Verifying Resource Compliance Requirements From Natural Language Text Over Event Logs

Henryk Mustroph[1], Marisol Barrientos[1], Karolin Winter[2], and Stefanie Rinderle-Ma[1]

[1] Technical University of Munich, TUM School of Computation, Information and Technology, Garching, Germany
{henryk.mustroph,marisol.barrientos,stefanie.rinderle-ma}@tum.de
[2] Eindhoven University of Technology, Department of Industrial Engineering and Innovation Sciences, Eindhoven, The Netherlands
k.m.winter@tue.nl

Abstract. Process compliance aims to ensure that processes adhere to requirements imposed by natural language texts such as regulatory documents. Existing approaches assume that requirements are available in a formalized manner using, e.g., linear temporal logic, leaving the question open of how to automatically extract and formalize them for verification. Especially with the constantly growing amount of regulatory documents and their frequent updates, it can be preferable to provide an approach that enables the verification of processes with requirements in natural language text instead of formalized requirements. To this end, this paper presents an approach that copes with the verification of resource compliance requirements, e.g., which resource shall perform which activity, in natural language over event logs. The approach relies on a comprehensive literature analysis to identify resource compliance patterns. It then contrasts these patterns with resource patterns reflecting the process perspective, while considering the natural language perspective. We combine the state-of-the-art GPT-4 technology for pre-processing the natural language text with a customized compliance verification component to identify and verify resource compliance requirements. Thereby, the approach distinguishes different resource patterns including multiple organizational perspectives. The approach is evaluated based on a set of well-established process descriptions and synthesized event logs generated by a process execution engine as well as the BPIC 2020 dataset.

Keywords: Compliance Requirements Verification · Resource Mining · Natural Language Text · Process Descriptions · Event Logs.

1 Introduction

Business process compliance is the task of ensuring that processes obey the rules, guidelines, and constraints imposed on them. Those compliance requirements are typically outlined in extensive regulatory documents such as legislative texts or ISO norms [12]. Compliance requirements need to be verified, i.e., they are

checked against the actual execution of a process captured by an event log. In order to enable verification, the compliance requirements are typically formalized using, e.g., linear temporal logic [2,15,25]. However, as regulatory documents change frequently, the compliance requirements have to be re-formalized equally frequently. Therefore, it can be desirable to enable direct compliance verification between requirements provided in natural language and event logs.

Business process compliance refers to multiple perspectives beyond control flow, i.e., time, resources, and data [15]. In recent work, we focus on verifying quantitative temporal compliance requirements over event logs [4]. In this paper, we consider the resource perspective which captures legally relevant concepts such as the segregation and binding of duties [24]. In general, process activities are carried out by resources, which can be classified into categories human and non-human [21]. Human resources refer to individuals involved in the process, while non-human resources are typically machines, robots, or computer-based systems. The presented approach can extract both, compliance requirements referring to human and non-human resources, but the main focus will be on human resources. In the case of human resources, in this paper four different types of resources are distinguished following [21]: *organizations*, such as company X, represent a larger grouping of resources. *Organizational units* such as departments or teams are subgroups within an organization and are responsible for specific activities. *Roles* are used to define the specific responsibilities and tasks assigned to different resources within an organization. The role of a software developer, for example, contains responsibilities such as developing and maintaining software applications. Finally, *users* are specific individuals. In the case of human resources, users can be identified by their name or an ID, while non-human resources may be identified by their unique identifiers or serial numbers.

The identification of resources paired with activities from natural language text has been addressed by, e.g., [5,11,20]. Yet, a) the organizational structure reflected by the resources, b) the different compliance requirement patterns associated with each resource, and c) the compliance verification of natural language texts over event logs have not explicitly been considered. Moreover, we see the extraction of resource activity pairs as a pre-processing step, and rely for this on the model GPT-4 [19]. Therefore, the main contribution is not how to identify and extract those pairs from natural language text, but how to use the output w.r.t. compliance verification over event logs. In order to address this question, we analyze in Sect. 2 i) the process perspective through workflow resource patterns [21], mining organizational structures from event logs [10] and the eXtensible event stream (XES) standard [1] , and ii) the compliance perspective by identifying resource compliance requirements [3,15,23,24]. We establish a mapping between both perspectives and extract a summary of resource compliance patterns that are feasible to detect from a natural language processing perspective. Moreover, we identify which assumptions an event log needs to fulfill to enable compliance verification. Based on those findings, in Sect. 3 we design an approach consisting of five steps divided into a pre-processing component and the actual compliance verification component. The approach is evaluated in

Sect. 4 followed by a discussion of evaluation results and limitations in Sect. 5. Section 6 presents related work and Sect. 7 concludes the paper with a summary and outlook on future work.

2 Resource Compliance Requirements Pattern Elicitation

In the following, we provide an analysis of which resource compliance requirements patterns exist and how they are reflected by the process and event log perspective. This analysis constitutes the fundamentals for the resource compliance requirements verification approach, presented in Sect. 3. First, Sect. 2.1 summarizes how resources are represented from the process and event log perspective by reviewing the literature on workflow resource patterns [21], mining organizational structures from event logs [10] and the XES standard [1]. Second, in Sect. 2.2, we analyze literature on resource compliance requirements [3,15,22,23,24], and related work. This body of literature was selected based on a literature search conducted on DBLP[3] using keywords "resource" or "organizational" or "role" and "mining", "organization" or "resource" and "compliance" or "requirements", and "resource-aware process verification". Table 1 presents a mapping between the resource patterns considered in [21] and those identified papers. A detailed analysis is provided in a spreadsheet, which can be accessed at https://www.cs.cit.tum.de/bpm/data/.

Pattern	Description	Process		Compliance					
		[21]	[10]	[23]	[15]	[24]	[3]	[22]	[26]
1. Performed by	A_1 must be performed by resource Re	✓	✓	✓	✓	✓	✓	✓	✓
2. Not Performed by	A_1 can not be performed by resource Re	✗	✓	✗	✗	✗	✓	✗	✗
3. Dynamic SoD (4-Eyes-Principle)	A_1 and A_2 must be performed by different resources, independently of roles.	✓	✓	✓	✓	✓	✓	✓	✓
4. Static SoD	A_1 and A_2 must be performed by different resources with different roles	✓	✓	✓	✓	✓	✓	✓	✓
5. Multi-segregated	A set of activities (A_1, A_2, A_3, ..., A_n) must be performed by (m) different resources	✗	✗	✗	✗	✓	✗	✗	✓
6. Dynamic Bonded	A_1 and A_2 must be performed by different resources with the same role	✓	✓	✓	✓	✓	✗	✗	✗
7. Static Bonded	A_1 and A_2 must be performed by the same resource with the same role	✓	✓	✓	✓	✓	✗	✓	✗
8. Multi-bonded	A set of activities (A_1, A_2, A_3, ..., A_n) must be performed by the same resource with the same role	✓	✗	✓	✓	✓	✗	✓	✗
9. Automatic	A_1 is automatically executed	✓	✗	✗	✗	✗	✗	✗	✗

Table 1. Mapping of Resource Patterns; ✓ = mentioned; ✗ = not mentioned

2.1 Process Perspective

In [21], 43 patterns describing the distribution and execution of activities in workflow systems are proposed. These patterns are classified into five categories:

[3] https://dblp.org/, last access: 2023-06-11

Creation patterns limit how activities are executed, determine which resources can perform an activity, and match activities with capable resources. *Push* and *Pull* are distribution patterns where activities are allocated by a central authority or chosen by resources respectively. *Detour* and *Visibility* allow resources to modify ongoing activities or view available ones, while *Multi-Resource* sets limits on how many activities a resource can perform at the same time. These patterns ensure efficient and effective execution while meeting business requirements. Our approach takes into account the *Creation* patterns, assuming that a resource will be able to perform the same activities throughout the entire process. In all the *Creation* patterns, the requirements only involve the relationship between a resource and the activities' performance. To represent these requirements, we introduce the concept of Resource-Activity Requirement (R-AR) which capture the patterns presented in Tab. 1. Algorithm 1 determines how R-ARs are built.

The sub-patterns related to *Allocation-based Creation* are omitted from Tab. 1. These sub-patterns, encompassing diverse allocation strategies like assigning tasks based on hierarchy or deferring assignments to future activities, are viewed as ancillary data rather than essential for forming resource-activity pairs. For example, we focus on whether *Resource R* executed *Activity A*, rather than whether *Resource R* was the original assignee. This approach aids to avoid unnecessary complexity and potential biases, thereby centering our research on fundamental resource-activity pairings.

Of all the literature analyzed, [10] adheres most closely to the resource patterns presented in [21]. Nonetheless, we suggest further research to extract control flow aspects from process descriptions, which could extend our approach and incorporate cross-perspective patterns.

As we aim at compliance verification over event logs we consider the XES standard [1], which contains an organizational extension describing human actors. Therein, three elements are distinguished, first resource which contains the "name, or identifier, of the resource having triggered the event."[1], a role which reflects the "role of the resource having triggered the event, within the organizational structure."[1] and a group that represents the "group within the organizational structure, of which the resource having triggered the event is a member."[1] Considering the terminology we chose for this paper, a resource refers to a rather generic term compared to the term resource as used within the XES standard. A user corresponds to resource, group corresponds to organizational unit and we additionally add organization in order to include the perspective that multiple organizations and interactions between them could be described in one natural language text. Overall, the event log must contain basic elements as specified by the XES standard, including trace names, event labels, event IDs and fields denoting the resource type as well as a field detailing the organization. The event log may not contain errors like duplicate events.

2.2 Compliance Requirements Perspective

The first aspect that can be observed in Tab. 1, is that the papers that focus on studying resource compliance verification do not consider the *Creation* au-

tomatic activity execution sub-pattern, even though it involves the system as a resource. We decided to maintain the automatic execution sub-pattern [21] because there may be cases where the system is the only one permitted to execute an activity, and no one else can. In contrast, [3,10] introduced the *Not Performed by* pattern, which is not covered in [21]. Similarly, the *Multi-segregated* pattern is only discussed in [24,26]. As there is little consensus among the papers regarding the definition of resource (e.g. using other terms like *agent*, or considering only *users* and *roles*), we decided to define the resource concept and its types in Sect. 1. In [23], the authors distinguish between resources and agents and consider both resource-aware and data-aware compliance requirements. This paper presents examples of cross-perspective patterns, such as *If attribute X has value v then resource Re must execute the activity* pattern. From a resource perspective, both [23] and [15] cover the same patterns. However, [23] discusses authorization, whereas [15] addresses the same issues but refers to *users* and *roles*. In [24], the same resource patterns as in [23,15] are addressed, with the addition of the *Multi-segregated* pattern. Both [3,22] focus on less than 10 specific resource examples, which overlap with each other and do not introduce any new patterns. Additionally, [26] focuses on the *Segregation of Duties* (SoD) when studying resource patterns, as their primary objective is task-based authorization constraints. In this paper, they include aspects that previous papers do not handle but we are, such as *The same activity can only be performed twice by the same resource, and A_1 and A_2 must be performed by different resources.*

3 R-AR Verification Approach

The resource compliance requirements verification approach is presented in Fig. 1, and illustrated based on a running example depicted in Fig. 2, and Fig. 3. The approach consists of a pre-processing component (cf. Sect. 3.1) and a compliance verification component (cf. Sect. 3.2). It takes as input a natural language text, e.g., a process description, and an event log, which must fulfill the assumptions as detailed in Sect. 2.1. Consistency in naming resources and activities within the process description and within the log is assumed, but synonyms of both, resources and activities, across the description and log are handled by the approach. Note that the textual document is always considered as the ground truth, i.e., compliance of a given event log is verified against the textual document.

The approach consists of five steps and allows for user intervention for compensating errors after steps 1, 2, and 3. These step's results are saved for possible necessity, as the approach's outcome hinges on their quality. In general, the approach is independent of the domain but with all intermediate results available, domain knowledge can be easily integrated, e.g., in order to tailor the GPT-4 prompt towards a specific dataset. Moreover, several parameters must be set by the user such as thresholds for similarity mappings. Further details are provided in the explanations of the single steps.

The output of the compliance verification component consists of two files. The first one, generated after step 3, details on activity matching results for trans-

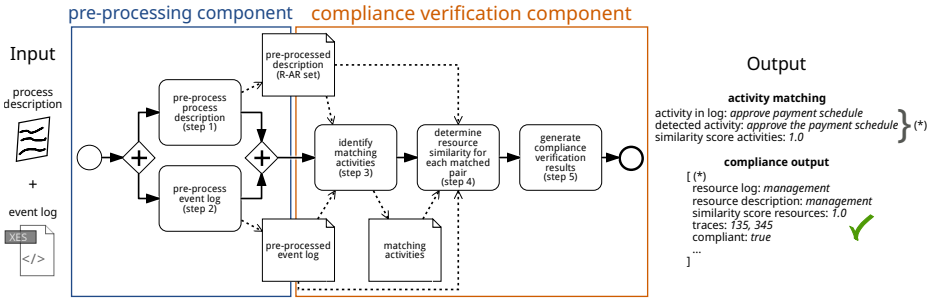

Fig. 1. Overview of Resource Compliance Requirements Verification Approach

parency allowing for tracing whether errors occurring later on were caused by the previous activity matching. The second file contains the compliance verification results. Both files are ordered by unique event log pairs, i.e., resource-activity pairs, for easier comprehension.

3.1 Pre-processing Component

The pre-processing component handles a process description in natural language text and in parallel the event log data. The intermediate pre-processed outputs serve as input for the second component, the compliance verification process.

Step 1 – Pre-process Process Description. Consider the running example, Fig. 2, depicting a process description along with examples of resource compliance patterns. The intended outcome of the running example's pre-processing is a collection of Resource-Activity Requirements (R-ARs). Each R-AR includes a resource (marked in bold), and activity (underlined), along with the necessary metadata (cf., output of Algo. 1) used to identify the appropriate resource pattern for each R-AR.

1. Whenever **Elite Holdings** receives a customer request, it demands a solvency check from **Miracle Credit**.
2. At **Miracle Credit** exactly **two clerks** perform a solvency check.
3. **Miracle Credit** hands back the results of the solvency check to **Elite Holdings**.
4. If the solvency check results is negative, a **clerk from the customer advisory** informs the customer and deletes the customer's request.
5. If the solvency check result is positive, **Anna** or **Hans,** bot not both, develop a payment schedule.
6. Afterward, the schedule is sent to the **manager**.
7. Both **he** and **another clerk from the management** must approve the payment schedule.
8. Approve payment schedule may never be executed by **Anna** or **Hans**
9. If the payment schedule has been approved, an email is sent to the customer **automatically**, otherwise, the **customer advisory** calls the customer to suggest an alternative.
10. In both cases, the request must be closed.

Fig. 2. Running Example – Natural Language Text

Examples of these patterns include *Performed by*, as in ④ *a clerk from customer advisory informs the customer*, and *Not Performed by*, which identifies resources that are not allowed to execute an activity, such as ⑧ *approve payment schedule may never be executed by Anna or Hans*. Additionally, more complex requirements involving multiple resources may be present, such as a *Multi-segregated*, where ⑦ *he and another clerk from the management must approve the schedule*. In this case while pre-processing the description we need to consider anaphora resolution in order to know which resource is meant by *he*. Note that resources are not limited to humans; automated tasks can also occur during process execution, such as ⑨ *sending an email to the customer* (*Automatic Pattern*). Moreover, the running example shows the different granularity levels of a resource: organization (*Elite Holdings, Miracle Credit*), organizational units (*customer advisory, management*), roles (*clerk, manager*), and particular users (*Anna, Hans, system*).

The extraction of R-ARs from process descriptions is carried out using the GPT-4 model from OpenAI [19]. The prompt schema is showcased in Algo 1. To ensure reproducibility, the original prompt is also provided as input for the evaluation of our implementation. The prompt provides fundamental information such as the meaning of a resource and an organization, along with examples for each. These examples are not taken from our dataset. Details about the required format, necessary parameters, and the type of each parameter are also included in the prompt. The delivery of this information is thoroughly explained within the prompt. For example, if the activity concerns two resources, the aim is to treat each as a separate resource instead of a collective group. In cases of lengthy process descriptions, even if there is not a specific question about the control flow, the model takes the initiative to propose a control flow concept, which could be exploited if the control flow is also analyzed in future work.

Algorithm 1: Process Description Pre-processing Prompt Schema

```
Input: Process Description
Output: Set of R-ARs with fields: "role", "user", "org_unit", "organization", "activity", "inclusion",
        "exclusion", "min", "max", "equals", "anaphora", "is_performed"
Function SetFields():
    Initialize: "role", "user", "org_unit", "organization", "activity"
    Set "inclusion" and "exclusion"
      // capture relations between R-ARs (e.g., conflicts between R-ARs)
    Set "min", "max", "equals"  //limit the number of resources performing an activity
    Set "anaphora"  //indicate the original resource which references the pronoun
    if activity is not performed then
      |  Set "is_performed" = false
    else
      |  Set "is_performed" = true
    end
```

Step 2 – Pre-process Event Log. Figure 3 illustrates an excerpt of an event log reflecting the process description in Fig. 2. For brevity, the trace ID in Fig. 3 is omitted and it is assumed that all depicted events belong to the same trace and that the trace is complete. The events that comply with the process

description are E1, E2 and E7 (Fig. 2 ①, ② and ⑨). Events E3, E4, E5 and E6 (Fig. 2 ③, ⑤, ⑦) do not comply. In event E3, the organization should be referred to as *Miracle Credit* not *Elite Holdings*. Events E4 and E5 must not occur simultaneously in one trace. Additionally, the activity in event E6 should be carried out by employees holding the role of manager, as well as by a specific employee in the clerk role, rather than being exclusively assigned to one unique employee with the clerk role.

	Event ID	concept:name	org:resource	org:role	org:unit	organization
①	E1	demand solvency check	unknown	unknown	unknown	Elite Holdings
②	E2	perform solvency check	Peter	clerk	unknown	Miracle Credit
③	E3	hand back results	unknown	unknown	unknown	Elite Holdings
⑤	E4	develop payment schedule	Anna	unknown	unknown	Elite Holdings
	E5	develop payment schedule	Hans	unknown	unknown	Elite Holdings
⑦	E6	approve payment schedule	unknown	clerk	management	Elite Holdings
⑨	E7	send email to customer	system	unknown	unknown	Elite Holdings
	...					

Fig. 3. Running Example – Event Log Trace containing Violations

The event log, i.e., also this trace, serves as input for the event log pre-processing step. This step constructs an object-oriented representation of the event log's structure, which includes an attribute, event, trace, and event log class. Each class contains methods to extract the most crucial information from the event log. To generate a pre-processed event log output for persistent storage, an event log is parsed into a data frame using PM4Py [7]. The data frame is then converted into an event log object from the custom-created class. The event log class features a method generating an output file containing every distinct event in the log. Thereby, a distinct event is defined as a unique resource-activity pair, where a resource is a combination of a user, role, organizational unit, and organization. An example of a pre-processed event is *Event = {"activity": "perform solvency check", "user": "Peter", "role": "clerk", "org_unit": "unknown", "organization": "Miracle Credit"}*

3.2 Compliance Verification Component

After pre-processing the process description and the event log, the actual compliance verification can be carried out. This requires a mapping between the R-ARs from process descriptions and event logs. The component unfolds in the following three steps.

Step 3 – Identify Matching Activities. In this step of the overall approach, activities in the event log output file are compared for similarity with those in the description output file. To make the approach more resilient in real-world settings, we accept variations in how activities are phrased between the event log and the process description, as we only expect terminology consistency within

each of these individually. All the activities in the pre-processed log are compared with all activities in the description output, identifying matches based on high similarity scores. Each match's score is then evaluated against a predefined threshold to decide whether to accept it. If the score falls short of this threshold, we denote the log activity as unmatched. This safeguards against finding matched resource-activity pairs in subsequent steps. The threshold and similarity score for matching pairs range from zero to one. If activities are few and similar, scores are near one; diverse activities yield scores near zero. An expert, familiar with the data, should set a threshold to eliminate misclassified matches. This threshold is adjustable to accommodate various activities and the selected similarity measure, impacting synonym levels in pre-processed files. Three similarity measures - TF-IDF/linear kernel[4], BERT/cosine similarity[5], and spaCy similarity[6] evaluate activity likenesses in the log and description. The user initially selects one, alongside the threshold for valid activity matches.

The output of this step contains a measurement-type section containing the compliance verification input information, e.g., the set threshold for activity similarity matching and the activity matching result information. *Activity Matching Output: Measure Types: {"similarity measure": "TF-IDF", "threshold": 0.65, ...}, "activity_matching_output": [{ "Activity Log": "accept order", "Detected Activity": "accept the order", "Similarity Score": 1.0}...]*

Step 4 – Determine Resource Similarity. In the previous step, each activity from the pre-processed event log was matched with an activity from the pre-processed process description. In this step, we evaluate whether the resource performing each activity for a specific event in the event log is similar to its counterpart in the process description. For that, the same similarity measures are used as within step 3. This semantic similarity needs to be executed since resources can be subject to different naming conventions, and wording. The threshold used for comparing resources can differ from the one used for activities, reflecting the importance of knowledge of the mentioned resource types in the text and event log. Furthermore different types of resources to be checked for similarity to ensure event log compliance can be selected. These resources can be organizational units or users with specific roles, depending on what is stored in the log and the information granularity provided in the text document. If the chosen resource structure type is not specified for a particular event in the log, a mechanism is in place to automatically build the resource to be checked using the available resource values in the log, as long as they are defined. For instance, if the initial choice was to check the organizational unit as the resource structure but the value is undefined for a specific event in the log, the mechanism will instead construct the resource based on the user, role and organization information.

Step 5 – Generate Compliance Verification Results. The last step creates, based on the outcome of the resource checks, the compliance output file. Hence, the implemented approach verifies the minimum criteria necessary to classify

[4] https://scikit-learn.org/stable/install.html, last access: 2023-06-11
[5] https://www.sbert.net, last access: 2023-06-11
[6] https://spacy.io/usage/linguistic-features, last access: 2023-06-11

a resource-activity pair as compliant or non-compliant. Two main distinctions are made for these checks. When a resource, whether human or non-human, is expected to perform an activity (Patterns 1 and 9), the resulting similarity score from step 4 must exceed a predefined threshold. On the other hand, if a specific resource should not perform an activity (Pattern 2), the similarity score should be lower than the threshold. The resulting output file maintains the same granularity, naming convention, and structure for both approaches. The order of events in the output file corresponds to the pre-processed event log file, as the compliance of the process is verified for each distinct event by comparing it to the description. Similar to activity matching, the output includes a section for measurement types and appears as follows: *Compliance Matching Output: Measure Types: ..., default compliance check output: ["Matched Activity": "accept the order", "Resource Log": "sales department", "Resource Description": "member of the sales department", "Similarity Score of Matched Resources": 0.65, "Corresponding Traces": ..., "Compliant": true, "Non-Compliant Reason": " ...]*. Furthermore, based on the information given in all resulting files from the overall process, users can manually verify if patterns incorporating multiple resource-activity pairs at once (Patterns 3-8) are also compliant, if necessary.

4 Evaluation

The approach has been implemented as a prototype in Python 3 and can be accessed publicly at the following location: https://www.cs.cit.tum.de/bpm/software/. All input, intermediate files, like JSON files from GPT-4 prompt execution, and pre-processed event logs, and output files are available, as well, via the above link. In Sect. 4.1, details on the datasets used in the evaluation are provided while the evaluation results for the pre-processing, as well as compliance verification component, are described separately in Sect. 4.2. All the synthesized and modified event logs are available at https://www.cs.cit.tum.de/bpm/data/.

4.1 Dataset Preparation

The evaluation features synthetic as well as real-world datasets. In the following, we describe how the synthetic datasets were generated and how the real-world dataset was prepared.

Synthetic datasets. First of all, we take into account the process description of the *Running Example* (RE), which was initially introduced in Sect. 3. This example was meticulously designed to allow the evaluation of an extensive set of patterns, which are comprehensively detailed in Tab. 1. Additionally, the PET dataset, a well-known collection of 45 process descriptions [6], with a total number of resource activity pairs of 449, was included in the analysis. On average each process description of the PET dataset contains 10 resource activity pairs. From this extensive dataset, two specific descriptions were singled out for consideration: *Bicycle Manufacturing* (BM) and *Schedule Meetings* (SM). On average, each of these two descriptions contain 10.50 resource activity pairs. For each of

these three process descriptions a model and event logs were generated using the Cloud Process Execution Engine [7] (CPEE) [16]. These event logs did initially not contain any resource violations. To test the compliance verification component of our approach, resource compliance violations were introduced into the logs. This was done by randomly selecting resource activity pairs and modifying the resource executing the activity. This process of modeling, log generation, and log alteration was overseen by one of our authors who was not involved in the technical implementation process.

Real-world dataset. In addition to the synthetic datasets, we had a look at real-world event logs used within the Business Process Intelligence Challenges (BPIC). Among those, we identified the BPIC 2020 to be suitable since it contains resources not only in the form of IDs but verbal (e.g., *budget owner*) and comes at the same time with a detailed textual process description[8]. This dataset, collected between 2017 and 2019, comprises a total of 270,216 events recorded across five logs. In BPIC, resources are classified as *STAFF MEMBER* or *SYSTEM*. The first type of resource can have a role while the system not. The column values for event and case *ID* were adjusted to match the terminology used in the implementation for pre-processing the event log. The names of the roles (e.g., *DIRECTOR*) were removed from the original label of the activity (e.g. *PERMIT REJECTED by DIRECTOR*). Offered as supplementary material, the script specifically designed for this task could be employed as a blueprint, enabling the adaptation of an event log from any domain, with a different data structure, to suit our approach.

4.2 Results

Owing to the significant impact of the pre-processing quality on the compliance verification results, we first provide a succinct overview of the results for the pre-processing component before diving deeper into the evaluation of the compliance verification component.

Pre-processing Component. To assess the performance of the pre-processing of process descriptions, a gold standard file for each process description is generated containing the desired set of R-AR results. We utilize this gold standard to compare the performance of GPT-3.5 and GPT-4 in retrieving the set of R-ARs. To accomplish this, the GPT models receive as input each process description together with the prompt, which was built following the steps outlined in Algo. 1. All intermediate files generated during this evaluation process are saved and stored in our repository for future reference and analysis. Table 2 displays the precision and recall values for the results obtained from both the GPT-3.5 and GPT-4 model.

[7] https://cpee.org, last access: 2023-06-11

[8] https://data.4tu.nl/collections/BPI_Challenge_2020/5065541 last access: 2023-06-11

Dataset	Precision		Recall	
	GPT-3.5	GPT-4	GPT-3.5	GPT-4
RE	0.92	1.00	0.92	1.00
BM	0.62	0.90	1.00	1.00
SM	0.50	0.89	0.17	1.00
BPIC	0.76	1.00	1.00	1.00

Table 2. Evaluation Results Pre-processing Component

A R-AR is deemed successfully detected if it contains both the activity and resource, representing a pair in the process description, and matches any of the patterns outlined in Tab. 1. By looking at Tab. 2, on average, the precision scores improved by 40% when using GPT-4 compared to its predecessor. Additionally, the recall demonstrated a substantial increase of 135%. Most GPT-3.5 errors stem from anaphoras, scattered information, passive voice, and activity boundary detection issues.

In [5], a GPT-3 model was utilized. However, this model had limitations in identifying the resource responsible for an activity if the word *perform* was not explicitly mentioned in the sentence. In contrast, our pre-processing approach was designed to handle more complex cases and successfully identified resources in all instances, even those involving anaphora, embedded conditions, and other related factors such as excluding and including activities. Misclassifications in the results produced by GPT-4 often stem from the model trying to assume too much information, which can result in false positives when the model tries to infer activities that were not clearly specified. As outlined in [18], they suggest improving the quality of the PET dataset by employing data augmentation methods. Another issue with models like GPT-4, such as generating labels with new words or synonyms, can be mitigated by narrowing the prompt, as indicated in [13]. This approach not only helps avoid these issues but also leads to more faithful and reasonable texts, reducing the hallucination, i.e., AI creating information without input basis, that occurs in natural language generation.

Compliance Verification Component. The evaluation of the compliance verification component is depicted in Tab. 3. It depicts the intermediate results for the matched activities (step 3), the extraction of resource similarity (step 4), and the final compliance verification outcomes (step 5). For the synthetic datasets, RE, BM, and SM, which had less contextual detail, TF-IDF was applied, and activity (step 3) and resource-activity (step 4) thresholds were between 0.60 and 0.80. For the compliance verification of BPIC, we used the BERT model to account for varying naming conventions and synonyms. To manage BERT's tendency to assign high similarity scores to dissimilar pairs, we set activity and resource-activity thresholds at 0.8 and 0.9. This approach maintained accuracy by ensuring context-specific matches, despite the presence of synonyms.

Upon examining the output of step 1 of BPIC, 17 unique resource activity pairs were identified. However, step 2 revealed a larger number, presenting 55 unique resource activity pairs. Once we processed the results of the compliance verification component, we deduced that only 7 pairs from the original event log were accurately represented in the process description. These pairs included (*declaration final approved, director*), (*declaration approved, administration*), (*declaration rejected, employee*), (*declaration submitted, employee*), (*request payment, automatic*), (*permit rejected, employee*), and (*request for payment rejected, em-*

ployee). The BPIC process description lacks explicit information about the remaining 48 pairs, preventing further interpretations without making extensive assumptions. This challenge that the BPIC dataset presents, is partially solved by the use of thresholds, and it is not present in the synthetic dataset because the total number of unique pairs in the log is very close to the total number of resource-activity pairs identify in the process description.

Considering the results for step 3 in Tab. 4, it is notable to see that the approach was able to match in all the cases the activities presented in the process description. The small proportion of mismatches (c.f., precision of BM or SM) from the activities presented in the event log was due to the presence of events not described in the process description. The results of steps 4 and 5 are evidence of the impact that has the good handling of the granularity of a resource. The RE contained a wider variability of granularity which ended up being more challenging and had a direct impact on the extraction of non-compliant traces.

		RE	BM	SM	BPIC
Step 3	Precision	0.94	0.87	0.81	1.0
	Recall	1.0	1.0	1.0	1.0
Step 4 & 5	Precision	0.64	1.0	1.0	0.71
	Recall	1.0	0.92	1.0	1.0

Table 3. Evaluation Results of Steps 3-5

In RE, BM, and SM, the majority of non-compliant traces were due to a conflict with another resource with different granularity, as indicated in the process description, performing an activity (e.g., *Elite Holdings* vs *Hans from Elite Holdings*). While in the BPIC dataset, non-compliant traces were primarily characterized by three factors. Firstly, traces that contained *missing* as a resource keyword were flagged. Secondly, traces that involved staff members who, by hierarchical implication, were permitted to perform the activities of their subordinates were noted. Lastly, instances, where system automation carried out activities intended for human staff members, were also marked as non-compliant. Future enhancements should thus address these distinct issues accordingly. The ensuing section will discuss the limitations of the current study and potential avenues for further research.

5 Discussion and Limitations

In order for the presented approach to achieve optimal results, it is essential that the pre-processing of the natural language text delivers all necessary information, such as activities, the different levels of granularity of a resource, and other fields, as shown in Algo. 1. We utilize state-of-the-art GPT-4 technology to accomplish this as it is undoubtedly powerful and provides the means to boost the performance of our approach. However, there are certain drawbacks to consider which also led to the decision to develop a customized compliance verification component without GPT-4.

Reproducibility. GPT-4 is a black box model, which makes ensuring reproducibility challenging due to its dependence on finely-tuned prompts. To address

this issue, we provide both the prompt we used and the output from GPT-4, which can then be incorporated into the compliance verification component. If we had used GPT-4 also for compliance verification we see a further challenge in describing the pattern-based check accurately in a prompt for GPT-4. Crafting a prompt that consistently produces reliable results for compliance verification might require a significant amount of effort and expertise.

Reliability, Explainability, and Transparency. Furthermore, GPT-4 is a third-party service and may not always be accessible when needed (e.g., having a cap of 25 messages every 3 hours). During the evaluation, we also recognized a certain instability in the results it produced, e.g., slower service, loss of history. GPT-4, being a large transformer model, is considered a black box meaning its results might be difficult to understand. In particular, the incorrect classification of events into compliant or non-compliant in the compliance verification would provide a lack of transparency without being able to explain the process. Our aim is to provide a transparent step-by-step resolution of compliance verification results, which is not easily achievable with GPT-4.

Technical Feasibility and Costs. As users must pay for each executed prompt, it might become expensive and also impractical to process a large event log containing thousands of events which is another reason we want to keep the usage of GPT-4 to a minimum. The final goal is to develop a fully automated compliance verification approach, however, this is difficult to offer when relying on a commercial product.

Suggestions for Improvement. One option for a customized pre-processing component could be built based on existing work, e.g., [5,11,20]. The task of identifying resources could be, e.g., follow a rule-based approach incorporating a custom-trained named entity recognition (NER) model. A second option is to explore OpenAssistant [14], a lightweight open-source project to collaborate on large language models. Further improvements include the implementation of a hierarchy compliance resolution verification system that evaluates compliance for role and department hierarchies. For instance, this would enable a manager to carry out tasks usually done by a junior developer. The method will also evaluate varying organizational structures like user-role and role-department, merging the best combinations for each scenario. Additionally, it could investigate handling cases with multiple process logs but only one process description, looking into how these logs can be combined and compliance ensured.

6 Related Work

Related work on how resources are handled from a process, compliance and event log perspective has been discussed in Sect. 2. In order to provide a holistic view of the topic, we outline additional related work in the following.

For the pre-processing component, literature on the identification and extraction of resource information from natural language text constitutes a further line of related work. This task has been addressed by approaches that aim at process

model generation from natural language text, like, e.g., [11]. Other approaches employ semantic role labelling in order to extract resources [20] or pre-trained language models and in-context learning in order to extract business process entities and their relations from natural language texts [5]. The latter makes use of GPT-3 [8]. However, when using GPT-3.5 in the pre-processing component, we could not come up with satisfying results. Only the latest released GPT-4 model [19] was capable of delivering the necessary quality for the pre-processing component. Another line of research is focusing on extracting access control policies from natural language text, e.g., [17]. They also make use of semantic role labelling like [20] but as demonstrated in Sect. 2, resource compliance patterns are more diverse. Moreover, none of the mentioned approaches has envisioned compliance verification over event logs as it is the aim of this paper.

In order to enable compliance verification, by now, compliance requirements need to be formalized as, e.g., LTL formulas manually. Recent efforts have focused on automatically extracting LTL formulas from natural language texts, cf., e.g., [9] for a comprehensive state-of-the-art analysis. However, according to this survey "a general enough solution, that is capable of translating free, natural English texts into unbounded, general LTL formulas is still missing."[9].

7 Conclusion and Future Work

In this paper, an approach for resource compliance requirements verification over event logs has been presented. Compared to existing work, resource compliance requirements do not need to be formalized in, e.g., LTL formulas, but can be kept as natural language text. The approach consists of a pre-processing component that makes, i.a., use of recent advances in deep learning, in particular GPT-4. The compliance verification component constitutes the main contribution of this paper and encounters several steps to achieve resource compliance verification. Each step of the approach was evaluated quantitatively using precision and recall on multiple synthetic as well as a real-world dataset, the BPIC 2020 dataset. The evaluation results are promising and provide clear pointers for future work. In particular, we plan to implement a customized pre-processing component for the requirements extraction from natural language text, e.g., using OpenAssistant [14], and compare it to the current solution which uses GPT-4. This allows us to cope with limitations arising due to possible downtimes of GPT-4, the necessity to have access to GPT-4, and investing time to fine-tune the employed prompt. Moreover, we plan to incorporate further perspectives like control flow, data, and time to come up with a holistic compliance verification approach.

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. 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
2. 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
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. Decis. Support Syst. **50**(3), 636–647 (2011). https://doi.org/10.1016/j.dss.2010.08.014
4. Barrientos, M., Winter, K., Mangler, J., Rinderle-Ma, S.: Verification of quantitative temporal compliance requirements in process descriptions over event logs. In: Advanced Information Systems Engineering, 35th International Conference, CAiSE. Springer (2023). https://doi.org/10.1007/978-3-031-34560-9_25
5. 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
6. Bellan, P., Ghidini, C., Dragoni, M., Ponzetto, S.P., van der Aa, H.: Process extraction from natural language text: the PET dataset and annotation guidelines. In: Proceedings of the Sixth Workshop on Natural Language for Artificial Intelligence (NL4AI 2022). vol. 3287, pp. 177–191. CEUR-WS.org (2022), http://ceur-ws.org/Vol-3287/paper18.pdf
7. Berti, A., van Zelst, S.J., van der Aalst, W.M.P.: Process mining for python (pm4py): Bridging the gap between process- and data science. CoRR **abs/1905.06169** (2019), http://arxiv.org/abs/1905.06169
8. Brown, T.B., et al.: Language models are few-shot learners. In: Annual Conference on Neural Information Processing Systems (2020), https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html
9. Brunello, A., Montanari, A., Reynolds, M.: Synthesis of LTL formulas from natural language texts: State of the art and research directions. In: 26th International Symposium on Temporal Representation and Reasoning, TIME. LIPIcs, vol. 147, pp. 17:1–17:19 (2019). https://doi.org/10.4230/LIPIcs.TIME.2019.17
10. Cabanillas, C., Ackermann, L., Schönig, S., Sturm, C., Mendling, J.: The ralph miner for automated discovery and verification of resource-aware process models. Softw. Syst. Model. **19**(6), 1415–1441 (2020). https://doi.org/10.1007/s10270-020-00820-7
11. 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
12. Hashmi, M., Governatori, G., Lam, H., Wynn, M.T.: Are we done with business process compliance: state of the art and challenges ahead. Knowl. Inf. Syst. **57**(1), 79–133 (2018). https://doi.org/10.1007/s10115-017-1142-1
13. Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y., Madotto, A., Fung, P.: Survey of hallucination in natural language generation. ACM Comput. Surv. **55**(12), 248:1–248:38 (2023). https://doi.org/10.1145/3571730
14. Köpf, A., Kilcher, Y., von Rütte, D., Anagnostidis, S., Tam, Z., Stevens, K., Barhoum, A., Duc, N.M., Stanley, O., Nagyfi, R., ES, S., Suri, S., Glushkov,

D., Dantuluri, A., Maguire, A., Schuhmann, C., Nguyen, H., Mattick, A.: Openassistant conversations - democratizing large language model alignment. CoRR **abs/2304.07327** (2023). https://doi.org/10.48550/arXiv.2304.07327
15. 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
16. Mangler, J., Rinderle-Ma, S.: Cloud process execution engine: Architecture and interfaces (2022). https://doi.org/10.48550/ARXIV.2208.12214
17. Narouei, M., Takabi, H., Nielsen, R.: Automatic extraction of access control policies from natural language documents. IEEE Trans. Dependable Secur. Comput. **17**(3), 506–517 (2020). https://doi.org/10.1109/TDSC.2018.2818708
18. Neuberger, J., Ackermann, L., Jablonski, S.: Beyond rule-based named entity recognition and relation extraction for process model generation from natural language text. CoRR **abs/2305.03960** (2023). https://doi.org/10.48550/arXiv.2305.03960
19. OpenAI: Gpt-4 technical report (2023)
20. 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
21. Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H.M., Edmond, D.: Workflow resource patterns: Identification, representation and tool support. In: Advanced Information Systems Engineering, 17th International Conference, CAiSE. vol. 3520, pp. 216–232. Springer (2005). https://doi.org/10.1007/11431855_16
22. Semmelrodt, F., Knuplesch, D., Reichert, M.: Modeling the resource perspective of business process compliance rules with the extended compliance rule graph. In: Enterprise, Business-Process and Information Systems Modeling - 15th International Conference, BPMDS 2014, 19th International Conference, EMMSAD. Lecture Notes in Business Information Processing, vol. 175, pp. 48–63. Springer (2014). https://doi.org/10.1007/978-3-662-43745-2_4, h
23. Taghiabadi, E.R., Gromov, V., Fahland, D., van der Aalst, W.M.P.: Compliance checking of data-aware and resource-aware compliance requirements. In: On the Move to Meaningful Internet Systems: OTM 2014 Conferences - Confederated International Conferences: CoopIS, and ODBASE. vol. 8841, pp. 237–257. Springer (2014). https://doi.org/10.1007/978-3-662-45563-0_14
24. Türetken, O., Elgammal, A., van den Heuvel, W., Papazoglou, M.P.: Capturing compliance requirements: A pattern-based approach. IEEE Softw. **29**(3), 28–36 (2012). https://doi.org/10.1109/MS.2012.45
25. 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
26. Wolter, C., Schaad, A.: Modeling of task-based authorization constraints in BPMN. In: Business Process Management, 5th International Conference, BPM. Lecture Notes in Computer Science, vol. 4714, pp. 64–79. Springer (2007). https://doi.org/10.1007/978-3-540-75183-0_5