Verification of Quantitative Temporal Compliance Requirements in Process Descriptions over Event Logs

Marisol Barrientos, Karolin Winter, Juergen Mangler, and Stefanie Rinderle-Ma[0000−0001−5656−6108]

Technical University of Munich, Germany; TUM School of Computation, Information and Technology
{marisol.barrientos,karolin.winter,
juergen.mangler,stefanie.rinderle-ma}@tum.de

Abstract. Process compliance verification ensures that processes adhere to a set of given regulatory requirements which are typically assumed to be available in a formalized way using, e.g., linear temporal logic. However, formalized requirements are rarely available in practice, but rather embedded in regulatory documents such as the GDPR, requiring extraction and formalization by experts. Due to the vast amount and frequent changes in regulatory documents, it is almost impossible to keep formalized requirements up to date in a manual way. Therefore this paper presents an approach towards compliance verification between natural language text and event logs without the need for requirements formalization. This enables humans to cope with an increasingly complex environment. The approach focuses on quantitative temporal requirements (QTCR) and consists of multiple steps. First, we identify clauses with temporal expressions from process descriptions. Second, we generate a set of QTCR by mapping the retrieved clauses to event log activities. Finally, in the third step, we verify that the event log is compliant with the QTCR. The approach is evaluated based on process descriptions and synthesized event logs. For the latter, we implement time shifting as a concept for simulating real-life logs with varying temporal challenges.

Keywords: Compliance Verification · Temporal Compliance Requirements · Natural Language Text · Process Descriptions · Event Logs

1 Introduction

Business process compliance aims at ensuring that business processes adhere to the constraints that are imposed on them and is a crucial task for companies as non-compliance can lead to severe fines. Compliance verification as the task of verifying process models or event logs against constraints and determining compliance violations has therefore been addressed extensively in the literature. However, most compliance verification approaches require already formalized compliance constraints [4,17,27]. Consequently, compliance verification can be

time-consuming, requires expert knowledge, and might hence be error-prone. Therefore, it is desirable to directly verify compliance of event logs with the source of compliance constraints which are natural language texts, e.g., process descriptions, internal policies, or regulatory documents. Compliance assessment between process models and natural language text has been addressed [28], but approaches for direct compliance verification of natural language text with event logs are, to the best of our knowledge, missing. Due to the complexity of textual descriptions and their interpretation, in this paper, we focus on one aspect of compliance requirements, i.e., temporal requirements, since *"time is one of the most important dimensions that a compliance rule language must tackle"* [17] and second most mentioned perspective in literature after control flow as stated in [27]. Moreover, we consider process descriptions as representative of natural text documents in order to tame the complexity of arbitrary regulatory documents.

The restrictions described in temporal compliance requirements are delimited by temporal information, which is defined as the information used to sequence events and quantify their duration or gaps between them. Temporal information can contain both quantitative and qualitative temporal expressions or relations between these types of expressions [17,20]. An example of a requirement including a qualitative temporal expression is: *activity A must be executed before activity B*. This emphasizes the importance of arranging activities in the proper sequence within the process control flow. Our paper focuses on quantitative temporal expressions, which gave rise to the concept of Quantitative Temporal Compliance Requirements (QTCR) which augment activity executions with temporal conditions, e.g., *activity A must be completed within 10 days*. A QTCR must contain a quantitative temporal expression, however, might not contain qualitative temporal expressions or relations between both of them. This paper aims to explore how to directly verify quantitative temporal compliance requirements expressed in natural language texts over event logs.

The approach takes a process description in natural language text and an event log as input and outputs a set of QTCR violations. The identification of QTCR is a challenging task that involves i) generating one QTCR, which consists of the temporal expression and its related activities, for each temporal expression extracted from the process description, and ii) utilizing event log information to extract a set of unique labels from the *label* attribute. The QTCR are then paired with the most similar label from this set. Once the QTCR are built, the event log is used to extract the desired set of violations, taking into consideration the *label*, *timestamp*, and *life cycle transition* of each event. The latter is essential for reasoning the duration of activities.

The remainder of the paper is structured as follows. First, the problem statement and challenges are outlined in Sect. 2, followed by the foundations for identifying QTCR from natural language texts in Sect. 3. The verification approach is described in Sect. 4 and evaluated in Sect. 5. Sect. 6 discusses the evaluation results and limitations of the approach, while Sect. 7 outlines related work. The paper concludes in Sect. 8.

2 Problem Illustration and Challenges

Figure 1 depicts an artificial running example illustrating the problem addressed in this paper, i.e., extracting QTCR from a process description (top left) and verifying whether they were violated or not based on an event log (bottom left). The corresponding process model is depicted on the right and will be picked up in the data generation section in Sect. 5.

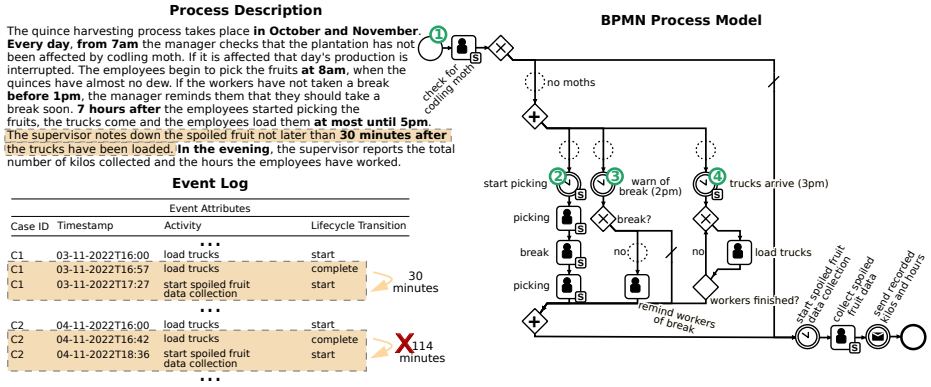

Fig. 1. Running Example: Agriculture Logistics Process

In Fig. 1 all quantitative temporal expressions present in the process description are marked in bold font, e.g., *before 1 pm*. Temporal expressions in our context are parts of text containing temporal information. QTCR are in turn temporal expressions combined with information on activities they refer to. In order to enable compliance verification of QTCR based on an event log, we, first of all, have to identify all sentences containing temporal expressions from the process description. Afterwards, the sentences are processed in order to process activities and their temporal relations resulting in the desired QTCR.

To illustrate this, consider the sentence marked in orange: *The supervisor notes down the spoiled fruit not later than 30 minutes after the trucks have been loaded.*. The relevant elements for QTCR verification are: i) (not later than) 30 minutes after, ii) trucks have been loaded, and iii) the supervisor notes down the spoiled fruit. This leads us to *Challenge 1: How to extract and normalize temporal expressions from text*, where first, we need to determine the type of temporal statement which can be either explicit, implicit, relative, or unspecified (for details cf. Sect. 3, Tab. 2). In the example, the temporal expression is relative because the timestamp is built by counting *30 minutes after the trucks have been loaded*, indicating the duration of the time lag between two activities (*load trucks* and *start spoiled fruit data collection*). However, if the QTCR would be: *after 4 pm, the supervisor notes down the spoiled fruit*, then it will be an explicit temporal expression, determining the starting of an activity. Together

with the temporal extraction, normalization is required, which makes temporal expressions comparable. For instance, the system should automatically deduce that *30 minutes* and *half an hour* represent the same temporal expression. Once temporal expressions are extracted and normalized, we tackle *Challenge 2: How to extract activities that are related to temporal expressions and their temporal relations*. Some temporal restrictions affect the whole process, a single activity or, as in the example, a set of activities, where in this case, in addition, we need to consider specific signal words like *before* and *after* in order to derive the relations between activities. This challenge also covers identifying the involved life cycle transitions, e.g., *30 minutes after* refers to the completion of an activity (*load trucks*), but also indicates the starting of another (*start spoiled fruit data collection*).

When having extracted activities and their temporal relations from the textual source, we scan the log for corresponding events. Hereby, we cannot assume to have exactly the same labels in the event log as in the process description. Therefore, we need to cope with text similarity aspects. (→ *Challenge 3: Include a notion of label similarity*). From the event log perspective, at least the event attributes *event label*, *timestamp*, and eventually, *life cycle transition start* and *complete* must be present in order to enable compliance verification. All three attributes are described within the XES standard [2] and constitute rather basic assumptions. Once we have mapped all relevant elements from the process description onto corresponding events of the event log, we can verify the QTCR.

3 QTCR Elicitation from Natural Language Text

One prerequisite of the presented approach is to identify how QTCR are phrased in natural language texts. In order to come up with a holistic view of this aspect we consider multiple sources to derive a set of possible QTCR phrases in natural language texts. In the following, we study literature on patterns for quantitative temporal expressions, also called time patterns, in the business process management and medical context, elicit and compare those time patterns with process descriptions, and propose to use an extension of Heideltime [25] for extracting time patterns from natural language text.

Process time patterns. [26] presents 15 temporal compliance rules (Petri Net oriented) and [16] 10 time patterns (workflow patterns) where [16] covers all the patterns presented in [26] and includes two additional patterns, i.e., Time-Dependent Variability, and Periodicity, that cannot be fully represented based on Petri Nets. [12] introduce multiple time patterns for BPMN, e.g., *Shifted Duration of an Activity with Reset* (*Hospitalization must last between 24 and 36 hours. If the patient has a fever after 30 hours, the duration count is reset and will re-start after fever disappears.*[12]), and *Shifted Duration of an Activity* (*Effective antibiotic therapy for endocarditis should last between 2 and 6 weeks, which are counted starting from the first day of negative cultures.*[12]). These patterns can be constructed as the combination of patterns presented in [16,26], plus an extra constraint that shifts the starting reference to count time, demonstrating the

complexity of temporal constraints. The semantic analysis of the predicates with quantitative temporal constraints is not considered in [12,16,26].

Process descriptions. [14,15] provide a set of well-established process descriptions. First, we select the process descriptions containing QTCR and identify common process time patterns among them. The patterns found in the process descriptions cover 9 out of 10 time patterns presented in [16], and all patterns provided in [26] (cf. Tab. 1). Note that in [16] the Pattern *Time Lag* is split into two patterns, *Time Lag Between Events* and *Time Lag Between Activities*. However, in our case, we have considered them together since in our setting we are abstracting to event labels.

Time Pattern	Example taken from Process Description
Time Lag	If no response is received after **five days**, a reminder is sent to the claimant.
Duration	If the request is not finished **within 30 days**, then the process is stopped and the employee receives an email cancellation notice and must re-submit the expense report.
Fixed Date	Halfway the week, **on**, a staff meeting of the entire medical team takes place.
Schedule Restricted	**Every morning**, the files which have yet to be processed need to be checked, to make sure they are in order for the court hearing that day.
Time-Based	In a small claims tribunal, callovers occur **once a month**, to set down the matter for the upcoming trials.
Validity Period	Every day, **from 7am** the manager checks that the plantation has not been affected by codling moth, if it is affected that day's production is interrupted.
Cyclic Elements	**On Day 14**, the Internet service is suspended until payment is received.
Periodicity	The process starts periodically **on the first of each month** when Assembler AG places an order with the supplier in order to request more product parts.

Table 1. Time Patterns Retrieved from the Set of Process Descriptions

According to Tab. 1, *each morning* is classified within pattern *Schedule Restricted* because it refers to a fixed schedule. However, by analyzing the phrase semantically it could also be classified as *Periodicity* because it refers to a periodical recurring process element. Further, we can observe that *five days*, belongs to *Time Lag*, it corresponds to the time between activities, but it could also be seen as a *Validity Period* because after those five days, the reminder is sent.

From the natural language processing point of view, these patterns help to understand the type of data we have, but they do not allow us to reach an automatic classification. Besides, they do not follow a standard temporal pattern nomenclature, which makes it unfeasible to compare with other temporal patterns proposed in the past. [16] emphasize that time constraints often come

as a side note, or as text added to the process specification, and might change depending on the process implementation.

Medical domain. For these reasons, we consider literature outside the business process management community. In the medical domain, the TimeML [21] (ISO-TimeML [22] based on the TIDES standard) has been consolidated, which is used to extract and normalize temporal information, events, and their relations. It follows Allen's interval algebra [13] for reasoning about quantitative temporal information, which defines the 13 possible temporal relationships that exist between two temporal intervals [7]. Temporal information is well defined when, given two points in time or two intervals, the relationship between them can be recognized. [6] survey diverse methods and corpora for the extraction of temporal relations in clinical-free texts. In this survey, those who obtained the best results made use of TimeML, and Time Ontology in OWL [1], among other time-related schemas and ontologies. In [5], they present an ontology of temporal concepts that extend Allen's interval algebra to handle uncertain time intervals, making it possible to solve some of the problems presented in [16].

In order to take advantage of the research done in the medical domain and align the common objectives of both communities, we decided to extract the quantitative temporal expressions from the text following the TimeML patterns, in particular, by following TIMEX3, its sub-standard that describes time. Within the tools used to extract temporal annotations, the following stand out: Heideltime [25] (rule-based), SUTime[11] (rule-based), and spaCy's extension timexy[1]. The last one is not evaluated yet in the existing datasets, such as TempEval-3[2] from SemEval workshop, therefore we excluded it. We used the web interface presented in [8] to test Heideltime and SUTime. The one that was able to identify temporal expressions from [14,15] with a higher scope was Heideltime. To improve the results given by Heideltime, a set of new rules was added, to cover temporal expressions which can be found in the context of business process management, e.g., *first working day of the month* or *day 1*. The new set of rules can be found in the repository. Heideltime introduces the concept of *Realization of point expression* [25], which encompasses all possible temporal expressions that can be found in natural language, allowing its normalization. Tab. 2 shows an example of each type of realization, together with the four different found types.

Realization	Compliance Requirement	TIMEX3	Normalized
Explicit	staff meeting **on Wednesday**	DATE	2022-11-23
Implicit	**in the evening**, the supervisor reports it	TIME	XX-XXTEV
Relative	**30 minutes** after the trucks have finished	DURATION	PT30M
Unspecified	callovers occur **once a month**	SET	P1M

Table 2. Realization and TIMEX3 Annotation Example

[1] `https://pypi.org/project/timexy/`, last access: 2023-03-24
[2] `https://paperswithcode.com/dataset/tempeval-3`, last access: 2023-03-24

TIMEX3 of type *DATE* are points of granularity *day*, *month*, or any greater, while the type *TIME* refers to a granularity smaller than a day (e.g., minutes). The type *SET* represents a periodical aspect of an event, describing a set of times or dates, or frequency within a time interval (e.g.,*once a month*). Explicit realizations of point expressions correspond to *DATE*s or *TIME*s, they do not need further knowledge to be normalized. However, implicit expressions need a point from which to normalize, as for example in the temporal expression *in the evening* it is required to know from and until when is considered to be *evening*, they must be defined by a user. Furthermore, in our particular use case, it is common to see expressions like *working days*, they must be defined by an user. Relative types cannot be normalized unless there is context information, as we can see in Tab. 2 it is necessary to know when the trucks have finished.

4 QTCR Verification Approach

Figure 2 depicts the proposed QTCR verification approach. The first step involves annotating temporal expressions from the process descriptions. These can be defined as abstract concept that contains the title of the process description, the original sentence, the identified time together with its TIMEX3 type, and its normalized value. Afterwards, we create one QTCR per annotated temporal expression through several intermediate steps (cf. sub-process in Fig. 2). Within this sub-process until the fifth step the knowledge from the event log is not required. In the last step, we determine QTCR violations. The approach is illustrated based on the running example presented in Sect. 2.

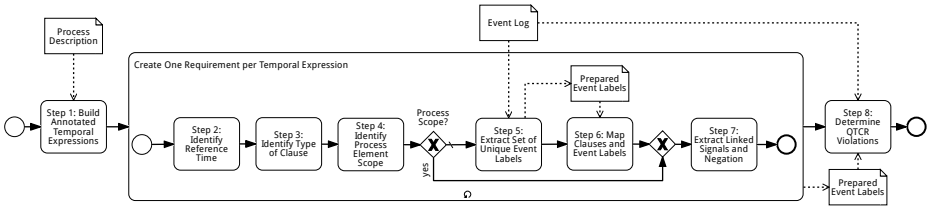

Fig. 2. Overview of QTCR Verification Approach

Step 1 - Build Annotated Temporal Expressions. Within this step, we, first of all, identify all sentences containing at least one temporal expression, i.e., in the case of the running example presented in Sect. 2 all sentences containing terms in bold font. Phrases without temporal expressions are not considered. We annotate each temporal expression using our extended version of Heideltime, as described in Sect. 3, resulting in a set of annotated sentences. Consider, e.g., the sentence from the running example: *The supervisor notes down the spoiled fruit not later than 30 minutes after the trucks have been loaded*. As we can deduce from Tab. 2, the identified time *30 minutes* will be annotated as *DURATION*, with a normalized value equal to *PT30M*. In the next step, we underline that, in order to complete its normalization in this case, context information is needed.

Step 2 - Identify Reference Time. At the end of Sect. 3, it was indicated that the relative realizations of point expressions, those of *DURATION* type, require a reference time in order to be correctly normalized. In our running example, the reference point is the instant (timestamp) corresponding to the activity when the trucks have been loaded (life cycle transition equaling complete). However, if we find expressions, implicit or unspecific, it is not necessary to calculate the reference point because their normalized temporary expressions already contain enough information. In this case, the beginning of each instance of the event log is taken as the reference point, e.g., if the first activity in the event log started on the 21st of November of 2022, then the expression *Wednesday* will be normalized as 23-11-2022. Step one and two address *Challenge 1.*

Step 3 - Identify Type of Clauses. Time constraints that are contained within conditional clauses have to be treated differently than those contained in declarative clauses. Conditions represent decisions that vary the control flow. This prompts us to differentiate between declarative, condition, and consequent clauses. By looking at the running example, in the sentence: *If the workers have not taken a break before 1 pm, the manager reminds them that they should take a break soon*, we can see that the manager reminds the workers, after 1 pm, to take a break only if the workers have not taken a break before 1 pm. This means that if they took the break before 1 pm they should not be reminded. Our approach splits this sentence into two clauses, one containing the condition clause and the other containing the consequence clause. In the following, steps is shown how each of those clauses is mapped with their corresponding activities and restricted time expressions (i.e. a temporal expression plus a signal). From this step until step number seven the *Challenge 2* is addressed. As mentioned in [16], temporal restrictions can be adjusted to different process elements (single activity, activity set, process model, or set of process instances), if the sentence has any conditional clause then the time restriction will affect a set of activities.

Step 4 - Identify Process Element Scope. Temporal expression boundaries can also refer to the general scope of the process by limiting the start and completion time of the first or last activity, e.g., *the quince harvesting process takes place in October and November.* It is clear that the temporal expressions *October* and *November* constitute a boundary for the duration of the process. By looking at Fig. 2, we can see that when the QTCR delimits the full process, then steps five and six of the approach are not needed because for instance, the only involved activities are the first, the last activity, or both. Otherwise, if the sentence refers to a specific activity then steps 5 and 6 cannot be left out.

Step 5 - Extract Set of Unique Event Labels. This step takes the event log as input and identifies all possible unique event labels from all traces. Moreover, the labels are transformed into a lemmatized bag-of-words representation. In the running example, the event label *load trucks* is transformed into {*load, truck*}. As during the next step, i.e., the mapping, we make use of similarity computations between activity and event labels, we ensure via such pre-processing, that noise is reduced and only relevant terms contribute to the comparison. The prepared

labels can now be compared with the clauses present in the sentence containing the temporal expression in the next step.

Step 6 - Map Clauses and Event Labels. After extracting the set of unique event labels from the event log, and identifying the different types of clauses present in each temporal expression sentence, each clause is turned into a lemmatized bag-of-words representation. The mapping between the lemmatized clauses and lemmatized event labels is performed by computing the cosine similarities between the embedding of tokens from all the event labels (from the event log), and the clauses (from the sentences containing a temporal expression). This task addresses *Challenge 3* and was carried out by using BERTScore[3] [29], a method used for automatically evaluating the performance of text creation systems. The running example sentence contains these two clauses *the supervisor notes down the spoiled fruit* and *the trucks have been loaded*. If we look at the event log label candidates extracted for the first clause the first top two are *Collect Spoiled Fruit Data* and *Start Spoiled Fruit Data Collection*, while for the second clause, they are: *Trucks Arrive (3 pm)* and *Load Trucks*. After identifying which event log label each clause belongs to, we proceed to identify signal words and whether or not negative constraints are present (e.g., *the staff meeting is never on Wednesday)*.

Step 7 - Extract Linked Signals and Negation. After knowing which activities are detected for each temporal expression, we proceed to extract their relations. This step is taken after the others in order to know to what activity or temporal expression belongs to each relation, otherwise in cases where in the same sentence there are more than one temporal expressions we could not distinguish between the relations from each of them. For example in the sentence *7 hours after the employees started picking the fruits, the trucks come and the employees load them at most until 5 pm*, the signal *after* will be linked with the activity *started picking the fruits* and the temporal expression *7 hours*, while the signal *until* will be linked to *load trucks* and *5 pm*. We summarize the signals in three categories: *AFTER*, *BEFORE*, and *IN*, which represent all of them. Then we check if the clause is negated to see if the temporal requirement has to be excluded, e.g., *this must not happen in October*. Lastly, we identify based on the extracted data the life cycle transition (start, complete, or both) corresponding to each QTCR.

Step 8 - Determine QTCR Violations. This step takes as input the event log and the extracted requirements and delivers all QTCR violations. For each requirement, it must be verified that the process or activities belonging to the consequence or declarative type have the desired timestamps and states, otherwise, they are considered violations. Consider again the running example Fig. 1. The extracted temporal requirement identified contains the temporal expression of *30 minutes* (of type *DURATION*, and norm. *PT30M*) Tab. 2, and the following activities involved Tab. 3. This allows to automatically check if the event log trace meets the requirement of: by counting 30 minutes from the timestamp cor-

[3] `https://github.com/Tiiiger/bert_score`, last access: 2023-03-24

responding to the complete state of the *load tracks* activity, all the traces from *start spoiled data fruit collection* (both started and finished) should be before.

Event Log Label	Type	Status	Signal	Reference
truck load	declarative	complete	in	true
start spoiled data fruit collection	declarative	start & complete	before	false

Table 3. Requirement Extracted from Running Example (short version)

5 Evaluation

The QTCR verification approach is prototypically implemented[4] in Python 3.10 using the tools and packages mentioned in Sect. 4 and additionally using *NLTK*[5] and the *Python Heideltime Wrapper*[6]. The data set[7] for evaluating the approach is based on 14 process descriptions from [14,15] and the running example. The data generation part is outlined in Sect. 5.1 and the evaluation results are detailed in Sect. 5.2. Each of the 8 steps in the approach has been evaluated independently and in addition, we provide an overall end-to-end evaluation result.

5.1 Data Generation

For the data generation part, we started by modeling the process descriptions using the Cloud Process Execution Engine[8] (CPEE) [18,19,24]. For each of the scenarios, we created executable models with the following properties: i) the labels follow the text as close as possible, ii) the duration of tasks is correct in relation to each other, but is scaled down to seconds, iii) the duration of tasks slightly varies so that both successful executions as well violations are possible (according to the scenario text), and iv) we included some plausible data flow, to aid future work based on the generated logs.

Based on these simple models, which albeit only span the duration of some seconds, we introduced a time-shifting mechanism, to generate logs that span a realistic time frame based on the description in the scenarios. The time-shifting algorithm works based on a set of annotations to the process model:

- **Start Event:**
  - Multiplication Factor (MF): how much should the duration in seconds be scaled up, e.g., if the value here is "1 hour" then every second is scaled up to one hour.

[4] `https://www.cs.cit.tum.de/bpm/software/`, last access: 2023-03-24
[5] `https://www.nltk.org/`
[6] `https://github.services.devops.takamol.support/PhilipEHausner/python_heideltime`, last access: 2023-03-24
[7] `https://www.cs.cit.tum.de/bpm/data/`, last access: 2023-03-24
[8] `https://cpee.org`, last access: 2023-03-24

  - Starting Point (SP): a natural language expression for shifting the start of the process, e.g., an arbitrary date and time might be used as a start. Another important requirement is, that this expression can be dynamically set when instantiating this process, to allow for the creation of a series of logs for different days or months.
  - Random +/- Starting Shift (R): a small piece of code that returns the amount of time to add or subtract from the starting point. As this can include logic for random values, instantiation the process multiple times leads to variations in the resulting logs.
- **Activity:**
  - Type (T): an activity might either be shifted to have (a) a specific duration (DUR), or (b) a specific end (END). For both cases all the subsequent task start events have to be shifted, for case (b) the duration of the task has to be adjusted accordingly.
  - Expression (E): a natural language expression. Again code snippets can be provided, to realize randomness.

For the running example presented in Fig. 1, the following values were chosen: (1) utilizes a fixed (MF) of 1 hour, a fixed starting last (SP) (e.g., 2022-11-01), and a random starting shift (R) of 15 minutes. In (2) (T) is set to (END) and (E) to a random range of 0 to 15 minutes after 08:00. In (3) (T) is set to (END) and (E) to 13:00. Finally in (4) (T) is set to (END) and (E) is set to 15:00 plus a random range of 0 to 15 minutes. We then executed this file 50 times, which resulted in a varied set of logs, which contain both violations and successful executions (see footnote 7).

5.2 Evaluation Results

In order to cope with the fact that errors can add up throughout the approach, we provide intermediate results for each of the steps taken in the approach as presented in Sect. 4. To do so, we define the following evaluation targets. We measure precision, recall, and F1 score for temporal expression extraction and annotation with evaluation targets a) original Heideltime rules and b) including the updated Heideltime rules covering Step 1. Moreover, we evaluate c) the identification of reference time for Step 2, d) the type of clause for Step 3, e) process element scope for Step 4, f) event log labels mapping for Steps 5 and 6, g) signals and negations for Step 7, and h) identification of QTCR violations for Step 8.

All 14 selected process descriptions from [14,15] as well as the running example are used for evaluating a) and b). For evaluation target a), i.e., using the original version of Heideltime we received a precision of 0.9091, recall of 0.6452, and F1 score of 0.7547. Updating the Heideltime rules ($\mapsto$ b)) results in a precision of 0.9744, recall of 0.9268, and F1 score of 0.9500. Those scores are an average overall of 15 process descriptions. The included rules provided support for temporal expressions found in business process descriptions, where they talk

about workdays, processing times, etc. However, the new rules do not completely allow for the identification of temporal expressions that refer to past or future activities, such as *at the same time as the next activity is being executed*.

Considering the temporal patterns and their possible phrasing as outlined in Sect. 3, we selected 4 process descriptions (including the running example) that are covering all possible QTCR phrases and are therefore representative for further evaluating targets c) to h). For each process description, we generated synthesized event logs with 50 traces each, as stated in Sect. 5.1.

After the evaluation of c) we observed that we were able to distinguish between the requirements that refer to the start or completion of another activity, an activity including shifting, or the starting of the process. In all the scenarios, the different types of clauses (i.e. declarative, condition, and consequence) were identified ($\mapsto$ d)), even for cases where a sentence contained multiple clauses accompanied by conjunctive and disjunctive operators. Moreover, our approach was able to differentiate whether the requirement was referring to the full process or not ($\mapsto$ e)). However, this could bring further challenges e.g., the case of evaluating the events of the log of a sub-process that is embedded in the event log of a general process, where not only the starting and completion of the main process is relevant but also the related sub-process.

The precision scores for evaluation targets f), g), and h) are summarized in Tab. 4. It can be deduced that the approach delivered in general satisfying results. In particular, for evaluation target f) the precision evaluates on average 0.76. The score for *Billing Process of ISP* is notably lower and that is due to the appearance of sentences with similar semantics, the differences in meaning are almost negligible even for an expert in the field (e.g., *charge late fee*, *debit outstanding amount*). For the evaluation of target g) we have at least a precision of 0.80 for the *Expense Report* example and for half of them, we achieved again a precision of 1. The cases that failed were due to the combination of signals, as in the case of *not later than 30 minutes after*. Of the 200 traces used to evaluate h), 20% of them violated a QTCR. In three of the scenarios, the precision was at least 0.89, however, in *Billing Process of ISP* the performance was affected by the presence of incorrect labels and an unusual QTCR that contained a compound noun referring to data attributes, i.e. *on Day 10, the transaction that failed on Day 8 is re-attempted* demonstrating the accumulation of errors within the pipeline.

Process Description	f)	g)	h)
Quince Harvesting	0.72	0.81	0.89
Meeting Related Activities	1.00	1.00	1.00
Billing Process of ISP	0.57	1.00	0.10
Expense Report	0.75	0.80	1.00

Table 4. Precision for Targets f), g), and h)

The evaluation results demonstrate how our approach is capable of satisfactorily identifying temporal expressions ($\mapsto$ a) and b)) in business process descriptions along with the reference time ($\mapsto$ c)). Each extracted expression gives shape to a QTCR and in all the cases mentioned in Tab. 4, the clauses and the process element scope, i.e., boundaries of the temporal expression, were correctly

identified ($\mapsto$ d) and e)). In the last three steps, the extraction of signals, negations, mapping between extracted clauses with the labels presented in the log, and identification of events that violated the QTCR were evaluated ($\mapsto$ f), g), and h)). Taking the average precision obtained to evaluate these three aspects, *Billing Process of ISP* was the case delivering the lowest precision accounting to 0.55. The case with the highest precision was *Meeting Related Activities* accounting for 1.00. The overall average precision when taking into account all four cases is 0.80.

6 Discussion and Limitations

The applicability and limitations of the approach are discussed below.

Applicability. In the domain of process compliance verification, the applicability of this approach is particularly relevant for organizations that need to ensure their processes adhere to regulatory requirements embedded in natural language text documents, such as the GDPR. For instance, consider the GDPR's requirement for timely notification of data breaches to relevant authorities within 72 hours. By focusing on QTCR, the approach can efficiently identify clauses with temporal expressions, map them to event log activities, and verify compliance without relying on the manual formalization of these requirements. Time-shifting can be used in this context to simulate real-life logs by introducing variations in the time taken for organizations to report breaches, assessing the proposed approach's ability to identify compliance issues accurately. This approach is applicable to a wide range of industries and organizations, allowing them to cope with increasingly complex regulatory environments. The evaluation using synthesized event logs and the implementation of time-shifting demonstrate the potential applicability to real-world scenarios, offering a more adaptable way of verifying process compliance and addressing the challenges associated with the manual formalization of regulatory requirements. Additionally, the manual creation of formal rules can be error-prone and time-consuming, making it challenging to maintain up-to-date compliance requirements, further emphasizing the need for an automated solution like the one proposed in this paper.

Generalizability. One crucial part of the approach consists of identifying QTCR from natural language text. In that regard, we need to cope with natural language flexibility and a potential lack of comprehensiveness w.r.t. covering all possible formulations of temporal compliance requirements. In particular, when considering more complex documents like regulatory documents, we might lack generalizability as the possibilities of how QTCR are formulated can be more diverse. During the mapping, one limitation of the approach relates to too similar event labels, e.g., *remind of break* and *take break* as well as *insufficient length of event labels* in the event log. For the latter consider again the running example, i.e., the label in the event log reflecting the activity of the workers needing to take a break is just given as *break*. Such short event labels make it almost impossible to detect the correct mapping between event labels and activities from the process description. This could be overcome by, e.g., including more information

from the log file like additional event attributes, in this case, the organizational resource. A similar observation holds for including data attributes which can contribute to extending the semantics of the labels to enhance the mapping.

Ambiguity and lack of contextual information. Moreover, as we realized during modeling the process descriptions for the log generation, missing activity links and references in the texts make it almost impossible to clearly determine which activities are involved in a temporal compliance requirement and what are the reference points to measure compliance violations. Furthermore, according to [9], there is a lack of objectivity in process descriptions, resulting in modelers adopting diverse modeling styles.

7 Related Work

This paper bridges the gap between multiple topics. First, related work on extracting process-related information from natural language text. Imperative model extraction resulting in BPMN models is presented in [15], while [3] extract Declare models from process descriptions and [23] focus on decision model extraction. Most recent approaches exploit deep learning, in particular, GPT-3 models for business process entity and relation extraction from natural language texts [10]. However, those papers do not explicitly address the extraction of quantitative temporal aspects from process descriptions as it is one of the aims of this paper. Moreover, those works solely focus on extracting process models from natural language text and do not touch upon compliance verification at all. Another aspect this paper addresses is ex-post quantitative temporal compliance verification. [4] present an LTL checker. [26] formalize 15 temporal compliance rules allowing for checking temporal compliance rules on a process execution log using alignments. In both cases, the formalization needs to be done manually, i.e., compared to the presented approach it is crucial to have experts formalize the rules based on the given natural language text. As compliance regulations can change frequently, existing approaches such as [4,26] are not as flexible as the approach presented in this paper since both require frequent re-formalization and adding of newly formalized rules. For related work on extracting temporal aspects from natural language text in the medical domain, we refer to Sect. 3.

8 Conclusion and Future Work

Compliance verification constitutes a crucial task within business process compliance management. Typically compliance requirements are formalized in, e.g., linear temporal logic and then checked against an event log. In order to reduce the manual effort, and errors that can arise during the formalization and cope with an increasingly complex regulatory environment subject to frequent changes, we have presented an approach that directly verifies quantitative temporal compliance requirements in natural language text over event logs. For this, we employed a multi-step approach based on natural language processing and evaluated it on

a set of well-established process descriptions. The corresponding event logs were generated using the novel concept of time shifting for the cloud process execution engine. For future work, we plan to extend the approach towards more complex documents such as regulatory documents, data constraint verification, and compliance monitoring at run-time, i.e., on process event streams.

Acknowledgements This work has been partly funded by SAP SE in the context of the research project "Building Semantic Models for the Process Mining Pipeline" and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – project number 277991500.

References

1. Time Ontology in OWL (2006), https://www.bibsonomy.org/bibtex/21e7da807dca7f94e75ddcd28567da745/zazi
2. IEEE standard for extensible event stream (xes) for achieving interoperability in event logs and event streams. IEEE Std 1849-2016 pp. 1–50 (2016). https://doi.org/10.1109/IEEESTD.2016.7740858
3. van der Aa, H., Ciccio, C.D., Leopold, H., Reijers, H.A.: Extracting declarative process models from natural language. In: Advanced Information Systems Engineering. pp. 365–382 (2019). https://doi.org/10.1007/978-3-030-21290-2_23
4. van der Aalst, W.M.P., de Beer, H.T., van Dongen, B.F.: Process mining and verification of properties: An approach based on temporal logic. In: On the Move to Meaningful Internet Systems. pp. 130–147 (2005). https://doi.org/10.1007/11575771_11
5. Achich, N., Ghorbel, F., Hamdi, F., Métais, E., Gargouri, F.: Approach to reasoning about uncertain temporal data in OWL 2. In: Knowledge-Based and Intelligent Information & Engineering Systems. pp. 1141–1150 (2020). https://doi.org/10.1016/j.procs.2020.09.110
6. Alfattni, G., Peek, N., Nenadic, G.: Extraction of temporal relations from clinical free text: A systematic review of current approaches. J. Biomed. Informatics **108**, 103488 (2020). https://doi.org/10.1016/j.jbi.2020.103488
7. Allen, J.F.: Maintaining knowledge about temporal intervals. Commun. ACM **26**(11), 832–843 (1983). https://doi.org/10.1145/182.358434
8. Aumiller, D., Almasian, S., Pohl, D., Gertz, M.: Online dateing: A web interface for temporal annotations. In: Research and Development in Information Retrieval. pp. 3289–3294 (2022). https://doi.org/10.1145/3477495.3531670
9. Beerepoot, I., Ciccio, C.D., Reijers, H.A., Rinderle-Ma, S.: The biggest business process management problems of our time. In: Proceedings of the International Workshop on BPM Problems to Solve Before We Die (PROBLEMS 2021). CEUR Workshop Proceedings, vol. 2938, pp. 1–5. CEUR-WS.org (2021), http://ceur-ws.org/Vol-2938/paper-PROBLEMS-01.pdf
10. Bellan, P., Dragoni, M., Ghidini, C.: Extracting business process entities and relations from text using pre-trained language models and in-context learning. In: Enterprise Design, Operations, and Computing. pp. 182–199 (2022). https://doi.org/10.1007/978-3-031-17604-3_11
11. Chang, A.X., Manning, C.D.: Sutime: A library for recognizing and normalizing time expressions. In: Conference on Language Resources and Evaluation. pp. 3735–3740 (2012)

12. Combi, C., Oliboni, B., Zerbato, F.: A modular approach to the specification and management of time duration constraints in BPMN. Inf. Syst. **84**, 111–144 (2019). https://doi.org/10.1016/j.is.2019.04.010
13. Drakengren, T., Jonsson, P.: Maximal tractable subclasses of allen's interval algebra: Preliminary report. In: Artificial Intelligence. p. 389–394 (1996)
14. Dumas, M., Rosa, M.L., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management. Springer (2013). https://doi.org/10.1007/978-3-642-33143-5
15. Friedrich, F., Mendling, J., Puhlmann, F.: Process model generation from natural language text. In: Advanced Information Systems Engineering. pp. 482–496. Springer (2011). https://doi.org/10.1007/978-3-642-21640-4_36
16. Lanz, A., Weber, B., Reichert, M.: Time patterns for process-aware information systems. Requir. Eng. **19**(2), 113–141 (2014). https://doi.org/10.1007/s00766-012-0162-3
17. Ly, L.T., Maggi, F.M., Montali, M., Rinderle-Ma, S., van der Aalst, W.M.P.: Compliance monitoring in business processes: Functionalities, application, and tool-support. Inf. Syst. **54**, 209–234 (2015). https://doi.org/10.1016/j.is.2015.02.007
18. Mangler, J., Rinderle-Ma, S.: Cloud process execution engine: Architecture and interfaces (2022). https://doi.org/10.48550/ARXIV.2208.12214
19. Mangler, J., Stuermer, G., Schikuta, E.: Cloud process execution engine-evaluation of the core concepts. arXiv preprint arXiv:1003.3330 (2010)
20. Meiri, I.: Combining qualitative and quantitative constraints in temporal reasoning. Artif. Intell. **87**(1-2), 343–385 (1996). https://doi.org/10.1016/0004-3702(95)00109-3
21. Pustejovsky, J., Ingria, R., Saurí, R., Castaño, J.M., Littman, J., Gaizauskas, R.J., Setzer, A., Katz, G., Mani, I.: The specification language TimeML. In: The Language of Time - A Reader, pp. 545–558 (2005)
22. Pustejovsky, J., Lee, K., Bunt, H., Romary, L.: ISO-TimeML: an international standard for semantic annotation. In: Language Resources and Evaluation (2010)
23. Quishpi, L., Carmona, J., Padró, L.: Extracting decision models from textual descriptions of processes. In: Business Process Management. pp. 85–102 (2021). https://doi.org/10.1007/978-3-030-85469-0_8
24. Stertz, F., Rinderle-Ma, S., Hildebrandt, T., Mangler, J.: Testing processes with service invocation: Advanced logging in cpee. In: Service-Oriented Computing. pp. 189–193 (2016)
25. Strötgen, J., Gertz, M.: Heideltime: High quality rule-based extraction and normalization of temporal expressions. In: Workshop on Semantic Evaluation. pp. 321–324 (2010)
26. Taghiabadi, E.R., Fahland, D., van Dongen, B.F., van der Aalst, W.M.P.: Diagnostic information for compliance checking of temporal compliance requirements. In: Advanced Information Systems Engineering. pp. 304–320 (2013). https://doi.org/10.1007/978-3-642-38709-8_20
27. Voglhofer, T., Rinderle-Ma, S.: Collection and elicitation of business process compliance patterns with focus on data aspects. Bus. Inf. Syst. Eng. **62**(4), 361–377 (2020). https://doi.org/10.1007/s12599-019-00594-3
28. Winter, K., van der Aa, H., Rinderle-Ma, S., Weidlich, M.: Assessing the compliance of business process models with regulatory documents. In: Conceptual Modeling. pp. 189–203 (2020). https://doi.org/10.1007/978-3-030-62522-1_14
29. Zhang, T., Kishore, V., Wu, F., Weinberger, K.Q., Artzi, Y.: Bertscore: Evaluating text generation with BERT. In: Learning Representations (2020)