M., Chwelos, G. Integrating Communication, Cooperation, and Awareness: The DIVA Virtual Office Environment. *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, Chapel Hill, October 22-26, 1994. pp. 331-343.

[15] Strauss, A. *Continual Permutations of Action.* Aldine De Gruyter, New York, 1993.

[16] Suchman, L. *Plans and Situated Actions.* Cambridge University Press, 1987.

[17] Thomas, R.K. Team-based access control (TMAC): A primitive for applying role-based access controls in collaborative environments. *Proc. of ACM RBAC'97*, 1997.

[18] Thomas, R. K., Sandhu, R. S. Task-based authorization controls (TBAC): A family of models for active and enterprise oriented authorization management. In *IFIP WG 11.3 Workshop on Database security*, Lake Tahoe, California, August 1997.

[19] Tolone, W.J., Ahn, G.J., and Pai, T. Access Control in Collaborative Systems. Technical Report. SIS Department. UNC Charlotte, 2002.

[20] Tolone, W. J., Kaplan, S. M., and Fitzpatrick, G. Specifying Dynamic Support for Collaborative Work within WORLDS. In *Proceedings of the ACM 1995 Conference on Organizational Computing Systems*, pages 55-65, Milpitas, CA, August 13-16, 1995.