Modeling Context-Aware RBAC Models for Business Processes in Ubiquitous Computing Environments

Sigrid Schefer-Wenzl, Mark Strembeck
Institute for Information Systems and New Media
Vienna University of Economics and Business (WU Vienna), Austria
firstname.lastname@wu.ac.at

***Abstract*—With ubiquitous computing technologies, business processes become more mobile and distributed and are executed in varying contexts. Context-aware access control mechanisms are an important prerequisite to protect sensitive data and services in secure ubiquitous computing environments. In an IT-supported workflow, process-related *context constraints* are a means to consider context information in access control decisions. A context constraint specifies that certain conditions must be fulfilled to permit the execution of a particular task. However, standard process modeling languages do not support the notion of context constraints in business processes. In this paper, we integrate context constraints with process-related role-based access control (RBAC) models and thereby support context-dependent task execution.**

***Keywords*-access control; business process modeling; context constraints; UML;**

I. Introduction

Business processes consist of tasks which are performed to reach certain corporate goals (see, e.g., [2]). In recent years, the rising popularity of ubiquitous computing technologies led to increasingly flexible business environments. Therefore, ubiquitous technologies also have a strong impact on corresponding business processes, such as inventory management or sales-oriented processes [20]. Many processes become more mobile, flexible, and distributed and are executed in different contexts [19]. They may be executed at varying places, times, and by different people or devices. To protect sensitive data and services, context-aware security mechanisms have been identified as a prerequisite for secure ubiquitous computing (see, e.g., [11], [12]). One important security aspect is *access control* which deals with the elicitation, specification, maintenance, and enforcement of authorization policies in software-based systems (see, e.g., [22]). In this paper, we focus on context-aware access control in a business process context.

In an IT-supported workflow, process-related *context constraints* are a means to consider context information in access control decisions (see, e.g., [27]). Typical examples for context constraints in organizational settings regard the temporal or spatial context of task execution, user-specific attributes, or the task execution history of a user (see, e.g., [7]). Yet, standard process modeling languages, such as BPMN [17] or UML Activity diagrams [18], do not provide native language support to model process-related context constraints.

One *objective* of our research is to define process-related context constraints via native modeling language constructs. Usually, process models focus on the process-flow perspective and are decoupled from access control-relevant context information. In practice, the lack of native modeling constructs for context constraints results in a number of workarounds [19], for example: Corresponding context constraints become part of the control flow by including several decision nodes, such as "check if location is Vancouver" or "check if user is registered". As a result, process models become larger and more complex to read. Alternatively, multiple process models for different contextual scenarios are designed, leading to highly redundant models.

In recent years, role-based access control (RBAC) [8], [21] has developed into a de facto standard for access control in both, research and industry. In RBAC, roles correspond to different job-positions and scopes of duty within a particular organization or information system [25]. Access permissions are assigned to roles according to the tasks a role has to accomplish. Human users and other active entities are assigned to roles. Thereby, each subject acquires all permissions necessary to fulfill its duties via his/her role memberships. In addition, RBAC supports the definition of context constraints on various parts of an RBAC model (see, e.g., [9], [27]).

In this paper, we integrate the notion of context constraints into process-related RBAC models [26] and thereby support context-dependent task execution. We use a generic context concept where constraints can be defined for any type of context information that can be tracked in an information system. To achieve this, we formally embed RBAC context constraints into a business process context. Based on the formal approach, we define a corresponding extension for a standard process modeling language. In particular, we define a UML extension for the integrated modeling of processes, RBAC concepts, and context constraints via extended UML2 Activity diagrams. Our approach supports the complete and correct mapping of process definitions and related context-aware access control policies to corresponding software systems. This is essential to assure consistency between modeling-level specifications and software systems enforcing the respective access control policies. Tool-support for the enforcement of context-aware RBAC models is integrated into the xoRBAC software component [27].

This paper is structured accordingly. Section II introduces our formal metamodel for context-aware RBAC models. The corresponding UML2 extension is defined in Section III. Next, Section IV discusses related work, before Section V concludes the paper.

II. Context-Aware RBAC Models

Each *task* in a process (e.g., to sign a contract) is typically associated with certain access permissions (e.g., to read and write the contract document). Therefore, *subjects* participating in a workflow, i.e. human users or software-agents, must be authorized to perform the tasks needed to complete the process (see, e.g., [9], [15]). A *role* is an abstraction containing the tasks and associated permissions of a certain subject-type [25]. The left-hand side of Figure 1 illustrates the essential relations between these elements in process-related RBAC models (see [26]).

Tasks defined in business processes are always performed within a certain context. These contextual attributes, e.g., time, location, or the executing subject, may influence access control decisions. Thus, depending on the context, different authorization rules might apply for executing a particular task. A *context constraint* is a modeling-level concept defining that certain contextual attributes must meet certain predefined conditions to permit the execution of a specific task [27]. In particular, context constraints consist of context conditions that include context attributes and context functions: A *context attribute* represents a certain property of the environment whose actual value might change dynamically (e.g., time, date, location). For each context attribute, a *context function* exists that can obtain the current value of a specific context attribute (e.g., date() returns the current date). A *context condition* is a boolean expression that restricts the permitted values of a context attribute (e.g., date $>$ 01/01/2012).

In this paper, we focus on the modeling of context-aware RBAC models for business processes. For this purpose, we formally embed context constraints into a business process modeling context. In Figure 1, the metamodel for process-related RBAC models (see [26]) is extended with context constraints. A process-related context constraint is associated with a task and one or more context conditions. All context conditions must evaluate to true in order for the context constraint to be fulfilled. Each context condition consists of an operator, e.g., an infix operator, such as $=, \geq, >, <, \leq, \neq$, and two or more operands. Hereby, one of the operands refers to a certain context attribute and the other operands are either context attributes or constant values (e.g., currentLocation $=$ Vancouver). Moreover, the type and range of values for each operand and for each operator is determined

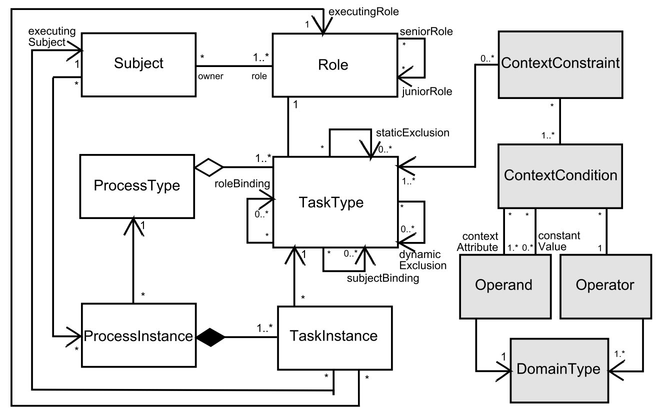

Figure 1. Extended metamodel for context-aware RBAC models

via its domain type (e.g., boolean, date, integer). Definition 1 formally specifies the essential elements of the extended metamodel and their basic interrelations.

Definition 1: **(Context-Aware Business Activity RBAC Model).** Let cBRM = (E,Q,D,CX) be a Context-Aware Business Activity RBAC Model, where E refers to the pairwise disjoint sets of the metamodel, Q to mappings that establish relationships, D to binding and mutual exclusion constraints, and CX to mappings for context constraints.

The **sets E of the Context-Aware Business Activity RBAC Model** are:

- An element of S is called *Subject*. $S \neq \emptyset$.
- An element of R is called *Role*. $R \neq \emptyset$.
- An element of P_T is called *Process Type*. $P_T \neq \emptyset$.
- An element of P_I is called *Process Instance*.
- An element of T_T is called *Task Type*. $T_T \neq \emptyset$.
- An element of T_I is called *Task Instance*.
- An element of CA is called *Context Attribute*. $CA \neq \emptyset$.
- An element of CV is called *Constant Value*. $CV \neq \emptyset$.
- An element of OD is called *Operand*. $OD \neq \emptyset$.
- An element of DM is called *Domain Type*. $DM \neq \emptyset$.
- An element of OT is called *Operator*. $OT \neq \emptyset$.
- An element of CD is called *Context Condition*.
- An element of CC is called *Context Constraint*.

For the mappings of the *Business Activity RBAC Model* ($Q = rh \cup rsa \cup es \cup er \cup tra \cup ti$, $D = sb \cup rb \cup sme \cup dme$) see [26]. Below, we define the additional **mappings for context constraints**: $CX = dm_{od} \cup dm_{ot} \cup od_{cd} \cup ot_{cd} \cup cond \cup fulfilled_{CD} \cup linked_{CD} \cup fulfilled_{CC} \cup linked_{CC}$ ($\mathcal{P}$ refers to the power set):

1) *An operand is either a context attribute or a constant value:* $\forall od \in OD : od \in CA \vee od \in CV$.
2) *Each operand has a certain domain type which determines the type and range of values for this operand (e.g., boolean, date, integer). For example, the context attribute* currentDate *has the domain type* date. *Note that each operand has exactly one domain type:* The

mapping $dm_{od} : OD \mapsto DM$ is called **operand-domain type**. For $dm_{od}(od) = dm$, we call $od \in OD$ *operand* and $dm \in DM$ the *domain type specified for an operand*.

3) *In a context condition, an operator is applied to the operands. In contrast to operands, operators may be linked to multiple domain types. Consider, e.g., the* lower-eqal ($\leq$) *operator which may be used to compare numbers or dates and thus is linked to the domains: integer, real, date, etc. Formally:* The mapping $dm_{ot} : OT \mapsto \mathcal{P}(DM)$ is called **operator-domain types**. For $dm_{ot}(ot) = DM_{ot}$, we call $ot \in OT$ *operator* and $DM_{ot} \subseteq DM$ the *set of domain types specified for an operator*.
4) *Context conditions are predicates that consist of operands and operators. Each context condition contains one operator and the number of operands required by this operator. In the example condition* "age > 18", *the context attribute* age *and the constant value* 18 *are the operands, whereas* $>$ *is the operator.*
   a) The mapping $od_{cd} : CD \mapsto \mathcal{P}(OD)$ is called **context condition operands**. For $od_{cd}(cd) = OD_{cd}$, we call $cd \in CD$ *context condition* and $OD_{cd} \subseteq OD$ is the *set of operands included in this condition*.
   b) The mapping $ot_{cd} : CD \mapsto OT$ is called **context condition operator**. For $ot_{cd}(cd) = ot$, we call $cd \in CD$ *context condition* and $ot \in OT$ is the *operator included in this condition*.
   c) The mapping $cond : (\mathcal{P}(OD) \times OT) \mapsto CD$ is called **context condition**. For $cond(OD_{cd}, ot) = cd$, we call $OD_{cd} \subseteq OD$ the *set of operands included in this condition*, $ot \in OT$ is called *operator*, and $cd \in CD$ is called *context condition*.
5) *Within each context condition, all operands must have the same domain type:* $\forall cd_x \in CD : \forall od_x, od_y \in od_{cd}(cd_x) : dm_{od}(od_x) = dm_{od}(od_y)$
6) *Within each context condition, the domain type of the operands must correspond to the domain type of the operator. Otherwise, the operator cannot be applied on the operands:* $\forall cd_x \in CD : \forall od_x \in od_{cd}(cd_x), ot \in ot_{cd}(cd_x) : dm_{od}(od_x) \in dm_{ot}(ot)$
7) *A context condition must contain at least one context attribute:* $\forall cd_x \in CD : \exists od_x \in od_{cd}(cd_x) : od_x \in CA$
8) *Context conditions are boolean expressions that are fulfilled iff the operator applied to the corresponding operands evaluates to true:* The mapping $fulfilled_{CD} : CD \mapsto BOOLEAN$ is called **context condition fulfillment**. For $fulfilled_{CD}(cd) = boolean$, we call $cd \in CD$ *context condition*. The mapping follows a two-valued logic returning exactly one truth value. Thus, the $fulfilled_{CD}$ mapping returns *true* iff the operator applied on the corresponding operands evaluates to true. Otherwise, it returns *false*.
9) *A context constraint is linked to one or more context conditions:* The mapping $linked_{CD} : CC \mapsto \mathcal{P}(CD)$ is called **context condition to constraint linkage**. For $linked_{CD}(cc) = CD_{CC}$, we call $cc \in CC$ *context constraint* and $CD_{CC} \subseteq CD$ the *set of conditions linked to this context constraint*.
10) *A context constraint is fulfilled iff all linked conditions evaluate to true:* The mapping $fulfilled_{CC} : CC \mapsto BOOLEAN$ is called **context constraint fulfillment**. For $fulfilled_{CC}(cc) = boolean$, we call $cc \in CC$ *context constraint*. The mapping follows a two-valued logic returning exactly one truth value. Thus, the $fulfilled_{CC}$ mapping returns *true* iff all conditions linked to the context constraint are true. Otherwise, it returns *false*. $fulfilled_{CC}(cc_x) = true \Leftrightarrow \forall cd_x \in linked_{CD}(cc_x) : fulfilled_{CD}(cd_x) = true$.
11) *Context constraints are linked to task types:* The mapping $linked_{CC} : T_T \mapsto \mathcal{P}(CC)$ is called **context constraint to task linkage**. For $linked_{CC}(t) = CC_T$ we call $t \in T_T$ *constrained task* and $CC_T \subseteq CC$ the set of *context constraints linked to this task*.

Definition 2 specifies the process flow model for Context-Aware Business Activities as a labeled transition system (see, e.g., [6], [14]).

Definition 2: **(Context-Aware Business Activity Process Flow Model)**. A Process Flow Model $PFM = (N, A)$ where $N = T_T \cup C_F \cup C_J \cup C_D \cup C_M \cup \{start, end\}$ refers to pairwise disjoint sets and $A \subseteq N \times N$.

An element of N is called $node$ and an element of A is called arc. Elements of T_T are called $task\ types$. An element of $C = C_F \cup C_J \cup C_D \cup C_M$ is called control node. An element of C_F is called fork, an element of C_J join, an element of C_D decision, and an element of C_M merge. $start$ is called start node and end is called end node. All nodes $n \in N$ are on a path from start to end.

Due to page restriction, we skip the definitions regarding the execution history, the state of a process instance, and the reachability graph (see also [26]). These definitions will be provided in an extended version of this paper.

III. Modeling Context Constraints in UML

The Unified Modeling Language (UML) [18] is the de facto standard for the specification of information systems. Modeling support for context constraints via a standard notation can help to bridge the communication gap between software engineers, security experts, experts of the application domain, and other stakeholders (see, e.g., [13]). Our modeling approach for context-aware access control concepts acts as an enabler to document and communicate how access control in general and context constraints in particular affect a business process.

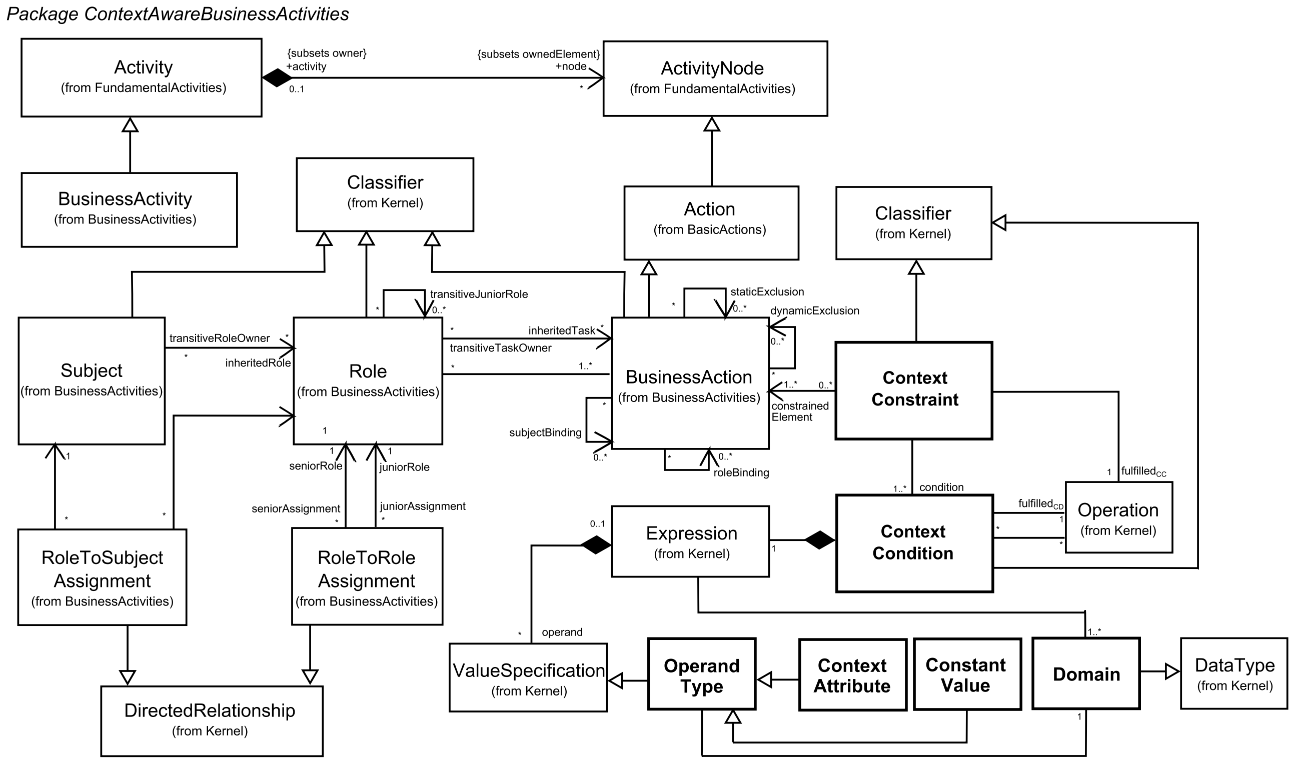

Figure 2. UML2 metamodel extension *ContextAwareBusinessActivities*

UML2 Activity models offer a process modeling language that allows to model the control and object flows between different actions. The main element of an Activity diagram is an Activity. Its behavior is defined by a decomposition into different Actions. An UML2 Activity thus models a process while the Actions included in the Activity are used to model tasks (for details on UML2 Activity models, see [18]).

In addition, we use the Object Constraint Language (OCL) [16] to formally define the semantics of the newly introduced UML elements and to ensure the consistency of the extended UML models. Corresponding software tools can enforce OCL invariants on the modeling-level as well as in runtime models. Thereby, we can ensure the consistency of our extended UML models with the definitions provided in Section II. However, note that our general approach does not depend on the UML and may also be applied to extend other process modeling languages.

The UML standard provides two options to adapt its metamodel to a specific area of application [18]: a) defining a UML profile specification using stereotypes, tag definitions, and constraints, which do not change the UML metamodel but extend existing UML meta-classes for special domains; b) extending the UML metamodel, which allows for the definition of new elements with customized semantics. However, UML profiles are not a first-class extension mechanism (see [18, page 660]). Moreover, the newly defined modeling elements for context constraints require new semantics which are not available in the UML metamodel. Thus, we introduce the UML metamodel extension *ContextAwareBusinessActivities* for modeling process-related context constraints (see Figure 2). Our UML extension extends the *BusinessActivities* package [26], which provides UML modeling support for process-related RBAC models.

A *BusinessActivity* [26] is a special UML Activity (see Figure 2). A *BusinessAction* corresponds to a task and comprises all permissions to perform the task. *Roles* and *Subjects* are linked to BusinessActions directly or transitively. Furthermore, mutual exclusion and binding constraints can be defined on BusinessActions (see [26] for further details).

For integrating context constraints into process-related RBAC models according to the definitions provided in Section II, we introduce the following new meta classes: The *ContextConstraint* meta class is a special UML2 Classifier (from the Kernel package, see [18] and Figure 2). Each ContextConstraint is linked to one or more BusinessActions and one or more ContextConditions indicating that the constrained BusinessAction can only be performed if all conditions specified in the ContextConstraint are fulfilled.

In UML, each Classifier may include an arbitrary number of Operations (see [18]). A ContextConstraint defines one mandatory Operation called *fulfilled*$_{CC}$. For each ContextConstraint, the corresponding fulfilled$_{CC}$ Operation checks if all linked ContextConditions are fulfilled and returns either true or false (see Constraint 4 listed at the end of this Section).

A *ContextCondition* is also defined as special type of Classifier. Moreover, each ContextCondition contains one UML Expression. According to [18], each Expression can define a set of operands of the type ValueSpecification. An Expression thereby represents an operator which is applied to those operands, e.g., $\geq (size, 5)$. A ContextCondition can include operands of the types *ContextAttribute* (e.g., age, size, date) or *ConstantValue* (e.g., constant integer numbers or Strings) (see Constraint 1). At least one of the operands included in a ContextCondition must be a ContextAttribute (see Constraint 2).

In addition, each ContextCondition includes one mandatory Operation called *fulfilled*$_{CD}$ and an arbitrary number of other Operations. For each ContextCondition, the corresponding fulfilled$_{CD}$ Operation checks if the context condition is fulfilled and returns either true or false (see Constraint 5).

A *Domain* is a subtype of the UML2 DataType meta class (see [18]). Operands are linked to a certain domain. Similarly, the operator defined in the Expression of a ContextCondition is linked to one or more domains (see Figure 2). Within the same ContextCondition, the domain of the operands needs to correspond to the domain of the operator (see Constraint 3). Otherwise, a ContextCondition can not be evaluated.

Figure 3 shows presentation options to visualize the relations between BusinessActions and ContextConstraints as well as between ContextConstraints and ContextConditions. Note that these relations are formally defined through our UML metamodel extension and therefore exist independent of their actual graphical representation. An example process modeled with our UML extension will be provided in an extended version of this paper.

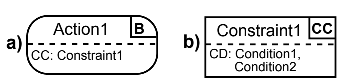

Figure 3. Visualizing (a) context-constrained BusinessActions and (b) ContextConstraints

Constraint 1: The operands specified in a ContextCondition are either ContextAttributes or ConstantValues:

```
context ContextCondition inv:
self.expression.operand.oclAsType(OperandType)->forAll(o |
   o.oclIsKindOf(ContextAttribute) or
   o.oclIsKindOf(ConstantValue))
```

Constraint 2: Each ContextCondition contains at least one ContextAttribute:

```
context ContextCondition inv:
self.expression.operand.oclAsType(OperandType)->exists(o |
   o.oclIsKindOf(ContextAttribute))
```

Constraint 3: Within the same ContextCondition, all operands must have the same domain. Moreover, the operands' domain has to correspond to one of the operator's domains:

```
context ContextCondition inv:
self.expression.operand->forAll(od1, od2 |
   od1.oclAsType(OperandType).domain.name =
   od2.oclAsType(OperandType).domain.name and
   self.expression.domain->exists(d |
      d.name = od1.oclAsType(OperandType).domain.name))
```

Moreover, the following two constraints must be satisfied which cannot be expressed in OCL (see [18]):

Constraint 4: The fulfilled$_{CC}$ Operations must evaluate to true to fulfill the corresponding ContextConstraint.

Constraint 5: The fulfilled$_{CD}$ Operations must evaluate to true to fulfill the corresponding ContextCondition.

IV. Related Work

Several approaches formally *integrate different types of context constraints into RBAC*. The RBAC model was extended to consider temporal aspects in access control decision (see, e.g. [3]) or the users' location (see, e.g. [4], [5]). In [9], the integration of contextual information with team-based access control (TMAC) is discussed. Hereby, users obtain permissions according to their membership in roles and teams. In [10], a context-aware RBAC model is presented, where context constraints are defined on different parts of the model. Moreover, role revocation is supported, in case values of the user attributes no longer satisfy the constraints.

The notion of *context constraints in workflow environments* has been studied, e.g., in the workflow authorization model (WAM) [1]. In this approach, subjects only can access objects during the execution of a task. Another approach for contextual security policies in organizational environments is presented in [7], where context is viewed as an extra condition that must be satisfied to activate a security rule. However, only certain types of contextual attributes, such as time or location are considered.

Other approaches for modeling context exist for various domains. There are some *UML-based modeling approaches* for considering context constraints in mobile or distributed systems. In contrast to our modeling approach, none of these works considers the business process/workflow perspective. In [24], a UML profile for modeling context for mobile distributed systems via special UML class diagrams was presented. The ContextUML language [23] is another UML profile for modeling context-aware Web-services.

To the best of our knowledge, our paper presents the first attempt to address RBAC context constraints from a business process modeling perspective. While many sophisticated approaches exist that formally integrate process-related context constraints into access control models, corresponding process modeling support is largely missing. Our approach complements existing context modeling approaches by providing modeling support for business processes and corresponding contextual authorization constraints in a consolidated modeling language.

V. Conclusion

This paper was motivated by the need for considering context-aware access control mechanisms in business processes. This is especially important due to the rising importance of ubiquitous computing technologies in business environments. We defined a formal metamodel to integrate context constraints into process-related RBAC models. Based on these definitions, we extended the UML to allow for the model-based specification of context-aware RBAC models for business processes in UML Activity diagrams. Moreover, we apply the Object Constraint Language (OCL) to define the semantics of the newly introduced UML meta classes. Our extension can be integrated with other UML-based approaches or tools.

References

[1] N. R. Adam, V. Atluri, and W.-K. Huang. Modeling and Analysis of Workflows Using Petri Nets. *Journal of Intelligent Information Systems*, 10(2), 1998.

[2] J. Becker, M. Rosemann, and C. v. Uthmann. Guidelines of Business Process Modeling. In *Business Process Management, Models, Techniques, and Empirical Studies*, London, UK, 2000. Springer.

[3] E. Bertino, P. A. Bonatti, and E. Ferrari. TRBAC: A Temporal Role-based Access Control Model. *ACM Transactions on Information and System Security (TISSEC)*, 4(3), 2001.

[4] E. Bertino, B. Catania, M. L. Damiani, and P. Perlasca. GEO-RBAC: A Spatially Aware RBAC. In *Proc. of the tenth ACM Symposium on Access control models and technologies (SACMAT)*, 2005.

[5] A. Corradi, R. Montanari, and D. Tibaldi. Context-Based Access Control Management in Ubiquitous Environments. In *Proc. of the Network Computing and Applications (NCA), Third IEEE International Symposium*, 2004.

[6] J. Cortadella, M. Kishinevsky, L. Lavagno, and A. Yakovlev. Deriving Petri Nets from Finite Transition Systems. *IEEE Transactions on Computers*, 47(8), 1998.

[7] F. Cuppens and N. Cuppens-Boulahia. Modeling Contextual Security Policies. *International Journal of Information Security*, 7(4), July 2008.

[8] D. F. Ferraiolo, D. R. Kuhn, and R. Chandramouli. *Role-Based Access Control*. Artech House, second edition, 2007.

[9] C. K. Georgiadis, I. Mavridis, G. Pangalos, and R. K. Thomas. Flexible Team-Based Access Control Using Contexts. In *Proc. of the sixth ACM symposium on Access control models and technologies (SACMAT)*, May 2001.

[10] D. Kulkarni and A. Tripathi. Context-Aware Role-Based Access Control in Pervasive Computing Systems. In *Proc. of the 13th ACM symposium on Access control models and technologies (SACMAT)*, 2008.

[11] N. Leavitt. Mobile Security: Finally a Serious Problem? *Computer*, 44(6), 2011.

[12] C. Miller. Mobile Attacks and Defence. *IEEE Security and Privacy*, 9(4), 2011.

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

[14] T. Murata. Petri Nets: Properties, Analysis and Applications. In *Proceedings of the IEEE*, 1989.

[15] S. Oh and S. Park. Task-Role-Based Access Control Model. *Information Systems*, 28(6), 2003.

[16] OMG. Object Constraint Language Specification. available at: http://www.omg.org/spec/OCL/, February 2010. Version 2.2, formal/2010-02-01, Object Management Group.

[17] OMG. Business Process Modeling And Notation (BPMN). available at: http://www.omg.org/spec/BPMN/, January 2011. Version 2.0, formal/2011-01-03, Object Management Group.

[18] OMG. Unified Modeling Language (OMG UML): Superstructure. available at: http://www.omg.org/spec/UML/, August 2011. Version 2.4.1, formal/2011-08-05, Object Management Group.

[19] M. Rosemann, J. C. Recker, and C. Flender. Contextualisation of Business Processes. *International Journal of Business Process Integration and Management*, 3(1), 2008.

[20] G. Roussos. Ubiquitous Computing for Electronic Business. In G. Roussos, editor, *Ubiquitous and Pervasive Commerce*, Computer Communications and Networks. Springer London, London, UK, 2006.

[21] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman. Role-Based Access Control Models. *IEEE Computer*, 29(2), 1996.

[22] R. S. Sandhu and P. Samarati. Access Control: Principles and Practice. *IEEE Communications Magazine*, 32(9), September 1994.

[23] Q. Z. Sheng and B. Benatallah. ContextUML: A UML-Based Modeling Language for Model-Driven Development of Context-Aware Web Services Development. In *Proc. of the International Conference on Mobile Business*, 2005.

[24] C. Simons. CMP: A UML Context Modeling Profile for Mobile Distributed Systems. In *Proc. of the 40th Annual Hawaii International Conference on System Sciences (HICSS)*, 2007.

[25] M. Strembeck. Scenario-Driven Role Engineering. *IEEE Security & Privacy*, 8(1), 2010.

[26] M. Strembeck and J. Mendling. Modeling Process-related RBAC Models with Extended UML Activity Models. *Information and Software Technology*, 53(5), 2011.

[27] M. Strembeck and G. Neumann. An Integrated Approach to Engineer and Enforce Context Constraints in RBAC Environments. *ACM Transactions on Information and System Security (TISSEC)*, 7(3), 2004.