Computer Science and Information Systems 19(2):547–572 https://doi.org/10.2298/CSIS210217061E

Reasoning on the Usage Control Security policies over Data Artifact Business Process Models

Montserrat Estañol[1], Ángel Jesús Varela-Vaca[2], María Teresa Gómez-López[2], Ernest Teniente[1], and Rafael M. Gasca[2]

[1] Universitat Politècnica de Catalunya
Barcelona, Spain
{estanyol,teniente}@essi.upc.edu
[2] Universidad de Sevilla
Sevilla, Spain
{ajvarela,maytegomez,gasca}@us.es

Abstract. The inclusion of security aspects in organizations is a crucial aspect to ensure compliance with both internal and external regulations. Business process models are a well-known mechanism to describe and automate the activities of the organizations, which should include security policies to ensure the correct performance of the daily activities. Frequently, these security policies involve complex data which cannot be represented using the standard Business Process Model Notation (BPMN). In this paper, we propose the enrichment of the BPMN with a UML class diagram to describe the data model, that is also combined with security policies defined using the UCON$_{ABC}$ framework annotated within the business process model. The integration of the business process model, the data model, and the security policies provides a context where more complex reasoning can be applied about the satisfiability of the security policies in accordance with the business process and data models. To do so, we transform the original models, including security policies, into the BAUML framework (an artifact-centric approach to business process modelling). Once this is done, it is possible to ensure that there are no inherent errors in the model (verification) and that it fulfils the business requirements (validation), thus ensuring that the business process and the security policies are compatible and that they are aligned with the business security requirements.

Keywords: Business Process, Security policy, Usage control model, Data artifact, Reasoning.

1. Introduction

Business processes specify the workflow of the activities in an organisation facilitating decision-making support [43] to achieve its objectives. These activities are not carried out in a void, but in many cases, they have to follow certain compliance rules which govern the operation of a company [17]. In this respect, compliance rules are also used for risk management [60] to control threats. Thus, a set of compliance rules may refer to security policies to control security threats. According to the SANS Institute definition, a security policy is a set of security requirements or rules (i.e., access control restrictions) that must be met in order to achieve the business goals. Thereby in this paper, we assume that a set of rules may represent a security policy of an organisation. The necessity of including security policies in business process models is well-known and has been studied in the literature [33], but security issues are mostly overlooked by default and not tackled in a practical way.

Security policies are not necessarily defined at the same time as the business process; rather, they are usually defined and implemented at later stages of software development. As a result, many times they are specified independently from one another. However, security policies and business process must be aligned [2]. Ensuring this will reduce risky situations and the propagation of errors during process deployment [53]. The combination of both business processes and policy rules is fundamental for the Business Continuity, as described in ISO 22301:2012.

Unfortunately, traditional approaches to process modelling are insufficient when it comes to defining security policies. The inclusion of security controls into process-aware information systems is currently an open challenge [26]. Most of the process-centric approaches try to incorporate access control mechanisms by adapting traditional access control models [29]. However, they fail to incorporate the flexible and complex security policies that modern business information systems demand. These process-centric approaches tend to focus on representing the sequence of activities in the process and disregard or place little importance on the data required. However, security policies may refer to complex data, which cannot be represented through the process-centric approaches. Further, security policies can represent restrictions regarding the number of uses of the resources. One such example is found in the context of a customer's loan request to a credit provider [32]. A security requirement could be that *it is not possible to request a loan when there are previously denied loan requests* or *a staff member is not allowed to review more than ten loan requests in a period of time*. Note that, due to the lack of a data model, it would be impossible to represent this policy.

On the other hand, artifact-centric approaches incorporate data in the definition of the process, and are more appropriate when security policies are involved, since it will be possible to represent them. Not only this but following an artifact-centric approach makes it possible to apply reasoning techniques to the process model and the security policies. Through these techniques, it can be checked that the process model and the security policies are aligned, i.e. there are no contradictions between them (verification), and that they fulfil the business requirements (validation).

As mentioned previously, the importance of specifying security-aware business processes is well-known [5]. Moreover, security policy complexity has been studied in [50], but it was only sketched how they could be modelled in an artifact-centric paradigm. In terms of reasoning, existing artifact-centric approaches do not yet consider verification and validation of security policies. For these reasons, there is a need for a proposal that deals with the verification of the security of business processes. As we have explained, it should be based on the use of an artifact-centric approach to be able to represent complex data structures related to security policies. Therefore, the main challenges to tackle are twofold: 1) the specification security policies based on UCON models into artifact-centric process models, and; 2) the provision of reasoning techniques to verify the security policies using an artifact-centric approach.

Summarising, the main contributions of this paper are:

1. **Definition of an enriched model that includes security policies over data artifacts**. We have defined a model which enables the definition of UCON-based secu-

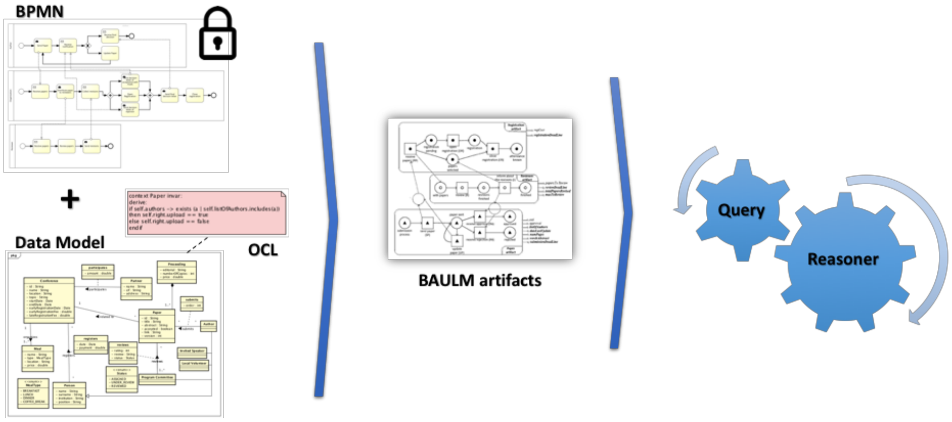

BPMN
+
Data Model
context Paper inwar:
derive:
if self.authors -> exists (a | self.listOfAuthors.includes(a))
then self.right.upload == true
else self.right.upload == false
endif
OCL
BAULM artifacts
Query
Reasoner

Fig. 1. Proposed Framework

rity policies [39] for artifact-centric process models. To do this, we use BPMN [36] and UML class diagram [37], which are *de facto* standards for process modelling and data specification respectively. We combine and enrich both notations by using security policies defined in Object Constraint Language (OCL) [10] following the UCON$_{ABC}$ model. This provides a data artifact process model with security policies defined in OCL.

2. **Transformation of partial models into the BAUML framework**. We have defined and implemented a transformation of the initial models into a BAUML framework [15] which can be used for reasoning on the model, detecting potential errors and ensuring that it fulfils the requirements and goals.
3. **Reasoning using the enriched model**. After a transformation process, we propose to reuse existing techniques [15] for verifying and validating the artifact-model as a whole, considering also the security policies and the data involved in the policies into a BPMN model.
4. **Evaluation of feasibility.** We have used a running example as a Proof-of-Concept (PoC) during the explanation in each stage and to demonstrate how our approach can reach the verification of a security policy.

The remainder of the paper is structured as follows. Section 2 presents the enriched model and the different parts that form it with an example. Section 3 details how these initial models can be translated into the BAUML framework for reasoning. Section 4 presents the types of reasoning that can be tackled thanks to using the enriched model. Section 5 analyses the related work. Finally, Section 6 presents the conclusions and further work.

2. Enriching Process Models with Security Policies

This section presents the models used in our approach: the UML class diagram, the BPMN diagram, OCL operation contracts and the UCON$_{ABC}$ framework to represent security policies. In addition, we introduce a running example, to make our proposal easier to understand.

The UML class diagram is used to represent the data in the domain of interest, and the BPMN diagram to model the process. Both models are interrelated in so far as UML diagrams are able to represent the data and their relations while BPMN provides an activity-centric perspective about the activities that can make changes to the data. Both languages are the standard and most common formalisms for representing data and processes, respectively. In addition, we use OCL operation contracts to formally specify each task in the business process, similarly to [12, 38]. This provides the ability for reasoning or executing the resulting models. We then enrich the models including security policies defined using the UCON$_{ABC}$ framework.

To illustrate our approach, a running example based on the customer's loan request [32] is used through the paper. The running example consists of a loan request to a credit provider which considers two acceptance reports before deciding on the request.

2.1. UML Class Diagram

A UML class diagram is formed of a set of classes (or concepts), which may be in a hierarchy, n-ary associations among such classes (where some of them might be reified, i.e, association classes), and some attributes inside these classes. In addition, a UML schema might be annotated with minimum/maximum multiplicity constraints over its association-ends/attributes, and hierarchy constraints (i.e., disjoint/complete constraints).

Figure 2 shows the UML class diagram representing the data required by the process in our running example. For example, a *LoanRequest* is defined by its *id*, *amount*, *pending*, *accepted*, *date* and *risky*. *Pending* and *accepted* attributes represent whether the loan is waiting for approval or has been accepted, respectively. In turn, a *Customer* may submit several loan requests, which are going to be revised by the *Operation Staff* of the *Credit Provider* to decide on the *risk* and the *rate* of the loan.

2.2. BPMN Diagram

BPMN (Business Process Model and Notation) is a widely used and well-known ISO and OMG standard language for modelling business processes known as the defacto standard for business process modelling [25]. In a nutshell, the language uses nodes to represent the activities or tasks of the process, whose execution order is determined by a set of directed edges. Different gateway nodes are available to control the flow, to allow for parallel or alternative execution paths, for instance. Moreover, using BPMN it is also possible to represent the interaction between different parties involved in the process, messages and business objects are able to flow between various business processes [41].

The BPMN model is shown in Figure 3 where three different pools cover its main functions: pool *Customer* manages the request process from the viewpoint of a customer; pool *Credit Provider* describes how the administration staff manages the loan request, obtains the acceptance reports and notifies customers about the decisions; and pool *Operation Staff* deals with the evaluation of the loan requests. We use in the example some collaboration components since various data with different cardinalities flow through business process instances [40, 31]. Single instances of the process *Credit Provider* need to

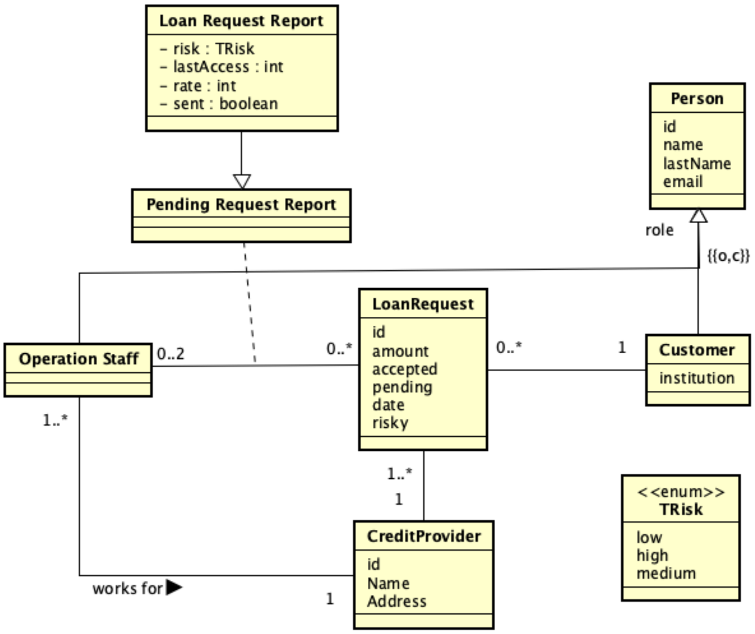


Fig. 2. Data Model of the Loan Request

interact and be synchronised with multiple instances of processes *Customer* and *Operation Staff* simultaneously [18]. Loop Activities (↻) and Parallel Activities (|||) are included to describe the synchronisation between the pools. Data objects, such as *LoanRequest* or *Customer*, appear in the BPMN diagram but their details are modelled in the UML class diagram, as we have shown.

2.3. OCL Operation Contracts

In order to define the behaviour of the tasks that perform work in the BPMN model, we propose the use of OCL operation contracts. Each contract contains: a header, including the operation name and input parameters; a precondition, stating the conditions that *must* be true before the task can be executed; and a postcondition, describing the state of the system *after* the successful execution of the activity. Below we present the contracts of two tasks of the example, *RequestALoan* and *SendApprovedNotification*. OCL contracts for the other activities would be defined similarly.

```
RequestALoan(pId: String, am: int, c: Customer, t: Date, cp: CreditProvider,
     r:boolean)
post: LoanRequest.allInstances()->exists(l | l.oclIsNew() and l.id = pId and
     l.amount = am and l.date = t and l.accepted = false and l.pending = true and
     l.creditProvider = cp and l.customer = c and l.risky=r)
```

Activity *ReceiveNotification* has no OCL operation contract because it waits until receiving a message through the incoming message flow.

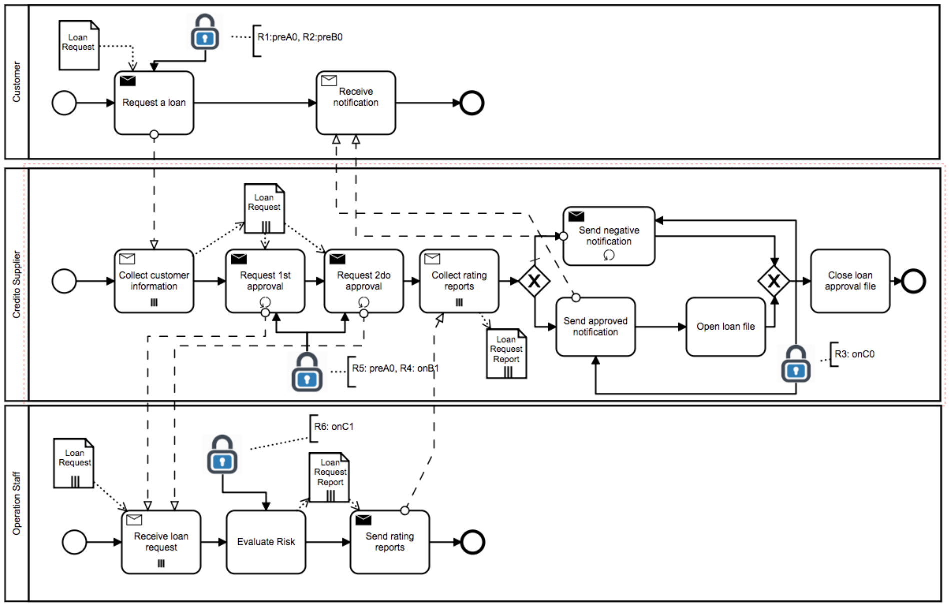


Fig. 3. BPMN Model for the Loan Request

```
SendApprovedNotification(l: LoanRequest, acc: boolean)
post: l.accepted = acc and l.pending = false
```

2.4. Describing Security Policies

Finally, the last elements of our proposal are the security policies. They can be seen as business compliance rules [17] with specific security semantics, such as Separation of Duties (SoD) [9]. There is not a standard framework or formalism to specify them. In this paper, we follow the UCON$_{ABC}$ model [39] which has emerged as a generic formal model to represent complex, adaptable and flexible security policies in new environments, such as Internet of Things (IoT). For instance, Digital Right Management (DRM) is an access control mechanism which can be modelled by UCON$_{ABC}$. Moreover, other traditional access control and trust management mechanisms can be defined by using this model. A UCON$_{ABC}$ model consists of the following components:

- A *Subject* is a component which holds or exercises certain rights on objects. An *Object* is an entity that a subject can access or use with certain rights.
- *Rights* are privileges that a subject can hold and exercise on an object.
- Predicates for the evaluation:
  - *Authorisations* (A) have to be evaluated for usage decisions and return whether the subject (requester) is allowed to perform the requested rights on the object or not.
  - *Obligations* (B) represent functional predicates that verify mandatory requirements a subject has to perform before or during a usage exercise.

- *Conditions* (C) evaluate environmental or systems factors to check whether relevant requirements are satisfied or not.

All these predicates can be evaluated before or while the rights are exercised. For this reason, the UCON$_{ABC}$ model splits each predicate into two types of sub-predicates depending on when it must be evaluated: (1) a pre-Authorisation (*preA*) predicate is evaluated before a requested right is exercised; and (2) an on-Authorisation (*onA*) predicate is checked while the right is exercised. Likewise, obligations and conditions can be divided into pre- and on-predicates. Further, UCON$_{ABC}$ introduces a new factor to be considered in predicates: the use of inmutable or upgradeable attributes. Thus, in certain predicates we may need to check the conditions based on the inmutability or the updating of subject and/or object's attributes. To summarise, all types of predicates and updating predicates supported are given in Table 1 as indicated in [39]. For instance, an onA_3 predicate represents a usage control scenario where the access decision is evaluated during beginning the usage and it also requires a post-update of attributes during the access. Furthermore, thanks to the use of UCON, it enables us to represent traditional access control (e.g., MAC) by means of $preA_0$ and $preA_1$ predicates.

Table 1. The predicates supported by $UCON_{ABC}$ [39].

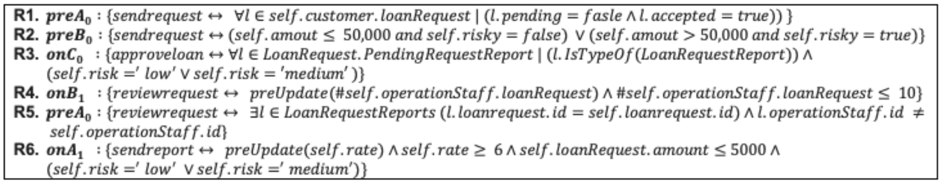
	0 (inmutable)	**1 (pre-update)**	**2 (ongoing-update)**	**3 (post-update)**
preA	✓	✓	✗	✓
onA	✓	✓	✓	✓
preB	✓	✓	✗	✓
onB	✓	✓	✓	✓
preC	✓	✗	✗	✗
onC	✓	✗	✗	✗

The policies according to [50] are shown in Figure 4. Although our approach can support most of the predicates, just some of them have been used in the running example. They have been modelled together with the BPMN model (cf. Figure 3). The reason behind this is that these policies can help to improve the documentation, the analysis and optimisation of the process model, and the alignment of systems according to the given security requirements, as proposed in [34].

The BPMN model has been designed in an extension of the bpmn.io modeller which integrates a security DSL [52] which enables the graphical specification of security policies employing locks (cf. locks in the diagram) attached to activities. Although the type of rules have been pointed out as text annotations attached to each security policy.

For each rule, the left-hand side of the double implication refers to the *right*, whereas the right-hand side states what needs to be evaluated. For example, policy *onB*$_1$ (cf. R4) states that the staff can only review a maximum of 10 loan requests. Some rules use a *preUpdate* predicate in the right-hand side. It establishes an update of an object's attribute prior to the usage. For instance, *onA*$_1$ (cf. R6) checks if the rate of a loan requests is greater or equal than six and the risk assigned is *low* or *medium*. In terms of subjects and objects, *onC*$_0$ (cf. R4) is an *on condition* (*C*), and its subject is the credit provider and its objects are a loan request and a loan request report.

For a better understanding, a brief description of the security policies is given below:



R1. $\boldsymbol{preA_0}$: $\{sendrequest \leftrightarrow\ \forall l \in self.customer.loanRequest \mid (l.pending = fasle \wedge l.accepted = true))\ \}$
R2. $\boldsymbol{preB_0}$: $\{sendrequest \leftrightarrow (self.amout \leq\ 50{,}000\ and\ self.risky = false)\ \vee (self.amout > 50{,}000\ and\ self.risky = true)\}$
R3. $\boldsymbol{onC_0}$: $\{approveloan \leftrightarrow \forall l \in LoanRequest.PendingRequestReport \mid (l.IsTypeOf(LoanRequestReport)) \wedge$
$(self.risk =' low' \vee self.risk = 'medium')\}$
R4. $\boldsymbol{onB_1}$: $\{reviewrequest \leftrightarrow\ preUpdate(\#self.operationStaff.loanRequest) \wedge \#self.operationStaff.loanRequest \leq\ 10\}$
R5. $\boldsymbol{preA_0}$: $\{reviewrequest \leftrightarrow\ \exists l \in LoanRequestReports\ (l.loanrequest.id = self.loanrequest.id) \wedge l.operationStaff.id\ \neq$
$self.operationStaff.id\}$
R6. $\boldsymbol{onA_1}$: $\{sendreport \leftrightarrow\ preUpdate(self.rate) \wedge self.rate \geq\ 6 \wedge self.loanRequest.amount \leq 5000 \wedge$
$(self.risk =' low'\ \vee self.risk =' medium')\}$

Fig. 4. Security Policies for the Loan Request

1. **R1** : $preA_0$: the request of more than one loan is not permitted when there are previously unaccepted loans.
2. **R2** : $preB_0$: the loan requests of more than fifty thousand are not permitted for the credit provider without accepting a clause of risk.
3. **R3** : onC_0: the loan request is accepted iff risk is medium or low.
4. **R4** : onB_1: an operation staff cannot review more than ten loan requests at the same time.
5. **R5** : $preA_0$: the second review must not be the same than the first one (Separation of Duties principles).
6. **R6** : onA_1: the reviewer is able to send a loan request report with a rate of six (or greater), an amount less than five thousand, and risk low or medium since requires supervision (Four eye principle).

3. Transforming the Models into an Integrated Solution

Given the models described in the previous section, our goal is to determine the correctness of the business process model as a whole (i.e. considering the data and process models, and the definition of the activities) and its security policies. This means checking that there are no errors, and that the requirements are fulfilled when the models and security policies are considered together, basing our reasoning approach on [15].

Three steps need to be carried out to achieve this objective:

1. Formalize the security policies, so that they can be incorporated into the models.
2. Transform our starting models to be able to reason with them.
3. Perform the reasoning itself, after merging the formalized security policies with the model.

The first two steps are described in the remainder of this section. Step 3 is explained in section 4.

3.1. Formalizing Security Policies Utilizing OCL Constraints

The security policies defined in Figure 4 give an intuitive idea of their meaning, but as they are, cannot be added to the model for reasoning due to a lack of formalization. To solve this, we specify them using OCL language. This is not a limitation since the transformation from informal models to specific formal models has been tackled in previous works [42] and using OCL as a formalism to specify $UCON_{ABC}$ policies has been considered in previous work [27].

Each security policy defines conditions over a set objects. We propose representing a security policy as follows:

```
<SecPolicyName>
Objects: <obj$_1$>:<Type$_1$>, ..., <obj$_N$>:<Type$_N$>
Condition: <OCL expression>
```

where `OCL expression` refers to `obj`$_1$ to `obj`$_N$ using OCL constructs and should result in a Boolean value. Note that these objects should either be input parameters of the tasks to which the policies are attached or be created by them.

We also allow the use of `@post` in `OCL expression`, to refer the new value of an attribute. This is necessary for policies that include the expression *preUpdate*, indicating that the new value of the element should be considered. Below, we show policies $R1$, $R2$, $R4$ and $R6$ expressed in OCL:

R1 ($preA_0$):

```
Objects: c:Customer
Condition: c.loanRequest->forAll(l | l.pending=false and l.accepted=true)
```

R2 ($preB_0$):

```
Objects: lr:LoanRequest
Condition: (lr.amount ≤ 50,000 ∧ lr.risky = false) ∨ (lr.amount > 50,000 ∧
     lr.risky = true)
```

R4 (onB_1):

```
Objects: op:OperationStaff
Condition: op.loanRequest@post->size() ≤ 10
```

R6 (onA_1):

```
Objects: l:LoanRequestReport
Condition:  l.rate@post ≥ 6 ∧ l.loanrequest.amount ≤ 5000 ∧ (l.risk = ’low’ ∨
     l.risk = ’medium’)
```

3.2. Transforming the Models into BAUML

As stated earlier, our goal is to be able to verify UCON$_{ABC}$ policies in the context of a business process model annotated with data artifacts that support complex data structures. To achieve this, we will apply the verification techniques for artifact-centric business process models [15]. In order to do so, we need to adapt our starting models to the input required by the BAUML framework.

BAUML uses four different models: a UML class diagram, a UML state machine diagram, UML activity diagrams and OCL operation contracts. Therefore, we will need to translate or map the starting models (BPMN diagram, UML class diagram, OCL operation contracts and security policies) into these, to be able to reuse the existing techniques. Intuitively, there will be an (almost) direct mapping between the class diagrams and the OCL operation contracts in both approaches. However, we will need to translate the BPMN diagram into a state machine diagram and a set of activity diagrams to obtain an equivalent BAUML model. For this reason, we introduce state machine and activity diagrams.

Definition 1. *A state machine diagram is defined as $S_A = \langle V, v_o, v_f, E, X, T \rangle$, where V is a set of states, $v_o \in V$ is the initial state, $v_f \in V$ is the final state, E is a set of events, X is a set of effects, and $T \subseteq V \times OCL_{\mathcal{M}} \times E \times X \times V$ is a set of transitions between pairs of states, where $OCL_{\mathcal{M}}$ is an OCL condition over $\mathcal{M}$ that must be true in order to the transition to take place. Note that v_o cannot have any incoming transition, and v_f cannot have any outgoing transition.*

Definition 2. *$\mathcal{P}$ is a set of UML activity diagrams, such that for every state machine diagram S=$\langle V, v_o, v_f, E, X, T \rangle \in \mathcal{S}$, and for every event $\varepsilon \in$ EXTEVENTS(S) there exists exactly one activity diagram $P_\varepsilon \in \mathcal{P}$. P_ε is a tuple $\langle N, n_o, n_f, F \rangle$, where N is a set of nodes, $n_o \in N$ is the initial node, $n_f \subset N$ is the set of final nodes and F is a set of transitions between pairs of nodes.*

Obtaining an equivalent BAUML model The main challenge is to translate the BPMN diagram into a state machine diagram and a set of activity diagrams. This is not a trivial task since the former shows the interaction among the evolution of different classes in the class diagram, whereas in the BAUML modelling approach this interaction is implicitly represented using state machine diagrams. Other approaches also tackled this type of problem as in [14] where the authors propose the synthesising of object life cycles (state machines) from business process models.

Due to this complexity we deal here with a fragment of the BPMN diagram. In particular, we will translate only one of the pools, the *Customer* one. We focus on this pool because its tasks have a direct effect on the evolution of class *LoanRequest*, as shown on the contracts of activities or tasks *RequestALoan* and *ReceiveNotification*.

For this purpose, we will distinguish two types of tasks in the BPMN diagram:

- Tasks that send information or perform certain work by themselves. They can be identified by the dark envelope or by the lack of a symbol. We will refer to them as *action tasks*.
- Tasks that receive information and, as such, they are waiting for something to happen outside the scope of the pool. They can be identified by a white envelope symbol. We will call them *receive message tasks*. If these tasks do not have an incoming message flow, we will refer to them as *passive tasks*.

The first type of task will correspond to events in the state machine diagram, whereas the second type to states, but will require the incoming message flows to be considered in the translation process. Moreover, XOR-split nodes will also correspond to states. We will globally refer to XOR-split nodes, initial nodes and passive tasks as *passive nodes*.

Obtaining the State Machine Diagram. Algorithm 1 begins the translation process by obtaining a list of all the nodes in the BPMN diagram and translating them into the corresponding element in the state machine diagram. Note that the algorithm merely translates the nodes and not the connections between them.

Action tasks will correspond to events E. Passive tasks, XOR-split nodes and the final node correspond to states V. Initial nodes correspond to the initial pseudo-state v_o of the state machine diagram. XOR-merge nodes do not correspond to a specific node in the resulting state machine diagram. Finally, each incoming message flow will correspond to an event.

Algorithm 1 translateNodesAndMessages()

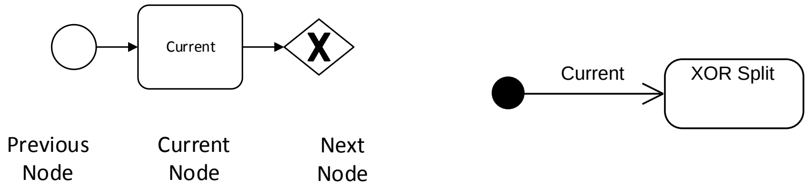
```
$nodeMap = \emptyset$
▷ We first create a map containing the task nodes in the BPMN diagram and their translation to an event or a state.
▷ $nodeList$ contains all the nodes in the BPMN diagram
for all $node \in nodeList$ do
    if $node$ is ActionTask then
        nodeMap.add(<node, new Event(node)>)
    else if $node$ is PassiveTask then
        nodeMap.add(<node, new State(node)>)
    else if $node$ is XOR-split then
        nodeMap.add(<node, new State(node)>)
    else if $node$ is InitialNode then
        nodeMap.add(<node, new InitialPseudostate(node)>)
    else if $node$ is FinalNode then
        nodeMap.add(<node, new State(node)>)
    end if
end for
▷ We then create an event for each incoming message flow in the pool
▷ $receiveMessageTaskList$ contains all the tasks with incoming message flows
▷ $messageMap$ will contain a map between the receive message task and the incoming message flows, translated to events
$messageMap = \emptyset$
for all $rm \in receiveMessageTaskList$ do
    incomingFlowList = rm.getIncomingMessageFlows()
    eventList = ∅
    for all $if \in incomingFlowList$ do
        sourceNode = if.getSource()
        event = new Event (sourceNode)
        eventList.add(event)
    end for
    messageMap.add(<rm, eventList>)
end for
```

Algorithm 2 initiates the processing of the nodes. It iterates over all the nodes and obtains the next nodes for the current node. Then it provides the current node, the next nodes and the node map (obtained by Algorithm 1) to Algorithm 3.

Algorithm 2 translateToSMD()

```
▷ $nodeList$ contains all the nodes in the BPMN diagram
▷ $nodeMap$ corresponds to the nodeMap obtained previously
for all $node \in nodeList$ do
    ▷ We go through the elements of the list in order (i.e. before the processing of a node all its previous nodes must have been processed)
    nextNodeList = node.getNextNodes()
    ▷ getNextNodes() ignores XOR-merge nodes and returns the targets of the XOR-merge
    if !$nextNodeList.isEmpty()$ then
        for all $nextNode \in nextNodeList$ do
            processNode(node, nextNode, nodeMap)
        end for
    end if
end for
```

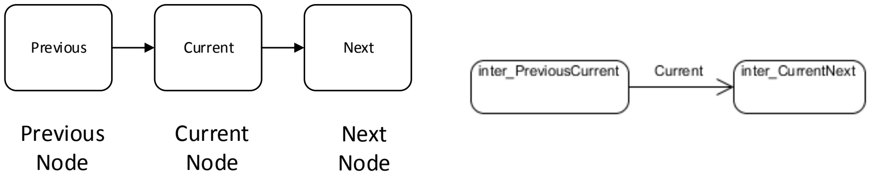
Algorithm 3 is executed for every node (and its next node) in the pool of interest in the initial BPMN diagram. As input, the algorithm receives the following: the current node ($node$), the next node ($nextNode$), and the node map ($nodeMap$), which contains the correspondence between the nodes in the BPMN diagram and the state machine diagram, previously created by Algorithm 1. The algorithm assumes that all the nodes that can be executed previous to the current node have already been translated and connected properly.



(a) BPMN fragment (b) Translation of fragment in (a)

Fig. 5. Translation of an action node surrounded by passive nodes

Then, it translates the connections between the current and the previous/next nodes according to their types. If the current node is an action task, it will correspond to an event in the state machine diagram. Hence, we will have to create a transition with the event, which will require a source and a target state. These source and target states will correspond to other BPMN nodes, if the surrounding nodes are passive nodes or a message receive task (see Figure 5). In contrast, if the surrounding nodes are action tasks, they will require an auxiliary state (see Figure 6).

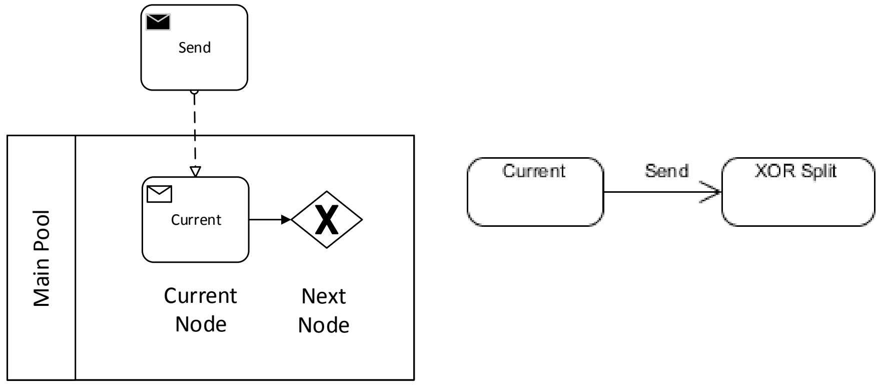


(a) BPMN fragment (b) Translation of fragment in (a)

Fig. 6. Translation of an action node surrounded by other action nodes

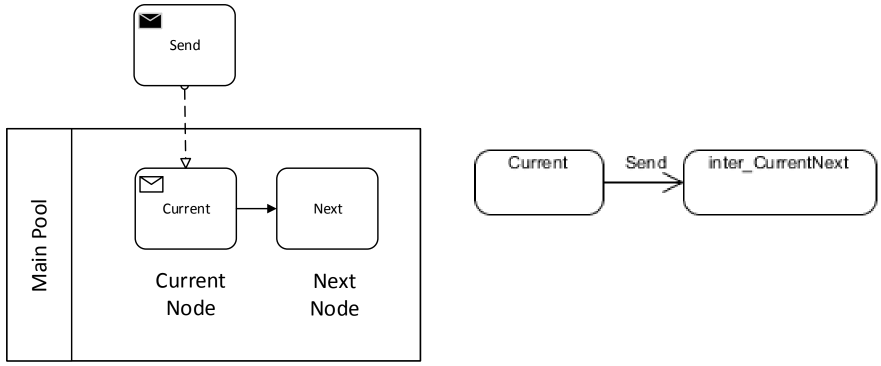
If the current node is a receive message task, the semantics of BPMN state that it is not possible to move to the next node until a message is received. Therefore, for each incoming message flow in the node, the state machine diagram will require a transition with the message represented as an event to move to the next state. Considering this, the current node corresponds to a state acting as the source state of the transition. Then we need to consider the next node. If the next node is another receive message task or a passive node (e.g. XOR-split), then the target state will be the corresponding state in the state machine diagram (see Figure 7). Otherwise, if the next node is an action task , an intermediate state will need to be created to act as the target state (Figure 8).

If the current and next node are passive nodes, the only thing that needs to be done is to create a transition between both (see Figure 9). This transition will be automatic as there will be no events between both. Note that we do not consider the case of the next



(a) BPMN fragment (b) Translation of fragment in (a)

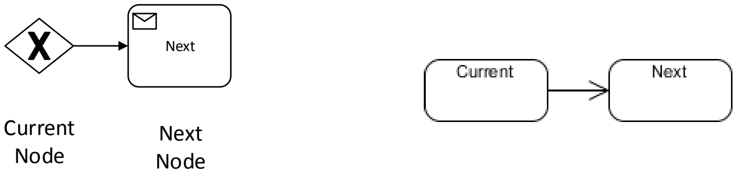
Fig. 7. Translation of a receive message task followed by a passive node



(a) BPMN fragment (b) Translation of fragment in (a)

Fig. 8. Translation of a receive message task followed by an action

node being an action or a receive message task, as in this case the necessary changes will be made by the algorithm in the next iteration.



(a) BPMN fragment (b) Translation of fragment in (a)

Fig. 9. Translation of a passive node followed by another passive node

Algorithm 3 processNode(node, nextNode, nodeMap)

```
if $node$ is ActionTask then
    Event e = nodeMap.get(node)
    $prevNodeList = node.getPrevious()$
    for all $prevNode \in prevNodeList$ do
        if $prevNode$ is PassiveNode then
            State source = nodeMap.get(prevNode)
            if $nextNode$ is PassiveNode $\vee$ $nextNode$ is ReceiveMessageNode then
                State target = nodeMap.get(nextNode)
                Transition t = new Transition (source, target, e)
            else if $nextNode$ is ActionTask then
                State target = new State (“inter_” + $node.getName()$ + $nextNode.getName()$)
                Transition t = new Transition(source, target, e)
            end if
        else if $prevNode$ is ActionTask then
            if $nextNode$ is PassiveNode $\vee$ $nextNode$ is ReceiveMessageNode then
                $\triangleright$ We obtain the target state of the previous ActionTask, which will be the source state for our new transition
                State source
                State target = nodeMap.get(nextNode)
                Transition t = new Transition (source, target, e)
            else if $nextNode$ is ActionTask then
                $\triangleright$ We obtain the target state of the previous ActionTask, which will be the source state for our new transition
                State source
                State target = new State (“inter_” + $node.getName()$ + $nextNode.getName()$)
                Transition t = new Transition (source,target, e)
            end if
        else if $prevNode$ is ReceiveMessageNode then
            State source = get intermediate state generated by Rec. Message Event
            if $nextNode$ is PassiveNode $\vee$ $nextNode$ is ReceiveMessageTask then
                State target = nodeMap.get(nextNode)
                Transition t = new Transition (source, target, e)
            else if $nextNode$ is ActionTask then
                State target = new State (“inter_” + $node.getName()$ + $nextNode.getName()$)
                Transition t = new Transition (source, target, e)
            end if
        end if
    end for
else if $node$ is ReceiveMessageTask then
    List<Event> eventList = messageMap.getIncomingMessageFlow($node$)
    State source = nodeMap.get(node)
    if $nextNode$ is ActionTask then
        State target = new State (“inter_” + $node.getName()$ + $nextNode.getName()$)
        for all e $\in$ eventList do
            Transition t = new Transition (source, target, e)
        end for
    else if $nextNode$ is PassiveNode $\vee$ $nextNode$ is ReceiveMessageTask then
        State target = nodeMap.get(nextNode)
        for all e $\in$ eventList do
            Transition t = new Transition (source, target, e)
        end for
    end if
else if $node$ is PassiveNode $\wedge$ $nextNode$ is PassiveNode then
    State source = nodeMap.get(node)
    State target = nodeMap.get(nextNode)
    Transition t = new Transition (source, target)
    $\triangleright$ If $nextNode$ is ActionTask, we do nothing because it will be processed in the next iteration.
end if
```

Figure 10 shows the resulting translation for the customer pool in the BPMN diagram. Note that the action task *RequestALoan* corresponds to an event. The receive message task *ReceiveNotification* corresponds to a state. The incoming message flow of *ReceiveNotification* connected action tasks *SendApprovedNotification* and *SendNegativeNotification* from another pool to *ReceiveNotification*. Therefore, these action tasks, *SendApprovedNotification* and *SendNegativeNotification*, has been translated to an event in the state machine diagram. Finally, state *FinalState* corresponds to the final state in the BPMN diagram. We have given it this name for easier readability.

The BAUML framework requires the state machine diagram to be linked to a class, called the *artifact*. In this case, it will be linked to class *LoanRequest*, since the changes made by the activities or tasks have an impact on it.

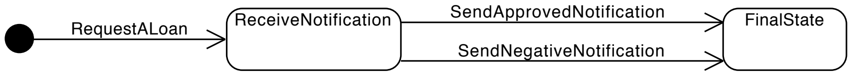


Fig. 10. State machine diagram

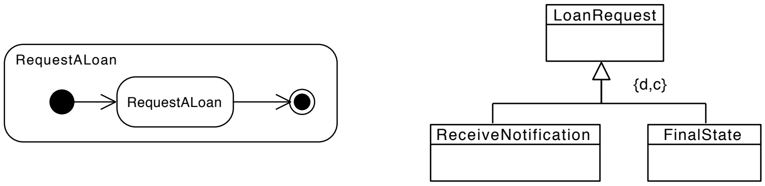


Fig. 11. Activity diagram and updated class diagram

Obtaining the Activity Diagrams The BAUML framework requires an activity diagram for each event in the state machine diagram. They can be automatically generated for each event obtained by the previous algorithms. Therefore, for each event we will have the corresponding activity diagram: it will have an initial node, a final node, and one task with the corresponding OCL operation contract. Figure 11 shows the activity diagram for event *RequestALoan*.

Once the state machine and activity diagrams from the BPMN diagram have been obtained, we already have all the components necessary to reason using the techniques in [15]. What needs to be done is to make some minor changes to the class diagram and the operation contracts before incorporating the security policies into the model.

Updating the Class Diagram and the Operation Contracts The last step in the process is to update the class diagram and the OCL operation contracts. In the BAUML framework, the state machine diagram shows the evolution of an artifact (i.e. a class in the class diagram), and each state corresponds to a subclass of the artifact. Therefore, once the state machine diagram and the activity diagrams have been obtained, the class diagram needs to be updated by including as many classes as states there are in the state machine diagram. In our particular example, class *LoanRequest* should be updated with subclasses *ReceiveNotification* and *FinalState*, forming a *disjoint*, *complete* hierarchy (see Figure 11).

Then, it is also necessary to update the OCL operation contracts to ensure the proper evolution of the class. Therefore, the postcondition of the contract of task *RequestALoan* must be updated to ensure that the *LoanRequest* has the type *ReceiveNotification*:

```
LoanRequest.allInstances()->exists(l | l.oclIsNew() and l.id = pId and l.amount =
     am and l.date = t and l.accepted = false and l.pending = true and
     l.creditProvider = cp and l.customer = c and l.risky=r and
     l.oclIsTypeOf(ReceiveNotification))
```

Similarly, *SendApprovedNotification* and *SendNegativeNotification* should be updated to change the state of *LoanRequest* to *FinalState* and to ensure it is no longer of the previous subtype (*ReceiveNotification*). Below we show the updated postcondition for *SendApprovedNotification*:

```
l.accepted = acc and l.pending = false and l.oclIsTypeOf(Finalstate) and not
     l.oclIsTypeOf(ReceiveNotification)
```

These updates can be performed automatically, as the changes can be inferred from the state machine diagram obtained previously. More specifically, considering that each task is part of a transition in the state machine diagram, the postcondition should ensure that the class has the subtype represented by a target state, and that it no longer is of the subtype of the source state.

4. Reasoning on Security Policies

Once the security policies have been defined by using OCL and the starting models translated to BAUML, we need to merge both with the goal of checking them as a whole. First, it is necessary to present how to add the policies to the models and later on to explain how reasoning with them.

4.1. Adding the Security Policies to BAUML

Intuitively, if the conditions established by a security policy are not met, the task to which they are linked should never take place. Considering that our modelling approach uses pre and postconditions for each task, this means that security policies should be added to the preconditions, to ensure that the task does not execute if they are not met, with the following considerations:

1. The tasks that these policies are attached to, should have as input the objects of the policy.
2. If this is not the case, these objects should be created or specialised by the task. This is easy to identify by looking for either `obj.oclIsNew()` (creation) or `obj.oclIsTypeOf(ObjType)` (specialization) in the postcondition.

3. If the OCL expression corresponding to the security policy contains `@post`, it needs to be modified before it can be added to the precondition. `@post` refers to the value of the attribute in postcondition time, which cannot be accessed in precondition time. To deal with this, we need to look for the parameter that will assign a new value to this attribute, and substitute it in the expression.

Note that if the task is a *receive message task* in the BPMN, the predicates will be added to the tasks that result from the translation of the incoming message flows to events.

Returning to our example, there are two security predicates that should be checked in the precondition of *RequestALoan*, as indicated by the BPMN diagram in Figure 3. We will first look at `R1`, whose OCL corresponds to the security policy as defined earlier: `c.loanRequest->forAll(l | l.pending=false and l.accepted=true)`. In this case, this expression can be incorporated directly into the precondition of the task. Since the object of the policy is *Customer c* and *RequestALoan* has a customer as input parameter (also `c` in our example), we only need to make sure that the policy refers to this customer, by rewriting its name if necessary.

In the case of `R2`, its object refers to *LoanRequest*. The operation contract does not have a *LoanRequest* as input; however, it does create the *LoanRequest* in its postcondition (`LoanRequest.allInstances()->exists(l | l.oclIsNew()...)`). In this case, we have to replace all references to `lr.amount` and `lr.risky` in the policy with the corresponding parameters which will be accessible at precondition time. Since `l.amount = am` and `l.risky = r`, as stated in the postcondition, we have to replace `lr.amount` with `am` and `lr.risky` with `r`, resulting in `(am<=50000 and r=false) or (am>50000 and r=true)`, as shown below.

```
RequestALoan(pId: String, am: int, c: Customer, t: Date, cp: CreditProvider,
     r:boolean)
pre: c.loanRequest->forAll(l | l.pending=false and l.accepted=true) and ((am<=50000
     and r=false) or (am>50000 and r=true))
post: LoanRequest.allInstances()->exists(l | l.oclIsNew() and l.id = pId and
     l.amount = am and l.date = t and l.accepted = false and l.pending = true and
     l.creditProvider = cp and l.customer = c and l.risky = r)
```

4.2. Reasoning over Security Policies

Once the security policies have been added to the BAUML model it is possible to run verification and validation tests as described in [15]. Given a BAUML model, this approach automatically translates it into the required logic for satisfiability checking, and then determines whether the model fulfils certain semantic correctness properties. The underlying satisfiability checker can deal with negated predicates and, in the case of unsatisfiability, provides the list of constraints that prevent it.

Internally, this is done by SVTe [16], a tool that uses the CQC_E method [47], and which has also been used successfully in [44, 46]. It is aimed at building a consistent state of a database schema that satisfies a certain goal. Starting from an empty solution (i.e. an empty database), and given a goal, the database schema, a set of constraints and derivation rules, it tries to obtain a set of base facts that satisfy the goal without violating the constraints. Note that for BAUML models, the derivation rules include the translation of the tasks, activity diagrams and state machine diagram. The CQC_E method is a semidecision procedure for finite satisfiability. That is, if the solution contains infinite elements, it does not terminate. However, termination is ensured if the model fulfils a set of properties, explained in [11]. They can be summarized as follows: 1) the cardinalities of the associations, 2) the number of classes should be bounded, 3) the OCL expressions should be unidirectional and navigational, i.e. when dealing with class instances that are modified by the model, they only refer to elements connected to the starting element. During the inference process, CQC_E uses Variable Instantiation Patterns (VIPs), which generate only the facts that are needed to achieve the goal. If there is no instance found, then VIPs guarantee that the goal cannot be achieved. Considering this, SVTe provides two different types of result, depending on the outcome of the reasoning process. On the one hand, if it finds a solution that fulfils the goal, it provides a sample instantiation. On the other hand, if there is no solution, it shows the constraints that prevent the goal's achievement.

As we mentioned earlier, it is possible to carry out both validation and verification tests. Verification tests looks for inherent errors in the model, answering the question "Is the model right?", and validation tests ensure that the model represents the domain appropriately (i.e. it fulfils the requirements), answering the question "Is it the right model?".

Given the BAUML model, verification tests can be generated and performed automatically, as shown by the prototype tool in [15]. Some examples of these are: ensuring the liveliness or a class or an association (i.e. ensuring that instances can be created), looking for redundancies in the integrity constraints or ensuring that tasks in the process model can execute. On the other hand, validation tests require manual definition, although they can be run automatically. For instance: *can loan requests for 60,000 be made without accepting the risky clause?* or *can a customer make a new loan request when some of his other loan requests have been denied?*.

$$\begin{aligned}
loanReq() &\leftarrow LoanRequest(oid, id, am, ac, pend, d, t, r) \land am = 60,000 \land r = false\\
newLoanReqWhenDenied() &\leftarrow LoanRequest(oid, id, am, ac, pend, d, t1)\\
&\land LoanRequest(oid2, id2, am2, ac2, pend2, d2, t2) \land pend2 = false\\
&\land ac2 = false \land oid \neq oid2 \land t1 = t2
\end{aligned}$$

These tests have the form of logic derivation rules. Each test has a head: if it is possible to generate the head of the rule, then the test is satisfiable. The body contains the representation of the elements in the model and the conditions which they should satisfy for the test.

Returning to the validation tests above, *LoanRequest* corresponds to an instance of class *LoanRequest*, and $am = 60,000 \land r = false$ state the value that these variables am, r should have. The second example states that, in order for the test to execute successfully, there must be two different *LoanRequests* which coexist simultaneously ($t1 = t2$), one of which has been denied.

If we run these tests, the first one will result in unsatisfiability, whereas the second test will execute successfully. In the first case, this is due to security policy `R2`. In the second case, although there is policy `R1` to prevent the creation of new *LoanRequests* when previous ones have been denied, the policy does not consider the case in which several pending loan requests from the customer may coexist simultaneously. Eventually, one of these loan requests may be denied, but there will be other pending loan requests which will require evaluation and which may be accepted.

5. Related Work

Compliance of business processes with security policies at the design and runtime stages has been considered in several stages of business process management [26]. The extension of BPMN with annotations related to security requirements is not new [45]. A vast number of works provide several ways to represent and verify security requirements at the design stage, such as [48, 57, 49]. Salnitri et al. [49] establishes mechanisms to represent security policies in BPMN by means of the extension SecBPMN. The authors also provide a language to verify if the business process model complies with the security policies established. [53] proposes a risk analysis of the business process models combining the partial risks of the activities that conform them. However, our approach focuses on the verification of the security policy in artifact-centric business process models to support the contexts where there various involved data and with different cardinalities between them involved in the policy rules. [3] detected the difficulty to combine data and business processes, but different data objects with various relations between them were not included.

Access control models and process-centric approaches. There exist several and diverse access control models, a good taxonomy is provided in [29]. The most prominent traditional access control models encompassed Mandatory Access Control (MAC), Discretionary Access Control (DAC), Role-based Access Control (RBAC), and; Attribute-based Access Control (ABAC). These access control models are based on the restriction on performing certain right. Depending on the properties used to evaluate the conditions to apply or not the restriction define the access control model. For instance, MAC and DAC are based on the ownership property but RBAC is based on the role property. There exist also some access control models that are emerged as extension for workflow systems such as TBAC [58][57], T-RBAC [35], and W-RBAC [55]. These access control models just consist of adapting RBAC to the process-centric systems. However, all these access control models focus on the request and grant the access only at request time skipping other usage control aspects such as the number of times of using resources/objects or the time of using. In conclusion, these access control models fail to integrate sophisticated decision-making as UCON does.

Currently, **monitoring and process mining techniques** are new trends in order to detect whether certain security requirements are fulfilled by analyzing event logs [1, 6]. The work in [51] enables the generation of security configuration workflows, whereas [54] provides a framework to design product lines to verify security policies in accordance to a set of available configurations. Nevertheless, these works only consider the activity-centric perspective and overlook the artifact-centric perspective in most cases. In terms of complying with security policies on activity-centric process models, different formalisms are proposed to define security compliance rules, such a logic approach based on LTL or declarative approach based on Answer Set [23]. The best approach for large-scale security policies verification depends on time-consuming and easy to understand/update. Not only security policies are important, the context-awareness of the business process should also be considered (e.g. one of the process's instances is executed in a platform and another instance is executed in a different platform). For this reason, a recent work proposes a language for the specification and design of context-aware and secure workflow systems [59]. The approaches to enforce security policies at runtime mainly focus on integrating security control mechanisms into business process management systems [9].

Compliance and security policy verification in artifact-centric business processes has been addressed in a previous work [28]. The authors extend the artifact-centric framework by including the modelling of compliance rules, and obtain a model that complies by design. Also, the paper [31] checks for conformance between process models and data objects at design time. Weak conformance is used to verify that the correct execution of a process model corresponds to a correct evolution of states of the data objects. Reachability and weak-termination are verified in artifact-centric models combining structural and data information [8].

Some previous works indicate the difficulty of security compliance using artifact-centric models since there exists no well-defined operational semantics for directly executing the defined models. But the work of [48] has been proposed to support process-aware secure systems modelling and automated generation of secure artifact-centric implementations.

In [50], an extension of artifact-centric process models based on the Usage Control Model introduces mechanisms to specify security policies and verify their correctness. However, it focuses on reachability and weak-termination of the model as a whole.

Verification and validation in artifact-centric business process models is also a related area of research. There are several works which focus on reasoning on Data-Centric Dynamic Systems (DCDSs) [21, 22], grounded on logic. However, DCDSs use condition-action rules and actions defined in logic. In contrast to our approach, these models are not as intuitive, and using condition-action rules changes the representation paradigm, in the sense that they do not force the execution of actions in a certain order. [30] encodes actions in the same way as DCDSs, but it uses a relational database for the data and Petri nets to establish the execution order. In terms of reasoning, the approach mainly targets reachability and model checking of properties defined in first-order logic. Decidability is achieved by state-boundedness.

Other approaches, such as [7, 4, 13], define the models and the properties to be checked in languages derived from logic. As a result, the models under consideration are formal, but they are not intuitive nor practical from the point of view of the business. Similarly, the properties to be checked have to be defined manually.

A more business-friendly representation for artifact-centric business processes can be found on the Guard-Stage-Milestone (GSM) approach. GSM models show the stages in the evolution of an artifact and the guard conditions which have to be true to enter a certain state. However, they also have the concept of milestone: a condition that, once it is achieved, it closes a state [24, 19, 20]. [24] simulates the behaviour of the system given certain data. [19] is able to reason on the models but data types are limited and it only allows one instance per artifact. Finally, [20] performs model checking from a multi-agent perspective, but the number of objects is bounded which may lead to unreliable results when the bound is exceeded.

The approach in [56] uses BPMN diagrams whose tasks may be annotated with pre-conditions and effects defined in logic, and use an optional ontology to define the underlying data. They have a prototype tool that can perform some tests. Since both the ontology and the details of the effects are not compulsory, the final results can only be partial.

Note also that none of these works take security policies into consideration as we do here.

6. Conclusions and Further Work

We have proposed in this paper a combined business process model that supports the definition, verification and reasoning of security policies involving different kinds of data objects. The enriched model consists of a BPMN model, a UML class diagram, OCL operation contracts for the BPMN activities, and UCON$_{ABC}$ security policies defined in OCL. All these components are then automatically translated into a BAUML model which supports a set of verification and reasoning techniques. Thanks to our proposal, organisations are able to describe their security policies into artifact-centric approaches for business process modelling, providing a mechanism for verifying and validate the model's correctness. The model provides an easier manner to include the security rules into business processes and allows us to ensure that they are compatible with the business requirements and goals.

Although our proposal meets the objectives stated in the introduction, also presents some limitations: 1) regarding UCON support, our approach demonstrates the use of only certain types of predicates and update predicates; 2) regarding the reasoning, it is only limited to the satisfactibility or not of a policy from a pool-by-pool perspective, we can go an step forward by considering several pools, and 3) regarding the tools, our approach depends on the use of different tools separately, and it would be interesting to integrate them. Assuming these limitations, as future work we plan to extend the proposal by including new use cases to fully test the whole set of security policies supported by UCON, and to improve the reasoning by considering various pools simultaneously.

Acknowledgments. This work has been supported by Project PID2020-112540RB-C44 funded by MCIN/AEI/ 10.13039/501100011033, Project TIN2017-87610-R funded by MCIN/AEI/10.13039/501100011033 and FEDER "Una manera de hacer Europa", Project 2017-SGR-1749 by the Generalitat de Catalunya, Projects COPERNICA (P20_01224) and METAMORFOSIS by the Junta de Andalucía.

References

1. Accorsi, R., Wonnemann, C., Stocker, T.: Towards Forensic Data Flow Analysis of Business Process Logs. In: 2011 Sixth International Conference on IT Security Incident Management and IT Forensics. IEEE (may 2011)
2. Ahmed, N., Matulevicius, R.: Securing business processes using security risk-oriented patterns. Computer Standards & Interfaces 36(4), 723–733 (2014), https://doi.org/10.1016/j.csi.2013.12.007
3. Alizadeh, M., Lu, X., Fahland, D., Zannone, N., van der Aalst, W.M.P.: Linking data and process perspectives for conformance analysis. Computers & Security 73, 172–193 (2018)
4. Belardinelli, F., Lomuscio, A., Patrizi, F.: Verification of deployed artifact systems via data abstraction. In: Kappel, G., Maamar, Z., Nezhad, H.R.M. (eds.) ICSOC 2011. LNCS, vol. 7084, pp. 142–156. Springer (2011)
5. Bentounsi, M., Benbernou, S., Atallah, M.J.: Security-aware business process as a service by hiding provenance. Computer Standards & Interfaces 44, 220–233 (2016), https://doi.org/10.1016/j.csi.2015.08.011
6. Bezerra, F., Wainer, J., van der Aalst, W.M.P.: Anomaly detection using process mining. In: Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Ukor, R. (eds.) Enterprise, Business-Process and Information Systems Modeling. pp. 149–161. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)

7. Bhattacharya, K., Gerede, C.E., Hull, R., Liu, R., Su, J.: Towards formal analysis of artifact-centric business process models. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 288–304. Springer (2007)
8. Borrego, D., Gasca, R.M., Gómez-López, M.T.: Automating correctness verification of artifact-centric business process models. Information & Software Technology 62, 187–197 (2015)
9. Brucker, A.D., Hang, I., Lückemeyer, G., Ruparel, R.: SecureBPMN: modeling and enforcing access control requirements in business processes. In: Atluri, V., Vaidya, J., Kern, A., Kantarcioglu, M. (eds.) 17th ACM Symposium on Access Control Models and Technologies, SACMAT '12, Newark, NJ, USA - June 20 - 22, 2012. pp. 123–126. ACM (2012), https://doi.org/10.1145/2295136.2295160
10. Cabot, J., Gogolla, M.: Object constraint language (ocl): A definitive guide. In: Bernardo, M., Cortellessa, V., Pierantonio, A. (eds.) Formal Methods for Model-Driven Engineering: 12th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM 2012, Bertinoro, Italy, June 18-23, 2012. Advanced Lectures. pp. 58–90. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
11. Calvanese, D., Montali, M., Estañol, M., Teniente, E.: Verifiable UML artifact-centric business process models. In: Li, J., Wang, X.S., Garofalakis, M.N., Soboroff, I., Suel, T., Wang, M. (eds.) CIKM 2014. pp. 1289–1298. ACM (2014)
12. De Giacomo, G., Oriol, X., Estañol, M., Teniente, E.: Linking data and BPMN processes to achieve executable models. In: Dubois, E., Pohl, K. (eds.) CAiSE 2017. LNCS, vol. 10253, pp. 612–628. Springer (2017)
13. Deutsch, A., Hull, R., Li, Y., Vianu, V.: Automatic verification of database-centric systems. ACM SIGLOG News 5(2), 37–56 (2018)
14. Eshuis, R., Van Gorp, P.: Synthesizing object life cycles from business process models. Software & Systems Modeling 15(1), 281–302 (Feb 2016)
15. Estañol, M., Sancho, M., Teniente, E.: Ensuring the semantic correctness of a BAUML artifact-centric BPM. Information & Software Technology 93, 147–162 (2018)
16. Farré, C., Rull, G., Teniente, E., Urpí, T.: Svte: a tool to validate database schemas giving explanations. In: Giakoumakis, L., Kossmann, D. (eds.) DBTest 2008. p. 9. ACM (2008)
17. Gómez-López, M.T., Pérez-Álvarez, J.M., Gasca, R.M.: Compliance validation and diagnosis of business data constraints in business processes at runtime. Information Systems 48, 26 – 43 (2015), http://www.sciencedirect.com/science/article/pii/S0306437914001306
18. Gómez-López, M.T., Pérez-Álvarez, J.M., Varela-Vaca, Á.J., Gasca, R.M.: Guiding the creation of choreographed processes with multiple instances based on data models. In: BPM 2016 International Workshops, Revised Papers. pp. 239–251 (2016)
19. Gonzalez, P., Griesmayer, A., Lomuscio, A.: Verifying gsm-based business artifacts. In: Goble, C.A., Chen, P.P., Zhang, J. (eds.) 2012 IEEE 19th International Conference on Web Services, Honolulu, HI, USA, June 24-29, 2012. pp. 25–32. IEEE Computer Society (2012)
20. Gonzalez, P., Griesmayer, A., Lomuscio, A.: Model checking gsm-based multi-agent systems. In: Lomuscio, A., Nepal, S., Patrizi, F., Benatallah, B., Brandic, I. (eds.) ICSOC 2013 Workshops. LNCS, vol. 8377, pp. 54–68. Springer (2013)
21. Hariri, B.B., Calvanese, D., Giacomo, G.D., Deutsch, A., Montali, M.: Verification of relational data-centric dynamic systems with external services. In: Hull, R., Fan, W. (eds.) PODS 2013. pp. 163–174. ACM (2013)
22. Hariri, B.B., Calvanese, D., Giacomo, G.D., Masellis, R.D., Felli, P., Montali, M.: Verification of description logic knowledge and action bases. In: Raedt, L.D., Bessiere, C., Dubois, D., Doherty, P., Frasconi, P., Heintz, F., Lucas, P.J.F. (eds.) ECAI 2012. Frontiers in Artificial Intelligence and Applications, vol. 242, pp. 103–108. IOS Press (2012)
23. Hewett, R., Kijsanayothin, P., Bak, S., Galbrei, M.: Cybersecurity policy verification with declarative programming. Applied Intelligence 45, 83 – 95 (2016)

24. III, F.F.T.H., Boaz, D., Gupta, M., Vaculín, R., Sun, Y., Hull, R., Limonad, L.: Barcelona: A design and runtime environment for declarative artifact-centric BPM. In: Basu, S., Pautasso, C., Zhang, L., Fu, X. (eds.) ICSOC 2013. LNCS, vol. 8274, pp. 705–709. Springer (2013)
25. Kocbek, M., Jost, G., Hericko, M., Polancic, G.: Business process model and notation: The current state of affairs. Comput. Sci. Inf. Syst. 12(2), 509–539 (2015), https://doi.org/10.2298/CSIS140610006K
26. Leitner, M., Rinderle-Ma, S.: A systematic review on security in Process-Aware Information Systems – Constitution challenges, and future directions. Information and Software Technology 56(3), 273–293 (mar 2014)
27. Li, M., Wang, H.: Specifying usage control model with object constraint language. In: 2010 Fourth International Conference on Network and System Security. pp. 391–397 (Sept 2010)
28. Lohman, N.: Compliance by design for artifact-centric business processes. In: BPM 2011 LNCS vol 6896 Springer. p. 99–115 (2011)
29. Majumder, A., Namasudra, S., Nath, S.: Taxonomy and classification of access control models for cloud environments, chap. 2, pp. 23–53. Springer London, London (2014)
30. Masellis, R.D., Francescomarino, C.D., Ghidini, C., Montali, M., Tessaris, S.: Add data into business process verification: Bridging the gap between theory and practice. In: Singh, S.P., Markovitch, S. (eds.) Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA. pp. 1091–1099. AAAI Press (2017), http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14627
31. Meyer, A., Pufahl, L., Batoulis, K., Fahland, D., Weske, M.: Automating data exchange in process choreographies. Inf. Syst. 53, 296–329 (2015)
32. Mpardis, G., Kotsilieris, T.: Bank loan processes modelling using bpmn. In: 2010 Developments in E-systems Engineering. pp. 239–242 (Sept 2010)
33. Müller, G., Accorsi, R.: Why are business processes not secure? In: Fischlin, M., Katzenbeisser, S. (eds.) Number Theory and Cryptography - Papers in Honor of Johannes Buchmann on the Occasion of His 60th Birthday. Lecture Notes in Computer Science, vol. 8260, pp. 240–254. Springer (2013), https://doi.org/10.1007/978-3-642-42001-6_17
34. Neubauer, T., Klemen, M.D., Biffl, S.: Secure business process management: A roadmap. In: Proceedings of the The First International Conference on Availability, Reliability and Security, ARES 2006, The International Dependability Conference - Bridging Theory and Practice, April 20-22 2006, Vienna University of Technology, Austria. pp. 457–464. IEEE Computer Society (2006), https://doi.org/10.1109/ARES.2006.121
35. Oh, S., Park, S.: Task-role based access control (T-RBAC): an improved access control model for enterprise environment. In: Ibrahim, M.T., Küng, J., Revell, N. (eds.) Database and Expert Systems Applications, 11th International Conference, DEXA 2000, London, UK, September 4-8, 2000, Proceedings. Lecture Notes in Computer Science, vol. 1873, pp. 264–273. Springer (2000), https://doi.org/10.1007/3-540-44469-6_25
36. OMG: Object Management Group, Business Process Model and Notation (BPMN) Version 2.0. OMG Standard (2011)
37. OMG: Object Management Group, Unified Modeling Language (UML) Version 2.5.1. OMG Standard (2017)
38. Oriol, X., De Giacomo, G., Estañol, M., Teniente, E.: Embedding reactive behaviour into artifact-centric business process models. Future Generation of Computer Systems p. Accepted for publication (2021)
39. Park, J., Sandhu, R.: The UCON ABC usage control model. ACM Transactions on Information and System Security 7(1), 128–174 (feb 2004)
40. Pérez-Álvarez, J.M., Gómez-López, M.T., Eshuis, R., Montali, M., Gasca, R.M.: Verifying the manipulation of data objects according to business process and data models. Knowledge and Information Systems (Jan 2020), https://doi.org/10.1007/s10115-019-01431-5

41. Poels, Geert and García, Félix and Ruiz, Francisco and Piattini, Mario: Architecting business process maps. COMPUTER SCIENCE AND INFORMATION SYSTEMS 17(1), 117–139 (2020), http://dx.doi.org/10.2298/csis181118018p
42. Pozo, S., Varela-Vaca, Á.J., Gasca, R.M.: Mda-based framework for automatic generation of consistent firewall acls with NAT. In: Computational Science and Its Applications - ICCSA 2009, International Conference, Seoul, Korea, June 29-July 2, 2009, Proceedings, Part II. pp. 130–144 (2009)
43. Pérez-Álvarez, J.M., Parody, L.P., Gómez-López, M.T., Gasca, R.M., Ceravolo, P.: Decision-making support for input data in business processes according to former instances. Comput. Sci. Inf. Syst. 18(3), 597–618 (2021), https://doi.org/10.2298/CSIS200522051P
44. Queralt, A., Teniente, E.: Verification and validation of UML conceptual schemas with OCL constraints. ACM Trans. Softw. Eng. Methodol. 21(2), 13:1–13:41 (2012)
45. Rodríguez, A., Fernández-Medina, E., Trujillo, J., Piattini, M.: Secure business process model specification through a UML 2.0 activity diagram profile. Decision Support Systems 51(3), 446–465 (2011), http://dx.doi.org/10.1016/j.dss.2011.01.018
46. Rull, G., Farré, C., Queralt, A., Teniente, E., Urpí, T.: Aurus: explaining the validation of UML/OCL conceptual schemas. Softw. Syst. Model. 14(2), 953–980 (2015)
47. Rull, G., Farré, C., Teniente, E., Urpí, T.: Providing explanations for database schema validation. In: Bhowmick, S.S., Küng, J., Wagner, R.R. (eds.) DEXA 2008. LNCS, vol. 5181, pp. 660–667. Springer (2008)
48. Salnitri, M., Brucker, A.D., Giorgini, P.: From Secure Business Process Models to Secure Artifact-Centric Specifications. In: Enterprise Business-Process and Information Systems Modeling, pp. 246–262. Springer Science + Business Media (2015)
49. Salnitri, M., Dalpiaz, F., Giorgini, P.: Designing secure business processes with secbpmn. Software and System Modeling 16(3), 737–757 (2017)
50. Varela-Vaca, Á.J., Borrego, D., Gómez-López, M.T., Gasca, R.M.: A usage control model extension for the verification of security policies in artifact-centric business process models. In: BIS 2016. pp. 289–301 (2016)
51. Varela-Vaca, A.J., Galindo, J.A., Ramos-Gutiérrez, B., Gómez-López, M.T., Benavides, D.: Process Mining to Unleash Variability Management:Discovering Configuration Workflows Using Logs. In: Proceeedings of the 23nd International Systems and Software Product Line Conference, SPLC 2019, Paris, France, September 9-13, 2019. pp. – (2019), https://doi.org/10.1145/3336294.3336303
52. Varela-Vaca, Á.J., Gómez-López, M.T.: Access control security policies DSL for BPMN. http://www.idea.us.es/securitydsl/ (2020)
53. Varela-Vaca, A.J., Parody, L., Gasca, R.M., López, M.T.G.: Automatic verification and diagnosis of security risk assessments in business process models. IEEE Access 7, 26448–26465 (2019), https://doi.org/10.1109/ACCESS.2019.2901408
54. Varela-Vaca, A.J., Gasca, R.M., Ceballos, R., Gómez-López, M.T., Bernáldez Torres, P.: CyberSPL: A framework for the verification of cybersecurity policy compliance of system configurations using software product lines. Applied Sciences 9(24) (2019), https://www.mdpi.com/2076-3417/9/24/5364
55. Wainer, J., Barthelmess, P., Kumar, A.: W-RBAC — a workflow security model incorporating controlled overriding of constraints. International Journal of Cooperative Information Systems 12(04), 455–485 (Dec 2003), https://doi.org/10.1142/s0218843003000814
56. Weber, I., Hoffmann, J., Mendling, J.: Beyond soundness: on the verification of semantic business process models. Distributed Parallel Databases 27(3), 271–343 (2010)
57. Wolter, C., Menzel, M., Schaad, A., Miseldine, P., Meinel, C.: Model-driven business process security requirement specification. Journal of Systems Architecture 55(4), 211–223 (apr 2009)
58. Wolter, C., Schaad, A.: Modeling of task-based authorization constraints in BPMN. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) Business Process Management, 5th International Conference, BPM 2007, Brisbane, Australia, September 24-28, 2007, Proceedings. Lecture Notes in

Computer Science, vol. 4714, pp. 64–79. Springer (2007), https://doi.org/10.1007/978-3-540-75183-0_5

59. Zedan, H., Al-Sultan, S.: The Specification and Design of Secure Context-Aware Workflows . Expert Systems With Applications 86, 367–384 (2017)
60. Zoet, M., Versendaal, J., Ravesteyn, P.: A business rules viewpoint on risk and compliance management. In: 24th Bled eConference: eFuture Creating Solutions for the Individual, Organisations and Society, Bled, Slovenia, June 12-15, 2011. p. 25 (2011), http://aisel.aisnet.org/bled2011/25

Montserrat Estañol obtained her Ph.D. in 2016, at Universitat Politècnica de Catalunya (UPC), where she currently teaches an undergraduate course on Software Engineering. She has worked as a postdoc researcher at InLab FIB, UPC, and at Barcelona Supercomputing Center (BSC). Her research interests include conceptual modeling and ontologies, artifact-centric business process modeling and automated reasoning on both conceptual schemas and artifact-centric business process models.

Ángel Jesús Varela-Vaca received an MSc in Software Engineering and Technology (2009) and obtained his PhD with honours at the University of Seville (2013). He is currently working as Associate Professor at the Universidad Sevilla and belongs to the IDEA Research Group. He has led various private and public research projects and he has published several papers in high-impact factor journals, including Computers in Industry, ACM Computing Surveys, Empirical Software Engineering, Decision Support Systems, Information and Software Technology, Journal System and Software, Information Systems, among others. He was nominated as a member of Program Committees in different conferences, ISD 2016, BPM Workshops 2017, SIMPDA 2018, SIMPDA 2019, SPLC 2019, and SPLC 2020. He has served as a reviewer for many reputed journals.

María Teresa Gómez-López is PhD in Computer Science, Lecturer at the University of Seville and the head of the IDEA Research Group. Her research areas include Business Processes and Data management in Big Data environment. She has led several private and public research projects and has published more than twenty impact papers (DSS, IS, DKE, IST ·). She was nominated as a member of several Program Committees (BPM, ER, EDOC, CAiSE Doctoral Consortium, ·), and she has been reviewing for international journals. She has been invited speaker at various conferences and summers schools.

Rafael M. Gasca holds a PhD in computer science from the Universidad de Sevilla, in Spain. He is full professor since 2018. He has led the Quivir Research Group since 2000, since 2015, he has been a member of the IDEA Research Group at the Universidad de Sevilla. He has been the leader of different public and private research projects and has directed twelve PhD theses. He has published tens dozens of papers in high-impact factor, including IEEE Computing, IEEE Communications Magazine, Information and Software Technology, Journal System and Software, Information Systems, Information and Software Technology, and Data and Knowledge Engineering. He has been a reviewer in relevant security conferences and journals and an organiser of artificial intelligence conferences and an international summer school on fault diagnosis of complex systems.

Ernest Teniente is Professor of Software Engineering in the Department of Service and Information System Engineering at the Universitat Politècnica de Catalunya (UPC). He is also Director of inLab FIB, the innovation laboratory of the Computer Science Faculty of Barcelona, and head of the Information Modeling and Processing (IMP) research group at the UPC. His research interests include ontologies and conceptual modeling, business process management, automated reasoning, automatic code generation, integrity constraints enforcement, and data integration.

Received: February 17, 2021; Accepted: October 20, 2021.