View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Repositório Institucional da Universidade de Aveiro

Extending RBAC Model

to Control Sequences of CRUD Expressions

Óscar Mortágua Pereira
Instituto de Telecomunicações
DETI, University of Aveiro
Aveiro, Portugal
omp@ua.pt

Diogo Domingues Regateiro, Rui L. Aguiar
Instituto de Telecomunicações
DETI, University of Aveiro
Aveiro, Portugal
{diogoregateiro,ruilaa}@ua.pt

Abstract—**In database applications, access control is aimed at supervising users' requests to access sensitive data. Users' requests are mainly formalized by Create, Read, Update and Delete (CRUD) expressions. The supervision process can be formalized at a high level, such as based on the RBAC model, but in the end the relevant aspect is the data being accessed through each CRUD expression. In critical database applications access control can be enforced not on a CRUD by CRUD basis but enforced at the level of sequences of CRUD expressions (workflow). This situation can occur whenever established security policies are based on strict procedures that define step by step the actions (sequences of CRUD expressions) to be followed. Current RBAC models do not support this type of security policies. To overcome this security gap, we leverage previous researches to propose an extension to the RBAC model to control for each role which sequences of CRUD expressions are authorized. We demonstrate empirical evidence of the effectiveness of our proposal from a use case based on Java and JDBC. Our use case is based on typed security layers built from a software architectural model and also from metadata based on the proposed RBAC model extension.**

Keywords-information security, access control, RBAC, software architecture, software engineering, components.

I. INTRODUCTION

Access control [1][2] *"is concerned with limiting the activity of legitimate* users." [3]. Four of the main strategies for regulating access control policies are: discretionary access control (DAC) [3], mandatory access control (MAC), attribute-based access control [4][5] (ABAC) and Role-based access control (RBAC) [6][7]. There are other strategies for regulating access control, such as credential-based access control (CBAC) [Li, '05; Yu, '03], content driven [Moffett, '91; Staddon, '08], location driven [Decker, '08], public key driven [Wang, '11] and certificate driven [Samarati, '01b]. Each one addresses specific security needs for the system under protection. In this paper we are focused on RBAC, which has emerged as one of the dominant access control policies [8], namely for relational database applications. RBAC policies comprise several concepts, among them: users, roles (they can be hierarchized), permissions, delegations and actions. Basically, legitimate (authenticated) users can only execute some action if he has been authorized to play the role that rules that action. At the end, actions are the four main operations on database objects (tables and views) defined by the data manipulation language of the SQL standard: Insert, Select, Update and Delete, herein referred to as Create, Read, Update and Delete (CRUD) expressions, respectively. Depending on the granularity and the used technique, the authorization to execute these actions can be defined at the level of database objects, at the level of columns, at the level of rows and at the level of cells. Another relevant aspect that has not been addressed by current RBAC models is the sequence in which CRUD expressions are executed. Changing the order in which CRUD expressions are executed can lead to disclosing not authorized data. For example, it is very usual to use values from a previous Select expression as runtime values for subsequent Select expressions. If this sequence is not enforced, security violations can occur because the provenance of the used runtime values cannot be guaranteed [9]. To overcome this situation, in this paper we propose an extension to the RBAC model to support the definition of sequences in which CRUD expressions must be executed. We demonstrate empirical evidence of the effectiveness of our proposal from a use case based on Java and JDBC.

This paper is organized as follows: section II presents the related work; section III presents our proposal; section IV presents the proof of concept and, finally, section V presents the final conclusion.

II. RELATED WORK

To the best of our knowledge no other researches have been conducted to provide RBAC models with the capability of controlling the sequences in which CRUD expressions are executed. Therefore, in this section we will present two main groups of aspects that are also closely related to this research: access control and service composition.

A. Access Control

In this sub-section we present access control in two main areas: models and techniques.

Models - Sandhu et at. [10] proposed the RBAC96, which comprises four models: RBAC0, RBAC1, RBAC2 and RBAC3. Since then, several proposals have been presented to extend these four RBAC models, among them we emphasize: credential based access control [11], temporal based access control [12], role delegation [13][14], context-aware [15],

system-to-system [16]. These and the remaining extensions are mainly focused on refining the role concept in order to adapt to particular contexts and scenarios.

Techniques - Several techniques have been proposed to protect the access to data, among them we emphasize: protection at tables and views level (vendors of RDBMS), the use of views [17], the use of parameterized views [18], the use of query rewriting techniques [19][20][17][21][22], extensions to SQL [23][24], programming languages extensions [25][26][27], security programming languages and tools [28][29][30][23][31] and, finally, semantic access control [32][33][34][35][36]. These proposals are based on a bundle of different techniques, each one with its own features. In spite of their relevance, none of them addresses the key issue of this research. They are mainly focused on controlling accesses to database objects (tables and views) on a CRUD by CRUD basis. A different technique is proposed in [37] where a security framework evaluates, at runtime, sequences of CRUD expressions in order to preserve data privacy. Beyond degrading the system performance this technique still needs to be subject to further experimentation.

Authors of this paper have also published about dynamic and distributed access control mechanisms [38][9]. Their researches were focused on techniques to implement static access control mechanisms at the level of business tiers. They were never focused on how to enforce sequences of CRUD expressions.

B. Service Coordination: Orchestration and Choreography

In our proposed approach, we needed to control the order in which users are authorized to use CRUD expressions. Two of the technologies that are used to control services workflow are: 1) service orchestration, which requires every service to be requested by a central control point; 2) service choreography, which allows a service to request the next service. Standard languages for each technique were proposed. Regarding orchestration, OASIS defined a standard language, which is still very active, called Web Services Business Process Execution Language [39] (WS-BPEL). WS-BPEL provides a set of functionalities that largely exceeds our needs. We will use some functionalities similar to those provided by WS-BPEL but tailored to our specific needs, such as graphs and life-cycle operations of active entities. Regarding choreography, the Web Service Choreography Description Language [40] (WS-CDL) is a language from W3C aimed at describing choreographies using the global view of the observable behavior of web services. However, the W3C Web Services Choreography working group was closed in 2009, leaving WS-CDL just as a candidate recommendation.

Several other languages exit, such as Yet Another Workflow Language [41] (YAWL) and XML Process Definition Language [42] (XPDL), but they clearly would not bring any advantage to our case.

The necessities to control the order in which sequences of CRUD expressions are executed are closer to a choreography process than an orchestration process. This way, we decided to implement our own technical approach in spite of having some similar features with the orchestration process, as already mentioned.

III. PROPOSED RBAC EXTENSION

This section is focused on presenting our RBAC extension. We start by presenting some of our previous work that is reutilized in this research, then we present the conceptual extension to be used in RBAC policies and, finally, a model extension is presented.

A. Business Schema

Before delving into the policy and its model, we start by analyzing some previous researches that we have been conducted around CRUD expressions [9][10][11][12][45]. From these researches we have defined and used the concept of Business Schema, which is basically a model from which source code can be automatically generated to handle CRUD expressions. The model, as in [9][37], can be driven by access control policies. Beyond being a model, Business Schemas have a cardinality of many to many with CRUD expressions. This means that one Business Schema can handle one or more CRUD expressions and one CRUD expression can be handled by one or more Business Schemas. Let us consider the next two Select expressions:

1) Select * from table; 2) Select * from table where col>10;

First we analyze the direction *"one Business Schema -> many CRUD expressions"*. Both expressions are Select and both have zero runtime values and the schema of the returned relations is the same. Then, the same Business Schema can be shared by both expressions. Now we analyze the direction *"one CRUD expression -> many Business Schemas"*. This case is simpler to explain and we can pick up any of the two Select expressions. In cases where different security policies are applied to the same Select expression, then we can use the Select expression in more than one Business Schema. For example, the same CRUD expression is managed by two Business Schemas where the runtime values are driven by different security policies.

B. RBAC Policy Extension

In this sub-section we present the new extension to the RBAC policy that is used to control the access requests to the data stored in relational database management systems (RDBMS). Traditionally, among other concepts, RBAC policies comprise: users, roles (they can be hierarchized), permissions, delegations and actions. Basically, legitimate (i.e. authenticated) users can only execute an action if he is authorized to play the role that controls that action. If a user is authorized to play a certain role, then he can perform all the actions controlled by that role. At the end, actions are the four main operations, provided by the Data Manipulation Language, on database objects (tables and views): read, insert, update and delete. The extension here presented aims at providing RBAC policies with the capability of controlling at the role level: 1) which Business Schemas and CRUD expressions are authorized; 2) in which sequence Business Schemas can be activated (instantiated) and, finally 3) the life-cycle of Business Schemas when the sequence moves forward one position. From this extension, security experts can now define new restrictions over the actions ruled by a role, particularly the ordered

sequences of actions (execution of CRUD expressions) users can perform.

C. RBAC Model Extension

In this sub-section we propose a model, shown in Figure 1, to formalize the extension proposed to the RBAC policy. This model is not unique and other formalizations can be used, depending on the practical scenario at hand. The extension herein proposed leverages our previous work where we proposed Business Schemas to model the access to relational databases based on CRUD expressions. The extension must take into account two main aspects. The first aspect is the functionality to connect Business Schemas and, therefore, to build sequences of Business Schemas. The second aspect is related to the life-cycle of Business Schemas when the sequence moves forward to the next Business Schema.

We start by presenting the first aspect. We will use directed graphs (usually known as digraphs) theory to formalize our approach for sequences (*S*) of Business Schemas. Basically, one vertex (*v)* is one Business Schema and one directed edge (*e*) connects one vertex (source vertex) to the next vertex (destination vertex). From a general digraph, there is the possibility to define several paths (sequence of vertices with each adjacent pair connected by a single direct edge). In an access control context, this type of freedom can raise some alerts when applied in systems where security is a key concern. Therefore, although our model relies on digraphs, some strict constraints need to be enforced. The main restriction lies on the impossibility of using diagraphs in their widest scope. Nevertheless, we can start by designing digraphs, although, from them we have to identify the paths that are considered to be in accordance with the access control policies. A path is a sequence of vertices, each one with one edge only (except the last one, which has no edge). Only these valid paths can be used and assigned to roles. This means that users authorized to play a role: 1) can only execute the Business Schemas (vertices) defined by the associated path and 2) in the order they are defined in the path (direct edges). In order to allow the assignment of several paths to one role, our model does not enforce any restriction at that level. This can be important in situations in which a role is defined to control the use of several forms, each one controlled with its own path, this way avoiding the need to define one role for each form.

Some definitions are now introduced. A root vertex (*r*) is the vertex where one path start. A leaf vertex (*l*) is a vertex with no edges and, therefore, is the last vertex of a path. A front vertex (*f*) is the vertex in which the sequence is running. Now we define the properties of paths in our model. Most of the concepts are shared by the theory of graphs. Even though, we present them to provide the necessary background for those who are not comfortable with graphs. The properties of paths are:

- one path comprises one and only one root vertex;
- one path comprises one and only one leaf vertex;
- self-edged vertices are not allowed in paths;
- loop-back vertices are not allowed in paths;
- any vertex of a digraph can be the root vertex of a path;
- any vertex of a diagraph can be the leaf vertex of a path;
- any vertex of a digraph can appear zero, one or more times in one path;
- adjacent vertices in paths must also be adjacent vertices in the parent digraph and connected with the same direct edge.

Next we present a simple example of a digraph, see Figure 1. It comprises 5 vertices and 6 edges, one of them is a self-edge (on vertex C). From this digraph, an indeterminate number of different paths can be defined. This indeterminate number of different paths has its origin on loop-back vertices (B to A) and self-edges (on C). Three of the possible paths that can be defined from Figure 1 are shown in Figure 2.

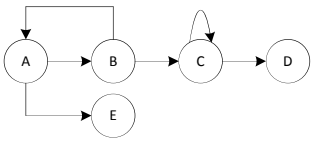

Figure 1. Example of a digraph.

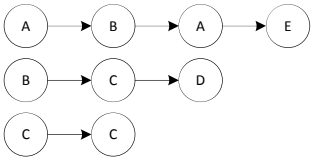

Figure 2. Examples of valid paths derived from Figure 1.

The second aspect is related to life-cycle of Business Schemas when a sequence moves to the next Business Schema. Once again, our approach provides a model in where security managers can freely choose the best option for their real scenarios. Basically, when the sequence moves to the next Business Schema, it is up to the security manager to decide which from the active Business Schemas are to remain active (their instances run normally) and which are not to remain active (their instances are running but they are disabled – methods do not execute the expected actions). The process to disable Business Schemas instances is herein referred to as the revocation process. In our model, each Business Schema has an associated list with all the previous Business Schemas to be revoked.

Finally, we present a possible and simplified model to formalize our proposal, see Figure 3. From it we can see that one role comprises one or more sequences (paths); a sequence comprises one or more Business Schemas (vertices); each Business Schema manages one or more CRUD expressions and, finally, each Business Schema (vertex - in a certain sequence and in a certain position) has a list of Business Schemas to be revoked.

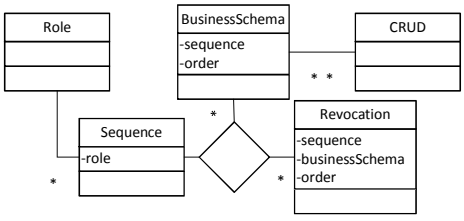

Figure 3. Example of a simplified diagram for our RBAC model extension.

IV. PROOF OF CONCEPT

In this section we present the scenario and the technical details of our implementation perspective for a RBAC model driven by our proposed extension.

A. Scenario

We implemented a scenario, which derives from previous researches, namely [11][12], where access control mechanisms have the following characteristics: 1) they are distributed in each client side application as typed security components; 2) they are automatically built and updated at runtime in accordance with users' profile and, 3) business logics use Call Level Interfaces (CLI) [46], such as ODBC [47] and JDBC [48], as the underlying middleware. Figure 4 presents the general block diagram for the implemented scenario. Basically, distributed security components are built from a metadata kept by a RBAC model (that implements our RBAC model extension) and modeled by a software architecture model. In our proof of concept, the software architectural model derives from previous software architectural models, which are closely aligned with CLI in order to keep their fundamental properties: fine tune control on the interactions with data stores and the use of native SQL languages (statements encoded inside strings this way keeping the full SQL expressiveness) [49].

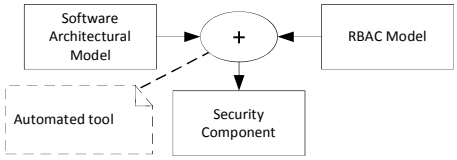

Figure 4. Implemented scenario general block diagram.

B. RBAC Model Extension

Figure 5 presents the implemented RBAC model. Part of it has been used in several researches we have conducted around distributed access control mechanisms. From the previous model we kept: subjects (Sub_Subject), applications (App_Applications), sessions (SES_Session), permissions (PER_Permission), delegations (Del_Delegation), hierarchized roles (Rol_Role), Business Schemas (Bus_BusinessSchema) and CRUD expressions (Crd_CRUD). Now, in order to support the extension herein proposed, some new entities were included: sequences (Seq_Sequence), revocations (Rev_Revocation) and aliases for Business Schemas (BSA_Alias). This extension clearly derives from the original proposal shown in Figure 3, but some additional clarifications are required. First of all, we extended the model in order to support many to many relational-ships between sequences, revocations and Business Schemas. Second, and perhaps the most noticeable modification, we included aliases for Business Schemas. In spite of not being mandatory, in our particular case, the use of aliases for Business Schemas led to significant improvements regarding the usability of our model by programmers of client applications. We now give the required details. From Figure 2 we see that the same Business Schema can be used in several vertices of the same path, for example in the first path, Business Schema A appears in positions 1 and 3. The first one is connected to Business Schema B and the second one is connected to Business Schema E. We could have built a unique Business Schema for the first path comprising edges for both destination Business Schemas. But this implementation, in spite of being possible, would require programmers to master the correct order of Business Schemas. The use of aliases, and also the aim of providing typed security components, allow us to instantiate different Business Schemas from the same base Business Schema. Basically, the main functionalities are shared by all Business Schemas instances derived from a Business Schema. The only two differences are: 1) the connection to the next destination Business Schema and 2) the list of Business Schemas to be revoked.

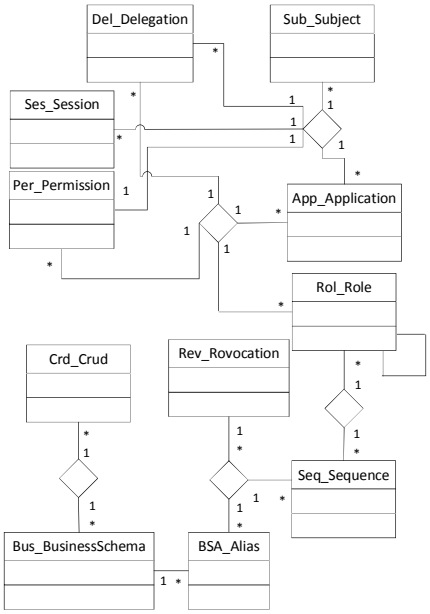

Figure 5. Implemented RBAC model with the proposed extension.

C. Architectural Model

In this sub-section we present the software architectural model, which is shown in Figure 6, for building automatically the enforcement mechanisms from the extended RBAC model. The presented architectural model represents the implementation of one role only. It is up to each system architect to decide how to expand it to support several roles. Moreover, it is focused on how to implement RBAC mechanisms and not on how to build complete and feasible implementations. For example, the architectural model does not address key issues such as the scrolling policy on local datasets (containers of data returned by Select expressions), sessions and database transactions. These and other issues are out of the architectural model context.

Our architectural model comprises three main types of entities: Factory, BusinessManager and Business Schemas. Next we describe each entity individually.

Factory - from applications' perspective, this entity is the entry point of our architectural model. Factory contains the identification of the first Business Schema of each valid sequence in that role. Programmers can select one of the root Business Schemas (one for each valid sequence) and also the CRUD expression to be executed (methods *getBS...*). There is one type safe method for each pair: root Business Schema and

each CRUD expression it supports.

Business Manager - Business Manager entity is the key entity in our architectural model. It is responsible for three main tasks: 1) to ensure that Business Schemas are instantiated in the correct order (sequences are kept in *sequences*); 2) to revoke active Business Schemas that are listed in the revocation list (*SequencyEntry - revokeList*) and, finally, 3) to validate calls to any Business Schemas' methods. This last task is required because when a Business Schema instance is revoked we need to programmatically ensure that all methods are put in a disabled state (in Java there is no possibility to explicitly destroy objects – they are destroyed by the garbage collector only when there is no reference to those objects).

The interaction between applications and our architectural model is as follows (please follow Figure 6): 1 - application selects one of the available sequences and requests the instantiation of its first Business Schema (in our example BusinessSchema_1) to manage the execution of one of the authorized CRUD expressions (there is one method for each Business Schema – CRUD expression pair); 1.1 – BusinessManager instantiates the Business Schema; 2 – Factory returns (to the application) a reference to the instantiated Business Schema 1; 2.1 – application uses the Business Schema 1 to interact with the host database; 2.1.1 – the call to every method is validated by Business Manager (if the Business Schema has been revoked, an exception is raised); 2.2) application requests the instantiation of BusinessSchema_2 (this is validated by Business Manager and eventually Business Schema 1 is revoked in case it is defined in the revocation list of Business Schema 2); 3 – Business Schema 2 is instantiated (only in case it has been validated by Business Manager) and a reference is returned to the application; 3.1 – application uses Business Schema 2 instance and 3.1.1 – every call to its methods is validated by Business Manager.

D. Use Case

From this architectural model and from metadata based on the proposed RBAC model extension, security layers are automatically built, see Figure 4. Our use case is based on Java, JDBC and uses the Microsoft Northwind database (http://www.microsoft.com/download/en/details.aspx?id=23654).

Table 1 and Table 2 partially present the information used in our use case, only for one role (Role_B1). Table 1 shows the supported Business Schemas and CRUD expressions. Table 2 shows the supported sequences. The bottom of both tables has some additional information to help with the understanding of their content. Now we present some snapshots of our concrete use case. Only some snapshots will be given in order to not overcrowd the paper. Figure 7 shows the high level data structures (classes are not shown) that were automatically created for Role_B1 and from the metadata represented by

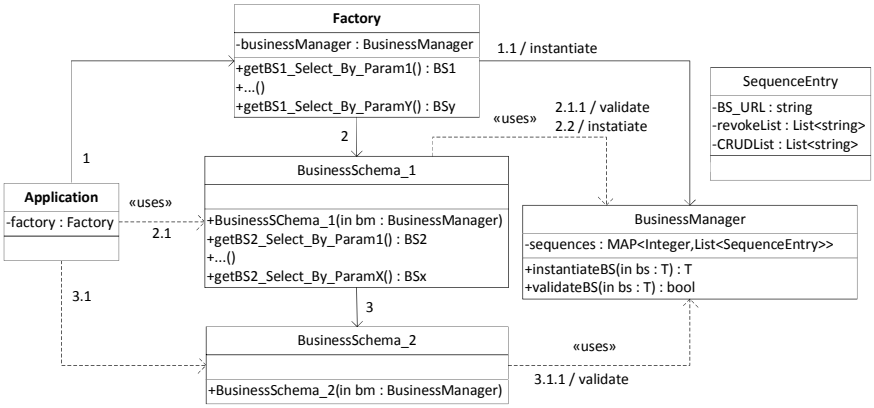

Figure 6. Software architectural model for the RBAC extension (one role).

Table 1. Roles, Business Schemas and CRUD expressions.

Role	Parent Role	BS	CRUD		
			Id	Ref	Expression
Role_A					
Role_B1	Role_A	S_Orders	1	byShipCountry	Select * From Orders Where CustomerId = ? and ShipCountry = ?
			2	byFreightLimit	Select * From Orders Where CustomerId = ? and Freight < ?
		I_Orders	3	withCustomerID	Insert Into Orders Values (?,?,?,?,?,?,?,?,?,?,?,?,?)
		S_Customers	4	all	Select * From Customers

Role: Role Reference
BS: Business Schema alias
Id: CRUD Identification
Ref: CRUD reference
Expression: CRUD Expression

Table 2. Roles, sequences and revocation lists.

Role	Sequence	Position	Sequence Entry		
			BS	RL	CRUDs
Role_B1	1	1	S_Customers		4
		2	S_Orders		1, 2
	2	1	I_Orders		3
		2	S_Customers	I_Orders	4
		3	S_Orders		1

Role: Role Reference
Sequence: The sequence identification
Position: The position in the sequence
BS: Business Schema alias
RL: The revocation list
CRUD: The list of authorized CRUDs

Table 1 and Table 2. They comprise the required information about the authorized Business Schemas and CRUD expressions

for Role_B1, only. These data structures are used during instantiation process of Business Schemas as we will show. Figure 8 presents the way applications interact with Factory classes. Factory classes are also built from metadata based on the extended RBAC model and have one method for each pair *Business Schema – CRUD expression*. From Figure 8 we see that the pair (Business Schema+CRUD expression) represented by *S_Customers_all* is requested to be instantiated (line 60) and then the CRUD expression is executed (line 61). Instances of root Business Schemas are instantiated by the Business Manager class under Factory's requests as shown in Figure 9 for the Business Schema S_Customers_all. Here we can see that the pair represented by S_Customers_all is s_*customers* and *s_customers_S_Customers_all* (lines 28, 29), which are internal data structures, shown in Figure 7, and, therefore, they

```
7  public abstract interface Role_IRole_B1 {
8      public static final java.lang.Class<II_Orders>
9              i_orders = II_Orders.class;
10     public static final int
11             i_orders_I_Orders_withCostumerID = 2;
12     public static final java.lang.Class<IS_Orders>
13             s_orders = IS_Orders.class;
14     public static final int
15             s_orders_S_Orders_byShipCountry = 1;
16     public static final java.lang.Class<IS_Customers>
17             s_customers = IS_Customers.class;
18     public static final int
19             s_customers_S_Custumers_all = 3;
20 }
```

Figure 7. Data structure comprising information about *Role_B1*

```
60      S_Cust = factory.get_S_Customers_all(session);
61      S_Cust.execute();
```

Figure 8. interaction between application and the Factory class.

```
25  public IS_Customers get_S_Customers_all(ISession session)
26  throws LocalTools.BTC_Exception {
27      return businessManager.instantiateBS(
28              Role_IRole_B1.s_customers,
29              Role_IRole_B1.s_customers_S_Custumers_all,
30              session);
31  }
```

Figure 9. The Factory requests an instance for a Business Schema.

```
65      S_Orders = S_Cust.get_S_Orders_by_shipCountry(session);
66      S_Orders.execute(S_Cust, "Portugal");
```

Figure 10. The root Business Schema provides an edge method to step forward to next Business Schema.

```
17  @Override
18  public IS_Orders get_S_Orders_by_shipCountry(ISession session)
19          throws BTC_Exception {
20      if(!businessManager.validateExecution(
21              activeSequence, "S_Customers.S_Customers")) {
22          throw new IllegalStateException(
23                  "S_Customers was used out of order!");
24      }
25
26      return businessManager.instantiateBS(
27              Role_IRole_B1.s_orders,
28              Role_IRole_B1.s_orders_S_Orders_byShipCountry,
29              session,
30              activeSequence
31      );
32  }
```

Figure 11. Validation process for the instantiation process of Business Schemas.

are not accessible from the application side. In order to access the second Business Schema, the root Business Schema provides a method to that goal, see Figure 10 (line 65). Before being instantiated, Business manager evaluates if the requested Business Schema can be instantiated, see Figure 11. If authorization is not granted, an exception is raised (line 22-23). Otherwise, an instance is returned (line 26-31).

These are only some snapshots of our use case. From these snapshots, from Table 1 and Table 2, and from Figure 6 we hope that readers can infer the remaining technical approaches on which our use case was built.

V. CONCLUSION

This paper presents an extension of the basic RBAC policy in order to control the sequence in which CRUD expressions are executed. The model leverages previous researches where the RBAC model was extended to support the concept of Business Schema and, therefore, the execution of CRUD expressions. To formalize our proposed extension we start by resorting to the theory of graphs. From it we defined an ordered sequence of Business Schemas as being a path. Basically, the model extension allows security experts to designate the set of Business Schemas sequences to be used by users authorized to play that role. A proof of concept is also presented, which, once again, leverages previous researches. The presented proof of concept, beyond providing the empirical evidence of the effectiveness of our proposal, it also conveys the following key advantages: 1) security layers are automatically built and 2) security layers are based on typed objects, this way relieving programmers of application tiers from mastering the established security policies.

REFERENCES

[1] P. Samarati and S. D. C. di Vimercati, "Access Control: Policies, Models, and Mechanisms," *Found. Secur. Anal. Des.*, vol. 2171, pp. 137–196, 2001.

[2] S. D. C. di Vimercati, S. Foresti, and P. Samarati, "Recent Advances in Access Control - Handbook of Database Security," M. Gertz and S. Jajodia, Eds. Springer US, 2008, pp. 1–26.

[3] R. S. Sandhu and P. Samarati, "Access Control: Principle and Practice," *Commun. Mag. IEEE*, vol. 32, no. 9, pp. 40–48, 1994.

[4] M. A. Al-Kahtani and R. Sandhu, "A Model for Attribute-Based User-Role Assignment," *Proceedings of the 18th Annual Computer Security Applications Conference*. IEEE Computer Society, pp. 353–362, 2002.

[5] D. R. Kuhn, E. J. Coyne, and T. R. Weil, "Adding Attributes to Role-Based Access Control," *Computer (Long. Beach. Calif).*, vol. 43, no. 6, pp. 79–81, 2010.

[6] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed NIST Standard for Role-based Access Control," *ACM Trans. Inf. Syst. Secur.*, vol. 4, no. 3, pp. 224–274, 2001.

[7] R. Sandhu, D. Ferraiolo, and R. Kuhn, "The NIST Model for Role-based Access Control: Towards a Unified Standard," *5th ACM Workshop on Role-based Access Control*. ACM, Berlin, Germany, pp. 47–63, 2000.

[8] L. Fuchs, G. Pernul, and R. Sandhu, "Roles in information security – A survey and classification of the research area," *Comput. Secur.*, vol. 30, no. 8, pp. 748–769, 2011.

[9] Ó. M. Pereira, R. L. Aguiar, and M. Y. Santos, "Runtime Values Driven by Access Control Policies Statically Enforced at the Level of the Relational Business Tiers," in *SEKE'13 - Intl. Conf. on Software Engineering and Knowledge Engineering*, 2013, pp. 1–7.

[10] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-Based Access Control Models," *Computer (Long. Beach. Calif).*, vol. 29, no. 2, pp. 38–47, 1996.

[11] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis, "The KeyNote Trust-Management System Version 2." RFC Editor, United States, 1999.

[12] E. Bertino, P. A. Bonatti, and E. Ferrari, "TRBAC: A Temporal Role-based Access Control Model," *ACM Trans. Inf. Syst. Secur.*, vol. 4, no. 3, pp. 191–233, 2001.

[13] X. Zhang, S. Oh, and R. Sandhu, "PBDM: A Flexible Delegation Model in RBAC," in *Proceedings of the Eighth ACM Symposium on Access Control Models and Technologies*, 2003, pp. 149–157.

[14] E. Barka and R. Sandhu, "Framework for Role-based Delegation Models," in *Proceedings of the 16th Annual Computer Security Applications Conference*, 2000, p. 168–.

[15] D. Kulkarni and A. Tripathi, "Context-aware role-based access control in pervasive computing systems," *13th ACM Symposium on Access Control Models and Technologies*. ACM, Estes Park, CO, USA, pp. 113–122, 2008.

[16] S. Haibo and H. Fan, "A Context-Aware Role-Based Access Control Model for Web Services," in *Proceedings of the IEEE International Conference on e-Business Engineering*, 2005, pp. 220–223.

[17] S. Rizvi, A. Mendelzon, S. Sudarshan, and P. Roy, "Extending Query Rewriting Techniques for Fine-grained Access Control," *ACM SIGMOD Int. Conf. on Management of Data*. ACM, Paris, France, pp. 551–562, 2004.

[18] A. Roichman and E. Gudes, "Fine-grained access control to web databases," *12th ACM symposium on Access Control Models and Technologies*. ACM, Sophia Antipolis, France, pp. 31–40, 2007.

[19] Oracle, "Using Oracle Virtual Private Database to Control Data Access," 2011. [Online]. Available: http://docs.oracle.com/cd/B28359_01/network.111/b28531/vpd.htm#CIHBAJGI.

[20] K. LeFevre, R. Agrawal, V. Ercegovac, R. Ramakrishnan, Y. Xu, and D. DeWitt, "Limiting disclosure in hippocratic databases," *30th Int. Conf. on Very Large Databases*. VLDB Endowment, Toronto, Canada, pp. 108–119, 2004.

[21] Q. Wang, T. Yu, N. Li, J. Lobo, E. Bertino, K. Irwin, and J.-W. Byun, "On the correctness criteria of fine-grained access control in relational databases," *33rd Int. Conf. on Very Large Data Bases*. VLDB Endowment, Vienna, Austria, pp. 555–566, 2007.

[22] S. Barker, "Dynamic Meta-level Access Control in SQL," *22nd Annual IFIP WG 11.3 Working Conf. on Data and Applications Security*. Springer-Verlag, London, UK, pp. 1–16, 2008.

[23] A. Chlipala, "Static checking of dynamically-varying security policies in database-backed applications," in *9th USENIX Conf. on Operating Systems Design and Implementation*, 2010, pp. 1–14.

[24] S. Chaudhuri, T. Dutta, and S. Sudarshan, "Fine Grained Authorization Through Predicated Grants," *IEEE 23rd ICDE - Int. Conf. on Data Engineering*. Istanbul, Turkey, pp. 1174–1183, 2007.

[25] B. J. Corcoran, N. Swamy, and M. Hicks, "Cross-tier, Label-based Security Enforcement for Web Applications," *35th SIGMOD Int. Conf. on Management of Data*. ACM, Providence, Rhode Island, USA, pp. 269–282, 2009.

[26] J. Fischer, D. Marino, R. Majumdar, and T. Millstein, "Fine-Grained Access Control with Object-Sensitive Roles," *23rd ECOOP - European Conference on Object-Oriented Programming*. Springer-Verlag, Italy, pp. 173–194, 2009.

[27] J. Zarnett, M. Tripunitara, and P. Lam, "google.pt," *Proceedings of the 15th ACM symposium on Access control models and technologies*. ACM, Pittsburgh, Pennsylvania, USA, pp. 79–88, 2010.

[28] L. Caires, J. A. Pérez, J. C. Seco, H. T. Vieira, and L. Ferrão, "Type-based access control in data-centric systems," *20th European conference on Programming Languages and Systems: part of the joint European conferences on theory and practice of software*. Springer-Verlag, Saarbrucken, Germany, pp. 136–155, 2011.

[29] N. Swamy, B. J. Corcoran, and M. Hicks, "Fable: A Language for Enforcing User-defined Security Policies," in *IEEE Symposium on Security and Privacy*, 2008, pp. 369–383.

[30] D. Zhang, O. Arden, K. Vikram, S. Chong, and A. Myers, "Jif: Java + information flow (3.3)," 2012. [Online]. Available: http://www.cs.cornell.edu/jif/.

[31] C. Ribeiro, A. Zúquete, P. Ferreira, and P. Guedes, "SPL: An Access Control Language for Security Policies with Complex Constraints," *Network and Distributed System Security Symposium*. San Diego,CA,USA, pp. 89–107, 2001.

[32] Y.-J. Hu and J.-J. Yang, "A semantic privacy-preserving model for data sharing and integration," *Proceedings of the International Conference on Web Intelligence, Mining and Semantics*. ACM, Sogndal, Norway, pp. 1–12, 2011.

[33] C.-C. Pan, P. Mitra, and P. Liu, "Semantic access control for information interoperation," *Proceedings of the eleventh ACM symposium on Access control models and technologies*. ACM, Lake Tahoe, California, USA, pp. 237–246, 2006.

[34] J. Warner, V. Atluri, R. Mukkamala, and J. Vaidya, "Using semantics for automatic enforcement of access control policies among dynamic coalitions," *Proceedings of the 12th ACM symposium on Access control models and technologies*. ACM, Sophia Antipolis, France, pp. 235–244, 2007.

[35] J. Lopez, A. Mana, E. Pimentel, J. M. Troya, and M. I. Y. e del Valle, "Access Control Infrastructure for Digital Objects," *Proceedings of the 4th International Conference on Information and Communications Security*. Springer-Verlag, pp. 399–410, 2002.

[36] K. Il Kim, W. Y. Kim, J. S. Ryu, H. J. Ko, U. M. Kim, and W. J. Kang, "RBAC-based access control for privacy preserving in semantic web," *Proceedings of the 4th International Conference on Uniquitous Information Management and Communication*. ACM, Suwon, Republic of Korea, pp. 1–5, 2010.

[37] Canfora, G.; Visaggio, C.A.; Paradiso, V., "A Test Framework for Assessing Effectiveness of the Data Privacy Policy's Implementation into Relational Databases," in *Intl. Conf. on Availability, Riliability and Security*, 2009, pp. 240–247.

[38] Ó. M. Pereira, R. L. Aguiar, and M. Y. Santos, "ACADA - Access Control-driven Architecture with Dynamic Adaptation," in *SEKE'12 - 24th Intl. Conf. on Software Engineering and Knowledge Engineering*, 2012, pp. 387–393.

[39] OASIS, "WS-BPEL Especification." [Online]. Available: http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf.

[40] W3C, "Web Services Choreography Description Language." [Online]. Available: http://www.w3.org/TR/ws-cdl-10/.

[41] "YAWL." [Online]. Available: http://yawlfoundation.org.

[42] "XML Process Definition Language." [Online]. Available: http://www.xpdl.org/.

[43] O. M. Pereira, R. L. Aguiar, and M. Y. Santos, "CRUD-DOM: A Model for Bridging the Gap Between the Object-Oriented and the Relational Paradigms," in *ICSEA 2010 - Int. Conf. on Software Engineering and Applications*, 2010, pp. 114–122.

[44] O. M. Pereira, R. L. Aguiar, and M. Y. Santos, "CRUD-DOM: A Model for Bridging the Gap Between the Object-Oriented and the Relational Paradigms - an Enhanced Performance Assessment Based on a case Study," *Int. J. Adv. Softw.*, vol. 4, no. 1&2, pp. 158–180, 2011.

[45] O. M. Pereira, R. L. Aguiar, and M. Y. Santos, "An Adaptable Business Component Based on Pre-defined Business Interfaces," in *6th ENASE: Evaluation of Novel Approaches to Software Engineering*, 2011, pp. 92–103.

[46] ISO, "ISO/IEC 9075-3:2003," 2003. [Online]. Available: http://www.iso.org/iso/catalogue_detail.htm?csnumber=34134.

[47] Microsoft, "Microsoft Open Database Connectivity," 1992. [Online]. Available: http://msdn.microsoft.com/en-us/library/ms710252(VS.85).aspx.

[48] M. Parsian, *JDBC Recipes: A Problem-Solution Approach*. NY, USA: Apress, 2005.

[49] M. David, "Representing database programs as objects," in *Advances in Database Programming Languages*, F. Bancilhon and P. Buneman, Eds. N.Y.: ACM, 1990, pp. 377–386.