Optimized Workflow Authorization in Service Oriented Architectures

Martin Wimmer, Martina-Cezara Albutiu, and Alfons Kemper

Technische Universität München, 85748 Garching b. München, Germany
{wimmerma, albutiu, kemper}@in.tum.de

Abstract. Complex business processes are usually realized by specifying the integration and interaction of smaller modular software components. For example, hitherto monolithic enterprise resource planning systems (ERP) are decomposed into Web services which are then again orchestrated in terms of Web service workflows, bringing about higher levels of flexibility and adaptability. In general, such services constitute autonomous software components with their own dedicated security requirements. In this paper we present our approach for consolidating the access control of (Web service) workflows. The proposed security engineering method allows, first, to determine for whom workflows are executable from a privileges point of view, second, to assess compliance with the principle of least privilege, and, third, helps to reduce policy enforcement costs.

1 Introduction

The service oriented computing paradigm has introduced a change in the design of future large-scale enterprise applications. Monolithic software systems like classical enterprise resource planning systems (ERP) are increasingly replaced through Web services which provide the basic ERP functionality in the form of self-contained modules. As services constitute fine grained software components that can easily be linked due to a standardized interface description and communication protocol, complex business processes can be realized as service workflows. Though this architectural change brings about a plus of flexibility and adaptability, it poses new challenges in the area of administration, in particular w.r.t. security. Each Web service can pose individual security requirements, e.g., with respect to client authentication and authorization. Consider, for instance, the e-health workflow illustrated in Fig. 1. This simplified example workflow will be executed when a patient is transferred to the cardiology department of a hospital. Depending on the diagnostic findings, either an in-patient treatment is applied or an electrocardiogram (ECG) is made in order to acquire further expertise. Sub-activities of the workflow on the one hand represent practical activities that require human interaction like a medication. On the other hand, they stand for information processing tasks, like an update of the stock of pharmaceuticals in the database. In the following we concentrate on the technical aspects of the workflow and assume the subsequent access rules to be defined:

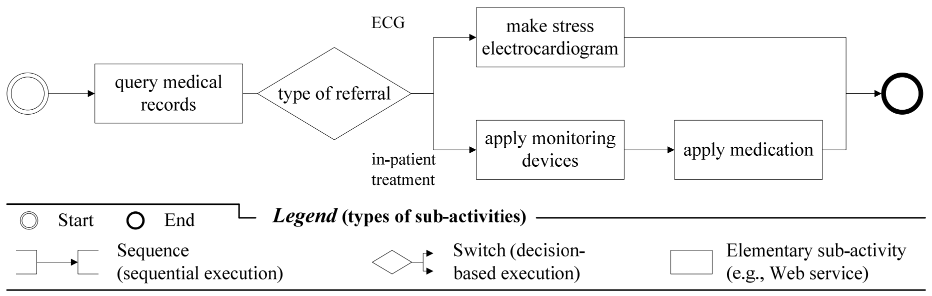

ECG
make stress electrocardiogram
query medical records
type of referral
in-patient treatment
apply monitoring devices
apply medication
Start
End
Legend (types of sub-activities)
Sequence (sequential execution)
Switch (decision-based execution)
Elementary sub-activity (e.g., Web service)

Fig. 1. Example for an e-health (Web service) workflow

- Health personnel with permanent employment and administrative employees are allowed to access the medical records of patients. These are stored in the table *MedicalRecordsTab* of the hospital's database (applies to *query medical records*).
- Nurses of the cardiology and internists are allowed to update medical records, e.g., by inserting ECG results (applies to *make stress electrocardiogram*).
- Internists are allowed to *apply monitoring devices* by marking them in the *DevicesTab* as *in use*.
- Nurses and physicians can *apply medications*, in case the patient is not allergic concerning the respective pharmaceutical.

Nowadays, access control is usually performed at the services' layer. This approach results from the mentioned autonomy of authorization but brings about two major shortcomings: First, security evaluations are performed redundantly. That is, authentication and authorization of the same client will be done repeatedly, which might be very costly considering for example certificate evaluation and verification. Second, further performance drawbacks can emerge, if services are needlessly invoked in cases when subjects lack privileges at later stages in the workflow. For instance, querying the medical records of a patient will be done unnecessarily, if the workflow is called by an administration employee that is neither able to pursue the *ECG*- nor the *in-patient-treatment*-branch of the workflow. Regarding Web service transactions even rollbacks or compensating actions can be required then. Therefore, at the time a workflow is designed, the following issues addressing access control arise:

1. *Who is allowed to execute the workflow?* We are interested in answering this question from the *single-user / single-role* perspective which applies to many business processes. Therefore, we determine the user profiles that grant workflow execution, without demanding for additional role activations or profile switches.
2. *What is the minimum set of required privileges?* This issue is also known as the *principle of least privilege* or the *need to know paradigm* which demand that subjects are only granted those privileges required to fulfill their tasks.

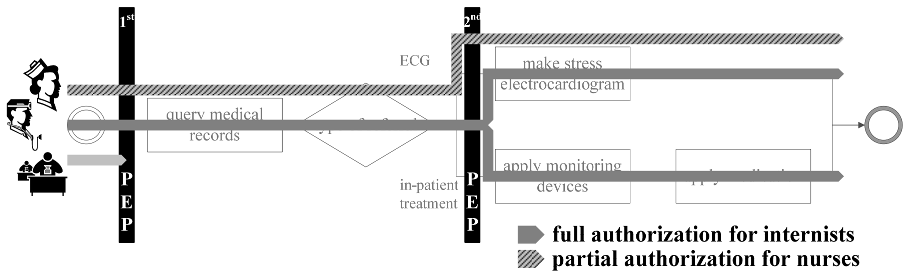

1st
PEP
query medical records
ECG
2nd
PEP
make stress electrocardiogram
in-patient treatment
apply monitoring devices
full authorization for internists
partial authorization for nurses

Fig. 2. Access control configurations

3. *Can policy evaluation be optimized?* At best, access control needs to be performed only once on top of the workflow layer, screening non-authorized requests and avoiding redundant policy evaluations at the services' layer.

Our contribution is a generic security engineering approach for (Web service) workflows that addresses these questions. It is based upon the consolidation of Web service policies that together define the access control settings of a workflow. That means, the minimum set of required privileges is derived from the privileges needed to execute the individual sub-activities. By means of a structural analysis it is analyzed whether subjects exist that are granted the necessary privileges to execute the workflow. In order to provide better scalability, role based access control can be employed. As we will show, our policy consolidation approach allows to determine the least-required roles that authorize the execution of the workflow.

To give an example, subjects being allowed to execute the *in-patient-treatment*-branch illustrated in Fig. 1 need to be granted privileges for the services *query medical records*, *apply monitoring devices*, and *apply medication*. Consequently, these subjects are in the intersection of the subjects authorized for the individual sub-activities. With regard to our informal policy specification, this applies to internists. As internists are also allowed to *make stress electrocardiograms*, they possess the privileges to invoke any branch within the workflow. We call this *full authorization*. On the contrary, nurses are granted *partial authorization*, as they are only allowed to execute the *ECG*-branch of the workflow. Other subjects, like administrative employees, – though being able to invoke *query medical records* – do not possess the required privileges to execute a complete branch and will be blocked right from the beginning. Fig. 2 illustrates the different access control configurations and shows how this information can be used to make access control more efficient. Instead of retaining access control at the services' layer and, thus, having to cope with the mentioned performance drawbacks, access control at the workflow layer can block unsuccessful execution at an early stage. To achieve this, policy enforcement points (PEP), i.e., points in the workflow where access control will be performed, are inserted at branches in the workflow. Moreover, if access control is constrained to *full authorization*, even one PEP (the first PEP in the figure) can be sufficient.

In this paper, we first introduce the underlying formal policy model in Sec. 2, which constitutes the basis for our policy consolidation approach that is described in Sec. 3. The different access control configurations and their impact on the complexity of the consolidation are discussed in Sec. 4. Finally, related work is presented in Sec. 5 before we conclude and sketch ongoing work in Sec. 6.

2 Policy Model

Our policy consolidation approach is based on a formal policy model supporting discretionary and role based access control (DAC and RBAC) schemes. The formal syntax and semantics of our policy model are based on those introduced by Bonatti et al. [1]. We adapted and extended this model where necessary, e.g., by introducing additional operators.

2.1 Syntax

Policies in our model are composed of individual access rules, i.e., a policy P is described through a set of rules $\{R_1, \ldots, R_n\}$. A rule $R = (S, O, A, c)$ assigns privileges specified by resource and action information (O and A) to subjects S. The assignment can further be restricted through a condition c. S represents a set of subjects sub_i and can be written as $S = \{sub_1, \ldots, sub_m\}$. For example, in the health service context, sub_i can stand for one individual nurse or physician. We call this the *set-based* representation of S. In comparison, the *attribute-based* description provides a higher level of expressiveness. Using *attribute-based* descriptions, subjects are specified through predicate conjunctions. A predicate is of the form (*attribute-identifier* $\circ$ *constant*). Depending on the attribute's domain, the comparison operator $\circ$ is in $\{<, \leq, =, \geq, >\}$ for totally ordered sets or in $\{\sqsubset, \sqsubseteq, =, \sqsupseteq, \sqsupset\}$ for partially ordered finite sets. A set of subjects S is represented as a disjunction of predicate conjunctions, i.e., $S \equiv (s_1 \vee \ldots \vee s_k)$, with $s_i = (s_{i,1} \wedge \ldots \wedge s_{i,l})$ (for $1 \leq i \leq k$). Thereby, $s_{i,d}$ represents a predicate conjunction that applies to one attribute. For instance, given the attribute identifiers *role* and *years of practice* (abbrev. *y-o-p*), the conjunction (role $\sqsupseteq$ Nurse $\wedge$ y-o-p ≥ 2) specifies users that are granted the role *Nurse* and have at least two years of operational experience.

Analogously, O and A are either described in the *set-based* or the *attribute-based* way. Subjects, resources, and actions are specified on disjoint sets of attribute identifiers, denoted as *S-Attr*, *O-Attr*, and *A-Attr*. S, O, and A are inequality-free, i.e., there is no predicate whose operator $\circ$ is $\neq$. The same does not hold for conditions, which are arbitrary Boolean formulae and can include user defined functions with Boolean codomain. Conditions are defined on environment attributes of *E-Attr*.

Projection-Operator Given a rule $R = (S, O, A, c)$, the operator $\Pi_{\mathcal{S}}$ projects on the subjects-part of R, i.e., $\Pi_{\mathcal{S}}(R) = S$. Similar operators are defined for projections on resources ($\Pi_{\mathcal{O}}$), actions ($\Pi_{\mathcal{A}}$), privileges ($\Pi_{\mathcal{O},\mathcal{A}}$), and conditions ($\Pi_{\mathcal{C}}$). Let $P = \{R_1, \ldots, R_n\}$ be a policy. $\Pi_{\mathcal{S}}(P)$ is defined as: $\Pi_{\mathcal{S}}(P) =$

$\{\Pi_{\mathcal{S}}(R_1), \ldots, \Pi_{\mathcal{S}}(R_n)\}$. Other projection operators on policies are defined in a similar way. We use the abbreviation $\mathcal{S}(P) = \bigcap_{1 \leq i \leq n} \Pi_{\mathcal{S}}(R_i)$ to denote those subjects that are granted all privileges defined in P.

2.2 Semantics

An evaluation context $e \in \mathcal{E}$ is a partial mapping of the attributes in *S-Attr* $\cup$ *O-Attr* $\cup$ *A-Attr* $\cup$ *E-Attr*. If $D_1, \ldots, D_m$ are the distinguished attribute domains, then $\mathcal{E}$ is defined as $D_1^{\perp} \times \ldots \times D_m^{\perp}$, with $D_j^{\perp} = D_j \cup \{\perp\}$ and $\perp$ representing an unspecified attribute value.

An evaluation context e is evaluated against the individual components of rules. A subject specification S applies to e, iff S maps to *true* w.r.t the attribute values of e. That is, $[\![S]\!]_e := S(e) = (\mathit{true}|\mathit{false})$. Evaluating objects, actions and conditions is defined similarly. A rule R applies to e, iff $[\![R]\!]_e := [\![S]\!]_e \wedge [\![O]\!]_e \wedge [\![A]\!]_e \wedge [\![c]\!]_e$ maps to *true*.

Policies aggregate the access rights that specify the access control requirements of applications which are composed of individual sub-activities. The semantics of a policy P depends on the employed policy evaluation algorithm. If the access rules of the individual sub-activities are enforced successively, P applies to e, iff *any* of its rules apply. That is $[\![P]\!]_e^{\mathit{pe\text{-}any}} := \bigvee_{R \in P} [\![R]\!]_e$. In contrast to this, the policy evaluation algorithm *pe-all* can be used to statically enforce a policy before any sub-activity is invoked. It is defined as $[\![P]\!]_e^{\mathit{pe\text{-}all}} := \bigwedge_{R \in P} [\![R]\!]_e$.

2.3 Role Based Access Control

Policy administration can easily become unmanageable if privileges are independently assigned to each user. Better scalability is provided through role based access control (RBAC), which was introduced by Sandhu et. al. in 1996 [2] and for which the standard [3] has been released in 2004. Using RBAC, privileges that are required for performing a certain task are grouped by roles and users acquire these privileges via the indirection of being granted those roles. Consequently, roles play two parts in our policy model. On the one hand, roles act as subjects when being used to group privileges (*privilege-to-role* assignment). On the other hand, roles are objects when they are assigned to other subjects – which then could be a user or a further role (*role-to-subject* assignment). That way, roles can be organized in a hierarchy. As an example consider the role hierarchy for our e-health scenario which is shown in Fig. 3. Through the hierarchy a partial order on roles is defined: Senior roles, which are at higher levels in the hierarchy, are granted all privileges that are also granted to their junior roles. To give an example, the role *Internist* is senior to *Physician*, denoted as *Internist* $\sqsupset$ *Physician*. Accordingly, *Physician* is called junior role of *Internist*. All subjects that are granted the role *Physician* or any senior role of it are represented through the predicate (role $\sqsupseteq$ Physician).

With regard to workflow authorization we are interested in determining the *least-required roles*. For example, if the roles *Head Nurse* and *Nurse* authorize the execution of the workflow illustrated in Fig. 1, then the least-required role

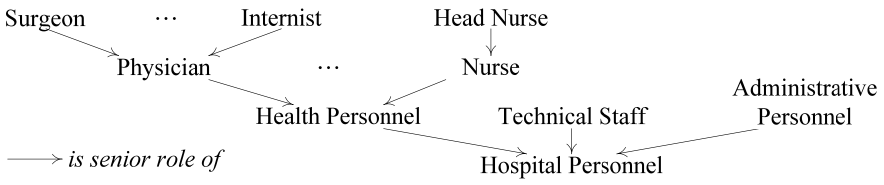


Fig. 3. Example role hierarchy for a hospital

with respect to the introduced role hierarchy is *Nurse*. If a role authorizes the execution of a workflow (or a workflow branch), we demand the role to authorize the execution of each sub-activity of the workflow (branch) without the activation of any further role. This *single-role*-approach is particularly relevant for ERP applications: Business processes like financial accounting or the admission of patients in our e-health scenario represent self-contained tasks that are to be performed by users that are granted business process specific roles, i.e., least-required roles in our terminology.

2.4 Example

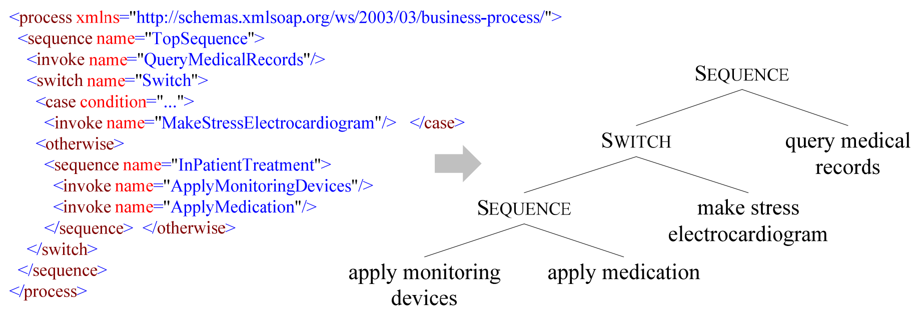
According to the informal access rules stated in the introduction, we define the following policies: P_{MR} (applies to *query medical records*), P_{ECG} (*make stress electrocardiogram*), P_{App} (*apply monitoring devices*), and P_{Med} (*apply medication*). The attribute identifiers o and a represent resources and actions, *f-o-a* is the abbreviation for *field of activity.*

$$\begin{aligned}
P_{\mathrm{MR}} =&\{(((\text{role} \sqsupseteq \text{Health Personnel} \wedge \text{employment} = \text{permanent})\vee\\
&(\text{role} \sqsupseteq \text{Administrative Personnel})),\\
&(o = \text{MedicalRecordsTab}), (a = \text{select}), (\text{true}))\}\\
P_{\mathrm{ECG}} =&\{(((\text{role} \sqsupseteq \text{Nurse} \wedge \text{f-o-a} = \text{cardiology}) \vee \text{role} \sqsupseteq \text{Internist}),\\
&(o = \text{MedicalRecordsTab}), (a = \text{select} \vee a = \text{update}), (\text{true}))\}\\
P_{\mathrm{App}} =&\{((\text{role} \sqsupseteq \text{Internist}), (o = \text{DevicesTab}), (a = \text{select} \vee a = \text{update}), (\text{true}))\}\\
P_{\mathrm{Med}} =&\{((\text{role} \sqsupseteq \text{Nurse} \vee \text{role} \sqsupseteq \text{Physician}), (o = \text{PharmaceuticalsTab}),\\
&(a = \text{select} \vee a = \text{update}), (\mathit{HighAnaphylaxisRisk}(\text{patient},\text{drug}) = \text{false}))\}
\end{aligned}$$

3 Access Control for Web Service Workflows

3.1 Workflow Model

Workflows are used to model and realize complex (business) processes by specifying the invocation order of fine-grained sub-activities. BPEL4WS [4] will be widely used for defining Web service workflows. Currently it is revised by OASIS for standardization under the name WS-BPEL. It provides five control patterns, namely *flow*, *while*, *sequence*, *switch*, and *pick*. We refer to [4] for their specifications. The only aspect being of relevance for access control is, whether all or



```xml
<process xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/">
 <sequence name="TopSequence">
  <invoke name="QueryMedicalRecords"/>
  <switch name="Switch">
   <case condition="...">
    <invoke name="MakeStressElectrocardiogram"/>  </case>
   <otherwise>
    <sequence name="InPatientTreatment">
     <invoke name="ApplyMonitoringDevices"/>
     <invoke name="ApplyMedication"/>
    </sequence>  </otherwise>
  </switch>
 </sequence>
</process>
```

Fig. 4. BPEL4WS-extract and workflow tree for the e-health business process

only some of the sub-activities grouped by any of these control patterns have to be invoked. As the first three control patterns require all sub-activities to be executed[1], we group them together and represent them through the SEQUENCE-template. The remaining patterns specify that at most one sub-activity will be called, e.g., due to the evaluation of conditions or because of the arrival of certain events. We abstract from the respective flow specification and aggregate them together to the SWITCH-template. SEQUENCE and SWITCH are also sub-activities in the meaning of our workflow model.

A Web service choreography can be represented by a *workflow tree* as illustrated in Fig. 4 for our example business process. Inner nodes of such a tree either represent SEQUENCE- or SWITCH-nodes, while leaves represent Web services. In order to obtain a consolidated view onto the access control settings of a workflow, we perform a structural analysis of the workflow tree. This analysis offers possibilities for design-revision and optimization as described in Sec. 3.6.

3.2 Preliminary Remarks

As mentioned before, the leaves of a workflow tree represent Web services. In our setting, Web services are autonomous software components that can be supplied by varying service providers and that might be integral parts of varying workflows. This autonomy implies that in the general case the policy administration authorities vary. In order to be able to perform the intended policy analysis, we demand the following three conditions to hold:

1. Policies can be expressed in one common policy language and rely on DAC and RBAC models.
2. The access control specifications of the Web services fulfill the principle of least privilege.
3. The subject specifications are based on the same description language.

If condition 2 holds, we can infer the consolidated policies to comply with the principle of least privilege. The minimality criterion for Web services can often be automatically ensured, as we showed in [5] for database dependent Web

[1] We assume that sub-activities of *while*-patterns will be executed at least once.

services. The third assumption refers to the employed terminology. As one of our aims is to determine the set of subjects authorized for the workflow, we require subjects to be uniquely identifiable. If all policies belong to the same (intra-organizational) repository, this would typically be the case. With regard to distributed Web service invocations, access control usually relies on a federated identity management, e.g., realized via WS-Federation and SAML. In this case, identity mappings have to be resolved as a preliminary step. As mentioned in the introduction, our focus is on intra-organizational application integration like the service oriented implementation of ERP systems, where the three conditions are typically met.

3.3 Policy Consolidation

Through a bottom-up analysis, access control policies for higher levels of the workflow tree are iteratively determined and, finally, a consolidated policy for the complete workflow is inferred. Generating the policy for an inner node depends on the node's type, i.e., wether it is a SEQUENCE- or a SWITCH-node. In the following we consider a node NODE that has n child nodes. The child nodes are numbered from 1 to n and each of them represents a sub-activity, i.e., a further SEQUENCE- or SWITCH-node or an elementary service. For each sub-activity i, P_i represents the related policy.

Consolidating the policies of Sequence-nodes A SEQUENCE node enforces the execution of all n sub-activities. Thus, each subject authorized for NODE must as well be authorized for each sub-activity. The consolidated policy for the SEQUENCE-node consists of all privileges that apply to the sub-activities and is restricted to the intersection of the subject specifications.

The set of subjects allowed to execute NODE is $S_{\text{all}} = \bigcap_{1 \leq i \leq n} \mathcal{S}(P_i)$. Consequently, the consolidated policy is constructed as follows:

$$P^{\text{opt}}_{(\text{all})} = \{(S_{\text{all}}, \Pi_{\mathcal{O},\mathcal{A}}(R), \Pi_{\mathcal{C}}(R)) \mid R \in P_i, 1 \leq i \leq n\} \tag{1}$$

When aggregating the policies for a SEQUENCE-node, rules whose privileges are relaxed by another rule of $P^{\text{opt}}_{(\text{all})}$ can be reduced by *AND*-ing the conditions, if the employed policy evaluation algorithm is *pe-all.* That is, given a partial order on policies, like the one we presented in [5], two rules $(S_{\text{all}}, O_1, A_1, c_1)$ and $(S_{\text{all}}, O_2, A_2, c_2)$ can be aggregated to $(S_{\text{all}}, O_2, A_2, c_1 \wedge c_2)$, in case $O_1 \subseteq O_2$ and $A_1 \subseteq A_2$. If $S_{\text{all}} \neq \emptyset$, the NODE is *subject-executable.* That means that there exists at least one subject or role that is granted the privileges to execute NODE.

Consolidating the policies of Switch-nodes Identifying the policy for a SWITCH-node is more challenging, as the permission to execute the node depends on the user context and the history of previous execution steps. Users can be authorized to execute branches of the workflow but need not be privileged to perform all sub-activities. Thus, each subject defined in any sub-activity's policy is granted privileges for the SWITCH-node.

Again, let $S_{\text{all}} = \bigcap_{1 \leq i \leq n} \mathcal{S}(P_i)$ be the set of subjects authorized to execute all sub-activities. Then, all subjects defined in S_{all} are granted *full authorization* for the SWITCH-node. Thus, the applicable policy is $P^{\text{opt}}_{(\text{all})}$ as defined in equation (1).

On the opposite side, *partial-authorization* distinguishes the different execution paths. We obtain the following policy for the subjects that are authorized for the i^{th} branch:

$$P^{\text{opt}}_{(i)} = \{(\mathcal{S}(P_i) - S_{\text{all}}, \Pi_{\mathcal{O},\mathcal{A}}(R), \Pi_{\mathcal{C}}(R)) \mid R \in \mathcal{P}_i\} \tag{2}$$

Thus, if any of the delta sets $\mathcal{S}(P_i) - S_{\text{all}}$ are non-empty, policy analysis branches. Each of the up to $n+1$ different cases are considered as virtual replications of the original workflow tree and analysis for them proceeds separately.

3.4 Computing Subject Intersections

In this section we discuss the computation of subject intersections, which is an integral part of the consolidation process presented in Sec. 3.3. As shown in Sec. 2, subjects (like resources and actions, too) are either described in the *set-based* or the *attribute-based* way. As the *attribute-based* description is more expressive, we assume this representation for explaining the computation of intersections.

Let S and S' be two subject sets. According to our policy model, both sets are represented in disjunctive normal form (DNF):

$$\begin{aligned} S &\equiv s_1 \vee \ldots \vee s_k = (s_{1,1} \wedge \ldots \wedge s_{1,l}) \vee \ldots \vee (s_{k,1} \wedge \ldots \wedge s_{k,l}) \quad \text{and} \\ S' &\equiv s'_1 \vee \ldots \vee s'_{k'} = (s'_{1,1} \wedge \ldots \wedge s'_{1,l}) \vee \ldots \vee (s'_{k',1} \wedge \ldots \wedge s'_{k',l}) \end{aligned}$$

S and S' are defined over attributes of *S-Attr*. The elements of *S-Attr* are also called the dimensions of a subject specification. We assume S and S' to be specified in each each dimension. If a conjunction s_i is not constrained in dimension d, the respective predicate $s_{i,d}$ represents the whole domain of d.

Alg. 1 gives a pseudo-code representation for computing the intersection of subject sets. We illustrate the computation of subject intersections by means of an example. Consider the following two subject descriptions:

$$\begin{aligned} S &\equiv s_1 = (\text{role} \sqsupseteq \text{Nurse} \wedge \text{y-o-p} \geq 1) \\ S' &\equiv s'_1 \vee s'_2 = (\text{role} \sqsupseteq \text{Admininistrative Personnel} \wedge \text{y-o-p} \geq 0) \vee \\ &\qquad\qquad\qquad (\text{role} \sqsupset \text{Health Personnel} \wedge \text{y-o-p} \geq 2 \wedge \text{y-o-p} \leq 4) \end{aligned}$$

S represents all subjects that are granted the *Nurse* role and that have at least one year of practice (*y-o-p*). S' represents administrative employees and all subjects that are granted senior roles of the *Health Personnel* role with at least two and at most four years of practice. Thus, the dimensions are *role* and *y-o-p*. While the domain of *role* is a finite lattice (defined by the role hierarchy shown in Fig. 3), the domain of *y-o-p* is $[0, +\infty[$.

s_1 and s'_1 represent disjoint sets. Both terms do not overlap in the *role* dimension as $s_{1,\text{role}} \equiv \{\textit{Nurse}, \textit{Head Nurse}\}$ and $s'_{1,\text{role}} \equiv \{\textit{Administrative Personnel}\}$.

Algorithm 1 ***intersect***(S, S'), with $S \equiv s_1 \vee \ldots \vee s_k$, and $S' \equiv s'_1 \vee \ldots \vee s'_{k'}$

```
 1: $\Psi$ = false
 2: for all conjunctions $s_i$ of $S$ do
 3:   for all conjunctions $s'_j$ of $S'$ do
 4:     for all dimensions $d = 1 \ldots l$ do
 5:       $\psi_d = \text{reduce}(s_{i,d} \wedge s'_{j,d})$
 6:     end for
 7:     $\Psi = \Psi \vee (\psi_1 \wedge \ldots \wedge \psi_l)$
 8:   end for
 9: end for
10: return $\Psi$
```

$$\boxed{S \cap S' \equiv \Psi = \bigvee_{1 \le i \le k, 1 \le j \le k'} \left(\bigwedge_{1 \le d \le l} \left(s_{i,d} \wedge s'_{j,d} \right) \right)}$$

That is, the intersection of both sets in this dimension is empty and ψ_{role} in line 5 of Alg. 1 is equivalent to *false*. Therefore, the overlap in the *y-o-p* dimension is ineffectual as the conjunctive add-on in line 7 evaluates to *false* and can be omitted.

In contrast to this, s_1 and s'_2 overlap in each dimension as illustrated in Fig. 5. The conjunction (y-o-p $\geq$ 1) $\wedge$ (y-o-p $\geq$ 2 $\wedge$ y-o-p $\leq$ 4) is reduced to (y-o-p $\geq$ 2 $\wedge$ y-o-p $\leq$ 4). In general, intersections of totally ordered sets are computed by comparing the respective lower and upper bounds. The predicates $s_{1,\text{role}}$ and $s'_{2,\text{role}}$ define the two finite sets $\Phi_1 = \{\textit{Nurse}, \textit{Head Nurse}\}$ and $\Phi'_2 = \{\textit{Nurse}, \textit{Head Nurse}, \textit{Physician}, \textit{Internist}, \textit{Surgeon}\}$. Thus, $(s_{1,\text{role}} \wedge s'_{2,\text{role}})$ is equivalent to $\Phi_1 \cap \Phi'_2 = \{\textit{Nurse}, \textit{Head Nurse}\}$. The infimum of $\Phi_1 \cap \Phi'_2$ is the role *Nurse*. Therefore, $(s_{1,\text{role}} \wedge s'_{2,\text{role}})$ can be reduced to (role $\sqsupseteq$ Nurse) so that the intersection of S_1 and S_2 is equivalent to

(role $\sqsupseteq$ Nurse $\wedge$ y-o-p $\geq 2 \wedge$ y-o-p ≤ 4)

That is, the intersection consists of those subjects that are granted the *Nurse* role and that have at least two and at most four years of practice.

3.5 Example

Performing the policy consolidation for our running example starts with analyzing the policies P_{App} and P_{Med} that apply to the activities *apply monitoring devices* and *apply medication* which are linked in sequence as illustrated in Fig. 1 and Fig. 4. The subjects allowed to execute both are those granted the *Internist* role. The consolidation process is continued by analyzing the SWITCH-node. The following cases have to be distinguished:

1. *Internist*s are in the intersection of the subject sets that are allowed to execute both branches (*ECG* and *in-patient treatment*).
2. *Nurse*s working at the cardiology are only granted privileges for the *ECG*-branch.

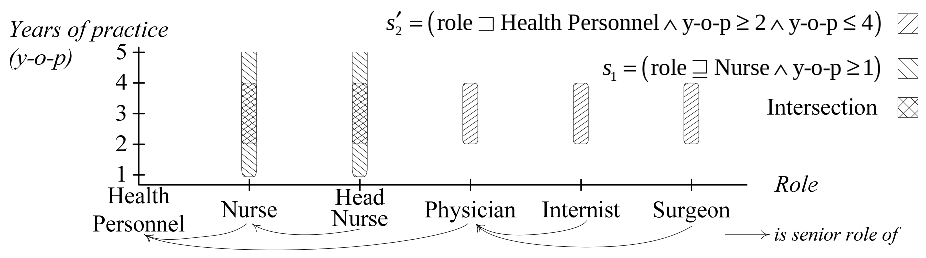


Fig. 5. Matching conjunctive terms

The last step is to analyze the top SEQUENCE-node for both cases that were derived in the previous step. We compute the intersection

$$\begin{aligned}&(\text{role} \sqsupseteq \text{Health Personnel} \vee \text{role} \sqsupseteq \text{Administrative Pers.}) \wedge (\text{role} \sqsupseteq \text{Internist})\\&\quad = (\text{role} \sqsupseteq \text{Internist})\end{aligned}$$

Thus, subjects granted the *Internist* role are allowed to execute the complete workflow. The applicable consolidated policy is $P^{\text{opt}}_{(\text{all})}$ with

$$\begin{aligned}P^{\text{opt}}_{(\text{all})} =\{&((\text{role} \sqsupseteq \text{Internist} \wedge \text{employment} = \text{permanent}), (o = \text{MedicalRecordsTab}),\\&\quad(a = \text{select} \vee a = \text{update}), (\mathit{true})),\\&((\text{role} \sqsupseteq \text{Internist} \wedge \text{employment} = \text{permanent}), (o = \text{DevicesTab}),\\&\quad(a = \text{select} \vee a = \text{update}), (\text{true})),\\&((\text{role} \sqsupseteq \text{Internist} \wedge \text{employment} = \text{permanent}), (o = \text{PharmaceuticalsTab}),\\&\quad(a = \text{select} \vee a = \text{update}), (\mathit{HighAnaphylaxisRisk}(\text{patient,drug}) = \text{false}))\}\end{aligned}$$

Analogously, according to the definition of P_{MR} and the role hierarchy depicted in Fig. 3, the subjects allowed to execute the *ECG*-branch are those that are granted the *Nurse* role. The consolidation for this branch results in

$$\begin{aligned}P^{\text{opt}}_{(\text{ECG})} =\{&((\text{role} \sqsupseteq \text{Nurse} \wedge \text{employment} = \text{permanent} \wedge \text{f-o-a} = \text{cardiology}),\\&\quad(o = \text{MedicalRecordsTab}), (a = \text{update} \vee a = \text{select}), (\text{true}))\}\end{aligned}$$

The conclusions to be drawn from the structural analysis are

1. The workflow is subject-executable for *Internist*s that have full authorization and for *Nurse*s that are only authorized for the *ECG*-branch. These two roles constitute least-required roles in the meaning of Sect. 2.3.
2. $\Pi_{\mathcal{O},\mathcal{A}}(P^{\text{opt}}_{(\text{all})})$ and $\Pi_{\mathcal{O},\mathcal{A}}(P^{\text{opt}}_{(\text{ECG})})$ entail the minimum sets of required privileges for *full* and *partial authorization*, respectively.
3. For *Internist*s, the authorization decision can be made at the workflow layer, and access control costs for sub-activities can be reduced as discussed in the next section. For *Nurse*s, access control has to be performed twice: On top of the workflow layer and when entering the *ECG*-branch. Other subjects, like those granted the *Administrative Personnel* role, can be blocked right from the beginning. Thus, access control can be optimized as illustrated in Fig. 2.

3.6 Interpretation

In this section we give a qualitative evaluation of the described consolidation approach with regard to the three objectives introduced in the beginning.

A workflow is subject-executable if the root node of the workflow tree has a non-empty policy that grants at least one subject the required privileges to execute a branch of the process. If the *full authorization*-approach is applied, subjects defined in the topmost policy are authorized to invoke any branch. The consolidation approach will also detect *dead paths* that are branches within the workflow which will never be subject-executable. Dead paths are identified via nodes with empty policies.

At the end of the policy consolidation process only those subjects and roles remain in the policy of the root node that are granted the required privileges to execute at least one branch of the workflow. If condition 2 of Sect. 3.2 holds, i.e., the Web services' policies fulfill the least-privilege criterion, then the principle of least privilege can be inferred for the complete workflow, too. By computing the infimum of authorized roles (see line 5 of Alg. 1 and Sect. 3.4) the least-required roles are determined. Providing workflow designers with this consolidated information, they can evaluate, whether only highly privileged users (or roles) remain – situations that are usually unintended and have to be revised.

The third objective is the reduction of access control costs. The described approach is a step towards saving recurrent policy enforcements, as those points in the control flow are determined, where policies have to be enforced at least. Without making use of this optimization, each Web service individually triggers policy evaluation. Using the *full authorization*-approach, the most costly part of access control will be processed on top of the workflow layer. If following the *partial authorization*-approach, intermediary policy enforcement points have to be realized for the Switch-nodes as shown in Fig. 2. As Web services are autonomous software components – a characteristics which will not be given up by our approach –, access control at the services' layer cannot be removed. Nevertheless, the enforcement costs can be reduced significantly by including workflow specific policies into the policies of the services. Instead of repeatedly initiating subject identification, only certain "pass-through"-credentials (e.g., realized as SAML assertions) are employed, allowing better performing security evaluation. Such "pass-through"- credentials are issued if access control at the workflow layer succeeds. This approach can be realized for intra-organizational business processes, when workflow architects have the possibility to optimize the Web services' policies4.

4 Complexity of Policy Consolidation

Policy Consolidation is performed at the time a workflow is designed and therefore is not as time critical as policy enforcement at runtime. The overall complexity is subdivided into the complexity for the calculation of subject intersections and the complexity for performing the structural analysis.

Computing the intersection of subject sets is needed to determine S_{all} in equations (1)–(2). Algorithm 1 is of polynomial complexity w.r.t. the number of conjunctions and number of dimensions. Furthermore, when following the *partial authorization* approach, also the difference of subject sets has to be computed (equation (2)). Using *set-based* descriptions this can be done in polynomial time w.r.t. set cardinalities. When employing *attribute-based* descriptions, computing the difference can be done by a modified implication test – instead of checking whether $\mathcal{S}(P_i) \Rightarrow S_{\text{all}}$ it is examined which predicates of $\mathcal{S}(P_i)$ are $\underline{\text{not}}$ subsumed by S_{all}. Due to space limitations we will not present further details of this algorithm. But, similar to the query subsumption problem [6], the complexity of such an algorithm is exponential in the worst case. This worst case arrives, if the conjunctive terms of $\mathcal{S}(P_i)$ overlap only partially with the conjunctions of S_{all} (and recursive iterations are necessary). Predicate conjunctions define subjects with similar characteristics (role attributes, age and so on). Thus, the likelihood of related subjects being described by different conjunctions within the same policy (which would be the reason for the worst case to occur) can be estimated to be quite low, so that the worst case is assumed to arise rarely.

The complexity for performing the structural analysis depends on whether the *full authorization*-approach is employed or not. In case of the *full authorization*-approach, the number of policy comparisons depends on the number of inner nodes of the workflow tree. Complexity can increase drastically, if *partial authorizations* should be determined. Then, if m represents the depth of the workflow tree and n its branching factor, up to $(n+1)^m$ cases have to be evaluated to derive the top level policy in the worst case. This worst case arises, if each inner node is a SWITCH-node, and for each such SWITCH-node the maximum number of subcases has to be considered. However, *partial authorization* is of minor practical relevance, if the top-level policies will reach an unmanageable size which reduces their interpretability by the process engineer. Thus, it is reasonable to consider *partial authorization* only if the workflow size and the number of switches are limited.

5 Related Work

Modeling processes as workflows is of importance for many applications, e.g., e-commerce, e-science or e-health. [7] gives an overview over approaches for defining Web service workflows also named Web service orchestrations or choreographies. In the course of this paper, we showed by use of an e-health example, how access rules for services determine the consolidated workflow policy. This example was simplified in order to keep the discussion concise. More details on access control for e-health scenarios are for instance provided by [8]. The authors of [9] present an approach to define and enforce conditions for the integration of Web services into workflows. [10, 11] describe architectures and algorithms for the enforcement of access control for workflows. [11] especially focuses on the enforcement of static and dynamic separation of duty. Therefore, they introduce a formalism that allows to specify and evaluate separation of duty constraints at the workflow level. In contrast to this, we concentrate on *single-user / single-role* execution schemes which we assume to be prevalent for most enterprise applications. Whether consolidated workflow policies have to be enforced successively (via *pe-any*) or can be evaluated statically (using *pe-all*), depends on whether dataflow and temporal interdependencies have to be taken into account or not [12].

Our consolidation approach is also related to work addressing access control for distributed and multi-layered applications [13–15] and the optimization of policies [16]. The authors of [16] show how policy sets can be optimized by detecting and resolving redundancies, which helps to reduce policy enforcement costs. [17] and [5] introduce partial orders on policies. These are useful for the consolidation of policies for application integration purposes, like the evaluation whether access control rules of dependent applications are compatible. We are addressing this issue in a larger context by evaluating the integrability of business services in workflows with the aim to optimize access control for complex business processes.

6 Conclusion and Future Work

The concise separation of security and business logic is a crucial aspect for the classical software engineering process. The same considerations apply to application integration and the design of business processes. We presented our approach to facilitate the reliable development of (Web service) workflows by providing consolidated views onto the access control requirements of workflows, which assists software engineers in revising business processes, determining whether or not processes are executable from a privileges point of view, and detecting possible security shortcomings like non-compliances with the principle of least privilege. Moreover, using our approach, policy enforcement for business processes can be optimized and unsuccessful invocations can be detected at an early stage, thus, avoiding unnecessary service executions and rollback costs or compensating actions. We realized a prototypical implementation of the presented consolidation technique, employing BPEL4WS as workflow specification language and XACML [18, 19] as policy language.

The described approach is particularly useful for intra-organizational workflows, where it is likely that access control policies rely on the same policy model. For distributed workflows, further preprocessing is required as mentioned before. Nevertheless, a comprehensive security engineering approach for the distributed case remains as future work.

References

1. P. Bonatti, S. De Capitani di Vimercati, and P. Samarati, "An Algebra for Composing Access Control Policies," *ACM Trans. Inf. Syst. Secur.*, vol. 5, no. 1, pp. 1–35, 2002.
2. 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.
3. ANSI INCITS 359-2004, *Role Based Access Control.* American National Standards Institute, Inc. (ANSI), New York, NY, USA, Feb. 2004.

4. S. Thatte *et al.*, “Business Process Execution Language for Web Services version 1.1 (BPEL4WS 1.1).” `http://www-128.ibm.com/developerworks/library/specification/ws-bpel/`, May 2003.
5. M. Wimmer, D. Eberhardt, P. Ehrnlechner, and A. Kemper, “Reliable and Adaptable Security Engineering for Database-Web Services,” in *Proceedings of the Fourth International Conference on Web Engineering*, vol. 3140 of *Lecture Notes in Computer Science (LNCS)*, (Munich, Germany), pp. 502–515, July 2004.
6. D. J. Rosenkrantz and H. B. Hunt, “Processing Conjunctive Predicates and Queries,” in *Proc. of the Intl. Conf. on Very Large Data Bases (VLDB)*, (Montreal, Canada), pp. 64–72, Oct. 1980.
7. R. Khalaf and F. Leymann, “On Web Services Aggregation,” in *Proceedings of the 4th International Workshop on Technologies for E-Services (TES 2003)*, vol. 2819 of *Lecture Notes in Computer Science (LNCS)*, (Berlin, Germany), pp. 1–13, Sept. 2003.
8. Deutsche Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie e.V., “AG Datenschutz in Gesundheitsinformationssystemen.” `http://info.imsd.uni-mainz.de/AGDatenschutz/`.
9. M. Altunay, D. Brown, G. Byrd, and R. Dean, “Trust-Based Secure Workflow Path Construction,” in *Proceedings of the 3rd International Conference on Service Oriented Computing*, vol. 3826 of *Lecture Notes in Computer Science (LNCS)*, (Amsterdam, The Netherlands), pp. 502–515, Dec. 2005.
10. W.-K. Huang and V. Atluri, “SecureFlow: a Secure Web-enabled Workflow Management System,” in *RBAC ’99: Proceedings of the 4th ACM Workshop on Role-based Access Control*, (New York, NY, USA), pp. 83–94, ACM Press, 1999.
11. E. Bertino, E. Ferrari, and V. Atluri, “The Specification and Enforcement of Authorization Constraints in Workflow Management Systems,” *ACM Trans. Inf. Syst. Secur.*, vol. 2, pp. 65–104, Feb. 1999.
12. C. Bettini, X. S. Wang, and S. Jajodia, “Temporal Reasoning in Workflow Systems,” *Distrib. Parallel Databases*, vol. 11, no. 3, pp. 269–306, 2002.
13. M. Rits, B. D. Boe, and A. Schaad, “Xact: a Bridge between Resource Management and Access Control in Multi-layered Applications,” in *SESS ’05: Proceedings of the 2005 Workshop on Software Engineering for Secure Systems*, (New York, NY, USA), pp. 1–7, ACM Press, 2005.
14. K. Rannenberg and G. Müller, *Security in Communications – Technology, Infrastructure, Economy.* Reading, MA, USA: Addison-Wesley, July 1999.
15. J. Biskup, T. Leineweber, and J. Parthe, “Administration Rights in the SDSD-System,” in *Proceedings of the Seventeenth Annual Working Conference on Database and Application Security*, (Estes Park, Colorado, United States), Aug. 2003.
16. F. Rabitti, E. Bertino, W. Kim, and D. Woelk, “A Model of Authorization for Next-Generation Database Systems,” *ACM Trans. Database Syst.*, vol. 16, no. 1, pp. 88–131, 1991.
17. E. B. Fernandez, E. Gudes, and H. Song, “A Model for Evaluation and Administration of Security in Object-Oriented Databases,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 6, no. 2, pp. 275–292, 1994.
18. T. Moses *et al.*, “eXtensible Access Control Markup Language (XACML) version 2.0.” `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml`, Feb. 2005.
19. A. Anderson, “Core and Hierarchical Role Based Access Control RBAC Profile of XACML version 2.0.” `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml`, Sept. 2004.