Enhancing the ArchiMate® Standard with a Responsibility Modeling Language for Access Rights Management

Christophe Feltus, Eric Dubois, Erik Proper
Public Research Centre Henri Tudor, EE-Team[†], Luxembourg
+352 42 59 91 – 1
{firstname.name@tudor.lu}

Iver Band
Standard Insurance Company
1100 SW Sixth Ave
Portland, Oregon
+1 503 – 321 – 6248
iver.band@standard.com

Michaël Petit
PReCISE Research Centre,
Faculty of Computer Science,
University of Namur, Belgium
+32 81 72 52 59
mpe@info.fundp.ac.be

ABSTRACT

In this paper, we describe an innovative approach for aligning the business layer and the application layer of ArchiMate to ensure that applications manage access rights consistently with enterprise goals and risk tolerances. The alignment is realized by using the responsibility of the employees, which we model using *ReMoLa*. The main focus of the alignment targets the definition and the assignment of the access rights needed by the employees according to business specification. The approach is illustrated and validated with a case study in a municipal hospital in Luxembourg.

Categories and Subject Descriptors

H.1.1 [**Models and Principles**]: Systems and Information Theory.

General Terms

Management, Performance, Design, Reliability, Experimentation, Security, Languages, Theory, Verification.

Keywords

Access rights, Business/IT Alignment, ArchiMate, Responsibility, Case study.

1. INTRODUCTION

The access rights management process is central to the field of information security because it impacts most of the functions of the information system, such as the configuration of the firewall, the access on the fileserver and the authorization to perform software operations. Furthermore, the management of access rights is complex because it involves the profiles of many different actors, from administrative assistants to top managers, and concerns all enterprise architecture layers, from business to technology. On one hand, access rights to IT components must be defined by functional requirements and, on the other, according to the governance needs. Functional requirements dictate that employees must have the access rights necessary to perform their jobs. Governance requirements, on the other hand, provide the protections that keep organizations safe, legal, reputable and functioning. They are typically focused on the accuracy and overall quality of access rights.
Existing access control models and rights engineering methods do not adequately represent functional and governance requirements. They depict access rights as technical data managed by IT processes only distantly related to functional and governance requirements. Instead, we introduce a responsibility modeling language *ReMoLa* to model responsibility at the business layer and link it to the application layer. This language extends the standard ArchiMate® visual modeling language for enterprise architecture. The language explicitly allows *ReMoLa* allows us to model the relationships between employees and the activities that they must perform, and to reflect those relationships within the application layer that must implement them.
The enterprise architecture models (EAM), and more particularly ArchiMate, enable the illustration of the interrelations between the different layers of an enterprise architecture, e.g. the business, application and the technology layers and, according to different aspects such as the behavior, the information or the static structure. Those models provide views that are comprehensible for all stakeholders and permit to make decisions knowing the impact on the whole enterprise. For instance, the enterprise model architecture permits to understand the impact on the technical layer of a new business service integrated in the business layer and, consequently permits to analyze the server capability. In the other sense, the failure of a server has an impact on an application and so on business services. The enterprise architecture model permits to overseen the impact and to improve the alignment.
For supporting the alignment between the enterprises' layers, the enterprise architecture models have undergone major improvements during the first decade of 2000 and ArchiMate has appeared to be a very interesting one, promoted and sustained by the Open Group[1]. Even if the advantages of the enterprise architecture models are not to be demonstrated anymore, the high abstraction level of the modeled concepts and of the links between those concepts make it sometimes difficult to use the architecture models to perform, verify or justify concrete alignments. Actually, enterprise architecture models do not permit to engineer precisely the access right provided to the employee at the application layer based on the specification from the business layer. Therefore, we complete the ArchiMate enterprise architecture model with *ReMoLa* in this paper.
The foreseen advantages of integrating both, ArchiMate and *ReMoLa,* are the enhancement of the alignment between the concepts of the business layer, between the concepts of the application layer and between concepts from both layers. Afterwards, this alignment leads to define the access rights to be provided to the employees based on their responsibilities.
In the next section, we introduce *ReMoLa,* the responsibility modeling language. Afterwards, we integrate *ReMoLa* with the

[1] http://www.opengroup.org/archimate/

[†] Enterprise Engineering Team is a collaboration between CRP Henri Tudor, Radboud University Nijmegen and the University of Applied Science Arnhem-Nijmegen (http://www.ee-team.eu)

business layer of ArchiMate, what allows us defining employee's responsibility. In section 4, we adapt the application layer of ArchiMate to support the provisioning of access right according to business specification. In the fifth section, we illustrate the whole with a case study of a municipal hospital from Luxembourg for concluding the paper.

2. MODELING RESPONSIBILITY

The elaboration of the responsibility meta-model (Figure 1) has been performed based on a literature overview. As explained in previous papers [1,2], we have, in the first place, analyzed how responsibility is included in information technology professional frameworks, in the field of requirements engineering and role engineering, and in the field of access right with the review of access control models. Afterwards, this literature overview has been completed by a literature review in the field of Human Sciences (psychology, sociology, and management).

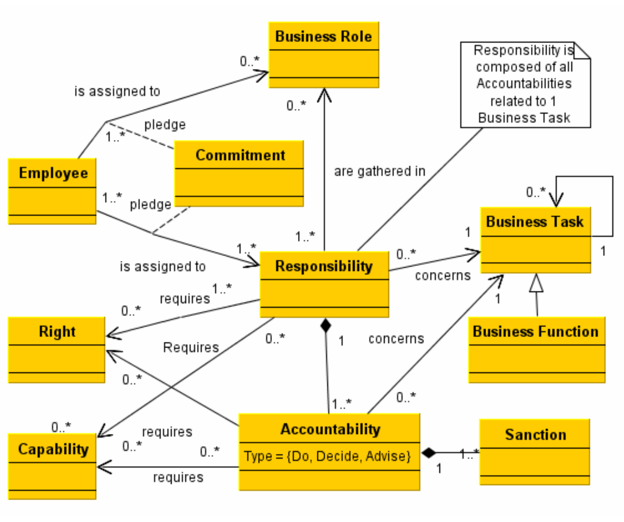

Figure 1. *ReMoLa* modeled in UML.

On figure 1, the most meaningful concepts of *ReMoLa*, are defined in the following way:

- The **responsibility** is a charge assigned to an employee to signify his accountabilities concerning a business task, and the right and capacity required to perform those accountabilities.
- The **accountability** represents the obligation of what have to be done concerning a business task and the justification that it is done to someone else, under threat of sanction
- The **capability** represents the qualities, the skills or the resources intrinsic to the employee and required to perform accountability.
- The **right** represents the resources provided by the company to the employee and required to perform accountability.
- The **assignment** is the action of linking an agent to a responsibility. Delegation process is the transfer of an agent's responsibility assignment to another agent.

3. DEFINING RESPONSIBILITY AT THE BUSINESS LAYER

In this section, we integrate *ReMoLa* with the business layer of the ArchiMate enterprise architecture modeling language. This integration allows defining the access rights provided to the employees at the application layer according to their defined responsibilities at the business layer.

The integration of *ReMoLa* with ArchiMate is achieved using a three steps approach to integrate models defined by Petit in [6]. The first step is the preparation of the integration, the second step is the investigation and the definition of the correspondences and the third step is the integration of both meta-models.

3.1 Preparation for integration

As defined in [6], this first step of the integration of two meta-models requires a certain preparation of the integration. Therefore, an integration strategy is defined and provides the baselines for the integration process such as the context of the integration, the selection of a common language for the representation of the meta-models being integrated, the expected abstraction layer of the concepts represented in the integrated meta-model and so forth.

The languages at our disposition are, e.g. Telos [7], UML, and so forth. Telos has the advantages of being based on mathematical foundations, of being expressive, and of using a limited number of concepts. UML has the advantages of offering more representation choices, of being less informal and, as a consequence, might be more intuitive [6]. For the integration, we have also selected UML as common representative language. ArchiMate has been traduced in UML in [8].

Our integration aims at enhancing the alignment between the business and the IT layer of ArchiMate and, thereby, enhancing the definition of the access right. As a consequence, some concepts of ArchiMate will not be considered such as e.g. those from the technical layer, the value and so forth. Some concepts from *ReMoLa* have also not been considered because they are not of the appropriate abstraction layer like the sanction, the commitment or the motivation.

3.2 Investigation and definition of the correspondences

In [6], the author explains that this second step analyzes the correspondences between the classes of the meta-models. Those correspondences exist if correspondences between instances of these classes, taken two by two, can be generalized. Therefore, it is advisable to carry out one or more case study(ies) to model real world elements with both languages and, to compare the semantics of the obtained models. The correspondences between the models' elements have been analyzed during complete case study at the municipal hospital which is summarized in section 5. During the investigation and definition of the correspondences, we have model this case study with ArchiMate and with *ReMoLa* first. Afterwards, we generalize the case modeling considering:

(1) Classes of both meta-models semantically equivalent:

- The business actor and the employee
- The business role (in ArchiMate) and the business role (in *ReMoLa*)
- The business object and the information
- The business function and the association accountability–task

(2) Classes not existing in ArchiMate:

- The concept of capability. This concept exists explicitly in *ReMoLa* and implicitly in ArchiMate that considers that a business function groups behavior according to, for instance, the required skill and knowledge [5]. This concept is explicitly introduced in the integrated model.
- The concept of right. This concept exists explicitly in *ReMoLa*. In ArchiMate, *the bu*siness function also aims at grouping behavior accord to the required resource [5] but the semantic of the resource and its difference with the business object is not obvious.

(3) UML association between those classes that are equivalents:

- The UML association that assigns a business actor to a business role

- The UML association that assigns a business role to business function
- The UML association that associates a business function with the business object which it accesses

3.3 Integration of *ReMoLa* in ArchiMate

The third step defined in [6] corresponds to the integration of the meta-models. During analyze of the correspondences between the classes and the UML associations between the classes, we have observed some minor divergences. Notwithstanding the influence of those divergences, to consider that a sufficient correspondence exists between the elements and to consider them during this third step of integration, we have to analyze that divergence in depth and formalize the integration rules to consider for having a perfect integration.

Our objective is to elaborate an integrated meta-model that enriches the business layer of ArchiMate with the meta-model of *ReMoLa*. Therefore, our integration strategy is as follows regarding the classes of both meta-models: (1) when an exact correspondence between one class from ArchiMate and one class from *ReMoLa exists*, we preserve the name of the ArchiMate class, (2) when the class of *ReMoLa* has no corresponding class in ArchiMate, this class is integrated in the integrated meta-model and it preserves its name from *ReMoLa*, (3) when a correspondence with conflicts between the definition of the classes exists, the classes are integrated in the integrated meta-model and we preserve the name of the ArchiMate class but, additionally, we includes integrations rules that need to be followed in case of using the integrated meta-model. We observe that, all the classes from ArchiMate have a corresponding class in *ReMoLa*. The correspondence between the UML associations in ArchiMate and in *ReMoLa* is also analyzed during the integration of both meta-models. Two situations coexist: Firstly, one direct association between two classes of ArchiMate corresponds to one direct association between the equivalent classes of *ReMoLa*. In this case, it can exist short semantic difference(s) between the associations, and integration rule sometimes needs to be defined to consider this difference. Furthermore, the name of the association from ArchiMate is preserved. Secondly, one direct association between two classes of ArchiMate corresponds to one indirect association between the equivalent classes in *ReMoLa*. In this case, the indirect UML associations are renamed.

1. <u>Classes that correspond exactly</u>

- The business role in ArchiMate and the business role in *ReMoLa*
- The business object in ArchiMate and the information in *ReMoLa*

2. <u>Classes that only exist in *ReMoLa*</u>

- Responsibility
- Right
- Capability
- Accountability

3. <u>Classes that correspond under integration rules</u>

- The business function and the task. The first integration rule is that the definition of the business function in ArchiMate is completed by a type of obligation for the business actor. As a consequence, the *ReMoLa*'s class of task corresponds to an obligation concerning a business function being performed by the business actor. This integration rule completes the business function that have to be accurate enough to define what type of obligation is expected by the business function, e.g. the business function producing a rapport should be defined more precisely like, for instance, make the report, review the report, manage the team that produces it and so forth. Moreover, the business actor that produces it must justify its realization.
- The business actor and the employee. The second integration rule is that the ArchiMate class of business actor is limited to a human actor. This integration rule exists due to the commitment which is pledge by an employee, once he is assigned to a responsibility. If the business actor is a machine, we assume there is no question of commitment since a machine executes the operation it is programmed for. If the business actor consists of a group of humans, the commitment is not able to be check individually and, by consequence, it is not possible to validate that responsibility assigned to an employee from the group will be fulfilled since no employee personally commits to it.

UML associations integration:

1. <u>Equivalent associations between two classes of AchiMate and the corresponding two classes of *ReMoLa*</u>

- The business actor is assigned to the business role

2. <u>Links from *ReMoLa* in the integrated meta-model complete or replace the associations from ArchiMate</u>

- The business actor is assigned to a responsibility that compose a business role is an alternative association to the association that associates the business actor with the business role.
- The business role is composed of responsibilities which are composed of business function replace the direct association between the business role and the business function.
- The association between the business function that composes a responsibility and the rights concerning a business object that are required to a responsibility replace the ArchiMate direct access association from the business function to the business object.

3. <u>New associations from *ReMoLa*, which do not exist in ArchiMate, are integrated in the integrated meta-model.</u>

- The responsibility required capability
- The capability is necessary for a business function

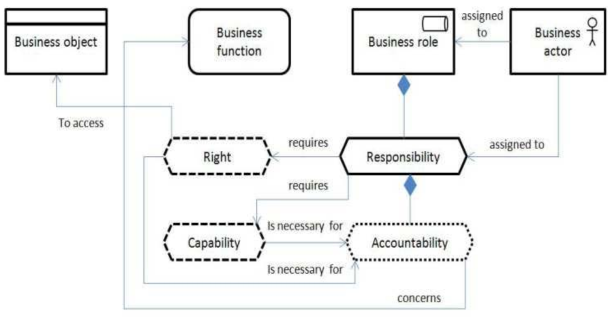

Figure 2. *ReMoLa* integrated with ArchiMate.

The integration of new classes from the responsibility meta-model and the consideration of the above integration rules from the analyze of the correspondence between the classes of business object and information, and the business role in ArchiMate and in *ReMoLa* permit to assemble both meta-models on a unique schema in UML, which is afterwards represented with ArchiMate symbols on Figure 2. To integrate responsibility in ArchiMate, we used the symbols of full line hexagonal for responsibility, dash line hexagonal for right and capability, and dot line hexagonal for accountability.

4. ACCESS RIGHT MANAGEMENT

4.1 Previous work

RBAC is a model that facilitates the management of the access rights to application component such as software or server. To manage these access rights related to application components at the application layer, it is necessary to be able to interpret the business components such as the business role, the business actor or the permission at the application layer.

Therefore, [4] introduces three data objects: the user, the role, and the permission at the application layer (see Figure 3). The interpretation of a concept from the business layer by a concept at the technical layer in ArchiMate is only possible by using a Realization link [5]. Unfortunately, this realization link only exists between a data object that realizes a business data. Since this realization does not formally exist in ArchiMate, [4] does not explicitly consider that: 1) the role from the application layer realizes the business role (or the concept of role from RBAC), 2) the user from the application layer realizes the business actor (or the concept of user from RBAC) and 3) the permission from the application layer realizes a type of access to a business object (or the permission concept from RBAC).

4.2 Realization link

Although it is not considered by [4], this realization link is necessary, especially, in the field of the access right management. Indeed, an access right management solution needs to be able to consider an electronic representation of the business user to calculate the access rights he needs. Therefore, we introduce a new realization relationship in ArchiMate between Business and Application layer concepts. This relationship specifies that a concept at the application layer is an electronic representation of a concept at the business layer.

This realization relationship is represented by using a double line with a stealth arrow in Figures 3, 4, 7 and 9.

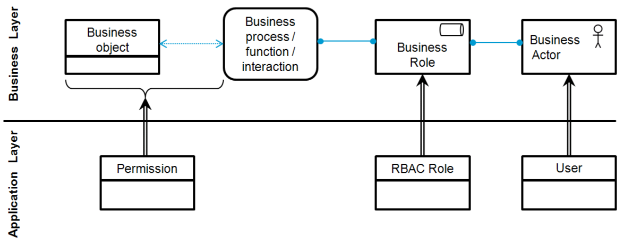

Figure 3. User-role and permission-role assignment.

Access rights management with RBAC is defined by a set of functions such as the assignment of users to roles and the review of this assignment, the assignment of permissions to roles and the review of this assignment, the management of the roles hierarchy, the management of the separation of duties constraint, the management of the sessions, and the performance of access check. [4] has explained all these functions. In Figure 3, we only highlight the two functions that are important for our research: the users to roles assignment and the permissions to roles assignment. These functions are represented by using a horizontal chevron symbol.

The users to roles assignment function requests the creation of a new data object named User-Role Assignments and the permissions to roles assignment function requires the creation of a new data object named Permissions-Roles Assignment. These two new data objects contain the list of the existing assignments. To execute the assignment of users to roles, the user to role assignment function reads the user and the role data objects, and reads and writes the user-role assignment data object. To execute the assignment of permissions to roles, the permission to role assignment function reads the permission and the role data objects, and reads and writes the permission-role assignment data object. The integration of RBAC at the business layer and the association of the RBAC role with ArchiMate's business role seem appropriate. In this case, we have a RBAC role that corresponds to a real role from the company such as the roles encountered in the organizational chart (we have, for instance, the business role of doctor, of medical secretary, of nurse, and so forth). Accordingly, we have an exact correspondence between the existing business roles of the company and the roles that are used to manage the access rights. However, the problem with this integration is that it requires a perfect alignment between the permissions needed by the employees assigned to these roles and the permissions really assigned to them. This is based on the postulate: Mostly all users with the same role have exactly the same tasks to perform and need exactly the same permissions.

In practice, this is slightly different since, although all users with the same role globally achieved the same tasks, there always exist some differences between the responsibilities of the employees assigned to the same role. For instance, although all the doctors from the role doctor, a priori, require the same access rights over the information system, in practice, the doctors never exactly perform the same tasks and never need access to the same information, e.g. some doctors, additionally to their roles, manage the unit and need access to financial software, while others are members of ethical committee and need access to reporting tools, others are specialized in specific professional in a specialty and need access to dedicated software. The weaknesses related to the roles' definition have been demonstrated in [3, 9, 10].

4.3 Our approach

Our approach considers the assignment of rights to the employee based on their responsibilities. Therefore, the responsibilities integrated with the business layer of ArchiMate needs ultimately, to be represented at the application layer.

4.3.1 Representation of responsibility, business role and business actor

At the application layer, we introduce a data object *responsibility* that realizes (according to [5]) the business concept of responsibility and a data object *business role* which realizes the business object of business role (see Figure 4). This data object of business role is necessary for representing the composition of business roles with responsibilities at the application layer.

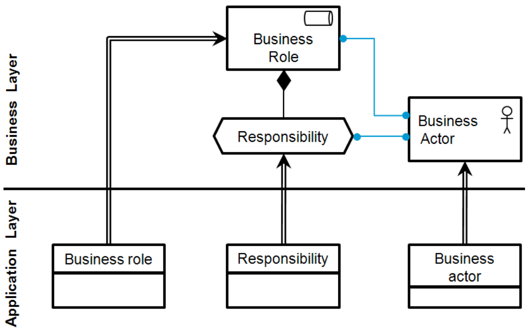

Figure 4. Business role with responsibility composition.

We also introduce the data object of *business actor* that corresponds to the computerized representation of the business actor from the business layer. The business actor representation corresponds to an electronic ID or to a unique identification. The data object of business actor does not perform a behavior or an application function, but it is used by some application functions to calculate whether the business actor from the business layer may access specific application functions or specific application data used by this application function.

4.3.2 Responsibility to business role assignment

As explained in Section 3, a business role is composed of one or more responsibilities. To administer this composition in ArchiMate, we integrate, at the application layer, an application function named Compose Business roles with Responsibilities, and a data object named Responsibility-Business role Compositions. This data object represents a set of Responsibilities that compose a set of Business roles. As explained on Figure 5, to compose the business roles with responsibilities, the application function *Compose Business roles with Responsibilities* needs to access the three following data objects: Responsibility, Business role and Responsibility-Business role Compositions.

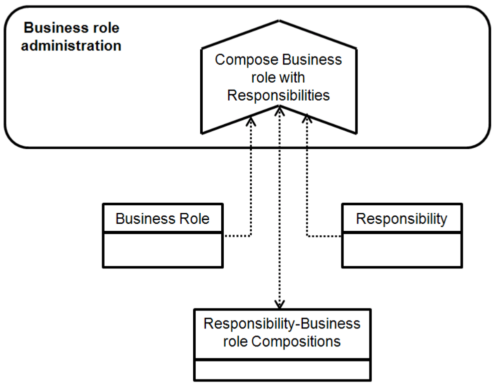
Business role administration
Compose Business role with Responsibilities
Business Role
Responsibility
Responsibility-Business role Compositions

Figure 5. Business role administration.

4.3.3 Business actor assignment

To administer the assignment of a business actor to a responsibility and/or to a business role, we integrate three new data objects named (1) Business actor, that realizes the Business actor from the business layer, (2) Business actor-Responsibility Assignment, that represents a set of Responsibilities assigned to a set of Business actors, and (3) Business actor-Business role Assignment, that represents a set of Business actors assigned to a set of Business roles (see Figure 6). We also integrate two new application functions: Assign Business actors to Responsibilities and Assign Business actor to Business role.

As explained in Figure 6, in order to assign responsibilities to business actors, the application function Assign Business actors to Responsibilities access the three following data objects: Business actor, Responsibility and Business actor-Responsibility Assignment. Equivalently, to assign a Business role to a Business actor, the application function Assign Business actor to Business role access the three following data objects: Business actor, Business role and Business actor-Business role Assignment

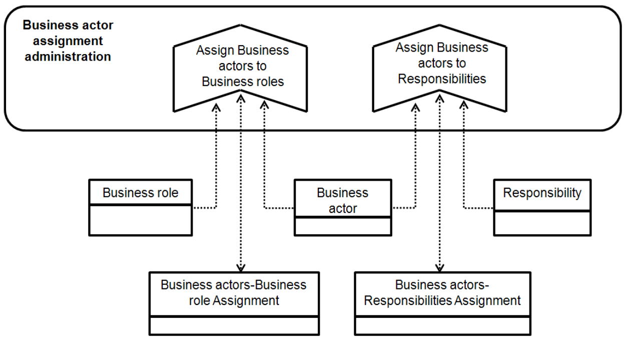
Business actor assignment administration
Assign Business actors to Business roles
Assign Business actors to Responsibilities
Business role
Business actor
Responsibility
Business actors-Business role Assignment
Business actors-Responsibilities Assignment

Figure 6. Business actor assignment administration.

4.3.4 Representation of Permissions

At the business layer, a permission, corresponds to a type of access right to a business object. The data object Permission realizes the Permission from the business layer to the application layer (see Figure 7).

4.3.5 Permissions to responsibilities

As explained in Section 2, a responsibility requires one or more rights. Permission is a type of right to access a business object. To administer this assignment of permissions too, we integrate an application function named Assign Permissions to Responsibilities, and a data object named Permission-Responsibility Assignment at the application layer.

As explained in Figure 8, to assign permissions to responsibilities, the application function Assign Permissions to Responsibilities needs to access the three following data objects: Permission, Responsibility, and Permission-Responsibility Assignment

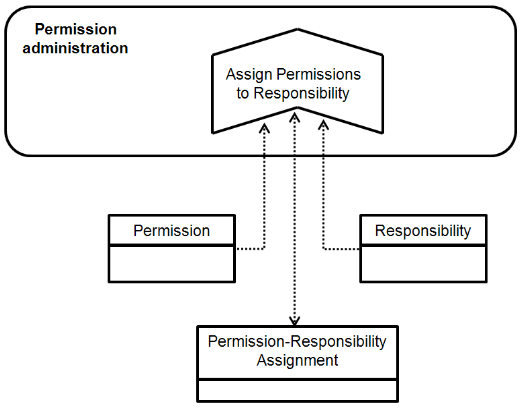
Permission administration
Assign Permissions to Responsibility
Permission
Responsibility
Permission-Responsibility Assignment

Figure 7. Permission administration.

4.4 Permissions Assignment Optimization

Our approach used the concept of responsibility as a pivot between the business layer and the application layer. Firstly, we consider that a business role is composed of a set of responsibilities (defined at the business layer) and secondly, we consider that permissions provided with application functions at the application layer are necessary to perform the responsibilities. These permissions are calculated at the business layer but are provided and managed at the application layer.

Regarding this second point, in practice, we are confronted at the application layer with a large amount of responsibilities that need a large amount of permissions and each permission may be assigned to a set of different responsibilities. This situation is close to the situation where a large amount of users are assigned to a large amount of permissions and each permission may be assigned to a large amount of users. Therefore, in the next chapters, we analyze how it is possible to consider the RBAC model to enhance the assignment of permissions to responsibilities, at the application layer and we consider two assignment functions: the responsibilities to roles assignment and the permissions to roles assignment.

4.4.1 Representation of RBAC

At the application layer, we need to introduce a data object for the RBAC role, such as realized in [4]. This data object facilitates not the assignment of permissions to users, but the assignment of permissions to responsibilities. It is only used at the application layer, to optimize the management of the permissions, and has no correspondences at the business layer. We keep the data object of responsibility that realizes the business concept of responsibility as explained in Figure 4 and, we keep the data object of permission that corresponds to a type of access to a business data as explained in Figure 8.

The data object of the RBAC role is a type of application role that corresponds with a logical gathering of business actor representations which have the same operations to perform on the information system (IS) and therefore, request the same permissions regarding the data objects. As a consequence, the RBAC role is different to the business role such as defined in ArchiMate.

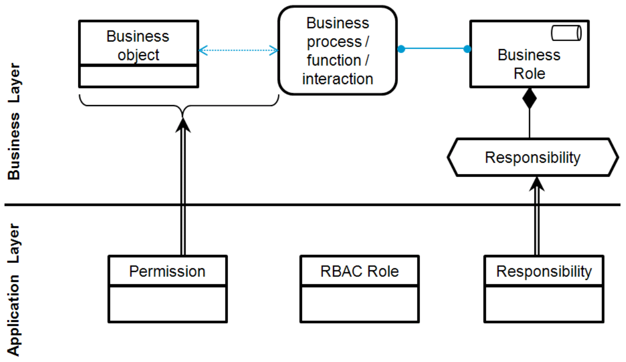

Figure 8. Permission-responsibility assignment optimization.

4.4.2 *RBAC role administration*

To administer the assignments of responsibilities to the RBAC role and the assignment of permissions to this role, we integrate, at the application layer, two application functions named Assign Responsibilities to RBAC role and Assign Permissions to RBAC role.

Additionally, we also integrate two new data objects named Responsibility-RBAC role Assignment, which represents a set of responsibilities assigned to an RBAC role, and Permission-RBAC role Assignment, which represents a set of permissions assigned to an RBAC role. As explained in Figure 9, to assign responsibilities to an RBAC role, the application function Assign Responsibilities to RBAC role needs to access the three following data objects: Responsibility, RBAC role and Responsibility-RBAC role Assignment. Equivalently, to assign permission to an RBAC role, the application function Assign Permission to RBAC role needs to access the three following data objects: Permission, RBAC role and Responsibility-RBAC role Assignment.

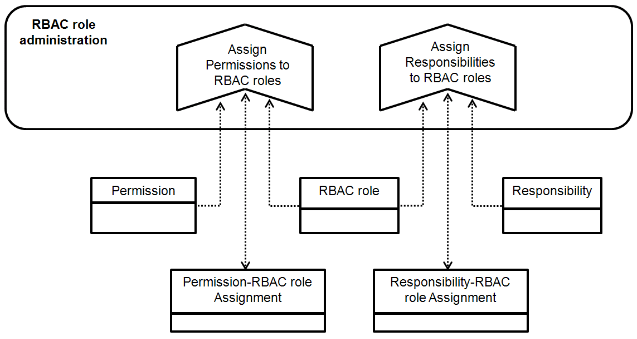
RBAC role administration
Assign Permissions to RBAC roles
Assign Responsibilities to RBAC roles
Permission
RBAC role
Responsibility
Permission-RBAC role Assignment
Responsibility-RBAC role Assignment

Figure 9. RBAC role administration.

4.4.3 *Discussion*

In our approach, we keep the use of the concept of a business role that exists in the company and is consequently useful to manage the business actors, as well as their responsibilities. In parallel, we introduce and define the concept of Responsibility to improve the definition of the responsibilities of the Business actors. Indeed, although the business role offers a macro list of responsibilities to be performed by the business actor, it does not allow managing the responsibilities that are sporadically assigned to or removed from the employees; it does not allow the management of delegation of some responsibilities, etc. The access rights provisioning using the RBAC model and using the mapping of the RBAC role with the business role, as explained previously, presents weaknesses, in terms of accuracy.

To face the question of accuracy, we have, considered providing the access rights to the business actor according to these responsibilities. In companies, access rights are managed with application components at the application layer. Therefore, it was necessary to translate business actor and responsibility at the application layer, to define business roles to responsibility assignment function and to define permissions to responsibility assignment function.

Using responsibility to provide permissions is interesting, but it reduces the advantage introduced by the role based access control model to manage a large amount of users and permissions using roles. As a result, after having introduced responsibility in ArchiMate, we had to face the management of a large amount of access rights to be provided to a large amount of responsibilities. To provide a solution to this problem, we have considered using the RBAC model and we have reintroduced the RBAC role at the application layer.

5. CASE STUDY AT THE HOSPITAL

At the municipal hospital, there is no formal alignment, in terms of access rights to professional software, between the business layer where business roles are defined and assigned to the employees, and the application layer where the access rights are provided to these employees. Therefore, the objective of this case study is to illustrate that the integrated ArchiMate with *ReMoLa* meta-model at the business layer, as well as the enhancement of the permissions to responsibilities assignment using RBAC at the application layer, is a solution that improves the provisioning of professional software access rights to the employees. All along this section, the case study is illustrated with the reception department from the hospital. The case study has been conducted between January 2011 and January 2012, to the rhythm of one meeting a month. During those meetings, the following persons have participated: the Application support manager, the Reception department manager and the Competences manager.

5.1 Hospital business roles analyze

At the municipal hospital, the employees are categorized based on their roles. In the Human Resources (HR) department, the roles of the employees are going to be formalized in the Job description. These job descriptions aim to describe the tasks which are to be performed by a role, as well as the necessary knowledge required to be assigned to this role. The job descriptions, however, do not specify the access rights required on professional software. In this case study, we consider that the business role from ArchiMate corresponds to the business roles from the hospital and that the employees assigned to a role are accountable for doing the tasks described in the job description. To illustrate this, let’s take the job description of the receptionist role which is a thirteen page document that formalizes the five main activities to be performed by this role, i.e.: welcome and inform the patient, perform the various technical and administrative tasks, contribute to the enhancement and evolution of professional practices, train and mentor new employees, and train and supervise trainees. Each of these activities are described by a set of tasks and by the required competences to perform them in terms of knowledge, methodological and technical know-how and, relational ability. The tasks to be performed for the activity: Perform the various technical and administrative tasks, are eg.: encode and control the data relating to the admission of ambulatory or hospital patients, print and give the admission form to the patients, manage daily access to the parking, receive deposits, issue invoices, and so forth.

An organization chart for the reception department structures the activities into eight sub-roles, as follows:

- SR1: Receptionist at the municipal hospital.
- SR2: Receptionist at the pediatric clinic and the maternity
- SR3: Phone reception
- SR4: Info desk
- SR5: Human resources management
- SR6: Department management
- SR7: Room operator
- SR8: Outsourced guardian

5.2 Analysis of the application layer

The architecture of the IS of the hospital is composed of:

1. Vertical software are applications which are used by well defined and well specified healthcare businesses. These are for instance: the management of the laboratory, the endoscopy software, or the management of the polyclinic.

2. Transversal software are those used together by all healthcare businesses. These are for instance: the dispatching of the laboratory's results or the medical imaging. The hospital ERP is the most important transversal software (see Figure 10).

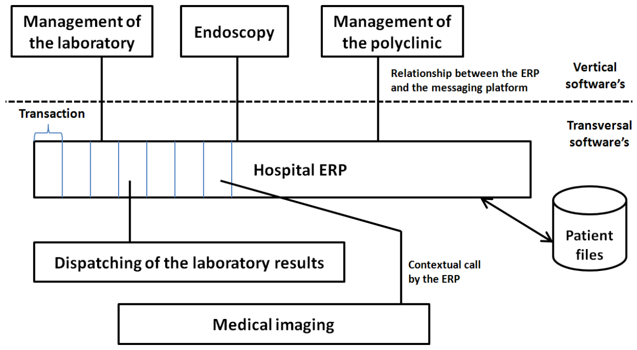
Management of the laboratory
Endoscopy
Management of the polyclinic
Relationship between the ERP and the messaging platform
Vertical software's
Transaction
Transversal software's
Hospital ERP
Patient files
Dispatching of the laboratory results
Contextual call by the ERP
Medical imaging

Figure 10. Hospital municipal application layer.

The hospital ERP is a business management software that offers the possibility to program specific application functions by the owner of the application himself. Therefore, it has been decided to use it, to manage the access rights to all the other software. As a consequence there exists links between the ERP and the vertical software and on the other hand links between the ERP and the other transversal software using contextual calls. With the hospital ERP, the access right management is realized using *AuthorityObject*. *AuthorityObject* is composed of zone(s) from 1 to n based on what authority check is performed. Practically, *AuthorityObject* correspondent to ERP transactions (see Figure 10) and for each transaction, a set of authorizations are defined such as create, modify, delete, view historic, and so forth.

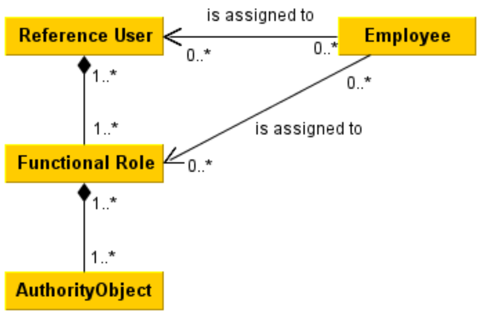

Figure 11. Hospital RBAC role equivalents.

To facilitate their management, *AuthorityObject* is assigned to *Functional roles* like, for instance, the *Functional role* of Search for a patient in the database, create a patient entry, create a transaction, show a transaction, and so forth. Additionally, the concept of *Reference user* has been created to gather a set of *Functional roles*. In practice, one user may be assigned to one or more *Reference user* or to one or more *Functional role* (See Figure 11). The mapping between the application layer of the hospital and the enhanced ArchiMate-*ReMoLa* meta-model allows defining the correspondences between the *AuthorityObject* that corresponds to the kind of right to perform an operation. In the *ReMoLa*-ArchiMate meta-model, that permission corresponds to the data object of permission. The *Functional role* corresponds to a set of *AuthorityObject* which may be assigned to a business user. Therefore, we consider that the *Functional role* component from the hospital application architecture corresponds to the concept of RBAC role of the application layer of the ArchiMate-*ReMoLa* meta-model.

Finally, the *Reference user* corresponds to a set of *Functional roles* and, on the second hand, is assigned to a set of business user. Therefore, we consider that the *Reference user* component from the hospital application architecture also corresponds to the concept of RBAC role of the application layer of the meta-model. Moreover, given that the *Reference user* is composed of *Functional role*, we consider that there exists a role hierarchy between both roles.

5.3 Illustration with the receptionist role

At the application layer, an authorization profile document is defined and formalizes the five *Functional roles* that may be assigned to the employees with the role of receptionist. These *Functional roles* are:

- Patient's basic data encoding, that means Add or create, modify, display, delete patient's basic data and entry, transfer or leaving data related to the patient
- Entry, transfer or leaving patient's data encoding
- Management of the beds status at the hospital
- Medical delivery encoding
- Patient invoices creation and modification

The three first *Functional roles* are aggregated in the *Reference user* of *REFRECEP*. For each of these *Functional roles*, a set of *AuthorityObject*s is defined. These *AuthorityObject*s are managed using an application interface that allows formalization of the concerned rights. In practice, the *Functional roles* and *Reference user*, as well as other rights to specific software, are assigned to the sub-roles as follows:

- SR1: *REFRECEP*, all rights related to equipment ordering software
- SR2: *REFRECEP*, medical delivery encoding, patient invoices creation and modification, all rights related to equipment ordering software
- SR3: *REFRECEP*, all rights related to equipment ordering software, right to read the planning of doctors on duty
- SR4: *REFRECEP*, all rights related to equipment ordering software
- SR5: *REFRECEP*, medical delivery encoding, patient invoices creation and modification, all rights related to equipment ordering software, read and write access to the Excel file: Timetable planning
- SR6: All rights provided to the other sub-roles
- SR7: Read access related to the room agenda in GroupWise multi-users, read access to the ticketing tool.
- SR8: Write access to the reporting software, all rights related to equipment ordering software

5.4 Enhanced permission assignment

As explained in Section 4.4, to align the business role with the application role, we have introduced the concept of responsibility as an intermediary and pivot component. Responsibility composes a business role at the business layer and is assigned with permissions (or with an RBAC role) at the application layer. The analysis of the receptionist job description has allowed defining sixteen responsibilities that required access rights on the IS.

Table 1: Responsibility – Access Rights – Sub-Roles.

ID	Responsibility	Required Access Right	Compose Sub-Roles
1	Perform the entry record	Add or create, modify, display, delete patient's basic data and entry, transfer, or leave data related to the patient	SR1, SR2, SR5
2	Perform the transfer management	Display entry, transfer or leave data related to the patient and all rights related to the statistic software	SR1,SR2, SR5
3	Perform the beds status management	All rights related to the beds status management	SR1,SR2, SR5
4	Perform equipment ordering	All rights related to the equipment ordering software	SR8
5	Perform the medical encoding for billing	All right related to the medical delivery encoding	SR2
6	Perform the creation and de modification of patient invoices (billing)	All rights related to the patient invoices creation and modification	SR2
7	Inform about the beds status	Display rights related to the beds status	SR1, SR2, SR3, SR4
8	Perform the realization of work plans	Read and write access to the Excel file: *Timetable planning*	SR5
9	Perform the control of the monthly worksheets	Read and write access to the Excel file: *Timetable planning*	SR5
10	Perform the management of HR indicators: Overtime, Days off, Hours of recovery	Read and write access to the Excel file: *Timetable planning*	SR5
11	Perform the management of the room	Read access related to the room agenda in Groupwise multi-users	SR7
12	Perform the verification of the infrastructure	Write access to the reporting software	SR8
13	Fix defective infrastructure	All rights related to equipment ordering software	SR8
14	Perform the management of the receptionists	All the rights provided to the sub-roles SR1, SR2, SR3, SR4, SR5, SR7 and SR8	SR6
15	Inform about the doctor on duty	Rights to read the doctors on duty planning	SR3
16	Perform the statistical analysis to follow up the daily business	All rights related to the statistical software	SR5, SR7

The definitions and analysis of the responsibilities of the receptionists have permitted to refine the required access rights for each sub-role recovered in the organization chart of the receptionist department (Table 1). By formalizing the responsibilities, we have isolated the tasks to be performed by each sub-role from the receptionist job description and we have analyzed the access rights they need.

Thereby, we have observed the following differences:

- SR3 and SR4 have too many rights. The employees assigned to the Phone reception and Infodesk role are authorized to add or create, modify, display, delete patient's basic data and entry, transfer, or leaving data related to the patient, although they do not require these rights. They possess all rights related to the beds status management, although, only some of them are required to display information related to the beds status.
- SR1, SR2, SR5 do not have to perform equipment ordering, although they have the right to do it.

6. CONCLUSIONS

We have proposed an approach for enhancing the alignment of the business layer with the application layer, and in particular the enhancement of the access right management and provisioning to employees according to business specifications. This approach takes responsibility as a link between both layers into account. Therefore, responsibility has been modeled in a responsibility modeling language named *ReMoLa* and has been integrated in ArchiMate using the methodology defined in [5].

To illustrate the approach, a case study in a municipal hospital in Luxembourg has been conducted with people in a variety of IT and non-IT roles. We have defined and used the responsibilities of the employees from the receptionist department to align the business role, and sub-roles, defined at the business layer with RBAC role defined at the application layer. At the application layer, the business role and sub-roles, as well as the tasks to be performed, have been recovered from the job description document and from an organizational chart. At the application layer, the RBAC roles have been analyzed in an authorization profile document that defines a set of *Functional roles,* sometimes aggregated in a *Reference user.*

We have observed that using responsibility allows a finer assignment of rights to the employees. For instance, responsibility 16 does not compose any business role or sub-roles, but it may be directly assigned to employees that are assigned to SR1 or SR2. This direct assignment allows the provisioning of some employees, who are responsible for making business analysis, from the SR1 and SR2 sub-role, but not all of them.

This case study has allowed validating the usability of responsibility to perform this alignment. Since the receptionist department was already existing and functioning, the case study did not engineer the access rights without relying on existing resources, but it allows confronting the existing access rights with those calculated by the modeling of the responsibilities. The results of these confrontations were that the employees of five sub-roles (over eight) were assigned to more permissions than they really required in practice.

7. ACKNOWLEDGMENTS

This work is partially supported by the *Fond National de la Recherche* in Luxembourg on the PEARL program ASINE.

8. REFERENCES

[1] C. Feltus, M. Petit, and M. Sloman, Enhancement of Business IT Alignment by Including Responsibility Components in RBAC, 5th Busital workshop, 2010, Hammamet, Tunisia.

[2] C. Feltus, M. Petit, and E. Dubois, Strengthening employee's responsibility to enhance governance of IT: COBIT RACI chart case study. 1st ACM Workshop on Information Security Governance. ACM, New York, NY.

[3] D. Richard Kuhn, Edward J. Coyne, Timothy R. Weil. Adding attributes to role-based access control. Computer, 43(6):79-81, 2010.

[4] I. Band, Modeling RBAC with SABSA, TOGAF and ArchiMate, Creating a Foundation for Understanding and Action, Open Group Conference, Austin, Texas, 2011.

[5] M. Lankhorst. Archimate language primer, 2004.

[6] M. Petit. Some methodological clues for defining a unified enterprise modelling language. ICEIMT '01, pages 359-369, Deventer, The Netherlands, 2003.

[7] J. Mylopoulos, A. Borgida, M. Jarke, M. Koubarakis. Telos: representingknowledge about information systems. ACM Trans. Inf. Syst., 8:325-362, October 1990..H. Jonkers, M.-E.Iacob, M. Wiering. Towards a uml profile for the archimate language, 2004

[8] B. Lang, I. Foster, F. Siebenlist, R. Ananthakrishnan, T. Freeman. A exible attribute based access control method for grid computing. Journal of Grid Computing, 7(2): 169-180, 2008.

[9] M. Covington and M. R. Sastry. A contextual attribute-based access control model. On the Move to Meaningful Internet Systems 2006 OTM 2006 Workshops, pages 1996-2006.