Consistency Checks for Duties in Extended UML2 Activity Models

***Abstract*—Process-aware information systems support the execution of business processes. In this context, organizations require the precise specification of security policies that govern the behavior of subjects in the systems. 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 (see, e.g., [12]). 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], [12], [13]. In a business process context, duties are often associated with a particular task [11] (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], [16], [18]. ME and binding constraints are usually specified on role- or task-level (see, e.g., [1], [14], [15], [17]). 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 [15], 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 for tasks. In [11], 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 *mutual exclusion and binding constraints for duties* and by formally defining respective *consistency checks* (see Figure 1). Defining ME and binding constraints on task- and duty-level provides two different abstraction levels

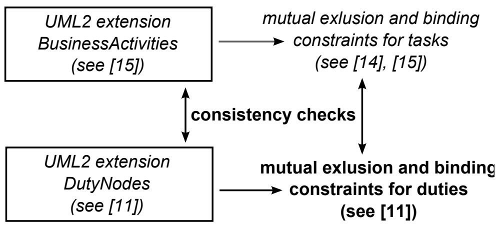
UML2 extension BusinessActivities (see [15])
mutual exlusion and binding constraints for tasks (see [14], [15])
consistency checks
UML2 extension DutyNodes (see [11])
mutual exlusion and binding constraints for duties (see [11])

Figure 1: Concept overview

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 the expected with the actually executed behavior, i.e. consistency between design-time and runtime models.

The remainder of this paper is structured as follows. Section II summarizes the combined UML2 extension [11], [15] 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 [11], [15], 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. In particular, a UML2 Activity models a process while tasks included in an Activity are modeled via Actions (for details on UML2 Activity diagrams, see [10]). Providing modeling support for security aspects via a standard notation like UML is intended to bridge the communication gap between software engineers, security experts, and non-technical stakeholders (see, e.g., [8]). In addition, the Object Constraint Language (OCL) [9] is applied in [11], [15] 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 *DutyNodes* 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 [11], [15].

A *BusinessActivity* is a special UML Activity (see Figure 2). It can include all elements available for ordinary UML Activities in addition to the newly introduced elements presented in [15]. A *BusinessAction*

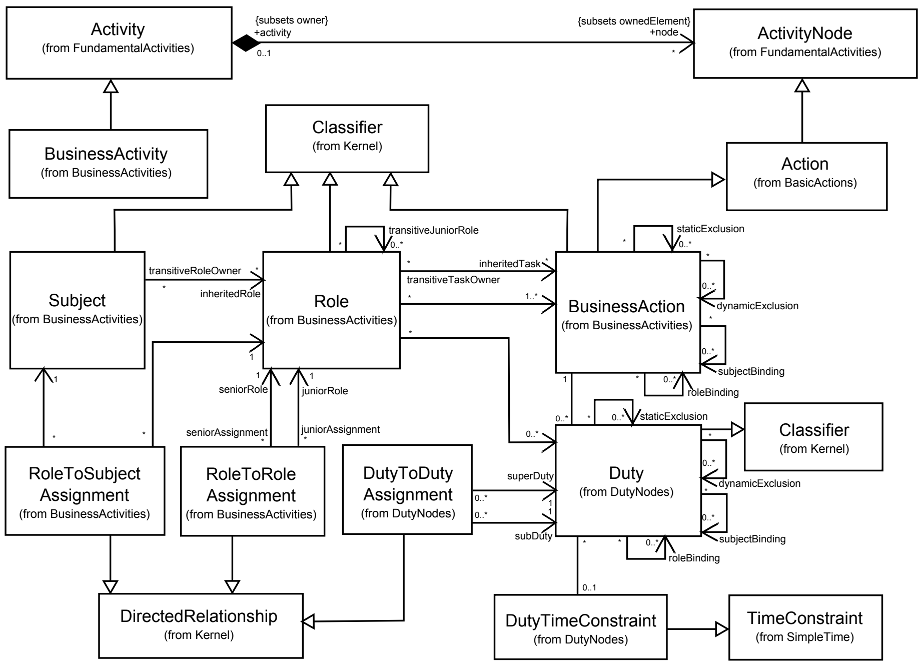

Figure 2: UML2 metamodel extension

corresponds to a task and comprises all necessary permissions to perform the task. Furthermore, ME and binding constraints can be defined for BusinessActions (see [15] 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 [11]. Each Duty is linked to a BusinessAction indicating that the Duty needs to be performed when carrying out the corresponding BusinessAction. Similar to BusinessActions, ME and binding constraints can also be defined for *Duties*. In addition, *Roles* and *Subjects* are linked to BusinessActions and Duties (see Figure 2).

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

Mutual 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], [7], [14], [15], [17]). 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., [14], [15], [16]).

In this paper, we introduce the notion of SME, DME, RB, and SB constraints for Duties in Business Activities (see Figure 2). In [15], the definition of SME, DME, RB, and SB constraints for 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. Therefore, 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 3):

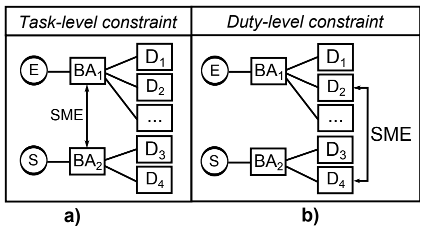

Figure 3: Task- and duty-level constraints

An employee E records his or her working hours (BA_1). A superior S has to control the employee's working hour records every month (BA_2). Task BA_1 is associated with the following Duties: The employee must only record actual working hours – e.g., no lunch break (D_1). Moreover, due to industrial safety regulations, the employee must only work at most ten hours per day (D_2). Task BA_2 is associated with the following Duties: The superior must sign the working hour records (D_3). In addition, the superior is obliged to check if the records follow all industrial safety regulations (D_4). 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. 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 between BA_1 and BA_2 (see Figure 3a). Therefore, a subject performing action BA_1 is not allowed to execute BA_2. Alternatively, the SME constraint can be defined on Duty-level between D_2 and D_4 (see Figure 3b). 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 between D_2 and D_4 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 [11]). Thus, the simultaneous definition of a SME constraint for two Duties and a SME constraint for 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 [9]. OCL is a pure specification language. Thus, evaluating an OCL expression does not change anything in the model. 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. These OCL invariants therefore ensure the correct design of UML models using the modeling elements proposed in [11], [15]. In Section III-C, the consistency between the designed and the actually executed processes is addressed. These OCL invariants ensure the correct enforcement of the corresponding process instances.

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 for Duties, constraints already defined for the associated Business Actions need to be considered to avoid inconsistent constraint definitions. Figure 4 shows conflicting combinations of constraints between Business Actions and Duties.

If a SME constraint has been defined between two Business Actions, the associated Duties cannot be defined as DME, role- or subject-bound (see Figure 4a and OCL constraint 1). For instance, the two Business Actions BA_1 (record working hours) and BA_2 (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, D_2 (follow industrial safety regulations) and D_4 (check if the records follow all industrial safety regulations) also must be discharged by two different subjects. Consequently, D_2 and D_4 cannot be defined as role- or subject-bound. Moreover, it is not possible to define a DME constraint for D_2 and D_4. 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. Therefore, two Business Actions and its associated Duties can either be statically or dynamically mutual exclusive (see also [15]). The separate definition of a SME constraint on Duty-level is redundant but possible (e.g. to ensure that these Duties can still not be assigned to the same subject if they are associated with other Business Actions).

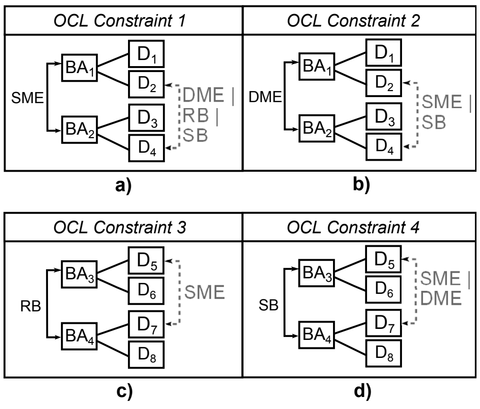

Figure 4: Consistency of Duty Constraints

Similarly, the simultaneous definition of a DME constraint for two Business Actions and a SME or a SB constraints for the associated Duties is not possible (see Figure 4b 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 [15].

If a RB constraint is defined for 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 (BA_3) and sign a contract (BA_4) need to be performed by subjects being member of the role bank clerk. Therefore, the associated duties D_5 (inform customer on associated risks) and D_7 (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 for D_5 and D_7 is not possible, because SME Duties must not be performed by the same subject or role (see Figure 4c 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 member of the same role (see [14] for details).

Two Duties being assigned to two SB Business Actions cannot be defined as mutually exclusive (see Figure 4d and OCL constraint 4). This is because two SB Business Actions must be performed by the same subject, while a ME constraint defines that the associated 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.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** 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.staticExclusion->select(sme1 |
        sme1.name = d2.name)->isEmpty() and
      d2.staticExclusion->select(sme2 |
        sme2.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.roleBinding->forAll(rb |
  self.duty->forAll(d1 |
    rb.duty->forAll(d2 |
      d1.staticExclusion->select(sme1 |
        sme1.name = d2.name)->isEmpty() and
      d2.staticExclusion->select(sme2 |
        sme2.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(sme1 |
        sme1.name = d2.name)->isEmpty() and
      d2.staticExclusion->select(sme2 |
        sme2.name = d1.name)->isEmpty() and
      d1.dynamicExclusion->select(dme1 |
        dme1.name = d2.name)->isEmpty() and
      d2.dynamicExclusion->select(dme2 |
        dme2.name = d1.name)->isEmpty() )))
```

B. Assignments of Duties

Duties are directly assigned to Business Actions and are transitively assigned to Roles and Subjects (see Figure 2). Besides checking the consistency between constraints defined for Business Actions and constraints defined for Duties (see Section III-A), we also need to ensure the consistency for all assignments of Duties.

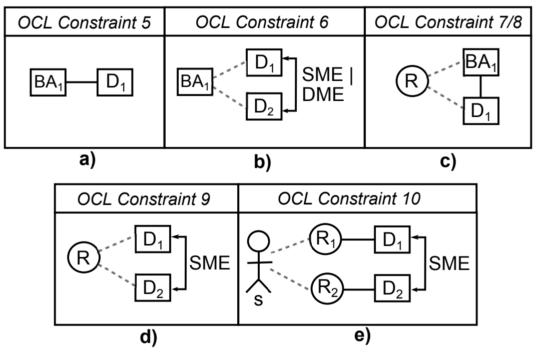

Figure 5: Consistency of Duty Assignments

In a business process context, each duty is associated with a particular task [11]. Thus, each Duty is discharged when performing the respective Business Action (see Figure 5a 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 5b 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 5c and OCL constraints 7 and 8). For instance, if a superior is obliged to check if the employees' records follow all industrial safety regulations, the superior also needs to be assigned to the corresponding Business Action authorizing him to fulfill this Duty.

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 5d 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 5e and OCL constraint 10). Otherwise, subject S would subsequently be obliged to discharge two SME Duties.

***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 |
     sme.name = d2.name)->isEmpty() and
     d1.dynamicExclusion->select(dme |
       dme.name = d2.name)->isEmpty() ))
```

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

```
context Duty inv:
self.role->exists(r1 |
  self.businessAction.role->exists(r2 |
    r1.name = r2.name))
```

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

Duties:

```
context BusinessAction inv:
self.duty->forall(d |
  d.role->exists(r1 |
    self.role->exists(r2 |
      r1.name = r2.name)))
```

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

```
context Role inv:
self.duty->forAll(d1, d2 |
  d1.staticExclusion->select(sme |
    sme.name = d2.name)->isEmpty()
```

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

```
context Subject inv:
self.roleToSubjectAssignment->forAll(rsa1, rsa2 |
  rsa1.role.duty->forAll(d1 |
    rsa2.role.duty->forAll(d2 |
      d1.staticExclusion->select(sme|
        sme.name = d2.name)->isEmpty() )))
```

C. Runtime Assignments

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 consistency requirements. The following consistency checks ensure the proper enforcement of Duties at runtime. OCL constraint 11 monitors if a subject being assigned to a Business Action also discharges the associated Duties. This is achieved by checking if a BusinessAction's executingSubject attribute corresponds to the associated Duty's responsibleSubject attribute in case they are included in the same BusinessActivity instance. The attributes *executingSubject* and *responsibleSubject* determine the Subjects that execute a particular Business Action or Duty instance, respectively (see [11], [15]).

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 13 and 14). In contrast, two RB and SB Duties must be associated with the same role or subject, respectively (see OCL constraints 15 and 16).

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

```
context Duty inv:
self.instanceSpecification->forAll(i |
  i.slot->select(si1 |
    si1.definingFeature.name = associatedProcessInstance
    i.slot->select(si2 |
      si2.definingFeature.name = responsibleSubject
      self.businessAction.instanceSpecification->forAll(j |
        j.slot->select(sj1 |
          sj1.definingFeature.name = owningProcessInstance
          j.slot->select(sj2 |
            sj2.definingFeature.name = executingSubject
            if (si1.value = sj1.value) then
               (si2.value = sj2.value)
            else true endif ))))))
```

***OCL Constraint** 12:* In the same BusinessActivity instance, a Duty's responsible Role corresponds to the associated Business Action's executing Role:

```
context Duty inv:
self.instanceSpecification->forAll(i |
  i.slot->select(si1 |
    si1.definingFeature.name = associatedProcessInstance
    i.slot->select(si2 |
      si2.definingFeature.name = responsibleRole
      self.businessAction.instanceSpecification->forAll(j |
        j.slot->select(sj1 |
          sj1.definingFeature.name = owningProcessInstance
          j.slot->select(sj2 |
            sj2.definingFeature.name = executingRole
            if (si1.value = sj1.value) then
               (si2.value = sj2.value)
            else true endif ))))))
```

***OCL Constraint** 13:* To enforce SME constraints on Duties, we specify that the instances of two SME Duties must never have the same responsible subject:

```
context Duty inv:
self.staticExclusion->forAll(sme |
  self.instanceSpecification->forAll(i |
    sme.instanceSpecification->forAll(j |
      i.slot->forAll(is |
        j.slot->forAll(js |
          if is.definingFeature.name = responsibleSubject and
             js.definingFeature.name = responsibleSubject then
             not (is.value = js.value)
          else true endif )))))
```

***OCL Constraint** 14:* To enforce DME constraints on Duties, for each BusinessActivity the instances of two DME Duties which are included in this BusinessActivity must be executed by two distinct subjects:

```
context Duty inv:
self.dynamicExclusion->forAll(dme|
   self.instanceSpecification->forAll(i|
      dme.instanceSpecification->forAll(j|
         i.slot->select(si1|
            si1.definingFeature.name = associatedProcessInstance
            j.slot->select(sj1|
               sj1.definingFeature.name = associatedProcessInstance
               i.slot->select(si2|
                  si2.definingFeature.name = responsibleSubject
                  j.slot->select(sj2|
                     sj2.definingFeature.name = responsibleSubject
                     if (si1.value = sj1.value) then
                        not (si2.value = sj2.value)
                     else true endif )))))))
```

***OCL Constraint** 15:* To enforce RB constraints on Duties, instances of these Duties must always be associated with the same responsible role.

```
context Duty
inv: self.roleBinding->forAll(rbt|
       self.instanceSpecification->forAll(i|
         rbt.instanceSpecification->forAll(j|
           i.slot->select(si1|
             si1.definingFeature.name = associatedProcessInstance
             j.slot->select(sj1|
               sj1.definingFeature.name = associatedProcessInstance
                 i.slot->select(si2|
                   si2.definingFeature.name = responsibleRole
                   j.slot->select(sj2|
                     sj2.definingFeature.name = responsibleRole
                     if (si1.value = sj1.value) then
                        (si2.value = sj2.value)
                     else true endif )))))))
```

***OCL Constraint** 16:* To enforce SB constraints on Duties, instances of these Duties must always be associated with the same responsible subject.

```
context Duty
inv: self.subjectBinding->forAll(sbt|
       self.instanceSpecification->forAll(i|
         sbt.instanceSpecification->forAll(j|
           i.slot->select(si1|
             si1.definingFeature.name = associatedProcessInstance
             j.slot->select(sj1|
               sj1.definingFeature.name = associatedProcessInstance
                 i.slot->select(si2|
                   si2.definingFeature.name = responsibleSubject
                   j.slot->select(sj2|
                     sj2.definingFeature.name = responsibleSubject
if (si1.value = sj1.value) then
   (si2.value = sj2.value)
else true endif )))))))
```

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. [16] 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 consistency requirements.

References

[1] G. Ahn and R. Sandhu. Role-based Authorization Constraints Specification. *ACM Transactions on Information and System Security (TISSEC)*, 3(4), November 2000.

[2] E. Bertino, E. Ferrari, and V. Atluri. The specification and enforcement of authorization constraints in workflow management systems. *ACM Transactions on Information and System Security (TISSEC)*, 2(1), 1999.

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

[4] D. D. Clark and D. R. Wilson. A comparison of commercial and military computer security policies. In *IEEE Symposium of Security and Privacy*, April 1987.

[5] J. Cole, J. Derrick, Z. Milosevic, and K. Raymond. Author Obliged to Submit Paper before 4 July: Policies in an Enterprise Specification. In *Proceedings of the International Workshop on Policies for Distributed Systems and Networks*, January 2001.

[6] D. Ferraiolo, J. Barkley, and D. Kuhn. A Role-Based Access Control Model and Reference Implementation within a Corporate Intranet. *ACM Transactions on Information and System Security (TISSEC)*, 2(1), February 1999.

[7] J. Mendling, K. Ploesser, and M. Strembeck. Specifying Separation of Duty Constraints in BPEL4People Processes. In *Proc. of the 11th International Conference on Business Information Systems*, volume 7 of *Lecture Notes in Business Information Processing*, May 2008.

[8] H. Mouratidis and J. Jürjens. From Goal-Driven Security Requirements Engineering to Secure Design. *International Journal of Intelligent Systems*, 25(8), 2010.

[9] OMG. Object Constraint Language Specification. available at: http://www.omg.org/technology/documents/formal/ocl.htm, February 2010. Version 2.2, formal/2010-02-01, The Object Management Group.

[10] OMG. Unified Modeling Language (OMG UML): Superstructure. available at: http://www.omg.org/technology/documents/formal/uml.htm, May 2010. Version 2.3, formal/2010-05-03, The Object Management Group.

[11] S. Schefer and M. Strembeck. Modeling Process-Related Duties with Extended UML Activity and Interaction Diagrams. In *Proc. of the International Workshop on Flexible Workflows in Distributed Systems, Workshops der wissenschaftlichen Konferenz Kommunikation in verteilten Systemen (WowKiVS)*, volume 37 of *Electronic Communications of the EASST*, March 2011.

[12] M. S. Sloman. Policy Driven Management for Distributed Systems. *Journal of Network and Systems Management*, 2(4), 1994.

[13] M. Strembeck. Embedding Policy Rules for Software-Based Systems in a Requirements Context. In *Proc. of the 6th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY)*, June 2005.

[14] M. Strembeck and J. Mendling. Generic Algorithms for Consistency Checking of Mutual-Exclusion and Binding Constraints in a Business Process Context. In *Proc. of the 18th International Conference on Cooperative Information Systems (CoopIS)*, volume 6426 of *Lecture Notes in Computer Science (LNCS)*. Springer Verlag, October 2010.

[15] M. Strembeck and J. Mendling. Modeling Process-related RBAC Models with Extended UML Activity Models. *Information and Software Technology*, 53(5), 2011. DOI: 10.1016/j.infsof.2010.11.015.

[16] K. Tan, J. Crampton, and C. A. Gunter. The Consistency of Task-Based Authorization Constraints in Workflow Systems. In *Proceedings of the 17th IEEE workshop on Computer Security Foundations*, June 2004.

[17] R. K. Thomas and R. S. Sandhu. Task-based Authorization Controls (TBAC): A Family of Models for Active and Enterprise-oriented Authorization Management. In *In Proceedings of the IFIP WG11.3 Workshop on Database Security, Lake Tahoe*, August 1997.

[18] J. Wainer, P. Barthelmess, and A. Kumar. W-RBAC - A Workflow Security Model Incorporating Controlled Overriding of Constraints. *International Journal of Cooperative Information Systems*, 12(4), 2003.