A domain-specific language for the specification of UCON policies

Antonia Reina Quintero, Salvador Martínez, Ángel Jesús Varela-Vaca, María Teresa Gómez López, Jordi Cabot

▶ **To cite this version:**

Antonia Reina Quintero, Salvador Martínez, Ángel Jesús Varela-Vaca, María Teresa Gómez López, Jordi Cabot. A domain-specific language for the specification of UCON policies. Journal of information security and applications, 2022, 64, pp.103006. ⟨10.1016/j.jisa.2021.103006⟩. ⟨hal-03524313⟩

HAL Id: hal-03524313

https://hal.science/hal-03524313v1

Submitted on 20 May 2025

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons CC BY-NC-ND 4.0 - Attribution - Non-commercial use - No Derivative Works - International License

Journal of Information Security and Applications 64 (2022) 103006

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

A domain-specific language for the specification of UCON policies

Antonia M. Reina Quintero [a,*], Salvador Martínez Pérez [b], Ángel Jesús Varela-Vaca [a], María Teresa Gómez López [a], Jordi Cabot [c]

[a] Dpto. Lenguajes y Sistemas Informáticos, Universidad de Sevilla, Spain[1]
[b] Lab-STICC, IMT Atlantique, Brest, France
[c] ICREA-UOC, Barcelona, Catalunya, Spain

ARTICLE INFO

Keywords:
Cybersecurity
Access control
Model-driven engineering
UCON
DSL

ABSTRACT

Security policies constrain the behavior of all users of an information system. In any non-trivial system, these security policies go beyond simple access control rules and must cover more complex and dynamic scenarios while providing, at the same time, a fine-grained level decision-making ability. The Usage Control model (UCON) was created for this purpose but so far integration of UCON in mainstream software engineering processes has been very limited, hampering its usefulness and popularity among the software and information systems communities. In this sense, this paper proposes a Domain-Specific Language to facilitate the modeling of UCON policies and their integration in (model-based) development processes. Together with the language, an exploratory approach for policy evaluation and enforcement of the modeled policies via model transformations has been introduced. These contributions have been defined on top of the Eclipse Modeling Framework, the de-facto standard MDE (Model-Driven Engineering) framework making them freely available and ready-to-use for any software designer interested in using UCON for the definition of security policies in their new development projects.

1. Introduction

The description and analysis of security policies are essential for an organization [1]. Moreover, the lack of incorporation of security policies into the daily processes and their information systems can bring about the failure to achieve the business objectives and probably security flaws. For this reason, the SANS Institute defines a security policy as a set of security requirements or rules (i.e., access control restrictions) that must be met to achieve the business goals. To ensure the proper operation of companies, it is necessary to follow both the defined processes and the security policies that restrict potential security threatening situations that put at risk the data and the stakeholders [2].

For this reason, any information system and, especially, any Process-Aware Information Systems (PAIS) [3] must include the security policies that provide the mechanisms to achieve properly the business goals. PAIS manage the operational processes that involve software applications, the data storage, and the people that interact with the processes; thereby, these systems cannot be misaligned with the security policies. Leitner et al. [4] states that security control remains an open challenge in PAIS. Current solutions based on traditional access control models [5] fail to contemplate and evaluate the flexible and complex security policies that modern business information systems demand. These traditional access control models include Mandatory Access Control (MAC), that restrict through a set of rules the access of an object or the execution of an operation; Discretionary Access Control (DAC), where the capacity of access relies on the owner of the object; and, Role-based Access Control (RBAC) that is oriented towards roles and privileges. However, these three access control models fail to integrate sophisticated decision-making, because they focus on the request and grant the access only at request time, losing the perspective of the ongoing controls that can change over time. Moreover, these traditional access controls fail to incorporate a temporal perspective and restrictions regarding the number of uses of the resources.

As a response to these limitations, the Usage Control (UCON) model [6] was proposed. UCON extends previous access controls models with the addition of the primitives that enable more complex security policies. An example of a security policy that UCON can handle, but not the previous access control models, is: *the operational staff is not allowed to review more than ten loan requests in less than three days*. This rule restricts the rights according to a specific period of time

* Corresponding author.
E-mail addresses: reinaqu@us.es (A.M. Reina Quintero), salvador.martinez@imt-atlantique.fr (S. Martínez Pérez), ajvarela@us.es (Á.J. Varela-Vaca), maytegomez@us.es (M.T. Gómez López), jordi.cabot@icrea.cat (J. Cabot).
[1] IDEA Research group: http://www.idea.us.es/.

https://doi.org/10.1016/j.jisa.2021.103006

Available online 2 November 2021
2214-2126/© 2021 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

and a number of accesses, two aspects that only can be managed with UCON. Therefore, integrating UCON in software and system design tools, including business process modeling tools such as BonitaSoft [7], jBPM [8], and Activiti [9] that currently do not support the modeling of complex security policies, as the ones enabled by UCON, would be a major step forward in the definition and implementation of secure information systems.

One first attempt to deal with UCON policies was published in [10] where an extension of artifact-centric business process models to deal with partial UCON policies was proposed. However, the solution was not generic and only useful for that specific type of models and, moreover, UCON expressiveness was not completely supported, and the metamodel was not formalized. In this paper, a domain-specific language for specifying UCON policies is proposed. This language is generic and fully compliant with the UCON specification, enabling definition and integration of UCON policies in any type of model. As such, as far as we know, ours is the first proposal that presents a UCON metamodel that covers the whole access control model and that is flexible enough to be added to any modeling toolchain. In other words, our approach aims at providing MDE tools to implement the original UCON.

Additionally, our approach covers not only the specification of UCON access control rules but also their enforcement by leveraging the same modeling infrastructure we used to create the language in the first place. Model-Driven Engineering (MDE) [11] promotes the rigorous use of software and systems models in all software engineering activities and also facilitates the development process, the code generation, and the integration in different environments [12,13]. Policy enforcement is explored by means of an application-independent structure based on a Policy Enforcement Point (PEP)-Policy Decision Point (PDP) architecture. This architecture is implemented by using model transformations, concretely, the ATL model transformation language and framework [14] are proposed to evaluate rules at run-time. Furthermore, to illustrate the usability of the proposed Domain-Specific Language, a complex running example that includes UCON policy rules related to both the processes and the data model has been employed.

The rest of the paper is structured as follows: Section 2 introduces the necessary concepts to understand the UCON model, the model-driven engineering, and the ATL model transformation language; Section 3 introduces the running example that illustrates the need for defining policies with a UCON-based language; Section 4 introduces the model-driven approach and its application to the motivating example; Section 5 goes over the most relevant previous papers related to access control; and, finally, conclusions are drawn and further works are pointed out.

2. Background

This section is structured as follows: Firstly, Section 2.1 introduces the main concepts behind the Usage Control Access Model (UCON). Afterwards, Section 2.2 gives a general overview of model-driven engineering and the notion of domain-specific language, and then focuses on model transformations as the tool in the model-driven field we will employ later on to provide a robust, flexible, and re-usable run-time evaluation engine for the UCON policies. But before diving into the following subsections, it is worth to remark that the word *model* has slightly different semantics in the field of access control and the field of model-driven engineering. While in access control, a model is a formal definition of a set of access control rules that is technology-independent of how these rules are later enforced, a model in model-driven engineering is an abstract representation of a system perspective that conforms to a specific language/metamodel [15].

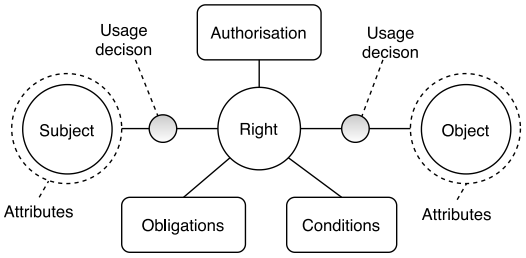


Fig. 1. UCON components [16].

2.1. The Usage Control access model (UCON)

Usage Control ($UCON_{ABC}$) [16] is a model that goes beyond traditional access control models by inheriting all their merits and providing a finer-grained decision-making ability. The UCON model comprises three core components (i.e., subjects, objects and rights); and three additional components (i.e., authorization rules, obligations and conditions) represented in Fig. 1. Note that the core components are similar to the ones defined in traditional models, such as the RBAC model, while the additional components provide the extra decision-making ability. The components are described as follows:

Core components:

- *Subjects* are the entities that hold rights on objects.
- *Objects* are the entities which subjects can access with certain rights.
- *Rights* are the actions that the subjects can perform on objects.

Additional components or predicates:

- *Authorizations (A)* are predicates that have to be evaluated to decide whether the subject is allowed to execute the rights on the objects. Authorizations can be pre-authorizations (preA), that are performed before the rights are exercised, or ongoing-authorizations (onA), that are evaluated while the rights are exercised.
- *Obligations (B)* are predicates that verify mandatory requirements that the subjects have to comply with before (pre-obligation) or during the exercise of the rights (ongoing-obligations).
- *Conditions (C)* are predicates that evaluate environmental or system status conditions as additional obligations for the access.

Subjects and *Objects* have attributes that represent properties or capabilities that can be used for deciding whether the subject can access the object. Attributes can be mutable or immutable. $UCON_{ABC}$ model introduces the concept of mutability which indicates whether certain attributes can be modified during the usage decision process. Note that, unlike authorizations and obligations, condition variables must be immutable. Furthermore, it also should be borne in mind that the mutability of attributes and the continuity (ongoing authorization) of an access decision are the two novel aspects that $UCON_{ABC}$ introduces in comparison with traditional control access approaches [16]. The incorporation of these new aspects implies that if an access attribute is changed while the access is in progress making the security policy not satisfied, then the granted access is revoked and the usage is terminated. Fig. 1 depicts how the usage decision functions depends on subject attributes, object attributes, authorizations, obligations and conditions.

The UCON policy identifies 24 usage control models (see Table 1) that are the result of combining pre and ongoing authorizations (preA, onA), obligations (preB, onB) and conditions (preC, onC) with attribute updates that can be performed before, during or after the access. For example, a UCON rule of $preA_3$ type represents a usage control scenario where the access decision is evaluated only once before beginning the

Table 1
The $UCON_{ABC}$ models obtained from [16].

	0 (inmutable)	1 (pre-update)	2 (ongoing-update)	3 (post-update)
preA	✓	✓	✗	✓
onA	✓	✓	✓	✓
preB	✓	✓	✗	✓
onB	✓	✓	✓	✓
preC	✓	✗	✗	✗
onC	✓	✗	✗	✗

usage. Note that traditional access control approaches can be expressed with UCON $preA_0$ and $preA_1$ (pre-authorizations without attribute update and with attribute pre-update, respectively).

2.2. Model-driven engineering

Model-driven engineering (MDE) [11] promotes the rigorous use of software and systems models in all software engineering activities.

A key element in any MDE approach is the choice of the right modeling languages for the task at hand. In this respect, Domain-Specific Languages (DSLs) are languages that are designed specifically for a certain domain, context, or company to ease the task of people that need to describe things in that domain.

The core element of any DSL is its metamodel that defines what modeling primitives are part of the language and how they can be combined (akin to the role of grammars in programming languages). The de-facto standard for defining metamodels in practice is EMF (Eclipse Modeling Framework) [17]. Metamodels can be complemented with well-formedness rules defined in OCL (Object Constraint Language) [18] to further restrict the set of valid models allowed in the language.

Once a model is defined it can be transformed into another type of model (a model-to-model transformation or M2M) or into a textual representation (model-to-text or M2T, also referred to as code-generation when the text artifact target of the transformation is a coding artifact).

Model transformations are fundamental in model-driven engineering as they provide the means for automating the manipulation of models. A model to model transformation (M2M) transforms a model M_a conforming to a metamodel MM_a into a model M_b conforming to metamodel MM_b where MM_a and MM_b can be the same or different metamodels. While this kind of transformation can be implemented by using general purpose languages, model-transformation languages and frameworks exist in order to ease its specification by providing facilities to efficiently query and manipulate models. In this respect, there exist many different model transformation languages and paradigms with different features, e.g., ATL (Atlas Transformation Language) [14], ETL (Epsilon Transformation Language) [19] or QVT (Query-Views-Transformation) [20]. Due to its associated tooling and maturity, ATL is one of the most popular transformation languages among academic and industrial practitioners. Thus, ATL is the model transformation language of choice for this work but our proposal could be similarly implemented on top of the other transformation proposals listed before.

In order to clarify how model transformations are specified, a simple ATL transformation module is shown in Listing 1. This example has been extracted from the ATL transformation Zoo.[2] It translates *Class* definitions to relational *Table* definitions. It uses the Class and Table metamodels depicted in Figs. 2a and 2b respectively. An ATL transformation (module) is composed of a set of transformation rules and helpers. Each rule describes how (part of) the target model should be generated from (part of) the source model. A helper can be seen as an auxiliary function that enables the possibility of factorizing ATL code used in different points of the transformation. Note that ATL uses OCL as its navigation and query language.

[2] https://www.eclipse.org/atl/atlTransformations/.

In Listing 1, lines 1–2 declare the transformation name along with the input and output metamodels. This transformation contains a rule called *Class2Table* (lines 7 to 22) that matches every *Class* in the *Class* model to produce a *Table* in the output model. It does so by using a helper function that returns the integer type defined as a datatype in the class model (lines 4 and 5). Rules are mainly composed of an input pattern (line 9) and an output pattern (line 10 to 21). The input pattern filters the subset of source model elements that are concerned by the rule. The output pattern details how the target model elements are created from the input ones (here two output pattern elements are created, a table and a key column). Each output pattern element can have several bindings (lines 12, 15, 16, 19 and 20) that can be used to initialize the values of the elements in the target model.

Listing 1: ATL Transformation Module Example

```
1  module Class2Relational;
2  create OUT : Relational from IN : Class;
3
4  helper def: objectIdType : Relational!Type =
5    Class!DataType.allInstances()->select(e|e.name = 'Integer'
         ↪)->first();
6
7  rule Class2Table {
8    from
9      c : Class!Class
10   to
11     out : Relational!Table (
12       name <- c.name,
13       -- Columns are generated from Attributes in another
             ↪rule
14       -- not explicitly called here !
15       col <- Sequence {key}->union(c.attr->select(e|not e.
             ↪multiValued)),
16       key <- Set {key}
17     ),
18     key : Relational!Column (
19       name <- 'objectId',
20       type <- thisModule.objectIdType
21     )
22 }
```

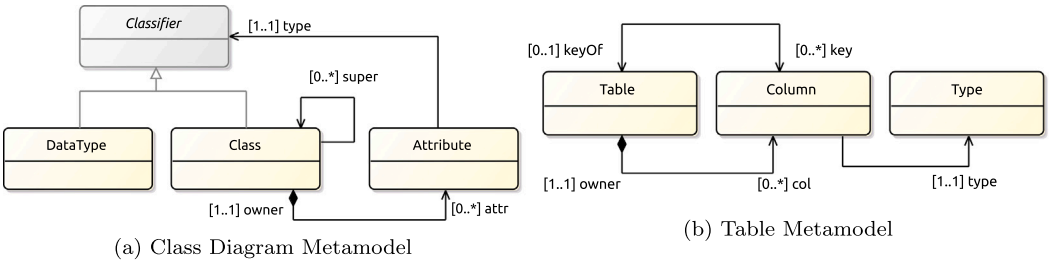
In our approach, model transformations feed the model transformation engine that is used as off-the-shelf access control evaluation engine [21]. Note that creating an access-control evaluation engine for a given access-control model from scratch is a complex and time-consuming task. For further details of how model transformations supports our approach see Section 4.2.

3. Running example

To introduce the scenarios where UCON is necessary and how our proposal can facilitate the description of the policies, we introduce a running example related to a customer's loan request process [22]. The example is presented in two different, but complementary, perspectives: the activity-centric and the data perspectives. The activity-centric perspective of the process includes the activities execution flow, the actors involved, and the interactions between them. The data perspective involves the data entities and how they are related through a data model.

Fig. 3 shows the activity-centric perspective by means of a business process model [23] in which three different actors are involved: the *Customer* pool shows how a customer requests a loan to a credit supplier and receives a notification (i.e., approved or non-approved); the *Credit Supplier* pool describes how the administration staff manages the loan request, obtains the acceptance report and notifies customers the decisions; and, the *Operation Staff* pool, that shows how they deal with the evaluation of the loan requests and generate reports.

This process model is insufficient to fully depict the data exchanged in the process activities. A business process model must be supported by a proper data model [24]. Fig. 4 depicts the UML class diagram for the running example, that involves some data entities and relations not included in the process model, but necessary for the complete



(a) Class Diagram Metamodel

(b) Table Metamodel

Fig. 2. Input and output metamodels.

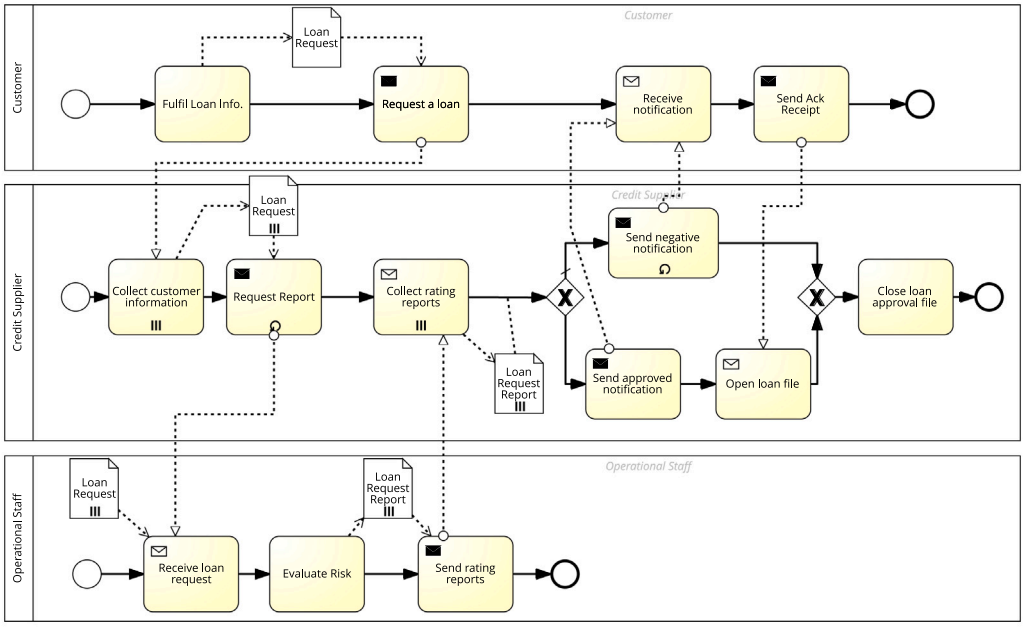


Fig. 3. Business process model for the running example.

description of the problem domain. Some entities (*classes* in UML terminology) of the data model are related to pools in the process (e.g., Operational Staff), others are part of the data flow of the process model (e.g., Loan Request Report), and others do not appear in the business process model (e.g., Account). The different entities and their relations (*associations* in UML terminology) can be constrained by a security policy. For instance, any *Customer* needs an account and validated credentials for the *Credit Supplier* before requesting a loan.

Nevertheless, both process and data models alone become also insufficient to reflect certain access and usage control policies that span a single system perspective. For instance, the *Customer* is limited to perform a certain number of loan requests during a period of time. The enforcement of this type of access control rule needs the analysis of both the data and process aspects.

To illustrate the need for this complex security polices, we define a potential ruleset for the running example. The ruleset is composed of a set of rules exemplifying the 16 types of predicates defined by the UCON model (see Table 1). The set of rights used in the ruleset are described as follows:

- *Request a loan*: the customer sends a loan request to the credit supplier.
- *Review a loan request*: the operational staff reviews the loan request received for the credit supplier.
- *Create a loan request report*: the operational staff creates a report for each loan request that he/she will review.
- *Update account balance*: the customer's balance is updated according to the loan terms.
- *Send notification*: the credit supplier sends a notification to the customer for each loan request.
- *Create a loan file*: the credit supplier creates a loan file for each accepted loan request.

The set of rules in the ruleset are described next. They are grouped by the type of predicate (authorization, obligation, condition). Note also that subjects (S), objects (O), and rights (R) involved in the rule have been included in order to better illustrate the rule context.

- **Authorization rules:**
  - $preA_0$ (S: Operational Staff, O: Loan Request, R: Review a loan request): the two persons (operational staff) who review the loan request cannot be the same (Separation of Duties).
  - $preA_1$ (S: Customer, O: Loan Request, R: Request a loan): the customer is allowed to request a loan if the remuneration after costs is greater than half the remuneration.
  - $preA_3$ (S: Customer, O: Loan Request, R: Request a loan): the customer is allowed to request no more than three loans.
  - onA_0 (S: Customer, O: Loan Request, R: Request a loan): the customer is allowed to do a loan request while its credentials are not revoked.

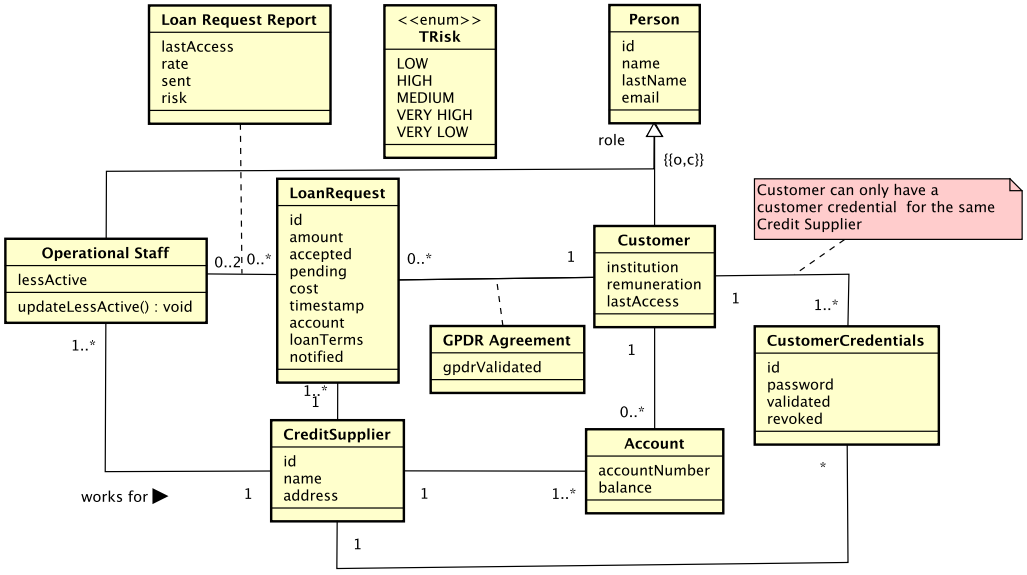


Fig. 4. Data model for the running example.

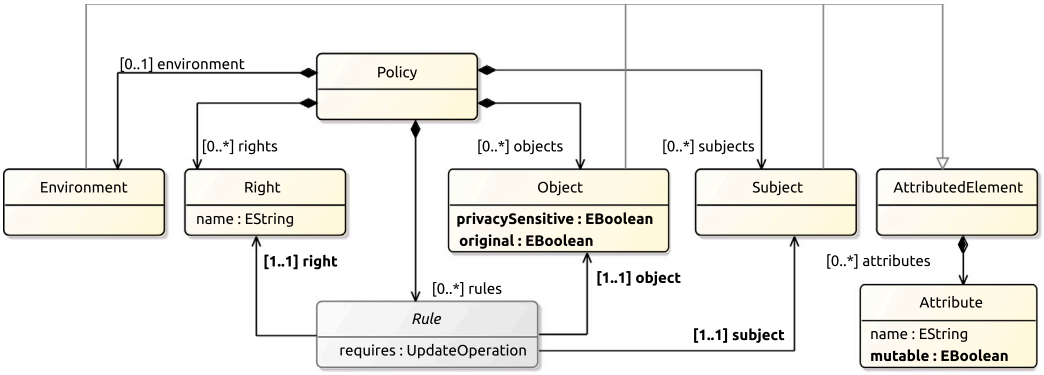


Fig. 5. UCON metamodel top elements.

- $onA_{1,3}$ (S: Operational Staff, O: Loan Request Report, R: Review loan request): the operational staff is not allowed to review more than ten loan requests at the same time.
- $onA_{1,2,3}$ (S: Operational Staff, O: Loan Request Report, R: Create a loan request report): the operational staff is allowed to create a new loan report if he/she has no more than five pending (i.e., non-sent) and inactive (i.e., non-recently updated) reports.

- **Obligation rules:**
  - $preB_0$ (S: Customer, O: Loan Request, R: Request a loan): the customer has to agree the GPDR statement before requesting a loan.
  - $preB_1$ (S: Customer, O: Loan Request, R: Request a loan): the customer has to agree the loan terms of all the customer's loan request before requesting a loan.
  - $preB_3$ (S: Credit Supplier, O: Account, R: Update account balance): after a loan request is approved, the customer's balance has to be updated according to the amount and costs of the loan.
  - onB_0 (S: Operational Staff, O: Loan Request, R: Review a loan request): the operational staff has to review a loan request while the control time system is active.
  - $onB_{1,3}$ (S: Task Collecting Rating Reports, O: Task Send Approved Notification, R: Send notification): the credit supplier has to ask for the pending reports to collect the rating of the loans before sending a acceptance notification.
  - onB_2 (S: Task Send Notification Receipt, O: Task Open Loan File, R: Create a loan file): to allow a credit supplier open a loan file, the customer has to send an acknowledgment of receipt before two days.

- **Condition rules:**
  - $preC_0$ (S: Customer, O: Loan Request, R: Request a loan): the customer is not allowed to request a loan from a mobile phone device.
  - onC_0 (S: Operational Staff, O: Loan Request, R: Review a loan request): the operational staff is allowed to review a loan request during office hours (9 am–17 pm).

Note that there are rules, such as $preA_0$ (SoD), that could be modeled with traditional access control models, but others cannot (see, for instance, $onB_{1,3}$). Regarding subjects and objects, there are rules in which the subject and the object refer to entities of the data model, such as *Customer* and *Loan Request* (in $preA_0$) and others that refer to elements of the process model, as the $onB_{1,3}$ rule. Both rules need the task context to define the predicate.

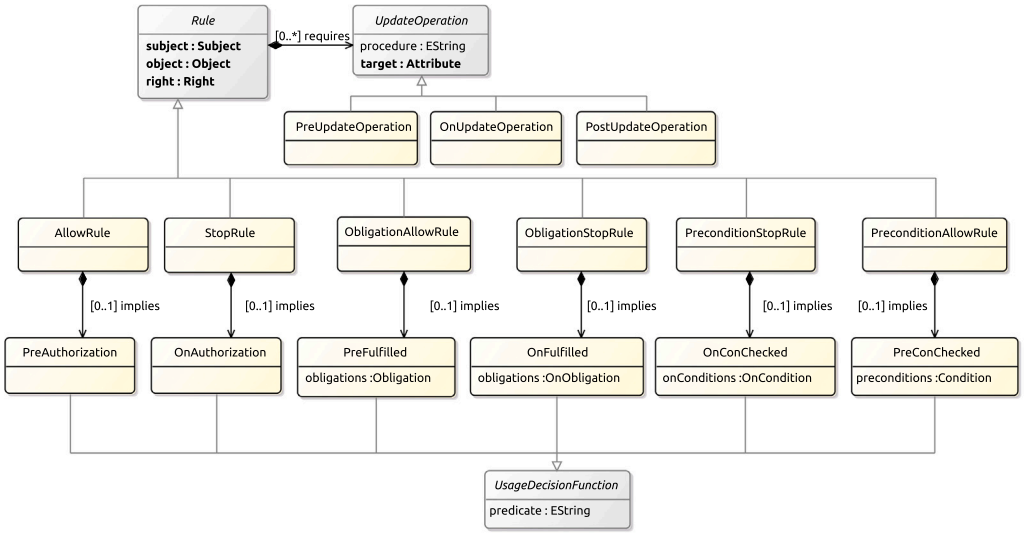


Fig. 6. UCON metamodel rules.

Regarding the update operations, the $preA_0$, onA_0, $preB_0$, onB_0, $preC_0$ and onC_0 predicates in the running example do not require any update on the subject's and on the object's attributes. For instance, the $preA_0$ predicate needs to consult the loan request attributes to check its status. Other predicates imply update operations, thus, in the following we specify the update operations needed for each predicate and the attributes involved:

- $preA_1$: the *remuneration* attribute of the customer must be *pre-updated* before granting the right.
- $onA_{1,3}$: while the operational staff is exercising the right, the number of loan requests reviewed must be *pre-updated* by adding a new one, and it must be *post-updated* by reducing the number of loan requests reviewed in case of revocation.
- $onA_{1,2,3}$: while the operational staff is exercising the right, the number of loan request reports created must be *pre-updated*, and it must be *post-updated* by reducing the number of loan request reports in the case of revocation. Additionally, we need to maintain indexed the less active loan request reports in each moment by *on-updating* the *lessActive* attribute within operational staff.
- $preB_1$: the customer must agree the loan terms, thereby, the *loanTerms* attribute of all the loan requests of a customer must be *pre-updated* and its value should be set to *true* before doing the loan requests.
- $onB_{1,3}$: the customer must fill in the form with the data of the loan to be requested, the number of loan requests sent must be *pre-updated* by adding a new one, and it must be *post-updated* by reducing the number of loan requests reviewed in case of revocation.
- $onB_{1,2,3}$ (Acceptance Notification and Negative Notification): the credit supplier must ask for the pending reports before sending notifications (acceptance or negative). The number of loan request reports received must be *pre-updated* by adding a new one, and it must be *post-updated* by reducing the number in the case of revocation. Additionally, we need to maintain updated the reports created in each moment by *on-updating* them in case of incoming new reports in the credit supplier.
- $preA_3$: the number of loan requests done by a customer must be *post-updated* before granting the right.
- $preB_3$: the account's *balance* attribute of the customer must be *post-updated* after checking that the loan request is accepted.

4. Definition, evaluation and enforcement of UCON policies

Integrating access-control in a given technical space requires, once a suitable access-control paradigm has been selected, the means to: (1) define *correct* access-control policies conforming to that selected paradigm; (2) evaluate access requests (e.g., decide whether the access is permitted or not) to protected resources with respect to the aforementioned policy; and, (3) control the access to resources so that it occurs only as the access-control policy prescribes.

Providing support to the aforementioned tasks is far from being trivial. In the following, we discuss how mature MDE tools and techniques alleviate this challenge and describe a full MDE approach for the definition, evaluation and enforcement of UCON access-control policies that enables full UCON support in modeling environments.

4.1. Policy definition

In order to allow the definition of UCON access control policies, we have created a Domain-Specific Language (DSL) that captures and organizes the UCON concepts as defined in [16]. In the following, we describe the abstract syntax of this DSL which we have implemented as an EMF metamodel[3] [17]. Note that as concrete syntax we will use off-the-shelf tree-based model editors and XMI serialization, and that execution semantics are provided by translating the access-control policies to executable model-transformations specifications as we will show later. Alternative concrete syntaxes (either textual or graphical) can be easily defined with other EMF-based tools like Xtext[4] or Sirius,[5] highlighting once again the side benefits we get once we enable UCON to the plethora of EMF modeling tools through the definition of our DSL.

The top-level elements of our UCON metamodel are depicted in Fig. 5. The root element is the *Policy* metaclass, that acts as a container of the five sets of elements needed to describe access-control in our context, namely: *Subjects*, *Objects*, *Rights*, *Rules* and *Environment*. *Subject*, *Object* and *Right* represent respectively, and as described in [16], the active entity which requests access, the resource being accessed, and the concrete action to be performed with the access. *Subjects* and

[3] The EMF metamodel is publicly available in https://gitlab.com/smartine/uconmde/.

[4] https://www.eclipse.org/Xtext/.

[5] https://www.eclipse.org/sirius/.

Objects may have *Attributes* which may be mutable or not (note that for simplicity in this version of the metamodel we consider all attributes to have basic types, such as Integer or String). The *Environment* metaclass is a place holder used to store environmental and context attributes used by conditions. Finally, the core concept of our metamodel is that of *Rule*.

Fig. 6 shows the elements of our metamodel devoted to rules. Access-control decisions are described in UCON as implications, meaning that granting (or revoking) a given access implies, depending on the concrete UCON model, the fulfillment of a number of predicates and the execution of update procedures. We organize these concepts by introducing the *Rule* metaclass, which is not explicitly defined in UCON. *Rules* refer to exact one *Subject*, *Object* and *Right*. *Rules* may require a certain number of *UpdateOperations* to be performed on attributes before (*PreUpdateOperation*), during (*OnUpdateOperation*) of after (*PostUpdateOperation*) the access is granted. Finally, concrete *Rules* (the *Rule* metaclass is abstract) imply the fulfillment of *UsageDecisionFunctions*.

We define 6 types of *Rules*: *AllowRule*, *StopRule*, *ObligationAllowRule*, *ObligationStopRule*, *PreConditionAllowRule*, and *PreConditionStopRule*. *AllowRule* and *StopRule* imply the fulfillment of a pre or continuous authorization respectively (*PreAuthorizations* and *OnAuthorizations* in the metamodel). *ObligationAllowRule* and *ObligationStopRule* imply the fulfillment of a list of pre or continuous obligations respectively (represented by the list of *Obligations* in the *PreFulfilled* and *OnFulfilled UsageDecisionFunctions* in the metamodel). Finally, *PreConditionAllowRule* and *PreConditionStopRule* imply the fulfillment of a list of pre or continuous environmental preconditions (represented as the lists of *OnConditions* and *Conditions* in the *PreConChecked* and *OnConChecked UsageDecisionFunctions* in the metamodel).

These rules combined with the different (valid) *UpdateOperations* cover the 24 basic UCON models, as described in Table 1, and allow us to describe all the rules in the running example presented in Section 3. As an example, an *AllowRule* combined with a *PostUpdateOperation* permits us to define the $preA_3$ rule, whereas the combination of an *ObligationAllowRule* with a *PreUpdateOperation* would permit us to define rules, such as $preB_1$.

Note that *UpdateOperations* and *UsageDecisionFunctions* contain a procedure and a predicate body respectively. The procedure is used in order to calculate the value needed for the update of a given attribute (e.g., in rule $preA_3$ the subject number of the loan requests is incremented) whereas the predicate is used by rules as a necessary condition for granting access (e.g., in rule $preA_3$ we verify that the number of loans is not bigger than 3). Both procedures and predicates may be arbitrarily complex expressions, therefore, the metamodel is needed to integrate a sufficiently expressive language to deal with them. In that sense, we chose to use OCL as the expression language for our metamodel. OCL [18] is a general-purpose (textual) formal language adopted as a standard by the Object Management Group[6] used to define several kinds of expressions that complement the information of models. OCL is a typed, declarative and side-effect free specification language.[7] Actually, we will use OCL with a double purpose: firstly, for the definition of procedures and predicates, and secondly, as a specification language to complement our metamodel and validate conforming models, as we show in the following subsection.

Listing 2 shows how the proposed approach is used to model a sample of the running example presented in Section 3. Concretely, it presents (part of) the model representing the authorization rule $preA_3$: (S: Customer, O: Loan Request, R: Request loan) stating that the customer is only allowed to request up to three loans. Lines 7 to 15 show the definition of two subjects, Customer and CreditSupplier, with their corresponding attributes such as accounts, loanRequests or name. A LoanRequest object is declared in lines 17 to 24, containing diverse attributes such as cost, credditSupplier and loanTerms. Line 26 shows the declaration of the RequestALoan right. Finally, in lines 28 to 39 we show the definition of the aforementioned loan request authorization rule. The rule subject, object and right, are referenced at the beginning of the rule in lines 30 to 32. Then a postUpdate operation on the number of loan requests made by the Customer is defined between lines 33 and 36. The authorization rule ends with the definition of the authorization predicate needed for the grating of the permission (lines 37 and 38).

Listing 2: UCON Policy Example in XMI

```
1  <ucon:Policy
2      xmi:version="2.0"
3      xmlns:xmi="http://www.omg.org/XMI"
4      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5      xmlns:ucon="http://www.example.org/ucon"
6      xsi:schemaLocation="http://www.example.org/ucon ucon.
           ecore">
7    <subjects id="Customer">
8      <attributes name="accounts"/>
9      <attributes name="customerCredentials"/>
10     <attributes name="loanRequests" mutable="true"/>
11   </subjects>
12   <subjects id="CreditSupplier">
13     <attributes name="id"/>
14     <attributes name="name"/>
15   </subjects>
16
17   <objects id="LoanRequest">
18     <attributes name="id"/>
19     <attributes name="cost"/>
20     <attributes name="creditSupplier"/>
21     <attributes name="customer"/>
22     <attributes name="account"/>
23     <attributes name="loanTerms" mutable="true"/>
24   </objects>
25
26   <rights id="R1-RequestALoan" name="RequestALoan"/>
27
28   <rules xsi:type="ucon:AllowRule"
29         id="preA3"
30         subject="Customer"
31         object="LoanRequest"
32         right="R1-RequestALoan">
33     <requires
34           xsi:type="ucon:PostUpdateOperation"
35           procedure="loanRequests->size()+1"
36           target="//@subjects.0/@attributes.8"/> //loanRequest
                 Attribute
37     <implies
38           predicate="subject.loanRequests->size() < 3"/>
39   </rules>
```

4.1.1. Well-formedness rules

Models created with our UCON metamodel will be all syntactically correct. However, not all will be semantically correct. This is so because the language used to create the metamodel is not expressive enough to specify all the details and/or because doing so would make the metamodel more complex and less usable. In order to solve this issue, we need to provide well-formedness (WFRs) rules to validate the model. We use OCL for this purpose as aforementioned. Note that OCL integrates seamlessly with (some) existing MDE tools and permits the automatic evaluation of these rules at design time in order to produce correct UCON policies. That is, once the WFRs for UCON are defined, state-of-the-art OCL tools can be used to automatically ensure designers only specify correct UCON models.

Listing 3 shows the set of the most important WFRs. The first invariant verifies that the update operations of a rule only updates attributes belonging to the Subject or Object of the rule. The second and third invariants verify that the policy only uses pre or post update operations for $preA$ and $preB$ UCON models. The fourth and fifth invariants verify

[6] https://www.omg.org/.

[7] Note that for simplicity we use String attributes as place holders for expressions. In future versions we intend to link directly to the OCL metamodel expression metaclass.

that update operations are not permitted for *preC* and *onC* UCON models. Invariant 6 verifies that an update operation only uses mutable attributes. Finally, Invariant 7 verifies that the environment attributes must be immutable.

Listing 3: Well-formedness Rules

```
// Invariant 1 – Context Rule
inv validUpdaters:
        subject.attributes->union(object.attributes)
                            ->includesAll(requires->collect(e
                                ↪ | e.target));
// Invariant 2, 3 – Context AllowRule, ObligationAllowRule
inv updateIsNotOn:
        not requires->oclIsTypeOf(OnUpdateOperation);
// Invariants 4, 5 – Contexts PreconditionAllowRule,
    ↪PreconditionStopRule
inv updateOperationsNotPermitted:
        requires->size()=0;
// Invariant 6 – Context UpdateOperation
inv isMutable:
        target.mutable;
// Invariant 7 – Context Environment
inv inmutableAttributes:
        attributes->forAll(attr | not attr.mutable);
```

Additionally to well-formedness rules verifying the correctness of UCON models, we can, in the same manner, define complex domain-specific and/or policy administration rules which are not tackled in the original definition of UCON. Or even WFRs that, for instance, adapt UCON when applied to a specific domain with very particular semantics. For more information on the use of MDE and OCL for the definition and evaluation of complex access-constraints see [25] and [26].

4.2. Policy evaluation

The metamodel we have described above allows us to define access-control rules that determine what subjects can do on the resources of a system. This policy must be then consulted each time an access request to a protected resource is performed in order to determine whether it is permitted or not. Thus, for the effective integration of access control in business processes, we must provide a mechanism for the automatic evaluation of access requests with respect to a given policy. However, creating an access-control evaluation engine for a given access-control model from scratch is a complex and time-consuming task. As a consequence, we have decided to use an existing model transformation engine as off-the-shelf access-control evaluation engine as proposed in [21]. In the rest of this section, we describe how to map UCON policies to model transformation rules that will then be the input of the transformation engine for access-control evaluation.

Similarly to model access-control policies, a model transformation function $Mt : \{SourceModel\} \rightarrow \{TargetModel\}$ taking as an input a source model and producing a target model can be seen as composed by rules of the form: $Mt_{ri} : \{Match \times Conditions\} \rightarrow \{Output \times BindingValues\}$ where a *Match* is a source model element, *Conditions* are a set of guards that must hold for the rule to fire, *Output* is a target model element and *BindingValues* are initialization values.

When the element to match and the element to produce for a rule are always of the same type (in our case, and as discussed next, of type *Request* and *Decision* respectively), this function can be simplified to be $Mt_{ri} : \{Conditions\} \rightarrow \{BindingValues\}$. If we make that *BindingValues* contain the values of an access decision (e.g., {accept, deny}), we have that the evaluation of access-control rules can be seen as a specific case of model transformation where input elements are always of the same type and the output model element contains a decision value.

As model transformation frameworks are mature tools with a large amount of research and development work committed to them, they are ideally placed to take over the task of evaluating access-control policies as a special case of their natural range of use. As mentioned in Section 2.2, we use ATL as our model transformation framework of choice due to its maturity and tooling. Additionally, ATL uses OCL as its navigation and query language, which facilitates the translation of our UCON policies to ATL.

Table 2
UCON policy to ATL module translation.

UCON policy	ATL module
UCON rule	ATL rule (with predicate called in guards)
Attributes	Helpers with context
Predicates	Helpers
PreUpdate procedures	Helpers called inside the helper representing the attribute
PostUpdate procedures	Helpers called on the output pattern
OnUpdate procedures	Delegated to enforcement environment

Practically, the translation of UCON policies to ATL transformations follows the translations rules described in Table 2 (note that we reuse the simple *Request* and *Decision* metamodels defined in [21] to use them as input and output models of our transformation. To the decision metamodel, we add a list of updated values for attributes). OnUpdate procedures and the corresponding revocation of permissions are delegated to the enforcement environment that will update attributes when required and then re-run the model transformation to obtain a new access decision. An example of translation from a UCON policy to an ATL model transformation specification is provided in Listing 4.

Listing 4 shows the ATL translation corresponding to the rule introduced in Listing 2. Lines 1 to 8 show three helpers used to rectify the loanRequest attribute, calculate the predicate and calculate the update procedure respectively. Lines 11 to 23 show the ATL rule corresponding to the authorization rule $preA_3$. It matches a subject usage request (lines 12 to 13) only when the predicate preAuth3 is true. Once the rule matches the request, it creates an access decision (lines 14 to 23) as a target model element. This access decision contains a value in the set {true, false} (lines 15 and 16) and a list of new values to updated attributes (lines 17 to 21).

Listing 4: ATL UCON Policy Excerpt

```
1  helper context Request!Subject def : loanRequests : String =
2  self.getRAttributeByName('loanRequests').value;
3
4  helper def : preAuth3(s:Request!Subject, o:Request!Object, r
       ↪:Request!Right)
5  : Boolean = s.loanRequests.toInteger() < 3;
6
7  helper def : postUpdate(s:Request!Subject, o:Request!Object)
8  : String = s.loanRequests.toInteger() + 1;
9
10
11 nodefault rule LoanRequestAuth3{
12   from
13     s:Request!UsageRequest (thisModule.preAuth3(s.subject, s
           ↪.object, s.right))
14   to
15     t:Decision!Decision(
16       value <- true,
17       updates <- update
18     ),
19     update: Decision!Update( -- updated values are returned
           ↪with the decision
20       name <- 'loanRequests',
21       newValue <- thisModule.postUpdate(s.subject, s.object)
             ↪.toString()
22     )
23 }
```

4.2.1. A word on performance

We have chosen to use model transformations as enabling technology for the checking of UCON policies to avoid building an UCON evaluation engine from scratch. This choice is key to facilitate the bridging between UCON and MDE but it is only adequate if the transformation engine is able to answer access requests fast, so that it does not constitutes a noticeable overhead. Note that the transformation engine evaluates access requests by executing the transformations derived

Table 3
Evaluation times for access-request.

Rules	10	50	100	500	1000	5000	10,000
Time (s)	0.1165	0.152	0.2065	0.4002	0.5946	1.2193	1.99

from the input policies as described above. Therefore, the relevant time is the execution time, as access requests need to be executed every time a permission request is issued, not the time to generate the transformations as this takes place only once during the design process.

In general, model transformation engines are very efficient as there has been plenty of research efforts in optimizing them using a variety of techniques. For instance, the performance of model transformation based policies was already studied in [21] with positive results (less than 1 s for access-control policies with more than 500 rules). However, the type of policies studied were different and arguably, less complex. Thus, we have designed and conducted a performance evaluation for ATL-based UCON policies.[8] For the evaluation, we evaluated an access request against policies (model transformations) of different sizes, with the number of contained rules ranging from 10 to 10,000. These policies are generated synthetically but using real and valid UCON rules as a base. Note that although single organizations with large number of users and resources having a single point of control may have many rules within a single policy, 10,000 rules is well above the normal values. As an example, the maximum policy size in existing performance evaluation works is of about four thousand [27] and one thousand [28] rules respectively.

We summarizes the results of the performance evaluation in Table 3. As it can be seen, the performance of the ATL engine for the evaluation of UCON policies is good. Access requests against policies containing up to 100 rules are evaluated in a fifth of a second. Evaluation time increases linearly with the size of the policies with access request taking half a second for policies of 1000 rules. Very large policies containing more that 5000 rules take more than a second for access request evaluation. Note however that as mentioned before, such policies are not common.

Therefore, we can conclude that model transformation engines are a feasible implementation strategy for UCON evaluation.

4.3. Policy enforcement

Once we have the means to define access-control policies and to evaluate subject access-requests to resources, we are ready to plug our access-control framework to the system to be protected. In order to do so, and having already the policy definition separated from the main application, we could use a reference monitor [29] and a Policy Enforcement Point (PEP) - Policy Decision Point(PDP) architecture. In such architecture, the PEP will capture any requests for access, and then ask the PDP to yield a decision access before granting or denying the access.

We left this integration step as further work, however, we provide a succinct description on the main run-time tasks and how they may be implemented in MDE: The PDP will be mainly composed of the model transformation engine coupled with the access-control policy encoded as a model transformation specification. The PEP will consist of catching the model access and edition operation and performing a call to the PDP before allowing the access. This can be done by extending the modeling framework as it is done in [30] where so-called *reactive* EObjects act in this manner.

[8] The evaluation tests were executed on a four core Intel(R) Core(TM) i7-8650U CPU @ 1.90 GHz running Ubuntu 20.04.2 LTS. We used ATL 4.2.0 under Eclipse Modeling Tools Version: 2020-03 (4.15.0).

Other run-time UCON operations may also take advantage of existing MDE facilities, such as the revoking of permissions when attributes change. Concretely, and w.r.t. our proposed framework, OCL predicates may be incrementally re-evaluated upon attribute's value changes as described in [31].

5. Related work

UCON and access control models have been widely studied in the literature. We classify the related works in two main categories: approaches that integrate access control models in MDE environments (Section 5.1) and approaches to provide alternative implementations and adaptations of UCON to other technical spaces (Section 5.2).

5.1. Model-driven engineering and access control models

There are a plethora of access control models that are used to implement security policies, as it can be seen in the taxonomy defined in [5]. In this context, different model-driven approaches have been used to raise the level of abstraction at which these policies are defined, especially aiming at their integration in software and system development projects. Pioneer approaches were [32] and [33]. They offer the possibility of modeling (role-based) access-control policies, but they use a rather simple access-control model and do not go beyond the specification phase, i.e. evaluation of policies is not covered.

Regarding the modeling of security policies, several model-driven approaches focus on different, more expressive, Domain-Specific Languages to deal with the concepts of one or more access control models in different contexts [34]. In that sense, [35–38] propose access control metamodels for web service-oriented architectures, web content management systems, distributed environments, and enterprise architecture, respectively. There are also attempts to integrate the concepts managed by different access control models in a unique metamodel. An example of this approach is [39], which proposes a metamodel integrating RBAC with other access control principles (Chinese wall, Bell LaPadula and BIBA). More focused on correctness, in Fadhel et al. [25], the authors define semantic checks to detect conflicts and inconsistencies among the policies written in the GemRBAC-DSL specification, a language to write RBAC. Additionally, there are approaches in the model-driven arena that use general purpose languages such as UML and OCL to formalize different properties of access-control paradigms such as static and dynamic Separation of Duty (SoD) [26,40]. Finally, in [41] OCL is proposed to represent UCON constraints (or predicates). They do so directly in an implicit model that integrates a mixture of UCON concepts, system state and domain concepts. Instead, our approach provides an explicit and complete UCON support to facilitate the definition of any UCON-based policy in modeling environments.

Regarding the enforcement of modeled security policies, [42] proposes an enforcement framework in which policies are written in GemRBAC-DSL, which allows the specification of RBAC policies. [43] uses UML and OCL to specify RBAC polices and implements PDP as a model-driven authorization engine in the context of web services. [44] presents an approach in which RBAC policies are written in a DSL defined as an extension of UML activity diagrams. These policies are then mapped to WS-BPEL specifications to be enforced at run-time. [21] defines an approach that can handle MAC, DAC, RBAC and ABAC policies and generates a PDP specification. Finally, [45] provides a general MDE-based PEP-PDP continuity-enhanced and configurable UCON enforcement engine. With it, they also provide a metamodel, but it is specialized for enforcement and, as a consequence, it is not fully aligned with the original UCON and it does not cover explicitly concepts such as subjects, objects or rights.

Table 4 summarizes this related work, showing the access control models covered by the aforementioned approaches. As it can be seen, usage control and concretely, the UCON model is only partially tackled in two works: [45] and [41] with the limitations discussed above. In short, as far as we know, ours is the first approach that provides complete UCON support in a modeling environment, from the policy definition to its execution, all in a generic way that can be reused no matter the type of software models we are interested in; and a native way that facilitates the interoperation of our solution with all the other existing tools around the EMF framework.

5.2. Usage control models

In [46] existing security approaches are surveyed and discussed following the OM-AM (Objectives, Models, Architectures, Mechanisms) design framework [47]. Following this survey, we reviewed the existing UCON approaches from the formalization, implementation, and the policy enforcement perspective.

Firstly, there is a set of approaches aiming to formalize UCON models using a variety of formal languages such as temporal logic and process algebra. The subgroup of approaches based on temporal logic include proposals based on TLA (Temporal Logic of Action) [48–52], others based on ITL (Interval Temporal Logic) [53], on OSL language [54,55], and on CTL (Computation Tree Logic) [56]. The subgroup relying on process algebra include [57–61], that formalize UCON using a policy language named POLPA (POlicy Language based on Process Algebra); [62], that uses TCP (Timed Constrained Programming), and [63] that uses CCA (Calculus of Context-aware Ambients). All of these approaches are aimed to provide a solid semantics and formalization of UCON and the verification of UCON policies but they are not focus on usability aspects since, as stated in [64], they provide hard notations, there is a lack of easy-to-use tools and also a lack of integration among formal methods and their associated tools.

Instead, our approach defines a metamodel that precisely specifies the core components of UCON, but at the same time, offers an easy interface (thanks to the integration with the EMF family of tools) to create and manipulate these components with a textual or graphical editor, and that these textual or graphical notations could be translated by means of model transformations to formal notations such as those above. In this sense, we can see both approaches as complementary. Both stem from the original conceptual UCON proposal in [49] but they realize and adapt it to different environments for different purposes.

UCON has also been specialized to specific technical domains such as operating systems [65,66], grid computing [58,59,61,67], collaborative computing [68,69], mobile computing [70,71] and cloud computing [72,73]. Nevertheless, these approaches are ad-hoc implementations for a specific scenario that cannot be exported to other domains. In contrast, our approach is more generic, reusable and flexible in the sense that it can be applied to any scenario thanks to its model-driven nature.

Finally, although the architecture and policy enforcement are not the focus of this paper, our approach bets on an architecture based on a reference monitor with PEP-PDP components as many of the UCON approaches [54,68,74–76]. The main difference is that our approach is more generic, in the sense that it could be used with other different access control models as it can be seen in [21] and leverages existing modeling infrastructure to keep the policy definition and enforcement within the same technical space. Furthermore, thanks to the use of model transformations, we can provide some other advanced capabilities such as traceability between the policy, its execution and the evaluation results.

6. Conclusions and further work

This paper presents a model-based approach to facilitate the definition of complex security policies based on the Usage Control (UCON) primitives as part of any software development project. The approach has been implemented on top of the EMF framework and validated by means of a complex running example, showing that our approach is expressive enough to capture complex UCON policies. Moreover some preliminary, promising performance measures have been obtained to show the feasibility of enforcing the modeled rules. Thanks to this bridge enabling the use of UCON in modeling environments, UCON security policies can be added to any modeling toolchain for further analysis, monitoring, enforcement and code generation.

As further work, we plan to explore in more detail some of the new application scenarios enabled by our proposal. For instance, we will link our proposed UCON metamodel with the BPMN one, to facilitate the definition of UCON policies on top of business processes, significantly enhancing the possibility of defining advanced security constraints in business process models. At the UCON DSL level, we will work on a hybrid graphical notation for UCON and on a code-generation approach to (semi)automatically implement the defined rules on top of existing PEP infrastructures. Empirical studies to validate the trade-offs between UCON and other (simpler but less powerful) languages will also be investigated.

CRediT authorship contribution statement

Antonia M. Reina Quintero: Conceptualization, Methodology, Software, Writing – original draft. **Salvador Martínez Pérez:** Conceptualization, Methodology, Software, Writing – original draft. **Ángel Jesús Varela-Vaca:** Conceptualization, Writing – original draft. **María Teresa Gómez López:** Conceptualization, Conceptualization, Writing – review & editing. **Jordi Cabot:** Conceptualization, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 101007260 (TRANSACT), the Spanish government (LOCOSS project - PID2020-114615RB-I00 and AETHER-US project - PID2020-112540RB-C44/AEI/10.13039/501100011033) and the Junta de Andalucía (METAMORFOSIS and COPERNICA - P20_01224 projects).

Appendix. Other examples of UCON rules

Listings 5, 6 and 7 contain additional examples of the three kinds of UCON rules, namely, authorizations, conditions and obligations. Concretely, Listing 5 corresponds to the preauthorisation rule introduced in Section 3 that states that the request of more than one loan is not allowed when there are previously unaccepted loans; Listing 6 corresponds to the ongoing condition introduced in Section 3 that states that the Operational Staff is allowed to review a Loan Request in office hours (9 am–17 pm); and Listing 7 corresponds to the ongoing obligation introduced in Section 3 that states that the credit supplier has to ask for the pending reports to collect the rating of the loans before sending an acceptance notification.

Table 4
Access control models that are handled by the different proposals of grouped by the kind of contribution.

Kind of contribution	Ref.	Year	Context	DAC	MAC	RBAC	ABAC	CW	BLP	BIBA	DEBAC	ANSI HRBAC	UCON
Stand-alone DSL	[35]	2007	Web services				✓					✓	
	[36]	2013	WCMS		✓								
	[37]	2014	Distributed environment		✓						✓	✓	
	[38]	2015	Enterprise architecture	✓	✓	✓	✓						
	[39]	2015	Enterprise			✓		✓	✓	✓			
	[25]	2016	General-purpose			✓							
UML extensions	[32]	2002	Security-critical systems			✓							
	[33]	2002	EJB			✓							
	[40]	2015	General purpose			✓							
	[26]	2013	General purpose			✓							
	[41]	2010	General purpose										✓
Policy enforcement	[42]	2018	General purpose			✓							
	[43]	2008	Web services			✓							
	[44]	2013	WS-BPEL			✓							
	[21]	2016	General purpose	✓	✓	✓	✓						
	[45]	2008	General purpose										✓

DAC=Discretionary Access Control; MAC=Mandatory Access Control; RBAC=Role-Based Access Control; ABAC=Attribute-Based Access Control; CW=Chinese Wall; BLP=Bell LaPadula; DEBAC=Dynamic Endpoint-Based Access Control.

Listing 5: UCON authorisation rule $preA_0$ example

```
<ucon:Policy
    xmi:version="2.0"
    xmlns:xmi="http://www.omg.org/XMI"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ucon="http://www.example.org/ucon"
    xsi:schemaLocation="http://www.example.org/ucon ucon.
        ecore">

  <subjects id="OperationalStaff">
    <attributes name="id"/>
    <attributes name="name"/>
    <attributes name="lastName"/>
    <attributes name="workFor"/>
    <attributes name="reports" mutable="true"/>
    <attributes name="lessActive" mutable="true"/>
  </subjects>

  <objects id="LoanRequest">
    <attributes name="id"/>
    <attributes name="cost"/>
    <attributes name="creditSupplier"/>
    <attributes name="customer"/>
    <attributes name="account"/>
    <attributes name="loanTerms" mutable="true"/>
  </objects>

  <rights id="R2-ReviewLoanRequest" name="ReviewLoanRequest"
      />

  <rules xsi:type="ucon:AllowRule"
      id="preA0-SoD"
      subject="OperationalStaff"
      object="LoanRequest"
      right="R2-ReviewLoanRequest">
    <implies predicate="object.operationalStaffs->excludes(
        subject)"/>
  </rules>
```

Listing 6: UCON condition rule onC_0 example

```
<ucon:Policy
    xmi:version="2.0"
    xmlns:xmi="http://www.omg.org/XMI"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ucon="http://www.example.org/ucon"
    xsi:schemaLocation="http://www.example.org/ucon ucon.
        ecore">

  <subjects id="OperationalStaff">
    <attributes name="id"/>
    <attributes name="name"/>
    <attributes name="lastName"/>
    <attributes name="workFor"/>
    <attributes name="reports" mutable="true"/>
    <attributes name="lessActive" mutable="true"/>
  </subjects>

  <objects id="LoanRequest">
    <attributes name="id"/>
    <attributes name="cost"/>
    <attributes name="creditSupplier"/>
    <attributes name="customer"/>
    <attributes name="account"/>
    <attributes name="loanTerms" mutable="true"/>
  </objects>

  <rights id="R2-ReviewLoanRequest" name="ReviewLoanRequest
      "/>

  <environment>
    <attributes name="device"/>
    <attributes name="currentTime"/>
  </environment>

  <rules xsi:type="ucon:PreconditionStopRule"
      id="onC0"
      subject="OperationalStaff"
      object="LoanRequest"
      right="R2-ReviewLoanRequest">
    <implies predicate="currentTime>17:00 and currentTime
        <00:00 or currentTime<09:00 and currentTime>00:00"/
        >
  </rules>
```

Listing 7: UCON obligation rule onB_{13} example

```
<ucon:Policy
    xmi:version="2.0"
    xmlns:xmi="http://www.omg.org/XMI"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ucon="http://www.example.org/ucon"
    xsi:schemaLocation="http://www.example.org/ucon ucon.
        ecore">

  <subjects id="TaskCollectingRatings">
    <attributes name="currentReport"/>
    <attributes name="taskReports" mutable="true"/>
    <attributes name="creditSupplier"/>
  </subjects>
  <objects id="TaskSendApproveNotification"/>

  <rules xsi:type="ucon:ObligationStopRule"
      id="onB13-a"
      subject="TaskCollectingRatings"
      object="TaskSendApproveNotification"
      right="R5-SendNotification">
    <requires xsi:type="ucon:PreUpdateOperation"
      procedure="subject.reports()->size() = subject.reports
          ()->size()@pre+1"
      target="//@subjects.3/@attributes.1"/>
    <requires xsi:type="ucon:PostUpdateOperation"
       procedure="subject.reports()->size() = subject.
           reports()->size()@pre-1"
       target="//@subjects.3/@attributes.1"/>
    <implies predicate="ob7">
      <obligations id="ob7"
          predicate="obs.reports->select(report |report.
              loanRequest = obs.currentReport.loanRequest)
              ->size()<2"
          obs="TaskCollectingRatings"
          obo="ObjectOperationalStaff"
          ob="askForPendingReports"
          duration="-1"/>
    </implies>
  </rules>
```

References

[1] von Solms B, von Solms R. The 10 deadly sins of information security management. Comput Secur 2004;23(5):371–6. http://dx.doi.org/10.1016/j.cose.2004.05.002.

[2] Sony pays dearly for attack. Comput Fraud Secur 2015;2015(2):3. http://dx.doi.org/10.1016/S1361-3723(15)30003-8.

[3] Dumas M, Recker J, Weske M. Management and engineering of process-aware information systems: Introduction to the special issue. Inf Syst 2012;37(2):77–9. http://dx.doi.org/10.1016/j.is.2011.09.003.

[4] Leitner M, Rinderle-Ma S. A systematic review on security in process-aware information systems - constitution, challenges, and future directions. Inf Softw Technol 2014;56(3):273–93. http://dx.doi.org/10.1016/j.infsof.2013.12.004.

[5] Majumder A, Namasudra S, Nath S. Taxonomy and classification of access control models for cloud environments. In: Continued rise of the cloud: Advances and trends in cloud computing. London: Springer London; 2014, p. 23–53. http://dx.doi.org/10.1007/978-1-4471-6452-4_2, Ch. 2.

[6] Park J, Sandhu RS. Towards usage control models: beyond traditional access control. In: 7th ACM symposium on access control models and technologies, SACMAT 2002, naval postgraduate school, Monterey, California, USA, June 3-4, 2002. ACM; 2002, p. 57–64. http://dx.doi.org/10.1145/507711.507722.

[7] BonitaSoft. Bonita. 2021, URL https://www.bonitasoft.com/.

[8] Red Hat. JBPM. 2021, URL https://www.jbpm.org/.

[9] Alfresco. Activiti. 2021, URL https://www.activiti.org/.

[10] Varela-Vaca AJ, Borrego D, Gómez-López MT, Gasca RM. A usage control model extension for the verification of security policies in artifact-centric business process models. In: Business information systems - 19th international conference, BIS 2016, Leipzig, Germany, July, 6-8, 2016, Proceedings. 2016, p. 289–301. http://dx.doi.org/10.1007/978-3-319-39426-8_23.

[11] Brambilla M, Cabot J, Wimmer M. Model-driven software engineering in practice. Synthesis lectures on software engineering, 2nd ed.. Morgan & Claypool Publishers; 2017, http://dx.doi.org/10.2200/S00751ED2V01Y201701SWE004.

[12] Hutchinson JE, Rouncefield M, Whittle J. Model-driven engineering practices in industry. In: Taylor RN, Gall HC, Medvidovic N, editors. Proceedings of the 33rd international conference on software engineering, ICSE 2011, Waikiki, Honolulu , HI, USA, May 21-28, 2011. ACM; 2011, p. 633–42. http://dx.doi.org/10.1145/1985793.1985882.

[13] Hailpern B, Tarr PL. Model-driven development: The good, the bad, and the ugly. IBM Syst J 2006;45(3):451–62. http://dx.doi.org/10.1147/sj.453.0451.

[14] Jouault F, Allilaire F, Bézivin J, Kurtev I. ATL: A model transformation tool. Sci Comput Program 2008;72(1–2):31–9.

[15] Bézivin J. On the unification power of models. Softw Syst Model 2005;4(2):171–88. http://dx.doi.org/10.1007/s10270-005-0079-0.

[16] Park J, Sandhu RS. The $UCON_{abc}$ usage control model. ACM Trans Inf Syst Secur 2004;7(1):128–74. http://dx.doi.org/10.1145/984334.984339.

[17] Steinberg D, Budinsky F, Paternostro M, Merks E. EMF: Eclipse modeling framework 2.0. 2nd ed.. Addison-Wesley Professional; 2009.

[18] OMG. Object Constraint Language (OCL), version 2.4. 2014.

[19] Kolovos DS, Paige RF, Polack F. The epsilon transformation language. In: Theory and practice of model transformations - 1st international conference, ICMT@TOOLS 2008, Zurich, Switzerland, July 1-2, 2008, Proceedings. LNCS, vol. 5063, Springer; 2008, p. 46–60. http://dx.doi.org/10.1007/978-3-540-69927-9_4.

[20] OMG. MOF and specification, QVT final adopted. 2007.

[21] Martínez Pérez S, García J, Cabot J. Runtime support for rule-based access-control evaluation through model-transformation. In: Proceedings of the 2016 ACM SIGPLAN international conference on software language engineering, Amsterdam, the Netherlands, October 31 - November 1, 2016. ACM; 2016, p. 57–69.

[22] Mpardis G, Kotsilieris T. Bank loan processes modelling using BPMN. In: 2010 developments in e-systems engineering. 2010, p. 239–42. http://dx.doi.org/10.1109/DeSE.2010.45.

[23] Weske M. Business process management - Concepts, languages, architectures. 2nd ed.. Springer; 2012, http://dx.doi.org/10.1007/978-3-642-28616-2.

[24] Pérez-Álvarez JM, Gómez-López MT, Eshuis R, Montali M, Gasca RM. Verifying the manipulation of data objects according to business process and data models. Knowl Inf Syst 2020;62(7):2653–83. http://dx.doi.org/10.1007/s10115-019-01431-5.

[25] Fadhel AB, Bianculli D, Briand LC. GemRBAC-DSL: A high-level specification language for role-based access control policies. In: Proceedings of the 21st ACM on symposium on access control models and technologies, SACMAT 2016, Shanghai, China, June 5-8, 2016. ACM; 2016, p. 179–90. http://dx.doi.org/10.1145/2914642.2914656.

[26] Kuhlmann M, Sohr K, Gogolla M. Employing UML and OCL for designing and analysing role-based access control. Math Struct Comput Sci 2013;23(4):796–833. http://dx.doi.org/10.1017/S0960129512000266.

[27] Liu AX, Chen F, Hwang J, Xie T. Xengine: a fast and scalable XACML policy evaluation engine. ACM SIGMETRICS Perform Eval Rev 2008;36(1):265–76.

[28] Turkmen F, Crispo B. Performance evaluation of XACML PDP implementations. In: Proceedings of the 2008 ACM workshop on secure web services. 2008; p. 37–44.

[29] ITU-T, International Standards Organisation. Information technology – Open Systems Interconnection — Security frameworks for open systems: Access control framework (ITU-T Rec X.812 / ISO/IEC-10181-3:1996). International Standard ISO-10181-3/X.812; 1996.

[30] Martínez Pérez S, Tisi M, Douence R. Reactive model transformation with ATL. Sci Comput Program 2017;136:1–16. http://dx.doi.org/10.1016/j.scico.2016.08.006.

[31] Jouault F, Tisi M. Towards incremental execution of ATL transformations. In: Theory and practice of model transformations - 3rd international conference, ICMT@TOOLS 2010, MáLaga, Spain, June 28-July 2, 2010. Proceedings. LNCS, vol. 6142, Springer; 2010, p. 123–37. http://dx.doi.org/10.1007/978-3-642-13688-7_9.

[32] Jürjens J. UMLsec: Extending UML for secure systems development. In: UML 2002 - the Unified Modeling Language, 5th international conference, Dresden, Germany, September 30 - October 4, 2002, Proceedings. LNCS, vol. 2460, Springer; 2002, p. 412–25. http://dx.doi.org/10.1007/3-540-45800-X_32.

[33] Lodderstedt T, Basin DA, Doser J. SecureUML: A UML-based modeling language for model-driven security. In: UML 2002 - the Unified Modeling Language, 5th international conference, Dresden, Germany, September 30 - October 4, 2002, Proceedings. LNCS, vol. 2460, Springer; 2002, p. 426–41. http://dx.doi.org/10.1007/3-540-45800-X_33.

[34] Kashmar N, Adda M, Atieh M. From access control models to access control metamodels: A survey. In: Advances in information and communication. Cham: Springer International Publishing; 2020, p. 892–911.

[35] Emig C, Brandt F, Abeck S, Biermann J, Klarl H. An access control metamodel for web service-oriented architecture. In: Proceedings of the second international conference on software engineering advances (ICSEA 2007), August 25-31, 2007, Cap Esterel, French Riviera, France. IEEE Computer Society; 2007, p. 57. http://dx.doi.org/10.1109/ICSEA.2007.15.

[36] Martínez Pérez S, García-Alfaro J, Cuppens F, Cuppens-Boulahia N, Cabot J. Towards an access-control metamodel for web content management systems. In: Current trends in web engineering - ICWE 2013 international workshops composableWeb, QWE, MDWE, DMSSW, EMotions, CSE, SSN, and PhD Symposium, Aalborg, Denmark, July 8-12, 2013. Revised selected papers. LNCS, vol. 8295, Springer; 2013, p. 148–55. http://dx.doi.org/10.1007/978-3-319-04244-2_14.

[37] Bertolissi C, Fernández M. A metamodel of access control for distributed environments: Applications and properties. Inform and Comput 2014;238:187–207. http://dx.doi.org/10.1016/j.ic.2014.07.009.

[38] Korman M, Lagerström R, Ekstedt M. Modeling enterprise authorization: A unified metamodel and initial validation. CSIMQ 2016;7:1–24. http://dx.doi.org/10.7250/csimq.2016-7.01.
[39] Abd-Ali J, Guemhioui KE, Logrippo L. A metamodel for hybrid access control policies. JSW 2015;10(7):784–97. http://dx.doi.org/10.17706/jsw.10.7.784-797.
[40] Fadhel AB, Bianculli D, Briand LC. A comprehensive modeling framework for role-based access control policies. J Syst Softw 2015;107:110–26. http://dx.doi.org/10.1016/j.jss.2015.05.015.
[41] Li M, Wang H. Specifying usage control model with Object Constraint Language. In: Fourth international conference on network and system security, NSS 2010, Melbourne, Victoria, Australia, September 1-3, 2010. IEEE Computer Society; 2010, p. 391–7. http://dx.doi.org/10.1109/NSS.2010.10.
[42] Fadhel AB, Bianculli D, Briand LC. Model-driven run-time enforcement of complex role-based access control policies. In: Proceedings of the 33rd ACM/IEEE international conference on automated software engineering, ASE 2018, Montpellier, France, September 3-7, 2018. ACM; 2018, p. 248–58. http://dx.doi.org/10.1145/3238147.3238167.
[43] Sohr K, Mustafa T, Bao X, Ahn G. Enforcing role-based access control policies in web services with UML and OCL. In: Twenty-fourth annual computer security applications conference, ACSAC 2008, Anaheim, California, USA, 8-12 December 2008. IEEE Computer Society; 2008, p. 257–66. http://dx.doi.org/10.1109/ACSAC.2008.35.
[44] Hummer W, Gaubatz P, Strembeck M, Zdun U, Dustdar S. Enforcement of entailment constraints in distributed service-based business processes. Inf Softw Technol 2013;55(11):1884–903. http://dx.doi.org/10.1016/j.infsof.2013.05.001.
[45] Katt B, Zhang X, Breu R, Hafner M, Seifert J. A general obligation model and continuity: Enhanced policy enforcement engine for usage control. In: 13th ACM symposium on access control models and technologies, SACMAT 2008, Estes Park, CO, USA, June 11-13, 2008, Proceedings. ACM; 2008, p. 123–32. http://dx.doi.org/10.1145/1377836.1377856.
[46] Lazouski A, Martinelli F, Mori P. Usage control in computer security: A survey. Comput Sci Rev 2010;4(2):81–99. http://dx.doi.org/10.1016/j.cosrev.2010.02.002.
[47] Sandhu RS. Engineering authority and trust in cyberspace: the OM-AM and RBAC way. In: Rebensburg K, Youman CE, Atluri V, editors. Fifth ACM workshop on role-based access control, RBAC 2000, Berlin, Germany, July 26-27, 2000. ACM; 2000, p. 111–9. http://dx.doi.org/10.1145/344287.344309.
[48] Zhang X. Formal model and analysis of usage control (Ph.D. thesis), USA: George Mason University; 2006.
[49] Zhang X, Parisi-Presicce F, Sandhu RS, Park J. Formal model and policy specification of usage control. ACM Trans Inf Syst Secur 2005;8(4):351–87. http://dx.doi.org/10.1145/1108906.1108908.
[50] Zhang X, Park J, Parisi-Presicce F, Sandhu RS. A logical specification for usage control. In: Jaeger T, Ferrari E, editors. 9th ACM symposium on access control models and technologies, SACMAT 2004, Yorktown Heights, New York, USA, June 2-4, 2004, Proceedings. ACM; 2004, p. 1–10. http://dx.doi.org/10.1145/990036.990038.
[51] Zhang X, Sandhu RS, Parisi-Presicce F. Safety analysis of usage control authorization models. In: Lin F, Lee D, Lin BP, Shieh S, Jajodia S, editors. Proceedings of the 2006 ACM symposium on information, computer and communications security, ASIACCS 2006, Taipei, Taiwan, March 21-24, 2006. ACM; 2006, p. 243–54. http://dx.doi.org/10.1145/1128817.1128853.
[52] Gouglidis A, Grompanopoulos C, Mavridou A. Formal verification of usage control models: A case study of usecon using TLA+. In: Bliudze S, Bensalem S, editors. Proceedings of the 1st international workshop on methods and tools for rigorous system design, MeTRiD@ETAPS 2018, Thessaloniki, Greece, 15th April 2018. EPTCS, vol. 272, 2018, p. 52–64. http://dx.doi.org/10.4204/EPTCS.272.5.
[53] Janicke H, Cau A, Zedan H. A note on the formalisation of UCON. In: Lotz V, Thuraisingham BM, editors. 12th ACM symposium on access control models and technologies, SACMAT 2007, Sophia Antipolis, France, June 20-22, 2007, Proceedings. ACM; 2007, p. 163–8. http://dx.doi.org/10.1145/1266840.1266867.
[54] Pretschner A, Hilty M, Basin DA. Distributed usage control. Commun ACM 2006;49(9):39–44. http://dx.doi.org/10.1145/1151030.1151053.
[55] Hilty M, Pretschner A, Basin DA, Schaefer C, Walter T. A policy language for distributed usage control. In: Biskup J, López J, editors. Computer security - ESORICS 2007, 12th European symposium on research in computer security, Dresden, Germany, September 24-26, 2007, Proceedings. Lecture notes in computer science, vol. 4734, Springer; 2007, p. 531–46. http://dx.doi.org/10.1007/978-3-540-74835-9_35.
[56] Lili X, Zhigang Z. Formal specification of concurrent enforcement UCON model with CTL logic. In: Sun X, Pan Z, Bertino E, editors. Artificial intelligence and security - 5th international conference, ICAIS 2019, New York, NY, USA, July 26-28, 2019, Proceedings, Part II. Lecture notes in computer science, vol. 11633, Springer; 2019, p. 627–41. http://dx.doi.org/10.1007/978-3-030-24265-7_54.
[57] Baiardi F, Martinelli F, Mori P, Vaccarelli A. Improving grid services security with fine grain policies. In: Meersman R, Tari Z, Corsaro A, editors. On the move to meaningful internet systems 2004: OTM 2004 workshops: OTM confederated international workshops and posters, GADA, JTRES, MIOS, WORM, WOSE, PhDS, and INTEROP 2004, Agia Napa, Cyprus, October 25-29, 2004. Proceedings. Lecture notes in computer science, vol. 3292, Springer; 2004, p. 123–34. http://dx.doi.org/10.1007/978-3-540-30470-8_30.
[58] Martinelli F, Mori P. A model for usage control in GRID systems. In: Third international conference on security and privacy in communication networks and the workshops, SecureComm 2007, Nice, France, 17-21 September, 2007. IEEE; 2007, p. 169–75. http://dx.doi.org/10.1109/SECCOM.2007.4550326.
[59] Martinelli F, Mori P, Vaccarelli A. Towards continuous usage control on grid computational services. In: Joint international conference on autonomic and autonomous systems 2005 / International conference on networking and services 2005, ICAS/ICNS 2005, Papeete, Tahiti, France, October 23-28, 2005. IEEE Computer Society; 2005, p. 82. http://dx.doi.org/10.1109/ICAS-ICNS.2005.93.
[60] Massonet P, Arenas A, Martinelli F, Mori P, Crispo B. GridTrust – A usage control based trust and decurity framework for service-based grids. In: At your service: Service engineering in the information society technologies program. MIT Press; 2008, p. 407–27, Ch. 16.
[61] Koshutanski H, Lazouski A, Martinelli F, Mori P. Enhancing grid security by fine-grained behavioral control and negotiation-based authorization. Int J Inf Secur 2009;8(4):291–314. http://dx.doi.org/10.1007/s10207-009-0083-4.
[62] Jagadeesan R, Marrero W, Pitcher C, Saraswat VA. Timed constraint programming: a declarative approach to usage control. In: Barahona P, Felty AP, editors. Proceedings of the 7th International ACM SIGPLAN conference on principles and practice of declarative programming, July 11-13 2005, Lisbon, Portugal. ACM; 2005, p. 164–75. http://dx.doi.org/10.1145/1069774.1069790.
[63] Almutairi A, Siewe F. Formal specification of CA-UCON model using CCA. In: 2013 science and information conference. 2013, p. 369–75.
[64] Gargantini A, Riccobene E, Scandurra P. Integrating formal methods with model-driven engineering. In: Boness K, Fernandes JM, Hall JG, Machado RJ, Oberhauser R, editors. The fourth international conference on software engineering advances, ICSEA 2009, 20-25 September 2009, Porto, Portugal. IEEE Computer Society; 2009, p. 86–92. http://dx.doi.org/10.1109/ICSEA.2009.22.
[65] Xu M, Jiang X, Sandhu RS, Zhang X. Towards a VMM-based usage control framework for OS kernel integrity protection. In: Lotz V, Thuraisingham BM, editors. 12th ACM symposium on access control models and technologies, SACMAT 2007, Sophia Antipolis, France, June 20-22, 2007, Proceedings. ACM; 2007, p. 71–80. http://dx.doi.org/10.1145/1266840.1266852.
[66] Kyle D, Brustoloni JC. Uclinux: a linux security module for trusted-computing-based usage controls enforcement. In: Ning P, Atluri V, Xu S, Yung M, editors. Proceedings of the 2nd ACM workshop on scalable trusted computing, STC 2007, Alexandria, VA, USA, November 2, 2007. ACM; 2007, p. 63–70. http://dx.doi.org/10.1145/1314354.1314371.
[67] Stagni F, Arenas A, Aziz B, Martinelli F. On usage control in data grids. In: Ferrari E, Li N, Bertino E, Karabulut Y, editors. Trust management III, third IFIP WG 11.11 international conference, IFIPTM 2009 , West Lafayette, in, USA, June 15-19, 2009. Proceedings. IFIP advances in information and communication technology, vol. 300, Springer; 2009, p. 99–116. http://dx.doi.org/10.1007/978-3-642-02056-8_7.
[68] Zhang X, Nakae M, Covington MJ, Sandhu RS. A usage-based authorization framework for collaborative computing systems. In: Ferraiolo DF, Ray I, editors. 11th ACM symposium on access control models and technologies, SACMAT 2006, Lake Tahoe, California, USA, June 7-9, 2006, Proceedings. ACM; 2006, p. 180–9. http://dx.doi.org/10.1145/1133058.1133084.
[69] Zhang X, Nakae M, Covington MJ, Sandhu RS. Toward a usage-based security framework for collaborative computing systems. ACM Trans Inf Syst Secur 2008;11(1):3:1–36. http://dx.doi.org/10.1145/1330295.1330298.
[70] Hilty M, Pretschner A, Schaefer C, Walter T. Usage control requirements in mobile and ubiquitous computing applications. In: Proceedings of the international conference on systems and networks communications (ICSNC 2006), October 29 - November 3, 2006, Papeete, Tahiti, French Polynesia. IEEE Computer Society; 2006, p. 27. http://dx.doi.org/10.1109/ICSNC.2006.75.
[71] Wang H, Zhang Y, Cao J. Ubiquitous computing environments and its usage access control. In: Jia X, editor. Proceedings of the 1st international conference on scalable information systems, Infoscale 2006, Hong Kong, May 30-June 1, 2006. ACM international conference proceeding series, vol. 152, ACM; 2006, p. 6. http://dx.doi.org/10.1145/1146847.1146853.
[72] Chen D, Huang X, Ren X. Access control of cloud service based on UCON. In: Jaatun MG, Zhao G, Rong C, editors. Cloud computing, first international conference, CloudCom 2009, Beijing, China, December 1-4, 2009. Proceedings. Lecture notes in computer science, vol. 5931, Springer; 2009, p. 559–64. http://dx.doi.org/10.1007/978-3-642-10665-1_52.
[73] Tavizi T, Shajari M, Dodangeh P. A usage control based architecture for cloud environments. In: 26th IEEE international parallel and distributed processing symposium workshops & PhD Forum, IPDPS 2012, Shanghai, China, May 21-25, 2012. IEEE Computer Society; 2012, p. 1534–9. http://dx.doi.org/10.1109/IPDPSW.2012.193.
[74] Syalim A, Tabata T, Sakurai K. Usage control model and architecture for data confidentiality in a database service provider. Inf Media Technol 2006;1(2):762–7. http://dx.doi.org/10.11185/imt.1.762.

[75] Liu Q, Safavi-Naini R, Sheppard NP. Digital rights management for content distribution. In: Johnson CW, Montague P, Steketee C, editors. ACSW frontiers 2003, 2003 ACSW workshops - the australasian information security workshop (AISW) and the workshop on wearable, invisible, context-aware, ambient, pervasive and ubiquitous computing (WICAPUC), Adelaide, South Australia, February 2003. CRPIT, vol. 21, Australian Computer Society; 2003, p. 49–58.

[76] Schaefer C. Usage control reference monitor architecture. In: Georgiadis P, López J, Gritzalis S, Marias GF, editors. Third international workshop on security, privacy and trust in pervasive and ubiquitous computing, SECPerU 2007, Istanbul, Turkey, July 19, 2007. IEEE Computer Society; 2007, p. 13–8. http://dx.doi.org/10.1109/SECPERU.2007.20.