Context-Aware Compliance Checking

Jan Martijn van der Werf, Eric Verbeek, and Wil M.P. van der Aalst

Department of Mathematics and Computer Science
Technische Universiteit Eindhoven
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
{j.m.e.m.v.d.werf,h.m.w.verbeek,w.m.p.v.d.aalst}@tue.nl

Abstract. Organizations face more and more the burden to show that their business is compliant with respect to many different boundaries. The activity of compliance checking is commonly referred to as auditing. As information systems supporting the organization's business record their usage, process mining techniques such as conformance checking offer the auditor novel tools to automate the auditing activity. However, these techniques tend to look at process instances (i.e., cases) in isolation, whereas many compliance rules can only be evaluated when considering interactions between cases and contextual information. For example, a rule like "a paper should not be reviewed by a reviewer that has been a co-author" cannot be checked without considering the corresponding context (i.e., other papers, other issues, other journals, etc.). To check such compliance rules, we link event logs to the context. Events modify a pre-existing context and constraints can be checked on the resulting context. The approach has been implemented in ProM. The resulting context is represented as an ontology, and the semantic web rule language is used to formalize constraints.

Keywords: auditing, compliance checking, process mining, business rules, ontologies.

1 Introduction

Organizations need to ensure that their business stays within boundaries. A *boundary* restricts the organization in its operation. Boundaries may be imposed by all kinds of external sources, like legislation, regulatory bodies, best practices, but also from sources within the organization itself, e.g., based on its corporate culture or management style. These boundaries are often expressed in terms of the business environment, called the *context*. A boundary may restrict the business in *behavior*, in *structure*, or in *data*.

As a consequence, organizations need to constantly monitor their business. The activity of checking whether the business execution adheres to the defined boundaries is called *auditing*. Traditionally, an audit can only provide reasonable assurance that the business is *compliant*, i.e., that the business is executed within the given boundaries. Auditors can only assess the operating effectiveness by checking samples of factual data.

The advance of information systems supporting organizations in their business enables a new form of auditing. The large amount of data recorded by these systems allow to constantly monitor the business execution. *Continuous auditing* [21, 25] focuses to bring auditing closer to the operational process, and away from the traditional backward-looking once-a-year examination of financial statements [6].

Continuous auditing is an automated form of auditing that can take all execution data into account [6, 11, 25]. As such, this form of auditing can provide more assurance. However in order to automate the process of auditing [4, 7, 16], it first requires the *formalization of the boundaries* in terms of the execution data, and secondly, it can only use context data if an automated link is provided *from the execution data to the context data*. As the boundaries are typically formalized in terms of the context data, and as the automated link from execution data to context data is typically not available, continuous auditing is restricted in its use.

Consider a journal that publishes papers. To support the journal's operation, it may use a system that supports the entire process starting from the submission of a paper to the final publication or rejection. This system typically records everything that happens to a submitted paper into a so-called *event log*. Next to this *Process-Aware Information System* (PAIS) [10], other systems may also collect and store data about this paper. In the remainder of this paper, this other data is referred to as *context data*, while the PAIS data is referred to as *execution data*.

To insure the quality of published papers, a journal needs to discourage undesirable behavior. Therefore, most journals impose a number of *boundaries* on its own operation to guarantee this high quality:

1. For each submitted paper a notification is sent;
2. Each paper needs to be reviewed by at least three different reviewers;
3. The author of a paper cannot be a reviewer of that paper;
4. A reviewer should work at a different affiliation than any of the authors;
5. A reviewer has never been a co-author with any of the authors;

To check whether the journal indeed respects these boundaries, *audits* are used. However, in practical situations, auditing can only be done on a small part of the available data. Hence auditing is incomplete. This may lead, for example, to a paper being rejected by a reviewer which has a conflict of interest, while this conflict does not get noticed.

This paper proposes a link from execution data to context data, which removes some of these restrictions on continuous auditing. As a result of this link, we can bring the execution data to the context data, which allows automated checking of the boundaries and automated linking with other context data.

Please note that although we use a journal to explain the issues at hand, the proposed approach can be used in any other compliance setting. It offers a generic approach for the problem that continuous auditing is restricted to the available execution data.

This paper is organized as follows. Sec. 2 introduces the necessary concepts on which our approach is based. Next, Sec. 3 presents our approach how to link the context data with execution data. Sec. 4 shows the applicability of the approach with a proof of concept, using both existing tools as developing our own. Last, Sec. 5 concludes this paper and hints at possible future directions.

2 Basic Notions

2.1 Business Context

The environment in which an organization like the journal operates is called its *context*. The context is "the combination of all situational circumstances that impact process design and execution" [18], and describes the concepts that influence and bound the business of an organization and the relations between these concepts.

In this paper, we use the notion of a *context model*, which can be viewed as a data model, e.g., using UML class diagrams, the relational database schema or Entity-Relationship diagrams (ERD). A context model defines the *concepts*, their *relationships*, and their *attributes*. Concepts can inherit from other concepts, have different relationships and different attributes. A relationship is from a *source* concept to a *target* concept, where a *cardinality* can be associated to both these concepts. Let $\mathcal{A}$ denote the *attribute name universe*.

Definition 1 (Context model). *A* context model *is a 6-tuple* $(\mathcal{O}, \alpha, \iota, \mathcal{R}, \sigma, \tau)$ *where*

- $\mathcal{O}$ *is a set of* concepts*;*
- $\alpha : \mathcal{O} \to \mathcal{P}(\mathcal{A})$ *is a function defining for each concept the set of attributes;*
- $\iota : \mathcal{O} \rightharpoonup \mathcal{P}(\mathcal{O})$ *is a partial function defining the inheritance relation. If for a concept* $A \in \mathcal{O}$ *a* $B \in \mathcal{O}$ *exists such that* $B \in \iota(A)$*, we say* B *is a* parent *of* A*. The transitive closure of* ι *has to be irreflexive;*
- $\mathcal{R} \subseteq \mathcal{O} \times \mathcal{A} \times \mathcal{O}$ *is a set of* relationships *between concepts such that if* $\{(A, l, B_1), (A, l, B_2)\} \subseteq \mathcal{R}$ *implies* $B_1 = B_2$*. Given a relationship* $r = (A, l, B) \in \mathcal{R}$*,* A *is called the* source *of* r*,* B *is called the* target *of* r*, and* l *is called the* name *of* r*;*
- $\sigma : \mathcal{R} \to (\mathbb{N} \times (\mathbb{N} \cup \{*\}))$ *defines the* source cardinality *(lower bound and upper bound) of each relationship;*
- $\tau : \mathcal{R} \to (\mathbb{N} \times (\mathbb{N} \cup \{*\}))$ *defines the* target cardinality *of each relationship.*

For inheritance, we do not allow overriding of attributes and relationships, i.e., for all $B \in \iota^+(A)$ *and* $l \in \mathcal{A}$ *that* $\alpha(A) \cap \alpha(B) = \emptyset$ *and if* $(B, l, \cdot) \in \mathcal{R}$*, then* $(A, l, \cdot) \notin \mathcal{R}$*, where* ι^+ *is the transitive closure of* ι*.*

An example context model for the journal (using an UML class diagram) is depicted in Fig. 1. In this example context model there are six concepts (Affiliation, Author, Journal, Paper, Researcher, and Reviewer), six relationships (authors, is_submitted_to, reviews, works_at, and two unnamed inheritance relations), and nine attributes (abstract, notificationdate, name, number, publicationdate, submissiondate, title, volume, and year). The works_at relation indicates that each

Researcher is connected to exactly one Affiliation and that an Affiliation is connected to multiple (possibly none) Researchers. Likewise, the reviews relation indicates that a Paper is connected to multiple (possibly none, if not under review) Reviewers, and that a Reviewer is connected to at least one Paper, and the authors relation indicates that an Author is connected to at least one Paper, and a Paper to at least one Author. A researcher can both write papers as well as review those. Therefore, the Author and Reviewer concepts inherit from the Researcher concept.

A context model can be *instantiated*, i.e., the context model can be populated with instances of each concept, and associations between these instances, reflecting the relationships defined on the concepts in the model. Let $\mathcal{U}$ denote the *concept instance universe*, i.e., the set of all possible concept instances, and let $\mathcal{V}$ be the *attribute value universe.*

Definition 2 (Instance of a context model). *An instance of a context model* $M = (\mathcal{O}, \alpha, \iota, \mathcal{R}, \sigma, \tau)$ *is a 3-tuple* (I_O, I_A, I_R) *where*

- $I_O : \mathcal{O} \to \mathcal{P}(\mathcal{U})$ *defines the available* concept instances. *Each concept has a, possibly empty, set of instances;*
- $I_A : (\mathcal{U} \times \mathcal{A}) \to \mathcal{P}(\mathcal{V})$ *defines for each concept instance and attribute the corresponding* set of attribute values*;*
- $I_R : \mathcal{R} \to \mathcal{P}(\mathcal{U} \times \mathcal{U})$ *is the set of* associations *between concept instances.*

Let $A, B \in \mathcal{O}$ *be two concepts, let* $l \in \alpha(A)$ *be an attribute of* A*, and let* $(A, r, B) \in \mathcal{R}$ *be a relationship. Given an instance* $I = (I_0, I_A, I_R)$ *of* M*, we write* $a \in A$ *for* $a \in I_O(A)$*,* $a.l = v$ *for* $((a, l), v) \in I_A$ *and* $(a, b) \in r$ *for* $(a, b) \in I_R((A, r, B))$*. The set of all instances of* M *is denoted by* $\mathcal{I}(M)$*.*

Given two instances $I_1 = (I_{O,1}, I_{A,1}, I_{R,1})$ *and* $I_1 = (I_{O,2}, I_{A,2}, I_{R,2})$*, we define their union by* $I_1 \oplus I_2 = (I_{O,1} \oplus I_{O,2}, I_{A,1} \oplus I_{A,2}, I_{R,1} \oplus I_{R,2})$ *where* $f \oplus g : V \to \mathcal{P}(O)$ *with* $(f \oplus g)(v) = f(v) \cup g(v)$ *for all sets* V*,*O*, functions* $f, g : V \to \mathcal{P}(O)$*, and* $v \in V$*.*

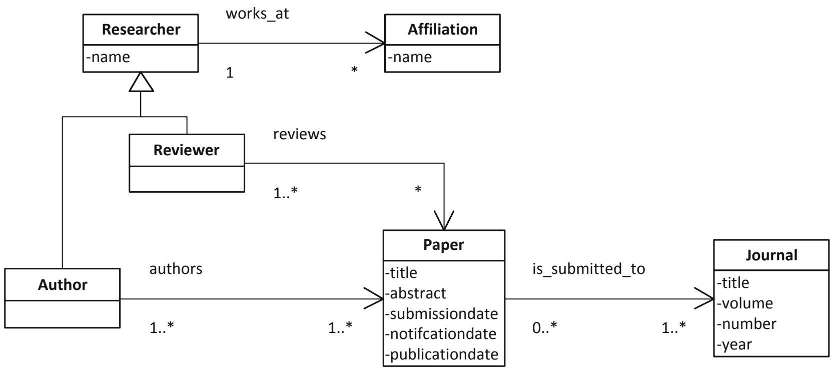


Fig. 1. Context model of the review process context

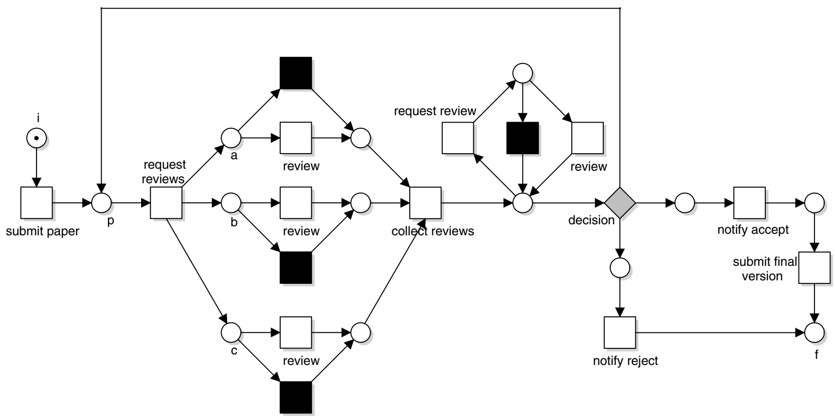


Fig. 2. The review process

Please note that, in principle, an attribute has a set of values, instead of just a single value. For example, a paper will have a set of authors, and not just a single author. However, many attributes will have a singleton set as value. In such cases, we allow the value of the attribute to be the single element of that set instead of the set itself. As a consequence, for a concept instance a and an attribute label l, $a.l$ can either be a set of values ($a.l \in \mathcal{P}(\mathcal{V})$), or the value of its only element in case of a singleton set ($a.l \in \mathcal{V}$). From the context, it will typically be clear which interpretation is used.

An instantiation of a context model is called *consistent* if all the constraints enforced by the cardinalities and inheritance relations are satisfied.

2.2 Process-Aware Information Systems

Business processes form the heart of any organization. A business process can be seen as a set of interdependent tasks and resources needed to produce some product or to deliver some service. In modeling and describing a business process, context plays a natural role. As an example, the names of the activities and resources in the business process are typically aligned with the context in which the process operates. In our example, the journal has a review process that defines for a paper the tasks and resources needed in order to publish the paper in the journal, as depicted in Fig. 2.

In this process model, authors may submit a paper, which will be reviewed by three researchers. Based on the review outcomes, the editor decides whether the paper is accepted, a revision is needed, or whether the paper is rejected, and notifies any accept/reject decision to the authors. Some reviewers may forget to review, or are too late, which is modeled as a skip, using the black transitions. The editor can decide to request an additional review, up to the point that he has sufficient information to make a proper decision.

More and more information systems are developed and implemented to support an organization like the journal in executing their business. A Process Aware Information System (PAIS) [10] assists an organization in the execution of its business process. The business process is then used to configure the system.

A PAIS records each and every step in the business process. For example, the PAIS of the journal records when a paper is submitted, when the reviewers are invited, and whether and when the reviews have been returned. As a result, a PAIS generates enormous amounts of execution data.

For each user action on the system, an *event* is raised. An event records its type, for which activity it has been raised, for which *case* or business process instance, when it was raised, by whom, and the data inserted by the user. Such a recording is called an *event log* [1]. Let $\mathcal{E}$ be the *event universe* and let $\mathcal{C}$ be the *case universe.*

Definition 3 (Event log). *An* event log *is a 3-tuple* $L = (C, E, \#)$ *where*

- $C \subseteq \mathcal{C}$ *is a set of* case identifiers *in the event log;*
- $E \subseteq \mathcal{E}$ *is a set of* event identifiers *in the log;*
- $\# : \mathcal{A} \times (C \cup E) \rightarrow \mathcal{P}(\mathcal{V})$ *is an attribute mapping.*

For an attribute $n \in \mathcal{A}$ *we write* $\#_n(\cdot)$ *as a shorthand for* $\#(n, \cdot)$.

Each event belongs to exactly one case, denoted by the mandatory attribute $case \in \mathcal{A}$, *i.e.,* $\#_{case} : E \rightarrow \mathcal{P}(C)$ *such that* $|\#_{case}(e)| = 1$ *for all* $e \in E$. *Each case has at least one event, i.e., for all cases* $c \in C$ *an event* $e \in E$ *exists such that* $\#_{case}(e) = c$. *An event may have a successor, denoted by* $next \in \mathcal{A}$, *i.e.,* $\#_{next} : E \rightarrow \mathcal{P}(E)$ *such that* $|\#_{next}(e)| \leq 1$ *for all* $e \in E$. *The transitive closure of* $\#_{next}$ *is irreflexive. Each case has exactly one start event, denoted by* $first \in \mathcal{A}$, *and exactly one end event, denoted by* $last \in \mathcal{A}$, *i.e.,* $\#_{first}, \#_{last} : C \rightarrow \mathcal{P}(E)$, *such that* $|\#_{first}(c)| = |\#_{last}(c)| = 1$ *for all* $c \in C$, *if* $\#_{next}(e) = \emptyset$ *then* $\#_{last}(\#_{case}(e)) = e$, *and if no event* $p \in E$ *exists with* $\#_{next}(p) = e$, *then* $\#_{first}(\#_{case}(e)) = e$ *for an event* $e \in E$. *The set of all possible event logs is denoted by* $\mathcal{L}$.

An event log represents a period of business execution. Event logs representing consecutive periods of business execution may be concatenated. Suppose event log L' represents the initial business execution, and L'' represents the consecutive business execution. When a case identifier, say c occurs in both L' and L'', it means that this case was not finished in L'. Consequently, all events for c in L'' follow after the last event for case c in L'. In the concatenation of two event logs representing two consecutive periods of business execution, these cases are concatenated: the first event of the second log is the next event of the last event for this case in the first log. Note that by definition of event logs, the sets of event identifiers of two consecutive event logs need to be disjoint. Further, remark that the functions $\#_{\text{first}}$ and $\#_{\text{last}}$ return singleton sets, i.e., these functions are always defined and return a single event.

Definition 4 (Concatenation of event logs). *Let* $L' = (E', C', \#')$ *and* $L'' = (E'', C'', \#'')$ *be two event logs such that* E' *and* E'' *are disjoint. The* concatenation *of* L' *with* L'', *denoted by* $L'; L''$, *results in a new event log* $(C, E, \#)$ *with*

Table 1. Event log of journal reviewing process

Case	Activity	Resource	Time stamp	Data
118	submit paper	system	24-12-2011 17:00:12	title: "Title paper 118", author: 192 author: 193
119	submit paper	system	24-12-2011 17:05:49	title: "Title paper 119", author: 194
...	...	...	...	...
118	request reviews	editor	29-12-2011 10:19:23	reviewer: 112 reviewer: 149 reviewer: 195
119	request reviews	editor	29-12-2011 10:22:43	reviewer: 112 reviewer: 149 reviewer: 195
118	review	149	07-01-2012 16:39:21	verdict: accept
...	...	...	...	...

$C = C' \cup C''$, $E = E' \cup E''$ *and* $\# = (\#' \setminus \{((\mathit{last}, c), e) \mid c \in C' \cap C'', e \in E'\}) \cup (\#'' \setminus \{((\mathit{first}, c), e) \mid c \in C' \cap C'', e \in E''\}) \cup \{((\mathit{next}, \#'(\mathit{last}, c)), \#''(\mathit{first}, c)) \mid c \in C' \cap C''\}$.

An example of an event log is shown in Tbl. 1. This example represents a small part of an event log generated by the journal review system. It shows for example the submission of papers 118 and 119 with one and respectively two authors, the editor who is requesting reviews for these papers, and a returned review for paper 118 by reviewer 149.

In an event log, we may group the events per case or business process instance. In this way, the events form *execution traces* per case. For example, from the partial event log in Tbl. 1, we have for case 118 the partial execution trace "submit paper", "request reviews" and "review".

The example event log in Tbl. 1 already shows the tight relation between the context and the business execution. For example, solely based on the event log, resource 149 is meaningless, whereas in an instance of the context model, it can be related to some researcher and his affiliation.

2.3 Auditing

Organizations like the journal need to constantly monitor their business to assure that they stay within their boundaries. The activity of checking whether the business execution adheres to the defined boundaries is called auditing [4,7].

Traditionally, an audit can only provide reasonable assurance that the business is *compliant*, i.e., that the business is executed within the given boundaries. As an audit is typically a manual chore, auditors can only assess the operating effectiveness by checking samples of factual data [7].

To audit a system, the auditors may place *process controls* to assess the boundaries, such that the control reports the violation of a boundary [9,14,19,20]. Only if a control is not in place or if it is not functioning as expected, the auditor will typically check factual data. In order to audit a PAIS that supports the review process, a control may be placed counting the number of different reviews per submitted paper. A second control may monitor the time between submission and notification, in order to check the boundary on the time to review a paper.

It is important to realize that a process control may not be sufficient for checking a boundary. Consider, for example, boundary 5, which states that a reviewer may not be a co-author. A control could check, at the moment of the review, whether the reviewer is a co-author, but it will not check this again in the future. Thus, if in the very near future some paper appears in some journal from which it is clear that the reviewer actually was co-authoring a paper, then the control would miss this fact.

Placing controls already increases the assurance an auditor can give. However, adding controls changes the business processes implemented in a PAIS. Hence, if either the business process or a boundary changes, both systems need to be changed. Therefore, process analysis techniques can help in separating process controls from the control flow [17].

3 Context-Aware Continuous Auditing

Since information systems more and more record their usage in event logs, the execution data can be used to check for compliance. Whereas traditional audit only relies on sample-based checks, *continuous auditing* [7, 21] focuses on the analysis of the execution data to perform an audit.

The omnipresence of execution data, coupled with process mining [1] techniques, allows the auditor to perform an audit on the whole business execution, rather than only sample-based [3, 4, 7, 16]. However, boundaries are typically defined on the context data, and not solely based on the execution data [5, 17]. As a result, to be able to check these boundaries, continuous auditing requires both execution data as well as context data. Only if the boundaries are formalized, algorithms and techniques can be developed to automate the process of the auditor. Consequently, not only the boundaries need to be formalized, also the context needs to be modelled.

3.1 Formalization of Boundaries

One way to formalize a boundary is by constraining the concepts in the context model by using the right cardinalities between concepts. In this way, one can easily formalize the boundary that a Paper has at least one Author. It is enforced by the cardinalities of the relationship authors. However, not every boundary can be formalized using only the cardinalities.

Dependencies over different concepts cannot be expressed by local constraints, like the cardinalities, only. For example, the boundary that a Reviewer and Author should be of a different Affiliation cannot be expressed by local constraints. For this reason, we use a constraint language on the context model in first-order logic. The aforementioned boundary can then be expressed by the following non-local constraint:

$$\begin{aligned}
&\forall p \in \text{Paper}, a \in \text{Author}, r \in \text{Reviewer}, w_1, w_2 \in \text{Affiliation} :\\
&\quad (\ (a,p) \in \text{authors} \wedge (r,p) \in \text{reviews} \wedge\\
&\quad\quad (a,w_1) \in \text{works\_at} \wedge (a,w_2) \in \text{works\_at}\\
&\quad) \implies w_1 \neq w_2
\end{aligned}$$

This constraint formalizes that in any instance of the context model, the affiliations of an author and a reviewer of the same paper should be different.

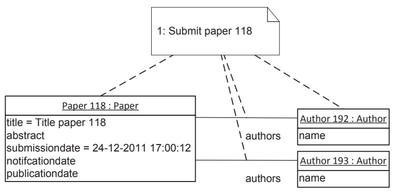
3.2 Relating the Process Execution with the Context

By formalizing the boundaries, compliance checking becomes checking whether each constraint is satisfied by the business execution. However, the constraints are expressed using the context data, while the execution is expressed using the execution data. As a consequence, we need to link both data: Either we transform the constraints in terms of the execution data, or the event log needs to be transformed in terms of the the context data. For example, the rule that a paper has been reviewed by three different reviewers, can be expressed like "Each execution trace contains at least three activities named 'review' executed by different resources". However, many constraints, like the one that corresponds to boundary 5 ("A reviewer has never been a co-author of any of the authors"), are global constraints, i.e., constraints over a set of traces, rather than local constraints over a single execution trace. As a consequence, replay techniques [2] cannot be used, as these only consider single execution traces. Such constraints cannot be transformed into a constraint in terms of the execution data, as the co-author relation does not only depend on the execution data, as authors may decide to submit papers to other journals as well. Clearly, this influences the co-author relation, while these submission will not be stored in the PAIS of the journal at hand.

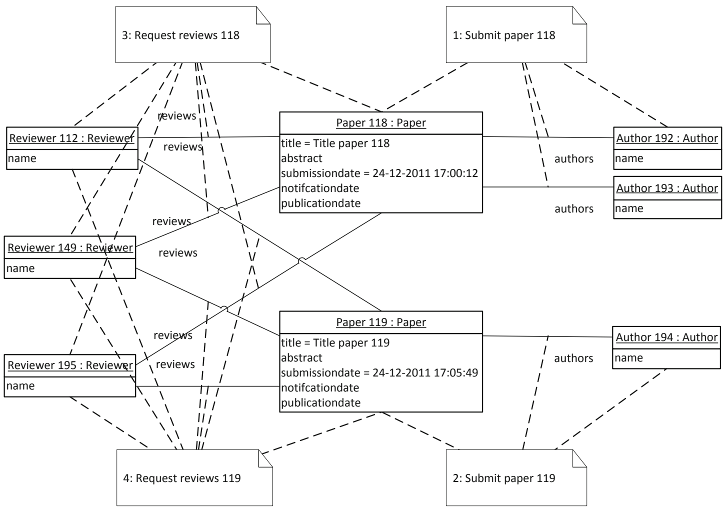
The context model describes the concepts and their relationships. An instance of the context model describes a state of the business. By execution a business process, the state of the business changes, and hence, the instance of the context model. For example, executing the "submit paper" activity corresponds to the insertion of a Paper concept in the current instance. Likewise, the "request reviews" activity corresponds in the insertion of new associations between one instance of Paper and several instances of Reviewer.

Each event in an event log corresponds to an update of the current state of business. As a consequence, executing an event can be seen as a function from one instance of the context model to a new instance. We denote with $\mathcal{I}(M)$ the universe of instances of context model M.

Definition 5 (Transformation function). *Given a context model M, a* transformation function *transforms a given context model instance and an event log into a new instance of the context model, i.e, a function $f : \mathcal{I}(M) \times \mathcal{L} \to \mathcal{I}(M)$ such that for $I \in \mathcal{I}(M)$ and $L_1, L_2 \in \mathcal{L}$ we have $f(I, \emptyset) = I$ and $f(I, L_1; L_2) = f(f(I, L_1), L_2)$.*



(a) After first step



(b) After four steps

Fig. 3. Replaying the event log of Tbl. 1 on the context

In this way, we are able to replay the event log in terms of the context. As each event updates the context, we define an additional function that transforms a context model instance and an event log into a new context model instance. Let us consider the journal example again. Let I be a consistent instance of the context model depicted in Fig. 1, let L be the current event log, and let e be the event that corresponds to the submission of paper 118. The event causes the creation of the Paper 118 object, the Author 192 and Author 193 objects, and it creates the author relation from these Author objects to the Paper object. We can formalize this to an update of the context model instance:

$$f(I, (\{c\}, \{e\}, \#)) = \begin{cases} I \oplus (e_O, e_A, e_R) & \text{if } \#_{\text{name}}(e) = \textit{`submit paper'}; \\ I & \text{otherwise,} \end{cases}$$

where

$$\begin{aligned} e_O &= \{\textit{Paper} \mapsto \{c\}, \textit{Author}, \textit{Researcher} \mapsto \#_{\text{author}}(e)\}, \\ e_A &= \{(c, \textit{title}) \mapsto \#_{\text{title}}(e), (c, \textit{submissiondate}) \mapsto \#_{\text{timestamp}}(e)\}, \\ e_R &= \{(\textit{Paper}, \textit{authors}) \mapsto \{(c, a) \mid a \in \#_{\text{author}}(e)\}\}. \end{aligned}$$

Note that the creation of an object is only necessary if the object does not already exist. If, for example, the object for Author 192 would have already existed, then the existing object would be reused. Likewise, the event that corresponds to the other submission creates the Paper 119 object, the Author 194 object, and the authors relation between both. Fig. 3 shows the result of replaying the first events from Tbl. 1 on an empty journal context model instance.

Similar functions can be created for the other events. The first request review event creates the Reviewer 112, Reviewer 149, and Reviewer 195 objects, and the reviews relations to the Paper 118 object, whereas the second request review reuses the Reviewer objects and creates reviews relations from these objects to the Paper 119 object. As a result of replaying the event log in the context model, we have now extended the context model with the execution data. Hence, we can use the execution data as if it were context data.

3.3 Compliance Checking

The first step in checking compliance, is the check whether the initial instance of the context satisfies each of the constraints. After this check, we can start to replay the event log. For this, we need to sort all events in the event log based on their time stamp, so that the context is updated in the same order as the business has been executed.

While replaying the events on the context model, after each update the current instance needs to be checked to see whether the constraints are still satisfied. If in a step a constraint is violated, we need to report this event and the violation. An auditor then needs to test the severity of the violation. For example, if after completing the trace, the constraint is repaired, the impact of the violation may be less than if the constraint remains violated.

In this approach, we only allow the event log to change the context. In reality, not only the business execution changes the context, also external events, like researchers that change affiliation, or people that write papers for different journals, may change the context. These external changes to the context are (for now) ignored by our approach.

4 Implementation

As a proof of concept, we implemented the approach in the process mining toolkit ProM 6 [22]. In literature, ontologies are often used to capture context (cf. [12,13,18]). An ontology can be seen as a collection of concepts with associations between these concepts [23]. Further, ontologies allow for modularization using the import concept. In this way, different ontologies can be combined into a single context model. For example the context of the journal could be split into an ontology for the publications and reviews, and a separate ontology for the researchers and their affiliation.

4.1 Event Logs as Ontologies

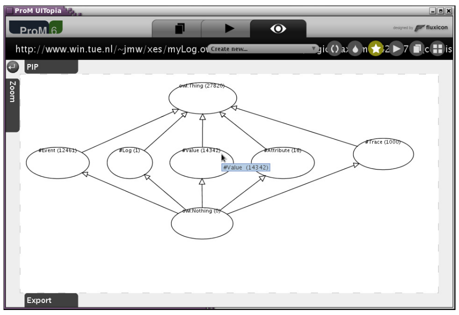
Not only the context model can be represented in an ontology. Also event logs can be represented in an ontology. This allows us to reason over both the context as well as the process execution in a single formal representation. The XES standard [22] defines the important concepts in an event log, and their relationships. In the XES standard, a Log uses zero or more Extensions that each define a set of Attributes. An Attribute has a set of values, represented by the Value concept in the ontology. The concepts Log, Event, Trace and Value are subclasses of the Attributable concept, meaning that they can have Values attached. Each Trace belongs to a Log, and has a start event and an end event. Events belong to a Trace, and each can have a predecessor and a successor Event. The ontology also stores the transitive closure of the predecessor and successor relation, i.e., the set of all predecessors or successors, respectively, of an Event.

The *Ontologies* package of ProM 6 provides an automatic transformation from event logs into the ontology format. For a log of the journal with 1000 cases and 12461 events, the transformation results in an ontology with 455.990 axioms, i.e., concept, individuals, and relations between these individuals.

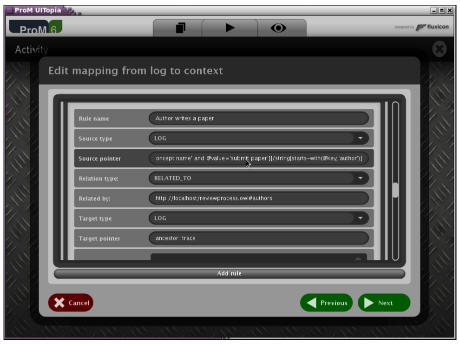
4.2 Linking Events to the Ontology

As seen in the previous section, each event belongs to an update of the context model. For example, the event 'submit paper' creates a paper. In the formalization, we represented this by stating that each trace in the event log creates an instance of the concept Paper. In terms of ontologies, this means that each trace of this event log is not only an individual of the concept Trace, but also of the concept Paper. Hence, we want to add this relation between the ontology of the event log and the ontology of the context model. To automate the process of relating the different individuals, we introduce the notion of annotation rules.

An *annotation rule* relates individuals between the ontologies of the event log and of the context model. It defines a source element, a target element and the



(a) Event log as ontology in ProM 6



Edit mapping from log to context

Rule name	Author writes a paper
Source type	LOG
Source pointer	oncept:name' and @value='submit paper']]/string[starts-with(@key,'author')]
Relation type:	RELATED_TO
Related by:	http://localhost/reviewprocess.owl#authors
Target type	LOG
Target pointer	ancestor::trace

(b) Annotation rule in ProM 6

Fig. 4. Implementation of context models as ontologies in ProM 6

relation between the source and target, like inheritance relation, instantiation, and an ontology object or data property. If the relation between source and target is an ontology property, we also need to specify which property is used.

To select elements from the event log, we use XPath, which allows us to select multiple elements in an event log. For example, the rule that each trace is a paper, can be expressed by the following annotation rule, assuming that the Paper ontology has the namespace `http://localhost/publication.owl`:

AR 1 (A paper) Creates an instance of the concept 'Paper' for each trace

source	`//trace`
relation	instance of
target	`http://localhost/publicatizon.owl#Paper`

This rule selects each trace (`//trace`) in the event log, and adds the corresponding Trace individual to the concept Paper. As the transformation also specifies that a paper has a title, we need another annotation rule to express this.

AR 2 (A paper has a title) The title is specified in the activity 'submit paper' as an attribute named 'title'

source	`//trace`
relation	has data property: `http://localhost/publication.owl#title`
target	`./event[/string[@key='concept:name' and @value='submit paper']]/string[@key='title']@value`

This rule iterates over each trace, as specified in the source of the annotation rule, and gets the corresponding title by retrieving the attribute 'title' of its 'submit paper' event. In case multiple elements are selected by the target XPath query, then a relation is created for each of the selected elements. In this way, we need to create five annotation rules for the transformation of the 'submit paper' event: one to add the Trace to the Paper concept (*AR 1*), two to add the title (*AR 2*) and a similar rule for the submission date of the paper, one to create the authors, and one to relate the authors to the paper.

Fig. 4(b) depicts an annotation rule as it is implemented in ProM 6. Similar annotation rules can be created for each of the events. In this way, we get an ontology that connects the ontology of the event log with the ontology of the context model. This connecting ontology serves as input for the compliance checking.

4.3 Compliance Checking

The approach results in three ontologies: one ontology being the initial context model instance, one ontology representing the event log and one that provides the connection between these two. In order to check whether the organization stayed within its boundaries, we need to check whether all three ontologies together satisfy these boundaries. The previous section showed how each of the boundaries can be formalized into a set of constraints expressed in first order logic. Next step is to express these constraints in a language that can be used on ontologies.

One such language is the Semantic Web Rule Language (SWRL) [8]. SWRL rules are expressed in terms of an ontology. Hence, we can create a separate ontology containing these rules.

For example, to express the rule that states that for each paper no reviewer may be a coauthor of one of the authors of that paper, we first add a data property to each paper, e.g., *violatesBound5*, with domain Paper and the Boolean values as range. Next, we can express when this rule is true: if a paper has been submitted earlier by both a reviewer and an author of the current paper. Expressed in SWRL:

$$\begin{aligned} \mathit{violatesBound5}(?p, \texttt{true}) \leftarrow\ & \mathit{Paper}(?p) \wedge \mathit{reviews}(?r, ?p) \wedge \mathit{authors}(?a, ?p) \\ & \wedge\ \mathit{Paper}(?y) \wedge \mathit{authors}(?a, ?y) \wedge \mathit{authors}(?r, ?y) \\ & \wedge\ \mathit{submitdate}(?y) < \mathit{submitdate}(?p) \end{aligned}$$

As ontologies are being used to express both execution data and context, other formalisms can be exploited to express constraints, like SBVR [15].

Important to realize when working with ontologies is the *open world assumption*. In the open world assumption, a statement is either true, false, or unknown, whereas in a closed world a statement is either true or false. This is an important difference. For example, in the rule above we can only express when the statement is true. If no witness is found, in the closed world assumption the constraint is satisfied, whereas in the open world assumption, the statement results in unknown.

To increase the expressivity of the constraints, the closed world assumption is needed. For example, a logic language like Prolog would be a natural next step, as the translation from ontologies to Prolog is straightforward [24].

5 Conclusions

All kinds of sources enforce boundaries on the way an organization runs its business. Typically, these boundaries are phrased in terms of the environment of the business, called the context. In this paper, we proposed a novel approach to support the auditor in checking whether the organization stays within its boundaries.

Mostly, the work of an auditor is manual. As more and more organizations are supported by information systems that record their usage in event logs, more and more data becomes available to automate the work of the auditor. The enormous amounts of data available allows the auditor to use techniques like process mining.

In order to automate the compliance checking, also the boundaries need to be formalized. In this paper, we propose context models. However, as event logs are in terms of the system, and the boundaries in terms of the context, we argue the need of transforming the information available in event logs into information available in terms of the context.

The approach presented in this paper is a next step towards continuous auditing. To show the applicability of the approach we implemented it in ProM 6 using ontologies as a context model. Although ontologies provide a powerful mechanism to reason over the context, more research is needed to further automate the task of the auditor.

Current replay techniques only visualize control flow related aspects, like conformance checking, of a business execution. The approach proposed in this paper allows to replay the business execution in its context. In this way, business analysts and auditors have the possibility to inspect how business execution changes the business environment.

References

1. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer, Berlin (2011)
2. van der Aalst, W.M.P., Adriansyah, A., van Dongen, B.F.: Replaying History on Process Models for Conformance Checking and Performance Analysis. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 2(2), 182–192 (2012)
3. van der Aalst, W.M.P., van Hee, K.M., van der Werf, J.M.E.M., Kumar, A., Verdonk, M.: Conceptual Model for Online Auditing. Decision Support Systems 50(3), 636–647 (2011)
4. van der Aalst, W.M.P., van Hee, K.M., van der Werf, J.M.E.M., Verdonk, M.: Auditing 2.0: Using Process Mining to Support Tomorrow's Auditor. IEEE Computer 43(3), 102–105 (2010)
5. Accorsi, R., Stocker, T.: On the Exploitation of Process Mining for Security Audits: The Conformance Checking Case. In: ACM Symposium on Applied Computing. ACM (2012)
6. Alles, M.G., Kogan, A., Vasarhelyi, M.A.: Putting Continuous Auditing Theory into Practice: Lessons from Two Pilot Implementations. Journal of Information Systems 22(2), 195–214 (2008)
7. Chan, D.Y., Vasarhelyi, M.A.: Innovation and Practice of Continuous Auditing. International Journal of Accounting Information Systems 12(2), 152–160 (2011)
8. World Wide Web Consortium. SWRL: A Semantic Web Rule Language Combining OWL and RuleML (2011), `http://www.w3.org/Submission/SWRL/`
9. Haworth, D.A., Pietron, L.R.: Sarbanes-Oxley: Achieving compliance by starting with ISO 17799. Information Systems Management 23(1), 73–87 (2006)
10. Dumas, M., van der Aalst, W.M.P., ter Hofstede, A.H.M.: Process-Aware Information Systems: Bridging People and Software through Process Technology. John Wiley & Sons, Inc. (2005)
11. Elliot, R.K.: Assurance Service Opportunities: Implications for Academia. Accounting Horizons 11(4), 61–74 (1997)
12. Filipowska, A., Kaczmarek, M., Kowalkiewicz, M., Markovic, I., Zhou, X.: Organizational Ontologies to Support Semantic Business Process Management. In: International Workshop on Semantic Business Process Management, pp. 35–42. ACM (2009)
13. Fox, M.S., Barbuceanu, M., Gruninger, M.: An Organisation Ontology for Enterprise Modelling: Preliminary Concepts for Linking Structure and Behaviour. Computers in Industry 29(1-2), 123–134 (1996); WET ICE 1995

14. Ghose, A., Koliadis, G.: Auditing Business Process Compliance. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 169–180. Springer, Heidelberg (2007)
15. Goedertier, S., Mues, C., Vanthienen, J.: Specifying Process-Aware Access Control Rules in SBVR. In: Paschke, A., Biletskiy, Y. (eds.) RuleML 2007. LNCS, vol. 4824, pp. 39–52. Springer, Heidelberg (2007)
16. Jans, M., van der Werf, J.M.E.M., Lybaert, N., Vanhoof, K.: A Business Process Mining Application for Internal Transaction Fraud Mitigation. Expert Systems with Applications 38(10), 13351–13359 (2011)
17. Ramezani, E., Fahland, D., van der Werf, J.M., Mattheis, P.: Separating Compliance Management and Business Process Management. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM Workshops 2011, Part II. LNBIP, vol. 100, pp. 459–464. Springer, Heidelberg (2012)
18. Rosemann, M., Recker, J.C., Flender, C.: Contextualisation of Business Processes. Int. Journal of Business Process Integration and Management 3(1), 47–60 (2008)
19. Green, S.: Manager's Guide to the Sarbanes-Oxley Act: Improving Internal Controls to Prevent Fraud. Wiley (2004)
20. Sadiq, S., Governatori, G., Namiri, K.: Modeling Control Objectives for Business Process Compliance. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 149–164. Springer, Heidelberg (2007)
21. Vasarhelyi, M.A., Halper, F.: The Continuous Audit of Online Systems. Auditing: A Journal of Practice & Theory 10(1), 110–125 (1991)
22. Verbeek, H.M.W., Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P.: XES, XESame, and ProM 6. In: Soffer, P., Proper, E. (eds.) CAiSE Forum 2010. LNBIP, vol. 72, pp. 60–75. Springer, Heidelberg (2011)
23. W3C. OWL 2 Web Ontology Language (2009)
24. Wielemaker, J., Schreiber, G., Wielinga, B.: Prolog-Based Infrastructure for RDF: Scalability and Performance. In: Fensel, D., Sycara, K., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, pp. 644–658. Springer, Heidelberg (2003)
25. Williams, B.C.: Auditing and recent Developments in IT. Managerial Auditing Journal 7(5), 18–25 (1992)