governance requirements. In this paper, we propose an integration of our responsibility model with RBAC [2] to minimize the three weaknesses identified in section 4. RBAC is an access control model that simplifies structuring the access right for a domain. Policies are elaborated using a policy language such as XACML (Extensible Access Control Markup Language) [36]. The basic RBAC model can be extended by modeling using OWL (Web Ontology Language) [35] that enables going beyond the basic semantics of RDF schema to perform reasoning tasks necessary to enforce specific constraints such as the separation of duty (SoD) or role hierarchies. We also use OWL for the representation of our responsibility-RBAC model.

This paper is organized as follows. Section 2 introduces the RBAC model and its user to role and permission to role assignment process. Section 3 presents our responsibility model, section 4 integrates both models into a single one, section 5 compares the representation of our model with two representative existing works and the last section concludes.

2 Background: RBAC

2.1 The RBAC Model

The concept of role has been introduced in software engineering about 35 years ago and has followed the development of traditional access control techniques such as the Mandatory Access Control or Discretionary Access Control. Role Based Access Control (RBAC-Fig 1.) has been introduced in the NIST standard for role-based access control [2] and embodies the entire previously developed notions in a single model which is now the reference access control mechanism for most software applications. The publication of this standard has been followed by many related papers which adapt the model for specific fields (e.g. eCommerce, [3]), to propose alternative solutions according to other constraints (Context Aware RBAC, [4]), or for proposing solutions for managing some of its aspects (e.g. ARBAC [5], URA97 [6] or PRA97 [7].

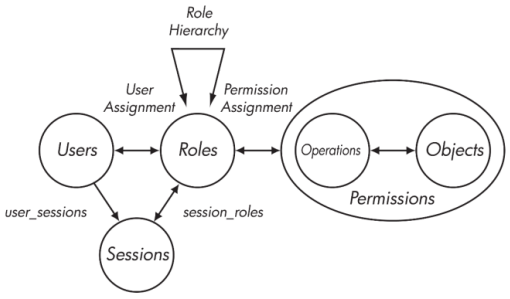

Fig. 1. RBAC model

RBAC is a high level model with the objective to simplify the management of granting permissions to users. This is especially necessary in multinational companies where the amount of employees often count in thousands. It provides access decisions

based on two associations – the association of users to roles based on the function that users assume and based on their responsibilities, and the association of permissions to roles describing that a role has the permission to perform specific operations on objects. This means that it is easy to change the assignment of people to roles without changing permissions.

2.2 User-Role and Permission-Role Assignment

The process of assigns users to roles and permissions to roles is normally a managerial function performed by the business manager or the process owner to decide which employee needs to access what application to achieve her job. The actual implementation of this may be delegated by the application business owner to a security administrator. URA97 [6] and PRA97 [7] are both part of the ARBAC97 [5] model (Administrative RBAC) that permits the assignment of the users to roles and permission to role by means of administrative roles and permissions. Both URA97 and PRA97 are defined in the context of RBAC96 model family but are applicable for most of the RBAC model. Their philosophy is the creation of administrative roles managed by security officers. These administrative roles are granted administrative permissions to assign or remove user to/from roles. In the same way that RBAC96 defines role hierarchies, ARBAC97 defines administrative role hierarchy so that a senior security officer inherits permissions from a junior security officer below him in the role hierarchy. For example, if the junior has assigned an employee to a inappropriate business roles, the senior security officer can remove that employee from the role or change the permissions associated with it. URA97 gives a detailed explanation of the administration of the assignment process.

The simplest way for a manager to assign permission to a user is to assign that user in to a role that encompasses specific tasks to perform and has the required permissions to perform the tasks. By doing so, the manager implicitly obliges the user to accept the responsibility to perform the tasks but does not actually know whether the employee has agreed to this. Not taking into account the employee's commitment is an authoritarian way of managing staff and may result in company goals not being achieved due to unwillingness of employees to perform assigned tasks (see section 3.3). Although this may seems unavoidable, especially in large companies, it could easily be improved by incorporating acceptance of responsibility by a user within the role assignment process, as shown in this paper.

3 Responsibility model

In this section, we present our generic responsibility model as a proposed enhancement to RBAC. The complete responsibility model (figure 2) is presented in detail in [1]. The analysis of the concept of responsibility [1,10] highlights that there is a plethora of definitions for it. A commonly accepted definition of responsibility encompasses the idea of *having the obligation to ensure that something happens*. The responsibility model is built around three sets of concepts. The first set concerns

accountability of the employee regarding the *obligation* targeted by the responsibility, the second set concerns the *rights* required to fulfilled the obligations and the third set concerns the *commitment* to be pledged by that employee.

3.1 Concept of obligation/accountability

We define an obligation as *a duty to perform an action*. Dobson et al. [11] classifies it following two perspectives: functional obligation as what a role must do with respect to a state of affairs (e.g. execute an activity) and a structural (managerial) obligation as what a role must do in order to fulfill a responsibility such as directing, supervising and monitoring.

Accountability and answerability are similar concepts that are composed of one or more obligation(s) to report the achievement, maintenance or avoidance of some given state [12] to an authority. For our model, we prefer the definition of answerability provided by Cholvy as *an obligation or a moral duty to report or explain the action or someone else's action to a given authority* [10] and the definition of accountability from Laudon and Laudon [15] as *a feature of systems and social institutions: It means that mechanisms are in place to determine who took responsibility of actions*. Accountability thus includes answerability as well as the possibility of sanctions for non-fulfillment of obligations [13]. Stahl [14] argues that accountability describes the structures, required to facilitate responsibility and that responsibility is the ascription of an object to a subject rendering the subject answerable for the object. Stahl also focuses on the sanction as being of central importance for responsibility. He nuances the sanction as positive or negative.

3.2 Concept of right

We define the right as *what is due to a employee*. This concept is common but is not systematically embedded in the IT frameworks [16, 34]. It encompasses facilities required by an employee to fulfill his accountabilities. These facilities could include, amongst others, capabilities, authorities or the right to delegate.

Capability describes the possession of requisite qualities, skills or resources to perform an action [12,16,17] and relate to a user. This may be implied through access rights, authorizations or permissions [18,19].

Authority describes the power or right to give orders or makes decisions. This concept is introduced in CIMOSA [16] as *the "power" to command and control other employees and to assign responsibilities*.

Delegation is a right to transfer some part of the responsibility to another employee that pledges commitment for it (see section 3.3). This transfer may concern the transfer of right or of accountability or both. The delegation of an obligation may or may not be accompanied by the delegation of right for the delegatee to further delegate the same obligation [12].

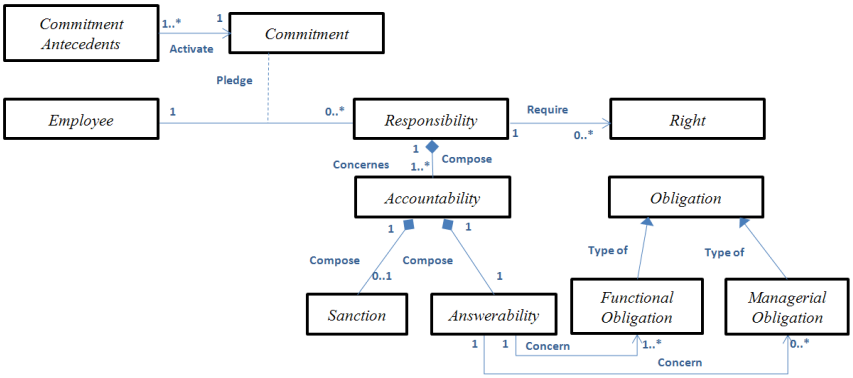

Fig. 2. UML responsibility model

3.3 Assignment/delegation process

We define assignment as *the action of linking an employee to a responsibility* and delegation is the transfer of an employee's responsibility assignment to another employee.

The commitment by an employee related to that assignment or delegation represents his moral obligation to fulfill the action and the assurance that he performs it with respect of an ethical code. The commitment remains a virtual concept, difficult to define as well as to integrate in a strictly formalized framework. In [20], Meyer and Allen acknowledge that *commitment should be conceptualized as a psychological state concerned with how people feel about their organizational engagements*. To bypass the integration difficulty, we propose to extend the model with the components that can be used to enforce the commitment.

Commitment's antecedent in the literature relate to pragmatic variables [21] that may influence a person's commitment e.g. the age of the employee and the time he spent in the organization [23,24,25], the perception of job security [26], management culture and style [27], the employee's investments in time, money and effort [28] or how his experience is valued by the company [22]. A scientific survey of commitment also highlights that *Commitment outcomes* may really influence the quality and efficiency of the action achieved. Pfeffer in [29] explains that *Employee commitment is argued to be critical to contemporary organizational success*. The following list summarizes commitment outcomes:

- Employee performance [30] – committed employees performed better when committed to both their organization and their profession.
- Retention of the employee – many studies demonstrate the link between the commitment and the employee's turnover [28,30,31].
- Citizen behavior[1] – research over these outcomes remain however inconclusive [32].

[1] According to [7] definition, it represents the *individual behavior that is discretionary, not directly or explicitly recognized by the formal reward system, and in the aggregate promotes the efficient and effective functioning of the organization*

Based upon the commitment outcomes and antecedent definition, we may assume that commitment for responsibility of an action means will increase trust in the achievement of an obligation or in the accountability attached to the responsibility, as well as increase efficiency (and consequently capabilities) for this employee to perform the action.

4 Mapping RBAC with the responsibility model

In this section we propose a novel model called r*esponsibility-RBAC* (figure 3). As seen in section 2, the three main elements of RBAC are User, Role and Permission (dashed boxes in figure 3) and the two main functions are User-role assignment (URA) and Permission-role assignment (PRA) indicated by dashed arrows in figure 3. Although RBAC presents many advantages such as facilities to grant or to remove permissions to a large number of employees, it also presents weaknesses regarding the following business IT alignment constraints:

1. Number of roles: the inflexibility of the model may result in more roles than users *if all permission assignments are very distinct* [33] or *in order to accommodate a user specific constraint* [38]. Moreover, in small organisation, the concept of role does not always map onto access rights.
2. Employee's commitment: RBAC does not offer cater for management of the employee's commitment regarding the tasks they are responsible for.
3. The representation of RBAC in OWL results in the following problems: inconsistencies in ontology [8], difficulty of detection of constraint violations using DL-reasoner [8], as well as the need to deploy complex architectures [9]

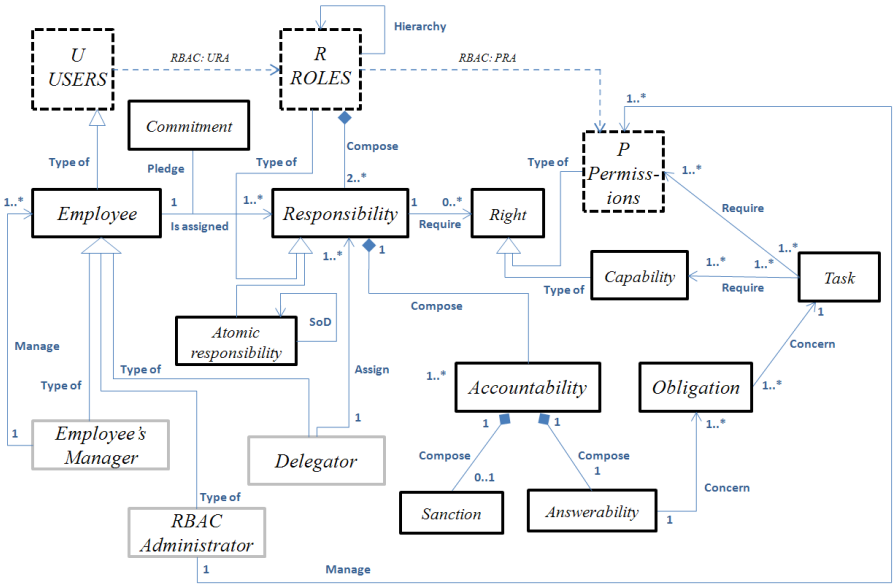

Fig. 3. UML responsibility-RBAC model

The three next sub-sections analyze the contribution of the responsibility-RBAC model to improve RBAC above listed weaknesses

4.1 Number of roles optimization

RBAC requires an employee (type of business USER) who needs a permission to achieve a task to be assigned to a role. Thus, if an employee needs to have permissions to perform a task which is independent of existing roles, then a specific role must be created or the task must be associated with an existing role, even if the latter is not directly related to the task. This is mainly due to the lack of granularity of RBAC that may lead to situations where the number of roles is larger than the number of users, or where roles do not reflect real job functions because they are assigned permissions for a too heterogeneous set of tasks.

Our proposal to solve those problems is to introduce the concept of responsibility as an intermediary concept between the user and the role in RBAC (figure 3). We consider that the role is a predefined set of responsibilities, that employees can be assigned specific responsibilities, independent of roles and that permissions are associated with the responsibilities for which they are required. This model allows us to refine the URA concept of RBAC: users are assigned to responsibilities as far as they commit to them. The responsibility is an abstract concept that could be either a concrete atomic responsibility or a concrete role (group of responsibilities). The PRA concept of RBAC is refined through associating permissions both to atomic responsibilities and to roles.

The tuple of concepts [user-role-responsibility] facilitates defining two types of user-role assignments and one type of responsibility-role assignment:

1. Direct role assignment: an employee is assigned to a role and gets the corresponding responsibilities and permissions. In that case, the role is often the main function of the employee and corresponds to his main function in the company.
2. Direct atomic responsibility assignment: An employee is assigned an atomic responsibility without any associated role and the employee then gets the corresponding permissions.
3. Indirect role assignment: an employee is assigned, by direct atomic responsibility assignment all the responsibilities that compose a predefined role, so he is implicitly assigned to the role and he gets the permissions corresponding to those responsibilities. This case reflects the situation where an employee is assigned to more and more responsibilities which happen to the responsibilities predefined in a role. Whereas from an IT point of view, the set of these responsibilities correspond to a role, the employee does not have the title corresponding to the role, from an organizational viewpoint.

The direct role assignment corresponds to the user-role assignment mechanism proposed in RBAC. The advantage of this solution a large number of permissions for users are granted or managed. For example, suppose that the role of project manager is composed of three responsibilities:

- management of the team,
- management of the project outcomes,

- management of the budget.

The employee who is assigned to that role receives all the permissions necessary for the management of the budget, the management of the team, and the management of the outcomes. If a new responsibility is added to the role, the employee is automatically assigned to it.

The direct atomic responsibility assignment: the user is assigned to an atomic responsibility and receives the permissions necessary to perform the tasks linked to that responsibility. E.g. an employee who is not project manager but who however performs the management of the outcomes is assigned responsibility for that task and receives the permissions necessary to perform it. This situation could occur for example in the case where the project manager assigns the management of the outcomes to a subaltern. In RBAC, representing this situation requires the definition of an explicit role for the management of outcomes. If the equivalent situation occurs for the budget management and for the team manager, the number of roles could considerably increase and the advantage of using roles for granting or removing permission to a user will diminish.

The indirect role assignment corresponds to a user-role assignment that exists when an employee is assigned to all responsibilities that compose the role. Whereas RBAC only offers the possibility to assign users to roles, the responsibility-RBAC model permits additionally to refine the granting of permissions to atomic responsibilities and to automatically assign an employee to a role when that employee performs all the atomic responsibilities that compose that role. E.g. an employee who is separately assigned responsibility for the budget management, then for the outcomes management, and afterward for the team management is, as result, implicitly assigned to the project manager role. In that perspective, the employee is assigned to a role from an IT point of view but that employee to role assignment is not recognized by the company. Detecting and officially acknowledging that employee to role association (and consequently make it a direct role assignment) is an improvement of the business IT alignment. If a new responsibility is added to the role, then it will be automatically assigned to the employee in the case of direct role assignment but not in the case of indirect role assignment.

There are three types of responsibility/role de-assignment: direct removal of role, direct removal of responsibility and indirect removal of role. In that last case, when all the responsibilities of a role are removed from an employee, this role is from an IT point of view no longer assigned to the employee whereas from an organizational point of view, this employee is still assigned to the role.

The delegation of responsibility is not the same as the removal of responsibility. In the case of delegation, the employee keeps the obligation of supervision [12].

4.2 Employees' commitment to the responsibility

In order to explain how the commitment may be included the user to role/responsibility assignment process, a conceptual assignment process is proposed as illustrated in figure 4. When being assigned to a role or to an atomic responsibility, the employee needs to explicitly commit to the achievement of the task(s) related to the role or to those related to the atomic responsibility. This concept of commitment

does not exist in RBAC as it considers the assignment of an employee to a role as an action performed solely by the employee's manager. Based on our review of the significance of the commitment in section 3.3 and according to the responsibility model, we propose to integrate the commitment in the employee to responsibility assignment process. The stakeholders involved in that process are indicated in figure 3 as grey boxes. The *employee* is assigned responsibility to achieve a task by the *delegator* who remains responsible and accountable for the management of the task, as in CobiT [34]. The *employee's manager* is responsible for the management of the employee. Sometimes the task manager and the employee's manager is the same person. The *RBAC administrator* is the security officer who manages the access rights.

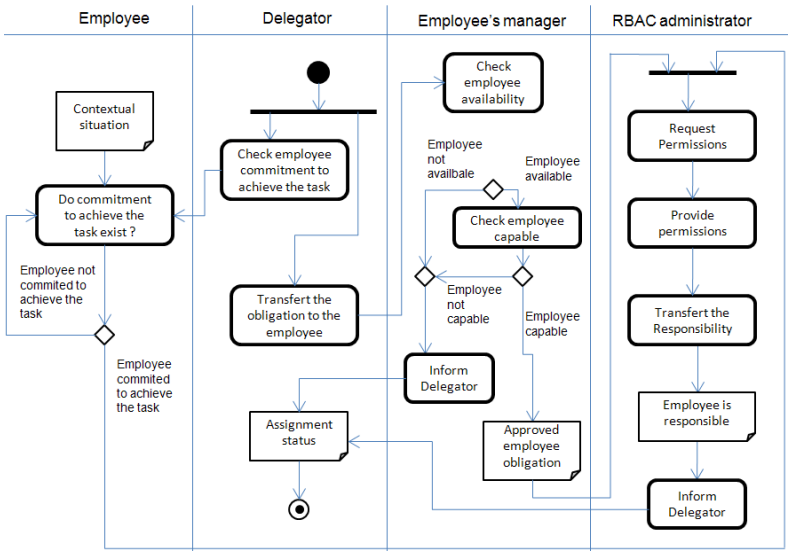

Fig. 4. Responsibility assignment process represented as a UML Activity diagram

An employee to responsibility assignment process may start with a request from a delegator to transfer the obligation related to a task to an employee (figure 4). This transfer is possible if the employee's manager accepts the assignment of the responsibility to the employee and if that employee explicitly commits to fulfill the task. The first condition corresponds to a double control which is: the employee availability and the employee capability. In some cases, the employee is also the manager and consequently, decides whether to accept or reject new responsibilities according to availabilities. The second condition corresponds to the commitment pledged by the employee according to his perception of the environment, guarantees received, interest in the task, etc. (see commitment antecedent in section 3.3).

Once the delegator receives the agreement from the employee's manager and the commitment from the employee, the delegator requests the RBAC administrator to provide the permissions needed to achieve the task. As soon as the permissions are granted, the employee is assigned the responsibility (figure 4).

4.3 Responsibility-RBAC representation with OWL

The Web Ontology Language OWL is a semantic markup language for publishing and sharing ontologies on the Web. OWL defines *classes*, *properties* (binary relation that specifies class characteristics), *instances* (individuals that belong to the classes) and *operations*. Recent research efforts [8,9] concern the translation of RBAC model onto policy languages using OWL. [8] argues that *Policy languages grounded in Semantic Web technologies allow policies to be described over heterogeneous domain data and promote common understanding among participants who not use the same information*, and using OWL *will help in developing security frameworks with well understood and verifiable security properties for open, dynamic environments, which require coordination across multiple organization […]*.
To represent the responsibility-RBAC model and remain aligned with the current research, we retain some elements of the R*OWL*BAC representation and extend it with the definition of a new domain for the responsibility-RBAC model, called `rrbac` (figure 5). R*OWL*BAC provides following classes: *Action*, *Subject*, *Object* (lines 1 to 3) and two subclasses of action: *permission* and *prohibition* (lines 5 to 8). We also prefer the representation of the role as a class (1st approach of [9], line 4) and the representation of the separation of duty (SoD) by the property disjointWith. The SoD is the concept of having at least two people required to complete a task to prevent too much power for a single person. In order to bypass the addition of new rules and to avoid the problem of detection of constraint violation by the DL-reasoner (see section 5), the SoD is represented at the responsibility layer. SoD can be static (SSoD) or dynamic (DDoD) if it is function of the run time environment. We do not consider the representation of the dynamic SoD in this paper. To represent the responsibility in the new rrbac domain a new owl class is needed (line 12). The user to responsibility and the responsibility to role assignments are represented by lines 13 to 18.

```
1  Action a rdfs:Class
2  Subject a rdfs:Class
3  Object a rdfs:Class
4  rbac:Role a owl:Class
5  PermittedAction rdfs:subClassOf Action
6     owL:disjonctionWith ProhibitiedAction
7  ProhibitiedAction rdfs:subClassOf Action
8     owL:disjonctionWith PermittedAction
9  Subject rdfs:property, owl:FunctionalProperty
10    rdfs:domain Action
11    rdfs:range Subjects
12 rbac:responsibility a OWL:Class
13 rbac:role owl:ObjectPropety rdf:ID=”isComposedOf”
14    rdfs:domain rbac:role
15    rdfs:range rrbac:responsibility
16 rrbac:responsibility owl:ObjectPropety rdf:ID=”isAssignedTo”
17    rdfs:domain rrbac:responsibility
18    rdfs:range rrbac:employee
```

Fig. 5. Responsibility-RBAC representation in OWL

Figure 6 illustrates the permission to responsibility association that is represented by the creation of a subclass of PermittedAction. E.g. Buy material for a project is created and only allowed to employees that are assigned to the role BudgetManager is represented with an OWL class expression to create classes of permitted actions (lines 14 to 25) for a specific action and whose subjects are employees assigned to the concerned responsibility. The role is represented as an exact set of responsibilities (lines 5 to 11) and to illustrate the SoD, suppose that BudgetManager is a sub-role of ProjectManager and that an employee may not have access to both roles BudgetManager and BuyerOfficer together (line 13). Finally, the hierarchical is represented using the rdfs constraint subClassOf at the roles layer. Line 26 represents the role project manager which is the superior hierarchical role of the buyer officer.

```
1  ProjectManager rdfs:subClassOf rbac:Role
2  BudgetManager rdfs:subClassOf rbac:Responsibility
3  TeamManager rdfs:subClassOf rbac:Responsibility
4  OutcomesManager rdfs:subClassOf rbac:Responsibility
5  owl:Class rdf:ID="ProjectManager"
6     owl:oneOf rdf:parseType=”Collection”
7        owl:Thing rdf:about=”BudgetManager”
8        owl:Thing rdf:about=”TeamManager”
9        owl:Thing rdf:about=”OutcomesManager”
10    /owl:one of
11 /owl:Class
12 BuyerOfficer rdfs:subClassOf rbac:Role
13 BudgetManager owl:disjointWith BuyerOfficer
14 PermittedBuyAction a rdfs:Class
15    rdfs subClassOf rbac:PermittedAction,
16    owl:equivalentClass [
17       a owl:Class
18       owl:intersectionOf
19         ( Buy
20            [ a owl:Restriction
21                owl:allValuesFrom ex:BudgetManager
22                owl:onProprty rbac:subject
23            ]
24         )
25      ]
26 BuyerOfficer rdfs:subClassOf ProjectManager
```

Fig. 6. Illustration of responsibility-RBAC representation in OWL

5 Related work regarding the translation of RBAC into policy

This section explains how our approach handles the weakness of other ones related to the translation of RBAC into policy. From the existing work, we focus our review on what we consider are the two most significant ones: R*OWL*BAC and XACML+OWL. In R*OWL*BAC [9], Finin et al. propose two approaches to define an OWL domain to represents RBAC. In the first approach, the role is considered as a class. The hierarchy between roles is represented using subClassOf and the SoD is represented

using the property disjointWith. The association of permission or prohibition to role is achieved with an OWL class expression equivalent to our representation of the permission to responsibility assignment. The second approach (figure 7) models a role as an instance of the generic role and uses the ObjectProperty role to link a subject to her possible role (lines 2 to 4). The hierarchy between roles, SoD and the permission to role association is represented by the creation of a new property, respectively: subRole (lines 5 to 7), ssod (for static SoD, lines 8 to 10), dsod (for dynamic SoD) and permitted (lines 11 to 13). Figure 8 illustrates that second approach.

```
1   rbac:Role a owl:Class
2   rbac:Role owl:ObjectProperty
3      rdfs:domain rbac:Subject
4      rdfs:range rbac:Role
5   rbac:subRole owl:TransitivePropety
6      rdfs:domain rbac:Role
7      rdfs:range rbac:Role
8   rbac:ssod owl:symmetricProperty, owl:TransitiveProperty
9      rdfs:domain rbac:Role
10     rdfs:range rbac:Role
11  rbac:permitted rdfs:propety
12     rdfs:domain rbac:Role
13     rdfs:range Action
```

Fig. 7. R*OWL*BAC second approach representation in OWL

```
1   BudgetManager rbac:subRole ProjectManager
2   BudgetManager rbac:ssod BuyerOfficer
3   BudgetManager rbac:permitted Buy
```

Fig. 8. Illustration of R*OWL*BAC second approach representation in OWL

For Ferrini et al. [8], the analysis of both R*OWL*BAC representations [9] shows that the first approach has the disadvantage of being inconsistent when 2 classes (D_i and D_j) are at the same time included (according to the role-hierarchy) and subject to SoD. Ferrini et al. also uses the ROWLBAC second approach to model RBAC in OWL (namely, the association between a subject and a role is represented by the ObjectProperty hasRole(subject,Role)). However, this has the disadvantage that constraints applying to properties to bind roles together (such as for DSoD or SSoD) is not handled by the standard DL-reasoner [8]. Ferrini et al. defines a framework to integrate XACML and OWL ontologies for supporting RBAC. It proposes to decouple the management of constraints such as the SoD from the specification and enforcement of XACML policies. The framework includes a critical module to support the DSoD that is based on an obligation to update the ontology with the information related to permissions granted to a subject. The principle is that when a DSoD exists and when a permission has already been granted to a subject, the obligation to update the ontology for another permission (that may not be assigned to the subject during the same session) will fail because it results in an inconsistency in the ontology. The failure of that obligation results in the denial of the second permission.

In XACML+OWL, a role is represented as a class and the hierarchy by the ObjectProperty subRoleOf (Role, Role). The SoD is represented with the property disjointWith. The disadvantage is that it solves the translation of the SoD constraint with the manipulation of an obligation generator module that supports the automatic creation of policy. This solution is not simple and could be complex to deploy in practice.

The responsibility-RBAC model proposes an innovative approach to represent both of those constraints:

- In RBAC, the SoD is positioned at the role level and specifies that two roles may not be activated together. We position the SoD at the responsibility level (figure 3) and state that two responsibilities may not be activated together. This improvement limits the SoD strictly to the concerned responsibilities and allows an employee to remain assigned to many roles under the condition that all responsibilities that compose that roles respect the SoD constraint. If this is not the case, conflicting responsibilities must be assigned to another employee.
- RBAC positions the concept of role-hierarchy at the role level (figure 3). We keep it as it is, since we agree that the hierarchy reflects the structure between job functions.

6 Conclusions and future works

In this paper we have proposed improvements to some aspect of business IT alignment by refining the assignment of permissions to users based on their business responsibilities. To achieve that, we have proposed an extension to RBAC with responsibility aspect to form the responsibility-RBAC model.

The main contributions are: the optimization of the number of roles by enhancing RBAC with the concept of responsibility and the association of permissions to responsibility, requiring an employee's explicit commitment regarding the tasks they are responsible for, and the representation of the responsibility-RBAC in OWL, including a new perspective to represent the constraint of SoD and hierarchy.

Future work will complete the innovative responsibility-RBAC model, deal with some of the above listed issues such as the translation of the model onto policies and evaluate our proposals with real case studies.

References

[1] Feltus, C., Petit, M., and Dubois, E. 2009. Strengthening employee's responsibility to enhance governance of IT: COBIT RACI chart case study. In Proceedings of the First ACM Workshop on information Security Governance (Chicago, Illinois, USA, November 13 - 13, 2009). WISG '09. ACM, New York, NY, 23-3

[2] Ferraiolo, D. F., Sandhu, R., Gavrila, S., Kuhn, D. R., and Chandramouli, R. 2001. Proposed NIST standard for role-based access control. ACM Trans. Inf. Syst. Secur. 4, 3 (Aug. 2001), 224-274.

[3] Yang, C. 2007. Designing secure e-commerce with role-based access control. Int. J. Web Eng. Technol. 3, 1 (Dec. 2007), 73-95

[4] Kulkarni, D. and Tripathi, A. 2008. Context-aware role-based access control in pervasive computing systems. SACMAT '08. ACM, New York, NY, 113-122.

[5] Sandhu, R.S., Bhamidipati, V., Munawer, Q., The ARBAC97 Model for Role-Based Administration of Roles, TISSEC, 1999

[6] Sandhu, R. S. and Bhamidipati, V. 1998. The URA97 Model for Role-Based User-Role Assignment. IFIP Tc11 Wg11.3 Eleventh international Conference on Database Securty Xi: Status and Prospects (August 10 - 13, 1997). T. Y. Lin and S. Qian, Eds. IFIP Conference Proceedings, vol. 113. Chapman & Hall Ltd., London, UK, 262-275.

[7] Sandhu, R. and Bhamidipati, V. 1998. An Oracle implementation of the PRA97 model for permission-role assignment. 3th ACM Workshop on Role-Based Access Control (Fairfax, Virginia, United States, October 22 - 23, 1998). RBAC '98. ACM, New York, NY, 13-21.

[8] Ferrini, R. and Bertino, E. 2009. Supporting RBAC with XACML+OWL. In Proceedings of the 14th ACM Symposium on Access Control Models and Technologies (Stresa, Italy, June 03 - 05, 2009). SACMAT '09. ACM, New York, NY, 145-154.

[9] Finin, T., Joshi, A., Kagal, L., Niu, J., Sandhu, R., Winsborough, W., and Thuraisingham, B. 2008. ROWLBAC: representing role based access control in OWL. SACMAT '08. ACM, New York, NY, 73-82.

[10] Cholvy, L., Cuppens, F., and Saurel, C., Towards a logical formalization of responsibility, Sixth International Conference on Artificial Intelligence and Law, pages 233-242, 1997.

[11] Dobson, J. and Martin, D., Enterprise Modeling Based on Responsibility, TRUST IN Technology: A Socio-Technical Perspective, Clarke, K., Hardstone, G., Rouncefield, M. and Sommerville, I., eds., Springer, 2006.

[12] Sommerville, I., Lock, R., Storer. T. and Dobson, J., Deriving Information Requirements from Responsibility Models, 21st International Conference, CAiSE 2009, Amsterdam, The Netherlands, June 8-12, 2009. ISBN 978-3-642-02143-5.

[13] Fox, J. A., The uncertain relationship between transparency and accountability" (August 1, 2007). Center for Global, International and Regional Studies. Reprint Series. Paper CGIRS-Reprint-2007-2.

[14] Stahl, B. C. & Wood, Ch. Forming IT Professionals in the Internet Age: A Critical Case Study" In: Yoong, Pak & Huff, Sid (eds.) (2006): Managing IT Professionals in the Internet Age. Idea Group, Hershey, PA: 120 – 139

[15] Laudon, K.C. and Laudon, J.P. (1999), Essentials of Management Information Systems, 4th edition London et al., Prentics Hall.

[16] Vernadat F. B., Enterprise Modelling and Integration, Chapman & Hall, London (1995), ISBN 0-412-60550-3

[17] Yu, E. S. and Liu, L. 2001. Modelling Trust for System Design Using the i* Strategic Actors Framework. Workshop on Deception, Fraud, and Trust in Agent Societies Held During the Autonomous, Eds. Lecture Notes In Computer Science, vol. 2246. Springer-Verlag, London, 175-194.

[18] Qingfeng He, Annies I. Antón, A Framework for Privacy-Enhanced Access Control Analysis in Requirements Engineering, REFSQ'03, Austria, June 2003.

[19] Roeckle, H., Schimpf, G., and Weidinger, R. 2000. Process-oriented approach for role-finding to implement role-based security administration in a large industrial organization. RBAC '00. ACM, New York, NY, 103-110.

[20] Meyer, J.P. & Allen, N.J. (1991). A three component conceptualization of organizational commitment. Human Resource Management Review. 1, 61-98

[21] Vandenberghe, C., Bentein, K., Stinglhamber, F., Affective commitment to the organization, supervisor, and work group: Antecedents and outcomes, Journal of Vocational Behavior, Volume 64, Issue 1, February 2004, Pages 47-71

[22] Mowday, R.T., Porter, L. W., Steers, R. M. (1982), Employee-Organization Linkages: The Psychology of Commitment, Absenteeim, and Turnover. New York: Academic Press.

[23] Buchanana, B., II. (194), Building organizational Commitment: The Socialization of Managers in work organizations, Administrative science Quarterly, 19, pp. 533 – 546.

[24] Hall, D. (1977), Organizational Identification as a function of Career Pattern and Organizational Type, Administrative Science Quarterly, 17, pp. 340 – 350.

[25] Lio, K. (1995), Professional Orientation and Organizational Commitment among Employees: an Empirical Study of Detention Workers, Journal of Public Administration Research and Theory, 5, pp. 231 – 246.

[26] Niehoff, B. P., Enz, C.A., Grover, R. A. (1990), The Impact of Top-Management Ctions on Employee Attitudes and Perceptions, Group & Organization Studies, 15, 3, 337 – 352.

[27] Florkowski, G., Schuster, M. (1992), Support for Profit Sharing and Organizational Commitment: A Path Analysis, Human Relations, 45, 5, pp. 507 – 523.

[28] Blau, G. J. (1985), The measurmement and Prediction of Career Commitment, Journal of Occupational Psychology, 58, pp. 277 – 288.

[29] Pfeffer, J. (1998). The Human Equation. Boston, MA., Harvard Business School Press.

[30] Meyer, J. P. Allen, N. J. (1984), Testing the 'Side-Bet Theory' of Organizational Commitment: Some Methodological Considerations, Journal of Applied Psychology, 69, pp. 372 – 378

[31] Porter, L.W., Steers, R. M., Mowday, R. T., Boulian, P. V. (1974), Organizational Commitment, Job Satisfaction, and Turnover Among Psychiatric Technicians, Journal of Applied Psychology, 59, pp. 603 – 9.

[32] Williams, E.S., Rondeau, K.V., Francescutti, L.H., Impact of culture on commitment, satisfaction, and extra-role behaviors among Canadian ER physicians, Leadership in Health Services, 2007, vol. 20, Issue 3, 147-158.

[33] Zhang, D., Ramamohanarao, K., Versteeg, S.,Zhang, R. RoleVAT: Visual Assessment of Practical Need for Role Based Access Control, ACSAC 2009.

[34] COBIT 4.1, Control Objectives for Information and Related Technology, Information Systems Audit and Control Association.

[35] McGuinness, D. L. and van Harmelen, F. Owl web ontology language overview, February 2004. http://www.w3.org/TR/owl-features/.

[36] Moses, T. eXtensible Access Control Markup Language (XACML) Version 2.0. OASIS Standard, 200502, 2005.

[37] Organ, D. W. (1988). Organizational Citizenship Behavior - The Good Soldier Syndrome. (1st ed.). Lexington, Massachusetts/Toronto: D.C. Heath and Company.

[38] Lu, S., Yuan, H., Liu, Q., Wang, L., Dssouli, R.,Access Control in e-Health Portal Systems, 4th Innovations 2007, IEEE Press, November 18-20, 2007, pages 88-92.

[39] Feltus, C., Petit, M., Building a responsibility model using modal logic - towards Accountability, Capability and Commitment concepts, aiccsa, pp.386-391, 2009 IEEE/ACS International Conference on Computer Systems and Applications, 2009

[40] ISO/IEC 38500 (2008), International Standard for Corporate Governance of IT.

[41] Sarbanes, P. S. and Oxley, M. (2002) "Sarbanes-Oxley Act of 2002"