A Closer Look to the V-Model Approach for Role Engineering

RADU CONSTANTINESCU, ANDREI TOMA
The Economic Informatics Department
Academy of Economic Studies
Bucharest, Piata Romana n^{o} 62
ROMANIA
radu.constantinescu@ie.ase.ro, andrei.toma@ie.ase.ro

Abstract: - Role engineering is a both necessary and critical topic in the development of Role Based Access Control system, which seems to be the most proficient access control approach nowadays. Even if the maturity of the *RBAC* model has been already achieved, the role engineering process is not a standardized approach. The paper aims to illustrate an enhanced process model for role engineering. The model is focused on the intuitive discovering of the roles and their assignment with permissions, using a test-driven approach.

Key-Words: - access control systems, role based access control systems – RBAC, role engineering, authorization, roles, permissions, constraints, role hierarchies.

1 Introduction

The most sensitive characteristics of an information security system are linked to the approach of addressing the access control issue. For mature organizations, with clear defined responsibilities, the most suitable model is role based access control – *RBAC* [6], [7], [11]. The implementation of an access control model should be an important objective for the informational system implementation process. [17], [18] However, many organizations have already automated their business processes without taking in consideration access control as a mandatory and also extended approach. This is why a rigorous and comprehensive analysis has to be in order to achieve the correct permissions and role assignments as a prerequisite for implementing an access control system based on roles [1]. The efficiency of the whole engineering process depends on the ability of the designers to extract the correct assignments. For this purpose, a flexible methodology sustaining a coherent process of role engineering is required. This article extends the concepts presented in the WSEAS conference on computing ICC 2009 [29].

2 Problem Formulation

The objective of this paper is to design a process model for role engineering. The process model should be enforced by a specific methodology suited to identify the permissions and roles inside the information system of an organization and to identify also the assignments between those two sets. The methodology should support the design of role hierarchy and the identification of constraints.

Regarding the status of research in this area, features of role engineering have been discussed starting with Coyne's paper. In this initial paper the main tasks of role engineering are defined. The mentioned tasks are: role definition, roles hierarchy definition, constraint definition and the mapping between roles and permissions [2].

Neumann and Strembeck have proposed an approach for role engineering based on scenarios [10]. In their model, each activity is described based on a set of scenarios and each scenario is then decomposed in several steps. Each step is then mapped to correspondent permissions. The approach has the disadvantage that it requires a great effort in order to comprehensively determine the possible scenarios.

Crook and Ince designed a conceptual framework for role engineering based on organizational structures. This approach helps determining roles but is not a comprehensive one [3]. Epstein suggested another approach of adding additional layers in order to ease role engineering. The approach is detailed in both top-down and bottom-up manner. The model takes the presumption that roles and permissions are already determined, so it doesn't describe how those items will be defined. Neither the role hierarchy nor constraints definition is documented [5].

Goncalves and Maranda have proposed a role engineering method based on *UML* in correlation with system's features. Functions are a middle layer between roles and permissions. A role can be mapped with several functions and each function will require access rights. The approach lacks non-functional items [8].

There are also approaches aimed to elicit the *RBAC* role model based on data mining methods as those in [21],

[22], [24], [25] or based on graph optimization as in [26].

3 Problem Solution

3.1 *RBAC* Model Overview

RBAC model is focused on roles which are associated to different functions existing in systems or in organizations [27], [28], [29]. The roles are associated to users depending on their responsibilities and/or qualification. Roles are also associated with permissions. The association of users and roles tends to change faster than the association between roles and permissions as the organizational frameworks tends to modify slower than the way in which users are allocated to tasks.
RBAC is also used as a foundation for complex access control mechanisms. *RBAC* is appropriate to be used in organizations that focus more on integrity than on confidentiality, so it is suitable for economical applications. It implements role hierarchies and constraints. The conceptual model is layered. The first layer is $RBAC_0$ which is the base model. On top of it are two independent layers $RBAC_1$ and $RBAC_2$ which are concerned with role hierarchies and respectively constrains. $RBAC_3$ consolidates both $RBAC_1$ and $RBAC_2$. On top of $RBAC_3$, security architects can define and implement customized levels for specific activities. [12]
The main components of *RBAC* model are: users (U), roles (R), permissions (P) and sessions (S). A user can be defined as a human being but in automated systems it also can represent an entity. Roles are job functions related to the organizations' or systems' particularities. A role implies, as we already mentioned, competency, responsibility and authority. Permissions are related to the specific functions inside organizations. A role can be assigned to one-to-many persons and also one person can have one-to-many roles. This is also valid for role-permissions relationship. A session is a mapping between a user and an activated subset of roles that are assigned to him. [11], [15], [16] In the basic model the relations defined are:

- Permission assignment between permissions and roles: $PA \subseteq P \times R$
- User assignment between users and roles: $UA \subseteq U \times R$
- Session-user mapping: $user: S \rightarrow U$
- Session-roles mapping: $roles: S \rightarrow 2^R$, $roles(s_i) \subseteq \{r | (user(s_i), r) \in UA\}$ which can change in time and session s_i has permissions $U_{r \in roles(s_i)} \{p | (p,r) \in PA\}$.

Regarding permissions, we distinguish between standard permissions and administrative permissions. Standard permissions refer to data and resource objects. Administrative permissions are required to modify the sets of users, roles, permissions and the relations between them. The administrative permissions are managed in a dual *RBAC* system. Sessions are in control of individual users. A user can create sessions and can choose to activate a subset of possible roles and then he can modify the active roles and even close the session.
$RBAC_1$ model introduces the role hierarchy concept. [7] This is used to structure the roles in order to reflect organizations' patterns. The most important roles stay in the top of the hierarchy inheriting permissions from the less important roles. This relation is a partially ordered relationship which means that it is reflexive, transitive and anti-symmetrical. In consequence, a role inherits its own permissions and the permissions of lower roles connected directly or indirectly with it and the bidirectional inheritance is denied in order to exclude redundancy. Therefore, $RBAC_1$ introduces a new relation: $RH \subseteq R \times R$, and the function roles is modified accordingly: $roles(s_i) \subseteq \{r | (\exists r' \geq r)[(user(s_i), r') \in UA\}$ and session s_i has the permissions $U_{r \in roles(s_i)} \{p | (\exists r'' \leq r) (p, r'') \in PA\}$.

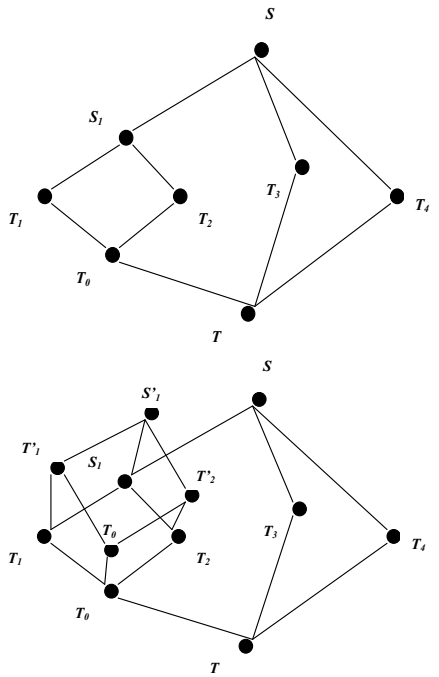

Figure 1. Standard role hierarchy and private role hierarchy

There are many common cases in which is desired that some permission should remain available only to specific roles and to avoid inheritance to superior levels. This issue can be solved by defining private roles that are not inherited. This is represented in *Figure 1*. In the first graph is represented a role hierarchy that contains the roles T, T_0, T_1, T_2, T_3, T_4, S_1 and S. The role T is the

base role for this example and is inherited by all the other roles. Role *S* is the top role. S_1 is a subproject role. In case there are permissions for S_1 that shouldn't be inherited by *S*, then the whole subproject area of the graph is replicated using private roles as in the second picture of *Figure 1*. [13]

$RBAC_2$ model introduces the concept of constrains which are mandatory for any organization. In the category of constrains we can mention the mutual exclusion of roles, cardinality restrictions or the prerequisite roles constraint. The restrictions should be simple in order to be easily implemented in an efficient way. Restrictions can be related to sessions and to the *roles()* and *user()* functions. Even though the role hierarchy is also a constraint, given its importance, it is discussed in the context of a different layer – $RBAC_1$. [13]

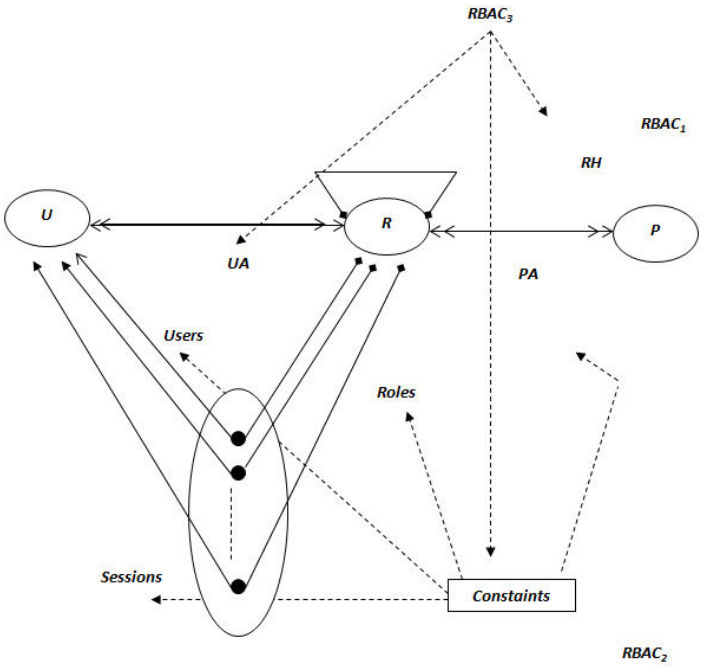

Figure 2. *RBAC* model [13]

The consolidated model combines $RBAC_1$ and $RBAC_2$ as depicted in *Figure 2* There are some problems that arise from this association. The constraints in $RBAC_2$ could limit the number of superior or inferior roles that a role can have. There are also situations in which the inheritance could violate restrictions like mutual exclusion between roles. This issue can be addressed using private roles, as in *Figure 1*.

Given the variables of the model that were already discussed in the article, we will present the formal description of the basic properties of *RBAC* model. The properties are known in literature as: consistent subject, role assignment, role authorization, privilege authorization and role hierarchy. [14]

The consistent subject property states that for any subject *s* associated with user *u*, the authorized role set *R[s]* includes role *r* if and only if *u* is authorized for *r*:

$$(\forall s)(\forall u)(\forall r) \mid U[s] = u, u \in M[r] \Leftrightarrow r \in R[s] \quad (1)$$

Where $M[r]$ represents the set of users authorized for role *r* and $R[s]$ is the set of roles for which subject *s* is authorized.

The role assignment property states that a subject can execute a privilege only if the subject is assigned a role that is active at that moment:

$$(\forall s)(\forall p) \mid X[s, p] \Rightarrow AR[s] \neq \phi \quad (2)$$

Where $X[s, p]$ is the true if and only if subject *s* is able to execute permission *p* and $AR[s]$ is the set of active roles for *s*.

The role authorization property states that a subject's active role must be in the set of authorized roles for the subject:

$$(\forall s) \mid r \in AR[s] \Rightarrow r \in R[s] \quad (3)$$

The privilege authorization property states that a subject can execute a certain permission only if the permission is assigned to the active role of the subject:

$$(\forall s)(\forall p)(\exists r) \mid X[s, p] \Rightarrow r \in AR[s] \wedge p \in PA[r] \quad (4)$$

Where $PA[r]$ is the set of permissions associated with role *r*.

The role hierarchy property states that a authorized role includes the permissions of the roles it inherits.

$$(\forall r, q)(\forall s) \mid (r \in AR[s] \wedge r \geq q \Rightarrow q \in AR[s]) \wedge (r \in R[s] \wedge r \geq q \Rightarrow q \in R[s]) \quad (5)$$

3.2 Solution Overview

The paper presents an extended *RBAC* model, designed by the authors in order to enhance the role engineering process. The proposed model, *VMRE-RBAC (V-Model Role Engineering RBAC)*, facilitates the decomposition of roles in permissions and then the results' testing. The model is incremental and iterative. Every decomposition stage will have a correspondent testing stage and for every testing stage a new optimization of results is achieved. The testing stage is concerned with validation and verification of the results.

The paper is based on *RBAC* model which was already described in the former section. The components of the initial *RBAC* model are: users, roles, permissions and also the relations between those elements. The initial *RBAC* model has been enhanced by adding role hierarchies and constraints. *RBAC96* cumulates those

issues. One of the advantages of *RBAC* model is that it implements several major principles of computer security: least privilege, separation of duty and administration and also data-abstraction [9], [11].
The role engineering is a critical stage for any *RBAC* implementation [23]. *VMRE-RBAC* model extends the standard *RBAC* model by adding extra layers between roles and permissions: profiles, tasks and steps. The roles are determined and defined starting from a well-known set of roles given the specific organizational structure. Those roles are associated with a set of goals and each goal determines responsibilities. Each responsibility is carried out through a specific profile. We suggest that profiles should be lower layer roles in the final role hierarchy. Each profile is then decomposed in tasks and the tasks are decomposed in steps. Steps are assigned to different sets of permissions. The decomposition can be driven by role or functionality issues.
After an initial decomposition process, the results will be tested incrementally. Testing means both verification and validation for the entities designed on each layer. In order to ease this process, we propose a set of nine properties which includes: equivalence, minimization, reuse, completeness, consistency and coherence. These properties are defined in order to obtain a simple, complete and non-equivocal model. The validation is driven by scenarios and responsibilities.
The model is flexible, new permissions, tasks, profiles and roles should be obtained in future iterations. Even new organizational responsibilities should be added to the existing roles by using profiles. The methodology contains all the steps needed for role engineering: identifying roles and permissions, mapping roles to permissions, identifying constraints and building role hierarchies.

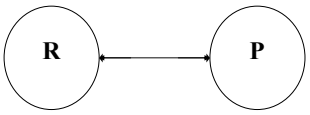

Figure 3. Roles and permissions association in NIST's RBAC standard

As a start, we use the standardized *RBAC96* relation between permissions and roles in order to build roles as a superset of permissions depicted in *Figure 3*. We suggest decomposing the mapping of roles and permissions in several mappings between several middle layers. We propose the usage of three middle layers between roles and permissions, which are: profiles, tasks and steps as in *Figure 4* Each of these layers should be treated independently.

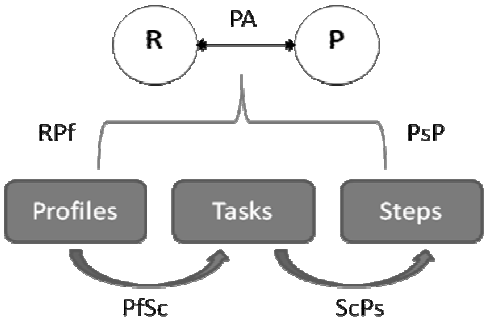

Figure 4. Extending Roles-Permission Association

The role engineering process consists in two main sub-processes. The first process is focused on a way to decompose the roles in permissions using a top-down approach. The second process tests the results using a bottom-up approach.

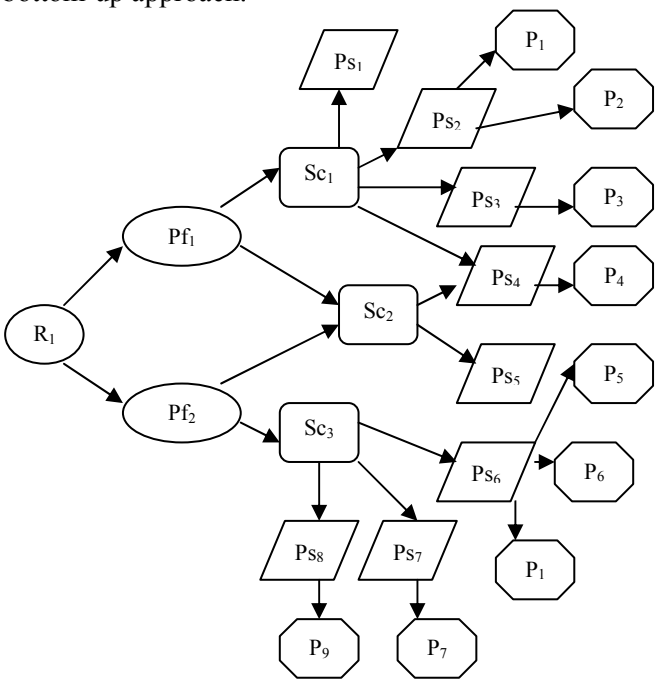

Figure 5. Decomposition of roles in permissions

The roles represent the upper layer of *VMRE-RBAC* model. A role is determined initially based on the profile of the company in which the model is applied. A role can be associated with several responsibilities. Each major responsibility is then associated with a specific profile. The proposed roles will be validated taking in account the goals of the activity which are elicited, defined and refined during the engineering process. Each user associated with a profile is responsible for a set of tasks specific to the organization. The validation of these tasks should be made by several test scenarios. The scenarios are designed as use-cases for the organization's informational system. Each task can be interpreted as a set of multiple steps which are executed in a specific logical order. In the end, each step is related

to a set of permissions. The grouping of multiple steps in a task determines the mapping between a set of permissions and a task. If a task is shared by different profiles, it will be reused. The same principle applies also to permissions, steps and profiles.
In *Figure 5*, we present a schema of possible results of *VMRE-RBAC* decomposition process. For simplicity, the picture includes a single role R_1 which is linked to two profiles Pf_1 and Pf_2. For each profile we identified the tasks assigned. In this particular case, the profiles have multiple tasks assigned from which Sc_2 is shared. Also, there are several steps reused and each step implies at least one assigned permission.

3.2 VMRE-RBAC Model

The standard elements of *RBAC* model - users, roles and permissions - are also used in the proposed extension. *RBAC* standard defines the mapping between roles and permissions, as depicted in *Figure 5*, but it doesn't explicit how this mapping is achieved. We aim to propose a role-engineering solution in this concern.

We will detail the relation between roles and permissions starting from the *NIST*'s standardized *RBAC* model conventions:

- R – roles set
- P – permissions set
- $PA \subseteq P \times R$ association between permissions and roles

The general model for *VMRE-RBAC* is presented in *Figure 6.*

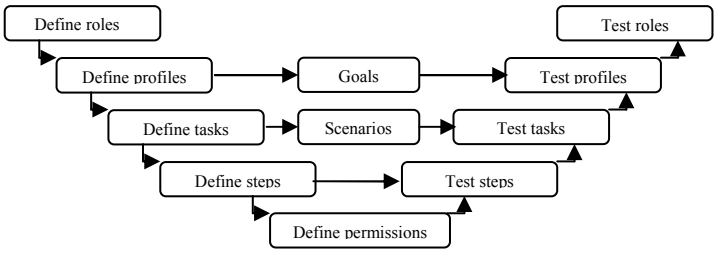

Figure 6. Decomposition and testing in VMRE-RBAC

Starting from the classic *RBAC96* model, we added three middle-layers which are: profiles, tasks and steps. Each decomposition stage has an equivalent testing stage. In order to formalize the model, we use the following notations:

- R – roles set
- Pf – profiles set
- Sc – tasks set
- Ps – steps set
- P – permissions set
- $RPf \subseteq R \times Pf$ – many to many relation between roles and profiles
- $PfSc \subseteq Pf \times Sc$ – many to many relation between profiles and tasks
- $ScPs \subseteq Sc \times Ps$ – many to many relation between tasks and steps
- $PsP \subseteq Ps \times P$ – many to many relation between steps and permissions

An important goal of the paper is to support a methodology which should optimize the process of mapping between roles and permissions. We suggest a series of operations that will help the role-engineering process: minimize the sets at every layer, reuse elements, verify if constraints are in place and verify the complete mapping between elements in adjacent sets. The testing and optimization criteria are formalized in nine properties. The properties deal both with the elements on the same layer and with the mappings between elements in adjacent layers. In other words, the properties apply both on elements and relations. The properties we propose are: equivalence, uniqueness, completeness, reused element, minimum, consistency and coherence. Properties like equivalence, uniqueness, completeness or reused element are also discussed in former papers as [5].
As we already mentioned, it is desirable that the number of roles, profiles, tasks and steps used in relations to be minimal. Ideally, each element is unique and the designer eliminates any duplicate. Additionally, the uniqueness of each element means that there is no other element that is equivalent to it. The equivalence can be demonstrated by comparing how sets of elements from a layer are mapped to elements on the upper layer. If there are equivalent elements, they should be minimized.
The role engineer verifies that all the roles defined have at least a permission assigned to it. Also, each element on one layer should be mapped to at least one element on both adjacent layers. This is formalized in the completeness property. The role engineer also test the consistency of each role and profile, meaning that the final set of permissions determined will sustain the achievement of all declared goals. The consistency of tasks is tested through scenarios.
We present a formal description of those nine properties using the following conventions:

- A – set of elements on one layer
- B – set of elements on the next layer
- $f:A \rightarrow B$, $f(a)=b$ where $a \in A$ and $b \in B$ – basic mapping
- $f^1:B \rightarrow A$, $f^1(b)=a$ where $a \in A$ and $b \in B$ –basic reverse mapping
- $f(a:A) \rightarrow 2^B$ – mapping between an element on layer A and a set of

elements on layer B, included in the set of the parts of B.
- P – permission set
- O – goals set
- S – scenario set
- C_p – constraints catalog for permission layer
- C_r – constraints catalog for role layer
- C_{pf} – constraints catalog for profile layer

Based on the former conventions, the following nine properties are defined:

Property 1: *Equivalence*
Let:
$f(a_i:A) \rightarrow B_k$
$f(a_j:A) \rightarrow B_l$
$a_i \neq a_j$

If $B_k=B_l$ then element a_i is equivalent with element a_j in the set A; $a_i \approx a_j$

Property 2: *Permission equivalence*
Let:
$f(a_i:A) \rightarrow P_k$
$f(a_j:A) \rightarrow P_l$
$a_i \neq a_j$

If $P_k=P_l$ then element a_i is permission equivalent with element a_j in the set A; notation: $a_i \approx_p a_j$

Property 3: *Uniqueness*
Let:
$a, a_i \in A$

If $! a \approx a_i (\forall) a_i \in A\text{-}\{a\}$ then a is unique

Property 4: *Minimum*
Let:
$a, a_i \in A$
$f(a:A) \rightarrow B$
$f(a_i:A) \rightarrow B_i$

For $(\forall) a_i$ equivalent, then a is the minimum between a_i, $B = \cup B_i$

Property 5: *Reused element*
Let:
$f(a_i:A) \rightarrow B_k$
$f(a_j:A) \rightarrow B_l$

If $b \in B_k \cap B_l$ the elements a_i and a_j reuse element b

Property 6: *Completeness*
If $f:A \rightarrow B$ and $f^1:B \rightarrow A$ are surjective then (A,B) is a complete relation

Property 7: *Consistency*
For $(\forall) s \in S$, $(\exists)\ pf \in Pf \wedge sc_1, \ldots, sc_n \in Sc(pf)$ so that:
$\bigcup_{k=1}^{n} sc_k \approx_p s$

Property 8: *The coherence of constraints at permissions layer for the upper layers*
Let:
$(\forall) c \in C_p \wedge (\forall) p \in P(c)$
then for
$(\forall) pf \in Pf(p) \wedge (\forall) r \in R(p)$

$Valid(pf,c)=1 \wedge Valid(r,c)=1$, where $Valid: A \times C_p \rightarrow \{0,1\}$, $A=Pf \cup R$

Property 9: *The coherence of constraints assigned to profiles and roles layers for the permission layers*
a) For:
$(\forall) c \in C_{pf} \wedge (\forall) pf \in Pf(c), (\forall) p \in P(pf)$
the following relation is achieved:

$Valid(p,c)=1$
where $Valid: P \times C_{pf} \rightarrow \{0,1\}$
b) For:
$(\forall)\ c \in C_r \wedge (\forall) r \in R(c), (\forall) p \in P(r)$
the following relation is achieved:

$Valid(p,c)=1,$
where $Valid: P \times C_r \rightarrow \{0,1\}$

The identification of constraints is one of the most challenging tasks in the design. It is important that all the constraints should be identified and tested. We propose the coherence property in this matter.
In *Table 1* we present the correspondence between the defined properties and the defined layers.
The equivalence property applies to *R, Pf*, *Sc* and *Ps* layers. Permission layer is not included here because it is the lowest hierarchical layer so the property is not appropriate. Two sets of elements on the same level are equivalent if they have the same elements linked to on the next level. The equivalence of permissions can be done by sorting the permission catalog, given the fact

that each permission is a tuple *(operation, object)*. The operations might be generically defined as elements of *{C,R,U,D,E}* set, where *C* is the create operation, *R* is read operation, *U* is update operation, *D* is delete operation and *E* is execute operation. The operations can be defined also as abstract functions determined by the application's specificity.

Table 1. Correlation of properties and steps in *VMRE-RBAC* model

	Role	Profile	Task	Step	Permission
Equivalence (*P1*)	×	×	×	×	
Equivalence/Perm (*P2*)	×	×	×		
Uniqueness (*P3*)	×	×	×	×	×
Minimum (*P4*)	×	×	×	×	×
Reused element (*P5*)	×	×	×	×	×
Completeness (*P6*)	×	×	×	×	×
Consistency (*P7*)	×	×	×		
Coherence/Roles (*P8*)	×	×			
Coherence/Perm (*P9*)					×

Permission equivalence makes sense for tasks, profiles and roles. Permission equivalent elements are not always compliant with *property 1* even if elements compliant with *property 1* are always permission equivalent. For example, let task *A* involves steps Ps_1, Ps_2 and Ps_3, and task *B* involves steps Ps_2 and Ps_3. If the permissions mapped to Ps_1 are the same with the reunion of those related to Ps_2 and Ps_3, then task *A* and *B* will be permission equivalent due to *propriety 2* but not equivalent as *propriety 1* states.

Uniqueness is a property which applies to all layers and tests if two or more elements are equivalent. In this case, the engineer should take in consideration an opportunity to minimize and reuse a single element. For example, the authentication step might be similar for different roles. In order to obtain the minimum, the equivalent entities should be minimized. In some cases the engineer could decide to minimize also permission equivalent elements, if appropriate. The set of permissions will be minimized without taking in consideration the mentioned properties.

The determination of reused elements is correlated to the determination of the minimum element. Reused elements might be elicited on R, Pf, Sc, Ps and P layers.

Completeness property applies to R, Pf, Sc and Ps layers. A layer is considered complete if all the elements on that layer are mapped to at least one element of the next layer and vice-versa. The engineer should consider also the completeness property in the context of the direct relations between roles and permissions and also profiles and permissions. If a permissions is not linked to a role, this means that the permission is useless or that the task that might be linked to that permission does not have a link to a role therefore it can not be operated. On the other hand, if there is a role that is not linked to any permission, it means that the role will not determine a single operation in the system.

Consistency applies both to role and profile layer. It tests if the set of permissions mapped to a profile will ensure the achievement of the goals linked to that profile. The same reasoning can be applied in roles' case. The goals are achieved through correspondent scenarios. Each scenario must be therefore tested. The engineer will use relevant test scenarios as the gathering of all scenarios is a large time-consuming activity. The permissions mapped to each profile should determine the proper execution of all the test scenarios related to the profile. In case all the scenarios are executed properly, it means that the relation of consistency is achieved. As multiple profiles determine a role, the consistency property can be extended at role level.

Constraints are an important topic in the design of an access control system [28]. Defining correct and comprehensive constraints rules is a very challenging task. A rule that is not well defined can determine a security breach for the system. The goal is to determine whether the set of constraints defined is complete [19], [20]. Access control constraints were classified as:

- Authentication constraints
- Contextual constraints
  - Temporal constraints
  - Location constraints
  - Relation constraints
  - Attribute constraints
  - State constraints
- Usage constraints
- Security constraints
  - Separation of duty constraints
  - Least privilege constraints
- Privacy constraints
  - Scope constraints
  - Recipient constraints
  - Consent constraints

The definition of constraints is determined by the identification of the conditions in which a subject is authorized to perform an operation on a specific object. The system requirements are a good source for constraints elicitation. In the description of the system the constraints are located after words like: when, if, while, before or after.

The coherence validates if the constraints given for a layer are propagated on the other layers of the model. For simplicity we will apply this property only on *R*, *Pf* and *P* layers. We suggested two approaches for coherence, given the mapping direction. *Property 8* uses the constraint catalog defined at permission level. For each of this constraint we should test if the roles/profiles which contains that permission are compliant with it, as

each of them has a certain set of permissions. For example, it is possible that a constrained linked to permission P_1 that is mapped to role R_1 through task Sc_1 could vitiate a constrained linked to P_2 which is mapped to role R_1 through task Sc_2.

In the case of *property 9*, the focus is on constraint catalog linked to roles and profiles. For each constraint that exists at role/profile level, it is tested if that is respected at the permission level, given the permissions linked to that role/profile. For example, a role can have a separation of duty constrained assigned which could contradict an affiliation constraint linked to one of the permissions that are mapped to the role.

Another important issue is this construction of the role-hierarchy. The role hierarchy is justified by the existence of non-disjunctive sets of permissions allocated to roles.

The premises for role hierarchy construction might be: the large number of permissions mapped to roles, dynamically or statically mutual exclusive permission sets or repetitive sets of permissions in different roles. The role hierarchy can be determined also based on the relation between roles and profiles. A profile that is contained by several roles might very well serve as a lower-layer role which contains the permissions linked to that profile. For simplicity, we suggest that all the profiles should be included in the role hierarchy as lower-level roles. Also, if there are different roles that are not mapped to similar profiles but they contain a relevant number of common permissions, it might be considered an opportunity to elicit a corresponding profile.

3.3 VMRE-RBAC Process

As we already stated, *RBAC* model has as a primary feature the many to many relation between roles and permissions. In *VMRE-RBAC* model, the *RBAC* model is extended by several middle layers. The association between layers is done by defining the following relations: roles with profiles (*RPf*), profiles with tasks (*PfSc*), tasks with steps (*ScPs*) and steps with permissions (*PsP*).

Each association has specific goals and features. The general process of role definition implies two major sub processes, which are role decomposition in permissions and testing. The decomposition process contains the following steps: identify the initial roles and role constrains, identify the main responsibilities and determine the candidate profiles and profile constrains, for each profile elicit the tasks and then define the steps, identify permissions and subsidiary constrains. The process is depicted in *Figure 7*.

The testing process involves the verification and validation for all the elements and relations defined in the decomposition stage. Verification means the process of evaluation applied to elements and relations in which the results are confronted with the requirements and conditions defined by the designer. In the verification stage will be used several properties that were already mentioned in the paper. On the other hand, validation assures that the process is defining the right system. This implies that the designers should confront the results with the beneficiaries. Also, validation implies that the model should be tested in relation with the goals and the scenarios already elicited. If there are any issues raised in the testing stage, then they will be addressed recurrently by re-iterating stages from the first main sub process.

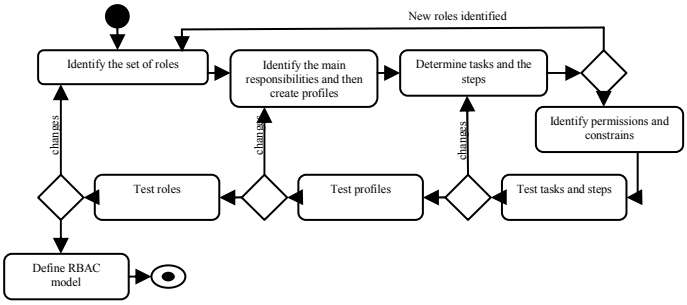

Figure 7. *VMRE-RBAC* process

The results of the role engineering process are:

- Roles catalog: starting from an initial set of candidates which is updated and modified in the future stages.
- Profiles catalog: includes the profiles elicited based on the major responsibilities. In the final stage, the profiles will be included in the roles catalog as hierarchically lower roles. Hence, the profile catalog will be a subset of the role catalog.
- Task catalog: list of all the tasks defined in the role-engineering process and the items mapped to them both in higher and lower layers.
- Constrains catalog: list of constrains, classified in permission and role constrains
- Goal catalog: list of goals determined with the beneficiaries in order to validate the profiles and roles.
- Scenario catalog: list of scenarios elicited with the beneficiaries in order to validate tasks.
- *RBAC* model: role hierarchy, permissions set and the constraints applied.

4 Conclusions

The idea of this research is driven by the complexity of the existing role-engineering processes and the lack of a standardized way for performing role engineering. The paper presents an approach for role engineering that aims to simplify the engineering process by linking it directly to the logic of business. The model includes both a decomposition process, from roles to permissions, and also introduces extra-layers in the *RBAC* standard model. As a possible application, the model should be a prerequisite for an information security access control system as the one presented in [4].

Even if the idea of adding new layers between the roles and permissions has been already discussed in a couple of scientific researches, the *VMRE-RBAC* process model brings a new perspective for role engineering. The *VMRE-RBAC* engineering process is also focused on the verification and validation of the results, an issue that should improve the quality of the whole role based access control future implementation.

There are several enhancements added by the current paper to the existing research in role engineering field. We can mention here the test-driven process model for role engineering, the relations defined between the intermediate layers added in the relation between roles and profiles and also several formal properties for results' validation and verification.

As a future research we aim to implement a software design tool for role engineering based on *VMER-RBAC* tool. The design tool should automatically generate both the system description and role architecture.

References:

[1] - Coyne, E., Davis, J., "Role Engineering for Enterprise Security Management", Artech House, 2008

[2] - Coyne, E., "Role Engineering", Proceedings of the 1st ACM Workshop on RBAC", Gaithersburg, 1996

[3] - Crook, R., Ince, D., Nuseibeh, B., "Towards an Analytical Role Modeling Framework for Security Requirements", Proc. of the 8th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'02), Essen, Germany, 2002

[4] - Constantinescu, R., Corlan, L., "An Adaptive Authorization Model Based on RBAC", 1st International Conference on Security for Information Technology and Communication, Bucharest, 27-28 November 2008

[5] - Epstein, P., "Engineering of Role/Permission Assignments", Ph.D. Dissertation, George Mason University, 2002

[6] - Ferraiolo, D., Kuhn, D., Chandramouli, R., "Role-Based Access Control - Second Edition", Artech House Publishers, 2007

[7] - Ferraiolo, D., Cugini, J., Kuhn, R., "Role Based Access Control: Features and Motivations", Proceedings Annual Computer Security Applications Conference, IEEE Computer Society Press, 1995.

[8]- Goncalves, G., Poniszewska-Maranda, A., "Role Engineering: From design to evolution of security schemes", Journal of Systems and Software, 2007

[9] - Kuhn, R., "Mutual Exclusion of Roles as a Means of Implementing Separation of Duty in Role-Based Access Control Systems", Second ACM Workshop on Role-Based Access Control, 1997

[10] - Neumann, G., Strembeck, M., "A Scenario-driven Role Engineering Process for Functional RBAC Roles", Proceedings of the 7th ACM Symposium on Access Control Models and Technologies, 2002

[11] - Sandhu, R., Ferraiolo, D.F., Kuhn, D., "The NIST Model for Role Based Access Control: Towards a Unified Standard", Proceedings 5th ACM Workshop on Role Based Access Control, 2000

[12] D.F. Ferraiolo and D.R. Kuhn (1992) "Role Based Access Control" 15th National Computer Security Conference, Oct, 1992

[13] D.F. Ferraiolo, D.R. Kuhn, R. Chandramouli, "Role Based Access Control" (book), Artech House, 2003, 2nd Edition, 2007

[14] D.R. Kuhn, "Mutual Exclusion of Roles as a Means of Implementing Separation of Duty in Role-Based Access Control Systems" Second ACM Workshop on Role-Based Access Control, 1997

[15] D.F. Ferraiolo, J. Barkley, D.R. Kuhn, "A Role Based Access Control Model and Reference Implementation within a Corporate Intranet", ACM Transactions on Information Systems Security, Volume 1, Number 2, February 1999.

[16] R.Chandramouli, "Specification and Validation of Enterprise Access Control Data for Conformance to Model and Policy Constraints", 7th World Multi-conference on Systemics, Cybernetics and Informatics, 2003

[17] R. Constantinescu, A. Barbulescu, "Systems Security through Capability Models", "Competitiviness and European Integration" International Conference, Cluj, Romania, Oct, 2007

[18] R. Constantinescu, F. Nastase, "Process Models for Security Architectures", Informatics in Economy Journal, no. 4, 2006

[19] He, Q., "Requirements-based Access Control Analysis and Policy Specification", Ph.D. Thesis, North Carolina State University, 2005

[20] He, Q., Anton, A, "A framework for Modeling Privacy Requirements in Role Engineering",

Proceedings of the 9th International Workshop on Requirements Engineering, Austria, 2003
[21] Jaideep Vaidya, Vijayalakshmi Atluri, Qi Guo, "The role mining problem: finding a minimal descriptive set of roles", SACMAT, 2007
[22] Ian Molloy, Hong Chen, Tiancheng Li, Qihua Wang, Ninghui Li, Elisa Bertino, Seraphin Calo, Jorge Lobo, "Mining roles with semantic meanings", SACMAT, 2008
[23] Schaad, A., Moffet, J., Jacob, J., "The Role-Based Access Control System of a European Bank: A Case Study and Discussion", SACMAT, Chantilly 2001
[24] Jürgen Schlegelmilch, Ulrike Steffens, "Role mining with ORCA", SACMAT, 2005
[25] Vijay Atluri, "Panel on role engineering", SACMAT, 2008
[26] Zhang, D., Ramamohanarao, Ebringer, T., "Role Engineering using Graph Optimization", SACMAT, 2007
[27] Andreas Mattas, Ioannis Mavridis, Christos Ilioudis, Ioannis Pagkalos DARBAC: Dynamically Administering Role Based Access Control, WSEAS Transactions on Information Science & Applications, Issue 10, Volume 3, October 2006
[28] Min Wu, Jiaxun Chen, Yongsheng Ding, Study on RBAC Model for Web Services and its Application, Proceedings of the 5th WSEAS International Conference on Telecomunication and Informatics, Istambul, Turkey, 2006
[29] Jonathan Keirre Adams, A Service-Centric Approach to a Parametrized RBAC Service, Proceedings of the 5th WSEAS International Conference on Applied Computer Science, Hang Zhou, 2006
[28] Yi Guo, Myeonggil Choi, Sangmoon Shyn, Guibam Bae, Yongsun Choi, An Approach for Implementation of RBAC Models with Context Constrained to Business Process Systems, WSEAS Transactions on Systems, Issue 6, Volume 5, June 2006
[29] Radu Constantinescu, Andrei Toma, Iuliana Scorta, Floarea Nastase, Razvan Zota, V-Model Approach for Role Engineering, 13th WSEAS International Conference on Computers, Rhodos, 2009