V-Model Role Engineering

Radu CONSTANTINESCU
Economic Informatics Department,
Academy of Economic Studies, Bucharest, Romania
radu.constantinescu@ie.ase.ro

The paper focuses on role engineering which is an important topic in the development of access control system, particularly when considering Role Based Access Control – RBAC models. Despite the wide use of RBAC in various applications, the role engineering process is not a standardized approach. The paper aims to define a methodology and a process model for role engineering.

Keywords: *Information security, access control systems, role based access control systems – RBAC, engineering methodologies, security policies, access control models.*

1 Introduction

Starting with the opening brought by the distributed computing environments during the last decade, the security issues over passed the limited area of mainframe organizations and became a widespread concern. Even more, the risks related to business area became more sharp and complex. The security issues were institutionalized and standardized. To obtain better results, the usage of security mechanisms must be done through a well defined organization process.

The core of a security system is the way in which the access control is addressed. The most suitable access control model for mature organizations, with clear defined responsibilities, is role based access control – *RBAC* [6], [7], [8], and [14]. Ideally, the implementation of an access control model is done concurrently with the informational system implementation. This case is less probable, because most organizations have already automated their business processes without taking in consideration access control as a mandatory and also extended approach. A rigorous analysis has to be done related to permissions and roles assignment in order to implement an access control system based on roles [1]. The model efficiency depends on the ability of the designers to identify the correct assignments. This desideratum can be obtained through the usage of a flexible methodology which should sustain a coherent process of role engineering.

2. Problem Formulation

The objective of this research is to identify a model aimed to address the security issues in an organization. In order to achieve this goal, this paper aims to define a 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 help the role hierarchy design and the identification of constraints. The results of the role engineering process will be used to design an authorization module for the information system implemented in a company. Possible usage is presented by the author in [4].

Features of role engineering have been discussed starting with Coyne's paper in which are defined the main activities that should be addressed in this concern. Those activities are: role definition, roles hierarchy definition, constraint definition and the mapping between roles and permissions [2].

Neumann and Strembeck have documented an approach for role engineering based on scenarios [13]. In this model, each activity is described based on a set of scenarios and each scenario is then decomposed in a suite of steps. Each step is then mapped to several permissions. The approach has the disadvantage that it requires a great effort in order to determine all 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 was detailed in both top-down and bottom-up manner. The model takes the presumption that the roles and permissions are already determined, so it doesn't specify 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 functions. 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 [9].

3. Problem Solution

3.1. General concepts

The paper proposes an extended *RBAC* model, in order to enhance the role engineering process. The paper also proposes a process aimed to identify the relations between roles and permissions. The proposed model, *VMRE-RBAC (V-Model Role Engineering RBAC)*, facilitates the decomposition of roles in permissions and then the testing of the results. The model is iterative. For every testing stage a new optimization of results is made. Every decomposition stage will have a correspondent testing stage. The testing stage is concerned with validation and verification of the results. The engineering process is enhanced with a set of properties proposed in order to achieve simplicity, consistency and coherence in the model.

The paper is based on *RBAC* model. The components of the initial *RBAC* model are: users, roles, permissions, and also the relations between those elements. The initial *RBAC* model is enhanced by adding role hierarchies and constraints. The integrated *RBAC96* model is defined by cumulating those issues. One of the advantages of *RBAC96* model is that it implements the principles of least privilege, separation of duty and administration and also data-abstraction [12], [14].

The role engineering is a compulsory stage for a *RBAC* implementation. *VMRE-RBAC* extends the standard *RBAC* model by adding some 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 which determines responsibilities. Each responsibility is carried out through a specific profile. I suggest that profiles should be low 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 issues or functionality issues.

After a first decomposition process, the results will be tested incrementally. Testing means both verification and validation for the entities on each layer. In order to ease this process, I propose a set of nine properties which include: 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 presented both in a static and dynamic view. 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.

As a start, I use the standardized *RBAC96* relation between permissions and roles in order to build roles as a superset of permissions. I suggest decomposing the mapping of roles and permissions in several mappings between several middle layers. I propose the usage of three middle layers between roles and permissions, which are: profiles, tasks and steps. Each of these layers will be treated independently. 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 in a bottom-up approach. The testing process is based on a couple of requirements formalized in nine properties. These features will be described in detail.

The roles are placed in the upper layer of *VMRE-RBAC* model. A role is determined initially based on the work-profile of the company in which the model is applied. A role can be associated with several responsibilities. Each major responsibility is 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. If there are steps that don't require permissions for the moment, they will be labeled as unconstrained. It is possible that an unconstrained task becomes constrained due to policy changes.

Before I continue to describe the model in depth, I consider that it is important to detail some concepts used in this methodology: goals and scenarios. The goals and the scenarios have complementary particularities [11]. The goals are abstract and declarative while scenarios are concrete, narrative and procedural issues. Scenarios describe real situations by examples. Eliciting the goals and scenarios will enhance the testing stage of the model. Having the goals as a central issue will facilitate de design phase because the goals are familiar and well understood by all the persons that are involved in the design process.

The goals can be hierarchically structured and are linked to different implementation scenarios. Hence, based on goals, the design team will be able to determine use-cases for the system. I consider as a very important issue in the first phase to identify the goals related to each profile. Given the goals, the design team will be able to determine and validate the tasks related to each profile.

3.2. RBAC Extended Model

The standard items of *RBAC* model - users, roles and permissions - are also used in the proposed extension and in the corresponding methodology. *RBAC* standard defines the mapping between roles and permissions, as depicted in Figure 1, but it doesn't detail how this mapping is achieved. I aim to propose a role-engineering solution in this concern.

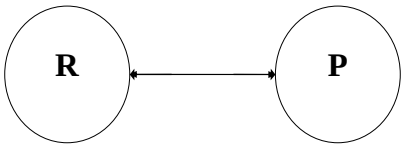

Fig. 1. The relation between roles and permissions in NIST's RBAC standard

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

- R – roles set
- P – permissions set
- $PA \subseteq P \times R$ many to many relation between permissions and roles

The roles in an organization can be classified as functional or administrative. In the first instance I suggest dealing with the first category. Usually, the system's users are already defined in an organization. The tasks of identifying roles and permissions should be shared by the system's administrator, developing team and also the beneficiaries. The general model for *VMRE-RBAC* is presented in Figure 2.

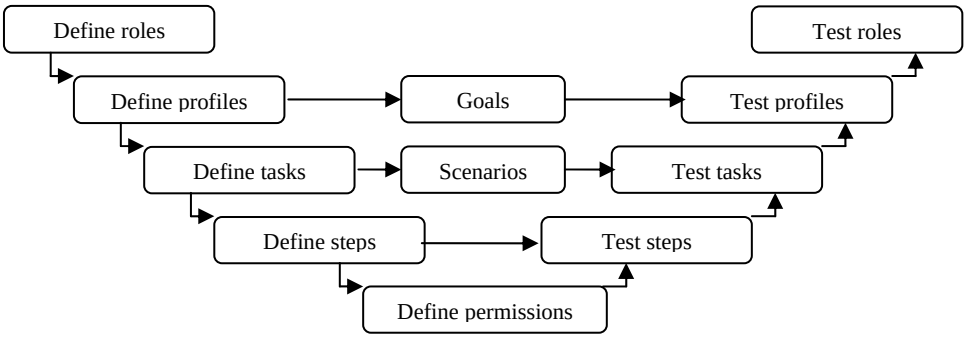

Fig. 2. The general model of decomposition and testing VMRE-RBAC

Starting from the classic *RBAC96* model, I added three middle-layers which are: profiles, tasks and steps. Each decomposition stage has an equivalent testing stage. In order to formalize the model, I 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 elaborate a methodology which should optimize the process of mapping between roles and permissions. I 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 I 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 I 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, 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.

The identification of constraints is one of the most challenging tasks in the design. It is required that all the constraints are identified and tested. I propose the coherence property in this matter.

I present a formal description of the 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

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; notation: $a_i \approx a_j$

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 $(\forall) s \in S$, $(\exists)\ pf \in Pf \wedge sc_{1, \ldots,} sc_n \in Sc(pf)$ that:

$$\bigcup_{k=1}^{n} sc_k \approx_p s$$

Property 8: The coherence of constraints at permissions layer for the upper layers:
For $(\forall) c \in C_p \wedge (\forall) p \in P(c)$ 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 equivalence property is applied to *R*, *Pf*, *Sc* and *Ps* layers. The equivalence between permissions is not included here because permissions represent the lowest layer of the hierarchy and hence the first property cannot be verified. Two sets on the same layer are equivalent if they have the same set of elements mapped on the next layer. In order to determine the equivalent permissions, one can sort the permissions catalog, given the fact that any permission is a tuple *(operation, object)*. The operations can be generally defined as elements of *{C, R, U, D, E}* set where *C* is create, *R* is read, *U* is update, *D* is delete and *E* is execute.
The permissions equivalence has sense in the context of tasks, profiles and roles. The permissions-equivalent elements are not necessary equivalent as property 1 states. Otherwise, elements equivalent due to property 1 are also permission equivalent.
Uniqueness is a property that applies to all layers and verifies if two or more elements are equivalent. In this case, the designer should minimize and reuse a single element. To determine the minimum, one should reduce the equivalent elements. If I discuss the permission equivalent issue, the minimization decision will be made by the role engi-

neering manager. The identification of reused elements is correlated with the determination of the minimum elements. If an element is used in more than one mapping with the following layer, the element is reused.

Completeness is applied for *R, Pf, Sc* and *Ps* layers. A layer is considered complete if all the elements in that layer are associated with at least one element on the next layer and vice-versa. The completeness will be verified also in the context of the relations between roles and permissions and between profiles and permissions. This is important because if a permission is not mapped to a role then this means that the permission is useless and needs to be cleared or that the task mapped to that permission is not linked to a role which means that is a problem in the relation between profiles and roles. Otherwise, if a role is not linked to permission, it means that the role is useless and will not carry out any activity in the system.

Consistency applies both to the profiles and roles layer. Consistency verifies if the permission set assigned to a profile/role will be enough in order to achieve the goals assigned to that profile/ role. The goals are realized by successive tasks described in scenarios. In order to verify the consistency, one could test each scenario from the scenario's list. Hence, for each profile *Pf* there will be mapped a set of test scenarios. If the permissions assigned to that profile allow completing the test scenarios then the profile will be consistent. A set of profiles determines a role, so the property can be applied then to the role layer.

Coherence tests the constraints assigned to a layer in the context of upper or lower layers. For simplicity reasons, I will apply this property only to the following layers: *R*, *Pf* and *P*. I suggested two approaches for coherence, given the mapping direction. Hence, for property 8, I start from constraints catalog defined at permission's layer. For each constraint, the designer should test if the profiles/roles that are linked to the constraint are compliant with it. For example, a permission P_1 associated with the role R_1 through the task Sc_1 might damage a constraint for the permission P_2 assigned also to R_1 through another task, say Sc_2.

Property 9 tests that each constrain defined at profile/role layer is propagated to the permission's layer. For example, a profile/role might enforce a separation of duty constraint that contradicts with an affiliation constrain defined at the permission's layer.

In Table 1, I present a summary of the correlation between the nine properties I have defined and the model's layers.

Table 1. The correlation between properties and layers in VMRE-RBAC model

	Role	**Profile**	**Task**	**Step**	**Permission**
Equivalence (P_1)	×	×	×	×	
Permission equivalence (P_2)	×	×	×		
Uniqueness (P_3)	×	×	×	×	×
Minimum (P_4)	×	×	×	×	×
Reused element (P_5)	×	×	×	×	×
Completeness (P_6)	×	×	×	×	×
Consistency (P_7)	×	×	×		
Coherence (P_8)	×	×			
Coherence (P_9)					×

As I already stated, *RBAC* model has as a primary feature the many to many relation between roles and permissions. Through *VMRE-RBAC*, the *RBAC* model is extended with 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 responsibil-

ities 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 3.

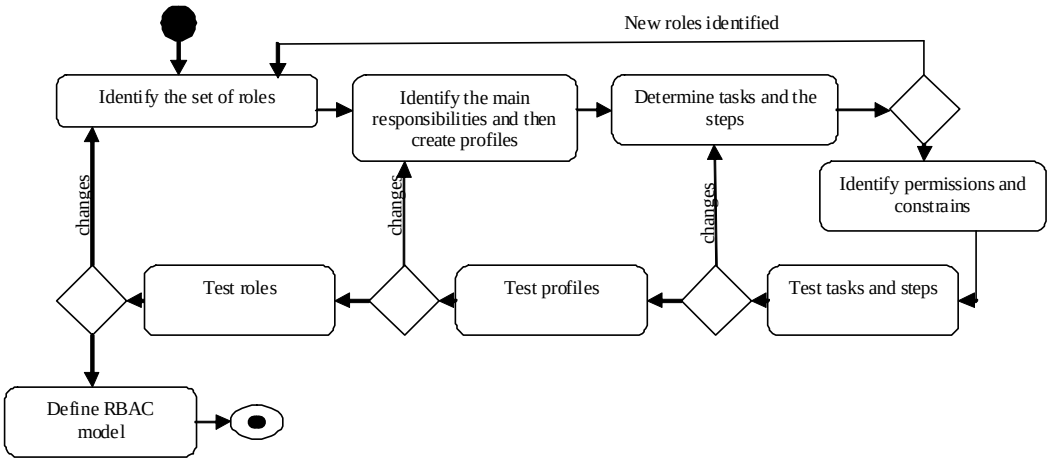

Fig. 3. The general model for *VMRE-RBAC* process

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 for the system. In the verification stage will be used several properties that are already presented in the paper, e.g. $P_1 - P_6$, P_8, P_9. 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 elicited. Property 7 helps in this matter. If there are issues raised in the testing stage, then they will be addressed recurrently by running again items from the first main sub process.

In Figure 4, I present a possible result of *VMRE-RBAC* process. For simplicity goals, I started with a single role R_1 which has two profiles Pf_1 and Pf_2. For each profile I 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 permission.

The following items are developed in the role engineering process:

- Roles catalog: starting from an initial set of candidates which is updated and modified in the future stages.

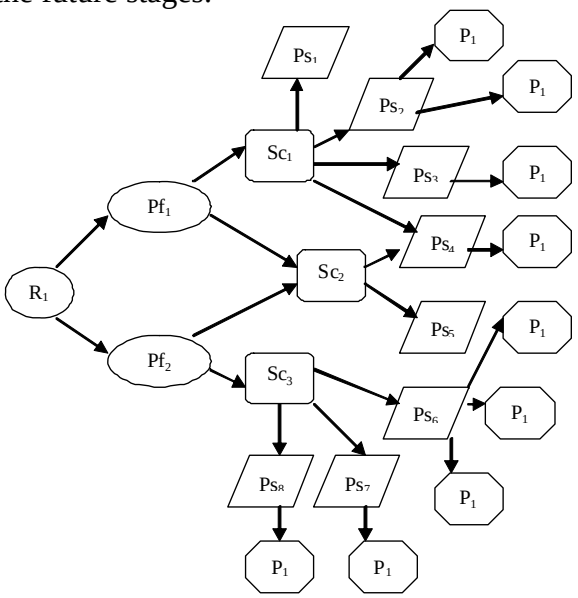

Fig. 4. Decomposition of roles in permissions

- 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, clas-

sified 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.

3.3. Constraints design

Defining right and comprehensive constrains for access control represents an important stage in role-engineering. Any omission or wrong rule can determine a security breach for the system. Determining constrains might be a difficult task as the finishing line is hard to acknowledge. Generally speaking, there are five major constraint classes: authentication constraints, contextual constraints, usage constraints, privacy constraints and security constraints.

Defining constraints involves the proper identification of the conditions in which a subject is authorized to take an action on an object. In order to identify constraints, one could use the system requirements defined in the 1st stage of the development life cycle for the informational system. A constraint can be found after words like: „*when*", „*if*", „*while*", „*before*" or „*after*". The constraints can be also verified on agreed work scenarios.

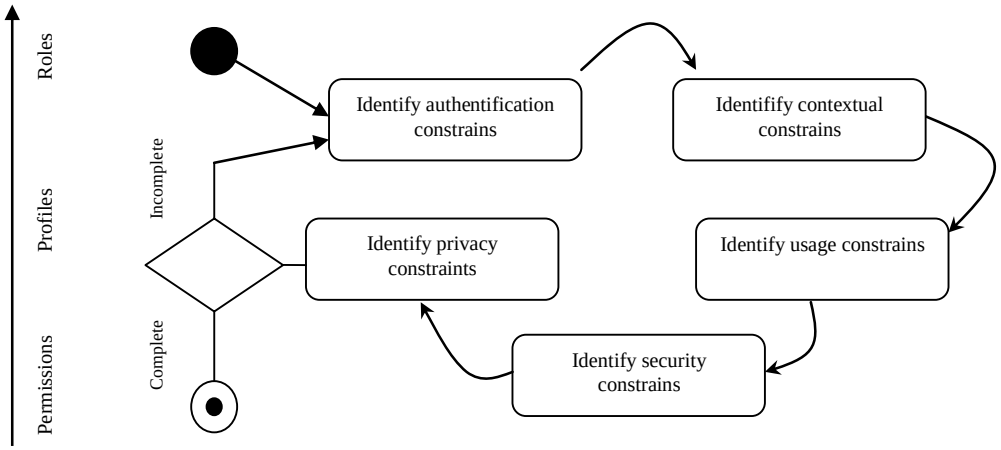

Fig. 5. Constraint determination process in VMRE-RBAC model

Figure 5 depicts the process for constraints engineering. The process involves the analysis on all the constraints types in an iterative manner, starting from role constraints, continuing with profile constraints and finishing with permission constraints.

4. Conclusions

The necessity of this paper is given by the complexity of the engineering process and the lack of a standardized way for performing role engineering. The paper grounds a consistent approach for role engineering. The methodology includes a decomposition process, from roles to permissions, and also introduces extra-layers in the *RBAC* standard model.

I identified a couple of future research areas to enhance *VMRE-RBAC* model. A first area of research is to introduce obligations in the model. An obligation is a task that has to be done when access is granted. *VMRE-RBAC* doesn't include also the concept of delegation which is used in the business activity. This issue can be another goal for the future research. An interesting research topic can be the development of a software product for role engineering based on *VMRE-RBAC* model or the development of specific dedicated tools. The reverse approach, from permissions to roles can be implemented also starting from *VMRE-RBAC* model.

References

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

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

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

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

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

[6] D. Ferraiolo, D. Kuhn and R. Chandramouli, *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] S. Gavrila and J. Barkley, "Formal Specification for Role Based Access Control User/Role and Role/Role Relationship Management," in *Third ACM Workshop on Role-Based Access Control,* 1998

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

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

[11] Q. He and A. Anton, "A framework for Modeling Privacy Requirements in Role Engineering," in *Proceedings of the 9th International Workshop on Requirements Engineering*, Austria, 2003

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

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

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

Radu CONSTANTINESCU has a background in computer science. He has graduated the Faculty of Cybernetics of the Academy of Economic Studies in Bucharest. He has finished his doctoral research at the Academy of Economic Studies, with the topic on Information Security. His fields of interest include computer security related issues.