On Evolving Organizational Models without Loosing Control on Authorization Constraints in Web Service Orchestrations

Stefanie Rinderle-Ma, Maria Leitner
University of Vienna, Austria
Faculty of Computer Science
{stefanie.rinderle-ma, maria.leitner}@univie.ac.at

Abstract

Providing adequate access control is crucial for the proper execution of any Web Service (WS) orchestration. Typically, access rules and authorization constraints are defined for a WS orchestration and are resolved over an organizational model at runtime in order to find authorized users to perform orchestration tasks. As known from many practical studies, organizational models are frequently subject to change (e.g., outsourcing or restructuring). Although the effects of organizational changes on access rules have been investigated so far, their effects on authorization constraints remain still completely unclear, albeit violating authorization constraints might lead to severe problems such as security holes. In this paper, we systematically investigate the effects of organizational changes on authorization constraints and propose different strategies to cope with possible violations. We evaluate our results along the most common types of authorization constraints and discuss the impact of the selected implementation choice.

1 Introduction

Access control has emerged as de facto standard for ensuring authorized access in Process-oriented Information Systems (POIS). In WS orchestrations, authorized agents are determined at runtime based on access rules assigned to the orchestration tasks (cf. [8]). Thus, access rules constitute an interface between POIS and organizational models as depicted in Figure 1. However, for certain security measures such as dynamic Separation of Duties (dSoD), the specification of access rules might not be sufficient [15]. Assume, for example, that for two tasks A and B within a WS orchestration, we want to assign a set of authorized agents {`Smith`, `Sharp`}. This can be achieved by specifying a corresponding access rule. However, this does not enforce that the agent performing task A has to be different from the agent performing task B. Hence specifying additional authorization constraints becomes necessary, leading to a co-existence between organizational models, access rules, authorization constraints and WS orchestrations.

Existing approaches enable the specification and verification of authorization constraints in POIS at design and runtime [15, 3, 2, 1]. However, no approach has dealt with verification of authorization constraints during change time. As we know from various case studies [11], organizational structures tend to change quite frequently in practice. Examples include outsourcing, merging of departments, or adding new hierarchical layers within enterprises. In our previous work [11, 9] we investigated the question of how such organizational changes can be reflected within the underlying information systems (Figure 1). More precisely, we showed how access rules are affected by organizational changes and how they can be adapted in order to avoid, for example, security holes or quality problems in the sequel.

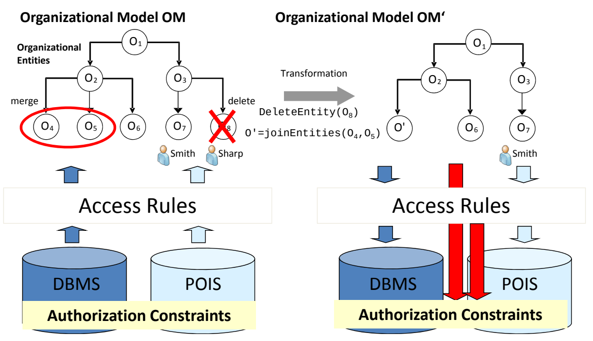


Figure 1. Access Control in POIS

Obviously, the effects of organizational changes on access rules have to be controlled. What about their effects on authorization constraints? Consider the dSoD as described above. One organizational change that affects the dSoD would be to lay agent `Smith` off. As a consequence, the

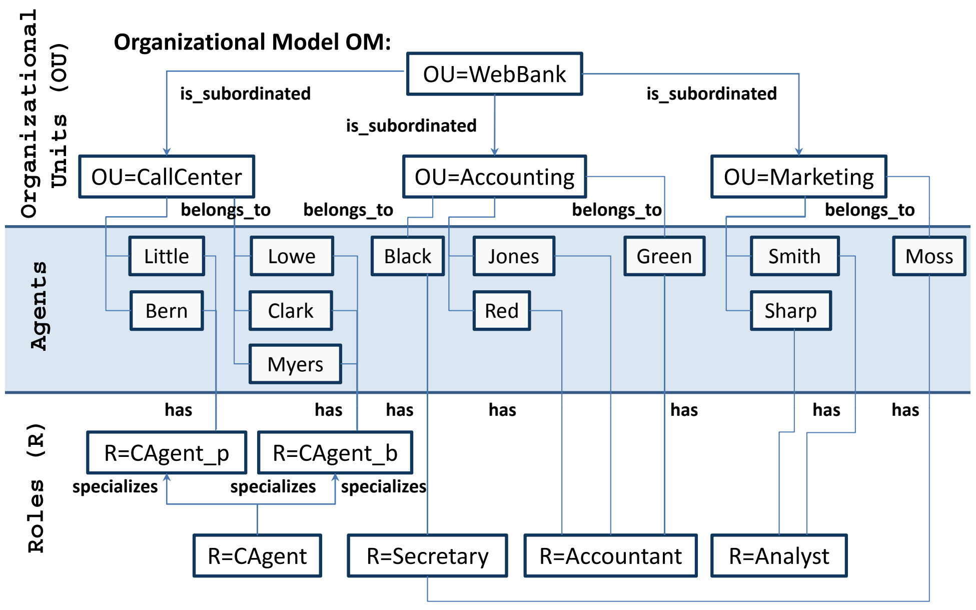


Figure 2. Organizational Model for Online Bank

separation of duties for tasks A and B would fail at runtime. Taking the approach presented in [1], this violation could be detected at runtime, as soon as task B is started. By contrast, we argue that the driving force to detect such violations lies within the organizational change and not within orchestration execution. Specifically, we argue that the violation can be detected as soon as the organizational change happens, very likely before B will be started. This gives the opportunity to pro-actively find solutions for the violation.

In addition, there are violations that cannot be dealt with by runtime checks. This refers to all organizational changes that cause inconsistencies within the associated access rules (e.g., orphaned references) and thus lead to an adaptation of the access rules in the sequel [11]. As we will show in this paper, such *indirect effects* might require an adaptation of authorization constraints as well in order to maintain a controlled execution of the WS orchestrations.

Altogether, in this paper, we introduce a comprehensive definition of *organizational compliance*, spanning WS orchestrations, organizational models, access rules, and authorization constraints. Furthermore, we show how direct effects of organizational changes can be detected and proactively handled. We also show how organizational changes indirectly affect authorization constraints after access rule adaptations and how the affected authorization constraints can be adapted accordingly. We evaluate our findings based on the most common authorization constraints [12]. Furthermore, we discuss different implementation choices for access rules and authorization constraints.

Sections 2 introduces our running example and necessary background information. Section 3 addresses direct effects of organizational changes on authorization constraints. We show how authorization constraints can be adapted after organizational changes in Section 4. Section 5 provides an evaluation and presents implementation choices. Section 6 discusses related work and Section 7 closes with a summary and outlook.

2 Access Control for WS Orchestrations

In this section, we introduce our running example together with background information necessary for understanding the remainder of the paper. In general, for WS orchestrations, *access rules* are specified for each task based on *organizational models* in order to restrict access to authorized agents. Commonly, an organizational model OM is defined based on an organizational meta model OMM. Figure 3 depicts the meta model used in this paper which reflects Role-Based Access Control mechanisms, i.e., enabling the use of organizational entities `Roles`, `Organizational Units`, and `Agents` within organizational models. Consider organizational model OM depicted in Figure 2: `Roles` characterize the authorization to work on a certain task (e.g., `Supervisor`). Examples for `Organizational Units` comprise `WebBank` and `CallCenter`. `Agents` denote human or non-human resources, e.g., `Miller` and `Bern`. Within organizational models, entities are related to each other. Agents have roles and belong to organizational units. Roles and organizational units can be hierarchically structured. More precisely, roles can be specialized into sub roles and organizational units can be subordinated to other organizational units.

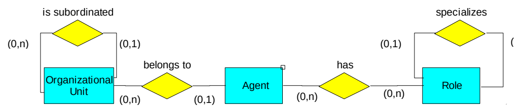
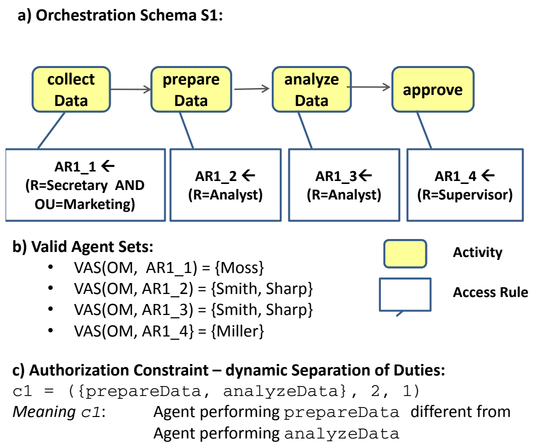


Figure 3. Organizational Meta Model

The WS orchestration in Figure 4 consists of four activities that are sequentially ordered. For each activity, an access rule is specified that defines the set of authorized agents for this activity. For activity `collect Data`, for example, access rule `AR1_1` specifies that only agents having role `Secretary` and belonging to organizational unit `Marketing` are authorized to perform this activity. According to [11], access rules are logical expressions that consist of elementary access rules

```
EAR ≡ (EAR1 ⟵ Role = r)
  | (EAR2 ⟵ Organizational Unit = o)
  | (EAR3 ⟵ Role+ = r)
  | (EAR4 ⟵ Organizational Unit+ = o)
```

where the + notation refers to the entity and all its subordinated entities. Elementary access rules can be combined using logical connectors `AND`, `OR`, and `NOT`. `AR1_1` is an example for an access rule consisting of two elementary access rules combined by `AND` (cf. Figure 4).



b) Valid Agent Sets:

- VAS(OM, AR1_1) = {Moss}
- VAS(OM, AR1_2) = {Smith, Sharp}
- VAS(OM, AR1_3) = {Smith, Sharp}
- VAS(OM, AR1_4} = {Miller}

c) Authorization Constraint – dynamic Separation of Duties:

`c1 = ({prepareData, analyzeData}, 2, 1)`

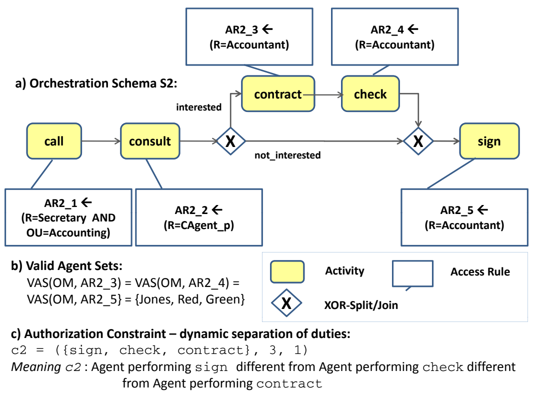
Meaning c1: Agent performing `prepareData` different from Agent performing `analyzeData`

Figure 4. Marketing Process (BPMN Notation)

Access rules are defined for WS orchestration schemas at design time. At runtime, *orchestration instances* are started and executed based on orchestration schemas and the specified access rules are resolved over the underlying organizational model. Specifically, for orchestration instance I and for task t, assigned access rule $\mathtt{AR}_t$ is resolved over underlying organizational model OM into set of valid agents $\mathtt{VAS(OM,\ AR}_t\mathtt{)}$. In Figure 4, access rule `AR1_2` is assigned to task `prepare Data`. The agents having role `Analyst` based on OM are `Smith` and `Sharp` and thus `VAS(OM, AR1_2) =` $\{$`Smith, Sharp`$\}$. As soon as `prepare Data` is activated, the respective work item is offered to `Smith` and `Sharp` in their worklists. If `Smith` then selects `prepare Data` from her worklist, the corresponding work list entry for `Sharp` is removed from his worklist. We denote the agent who selected and worked on task t in an orchestration instance I as `Performer(t,I)` of t in I.

Obviously, it is not possible to resolve certain authorization constraints based on access rules [2]. Consider activities `prepare Data` and `analyze Data`, for which their valid agent sets both comprise agents `Smith` and `Sharp`. Based on associated access rules `AR1_2` and `AR1_3` we cannot express that these activities should be controlled by a four eye principle (dSoD), but need to impose additional authorization constraint $c1$. In order to define authorization constraint c, we adopt the notion proposed in [15].

$c := (T, n, m)$ *where* T *denotes the set of orchestration tasks* c *refers to,* $n \in \mathbb{N}$ *denotes the minimal number of agents associated with* c *and* $m \in \mathbb{N}$ *denotes the maximum number of tasks an associated agent can work on.*



b) Valid Agent Sets:

VAS(OM, AR2_3) = VAS(OM, AR2_4) = VAS(OM, AR2_5} = {Jones, Red, Green}

c) Authorization Constraint – dynamic separation of duties:

`c2 = ({sign, check, contract}, 3, 1)`

Meaning c2 : Agent performing `sign` different from Agent performing `check` different from Agent performing `contract`

Figure 5. Contract Process (BPMN Notation)

In Figure 5, for example, authorization constraint $c2$ defines that for activities `contract`, `check`, and `sign`, at least three agents have to be assigned, where each agent is authorized to perform exactly one of these activities. In other words, $c2$ defines a six eyes principle (dSoD) in this example.

As many practical examples show, changes of the organizational structures are quite common in practice [11]. Table 1 summarizes a selection of change operations on organizational models as introduced in [9]. By using these change operations we can, for example, create a new role or delete an existing organizational unit. The effects of organizational change operations on access rules have been formally defined based on the effects on the corresponding valid agent sets [9].

Table 1. *Change Operations on Organizational Models (Selection)*

Change Δ transforms organizational model OM into org. model OM':
`CreateEntitiy(OM, eId, eType) = OM'` adds a new organizational entity id of entity type eType to OM
`DeleteEntity(OM, e) = OM'` deletes entity e from OM
`CreateRelation(OM, e1, e2, relType) = OM')` adds a new relation between org. entities e1 and e2 of type relType to OM
`DeleteRelation(OM, relation) = OM'` deletes relation relation from OM
`ReAssignRelation(OM, r, e, eNew) = OM'` reassigns relation r entity e in OM to entity eNew in OM'
`JoinEntities(OM, e1, e2, nId) = OM'` merges two entities e1 and e2 into new entity nId of same entity type
`SplitEntity(OM, eOld, e1, e2) = OM'` splits organizational entity eOld into new entities e1 and e2 of same type

Consider Figure 6 where organizational change Δ has been applied to OM resulting in new model OM'. Δ consists of several `DeleteRelation(...)` operations and one `DeleteEntityRelation(...)` in order to remove agent `Sharp`. Furthermore, entity `Accountant` is split into two new entities `SeniorAcc` and `JuniorAcc`.

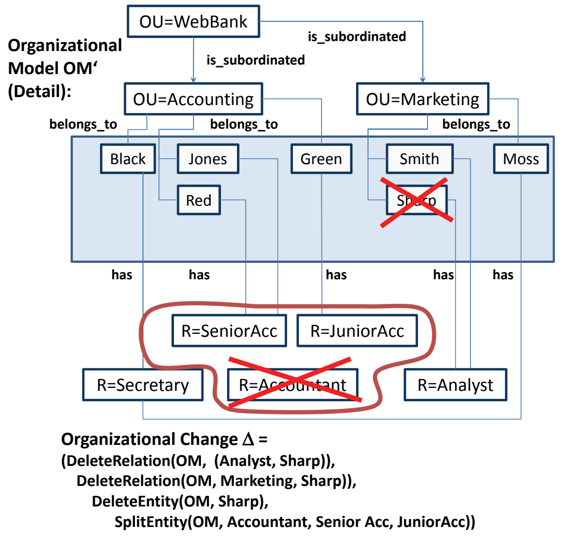


Figure 6. Evolution of Organizational Model

3 Organizational Changes Directly Affecting Organizational Compliance

In this section, we address the question of *organizational compliance* in general, and specifically after changes of the underlying organizational model.

3.1 Organizational Compliance

Access rules are the de facto standard to establish an interface between organizational models and WS orchestrations in existing POIS. Authorization constraints are defined on top of access rules in order to impose dynamic rules on valid agent sets. Hence, in order to come up with a comprehensive view on proper access control for WS orchestrations, we have to consider all aspects – organizational models, access rules, authorization constraints, and WS orchestration within one verifiable notion. Thus, we define *organizational compliance* in the following which establishes the connection between all these aspects.

Definition 1 (Compliance of Organizational Models) *Let OM $\in \mathcal{OM}$ be an organizational model and C be a set of authorization constraints defined over OM (for a set of WS orchestrations $\mathcal{S}$). Then:*

OM is compliant with C for S $\in \mathcal{S}$, iff $\forall$ c $\in$ C:

c = (T, n, m) is valid over OM for S, i.e.,

$\forall\, t \in T$: $|\bigcup_t \texttt{VAS(OM, AR}_t\texttt{)}| \geq n$

where $\texttt{VAS(OM, AR}_t\texttt{)}$ *denotes the set of valid agents for access rule* $\texttt{AR}_t$ *over* OM.

Consider authorization constraint $c2$ = ({`sign, check, contract`}, 3, 1) as depicted in Figure 5. Access rules `AR2_3, AR2_4, AR2_5` all have the same valid agent set `VAS(OM, AR2_i)` = {`Jones, Red, Green`} (i = 3, 4, 5). Thus, $|\bigcup_i \texttt{VAS(OM, AR}_i\texttt{)}| = 3$ holds and consequently $c2$ is valid for S over OM.

When looking closer at Definition 1 and the above mentioned example, we see in task set of $c2$ for all tasks that the access rules and consequently the valid agent sets are equal. Hence, the union over all valid agent sets trivially also equals the valid agent set of each of the access rules. An implicit assumption is that we only check for organizational compliance of authorization constraints that are imposed when access rules cannot express dynamic restricitions on the valid agent sets anymore [8]. An example is dSoD. By contrast, static Separation of Duties (sSoD), can be expressed on basis of access rules. For two tasks, for example, we define different access rules for each of the tasks where the corresponding valid agent sets are disjoint.

The interesting case in connection with Definition 1 is the one where dSoD is imposed for a set of tasks for which the access rules are not necessarilty describing the same valid agent sets, but the intersection of the valid agent set is non-empty. This is particularly true when agents have more than one role or belong to several organizational units. Consider the example depicted in Figure 7 where agent Jack has roles R1 and R2. To check whether $OM*$ is compliant with $c*$, it is not sufficient to consider the valid agent set of one of the access rules, since they overlap, but are not the same. In fact, by considering the union of the valid agent sets of all access rules, we can decide that $c*$ is not compliant with $OM*$.

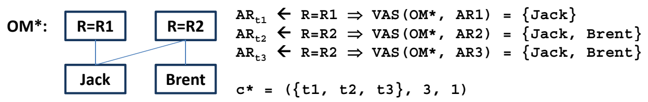

OM*:
R=R1
R=R2
Jack
Brent
AR_{t1} ← R=R1 ⇒ VAS(OM*, AR1) = {Jack}
AR_{t2} ← R=R2 ⇒ VAS(OM*, AR2) = {Jack, Brent}
AR_{t3} ← R=R2 ⇒ VAS(OM*, AR3) = {Jack, Brent}
c* = ({t1, t2, t3}, 3, 1)

Figure 7. Overlapping Valid Agents Sets

Organizational compliance can be checked at design time by comparing the specification of each authorization constraint with the set of valid agents for the associated access rule in a WS orchestration S. If no organizational change is conducted, it is guaranteed that the authorization constraints can be fulfilled at runtime. As pointed out in [15], it has to be additionally ensured that authorization constraints are enforced, e.g., based on checking their history.

3.2 Direct Organizational Change Effects

So far, existing approaches have focused on ensuring compliance of organizational models at design and runtime, but not at change time. However, as discussed in [11], changes of the organizational model happen quite often in practice and might lead to changed valid agent sets. As consequence, organizational changes might not only harm access rules, but also organizational compliance. The latter aspect is addressed within this paper for the first time.

As specified in Definition 1, organizational compliance is violated if the cardinality of the united valid agent sets of the associated access rules decreases under a certain threshold. Thus, organizational changes are potentially critical if they lead to a reduction of the valid agent sets of access rule assigned to tasks which are also subject to authorization constraints. Consider the organizational change that deletes agent Sharp from OM as depicted in Figure 4. This change leads to a reduction of the valid agent set of access rules AR1_2 and AR1_3. In turn, since AR1_2 and AR1_3 are associated with tasks prepare Data and analyze Data that are subject to authorization constraint $c1$ this organizational change might be potentially critical. Checking Definition 1, we see that for $c2$ the threshold of $n = 2$ is not reached anymore, since $|\bigcup_{i=2,3} \mathtt{VAS(AR\_i, OM)}| = 1 < 2$.

In [9], we have elaborated on how valid agent sets change for all different kinds of organizational change operations and access rules. As the above example might seem straightforward, there are more interesting cases. Think, for example, of a change operation which reassigns agent Sharp from having role Analyst to having role Accountant. Then the effects on the associated valid agent sets are not that obvious. However, taking the results form [9], it can be precisely determined, how the valid agent sets are affected by organizational changes. Particularly, we focus on *reduction* of valid agent sets since this is the critical case with respect to organizational compliance. Contrary to reduction, expanding valid agent sets is not harming organizational compliance at all. Think, for example, of a change that inserts a new agent Parker into OM by assigning Parker to role Analyst. Then, the valid agent set for tasks prepare Data and analyze Data becomes bigger and the separation of duty constraint $c2$ can still be fulfilled.

Applying the findings presented in [9] to control effects of organizational changes on organizational compliance, we can conclude the following results: creation and deletion of entities is always accompanied by associated CreateRelation(...), ReassignRelation(...), or DeleteRelation(...) operations (cf. Table 1), since new entities have to be embedded into the overall context of the organizational model and entities to be deleted have to be resolved from the organizational model first. In Figure 4, not the final deletion of agent Sharp is the critical change operation, but the precedent deletion of the relation connecting Sharp and role Analyst, i.e., DeleteRelation(OM, (Analyst, Sharp)). As soon as Sharp is not related to Analyst anymore, the valid agent sets of access rules AR1_2 and AR1_3 are reduced. Hence, in order to analyze effects on organizational compliance, it is sufficient to investigate the effects of changes on relations within organizational models. As shown in [9], for access rules that do not contain any negation, the application of CreateRelation(...) is uncritical with respect to a reduction of the valid agent sets. By contrast, DeleteRelation(...) and ReassignRelation(...) operations might reduce valid agent sets of affected access rules and consequently the affect authorization constraints as well. Thus if any DeleteRelation(...) or ReassignRelation(...) operation is applied to an organizational model OM that is basis to some authorization constraint c, organizational compliance of OM with respect to c must be re-evaluated based on Definition 1. The re-evaluation can be automatically done by the system, resulting in reports on which organizational changes have caused compliance violations. This is particularly helpful for complex organizational models with hundreds or thousands of access rules and authorization constraints.

4 Indirect Compliance Violations via Access Rule Adaptations

As discussed in Section 3, certain organizational change operations might have direct effects on organizational compliance by reducing the set of valid agents of associated access rules. One example is `DeleteRelation(OM, (Analyst, Sharp))` in Figure 6, directly affecting organizational compliance of OM with $c1$ for WS orchestration $S1$ (cf. Figure 4).

By contrast, organizational change `SplitEntity(OM, Accountant, SeniorAcc, JuniorAcc)` (cf. Figure 6) does not directly affect authorization constraint $c2$ for orchestration $S2$ as depicted in Figure 5. Specifically, after changing organizational model OM, the effects on $c2$ cannot be directly determined. After splitting role `Accountant` into two new roles `JuniorAcc` and `SeniorAcc` access rules `AR2_3`, `AR2_4`, and `AR2_5` cannot be resolved over changed organizational model OM' anymore. Reason is that for all access rules referring to role `Accountant`, an orphaned reference is resulting on changed organizational model OM'. Hence, no statement on organizational compliance is possible before the access rules have been adapted. Note that adaptation of affected access rules is indispensable in order to avoid any undesired behavior of the system such as offering activities to non-authorized agents. In [10, 11], we have proposed the following adaptation strategies (cf. Table 1) for access rules after organizational changes `JoinEntities(...)` and `SplitEntity(...)`:[1]

- `JoinEntities(OM, e1, e2, nId) = OM'`:
  $\forall$ access rules `AR` with
    `AR` $\leftarrow$ `eType=e1` or `AR` $\leftarrow$ `eType=e2`:
      replace `AR` by `AR'` $\leftarrow$ `eType = nId`
        `(e1, e2, and nId of type eType)`
- `SplitEntity(OM, eOld, e1, e2) = OM'`:
  $\forall$ access rules `AR` with `AR` $\leftarrow$ `eType=eOld`:
    replace `AR` by `AR1` $\leftarrow$ `eType=e1`
      or `AR2` $\leftarrow$ `eType=e2`
        where `eType` $\in$ {`Agent, Role, OrgUnit`}

Please note that we abstract from negated terms within access rules. Picking the adaptation strategies for `SplitEntity(...)` operation, the adaptation of access rules `AR2_3`, `AR2_4`, and `AR2_5` states that the adapted access rules will refer to either role `JuniorAcc` or role `SeniorAcc` instead of referring to role `Accountant`. Unfolding all possible combinations, adaptation could theoretically result in 8 different scenarios (all activities are assigned to agents having role `SeniorAcc`, all are assigned to one having role `JuniorAcc` and so on). Let us assume that based on functional requirements, for task `contract` the assigned role should be adapted to `JuniorAcc` and for task `sign` to `SeniorAcc`. Hence dSoD between tasks `contract` and `sign` is not required anymore, since assigning these tasks to different roles already results in static separation of duties. After adapting the access rules accordingly, `VAS(OM', AR2_3')` = {`Green`} is disjoint to `VAS(OM', AR2_5')` = {`Jones, Red`}. Hence, task `contract` will be always offered to a different agent than task `sign`.

As a consequence, authorization constraint $c2$ must be adapted . The question is now, to which role task `check` is to be assigned. If it is assigned to role `JuniorAcc`, the adapted authorization constraint $c2'$ should regulate dSoD between `check` and `contract`. If `check` is assigned to role `SeniorAcc`, it dSoD between `check` and `sign` is required within $c2''$, i.e.:

- `AR2_4'` $\longleftarrow$ `R=JuniorAcc` $\Longrightarrow$
  c2' = ({`contract, check`}, 2, 1)
- `AR2_4''` $\longleftarrow$ `R=SeniorAcc` $\Longrightarrow$
  c2" = ({`sign, check`}, 2, 1)

Checking now compliance of both adapted authorization constraints $c2'$ and $c2''$ over changed organizational model OM' (cf. Figure 6), we see that OM' is not compliant with $c2'$, but it is compliant with $c2''$. Thus, when there are no functional or other requirements, it would be decided to adapt $c2$ to $c2''$ instead of $c2'$, since adaptation to $c2''$ maintains organizational compliance without any further actions. This information can be of valuable help when adapting authorization constraints.

5 Evaluation

In this section, we discuss the applicability of the above mentioned strategies for the most prominent types of authorization constraints as discussed in [12]. We also provide considerations on implementation choices for authorization in WS orchestrations.

5.1 Authorization Constraint Types

DSoD and Retain Familiar (RF) both require a certain number of agents to work on the assigned tasks. Retain familiar specifically requires at least one agent to work on a set of tasks. Hence, organizational change operations that possibly reduce valid agent sets are possibly critical for the fulfillment of dSoD and RF constraints and the checking mechanisms provided in Section 3 can be applied to detect violations as soon as possible. RF constraints on top of access rules are imposed if the underlying access rules

[1]For other organizational change operations such as `CreateEntity(...)` no access rule adaptations become necessary. Note that for `DeleteEntity(...)` (semi-)automatic adaptation strategies are hard to specify without knowing the application context.

describe different valid agent sets. Hence, in case organizational changes require adaptation of access rules, RF constraints might be subject to adaptation as well. Altogether, the findings of this paper can be directly applied to dSoD and RF that constitute the most commonly applied authorization constraints in practice.

Cardinality constraints (CC) require that a certain minimum or maximum number of organizational entities is assigned to an orchestration model or to an activity. One example is the following constraint: "There must be at least three sub roles of role `Doctor` assigned to the chemotherapy treatment" (adapted from [1]). This CC cannot be formalized using the notion proposed by [15] since it refers to a minimum number of `Roles` instead of `Agents`. Hence, we adapt the notion proposed in [15] to

`c = (T, eType=eID,` $[< | \leq | = | \geq | >]$`, n)`

with

- T denotes the task set, c refers to. If T is the complete task set of a WS orchestration S, c has to apply for S
- entity type `eType` $\in$ `{Agent, Role, OrgUnit}`
- n denotes the required number of assigned entities of type `eType`.

Then, the cardinality constraint stated above can be formalized as `c= (T, Role=Doctor(+),` $\geq$`, 3)` where T corresponds to the task set of S. Recall that `(+)` refers to the sub roles of role `Doctor` in corresponding organizational model OM. Obviously, if OM is changed by deleting sub roles of role `Doctor`, cardinality constraint `c` might be potentially violated. Adding more sub roles is not critical. Even if a cardinality constraint possesses maximum boundaries, adding organizational entities will not violate these boundaries.

Consequently, if an organizational model is changed by deleting an organizational entity of type `eType`, the set of imposed cardinality constraints must be checked for referring to organizational entities of this type. In case a violation is detected, the question is how to deal with the violation. One possibility is to adjust the cardinality constraint to the new organization model. Assume that for the above example, after deleting one of the sub roles of `Doctor`, we heal the resulting violation of `c` by adjusting `c` to `c' = (T, Role=Doctor(+),` $\geq$`, 2)`. However, as a consequence, the information is lost that we originally wanted 3 sub roles to be assigned to WS orchestration S. Thus, adapting the authorization constraint must not be done automatically, but the system should report the violation to the orchestration designer, possibly together with a suggestion for adapting the authorization constraints.

5.2 Impact of Implementation Choice

So far we have assumed that authorization constraints are stated on top of access rules and organizational models. However, authorization constraints can be also modeled and enforced as dependent access rules defined at task level (so called *task level implementation*). As an example take Figure 4, where task `analyze Data` must not be executed by the same analyst as for `prepare Data`. We can formulate the resulting dSoD constraint `c1` instead at task level as follows:

`Role = (Analyst) AND`
`(Performer(prepareData,I)` $\neq$ `Performer(analyzeData,I))`

for some orchestration instance I. Note that a performer is an agent who selects and works on task t in an orchestration instance I.

Interestingly, it appears that there is a difference in implementation between scientific research and commercial solutions. We could only discover constraint base implementations in research, i.e. Bertino et al. [1] and SecureFlow [6] whereas commercial and open source applications are implementing authorization constraints at task level (e.g., by using individual task definitions and programmatic extensions) [12]. Specifically, WebSphere MQ, FLOWer, iPlanet, and YAWL are realizing separation of duties by task level dependencies or indirectly by user access rights (i.e. COSA) [12, 13]. Most commercial applications have implemented the four eyes principle (cf. role `Analyst` in Figure 4). Few application systems (i.e. Websphere MQ) are able to apply separation of duties on more than two corresponding tasks (cf. role `Accountant` in Figure 5).

In summary, organizational changes do affect authorization constraints whether implemented on top of the access rules or implemented within access rules at task level. In the latter case, the findings presented in [9] have to be extended to such dependent access rules.

6 Related Work

How to specify and enforce authorization constraints within workflow systems is shown in [1]. Specifically, authorization constraints are checked for each task execution. Furthermore, a planning algorithm suggests possible assignment adaptations for violated constraints. The workflow authorization framework presented in [3] uses the concept of roles, organizational levels and authorization constraints. The enforcement of authorization constraints is executed by Event-Condition-Action rules. An extension for BPMN has been proposed to support workflow resource patterns [15]. We used the constraint formalization as presented in the paper and adapted it for cardinality constraints in our paper. Dynamics in organizations are examined in [14] where they identify top-down (i.e. structural changes) and bottom-up dynamics (i.e. autonomous agents). Whilst their approach focuses on the adoption of organizational rules into the decision making process of agents (bottom-up), we consider evolving organizational models and how they affect existing authorization constraints (top-down). Furthermore, this has been adapted to web services [5]. A methodology for modeling evolving cross-organizational business processes is presented in [4] considering among others structural changes mainly delegation or assignment of interaction requirements to new or existing agents.

Altogether, the challenge of integrating and enforcing authorization in POIS has been addressed by different approaches. In our previous work, we have elaborated on the effects of organizational changes on access rules [9] as well as access rule changes themselves [10, 11]. However, to our best knowledge, the question of how organizational change affects authorization constraints as presented in this paper has not been addressed by any other approach so far.

7 Summary and Outlook

In this paper, we introduced organizational compliance as central concept taking into consideration the coexistence of organizational models, access rules, authorization constraints, and WS orchestrations. We showed how adaptations of the organizational model directly affect organizational compliance, for example, when agents are reassigned to different roles. Furthermore, we elaborated on how indirect effects of organizational models might lead to invalidity of associated authorization constraints. Specifically organizational changes might require an adaptation of access rules and subsequently necessitate adaptations of authorization constraints as well. We showed that without taking adequate actions, access control for WS orchestrations might be at stake after organizational changes. The findings were evaluated based on common types of authorization constraints and the impact of different implementation choices (separating authorization and access control versus integrated task-dependent implementation) was discussed.

In future work, we plan to integrate an adaptive organizational model component with our SeaFlows framework for the definition and verification of compliance constraints in POIS [7]. In addition, we want to compare the separated implementation of access control and authorization with task-dependent implementation. Furthermore, we aim at extending our findings to process choreographies describing collaborations between different business partners.

Acknowledgements

The work presented in this paper has been partly conducted within the OCIS project which is funded by the German Research Foundation (DFG).

References

[1] E. Bertino, E. Ferrari, and V. Alturi. The specification and enforcement of authorization constraints in WFMS. *ACM Transactions on Information and System Security*, 2(1):65–104, 1999.

[2] R. A. Botha and J. H. P. Eloff. Separation of duties for access control enforcement in workflow environments. *IBM Systems Journal*, 40(3):666–682, 2001.

[3] F. Casati, S. Castano, and M. Fugini. Managing workflow authorization constraints through active database technology. *Information Systems Frontiers*, 3(3):319–338, 2001.

[4] N. Desai, A. K. Chopra, and M. P. Singh. Amoeba: A methodology for modeling and evolving cross-organizational business processes. *ACM Trans. Softw. Eng. Methodol.*, 19(2):1–45, 2009.

[5] F. Dignum, V. Dignum, J. Padget, and J. Vazquez-Salceda. Organizing web services to develop dynamic, flexible, distributed systems. In *Proc. Int'l Conf. Information Integration and Web-based Applications & Services*, pages 225–234, 2009.

[6] W. Huang and V. Atluri. SecureFlow: a secure web-enabled workflow management system. In *Proc. ACM Workshop on Role-based access control*, pages 83–94, 1999.

[7] L. Ly, S. Rinderle-Ma, and P. Dadam. Design and verification of instantiable compliance rule graphs in Process-Aware information systems. In *Proc. Int'l Conf. on Advanced Systems Engineering*, pages 9–23, 2010.

[8] J. Mendling, K. Ploesser, and M. Strembeck. Specifying separation of duty constraints in BPEL4People processes. In Springer, editor, *Proc. Int'l Conf. Business Information Systems (BIS 2008)*, volume 7 of *LNBIP*, pages 273–284, 2008.

[9] S. Rinderle and M. Reichert. A formal framework for adaptive access control models. *Journal on Data Semantics*, (IX):82–112, 2007.

[10] S. Rinderle-Ma and M. Reichert. Managing the life cycle of access rules in CEOSIS. In *Proc. Int'l Enterprise Computing Conference*, pages 257–266, 2008.

[11] S. Rinderle-Ma and M. Reichert. Comprehensive life cycle support for access rules in information systems: The CEOSIS project. *Enterprise Information Syst.*, 3(3):219–251, 2009.

[12] N. Russell, W. M. van der Aalst, A. H. ter Hofstede, and D. Edmond. Workflow resource patterns: Identification, representation and tool support. In *Proc. Int'l Conf. Advanced Information Systems Engineering*, pages 216–232. Springer, 2005.

[13] A. H. M. ter Hofstede, W. M. P. van der Aalst, and M. Adams. *Modern Business Process Automation*. Springer, Nov. 2009.

[14] B. van der Vecht, F. Dignum, J. Meyer, and V. Dignum. Organizations and autonomous agents: Bottom-Up dynamics of coordination mechanisms. In *Coordination, Organizations, Institutions and Norms in Agent Systems IV*, pages 17–32. Springer, 2009.

[15] C. Wolter and A. Schaad. Modeling of Task-Based authorization constraints in BPMN. In *Business Process Management*, pages 64–79. Springer, 2007.