Consistency Checks for Duties in Extended UML2 Activity Models

Sigrid Schefer
Institute for Information Systems and New Media
Vienna University of Economics and Business (WU Vienna), 1090, Austria
sigrid.schefer@wu.ac.at

Abstract—Process-aware information systems support the execution of business processes. Organizations require the precise specification of security policies that govern the behavior of subjects in these systems. Thereby, obligation policies specify duties to be fulfilled by certain subjects. In organizational contexts, duties are often associated with a certain task in a business process.

In this paper, we further elaborate two UML2 extensions which provide modeling support for roles, tasks, and duties in a business process context. In particular, we introduce the notion of mutual exclusion and binding constraints for duties in process-related RBAC models. Furthermore, we formally define respective consistency checks for design-time and runtime models.

Keywords—Binding of duty; OCL; RBAC; Security; Separation of Duty; UML;

I. INTRODUCTION

Organizations require the precise specification of policies that govern the behavior of subjects in information systems. While authorization policies define a subject’s permissions, obligation policies specify duties which must be fulfilled by certain subjects to meet legal or organizational requirements [5], [10]. In a business process context, duties are often associated with a particular task [9] (e.g., a bank clerk negotiating a contract is obliged to inform the customer on associated risks).

Security constraints such as mutual exclusion (ME) and binding constraints are increasingly important in process-aware information systems to control the execution of workflows. They can be applied to enforce process-related separation of duty (SOD) and binding of duty (BOD) policies related to a corresponding role-based access control (RBAC) model [3], [13], [15]. ME and binding constraints are usually specified on role- or task-level (see, e.g., [1], [11], [12], [14]). We propose their definition on duty-level which provides the finest-grained abstraction level for specifying constraints.

This paper is based on the following two UML2 extensions. In [12], the UML2 metamodel extension BusinessActivities was presented which allows for modeling process-related RBAC models. Moreover, this UML extension supports the definition of ME and binding constraints on tasks. In [9], a UML2 metamodel extension for modeling process-related duties in Business Activities was introduced.

The contribution of this paper is to further integrate these UML2 metamodel extensions by considering ME and binding constraints on duties and by formally defining respective consistency checks. Defining ME and binding constraints on task- and duty-level provides two different abstraction levels for constraint specifications to support their precise implementation. Consistency checks are formally defined to ensure the compliance of design-time process models with corresponding consistency requirements. Moreover, they check the consistency between design-time and runtime models.

The remainder of this paper is structured as follows. Section II summarizes the combined UML2 extension and introduces mutual exclusion and binding constraints for duties. Subsequently, in Section III relevant consistency checks are formally defined. Section IV discusses related work and Section V concludes the paper.

II. UML2 METAMODEL EXTENSION FOR DUTIES

In [9], [12], modeling support for roles, tasks, and duties in business processes is provided via extended UML2 Activity diagrams. UML2 Activity diagrams provide a process modeling language that allows to model the control and object flows between different actions (for details on UML2 Activity diagrams, see [8]). In addition, the Object Constraint Language (OCL) [7] is applied in [9], [12] to formally define the semantics of the newly introduced UML elements and to ensure the consistency of the extended UML models.

A. Main Elements of the UML Extensions

This summary of the BusinessActivities and Duty-Nodes extensions focuses on the parts relevant for the subsequent definition of consistency requirements. For a complete definition of the semantics of all new modeling elements, please refer to the definition of the metamodel extensions in [9], [12]. A BusinessActivity is a special UML Activity (see Figure 1). It can include all elements available for ordinary UML Activities in addition to the newly introduced elements presented in [12]. A BusinessAction corresponds to a task and comprises all necessary permissions to perform the task. Furthermore,
ME and binding constraints can be defined on Business Actions (see [12] for further details). A Duty is used in a UML Activity Diagram to model that an action must be performed by a Subject which is assigned to this Duty. Each Duty is linked to a BusinessAction indicating that the Duty needs to be performed when carrying out the corresponding BusinessAction. In addition, Roles and Subjects are linked to Business Actions and Duties.

**B. SME, DME, RB, and SB Constraints for Duties**

Mutually exclusive duties result from the division of powerful responsibilities to prevent fraud and abuse and must never be assigned to the same subject (e.g., to sign a contract and to approve a contract). Duties can be defined as **statically mutual exclusive (SME)** (see, e.g., [3], [11], [12], [14]). A SME constraint is global with respect to all process instances in an information system. Therefore, two SME duties can never be assigned to the same subject or role. Two duties can also be defined as **dynamically mutual exclusive (DME)**. DME duties can be assigned to the same subject but must not be discharged by the same subject in the same process instance. SME and DME constraints can be used to enforce separation of duty constraints (see, e.g., [1], [4]). In contrast, binding of duty constraints specify that two bound duties must be performed by the same subject or role, i.e. **subject binding (SB)** or **role binding (RB)** of duties (see, e.g., [11], [12], [13]).

In this paper, we introduce the notion of SME, DME, RB, and SB constraints for Duties in Business Activities (see Figure 1). In [12], the definition of ME and binding constraints on Business Actions is motivated. Duties provide a more fine-grained view on a Business Action, as each Business Action may be associated with several Duties. The definition of constraints on Duty-level is reasonable in case the violation of a special Duty-related constraint would lead to, e.g., a penalty. Consider the following example (see Figure 2):

**Figure 1. UML2 metamodel extension**

An employee E records his or her working hours (BA1). A superior S has to control the employee’s working hour records every month (BA2). Task BA1 is associated with the following Duties: The employee must only record actual working hours – e.g., no lunch break (D1). Moreover, due to industrial safety regulations, the employee must only work at most ten hours per day (D2). Task BA2 is associated with the following Duties: The superior must sign the working hour records (D3). In addition, the superior is obliged to check if the records follow all industrial safety regulations (D4). Due to high workloads, employees in company X usually work overtime. For company X it is still important that employees follow the industrial safety regulations.

**Figure 2. Task- and duty-level constraints**

In [12] the definition of ME and binding constraints on Business Actions is motivated. Duties provide a more fine-grained view on a Business Action, as each Business Action may be associated with several Duties. The definition of constraints on Duty-level is reasonable in case the violation of a special Duty-related constraint would lead to, e.g., a penalty. Consider the following example (see Figure 2):
Otherwise, company X risks to be penalized for violating these regulations. There are two options to ensure that no employee can control his or her own working hour records. Firstly, it is possible to define a SME constraint on BA1 and BA2 (see Figure 2a). Therefore, a subject performing action BA1 is not allowed to execute BA2. Alternatively, the SME constraint can be defined on Duty-level between D2 and D4 (see Figure 2b). In this example, defining the SME constraint on Duty-level has the following advantages: It better reflects the actual intention of company X for separating these Duties. Furthermore, even if one of the Duties is assigned to another task, the constraint on D2 and D4 assures that they still cannot be discharged by the same subject.

If a constraint is defined on two Duties, this Duty-related constraint also applies for the corresponding Business Action, and vice versa. This is because each subject being assigned to a Business Action is necessarily also assigned to the associated Duties (see Section III-B and [9]). Thus, the simultaneous definition of a SME constraint on two Duties and a SME constraint on the two corresponding Business Actions is redundant, because the definition of a SME constraint on two Duties already implies that the associated Business Actions cannot be discharged by the same subject. Whether a constraint is defined on Business Action- or Duty-level depends on the required abstraction level. Definitions on Business Action-level are more abstract than definitions on Duty-level.

III. CONSISTENCY REQUIREMENTS

UML diagrams often can not provide all the relevant aspects of a specification. Therefore, there is a need to define additional constraints about the modeling elements. Such constraints are often described in natural language which may result in ambiguities. OCL has been developed to provide a formal language that remains easy to understand [7]. We apply the OCL to define additional constraints for Duties. In particular, the constraints defined in Sections III-A and III-B ensure the compliance of the designed process model including Duties with corresponding consistency requirements. In Section III-C, the consistency between the designed and the actually executed processes is addressed.

A. Constraints on Duties

Each SME, DME, RB, or SB constraint defined on Business Action-level also applies for the corresponding Duties, and vice versa. Thus, when defining SME, DME, RB, or SB constraints on Duties, constraints already defined for the associated Business Actions need to be considered to avoid inconsistent constraint definitions. Figure 3 shows conflicting combinations of constraints between Business Actions and Duties.

If a SME constraint has been defined on two Business Actions, the associated Duties cannot be defined as DME, role- or subject-bound (see Figure 3a and OCL constraint 1). For instance, the two Business Actions BA1 (record working hours) and BA2 (control working hour records) are defined as SME and therefore must always be performed by two different subjects. Each subject being assigned to a Business Action also needs to discharge the associated Duties. Thus, D2 (follow industrial safety regulations) and D4 (check if the records follow all industrial safety regulations) also must be discharged by two different subjects. Consequently, D2 and D4 cannot be defined as role- or subject-bound. Moreover, it is not possible to define a DME constraint on D2 and D4. DME Duties are executed by two different subjects during one process instance, while SME Business Actions need to be executed by two different subjects in all process instances (see also [12]).

Similarly, the simultaneous definition of a DME constraint on two Business Actions and a SME or a SB constraint on the associated Duties is not possible (see Figure 3b and OCL constraint 2). DME and SB constraints conflict, because a SB constraint defines that in the context of the same process instance the instances of two bound duties must be performed by the same subject. In contrast, a DME constraint defines that during one process instance the instances of two Business Actions must not be performed by the same subject [12].

If a RB constraint is defined on two Business Actions, they need to be performed by subjects being member of the same role. For instance, the role-bound tasks negotiate a contract (BA3) and sign a contract (BA4) need to be performed by subjects being member of the role bank clerk. Therefore, the associated Duties D5 (inform customer on associated risks) and D7 (approve the contract by a second bank clerk) are also discharged by members of the bank clerk role. Thus, the definition of a SME constraint on D5 and D7 is not possible, because SME Duties must not be performed by the same

Figure 3. Consistency of Duty Constraints
subject or role (see Figure 3c and OCL constraint 3). However, DME constraints and RB constraints do not conflict, as a DME constraint only requires different subjects discharging two Duties which also may be members of the same role (see [11] for details). Two Duties being assigned to two subject-bound Business Actions cannot be defined as mutually exclusive (see Figure 3d and OCL constraint 4). This is because two subject-bound Business Actions must be performed by the same subject, while ME duties must be discharged by two different subjects. Below we provide the respective OCL invariants which check the consistency of UML models including Duties with these requirements.

**OCL Constraint 1**: Two Duties being associated with two SME Business Actions must not be defined as DME, role- or subject-bound:

context BusinessAction inv:
self.staticExclusion->forAll(sme | self.duty->forAll(d1 | sme.duty->forAll(d2 |
   d1.staticExclusion->select(sme1 | sme1.name = d2.name)->isEmpty() and
d2.staticExclusion->select(sme2 | sme2.name = d1.name)->isEmpty() and
d1.roleBinding->forAll(rb1 | rb1.name = d2.name)->isEmpty() and
d2.roleBinding->forAll(rb2 | rb2.name = d1.name)->isEmpty() and
d1.subjectBinding->forAll(sb1 | sb1.name = d2.name)->isEmpty() and
d2.subjectBinding->forAll(sb2 | sb2.name = d1.name)->isEmpty()))

**OCL Constraint 2**: Two Duties being associated with two DME Business Actions must not be defined as SME or subject-bound:

context BusinessAction inv:
self.dynamicExclusion->forAll(dme | self.duty->forAll(d1 | dme.duty->forAll(d2 |
   d1.dynamicExclusion->select(dme1 | dme1.name = d2.name)->isEmpty() and
d2.dynamicExclusion->select(dme2 | dme2.name = d1.name)->isEmpty() and
   d1.roleBinding->select(rb1 | rb1.name = d2.name)->isEmpty() and
   d2.roleBinding->select(rb2 | rb2.name = d1.name)->isEmpty() and
   d1.subjectBinding->select(sb1 | sb1.name = d2.name)->isEmpty() and
   d2.subjectBinding->select(sb2 | sb2.name = d1.name)->isEmpty()))

**OCL Constraint 3**: Two Duties being associated with two role-bound Business Actions must not be defined as SME:

context BusinessAction inv:
self.staticExclusion->forAll(sb | self.duty->forAll(d1 | sb.duty->forAll(d2 |
   d1.staticExclusion->select(sme | sme1.name = d2.name)->isEmpty() and
d2.staticExclusion->select(sme | sme2.name = d1.name)->isEmpty() and
   d1.roleBinding->select(rb | rb1.name = d2.name)->isEmpty() and
   d2.roleBinding->select(rb | rb2.name = d1.name)->isEmpty()))

**OCL Constraint 4**: Two Duties being associated with two subject-bound Business Actions must not be defined as SME or DME:

context BusinessAction inv:
self.subjectBinding->forAll(sb | self.duty->forAll(d1 | sb.duty->forAll(d2 |
   d1.staticExclusion->select(sme | sme1.name = d2.name)->isEmpty() and
   d2.staticExclusion->select(sme | sme2.name = d1.name)->isEmpty()))

Besides checking the consistency between constraints defined on Business Actions and constraints defined on Duties (see Section III-A), we also need to ensure the consistency for all assignments of Duties. In a business process context, each duty is associated with a particular task [9]. Thus, each Duty is discharged when performing the respective Business Action (see Figure 4a and OCL constraint 5). Consequently, two or more Duties assigned to the same Business Action must not be mutually exclusive (see OCL constraint 6). Otherwise, a subject which is assigned to a Business Action is unable to discharge all associated Duties. Figure 4b shows conflicting Duty assignments, because the two ME Duties $D_1$ and $D_2$ cannot be assigned to the same Business Action. Moreover, a Role being assigned to a Duty must also be authorized to perform the respective Business Action, and vice versa (see Figure 4c and OCL constraints 7 and 8). Two SME Duties must not be discharged by the same subject or role. Thus, two SME Duties $D_1$ and $D_2$ must not be assigned to the same Role $R$ (see Figure 4d and OCL constraint 9). Otherwise, each subject being member of $R$ has to perform two SME Duties. Moreover, no subject must be assigned to two Roles $R_1$ and $R_2$ which are assigned to two SME Duties (see Figure 4e and OCL constraint 10).

**OCL Constraint 5**: Each Duty must be associated with a certain Business Action:

context Duty inv:
self.businessAction->notEmpty()

**OCL Constraint 6**: Two SME or DME Duties must never be assigned to the same Business Action:

context BusinessAction inv:
self.duty->forAll(d1, d2 | d1.staticExclusion->select(sme | sme1.name = d2.name)->isEmpty() and
d2.staticExclusion->select(sme2 | sme2.name = d1.name)->isEmpty()))

Figure 4. Consistency of Duty Assignments
self.roleToSubjectAssignment->forAll(rsa1, rsa2 | context Subject inv:
assigned to two Roles which own SME Duties
executingSubject
sActivity instance. The attributes
corresponds to the associated Duty's responsibleSubject
if a Business Action's executingSubject attribute
charges the associated Duties. This is achieved by check-
if a subject being assigned to a Business Action also dis-
ment of Duties at runtime. OCL constraint 11 monitors
following consistency checks ensure the proper enforce-
instances comply with all consistency requirements. The
In addition, we also need to secure that runtime process
consistency of elements and relationships at design-time.

C. Runtime Allocations

All OCL constraints presented above refer to the
consistency of elements and relationships at design-time. In
addition, we also need to secure that runtime process
instances comply with all consistencies requirements. The
following consistency checks ensure the proper enforce-
munity. OCL constraint 11 monitors
if a subject being assigned to a Business Action also dis-
charges the associated Duties. This is achieved by check-
if a Business Action's executingSubject attribute
in case they are included in the same Business-
Activity instance. The attributes executingSubject
and responsibleSubject determine the Subjects that execute
a particular Business Action or Duty instance, respectively
(see [9], [12]). To enforce constraints on Duties, two
SME Duties must always be executed by two different
subjects, while two DME Duties only need two different
responsible subjects if they are included in the same
BusinessActivity (see OCL constraints 12 and 13). In
contrast, two role- or subject-bound Duties must be
associated with the same role or subject, respectively
(see OCL constraints 14 and 15).

OCL Constraint 11: In the same BusinessActivity
instance, a Duty's responsible Subject corresponds to
the associated Business Action's executing subject:

OCL Constraint 7: Each Role assigned to a Duty
is also assigned to the associated Business Action:

OCL Constraint 8: Each Role assigned to a
Business Action is also assigned to the corresponding
Duties:

OCL Constraint 9: A Role must never be assigned
to two SME Duties:

OCL Constraint 10: A Subject must never be assigned
to two Roles which own SME Duties

context BusinessAction inv:
self.duty->forAll(d |

d.role->exists(r1 | self.role->exists(r2 | r2.name = r1.name)))

OCL Constraint 12: The instances of two SME
Duties must never have the same responsible subject:

context Duty inv:
self.staticExclusion->forAll(sme |
sme.instanceSpecification->forAll(i |

i.slot->select(si1 |

si1.definingFeature.name = associatedProcessInstance and

i.slot->select(si2 |

si2.definingFeature.name = responsibleSubject and

self.businessAction.instanceSpecification->forAll(j |

j.slot->select(sj1 |

sj1.definingFeature.name = owningProcessInstance and

j.slot->select(sj2 |

sj2.definingFeature.name = executingSubject and

si1.value = sj1.value implies

si2.value = sj2.value ))))))

OCL Constraint 13: In each BusinessActivity
instance, the instances of two DME Duties must be
executed by two distinct subjects:

context Duty inv:
self.dyna}\text{micExclusion->forAll(dme |
sme.instanceSpecification->forAll(i |

i.slot->forAll(is |

is.definingFeature.name = responsibleSubject and

j.slot->forAll(js |

js.definingFeature.name = responsibleSubject implies

not is.value = js.value )))}

OCL Constraint 14: Instances of role-bound Duties
must be associated with the same responsible role.

OCL Constraint 15: Instances of subject-bound
Duties must have the same responsible subject.
IV. Related Work

Many contributions discuss constraint specifications when defining SOD and BOD constraints. Our approach complements existing approaches by considering SOD and BOD constraints for duties in a process-related RBAC context. In [1], the RCL 2000 language for the specification of role-based authorization constraints is introduced. Separation of duty constraints can also be expressed in RCL 2000. In addition, the authors discuss different conflicts that might occur when specifying constraints via RCL 2000. In [2], a language for expressing SOD constraints and respective consistency checks for these constraints in a workflow-context is presented. Botha and Eloff [3] discuss possible conflicts of static and dynamic SOD constraints in workflows. Tan et al. [13] define a model for constrained workflow systems, including SOD and BOD constraints. They discuss several consistency issues regarding these constraints and define formal consistency rules. In [6], RBAC/Web is introduced, which provides a model and implementation for RBAC in Web servers and discuss consistency issues of SOD constraints in role-hierarchies.

V. Conclusion

SME, DME, RB, and SB constraints can be defined for duties to regulate which role/subject is allowed to execute a particular duty. In this paper, we extended existing approaches for modeling security aspects in business process models by considering mutual exclusion and binding constraints for duties in a process-related RBAC context. Moreover, we formally defined consistency checks to ensure the design-time and runtime compliance of UML models with the respective requirements.

REFERENCES


