A Financial Institution's Legacy Mainframe Access Control System in Light of the Proposed NIST RBAC Standard

Andrew D. Marshall
TD Bank Financial Group
275 Dundas Street
London, Ontario, Canada
andrew.marshall@td.com

Abstract

In this paper we describe a mainframe access control system (DENT) and its associated delegated administration tool (DSAS) that were used in a financial institution for over 20 years to control access to banking transaction products. The first part of this paper describes the design and operation of DENT/DSAS as an example of a long-lived access control system in a financial institution.

A standard for Role-Based Access Control (RBAC) has recently been proposed by the United States National Institute of Standards and Technology (NIST). The second part of this paper discusses how the functionality of DENT/DSAS could be achieved by applying its principles of operation within the NIST model. In so doing we also evaluate the proposed standard by validating it against the requirements embodied in a successful access control system.

We conclude with some observations about the design of DENT/DSAS and suggestions for changes in the proposed RBAC standard to accommodate some features of DENT/DSAS that it does not appear to support.

1. Introduction

The The Toronto-Dominion Bank (TD) is one of Canada's largest financial institutions. On February 1, 2000, it acquired the assets of CT Financial Services (CT), which operated a trust company and related financial services businesses under the name Canada Trust. With the purchase TD acquired a portfolio of software that constituted the banking systems of the former Canada Trust. As a consequence of system integration decisions driven by business requirements, much of the CT software portfolio has been or shortly will be retired, including the access control system described in this paper.

CT's mainframe banking software included a number of "product systems", each of which implemented a particular product or family of products. For example, the Savings system implemented traditional savings accounts, and the Mortgage system implemented mortgages. All the banking product systems ran in a common transactional environment using IBM's Customer Information Control System (CICS). Around 1978, an access control system called "Data Entry" was written as a front end for CICS-based transactional programs[1]. Developers of product systems were provided with code templates which enabled them to wrap each "business function" (i.e., one or more transaction calls and associated business rules) with a call to DENT's access control routine. DENT thus provided a common access control mechanism across all product systems. Each business function protected by DENT had an associated "keyword". DENT made its access control decisions by determining whether the user requesting access to a business function possessed the corresponding keyword in his or her "security profile". A user's security profile, then, was a list of products which the user was allowed to run, and a list of keywords for each product representing the product-specific business functions to which the user had been granted access. DENT's security database contained the security profiles for all users, as well as some related tables (e.g., a list of all products and all valid keywords).

One of the products controlled by DENT was "Data Security Administration System" (DSAS), which was the administrative tool for DENT's security database. DSAS was used most often by retail bank branch managers to grant and deny access to business functions for their employees without intervention by a central security department. DSAS updated the DENT security database; DENT read its security database to enforce the access rights set up by the users of DSAS. The relationships among the DENT/DSAS compo-

[1] DENT also provided a number of other common functions, including data presentation, journalling, logging, etc., but we are only concerned here with its access control component.

Proceedings of the 18th Annual Computer Security Applications Conference (ACSAC'02)
1063-9527/02 $17.00 © 2002 IEEE

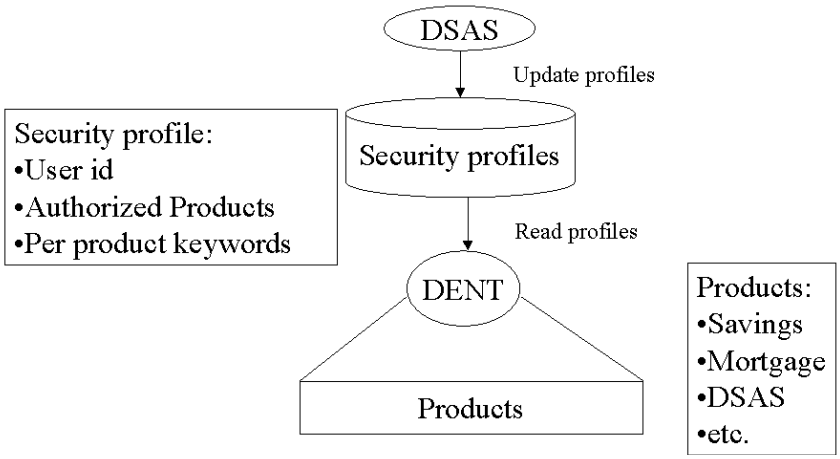


Figure 1. DENT/DSAS component relationships

nents are shown in Figure 1. The programs and databases in DENT/DSAS were protected from unauthorized access outside of that mediated by DENT/DSAS by the native mainframe security system, which was Computer Associates' ACF2 and then IBM's RACF. In general, the native mainframe security system was used to guard access to the files and programs of the product systems; DENT was used to grant finer-grained access to specific banking functions implemented in these systems.

A proposed standard for Role-Based Access Control (RBAC) has been put forward by the United States National Institute of Standards and Technology (NIST). The proposed standard has generated interest within TD and other large corporations (see, for example, the theme of the October 2001 meeting of the Network Applications Consortium, `www.netapps.org`) facing the headaches associated with managing the access rights of thousands of users to thousands of business functions. The role of a "role" as an intermediary between users and permissions offers simplified processes for user rights administration. By comparing the proposed RBAC standard against the functionality embedded in a successful access control system, we believe we can offer suggestions that will make the RBAC standard even more appealing and useful within financial institutions.

The rest of this paper is organized as follows. Section 2 presents an overview of DENT, the legacy access control system, and DSAS, its delegated administration tool. Section 3 outlines the software developer's view of DENT. We give a brief overview of the NIST's proposed RBAC standard in Section 4. Section 5 demonstrates the power of the NIST RBAC standard by describing a scheme through which it could be applied to offer much of the functionality of DENT/DSAS. Finally, in Section 6, we conclude with some observations on the design of DENT/DSAS and with some suggestions for changes to the proposed NIST RBAC standard.

2. Overview of DENT and DSAS

The environment protected by DENT was a collection of products, each one responsible for one part of the overall banking functions offered by CT. Each product had a "product code", e.g., `SVG` for the Savings product, `MTG` for the Mortgage product, etc. Each product was associated with a business unit within CT that was responsible for the strategy, marketing, and business rules for the product. Each such business unit would designate one or more individuals within the unit as "product owners" for the product. The software for each product was designed and maintained by developers in the Business Systems department.

Associated with each product was a set of keywords (character strings, chosen where possible to have mnemonic value). Each keyword was used to protect a "business function" within a product. By "business function", we are referring to a specific CICS transaction and the associated business rules and context around the transaction. The designation of business functions that were of a sensitive enough nature to require access control was the job of the product's developer, in concert with the product owner(s). For example, most products offered an Inquiry function, access to which was typically controlled by the presence (or absence) of an `INQ` keyword in a user's security profile. Note that keywords were product specific: two products could have the same keyword, with each controlling access to vastly different business functions in the products.

Keywords were divided into two classes: "normal" and "restricted". When a keyword in the normal class was requested for a user's profile it was automatically added to

the profile. A keyword in the restricted class required approval from the product's owner before it was added to a user's profile. The keyword mechanism was used to denote product owners: a product owner for product FOO would have the keyword KYAPFOO in his or her security profile for product FOO.

As an example of the distinction between normal and restricted keywords, consider a simplified version of the "Deposit" business function of the Savings product. Suppose that for legal reasons the Savings business unit distinguished between two kinds of deposits: those under $10,000, and those of $10,000 or more. The logic of the deposit business function would be split into two cases, each depending on a different keyword. The normal keyword, DEP, would apply to deposits of less than $10,000, and the restricted keyword, HIDEP, would permit the holder to process deposits of $10,000 or more. The Savings product owner would freely grant the DEP keyword to any user, but would not grant users the HIDEP keyword until it had verified that the user had met the requirements of the Savings business unit.

One of the products controlled by DENT was DSAS (Data Security Administration System), the delegated administration tool used by managers to assign product keywords to the company employees under their direct control. The philosophy employed by the designers of DENT/DSAS was one of centralized oversight with decentralized administration. Keywords were created by a central authority (the DSAS Administrator; see Section 3) but were assigned to users by local decision makers. The rationale behind this philosophy was that local managers were in the best position to determine the access needs of their employees, but the keyword creation and deletion process required central coordination.

To be able to run DSAS, a user had to have the DSAS product in his or her security profile and the profile had to include the appropriate DSAS keywords. The DSAS keyword, BRAUTH, permitted its holder to create security profiles within the holder's branch (a branch number was an attribute of each user's identity). Someone holding this keyword could automatically add normal keywords to a user's profile and generate a request (to a product owner) to add a restricted keyword to a user's profile.

3. The Developer's View of DENT

We now describe the steps involved in developing products that ran under the DENT system.

1. The product owner, in cooperation with the developer assigned to the product, developed a plan for adding a new business function to the product (or changing an existing business function).

```
MOVE LOW-VALUES         TO DE059-INTERFACE.
MOVE DTE-TRANS-COMP-A   TO DE059-PRODUCT.
MOVE 'KEYUPDT '         TO DE059-KEYWORD.
MOVE SPACE              TO DE059-INTERBR-IND.
MOVE SPACE              TO DE059-BYPASS-ABEND-IND.
MOVE 'OPER'             TO DE059-REQUEST-CODE.

EXEC CICS LINK
          PROGRAM('DE059O')
          COMMAREA(DE059-COMMON-AREA)
          LENGTH(DE059-COMM-LENGTH)
          RESP(W-CICS-RESPONSE)
END-EXEC

EVALUATE W-CICS-RESPONSE
WHEN DFHRESP(NORMAL)                    <---  executed ok
     IF DE059-KEYWORD-MATCHED
        PERFORM ...
     ELSE
        MOVE DE059-RETURN-MESSAGE ....
     END-IF
WHEN OTHER                              <---  other error
     .....
END-EVALUATE
```

Figure 2. Keyword check library call template.

2. The developer determined whether it would be appropriate to protect the new business function with an existing product keyword. If so, step 3 was skipped.

3. The developer sent a request to the DSAS Administrator—the person within the company's Data Security team responsible for maintaining the DENT/DSAS security database—for a new product keyword. The request included the name of the product, the name of the new keyword, a short description, an indication whether the keyword was normal or restricted, and guidelines for its assignment to users. Normally, the DSAS Administrator granted the request and added the keyword to the list of keywords for the product maintained in the DENT security database.

4. The developer wrote (or modified) the COBOL code needed to implement the desired new or changed business function. The code included a call to a library routine to check whether the user executing the code had the designated keyword in his or her profile. The general form of the library routine calling sequence is given in Figure 2.

The business rules around some business functions required a slightly more complicated approach. For various reasons (e.g., the sensitivity of the transaction, the user's transaction limits or job level) some business functions required a Dual Control security policy to be applied to them. As an example, consider the "Correction" business function in the Savings product. This function allowed transactions made in error to be reversed. Because of its power, the Correction function required the participation of two people: an

initiator and an approver. The code implementing the Correction function was structured as follows:

1. Is the keyword for the Correction function of the Savings product (e.g., SVG, COR) in the initiator's security profile? [2]

2. A routine was then called to pop up a request for a supervisor override on the initiator's display. At this point, the initiator would call an approver (supervisor) to the initiator's terminal, who would authenticate to the system by supplying his or her logon id and password.

3. The approver's profile was then checked for the presence of a Correction override keyword for the Savings product (say, COROVR). If the approver held the keyword, then the transaction continued, otherwise, it was aborted.

4. Overview of the NIST Proposed RBAC Standard

Space limitations do not permit a full exposition of the RBAC standard proposed by NIST. The diagram in Figure 3 shows the entities and relationships and each of them is briefly described in the chart below. For more details, the reader is directed to the standard [2].

Entity Description

USERS Set denoting the users whose access privileges are to be managed.

ROLES Set of roles defined for the organization.

PRMS Set of Permissions. Permissions are defined as abstract operations on objects.

OBS Set of objects used to construct PRMS.

OPS Set of abstract operations used to construct PRMS.

UA (User Assignment) A many-to-many mapping between Users and Roles: one user may be assigned many roles, and one role may be assigned to many users.

PA (Permission Assignment) A many-to-many mapping between ROLES and PRMS. A given role may have (is likely to have) more than one permission, and one permission may be assigned to multiple roles.

[2] Note that, strictly speaking, the initiator could not have made it this far without possessing the COR keyword, as the screen display routines generated by DENT—another of its functions—used the user's security profile to generate the list of transaction codes available to the user. The display routine only displayed transaction codes for which the user had permission (i.e., possessed the corresponding keyword).

SESSIONS Set denoting all the active sessions. A session is a mapping of one user to one or more roles. Each session is associated with only one user; a user may have more than one active session.

user_sessions Mapping of a user, u, onto a set of sessions.

session_roles Mapping of a session, s, onto a set of roles.

The entities described above are those of "Core RBAC". The standard also offers "Hierarchical RBAC", which introduces role hierarchies, and "Constrained RBAC", which adds constructs to support Static and Dynamic Separation of Duty relations.

5. Modelling DENT/DSAS in the RBAC Model

In this section we demonstrate how most features of DENT/DSAS's operation could be modelled in the proposed NIST RBAC standard. The presentation is structured from the point of view of an organization migrating from a a legacy access control system to one based on the proposed NIST RBAC standard. We define the new entities, those in the RBAC standard, in terms of those in the legacy system. We also identify where it is not possible within the RBAC-based system to completely implement concepts or functionality from the legacy system. As an example throughout this section, we will be referring to the sample DENT security database in Table 1. We begin by formalizing what we have so far referred to as the DENT security database.

- U, a set of users.
- P, a set of product codes.
- We represent a keyword k, associated with product p, as a tuple (p, k). $K = K_N \cup K_R$, is the set of keywords, where K_N is the set of all normal keywords, K_R is the set of all restricted keywords, and $K_N \cap K_R = \emptyset$.
- S represents the users' security profiles. $S = \{(u, N_u, A_u)\}\ \forall u \in U$, where $N_u \subseteq P$ is the set of product codes for which user u is authorized, and $A_u \subseteq K$, is the set of keywords which user u has been assigned.

For the example in Table 1, we have the following sets (only the first entry of set S is shown):

$$\begin{aligned} U &= \{\text{Bob}, \text{Carol}, \text{Ted}, \text{Alice}\} \\ P &= \{\text{SVG}, \text{DSAS}\} \\ K_N &= \{(\text{SVG}, \text{INQ}), (\text{SVG}, \text{DEP}), \\ &\quad (\text{DSAS}, \text{INQ})\} \end{aligned}$$

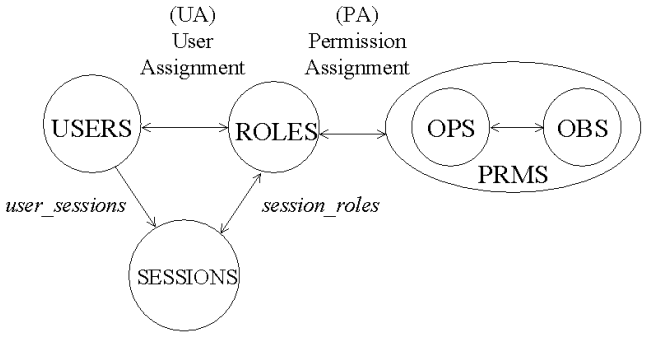


Figure 3. Core RBAC (adapted from Ferraiolo *et al.* [2]).

$$
\begin{aligned}
K_R &= \{(\text{SVG}, \text{COR}), (\text{SVG}, \text{COROVR}), \\
&\quad (\text{SVG}, \text{KYAPSVG}), (\text{DSAS}, \text{BRAUTH})\} \\
S &= \{(\text{Alice}, \{\text{SVG}, \text{DSAS}\}, \\
&\quad \{(\text{SVG}, \text{INQ}), (\text{SVG}, \text{DEP}), (\text{SVG}, \text{COR}), \\
&\quad (\text{DSAS}, \text{INQ})\}), \ldots\}
\end{aligned}
$$

We now look at mapping from the DENT/DSAS "world" into the RBAC "world". Our approach is to develop the mapping as a design exercise, structured as a series of questions and answers.

5.1. Who are the users of our system?

The users of DENT/DSAS will be the users of the new RBAC system, so $USERS = U$. Note that the proposed NIST RBAC standard allows only people to be represented in its $USERS$ set, which conflicts with the DENT/DSAS definition of a user. In the DENT/DSAS world, a user is most often a person—an employee—but may also be a machine: an application server or Automated Banking Machine (a "tin teller") operating like a human teller on behalf of a customer.

5.2. What are the roles to which users can be assigned?

The set of roles we define is the union of three kinds of roles: user roles, product owner roles (one for each product), and the `Brauth` role, i.e., $ROLES = ROLES_U \cup ROLES_O \cup \{\texttt{Brauth}\}$.

As we have mentioned, DENT/DSAS was not a role-based system. We can still view it as role-oriented if we consider that every user in the organization has, by definition, his or her own role, which has the same permissions in the RBAC world as the keywords permitted in the DENT/DSAS world. In other words, $ROLES_U$ = U. We assume that every product has a product owner, so $ROLES_O = P$.

5.3. What are the objects we are protecting?

The basic objects we are protecting are business functions as represented by keywords. So the keywords in banking product systems become the "user" objects (as with the keywords themselves, we separate "normal" objects and "restricted" objects). We also need objects to represent roles in the RBAC system so that we can assign, for example, permissions to managers to permit them to change the permissions assigned to their employees' roles, and to product owners so that they can fulfill requests for restricted keywords. We refer to these objects as "admin" objects. More formally: $OBS = OBS_U \cup OBS_A$ where

$$
\begin{aligned}
OBS_U &= OBS_N \cup OBS_R \\
OBS_N &= \{k = (p, w) \in K_N\} \\
OBS_R &= \{k = (p, w) \in K_R\} \\
OBS_A &= ROLES_U
\end{aligned}
$$

5.4. What are the operations on the objects?

We divide the operations into two kinds: user (those used to construct end user permissions) and administrative (those used to manage roles and permissions). $OPS = OPS_U \cup OPS_A$, where

$$
\begin{aligned}
OPS_U &= \{\text{exec}\} \\
OPS_A &= \{\text{assign}, \text{request}, \text{approve}, \text{change}\}
\end{aligned}
$$

5.5. How are the objects and operations combined into permissions?

$PRMS = PRMS_U \cup PRMS_A$ where

$$
PRMS_U = \{(\text{exec}, o) \forall o \in OBS_U\}
$$

User Security Profiles				
User	**Branch**	**Product**	**Keyword**	**(Implicit Role)**
Alice	1	SVG	INQ DEP COR	Teller
		DSAS	INQ	
Bob	1	SVG	INQ DEP COR COROVR	Supervising Teller
		DSAS	INQ	
Carol	1	SVG	INQ DEP COR COROVR	Manager
		DSAS	INQ BRAUTH	
Ted	2	SVG	KYAPSVG	Product Owner
		DSAS	INQ	

Product and Keyword Table			
Product	**Keyword**	**Normal/Restricted**	**Guidelines**
SVG	INQ	N	Any employee
	DEP	N	Any Teller
	COR	R	Any Teller
	COROVR	R	Supervising Teller
	KYAPSVG	R	Product Owner
DSAS	INQ	N	Any employee
	BRAUTH	R	Branch Manager

Table 1. A sample DENT/DSAS security database

$$PRMS_A = PRMS_R \cup PRMS_M \cup PRMS_O$$

and where

$$\begin{aligned} PRMS_R &= \{(\text{change}, k)\ \forall k \in OBS_A\} \\ PRMS_M &= \{(\text{assign}, k)\ \forall k \in OBS_N\} \cup \\ &\quad \{(\text{request}, k)\ \forall k \in OBS_R\} \\ PRMS_O &= \{(\text{approve}, k)\ \forall k \in OBS_R\} \end{aligned}$$

Each permission in $PRMS_U$ conveys the right to execute the business function represented by the permission's object. These are the permissions that constitute the majority of a normal user's role's permissions. Each permission in $PRMS_R$ conveys the right to change the role whose name is the object of the permission. These permissions are assigned to the `Brauth` role and to all product owner roles. Permissions in $PRMS_M$ are attached to the `Brauth` role: an "assign" operation on object k conveys the right to add the (exec, k) permission to a role; a "request" operation on object k conveys the right to request of a product owner that the (exec, k) permission be added to a role. Finally, the permissions in $PRMS_O$ are assigned to product owner roles: an "approve" operation on object k conveys the right to add the (exec, k) permission to a role.

5.6. What is the Permission Assignment (PA) relation?

This relation, the mapping between roles and permissions, is given in Table 2. We use the notation, $\texttt{foo}_{\mathcal{P}}$, to refer to the permissions assigned to role `foo`.

Note that the permissions assigned to the `Brauth` role do not capture the constraint of the corresponding keyword semantics in the DENT/DSAS system. In DENT/DSAS, someone holding the `Brauth` keyword could only change or create profiles for employees with the same branch number. The NIST RBAC model offers no obvious way to enforce this constraint. Note also that a product owner can add the keyword(s) it owns to any user's role without a preceding "request" from a user in the `Brauth` role. Within the proposed NIST RBAC standard there appears to be no way to prevent this situation. We will return to these issues in Section 6.2.

Role	Permissions
$\forall u \in ROLES_U$	$u_{\mathcal{P}} = \{(\text{exec}, (p, w)) \mid (u, N_u, A_u) \in S \wedge p \in N_u \wedge (p, w) \in A_u \wedge p \neq DSAS\}$
`Brauth`	$\texttt{Brauth}_{\mathcal{P}} = PRMS_R \cup PRMS_M$
$\forall r \in ROLES_O$	$r_{\mathcal{P}} = PRMS_R \cup \{(\text{approve}, (p, w)) \mid p = r\}$

Table 2. The Permission Assignment (PA) relation

Role	Users assigned to the role
$\forall r \in ROLES_U$	$r_{\mathcal{U}} = r$
`Brauth`	$\texttt{Brauth}_{\mathcal{U}} = \{u \in USERS \mid (u, N_u, A_u) \in S \wedge (DSAS, BRAUTH) \in A_u\}$
$\forall o \in ROLES_O$	$o_{\mathcal{U}} = \{u \in USERS \mid (u, N_u, A_u) \in S \wedge (o, KYAPo) \in A_u\}$

Table 3. The User Assignment (UA) Relation

5.7. What is the User Assignment (UA) relation?

This relation, the mapping between roles and users, is given in Table 5.4. We use the notation, $\texttt{foo}_{\mathcal{U}}$, to refer to the users assigned to role `foo`.

We complete our mapping by referring the reader to Figure 4 which illustrates the UA and PA relations for each of the users in our sample database.

6. Conclusions

The major contribution of this paper is to demonstrate the power of the proposed NIST RBAC standard by using it to model a legacy mainframe access control system. DENT and DSAS, although successful, were not perfect. Section 6.1 describes some improvements that would be included if a redesign were to be undertaken now. Similarly, NIST's proposed RBAC standard is a creature of compromise and constraint; for example, the authors required that each feature be supported by a a proven implementation. Our experience with DENT/DSAS suggest, however, that the proposed model is missing some features that are important in the financial industry, as evidenced by their support by DENT/DSAS. Section 6.2 describes these features.

6.1. DENT/DSAS: A Retrospective Redesign

DENT/DSAS was in service for 22 years before being decommissioned, and the decommissioning was triggered by system integration activities, not by some inherent flaw in the system. However, taking advantage of hindsight, a number of issues have been identified with it that were not completely solved in the initial design.

Employment practices within CT, common also to TD, raise issues with role activation. It is not uncommon for one person to work in two different branches at different times and to take on different roles in each branch. Ideally, their profile would be automatically adjusted according to their location or time of day or some other external clue. DENT/DSAS essentially side-stepped this issue by mandating that a user in this situation would have two different logon ids: one for each branch, each with its own security profile. Similar issues arise in health care, as location—the notion of a "floor"—is important in determining access to a patient's records [1].

A related issue is dealing with the privileges assigned to internal auditors. This class of employees is not assigned to any one branch but has access to the records of any branch. Their access rights are somewhat contradictory: they have complete access, much greater than that afforded to normal employees, to transaction log files, but simultaneously have no access to the transactional applications themselves. The designers of DENT/DSAS were unable to integrate access control for internal auditors into DENT/DSAS in a way that they found satisfying.

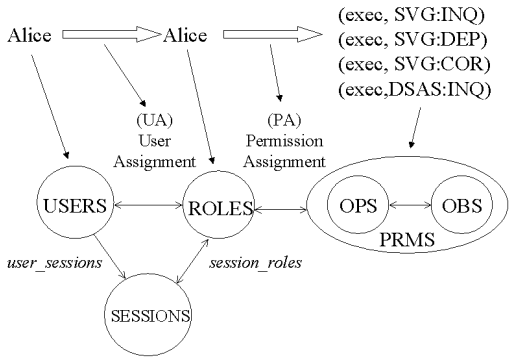
6.2. Suggestions for the Proposed RBAC Standard

As noted in Section 5.1, the proposed RBAC standard requires its $USERS$ set to denote only human users, a restriction which appears to be unnecessarily restrictive.

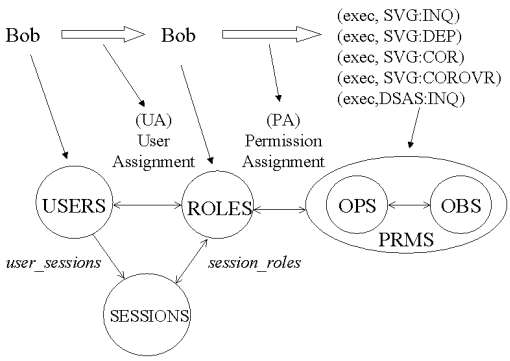
As described in Section 5.6, there is no obvious simple solution to permit us to constrain a holder of the `Brauth` role only to changing the roles of the employees in the same branch. However, there do appear to be two options:

1. Create one `Brauth` role for each branch, one change permission for each branch, and assign each manager to a branch-specific `Brauth` role rather than the generic `Brauth` role. This creates administrative complications that seem at odds with the goal of RBAC to simplify access control administration.

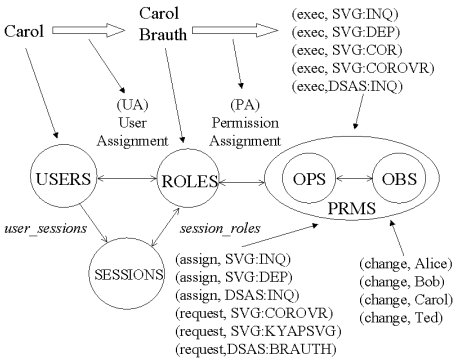
2. A more aesthetically pleasing solution would be to en-



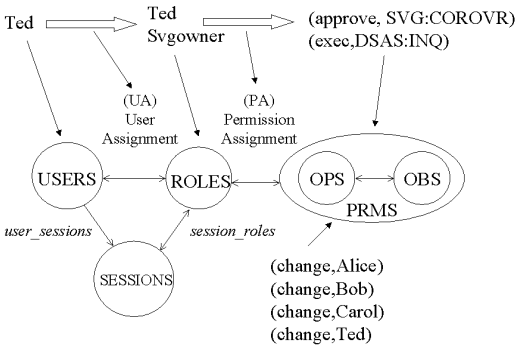
(a) User and permission assignments for Alice (Teller)



(b) User and permission assignments for Bob (Supervising Teller)



(c) User and permission assignments for Carol (Manager)



(d) User and permission assignments for Ted (Product Owner)

Figure 4. Mapping of the sample security database from Table 1 into the RBAC world

courage NIST to modify its RBAC model to support general constraints. Constraints are well-understood and useful (e.g., see the $RBAC_2$ model in Sandhu's discussion of RBAC96 in [5]): in this context a constraint would be a rule attached to a role which indicates that it only applies if the holder of the role and the target object have the same value in their "branch" attributes.

The Dual Control policy presented in Section 3 requires two users, an initiator and an approver, to cooperate to complete a transaction. The proposed NIST RBAC standard does not appear to support this policy since it does not allow more than one user to be connected to a single session. The most obvious way to adjust the standard to support Dual Control would be to relax this restriction, and allow more than one user to be attached to a session.

Alternatively, is could be supported by adding a workflow or temporal ordering constraint on permissions. This would also help with the issue of product owners being able to make arbitrary changes to any user's profile, discussed in Section 5.6.

The bulk of the permissions and roles we created in mapping DENT/DSAS to RBAC were concerned with administration of the RBAC system itself. The absence of administrative roles and permissions from the proposed NIST RBAC standard is a surprising omission. Administrative role hierarchies are a familiar notion in the RBAC literature (see, again, Sandhu's RBAC96 article [5]). In their paper demonstrating how RBAC can simulate other access control models, Osborn, Sandhu and Munawer note that, "The Administrative Role Hierarchy is essential in the enforcement of DAC policies" [4]. Finally, in their critique [3] of an earlier version of the standard, Jaeger and Tidswell remark on the "surprising omission" of administrative roles in the standard.

7. Acknowledgements

Chuck Burns was the principal architect of DENT/DSAS. He provided many details of its history and development. Discussions with Val Woodward, Ron Brubacher, and Barb Edwards provided additional useful information.

The author would like to thank Sylvia Osborn for reviewing early drafts of this paper and for providing expert guidance to the RBAC literature; the anonymous reviewers for their valuable comments; and, finally, Cristina Scassa and Frank Coletti for supporting RBAC research within the TD Bank Financial Group.

References

[1] K. Beznosov. Requirements for access control: US healthcare domain. In *Proceedings of the 3rd ACM Workshop on Role-Based Access Control (RBAC-98)*, pages 43–46, New York, Oct. 22–23 1998. ACM Press.

[2] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli. Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security*, 4(3):224–274, Aug. 2001.

[3] T. Jaeger and J. E. Tidswell. Rebuttal to the NIST RBAC model proposal. In *Proceedings of the 5th ACM Workshop on Role-Based Access Control (RBAC-00)*, pages 65–66, N.Y., July 26–27 2000. ACM Press.

[4] S. Osborn, R. Sandhu, and Q. Munawer. Configuring role-based access control to enforce mandatory and discretionary access control policies. *ACM Transactions on Information and System Security*, 3(2):85–106, May 2000.

[5] R. Sandhu. Role-based access control. In M. V. Zelkowitz, editor, *The Engineering of Large Systems*, volume 46 of *Advances in Computers*, pages 238–286. Academic Press, 1998.