Secure Governance in Enterprise Architecture - Access Control Perspective

Khaled Gaaloul[1] , Marwane El Kharbili[2] and Henderik A. Proper [3]
[1]Centre de Recherche Public Henri Tudor, Luxembourg
[2]University of Luxembourg, Luxembourg
[3]Radboud University Nijmegen, the Netherlands
{fkhaled.gaaloul, erik.properg}@tudor.lu, marwane.elkharbili@uni.lu

Abstract—Enterprise Architecture (EA) models have proven to be very useful for the management and governance of enterprises. Such EA models are used for analysis and steering purposes, thereby leading to a competitive advantage for the enterprise. However, the management of EA model evolution from an initial (As-is) to an a posterior (To-be) state is a challenging task for EA modelers, due to the huge number and the complex dependencies amongst models.

In this paper, we tackle the challenge of a controlled evolution of EA models which seeks to give more control to EA modelers over what the impact of EA evolution means in terms of properties (e.g. security) of the EA. We propose a core knowledge model for representing EA evolution which supports the EA modeler in deciding about the (To-be) model compliance.

Our model is based on the three notions of change operation, artifact-to-artifact dependency, and reactive event-condition-action (ECA) rules. We instantiate our approach for the case where security properties must be maintained through EA evolution.

Keywords: Enterprise architecture, change management, security, access control, knowledge model

I. INTRODUCTION

Enterprises need to negotiate many challenges, such as changes in the economic climate, mergers, acquisitions, and novel technologies. As a result, enterprises need to be agile to improve their chances of survival [1]. Dealing with such changes requires a good steering instrument supporting the ability to analyze the current state of the enterprise, identify and describe alternative future states, guard the cohesion and alignment between the different aspects of an enterprise such as business processes and their ICT (Information and communications technology) support. Enterprise architecture management (EAM) is generally considered to provide such a mechanism for cohesive steer-ing [2], [3]. As formulated in [3], the suggested mission of EAM is to add value by providing management with a means for informed governance of enterprise transformation, thereby ensuring appropriate indicators and controls to steer the transformation of an enterprise into the desired direction.

Identifying, gathering and maintaining knowledge about the EA is a challenge emerging in the context of EA management, which is only addressed by isolated approaches [4], [5], [6]. However, we find no concrete description (i) to acquire and incorporate knowledge about the evolution of EA models, (ii) to operationalize their governance and finally (iii) to reason about these knowledge. More specifically, in this paper, we are interested in applying knowledge management to achieve a controlled EA model evolution with regards to security prop-erties which must be maintained throughout the EA evolution. We aim to ensure a secure governance when moving to future states in EA models. In doing so, we present a knowledge model that captures artifacts evolution dependencies. Artifacts changes will define inputs to reason about security properties compliance using Event Condition Action (ECA) rules. Note that we only limit the security compliance issue to access control management in EA.

The remainder of this paper is structured as follows. Section 2 presents the research background. The Problem is discussed in section 3, while Section 4 is dedicated to the approach including the knowledge model used and the reasoning model associated with it. Section 5 illustrates the usage of the solution approach. Section 6 presents related work and finally, section 7 concludes and outlines future work.

II. BACKGROUND

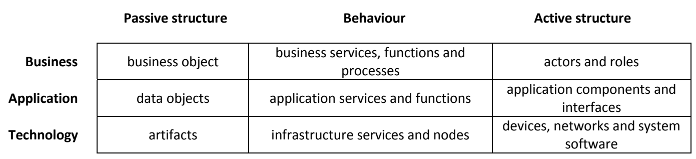
In this section, we present the main ingredients building our approach. We start with the enterprise context and focus on a specific language offering a holistic view for organizations when modeling service-oriented architecture: the ArchiMate modeling language. We remind also about security requirements in organizations and introduce a standard dealing with role-based access control management: the RBAC model. Both models are chosen based on our research focus and are explained step-by-step in the following sections.

A. Enterprise Architecture

Enterprise Architecture (EA) is generally considered to provide a good steering instrument to analyze the current state of the enterprise (As-is), identify and describe alternative future states (To-be), guard the cohesion and alignment between the different aspects of an enterprise. Architecture is a consistent whole of principles, methods and models that are used in the design and realization of organizational structure, business processes, information systems, and infrastructure [7].

The unambiguous specification and description of components and especially their relationships in architecture requires

978-1-4799-3392-1/13/$31.00 ©2013 IEEE



	Passive structure	Behaviour	Active structure
Business	business object	business services, functions and processes	actors and roles
Application	data objects	application services and functions	application components and interfaces
Technology	artifacts	infrastructure services and nodes	devices, networks and system software

Fig. 1. The Core Concepts of ArchiMate

a coherent architecture modeling language [7]. Current languages for modeling in the area of organizations, business processes, applications, and technology share a number of aspects on which they score low. For instance, the relation between domains is poorly defined, and the models created in different views are not further integrated. Besides, most languages miss the overall architectural vision and are confined to either the business or the application and technology sub domains [2].

B. The ArchiMate language

ArchiMate is an Open Group standard [8] for the modeling of enterprise architectures, emphasizing a holistic view of the enterprise. This means that architects can use ArchiMate to model, amongst others, an organization's products and services, how these products and services are realized and delivered by business processes, and how in turn these processes are supported by information systems and their underlying IT infrastructure. Such a holistic perspective on an enterprise helps to guide change processes [7], provides insight into cost structures, and more [9].

The ArchiMate language defines three main layers [2]:

- The Business layer offers products and services to external customers, which are realized in the organization by business processes (performed by business actors or roles).
- The Application layer supports the business layer with application services which are realized by (software) application components.
- The Technology layer offers infrastructure services (e.g., processing, storage, and communication services) needed to run applications, realized by computer and communication devices and system software.

The core concepts that are found in each layer of the language are depicted in Figure 1. A distinction is made between structural or static aspect and the behavioral or dynamic aspect. Behavioral concepts are assigned to structural concepts, to show who or what displays the behavior [2]. In addition to the active structural elements (business actors, application components and devices that display actual behavior), the language recognizes passive structural elements, i.e., the objects on which behavior is performed.

In [7], the authors have compared a selection of standards and languages (e.g., RM-ODP, UML, BPMN and ARIS) to ArchiMate, using three criteria for comparison: frameworks, architectural viewpoints and domains that are covered by each

http://www.opengroup.org/archimate/

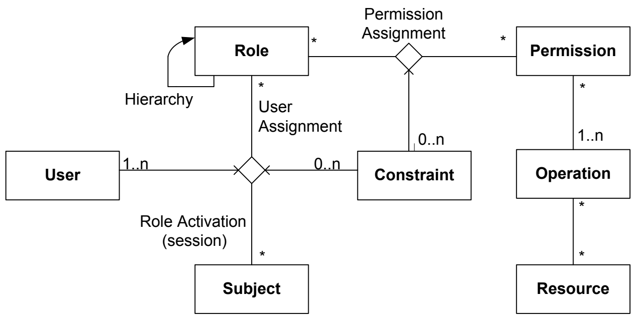


Fig. 2. The RBAC Model

language. According to their comparison, ArchiMate distinguishes itself from most other languages by its well-defined meta-model, concepts and, most importantly, its relations. The abstraction level of ArchiMate simplifies the construction of integrated models, where most languages appear to persuade architects to detailed modeling [7].

C. Access control management

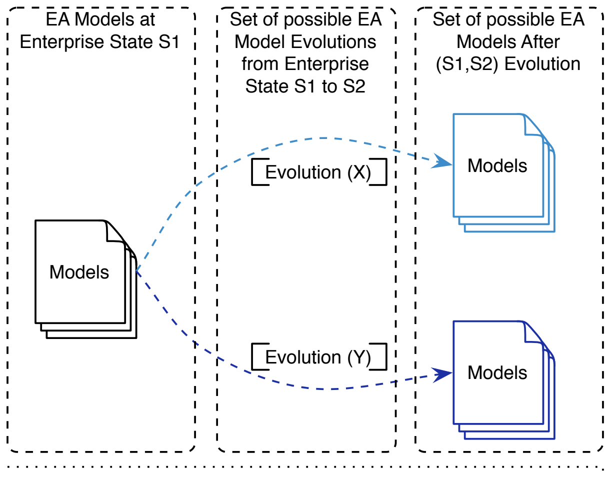
Access control is considered by most information systems security professionals to be the cornerstone of their security programs. The various features of physical, technical, and administrative access control mechanisms work together to construct the security architecture so important in the protection of an organizations critical and sensitive information assets [10].

A security policy defines the expected standard of security enforcement using access control within an enterprise at the organizational level. Primarily, a security policy addresses who has access to what resources, as well as how this access has to be regulated and managed [11]. In most organizations, a security policy must be applied to hundreds, if not thousands, of employees. To simplify security administration, many organizations define roles with which multiple individuals can be associated. The security policy of the organization then defines how permissions are to be associated with these roles.

Sandhu *et al.* presented the RBAC approach which is particularly effective when changes are made to the organizational security policy. The RBAC model needs only to be made to roles assignments, which are significantly fewer than individual assignments [12]. Figure 2 presents the RBAC model with a set of users, roles, permissions and constraints. A user defines a human being. A role is a job function or a job title. Permission is an approval of executing (i.e. operation) an object (i.e. resource). A session is a mapping between a user and possibly many roles where it is associated with a single user (so-called a subject) and each user may establish zero or more sessions. Constraints restrict permissions depending on contextual information such as a separation of dutie (SoD) [13].

D. Secure change in EA

An architecture is a blueprint that describes how an enterprise operates in terms of business processes and technology,



EA Models at Enterprise State S1
Set of possible EA Model Evolutions from Enterprise State S1 to S2
Set of possible EA Models After (S1,S2) Evolution
Models
Evolution (X)
Models
Evolution (Y)
Models

Fig. 3. EA Evolution - A Definition Based on Models

how it intends to operate in the future, and how it plans to transition to the future state [14]. The change from the current state of the enterprise (As-is) to its future states (To-be) has to guard the cohesion and alignment between the different aspects of an enterprise such as business processes and their ICT. One specific change concerns security and has an impact on the access control management of the enterprise. Dealing with organizational change, a security perspective focuses on dynamic access control management, describing the role of each actor and the scope (e.g., obligations, restrictions) of his action when accessing sensitive data. EA transformations in general and secure changes in particular are discussed in the rest of the paper.

III. Research problem

There exists different definitions and understandings of enterprise architecture transformation [14], [2], [3]. EA transformation may be related to business changes, new regulations, economic context, etc.

An evolution in EA is regarded as an evolution of the set of models describing an EA in a given enterprise state. The evolution of an EA model itself is just a set of changes to the artifacts contained in this EA model. Figure 3 illustrates this view on EA evolution. Hence, describing EA evolution enables us to reason on alternatives for EA evolutions and therefore decide upon alternatives or analyze potential evolutions from a given EA state.

This paper is set on a context where an EA is composed of a multitude of EA models each being concurrently edited by different modelers. These modelers have different responsibilities and may not be fully aware of the dependencies between the models they are currently working on an other models. This may lead to creating flaws and inconsistencies in EA models, when changes made on given EA models indirectly impact other EA models. A typical example of this is when a modeler X modifies the credentials or permissions assigned to a given role, in order to update a business process model Y, but oversees that this has an impact on another business

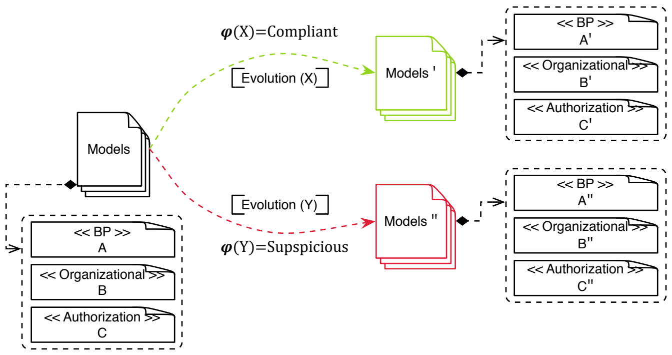


Fig. 4. EA Evolution Decision- Binary Model

process model Z where some security requirement is broken by this change.

Our objective is to assist EA managers in deciding which EA evolutions are fully compliant and which ones are not compliant or should be considered as suspicious and need be more thoroughly analyzed by an EA expert. This expert will then have the responsibility of taking a final decision whether the EA evolution should be committed to or reverted.

In short, we would like to implement the vision illustrated in Figure 4. In this model, an evolution from a set of models (A,B,C) to either (A',B',C') or (A",B",C") is taken as an input by a predicate $\varphi(Evolution) \mapsto \mathbb{B}$ which decides whether or not the evolution is suspicious or not. The $Evolution$ is defined as the couple of the states of the set of models impacted by the evolution before and after the evolution $((A,B,C);(A',B',C'))$.

Several questions are raised by this problem description, if we are to propose any solution.

- Assuming the intuition given of an EA evolution, how do we define it in a simple enough yet adequate way for our problem?
- What other information is needed by the $\varphi(Evolution)$ predicate to make it decidable?

IV. Approach

In this section, we propose a solution to be able to decide whether an EA evolution is suspicious or compliant. We do this by relying on three notions: change operation, dependency, and event-condition-action rules. In the following, we concisely explain how these three elements are combined to propose a solution to govern EA evolution.

First of all, we define a knowledge model as a set of *operations* of various kinds (adding, deleting, modifying, replacing elements) on artifacts occurring in an EA model. The second element are *dependencies* amongst EA artifacts and models. Dependencies are basically relations stating that any operation executed on an artifact respectively model shall have probable repercussions on another artifact respectively model. Therefore, such dependencies must be provided by the EA modeling experts as domain knowledge. we distinguish three types of dependencies: Artifact-to-Artifact (A2A), Artifact-to-Model (A2M), Model-to-Model (A2M). Each of these

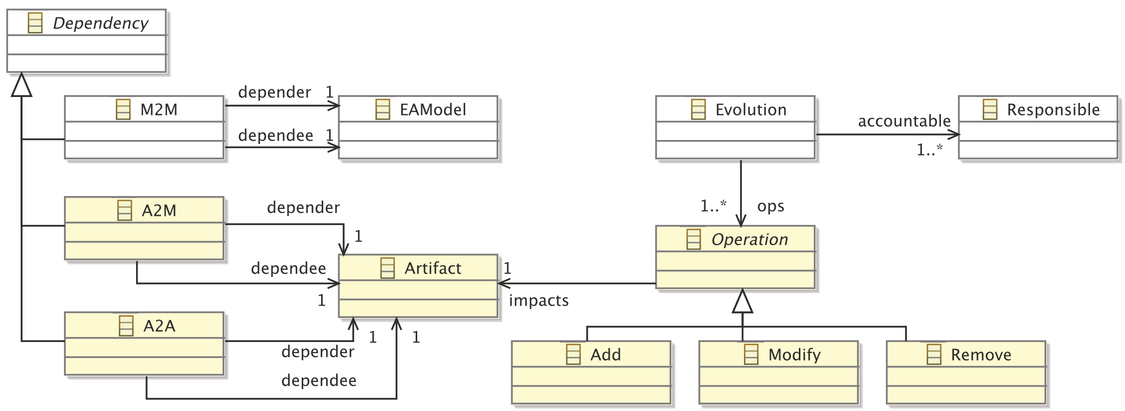


Fig. 5. Evolution Dependency Meta Model

dependencies is enacted differently but in the scope of this paper we limit ourselves to explaining and illustrating A2A dependencies (e.g., a dependency between two organizational roles).

These dependencies are what distinguishes our approach from standard EA modeling approaches. Several paradigms may be followed to extract, mine for or infer such dependencies, in order to assist the EA modelers in expressing them. For example, assuming we are only interested in security aspects of EA model evolution, we may define a trivial inference rule which extracts dependencies between organizational roles automatically from authorization models, based on the presence of any relation among these roles. The previous two solution elements are introduced in the following meta model, in Figure 5, whose instances can populate a knowledge base about simple model evolutions, as defined in the scope of this paper.

The final element of the approach is about solving EA governance problem. We propose the implementation of the $\varphi(Evolution)$ predicate. The definition of the latter predicate is strongly purpose- and application-dependent. We define as a single formalism the definition of Event-Condition-Action (ECA) rules, as a means of implementing a reactive mechanism to alert the enterprise architects about a suspicious evolution. The idea is as follows:

1) provided the occurrence of an event ε, which incidentally corresponds to the execution of a given type of change operation θ on an instance of a given artifact type α: ***ON Change REPLACE OF ArtifactType A***
2) and under the condition that there is a dependency between this artifact α and any other artifact α': ***IF*** $\exists$ ***A' / Dependency(A,A')***.
3) Then a given predicate should be evaluated on all artifacts α': ***THEN Check ConstraintPredicate***. Based on the evaluation of this predicate, the decision whether the evolution is suspicious or not is taken. Note that we did not set the number and type of the parameters of the predicate because this will depend on the application.

The predicate evaluated in the action part of the ECA rule is also application- and purpose-dependent. For instance, for checking whether the fulfillment of separation of duty constraints is being endangered by the evolution or not, and assuming the α artifact to be a role, one must define the predicated for checking the non-overlap in individuals carrying the same role, or the non-overlap in authorizations held by the roles α and α'. The approach is exemplified in the next section.

V. Access Control Governance in EA Evolution

In this section, we illustrate the usage of the solution approach we elicited in Section IV, by instantiating the problem for governing access control during EA evolution. In particular, we are interested in understanding how access control properties are impacted by EA model evolutions and raise alerts about suspicious evolutions where necessary. We first motivate the relevance of such an application of our approach then describe it.

EA transformations may be defined at different levels of an enterprise such as strategic, organizational or infrastructure. Here we focus on the organizational level where roles and actors are defined with their obligations and responsibilities. Obligations reflect responsibilities associated with certain privileges where privileges defines permissions based on the security policies [15], [16].

Access control enforcement is common when an enterprise needs to evolve due to new regulations and strategies. Such an enforcement implies new authorization policies with new access control specifications during EA evolution (To-be state). Let us assume that a new regulation enforced existing roles with additional permissions (see Figure 6).

Adding permissions implies new rights on business assets. This can be depicted as new operations on resources (i.e. business objects) in RBAC models (see Figure 2). In addition, access control constraints, such as separation of duties (SoD), have to be compliant with when EA evolve to the To-be state. In the context of EA models, this issue is of high relevance. In the example of the Archimate language, because of its inherent

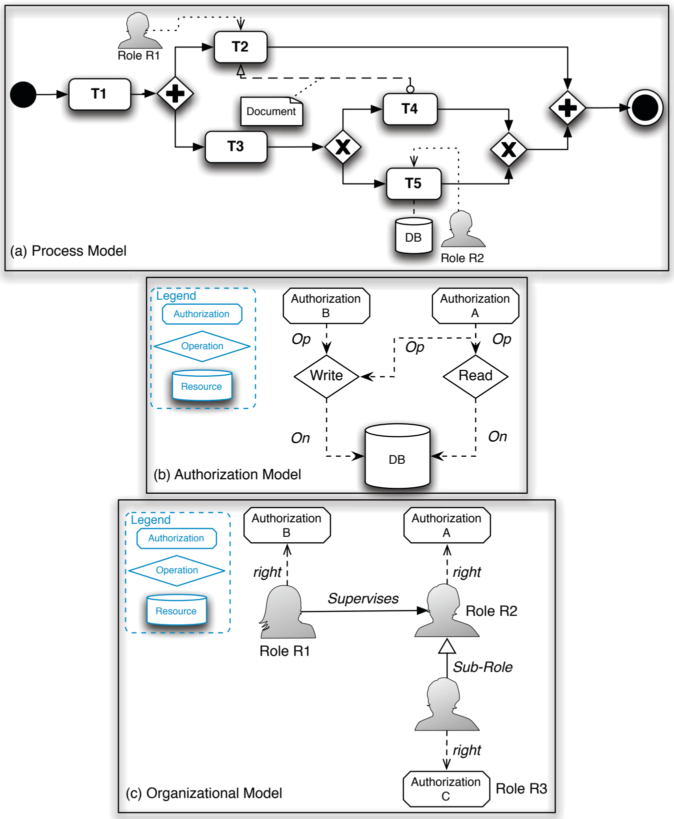


Fig. 6. EA Evolution Example

holistic nature, ArchiMate lacks specific guidelines for modeling an enterprise from a security perspective [17], [2], [8]. Knowing whether architecture evolution are being compliant requires defining the architecture's changes, establishing a method to check the dependency between As-is and To-be models in EA. Specifically, we have to make sure that security properties (e.g. SoD) remain unviolated when transforming EA models.

In the example depicted in Figure 6, we have a set of three arbitrary EA models: (a) is a business process model, (b) is an authorization model while (c) is an organizational model. The process model defines an arbitrary orchestrate of tasks. Task T2 is executed by a role R1. Task T5 is executed by a role R2 and accesses a database. Additionally, the authorization model states that both authorization A and B allow to write into the database. Finally, the organizational model states that role R1 supervises the work of role R2 (e.g., his hierarchical superior) while role R3 is a subtype of role R2.

Now let us assume that there is a separation of duty (SoD) constraint that states that the relation role-permission assignment (RPA) is defined as follows: $RPA(R1, R2) = \varnothing$, which simply means that roles R1 and R2 may not share any permissions. We implement this constraint from the perspective of tasks in the process model by stating that role R2 may not be allowed to execute both tasks T2 and T5, as role R1 is supposed to execute task T2.

In order to see the motivation behind the definition os such rules, we may consider the classical budget validation example. Assume that Task T5 accesses the accounting database to validate purchase orders, while task T2 requests such purchase orders. It becomes quite obvious that it is undesirable to have a purchaser (any role with a permission to request purchases) be allowed to validate purchase orders too.

This constraint may be expressed by the following *Role Task Assignment* predicate: $RTA(T2, T5)$!= $(R2)$. However, the organizational modeler is not aware of this constraint and for some reason, is forced to replace role R2 in the process model with another role. He checks the authorization model and finds that only authorization B permits write access to the database in the process model. Therefore, the modeler decides to select role R1 as the only alternative to role R2 according to both the organizational and authorization model.

This choice might seem trivial in this example but given hundreds of models, with even more complicated sets of authorizations and organizational dependencies, this design choice might be much harder to make. As a decision-support to the modeler, this is where our approach comes in.

Following the approach described in IV, we must define dependencies between the relevant modeling artifacts. Here, we may select as artifacts of interest the two tasks T2 and T5. Therefore, we define the following dependency to hold: ***Dependency***$(T2, T5)$.

The next step is to define the following ECA rule, in order to allow our modeling framework to react adequately to relevant events:

1) ***ON Change MODIFY of Task T***
2) ***IF*** $\exists$ ***Task T' / Dependency(T, T')***
3) ***THEN Check RTA(T,T')***

What will happen is as soon as a modification of a task T2 is operated in the process model, an event will activate the ECA rule we just defined. As there is a dependency between this task T2 and another task T5, the condition part of the ECA rule will hold. Finally, the application-dependent constraint RTA(T2,T5) will be checked (in the knowledge base which stores all the predicated such as dependencies) which does not hold. Therefore, this evolution shall be escalated to the EA modeler as suspicious and requires manual validation for the evolution to be committed.

Note that our rule definition is very broad and does not restrict the dependee task to occur in the same model as the depender task. This is a simplification worth considering in some application scenarios as it reduces the complexity of the approach by restraining the set of matched tasks in the condition part of the ECA rule.

VI. Related Work

There exist several IT Governance frameworks that have some focus on enterprise security. One of the most known frameworks is the Control Objectives for Information and related Technology (COBIT) [18] which is already in the version 5 and has specific internal IT related goals with security (e.g., security of information, processing infrastructure and applications). One standard that focused on IT security is the ISO/IEC 2700 [19] which has a practice guide addressing access control issues. In the meanwhile, our work focuses on mainly the dynamic aspect of secure governance in enterprise architecture (EA) and the primer results provide a secure knowledge model to reason about compliant evolution in EA models.

The Zachman framework [20] is an enterprise architecture framework for enterprise architecture, which provides a formal and highly structured way of viewing and defining an enterprise. The framework defines six different perspectives (Scope, Business model, Information system model, Technology model, Detailed description and Actual system) describing the information which is considered essential in an enterprise architecture. These perspectives should be described in six different ways (Data, Function, Network, People, Time and Purpose). The Open Group Architecture Framework (TOGAF) is a framework for enterprise architecture which provides a comprehensive approach for designing, planning, implementing, and governing an enterprise information architecture [8]. Nevertheless, the Zachman framework and TOGAF lacks of guidelines supporting organizational flexibility, specially, the access control enforcement in EA.

In [17], authors presented an approach that enhances the ArchiMate standard with a responsibility modeling language for access rights management. The idea consists of 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 where the main focus of the alignment is the definition and the assignment of the access rights needed by the employees according to business specification. We differentiate from this work in mainly two aspects: (i) Our approach is driven by a knowledge model and offers an overview of EA's state evolution based on its artifacts; and (ii) The identification of relevant concepts to the RBAC model helps to reason about security policies as part of EA regulations compliance.

VII. Conclusion

In this paper, we have proposed a knowledge model supporting access control management in enterprise architecture (EA). The goal is to secure governance for enterprises transformations. One specific transformation is that of EA models evolution. Models evolution have to be compliant with enterprises regulations such as security policies. In doing so, we have focused on access control enforcement and monitored by means of changes tracking using our EA knowledge model. The approach is then instantiated with an access control example.

We believe that a step in future research can be represented by adopting this model to the whole EA framework by extending the compliance features in ArchiMate. Another challenging topic will be the auditing and evaluation of security policies during the architecture development life-cycle in EA frameworks such as TOGAF.

References

[1] A. Mulholland, C. Thomas, P. Kurchina, and D. Woods, *Mashup Corporations - The End of Business as Usual*. Evolved Technologist Press, New York, New York, 2006.

[2] M. M. Lankhorst, *Enterprise Architecture at Work - Modelling, Communication and Analysis (4. ed.)*, ser. The Enterprise Engineering Series. Springer, 2013.

[3] M. Op 't Land, H. Proper, M. Waage, J. Cloo, and C. Steghuis, *Enterprise Architecture – Creating Value by Informed Governance*. Springer, Berlin, Germany, 2008.

[4] S. Aier, S. Buckl, U. Franke, B. Gleichauf, P. Johnson, P. Nrman, C. M. Schweda, J. Ullberg, C.-S. Gallen, and T. U. Mnchen, "A survival analysis of application life spans based on enterprise architecture models," in *In 3 rd International Workshop on Enterprise Modelling and Information Systems Architectures*, 2009, pp. 141–154.

[5] R. Fischer, S. Aier, and R. Winter, "A federated approach to enterprise architecture model maintenance," in *Enterprise Modelling and Information Systems Architectures - Concepts and Applications , Proceedings of the 2nd International Workshop on Enterprise Modelling and Information Systems Architectures (EMISA'07)*, 2007, pp. 9–22.

[6] Wegmann A., "The Systemic Enterprise Architecture Methodology (SEAM), Technical report, EPFL," 2002.

[7] H. Jonkers, M. Lankhorst, R. v. Buuren, S. Hoppenbrouwers, M. Bonsangue, and L. Van der Torre, "Concepts for Modeling Enterprise Architectures," *International Journal of Cooperative Information Systems*, vol. 13, no. 3, pp. 257–288, 2004.

[8] *The Open Group – TOGAF Version 9*. Van Haren Publishing, Zaltbommel, The Netherlands, 2009.

[9] M. Lankhorst, H. Proper, and H. Jonkers, "The Architecture of the ArchiMate Language," *Enterprise, Business-Process and Information Systems Modeling*, pp. 367–380, 2009.

[10] H. Tipton, *Information Security Management Handbook*, 5th ed., M. Krause, Ed. Boca Raton, FL, USA: CRC Press, Inc., 2003.

[11] C. P. Pfleeger and S. L. Pfleeger, *Security in Computing (4th Edition)*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2006.

[12] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models," *IEEE Computer*, vol. 29, no. 2, pp. 38–47, 1996.

[13] R. A. Botha and J. H. P. Eloff, "Separation of duties for access control enforcement in workflow environments," *IBM Systems Journal*, vol. 40, no. 3, pp. 666–682, 2001.

[14] United States Government Accountability Office, "Organizational transformation: Enterprise architecture value needs to be measured and reported," 2012.

[15] A. Schaad, "An extended analysis of delegating obligations," in *Research Directions in Data and Applications Security XVIII, IFIP TC11/WG 11.3 Eighteenth Annual Conference on Data and Applications Security, July 25-28, 2004, Sitges, Catalonia, Spain*. Kluwer, pp. 49–64.

[16] K. Gaaloul, "A Secure Framework for Dynamic Task Delegation in Workflow Management Systems. Ph.D. thesis, The University of Henri Poincaré, Nancy, France," 2010.

[17] C. Feltus, E. Dubois, E. Proper, I. Band, and M. Petit, "Enhancing the Archimate standard with a responsibility modeling language for access rights management," in *Proceedings of the Fifth International Conference on Security of Information and Networks*. New York, NY, USA: ACM, 2012, pp. 12–19.

[18] I. G. I. ITGI, *COBIT 4.1*. ISA, 2007.

[19] ISO/IEC, "ISO/IEC 27002: Information technology Security techniques Code of practice for information security management," 2005.

[20] J. Zachman, "A framework for information systems architecture," *IBM Systems Journal*, vol. 26, no. 3, 1987.